trackme Documentation

Release 2.4.25

TrackMe Limited, U.K.

Nov 22, 2026

USE CASES

1 Getting started with TrackMe **3**

2 Use Cases: **7**
- 2.1 Quickstart — hands-on walkthroughs . . . 7
- 2.2 Use-case demos . . . 310
- 2.3 TrackMe Use Cases Referential — Overview . . . 433

3 Artificial Intelligence: **497**
- 3.1 Artificial Intelligence in TrackMe . . . 497
- 3.2 Configuring AI Provider accounts . . . 506
- 3.3 AI Assistant & Concierge . . . 514
- 3.4 The AI Advisors . . . 537
- 3.5 AI Agents for findings . . . 577
- 3.6 AI Routines & Automation . . . 586
- 3.7 Governance . . . 623

4 License & support: **651**
- 4.1 EULA License . . . 651
- 4.2 Terms & Conditions . . . 657
- 4.3 Support . . . 663

5 Compatibility and download: **667**
- 5.1 Compatibility . . . 667
- 5.2 Download . . . 670

6 Requirements: **673**
- 6.1 Requirements for TrackMe . . . 673

7 Installation: **677**
- 7.1 Installation of TrackMe . . . 677
- 7.2 Upgrading TrackMe . . . 680
- 7.3 Migrating and Cloning TrackMe . . . 716
- 7.4 License registration . . . 717

8 Product guide: **725**
- 8.1 Introduction . . . 725
- 8.2 Virtual Tenants & Access . . . 748
- 8.3 Monitoring Components . . . 799
- 8.4 Entity State & Scoring . . . 1216
- 8.5 Classification & Protection . . . 1228
- 8.6 Alerting . . . 1286
- 8.7 Topology Studio . . . 1344
- 8.8 Machine Learning . . . 1445
- 8.9 Integrations & Remote Deployments . . . 1491
- 8.10 Operations & Administration . . . 1650

9 White papers: **1777**
9.1 TrackMe's White Papers . . . 1777
9.2 Running a TrackMe Proof of Concept . . . 1777
9.3 Splunk infrastructure monitoring . . . 1783
9.4 Detection & alerting use cases . . . 1865
9.5 Operations & administration . . . 1944

10 Troubleshoot & FAQ: **1969**
10.1 Troubleshooting TrackMe . . . 1969
10.2 Frequently asked questions (FAQ) . . . 1975

11 Versioning and build history: **1979**
11.1 Release notes: trackme . . . 1979
11.2 Release notes: trackme-broker . . . 2257

12 Third party integrations: **2259**
12.1 Palo Alto XSOAR Integration . . . 2259
12.2 Cribl Stream & Cribl Edge . . . 2263

13 Various **2395**
13.1 Third-party components credits . . . 2395

14 Archives **2687**
14.1 Migrate from TrackMe V1 to TrackMe V2 . . . 2687

TrackMe has transitioned to a new licensing model.

- The Free Community Edition has been discontinued.
- What this means for you:
  - Existing licensed customers are not impacted. Your current license continues unchanged.
  - Community users are automatically upgraded to a Foundation Edition trial for 90 days, with full functionality enabled.
- After the trial period:
  - TrackMe remains active and fully usable for monitoring
  - The platform enters read-only mode
  - You can continue to view data, dashboards, alerts, and history
  - Creation of new tenants, trackers, or entities will be disabled
  - To continue using TrackMe with full capabilities after the trial, a Foundation license is required.

Buy Foundation Edition: https://trackme-solutions.com/get-foundation/

See all plans and pricing: https://trackme-solutions.com/pricing/

Contact us: contact@trackme-solutions.com

The PDF version of this documentation is available here. For a nine-page overview of the product, see the `product brochure (PDF)`.

TrackMe for Splunk is the ideal companion for your Splunk deployment, no matter the size of your environment. Its unique capabilities help you on a daily basis to get the best value from your Splunk investments:

- Discover and maintain Splunk entities at scale, track availability and quality of any kind of data in Splunk
- Virtual tenancy is a key concept in TrackMe which allows creating on the fly knowledge objects in a repeatable way: create and scope tenants, experiment, destroy and restart as needed
- TrackMe allows tracking your local Splunk deployment, or transparently any number of Splunk remote deployments (subject to licensing restrictions)
- TrackMe's unique workflow combines best Splunk capabilities, from a comprehensive user interface to notable events generation, SLA tracking and many more
- Get the best from TrackMe components, using splk-feeds components provide deep Splunk feed tracking, splk-vol (Volume Outliers) detects abnormal drops and spikes of the licensed volume of every index, splk-cim provides Common Information Model (CIM) compliance tracking, splk-flx (FLEX) adapts to any kind of Splunk magic query, and the dedicated tenant types raise findings: splk-uam (User Activity Monitoring) tells you who is doing what on your Splunk, the Cribl Stream tenant monitors your Cribl Stream deployment from its own metrics, the Cribl Edge tenant follows your Cribl Edge Fleets and Edge Nodes at any fleet size, the Splunk HEC tenant monitors your HTTP Event Collector endpoints, pools and tokens from Splunk's own telemetry, the Splunk S2S tenant watches your receivers and intermediate forwarders pool by pool, the Splunk UF Fleet tenant follows every forwarding agent and its Deployment Servers, and the Splunk Forwarder Collectors tenant follows your collection points and every input they run! (components are subject to license restrictions)
- Extend the visibility at any point in time with Hybrid and Elastic trackers, use Machine Learning outliers detection with deep and easy control, TrackMe is incredibly rich in features

CHAPTER

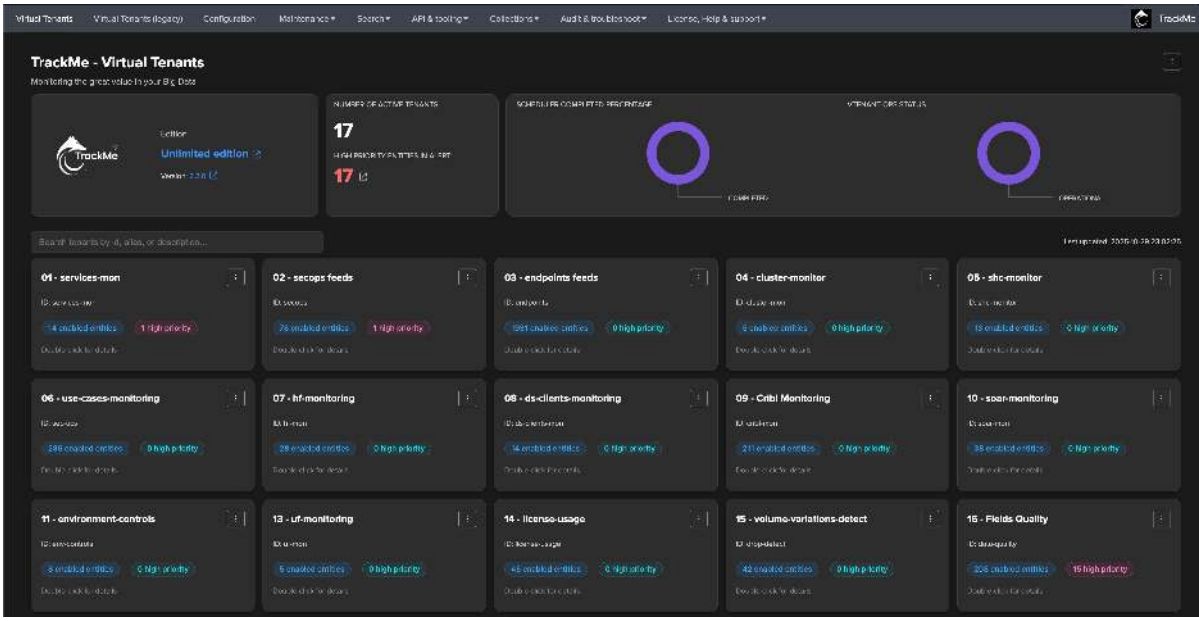
ONE

GETTING STARTED WITH TRACKME

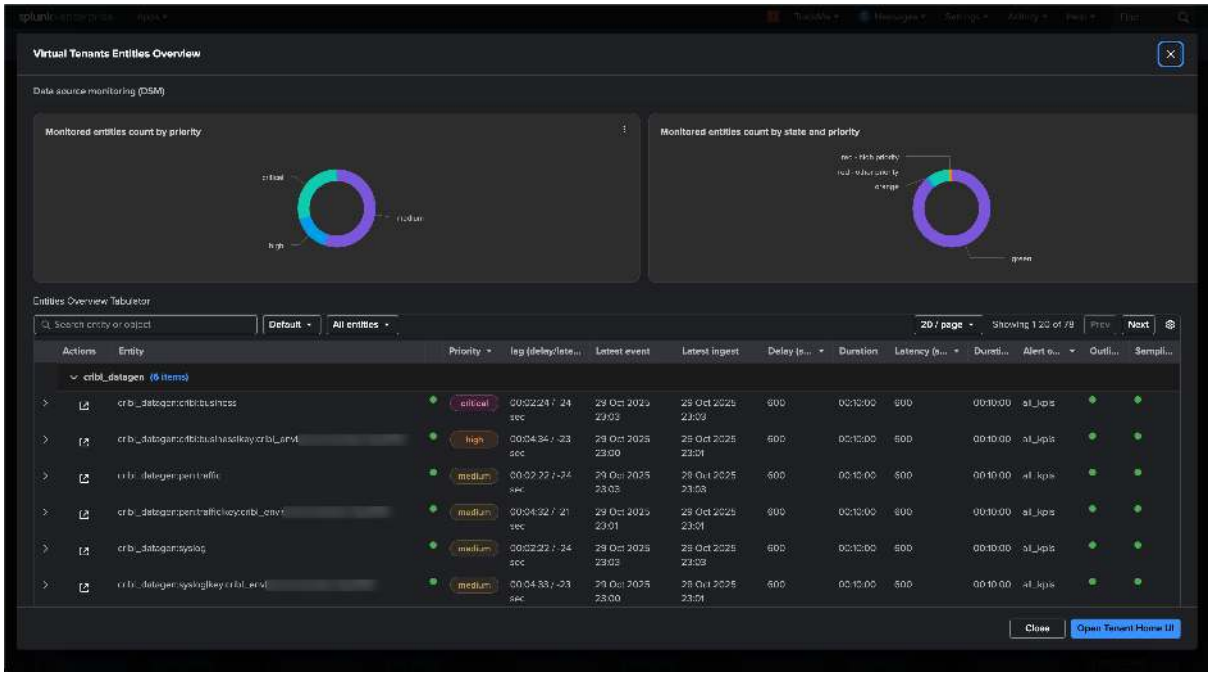
Quickstart Guide Our quickstart guide covers all the steps needed to get TrackMe up and running, from installation to your first tracker configuration.

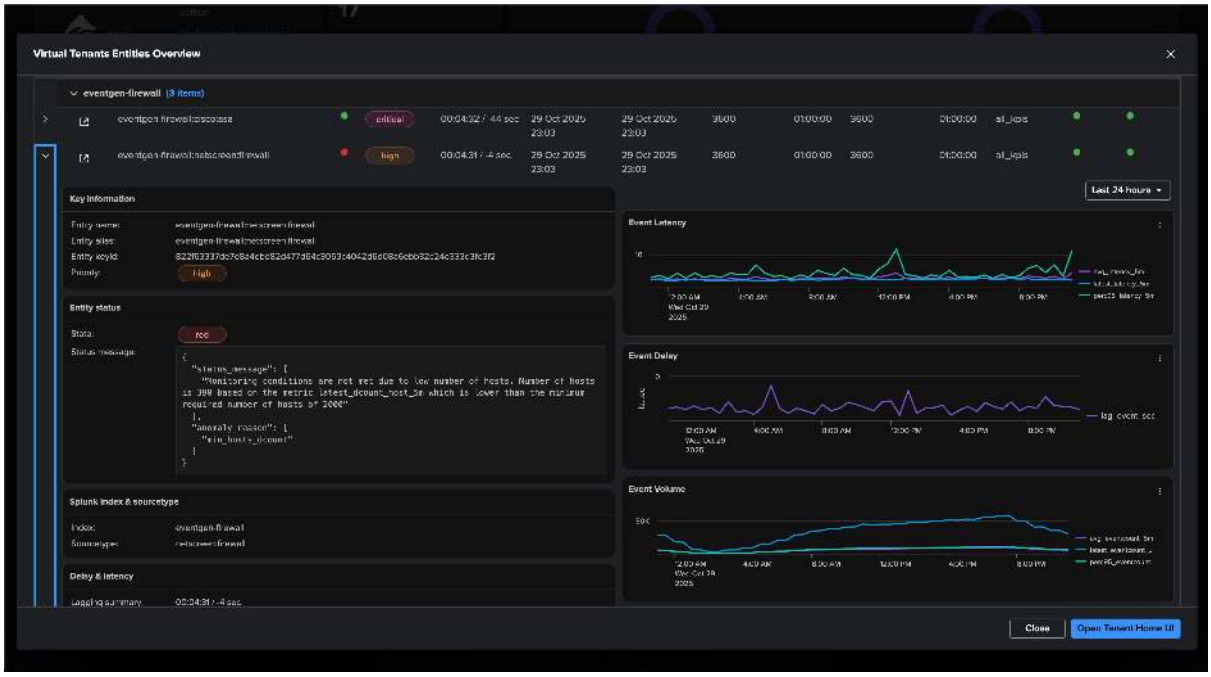
QUICK START - Starting with TrackMe: (feed tracking quickstart) Installation Step-by-step installation instructions for TrackMe on Splunk Enterprise and Splunk Cloud, including requirements and configuration.

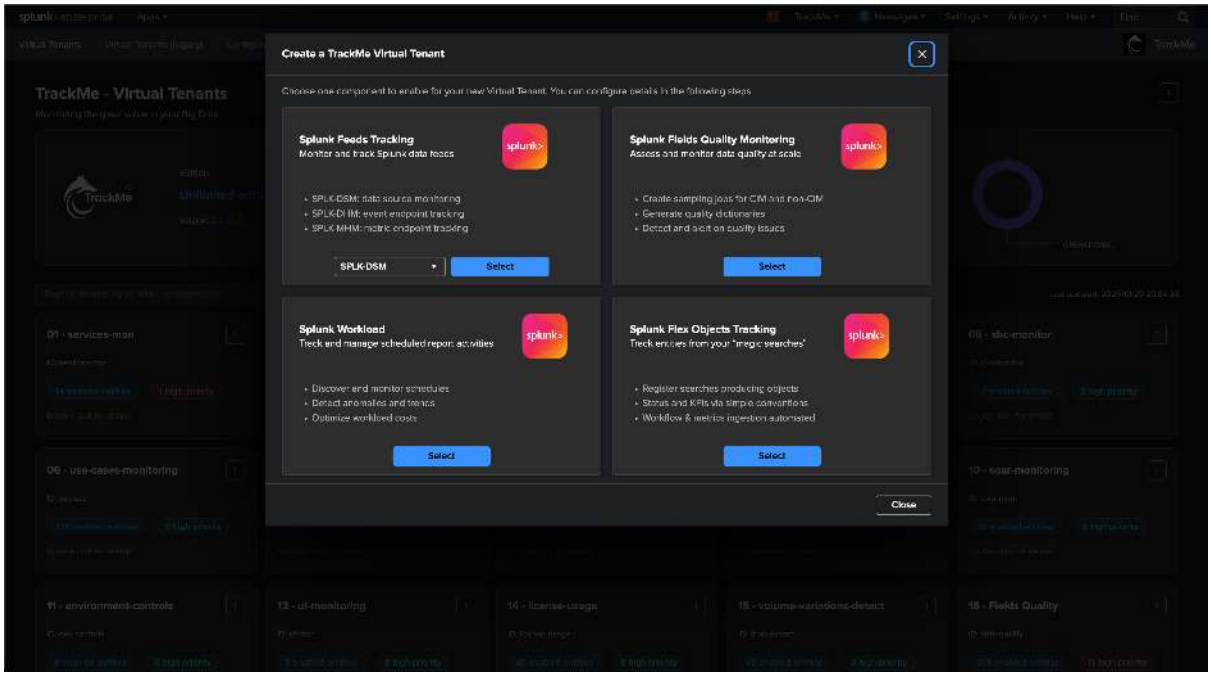
Installation of TrackMe Product Guide The complete guide to configuring and using TrackMe: virtual tenants, trackers and components, monitoring states, alerting, machine learning, operations and integrations.

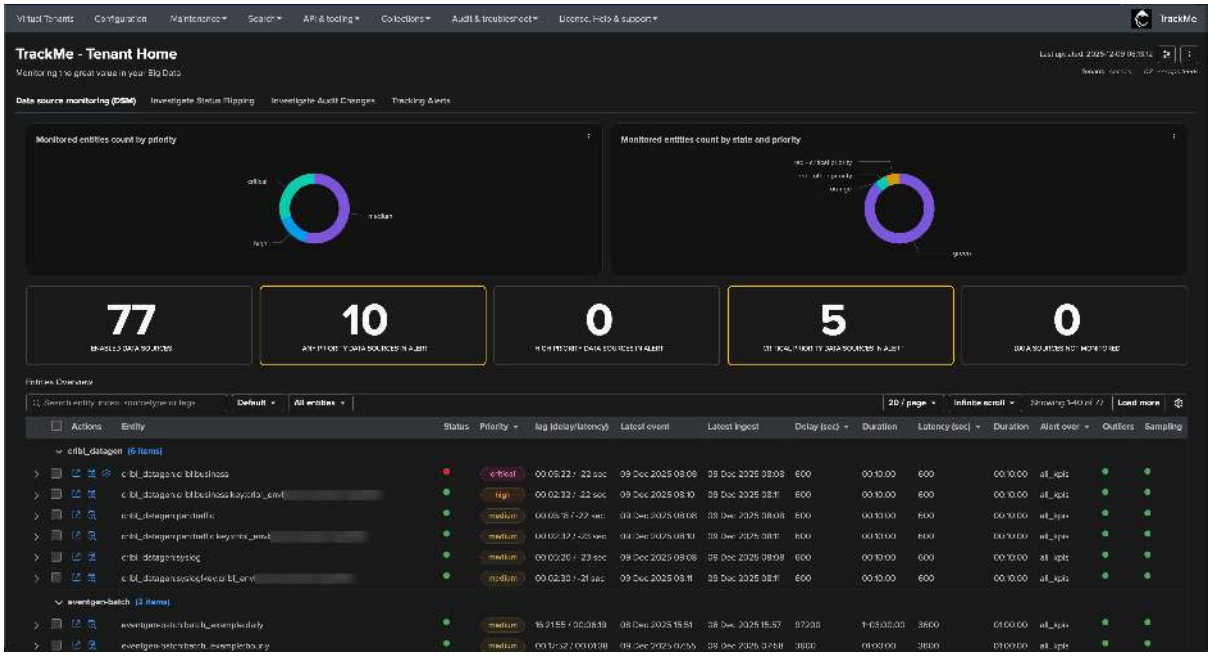


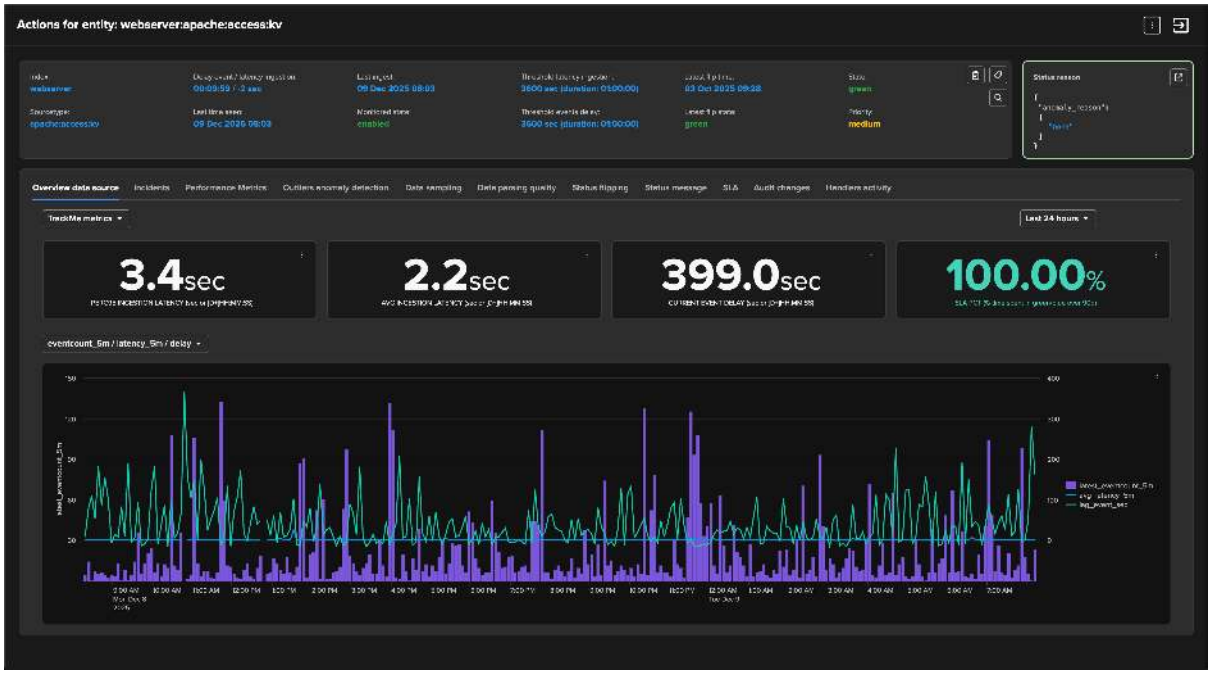
Introduction


Virtual Tenants Entities Overview
Data source monitoring (DSM)
Monitored entities count by priority
Monitored entities count by state and priority
Entities Overview Tabulator
Actions
Entity
Priority
Close


Virtual Tenants Entities Overview
eventgen-firewall (3 items)
Key information
Entity status
Splunk index & sourcetype
Delay & latency
Event Latency
Event Delay
Event Volume
Close


Create a TrackMe Virtual Tenant
Splunk Feeds Tracking
Monitor and track Splunk data feeds
Splunk Fields Quality Monitoring
Assess and monitor data quality at scale
Splunk Workload
Track and manage scheduled report activities
Splunk Flex Objects Tracking
Track entities from your "mega searches"
Select
Close


TrackMe - Tenant Home
Monitored entities count by priority
Monitored entities count by state and priority
77
10
0
5
0


Actions for entity: webserver:apache:accesskv
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
TrackMe metrics
Last 24 hours
3.4sec
2.2sec
399.0sec
100.00%
eventcount_5m / latency_5m / delay

CHAPTER

TWO

USE CASES:

2.1 Quickstart — hands-on walkthroughs

Guided walkthroughs of TrackMe, captured live on the product: every page is a slide deck with a description of what each slide shows — read it top to bottom, present it, or download it.

Suggested order for a first contact with the product: the introduction, then the three parts of the series (a first tenant, classification, alerting), then the walkthroughs of the components you plan to use.

Product introduction What TrackMe watches — your data, the people, the forwarders and collectors, the HEC endpoints and the Cribl pipelines around it — how it decides and how it tells you: the vocabulary before the live product. *30 slides · 25 min.*

Product introduction Part 1 — Your first DSM tenant Create a Virtual Tenant, let the hybrid tracker discover the feeds, read the first entity in alert and why. *16 slides · 20 min.*

Part 1 — Your first DSM tenant Part 2 — Classify with policies Priority and tags from a CMDB lookup, simulated before saving; the on-call and ownership views. *12 slides · 15 min.*

Part 2 — Classify with policies Part 3 — Stateful alerting One alert scoped to critical and high; the incident lifecycle, the email with charts and the AI report. *16 slides · 20 min.*

Part 3 — Stateful alerting Topology Studio A map generated from a tenant, one authored from scratch, Topology Alerts, a map built with AI, and the maps the findings tenants draw. *40 slides · 40 min.*

Topology Studio Field Quality Monitoring CIM and custom dictionaries, sampling, the field-quality advisors. *35 slides · 40 min.*

Field Quality Monitoring Volume Outliers Licensed volume per index, verified figures, ML outliers on drops and spikes, the ML Advisor. *43 slides · 35 min.*

Volume Outliers User Activity Monitoring Who is doing what on your Splunk: Finding groups, privileged users, knowledge objects, custom policies, emails to the account concerned, and which data is read. *48 slides · 48 min.*

User Activity Monitoring Cribl Stream A Cribl Stream deployment from its own metrics: Finding groups and triage, traffic ML models and their tuning, Workers and data paths, the AI Agent. *33 slides · 35 min.*

Cribl Stream Cribl Edge The Fleets and Edge Nodes of a Cribl Edge Leader at any size: connectivity, rollouts, version drift, any node up close, and the AI Agent. *30 slides · 30 min.*

Cribl Edge Splunk HEC The HTTP Event Collector of a deployment: endpoints from their heartbeats, pools per data centre, tokens by name, a heavy forwarder stopped, and the AI Agent. *32 slides · 30 min.*

Splunk HEC Splunk S2S The receiving side of a deployment: an indexer cluster as one pool per site, the heavy forwarder tier as an automatic pool, what is blocked, the data path, a heavy forwarder stopped, and the AI Agent. *26 slides · 30 min.*

Splunk S2S Splunk UF Fleet The forwarding agents of a deployment at any fleet size: Deployment Server accounts, CMDB enrichment, groups, expected agents, a site gone dark, and the AI Agent. *26 slides · 30 min.*

Splunk UF Fleet Splunk Forwarder Collectors The collection points of a deployment and every input they run: groups and their minimum, syslog collectors declared, inputs that must stay enabled, and a collector stopped on purpose. *30 slides · 30 min.*

Splunk Forwarder Collectors

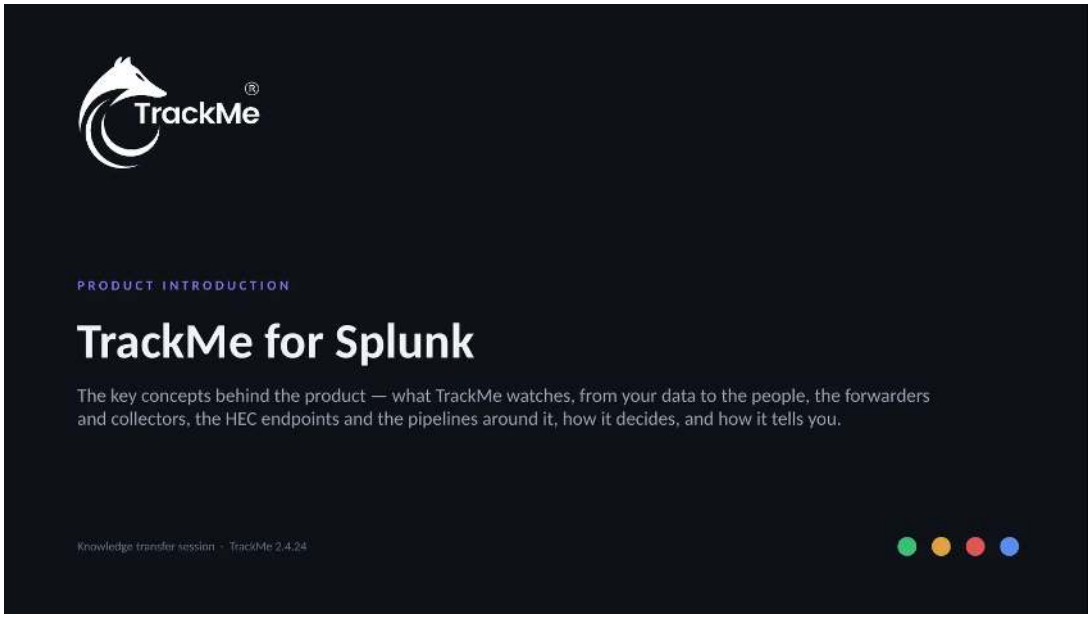
2.1.1 Product introduction

What TrackMe is and what it watches, from your data to the licensed volume, the people, the forwarders, receivers and collection points, the HTTP Event Collector endpoints and the Cribl pipelines around it; the two ways of watching — entities and findings — how it decides and how it tells you: the vocabulary and the mental model before the live product.

`Download the deck (PDF) · PowerPoint`

about 25 minutes · 30 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



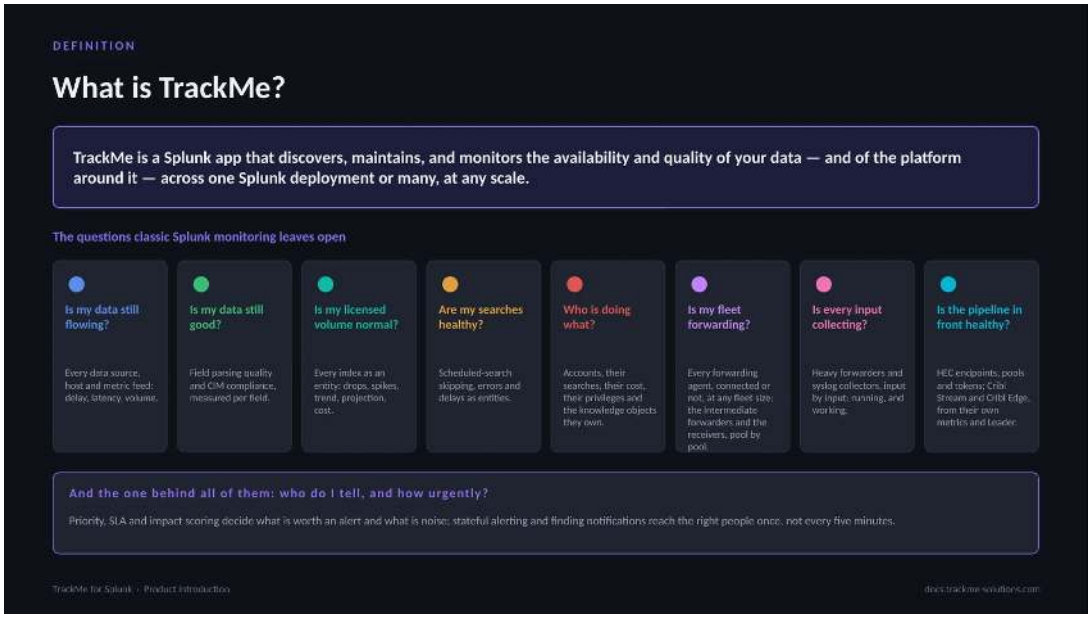
This introduction sets the vocabulary and the mental model of TrackMe 2.4.24. Everything here maps one to one to the Product Guide on docs.trackme-solutions.com. TrackMe started with the data — is it arriving, is it good — and today it covers the platform around it too: the licensed volume of every index, the people and service accounts using Splunk, the forwarders and receivers carrying the data, the collection points running add-on and syslog inputs, the HTTP Event Collector endpoints receiving it, and the Cribl pipelines in front of it.

What breaks silently



Most teams come to TrackMe because of one specific pain: a critical feed died hours ago and nobody noticed; a host went silent while the index still looked busy; a field extraction drifted and a detection stopped firing; scheduled searches started skipping. And the platform around the data fails just as quietly: an index doubling its licensed volume, an account running all-time searches on a schedule nobody owns, two hundred forwarders dropping at once behind a receiver tier, an intermediate forwarder whose output pauses and holds every sender behind it, a database input that stopped collecting, a HEC token gone silent behind a load balancer that still answers, a Cribl Destination in backpressure in front of Splunk. One root cause: Splunk does not monitor itself, and you cannot watch or classify all of it by hand.

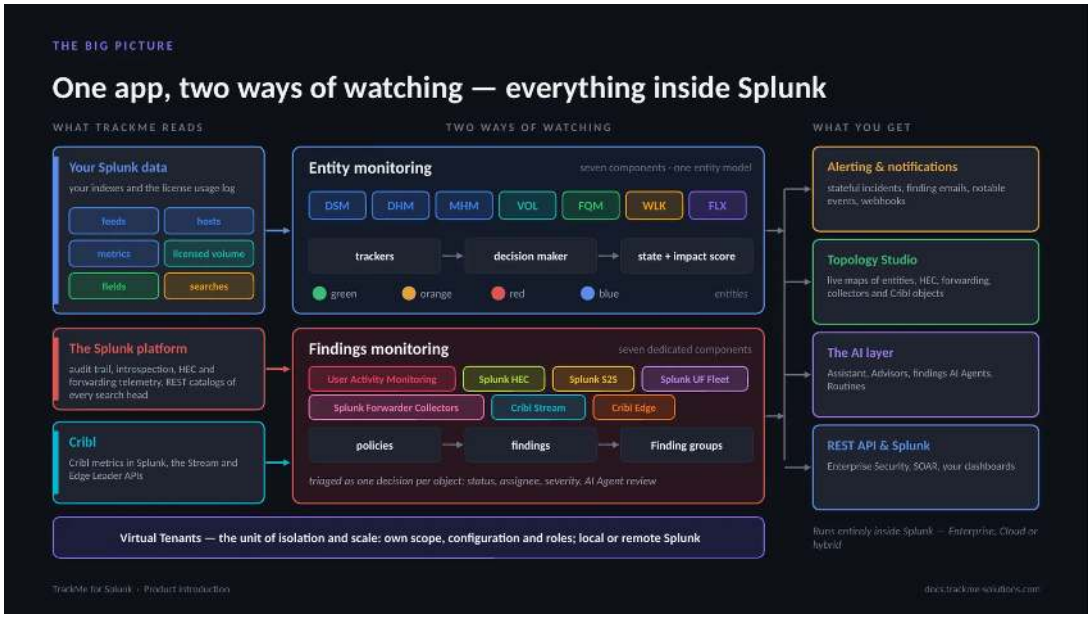
What TrackMe is



TrackMe discovers, maintains and monitors the availability and quality of your data. It is a control plane on top of Splunk that runs entirely inside Splunk — on-premises, Splunk Cloud or hybrid — with no outbound SaaS. It answers the questions classic Splunk monitoring leaves open, from the data itself to the platform around it: is the data flowing, is it good, is the licensed volume normal, are the searches healthy, who is doing what, is my fleet forwarding, is every input collecting, is the pipeline in front healthy, from the HEC endpoints to Cribl Stream and Cribl Edge. Who it is for: Splunk administrators proving the platform is healthy, SOC and detection engineers who depend on

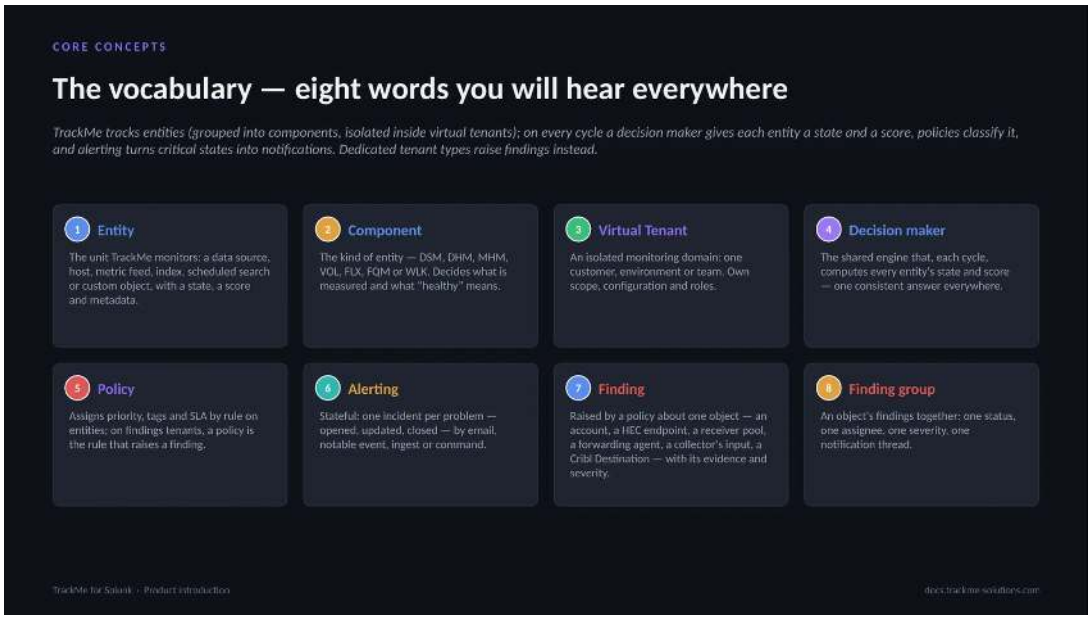
complete, well-parsed data, platform owners who answer for the licence and the usage, service owners reporting SLA attainment, and managed service providers.

The big picture



On the left, what TrackMe reads: your Splunk data, the Splunk platform itself, and Cribl. In the middle, two ways of watching. Entity monitoring is the historical core: seven components turn data feeds, hosts, metrics, the licensed volume of each index, field quality, scheduled searches and any SPL into entities with a state and an impact score. Findings monitoring is the second model, used by seven dedicated components — User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge: policies raise findings, grouped per object into Finding groups that you triage. Both live in Virtual Tenants. On the right, what you get from both: alerting and notifications, live maps in Topology Studio, the AI layer, and the REST API.

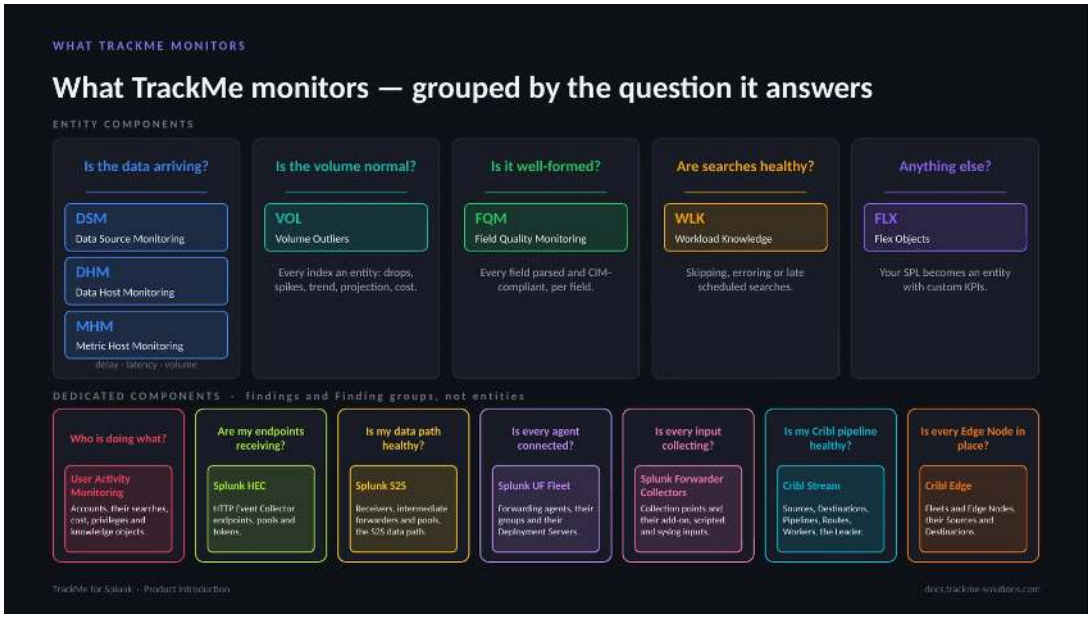
The vocabulary



TrackMe has a small, consistent vocabulary that appears on every page of the guide and in the UI. Entity monitoring: an entity belongs to a component, lives in a virtual tenant, gets its state from the decision maker; policies classify it and alerting turns critical states into notifications. Findings monitoring adds two words: a finding, raised by a policy about one object — an account, a HEC endpoint, a receiver pool, a forwarding agent, a collector's input, a Cribl

Destination — and the Finding group that gathers an object's findings into one decision. There is no global list of entities: everything lives inside exactly one tenant.

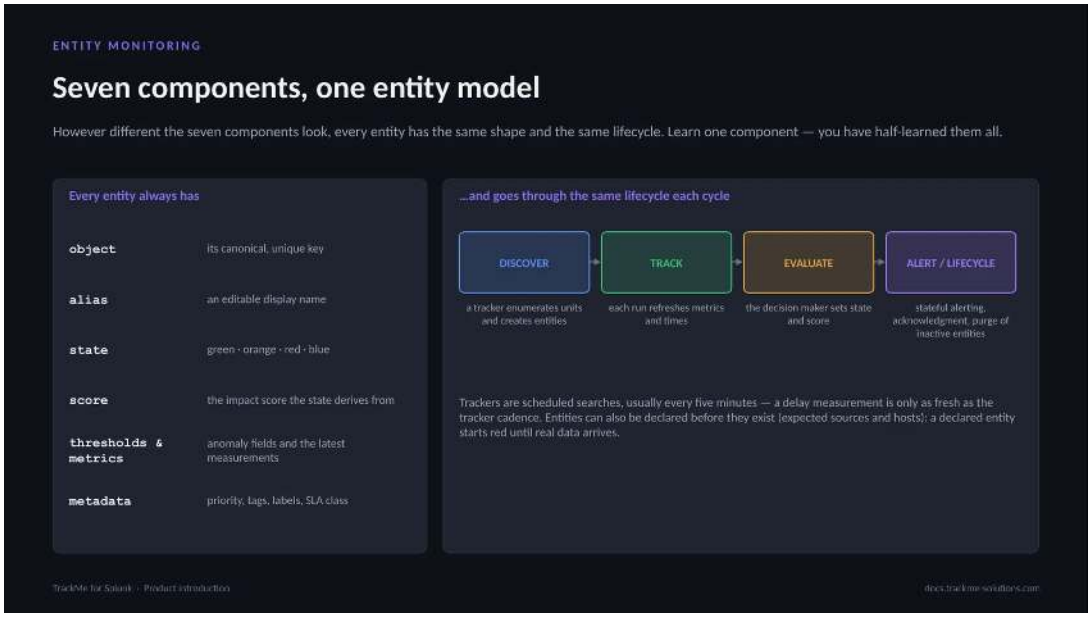
What TrackMe monitors



TrackMe monitors two families, grouped by the question each answers. Seven entity components: DSM, DHM and MHM answer whether the data is arriving, VOL whether the licensed volume of each index is normal, FQM whether the data is well-formed, WLK whether the scheduled searches are healthy, and FLX monitors anything else from the SPL you write.

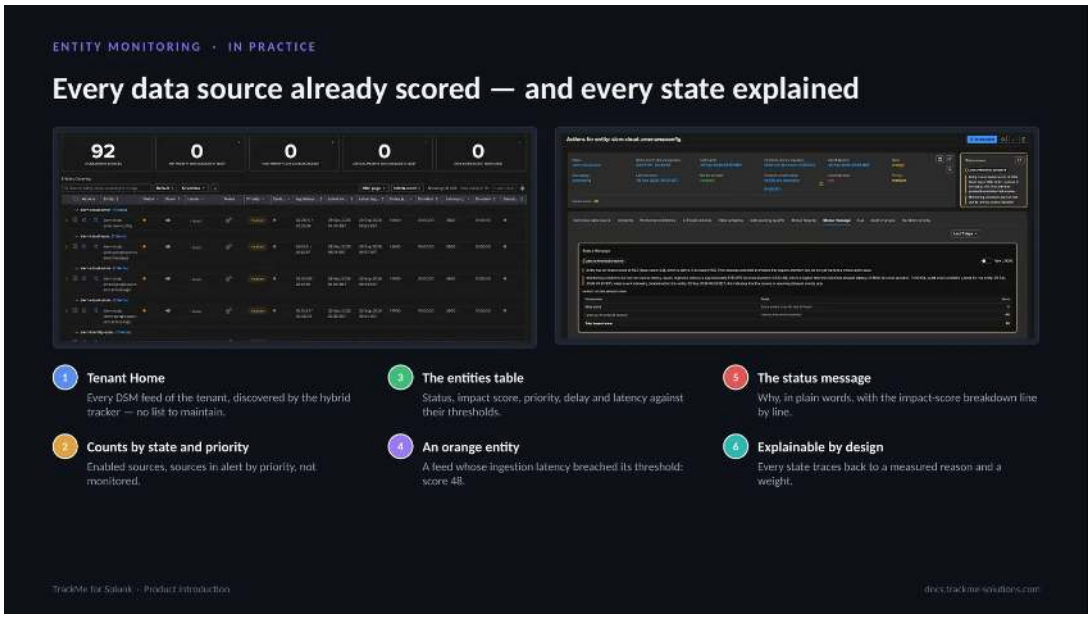
Beside them, seven dedicated components raise findings and Finding groups instead of tracking entities: User Activity Monitoring answers who is doing what on Splunk, Splunk HEC whether every HTTP Event Collector endpoint, pool and token is receiving, Splunk S2S whether the receivers and the intermediate forwarders accept and keep up, pool by pool, Splunk UF Fleet whether every forwarding agent is connected and its Deployment Servers are healthy, Splunk Forwarder Collectors whether every input keeps collecting, Cribl Stream whether the Cribl pipeline in front of Splunk is healthy, and Cribl Edge whether every Edge Node is where it should be. Almost every deployment starts with DSM.

One entity model



The shared entity model is what makes TrackMe consistent: the same object, alias, state and score shape, the same lifecycle. Classification, protection, stateful alerting, scoring and ML outliers are documented once and apply to all seven components. Where they differ: variable delay and lagging classes are DSM and DHM; data sampling and elastic sources are DSM; dynamic thresholds are FLX and FQM; versioning and orphan detection are WLK; VOL has one entity per index, fed incrementally from the license usage log, with ML outliers as its state.

Entity monitoring in practice



Tenant Home lists every data source the hybrid tracker discovered, already scored: the counts by state and priority on top, then the entities with their status, impact score, priority, delay and latency against their thresholds. Here several feeds are orange with a score of 48.

The status message of one of them explains why in plain words: its ingestion latency breached the threshold, which adds 48 to the impact score, above zero and below 100, so orange. The breakdown lists every contribution line by line: every state traces back to a measured reason and a configurable weight.

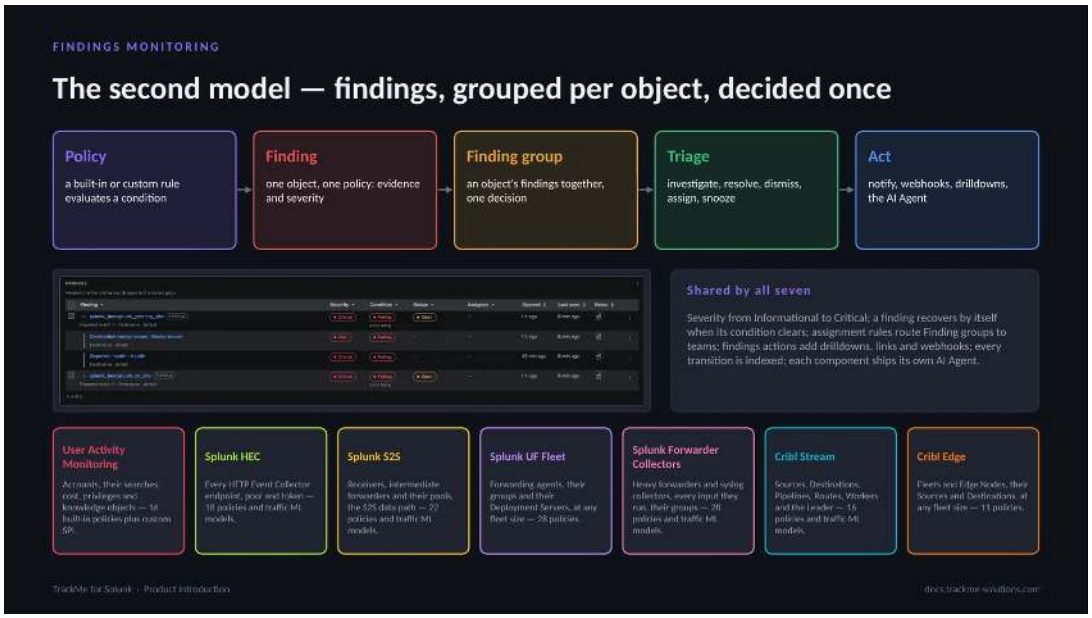
Volume Outliers



Volume detection has always been part of TrackMe — DSM volume outliers, FLX licence-usage use cases. VOL is

its standardised, first-class form at scale: one entity per Splunk index, the licensed volume read incrementally from the license usage log with one search over a single source every five minutes, never a 24-hour rescan; ML outliers on drops and spikes, one model per index; trend and month projection. The Global license usage view answers the licence questions people otherwise build dashboards for: who consumes, the trend, days over quota, working-day averages, costs and chargeback. VOL is part of the Foundation edition.

The findings model



Some things are not entities with a health score: an account that runs all-time searches, a HEC token gone silent, a receiver pool with a member silent, a fleet of forwarders disconnecting together, a database input that stopped collecting, a Cribl Destination in backpressure, an Edge Fleet left on an old version. For those, seven dedicated components — User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge — share one findings model. A policy evaluates a condition and raises a finding about one object, with its evidence and a severity on the TrackMe scale from Informational to Critical. All findings about one object form a Finding group: one status, one assignee, one severity, one notification thread. Findings recover by themselves when the condition clears; a person decides the rest. The same model brings assignment rules, webhooks and notifications, every component ships its own AI Agent, and nothing is installed on the platforms being watched.

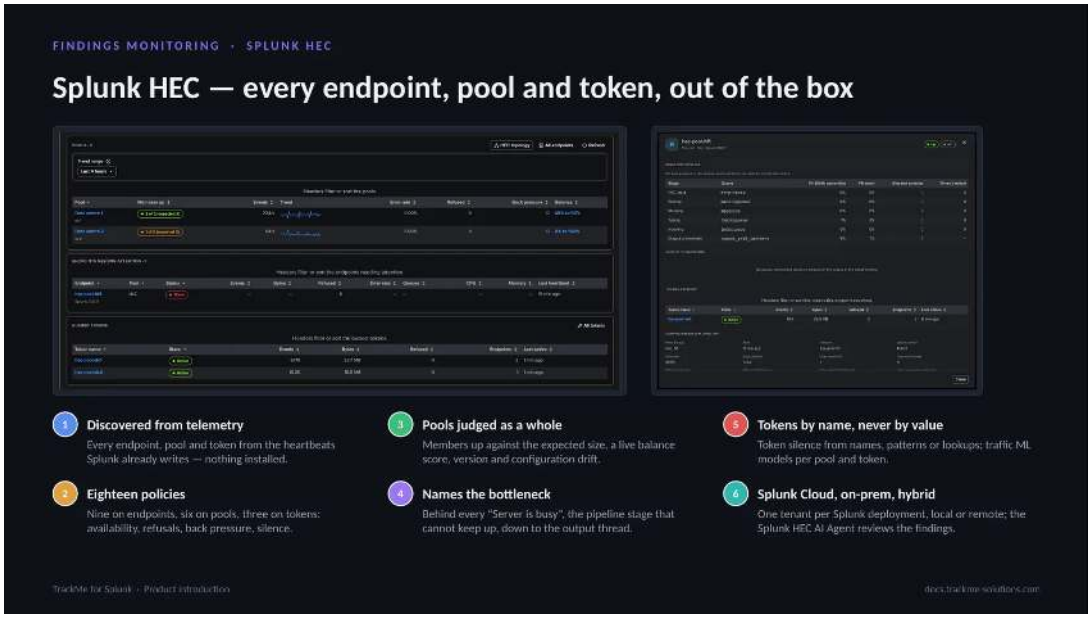
User Activity Monitoring



User Activity Monitoring is a dedicated tenant type for one indexing layer seen through one to ten search head tiers. The Findings tab groups the findings about an account and the saved searches it owns into one Finding group; the account view in front shows one person's activity over the last 24 hours, its executions by search type, run time and the apps used, next to the catalogs of what the account owns.

Sixteen built-in policies in four families — inventory, activity, resources and privileged users — and custom policies in SPL raise the findings; a Privileged users tab shows which accounts hold high privileges and why. UAM can email the account concerned, the UAM AI Agent reviews findings on demand or on a schedule, and nothing is ever changed on the platform.

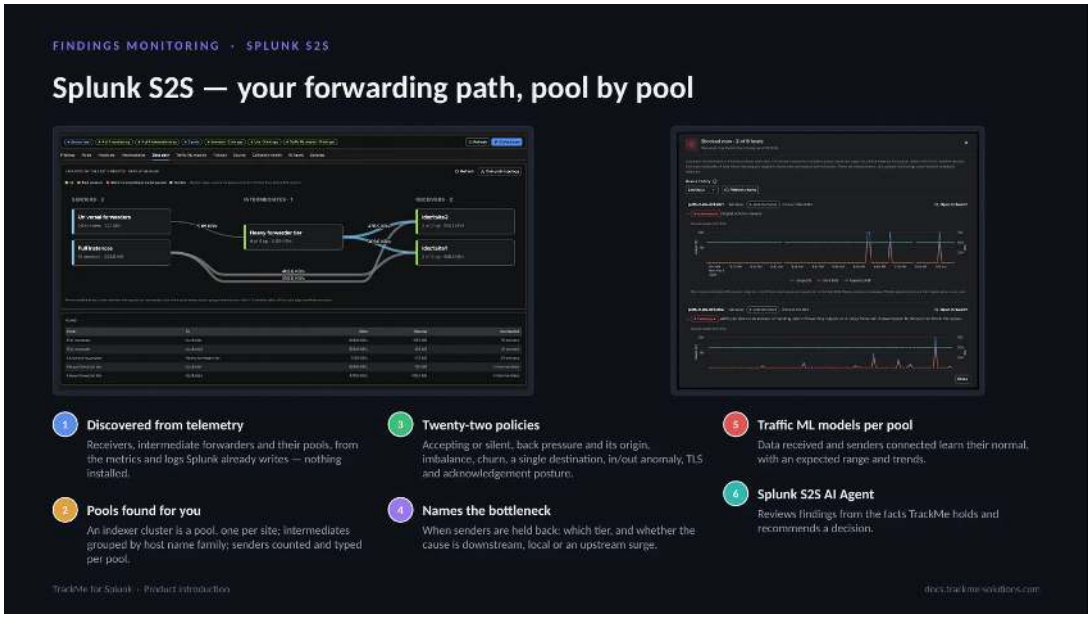
Splunk HEC



A Splunk HEC tenant watches the HTTP Event Collector of one Splunk deployment, Splunk Cloud included, from the telemetry Splunk already writes: every HEC instance logs a heartbeat to _introspection every minute. Endpoints are discovered automatically and grouped into pools, the hosts behind one load balancer, judged as a whole: members up against the expected size, a live balance score, version and configuration drift. Tokens are known by name only, never by value.

Eighteen built-in policies raise findings — availability, senders refused, back pressure naming the pipeline stage that cannot keep up, pool imbalance, token silence — and traffic ML models learn every pool and token. On the left, the pools with one member gone silent; on the right, one endpoint's ingestion pipeline, every queue with its fill. Nothing is installed on the endpoints.

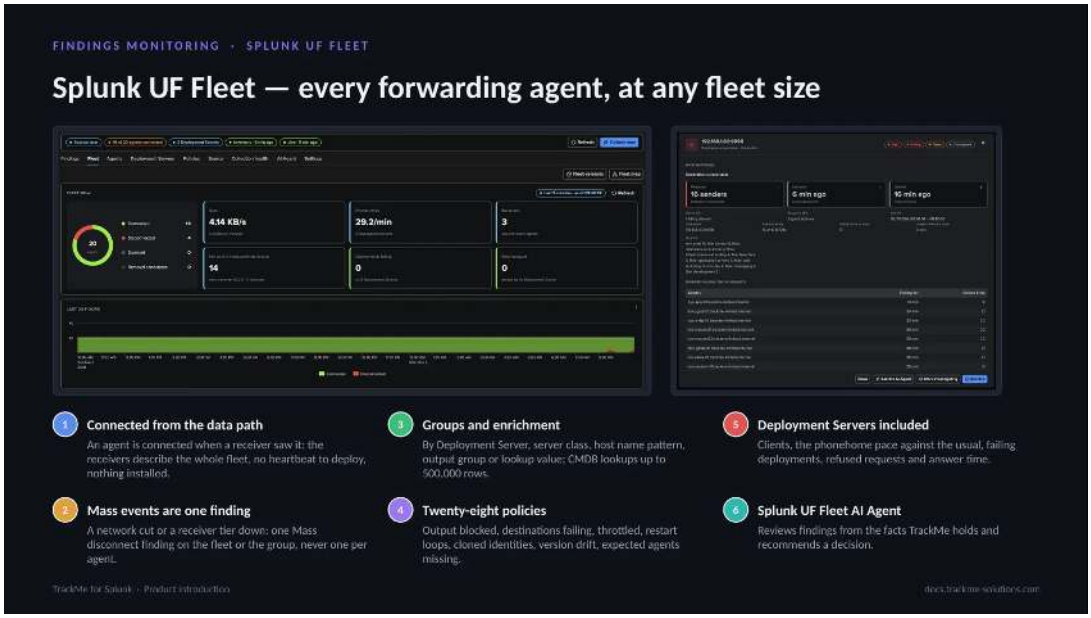
Splunk S2S



A Splunk S2S tenant watches the receiving side of forwarding for one Splunk deployment: the receivers, the intermediate forwarders and the pools they form, from the metrics.log connection lines, queues and throughput, the splunkd.log messages and the health report Splunk already writes. An indexer cluster is a pool, one per site of a multisite cluster; intermediate forwarders are grouped by host name family; senders are counted and typed per pool. On the left, the data path of the last five minutes with the rate on every edge; on the right, Blocked now names the queue that cannot keep up on each host.

Twenty-two built-in policies raise findings: a receiver that stops accepting or goes silent, back pressure and its origin, imbalance, churn, a single destination, TLS and acknowledgement posture. Traffic ML models learn every receiver pool, and the Splunk S2S AI Agent reviews the findings. Nothing is installed on any Splunk instance.

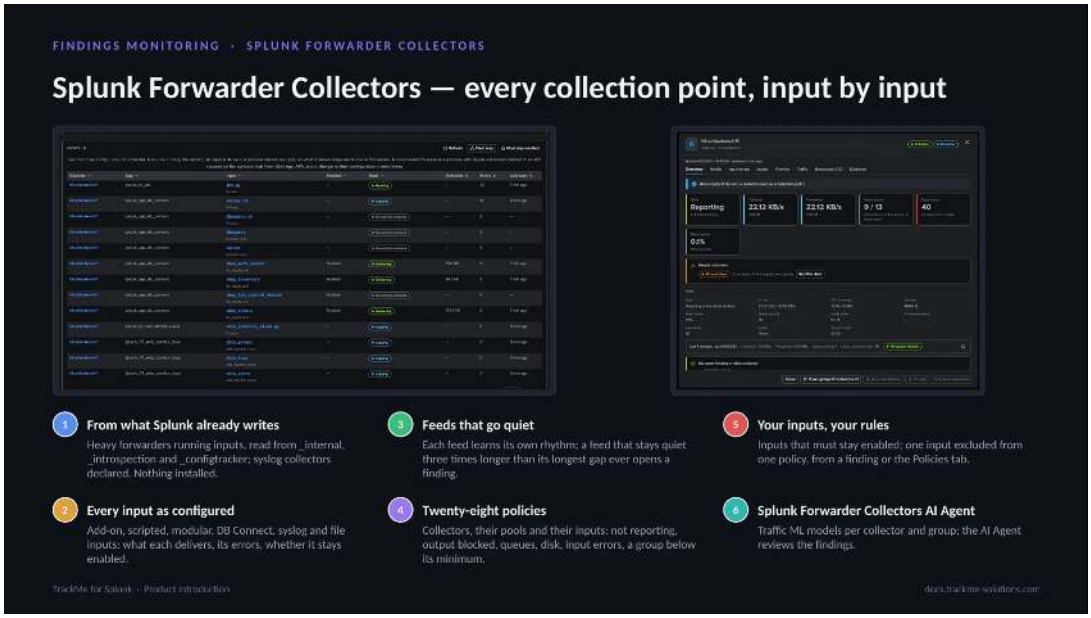
Splunk UF Fleet



A Splunk UF Fleet tenant follows the forwarding agents of one Splunk deployment and the Deployment Servers that manage them. An agent is connected when a receiver saw a connection from it within the connected window: the receivers describe the whole fleet, whatever its size, with no heartbeat to deploy. On the left, the fleet now: connected, disconnected and dormant agents, versions, Deployment Servers and the last 24 hours; on the right, a Destination unreachable finding with its sixteen failing senders listed by group.

Mass events are one finding: when many agents disconnect together, one Mass disconnect finding on the fleet or on the group says it. Groups come from the Deployment Servers, server classes, host name patterns, output groups or lookup values; enrichment lookups carry the CMDB onto every agent. Twenty-eight built-in policies cover the agents, the fleet and the Deployment Servers; the Splunk UF Fleet AI Agent reviews the findings.

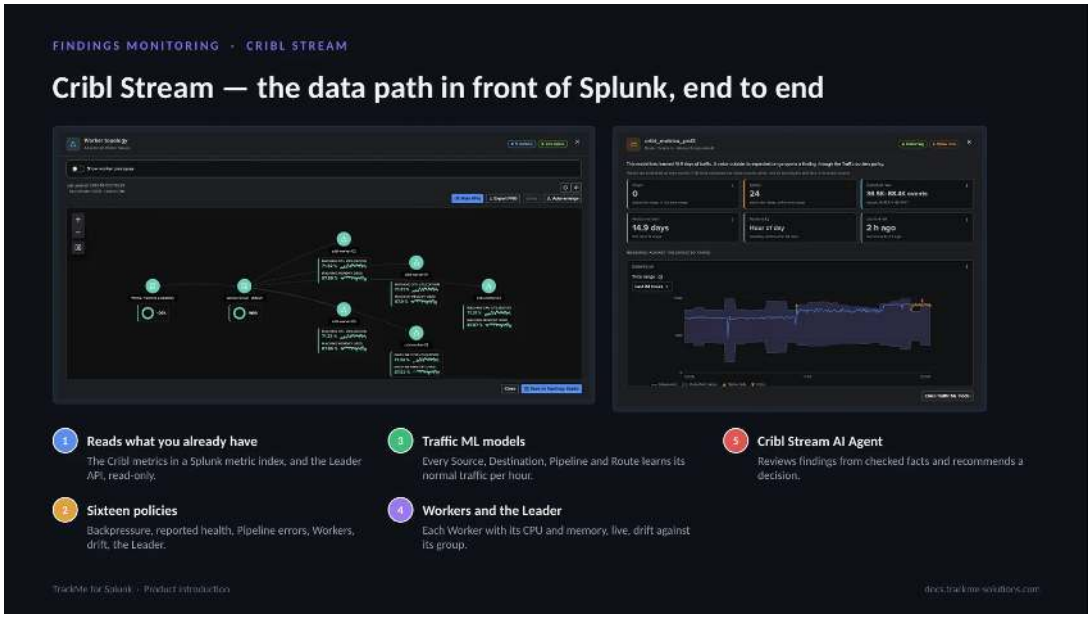
Splunk Forwarder Collectors



A Splunk Forwarder Collectors tenant follows the forwarders a Splunk deployment uses as collection points: heavy forwarders running DB Connect, cloud and API add-ons, scripted and modular inputs, and the forwarders reading syslog files, declared on a group. It reads what Splunk already writes; nothing is installed. On the left, every input as configured, per collector, with its state, the data it delivered and its errors; on the right, one collector opened on its throughput, inputs and errors.

An input is judged by what it delivers, not only by whether it runs. Each feed learns its own rhythm and opens a finding when it stays quiet three times longer than its longest gap ever. Twenty-eight built-in policies cover the collectors, their groups and their inputs; the Splunk Forwarder Collectors AI Agent reviews the findings.

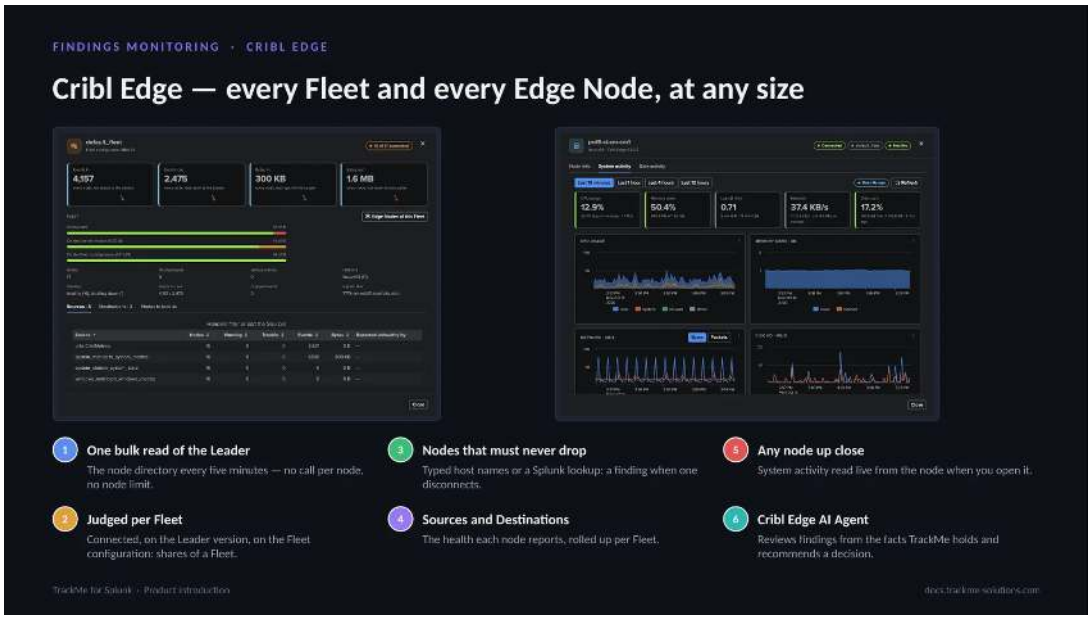
Cribl Stream



A Cribl Stream tenant watches one Cribl Stream deployment end to end, from the metrics the Workers already send to a Splunk metric index and from the Leader API, read-only within a shared request budget. On the left, the Worker topology of a self-managed deployment, each Worker with its CPU and memory, live. On the right, a traffic ML model of a Route: what it learned over two weeks, its expected band and the spikes above it.

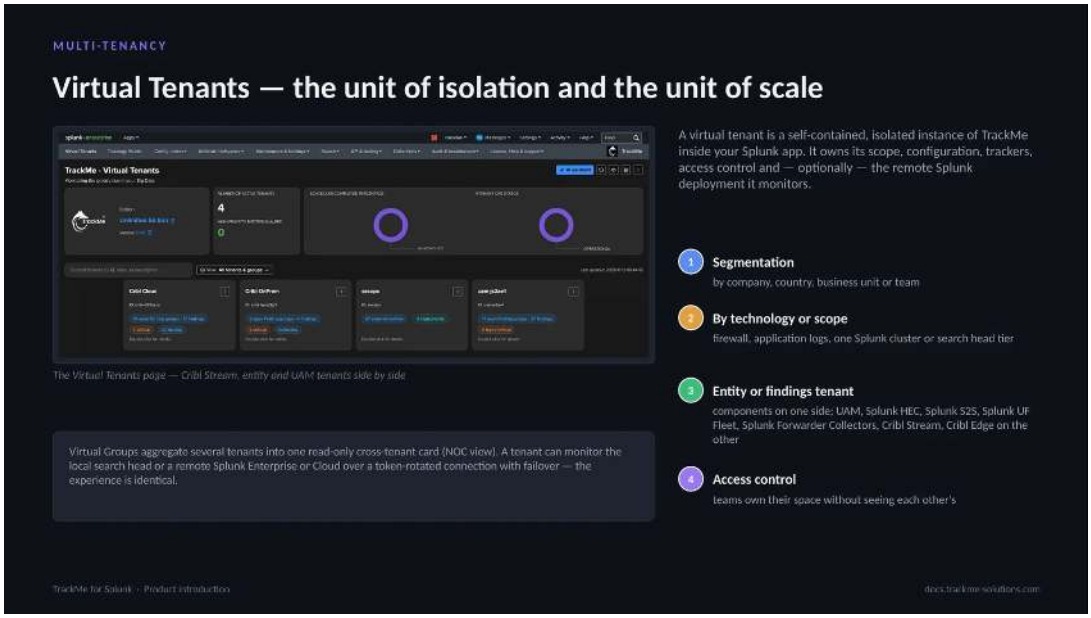
Sixteen built-in policies raise findings: backpressure, reported health, Pipeline errors, Workers that stop or drift. The Cribl Stream AI Agent reviews findings from checked facts. TrackMe never changes anything in Cribl.

Cribl Edge



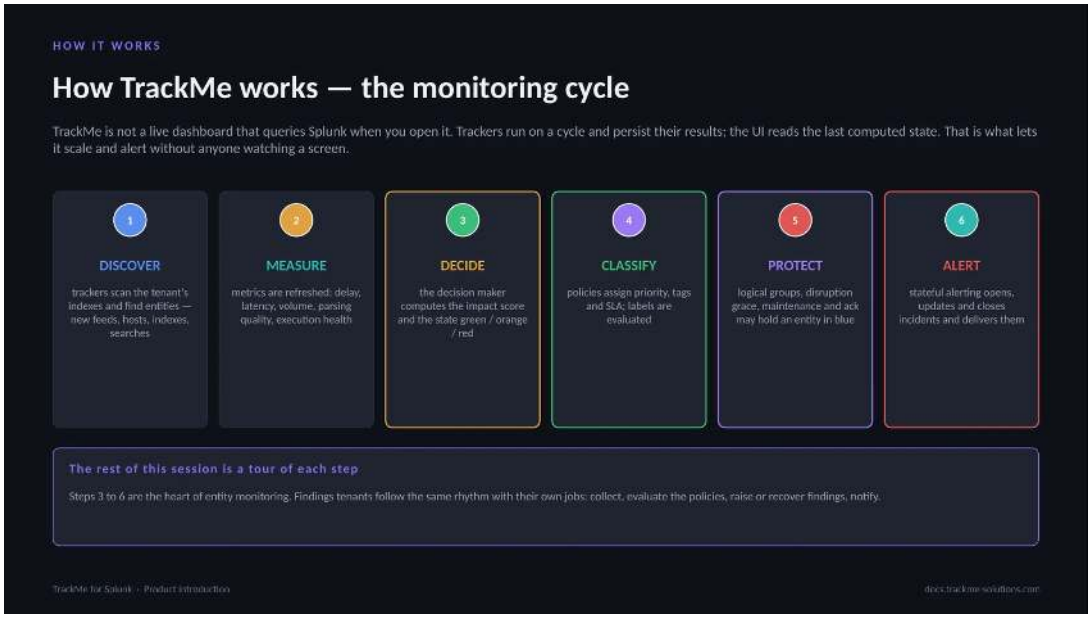
A Cribl Edge tenant follows the Fleets of one Cribl Edge Leader, the Edge Nodes enrolled in them and the Sources and Destinations they run. It reads the Leader's node directory in bulk, so its cost does not grow with one call per node and there is no node limit. Fleets are judged as a share of their nodes: one laptop going to sleep opens nothing, a fifth of a Fleet dropping does. Version drift after a rollout, configuration drift after a deploy, failing inputs and outputs and full disks become findings. Any node opens with its system activity read live through the Leader. Edge has no machine learning by design; its AI Agent reviews the findings.

Virtual Tenants



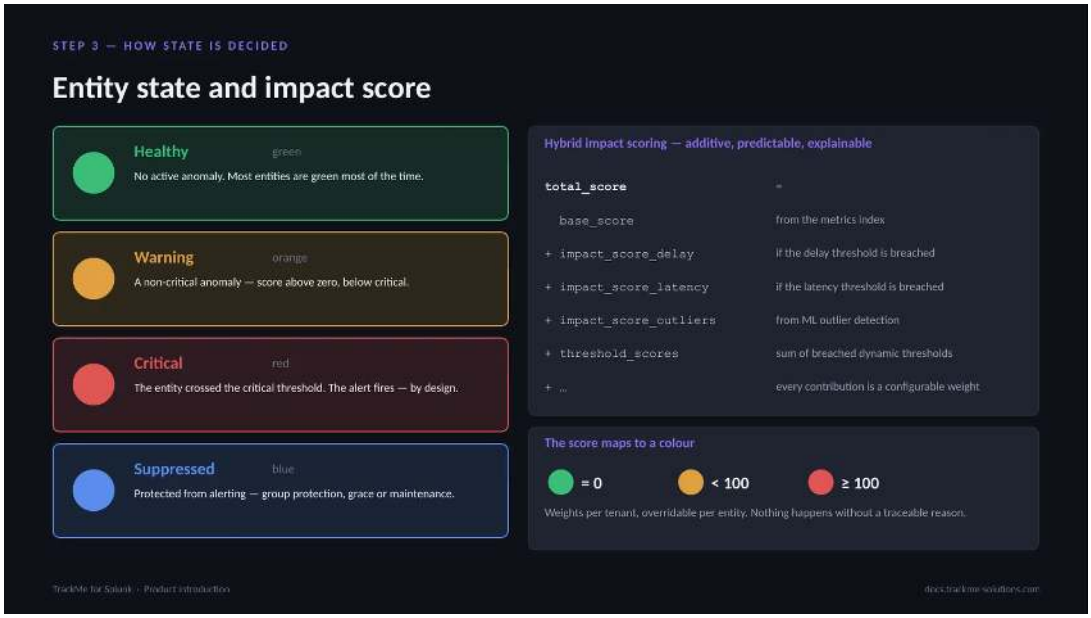
A single flat monitoring space does not survive contact with a real Splunk estate. Tenants carve the estate into domains that match how the organisation works. Everything — entities or findings, policies, thresholds, alerts — is scoped to a tenant. An entity tenant enables one or more components: splk-feeds for DSM, DHM and MHM, splk-vol, splk-wlk, splk-flx, splk-fqm. User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge are dedicated tenant types. RBAC is per tenant through Splunk roles, Virtual Groups aggregate several tenants into one read-only card, and Saved Views let a team land on its own tenants.

The monitoring cycle



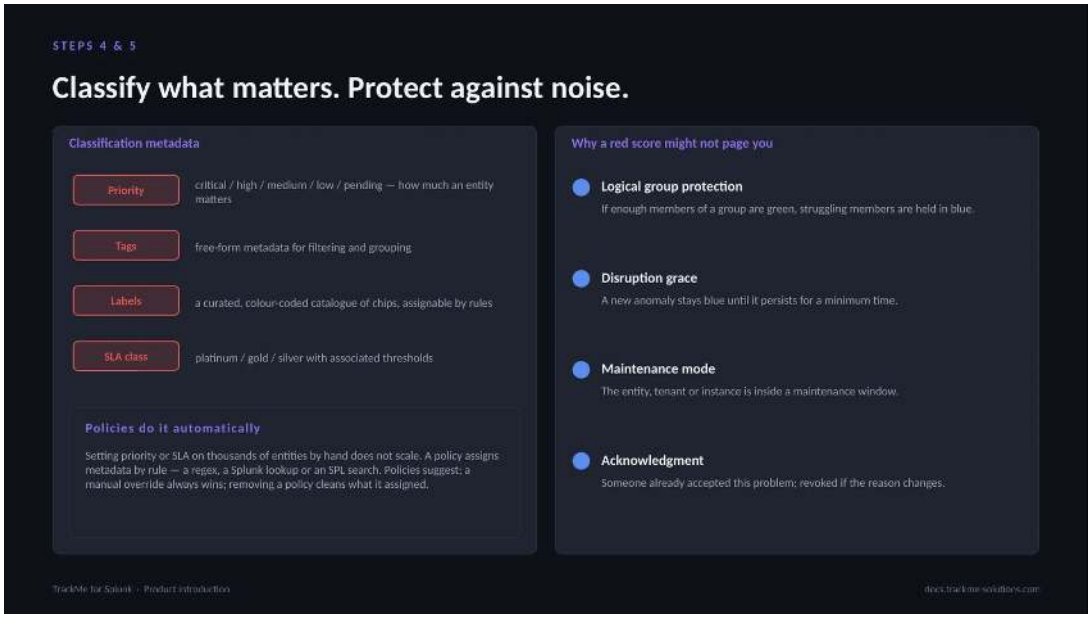
Each cycle, for every tenant: discover, measure, decide, classify, protect, alert. The deliberate separation between what state an entity is in, step 3, and whether to tell you, steps 5 and 6, is why TrackMe is quiet in practice despite tracking so much. Findings tenants run the same idea with their own jobs: collect, evaluate the policies, raise or recover findings, notify. Trackers are scheduled searches, typically every five minutes.

State and impact score



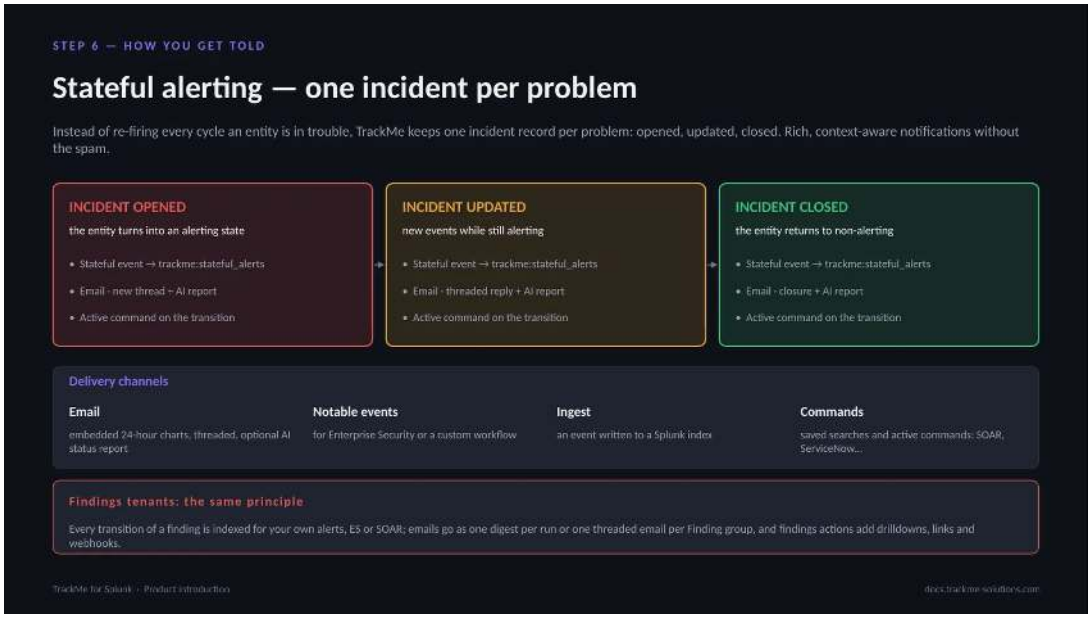
Every cycle each entity gets one of four colours. The state derives from a numeric impact score: TrackMe adds up everything currently wrong with an entity into one total score — 0 is green, below 100 orange, 100 and above red. Because every contribution is a number with a configurable weight, you control exactly what makes an entity critical: one major breach trips red, several small ones accumulate. Blue is not a fifth severity — it means "I see this, but I am not waking you up".

Classification and protection



Step 4, classification: priority, tags, labels and SLA are the four kinds of metadata attached to entities. Tags are free-form strings, labels come from a managed coloured catalogue. Three policy families assign them by rule — priority, tags and SLA. Step 5, protection: four mechanisms can hold an entity in blue instead of red. This separation between what state and whether to tell you is deliberate.

Stateful alerting



Instead of re-firing every cycle, TrackMe keeps one incident record per problem: it opens when the entity first goes into alert, updates while the problem persists, and closes when the entity recovers. On those transitions it can deliver an email with embedded charts and optionally an AI-written status report, write a notable event, ingest an event and run a command. Findings tenants follow the same principle: every transition of a finding is indexed, and email notifications go one per run or one thread per Finding group.

Topology Studio



Topology Studio is the canvas for a view of the environment: a payment flow, a data centre, an ingestion pipeline, a business service. On the left, a map of web services built from TrackMe entities, feeds, hosts and heartbeat checks, with their live state and KPIs. On the right, the flow topology of a Cribl Stream deployment filtered on its nginx Routes, with three Destinations coloured by their findings, ready to save as a live view.

A Splunk HEC tenant generates its own map in one click: pools, endpoints and tokens with live status and sparkline KPIs; a Splunk S2S tenant saves its data path and the Splunk UF Fleet and Splunk Forwarder Collectors tenants their fleet maps the same way. Views are shared under RBAC, aggregates roll many objects into one node, Topology Alerts notify on the health of a whole map, and the AI Assistant can build a view from a sentence. Available in every

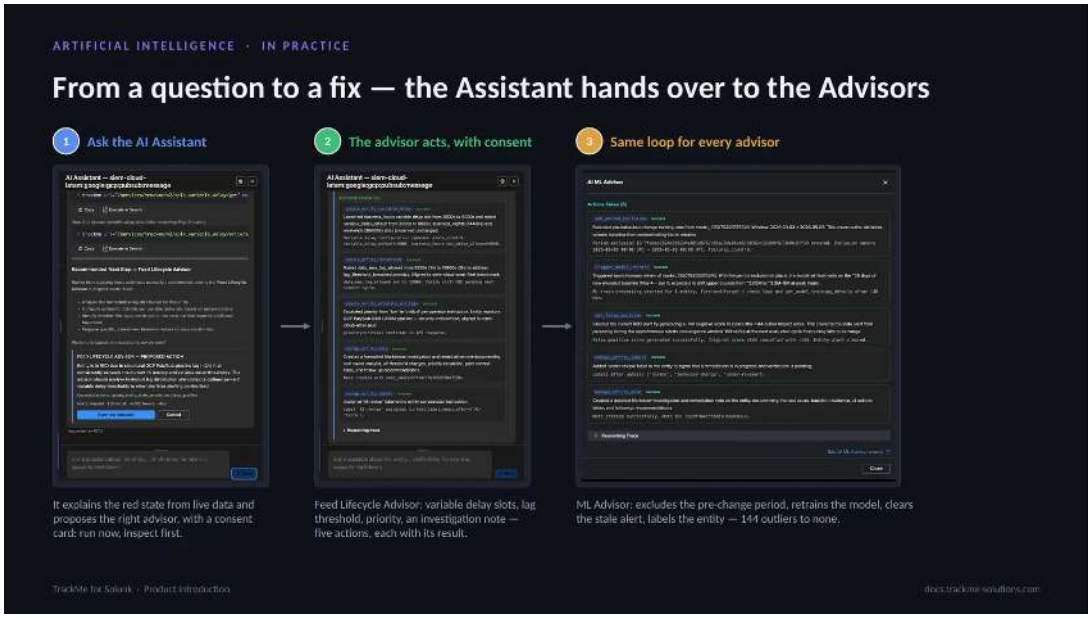
edition.

The AI layer



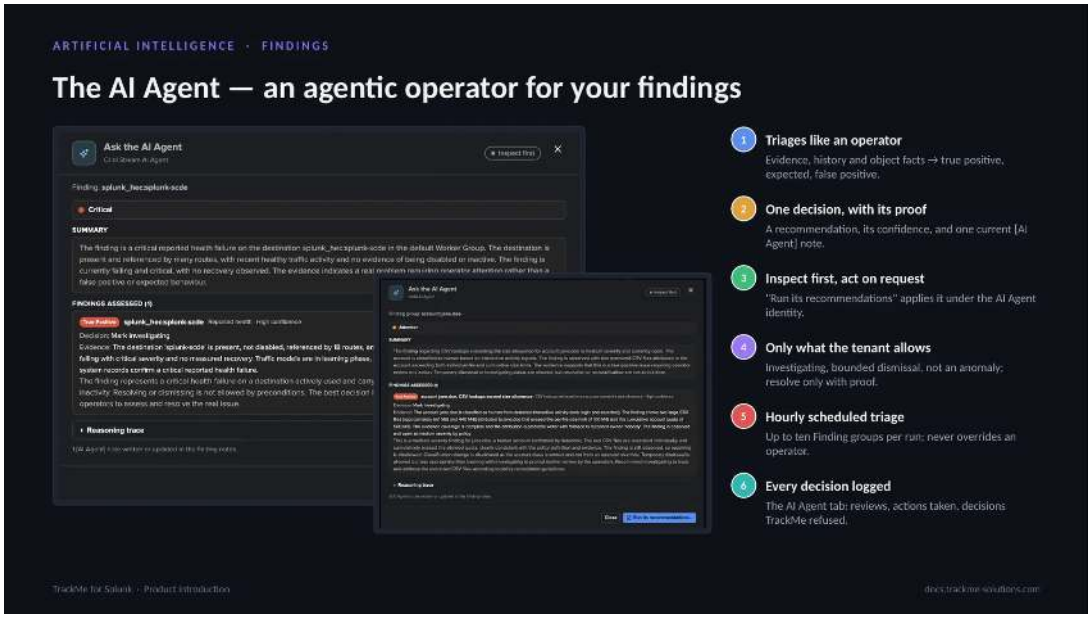
The AI layer helps operators understand and resolve what TrackMe detects, grounded in the live state of the environment and keeping the operator in the loop. It grew in four levels: the AI Assistant on every page; the AI Advisors and the Concierge, which inspect and, with consent, act; AI Agents automation, running the advisors unattended within guardrails; and AI Routines, your own standing instructions. For findings, the seven dedicated components share the AI Agent, an agentic operator that triages findings, applies the decisions the tenant allows and follows custom instructions. The whole layer is opt-in and inert until an administrator configures a provider.

From a question to a fix



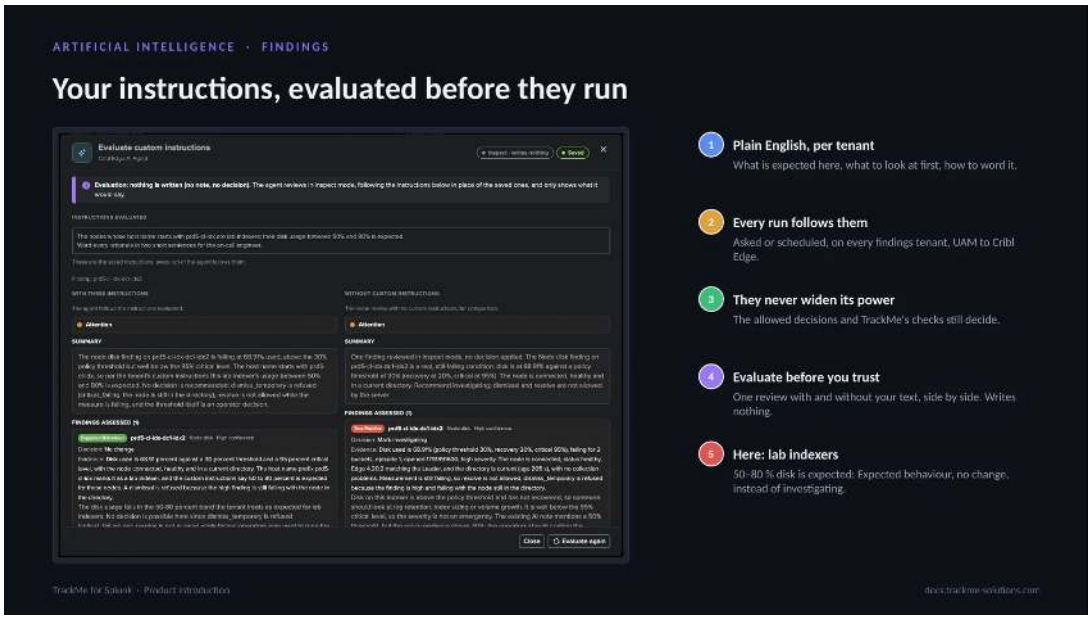
A feed is red and the operator asks the AI Assistant why. The Assistant reads the entity, explains the lag and, rather than guessing values, proposes the Feed Lifecycle Advisor with a consent card. The advisor inspects first; on request it acts: it widens the variable delay slots, raises the lag threshold, escalates the priority and documents everything in a note. Every advisor follows the same loop. The ML Advisor, facing a permanent change of baseline, excludes the old period, retrains the model, cancels the stale alert and labels the entity under review: 144 outliers become none. Each action is a typed tool call with its result, audited.

The AI Agent for findings



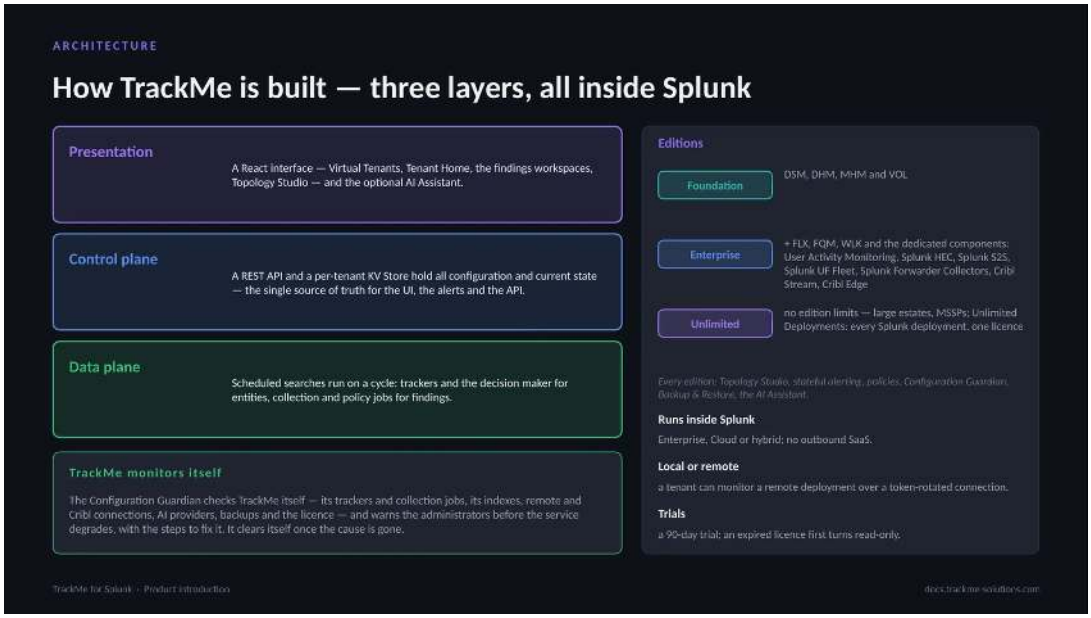
Each of the seven dedicated components, from User Activity Monitoring to Splunk Forwarder Collectors and Cribl Edge, has its own AI Agent on one shared framework. Asked about a finding, or during its hourly scheduled triage, it reads the evidence, the history and the facts about the object, assesses the finding and recommends one decision with its confidence. On the left, the Cribl Stream agent confirms a real health failure on a busy Destination; on the right, the UAM agent confirms oversized CSV lookups on a human account. In act mode it applies only what the tenant allows, resolves only with proof, never overrides an operator, keeps one [AI Agent] note per finding and logs every decision.

Custom instructions, evaluated



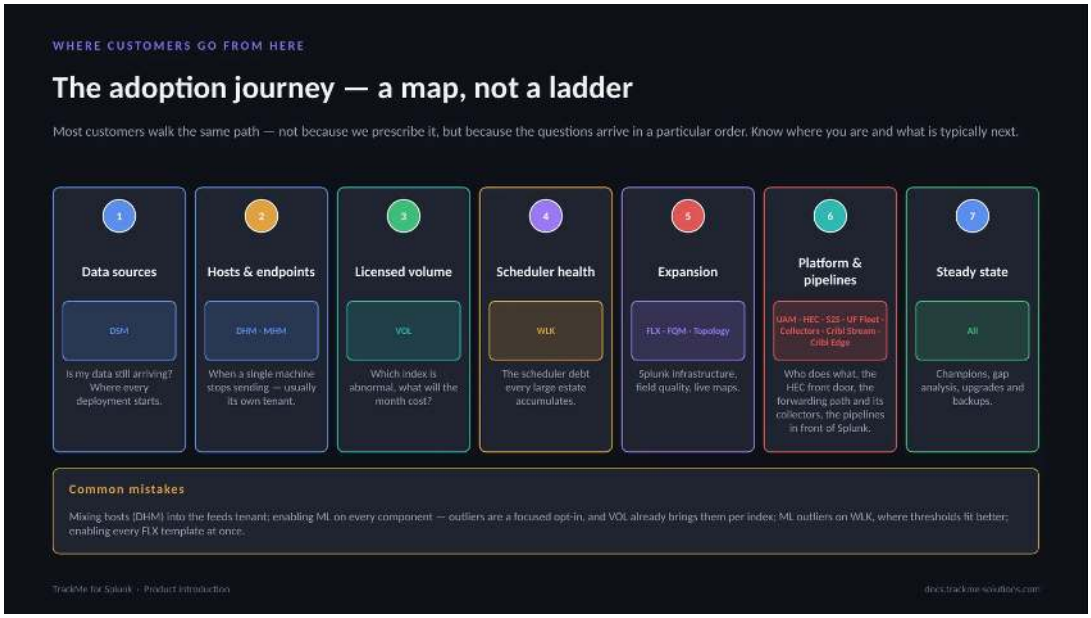
Administrators give the AI Agent of a findings tenant custom instructions in plain English: what to treat as expected, what to look at first, how to word its recommendations. Every run follows them, asked or scheduled, on every findings tenant, and they never widen what the agent may do. Evaluate runs the agent in inspect mode with the text, saved or not, and shows the same review with and without it, side by side, writing nothing. Here the Cribl Edge agent learns that lab indexers between 50 and 80 percent disk are expected: the same finding reads Expected behaviour instead of a true positive to investigate.

Architecture and editions



Three layers: the data plane, trackers and the decision maker or the findings jobs; the control plane, a REST API and per-tenant KV Store; the presentation, a React interface and the AI Assistant. Everything runs inside Splunk, and TrackMe monitors itself: the Configuration Guardian checks its own trackers, connections, indexes and licence and warns before the service degrades. Editions: Foundation covers DSM, DHM, MHM and VOL. Enterprise adds FLX, FQM and WLK and the dedicated components: User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge. Unlimited removes the edition limits; Unlimited Deployments covers every Splunk deployment under one licence. Topology Studio, stateful alerting, policies, the Configuration Guardian, Backup & Restore and the AI Assistant are in every edition. An expired licence first switches to read-only while monitoring keeps running.

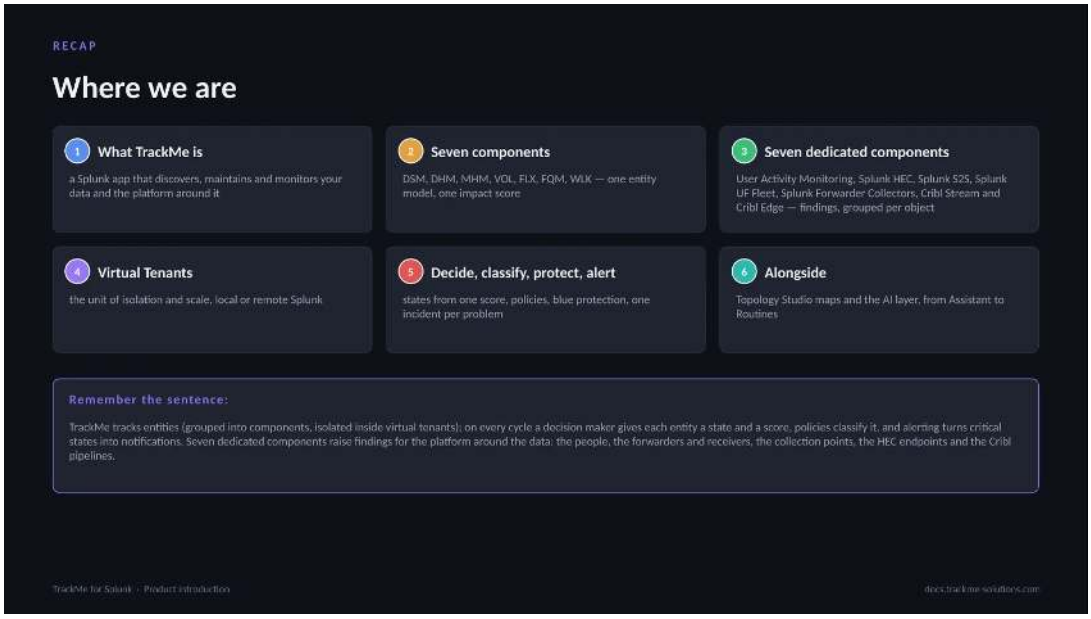
The adoption journey



This journey is a map, not a ladder: the questions arrive in a particular order and most teams are in several phases at once. Phase 1, data sources with DSM, is where every deployment starts. Phase 2 covers hosts and endpoints with DHM and MHM, usually in a dedicated tenant. Phase 3 is the licensed volume of every index with VOL. Phase 4 is the scheduler with WLK. Phase 5 expands with FLX, FQM and Topology Studio. Phase 6 goes beyond the data: who does what with User Activity Monitoring, the HTTP Event Collector with Splunk HEC, the forwarding path with

Splunk S2S and Splunk UF Fleet, the collection points with Splunk Forwarder Collectors, and the Cribl pipelines with Cribl Stream and Cribl Edge. Phase 7, the steady state, is a continuous practice.

Where we are



TrackMe is a Splunk app that discovers, maintains and monitors the availability and quality of your data and of the platform around it. Entity monitoring: seven components — DSM, DHM, MHM, VOL, FLX, FQM and WLK — share one entity model, with a state from one impact score, policies that classify, protection that holds noise in blue and stateful alerting. Findings monitoring: seven dedicated components — User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge — raise findings, grouped per object and decided once. Everything is isolated in virtual tenants, with Topology Studio and the AI layer alongside.

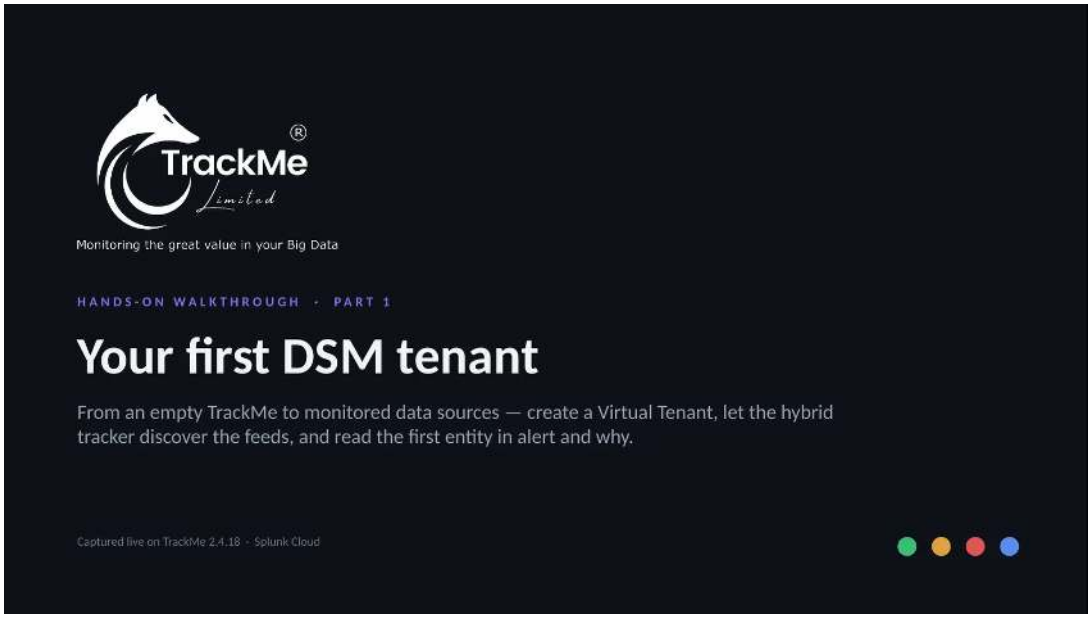
2.1.2 Part 1 — Your first DSM tenant

From an empty TrackMe to monitored data sources: create a Virtual Tenant, let the hybrid tracker discover the feeds, and read the first entity in alert and why.

`Download the deck (PDF) · PowerPoint`

about 20 minutes · 16 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

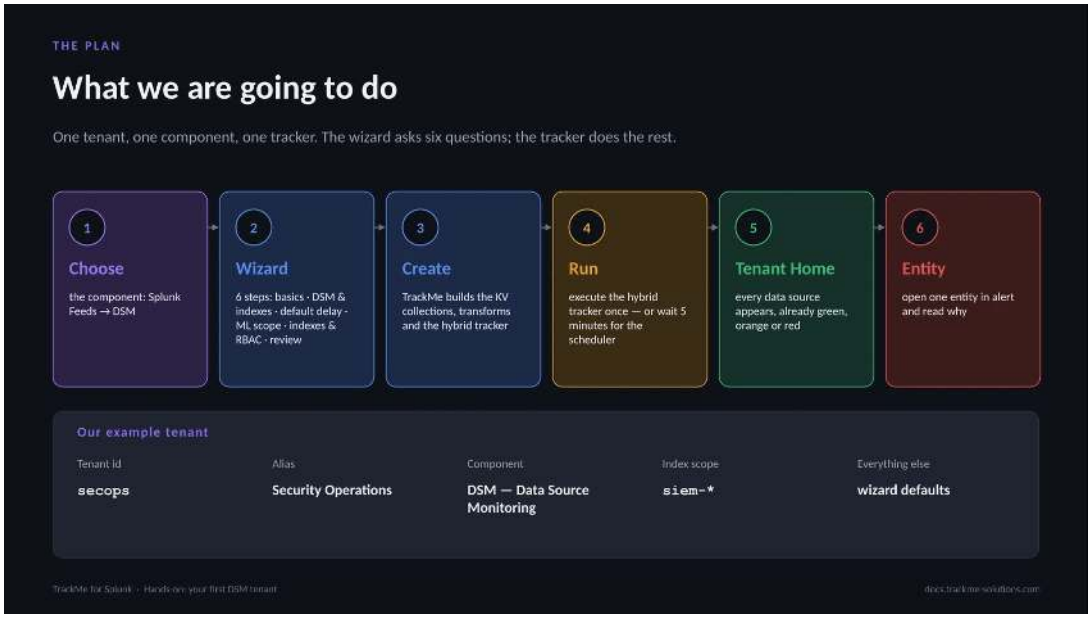
Overview



This walkthrough goes from an empty TrackMe to monitored data sources: a Virtual Tenant is created, the hybrid tracker discovers the feeds, and the first entity in alert is read together with the reason for its state. Everything shown was captured on TrackMe 2.4.18 on Splunk Cloud, starting from a tenant that did not exist.

The aim is to show the product doing what the introduction described, discover, measure and decide, on a fresh tenant with no configuration beyond an index pattern.

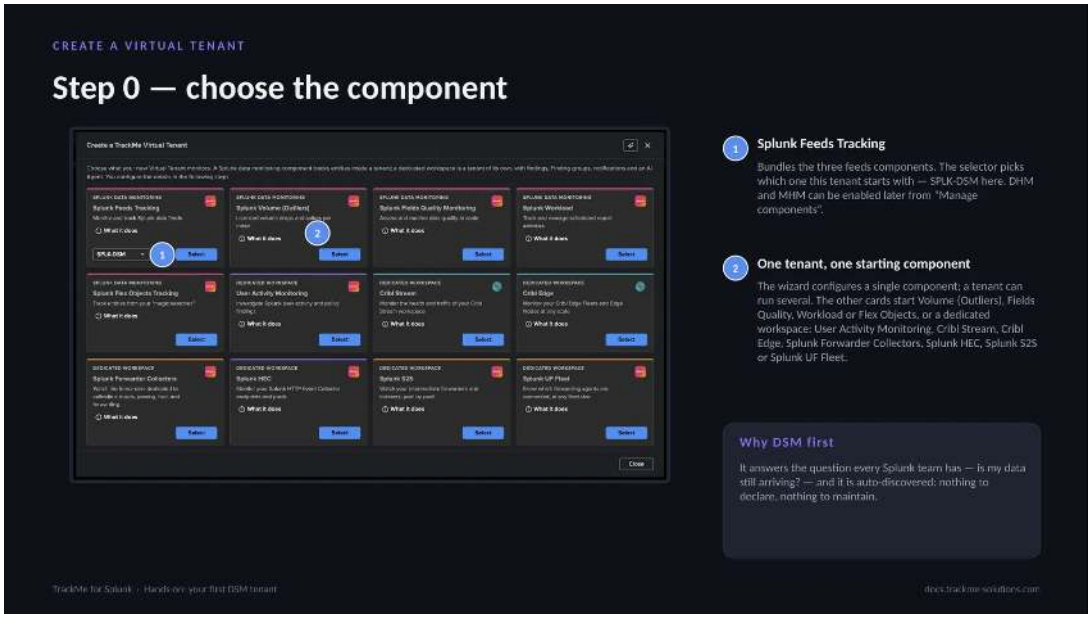
Six steps and the example tenant



One tenant, one component, one tracker. Callouts 1 to 6 lay out the path: Choose the component (Splunk Feeds, then DSM); Wizard, six steps covering basics, DSM and indexes, default delay, ML scope, indexes and RBAC, and review; Create, where TrackMe builds the KV collections, transforms and the hybrid tracker; Run, executing the tracker once or waiting 5 minutes for the scheduler; Tenant Home, where every data source appears already green, orange or red; and Entity, opening one entity in alert to read why.

The example tenant uses the id secops, the alias Security Operations, the component DSM (Data Source Monitoring) and the index scope siem-*. Everything else keeps the wizard defaults, each revisitable from the Tenant Home later.

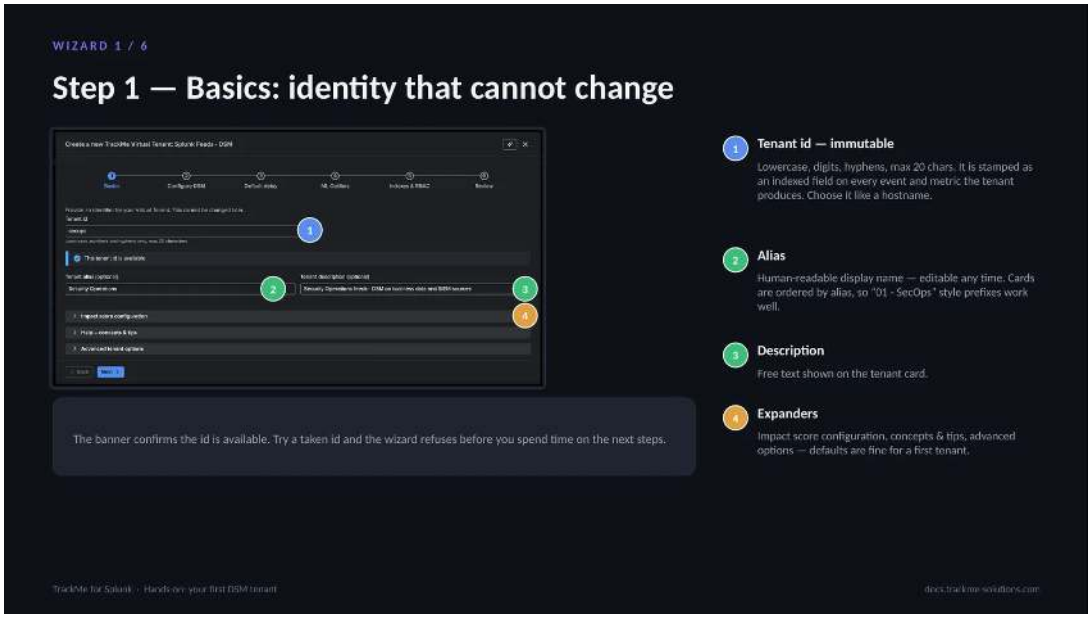
The tenant type chooser



The chooser opens from Virtual Tenants, Actions, Create a tracking tenant, with twelve cards: the Splunk data monitoring components, then the dedicated workspaces. Splunk Feeds Tracking (callout 1) bundles the three feeds components, DSM (index and sourcetype), DHM (host and sourcetype) and MHM (metric hosts); its selector picks the starting component, SPLK-DSM here, and DHM and MHM can be enabled later from Manage components. The other cards start Volume (Outliers), Fields Quality, Workload or Flex Objects tenants, or a dedicated workspace: User Activity Monitoring, Cribl Stream, Cribl Edge, Splunk Forwarder Collectors, Splunk HEC, Splunk S2S or Splunk UF Fleet.

The wizard configures a single component, although a tenant can run several (callout 2). DSM comes first because it answers whether the data is still arriving and is auto-discovered: nothing to declare, nothing to maintain.

Tenant id, alias and description

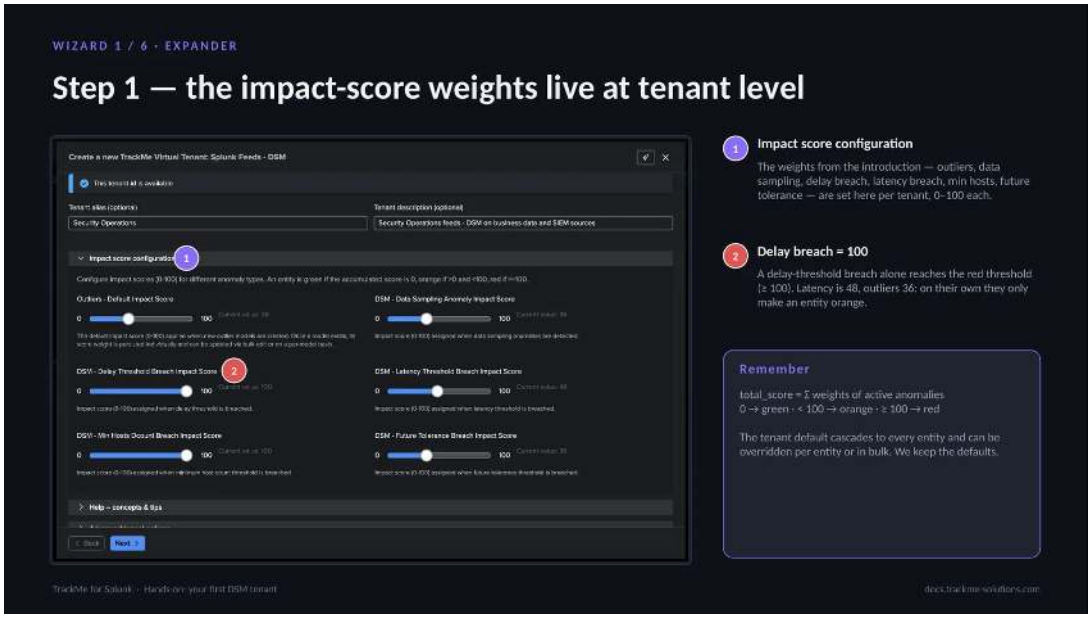


Step 1 of 6 sets identity. The Tenant id (callout 1) is immutable: lowercase, digits and hyphens, 20 characters at most. It is stamped as an indexed field on every event the tenant produces and is part of every KV collection and report name, so it should be chosen like a hostname. A banner confirms the id is available; a taken id is refused immediately.

The Alias (callout 2) is a human-readable display name, editable any time; cards are ordered by alias, so prefixes like

01 - SecOps work well. The Description (callout 3) is free text shown on the tenant card. The Expanders (callout 4) hold Impact score configuration, concepts and tips, and advanced options.

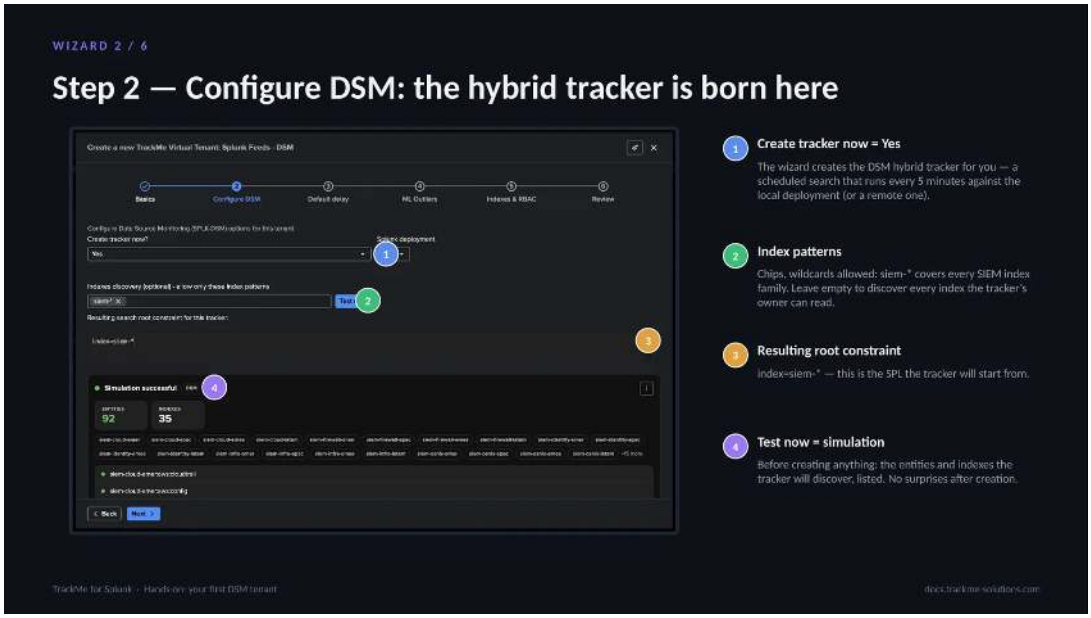
Impact score weights per tenant



The Impact score configuration expander (callout 1) sets the weights of the scoring model, outliers, data sampling, delay breach, latency breach, min hosts and future tolerance, per tenant, from 0 to 100 each.

Delay breach carries 100 (callout 2), so a delay-threshold breach alone reaches the red threshold: a feed that stopped is by definition critical. Latency is 48 and outliers 36; on their own they only make an entity orange. The rule is total_score equals the sum of the weights of active anomalies: 0 is green, below 100 orange, 100 and above red. The tenant default cascades to every entity and can be overridden per entity or in bulk.

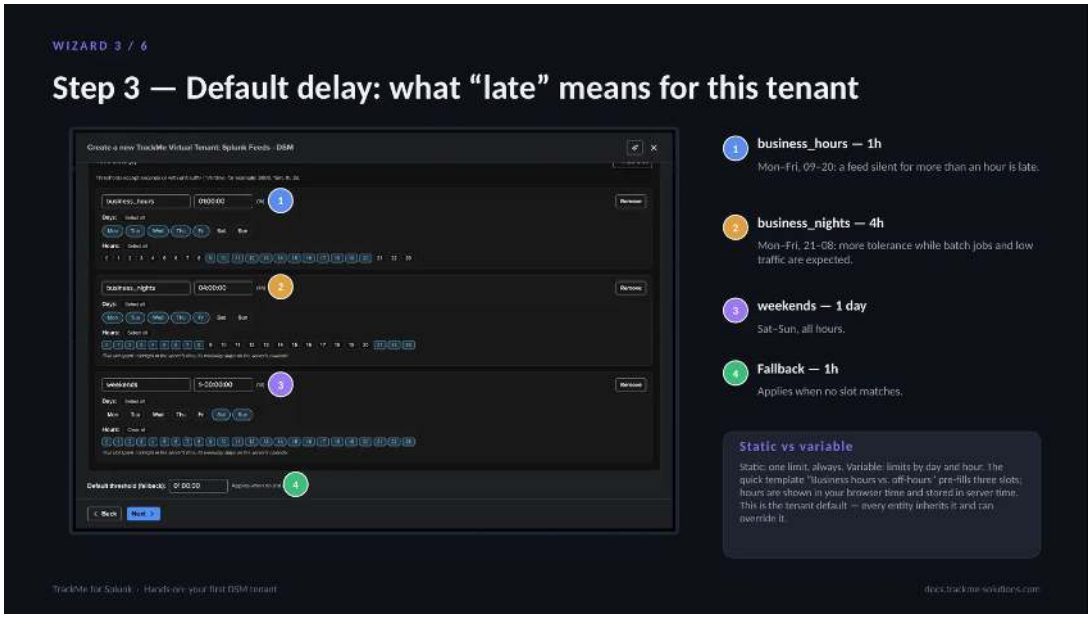
Discovery scope and tracker creation



Step 2 of 6 scopes discovery and creates the tracker. Create tracker now set to Yes (callout 1) makes the wizard create the DSM hybrid tracker, a scheduled search that runs every 5 minutes against the local deployment or a remote one. Index patterns (callout 2) are chips with wildcards allowed: siem-* covers every SIEM index family, and an empty list discovers every index the tracker's owner can read.

The Resulting root constraint (callout 3) shows the SPL the tracker starts from, index=siem-*. Test now (callout 4) runs the discovery search as a simulation before anything is created, listing the entities and indexes the tracker would discover. It is the fastest way to validate index patterns and permissions before committing.

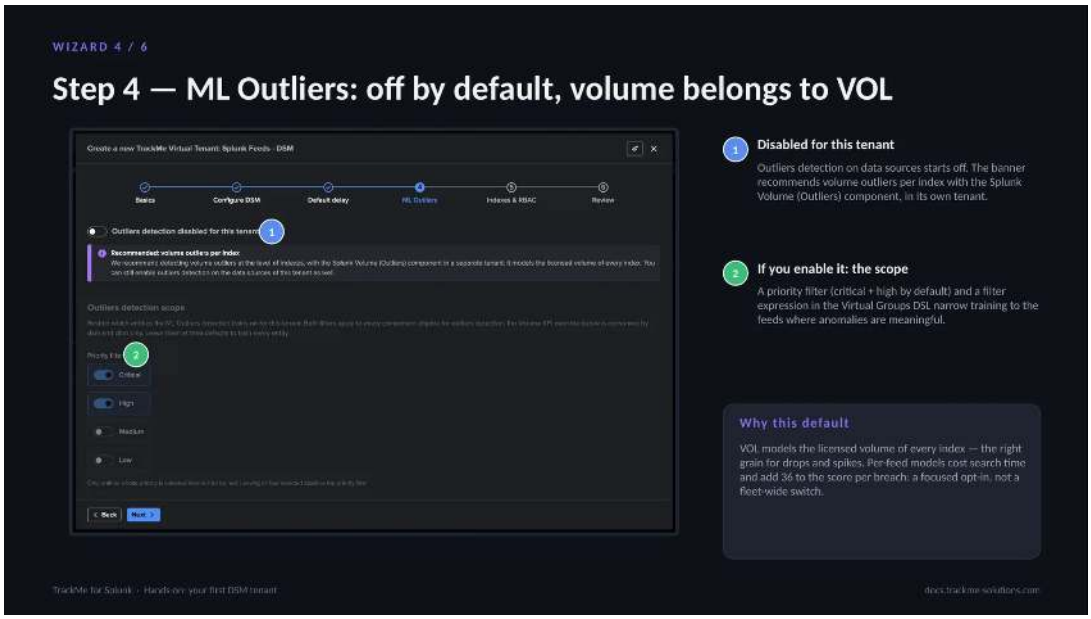
Variable delay slots



Delay is the core DSM metric: the time since the last event was indexed, compared to a threshold. A static threshold is one limit always; a variable one sets limits by day and hour. The quick template Business hours vs. off-hours pre-fills three slots: business_hours (callout 1), Monday to Friday 09 to 20, 1h; business_nights (callout 2), Monday to Friday 21 to 08, 4h; weekends (callout 3), Saturday and Sunday, 1 day. The Fallback (callout 4), 1h, applies when no slot matches.

Hours are shown in browser time and stored in server time. Every entity inherits this tenant default and can override it, which keeps TrackMe quiet at night and on weekends without a rule per feed.

ML outliers scope and the VOL recommendation

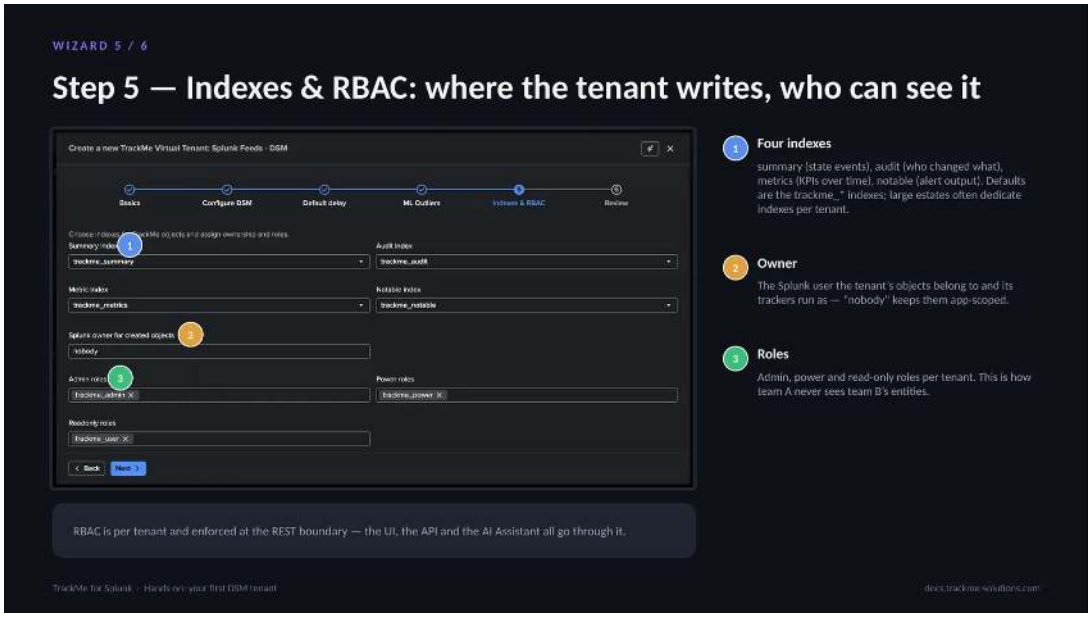


Step 4 of 6 covers ML outliers. Outliers detection on data sources starts disabled for the tenant (callout 1), and the banner recommends volume outliers per index with the Splunk Volume (Outliers) component in its own tenant.

Nothing needs changing for a first tenant.

When enabled, the scope (callout 2) narrows training to the feeds where anomalies are meaningful: a priority filter, critical and high by default, and a filter expression in the Virtual Groups DSL. VOL models the licensed volume of every index, the right grain for drops and spikes, whereas per-feed models cost search time and add 36 to the score per breach. Outliers are a focused opt-in, not a fleet-wide switch.

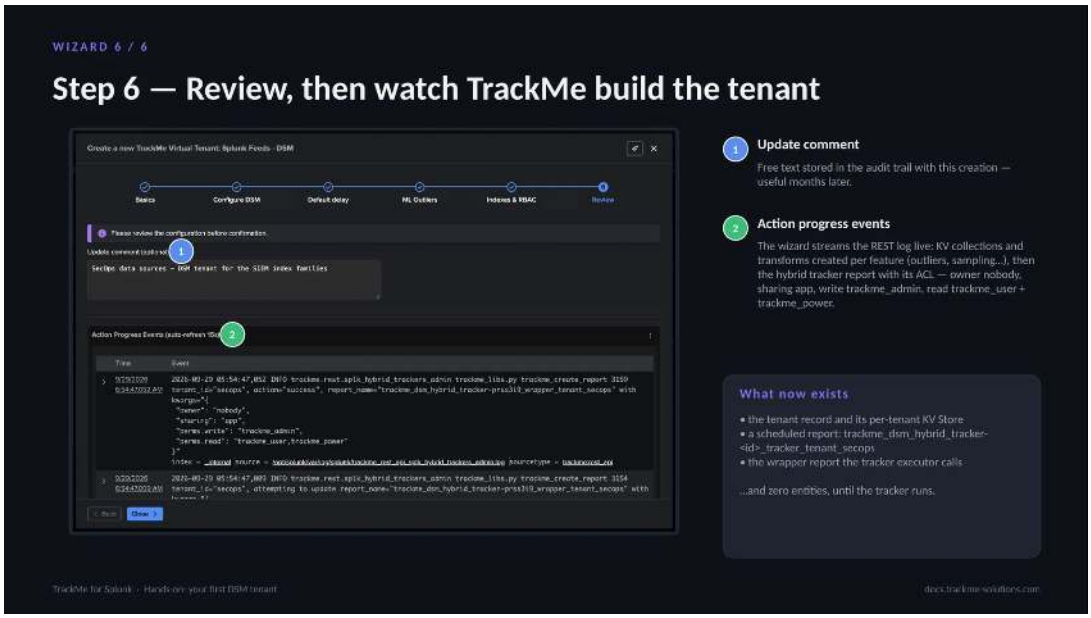
Tenant indexes, owner and roles



Step 5 of 6 sets where the tenant writes and who sees it. Four indexes (callout 1) receive its output: summary for state events, audit for who changed what, metrics for KPIs over time and notable for alert output. The defaults are the trackme_* indexes; large estates often dedicate indexes per tenant.

The Owner (callout 2) is the Splunk user the tenant's objects belong to and its trackers run as; nobody keeps them app-scoped. Roles (callout 3) assign admin, power and read-only Splunk roles per tenant, so one team never sees another's entities. RBAC is per tenant and enforced at the REST boundary: the UI, the API and the AI Assistant all go through it.

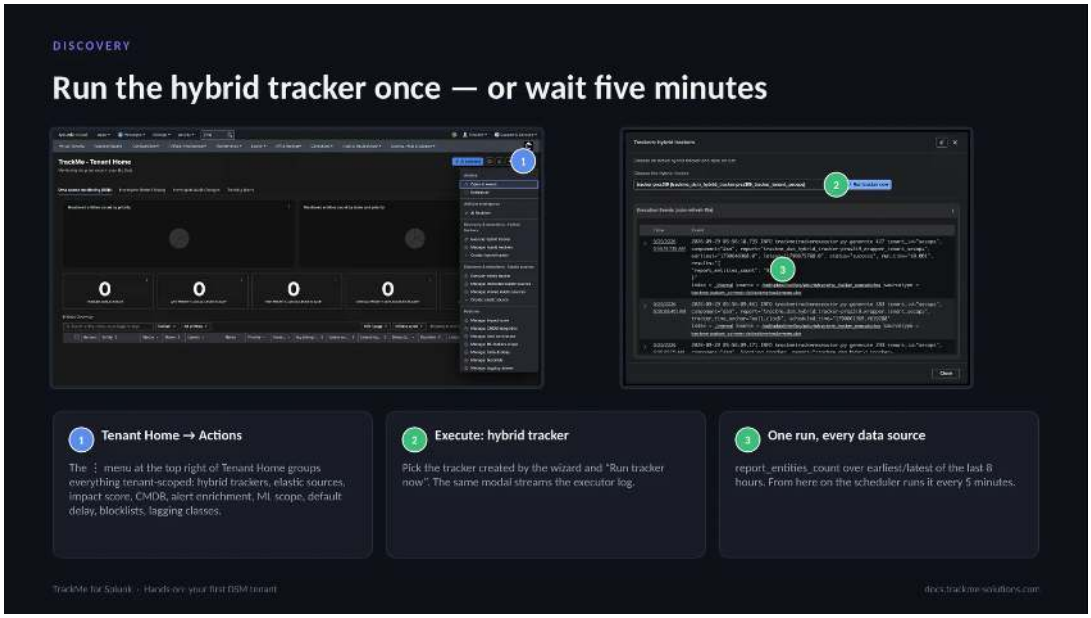
Creation log and resulting objects



Step 6 of 6 reviews and creates the tenant. The Update comment (callout 1) is free text stored in the audit trail with this creation. The Action progress events panel (callout 2) streams the REST log live, the same log an administrator would read in the TrackMe REST API logs: KV collections and transforms created per feature (outliers, sampling and so on), then the hybrid tracker report with its ACL, owner nobody, sharing app, write trackme_admin, read trackme_user and trackme_power. Creation takes seconds.

What now exists: the tenant record and its per-tenant KV Store, a scheduled report named trackme_dsm_hybrid_tracker-<id>_tracker_tenant_secops, and the wrapper report the tracker executor calls. There are zero entities until the tracker runs.

Executing the tracker from Tenant Home



Nothing appears in Tenant Home until a tracker has run, because TrackMe is not a live dashboard. Rather than waiting for the 5-minute schedule, the tracker is executed from Tenant Home, Actions (callout 1): the menu at the top right groups everything tenant-scoped, hybrid trackers, elastic sources, impact score, CMDB, alert enrichment, ML scope, default delay, blocklists and lagging classes.

Execute: hybrid tracker (callout 2) picks the tracker created by the wizard and offers Run tracker now; the same modal streams the executor log. One run covers every data source (callout 3): the panel reports report_entities_count over the last 8 hours. From here on the scheduler runs it every 5 minutes.

Health strip, donuts and entities table



Four tabs run across the top of Tenant Home: the component view, status flipping, audit changes and tracking alerts. The Health strip (callout 1) counts entities per state, computed by the decision maker on the tracker run just made. The State by priority donut (callout 2) shows everything at medium priority: discovery does not yet know what matters, which is what policies bring.

The Single-stat drilldowns (callout 3) and every state in the health strip filter the table on click; every number on the page is a filter. The Entities table (callout 4) has one row per (index, sourcetype) with state, score, labels, priority, lag (delay and latency), last event, last ingest, thresholds, outliers and sampling.

Score, lag and thresholds in a row

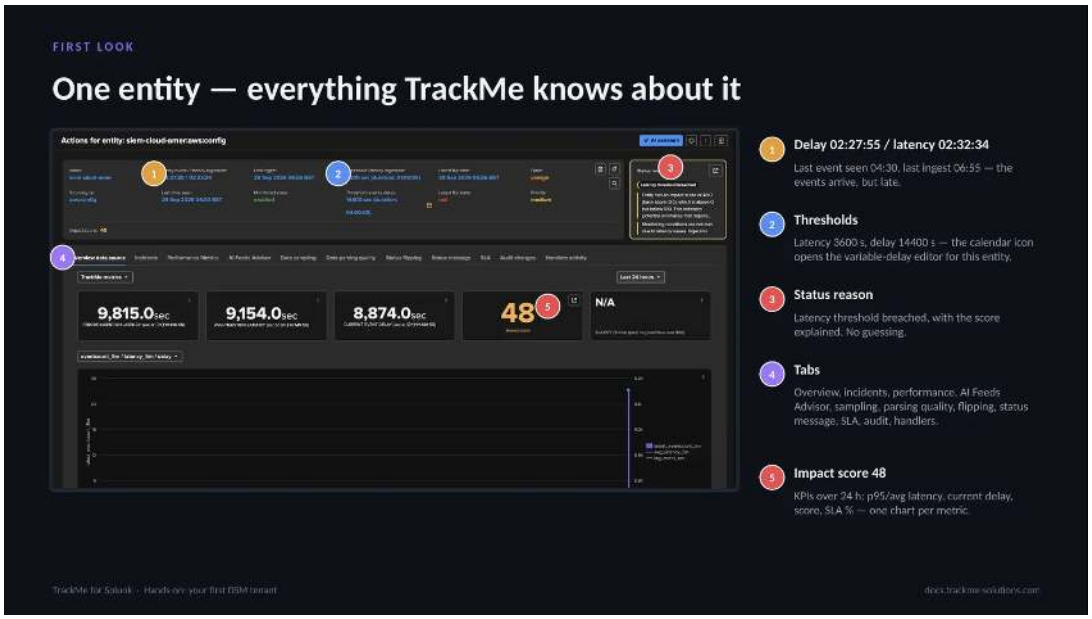


With the table filtered on orange from the health strip and grouped by index (callout 4), one row reads siem-cloud-amer:aws:config, orange, score 48, medium. Score 48 (callout 1) means latency only: 48 is below 100, hence orange. Adding a delay breach (100) would turn the same entity red at 148, additive exactly as configured in step 1. Sorting by score shows at a glance how many anomalies overlap.

The lag column (callout 2) shows delay and latency, 02:26:19 / 02:32:34: the last event is 2h26 old and was indexed 2h32 after its timestamp, so latency is what breached. The inline thresholds (callout 3) show Delay 14400 with

duration 04:00:00, the business_nights slot applied at run time, and Latency 3600.

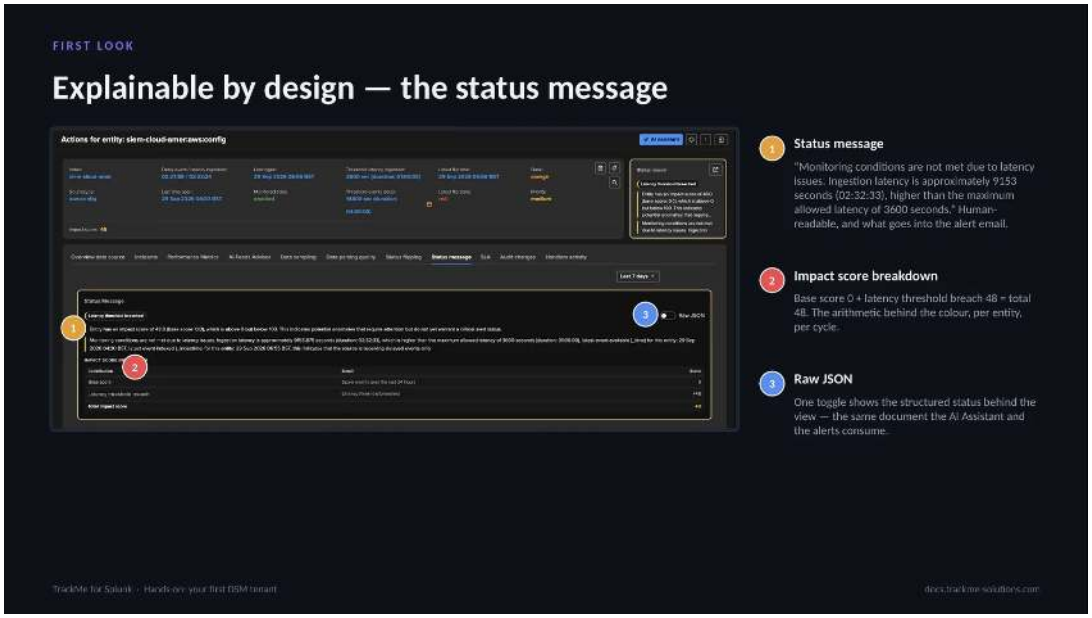
The entity view



The view is the same for every component; only the KPIs differ. The header (callout 1) is the decision maker's output: Delay 02:27:55 and latency 02:32:34, last event seen at 04:30, last ingest at 06:55, so the events arrive, but late. Thresholds (callout 2) show Latency 3600 s and delay 14400 s; the calendar icon opens the variable-delay editor. The Status reason (callout 3) states that the latency threshold is breached.

The Tabs (callout 4) are Overview, incidents, performance, AI Feeds Advisor, sampling, parsing quality, flipping, status message, SLA, audit and handlers. Impact score 48 (callout 5) heads the KPIs over 24 h, p95 and average latency, delay, score and SLA %, one chart per metric.

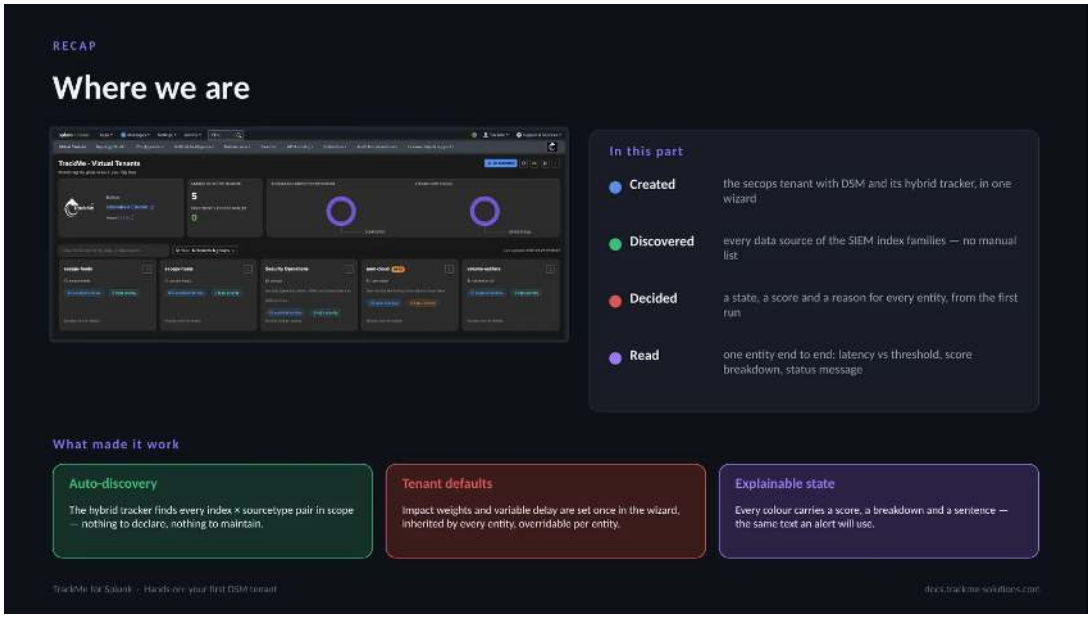
Status message and score breakdown



The Status message (callout 1) is human-readable and is the text that goes into the alert email: Monitoring conditions are not met due to latency issues. Ingestion latency is approximately 9153 seconds (02:32:33), higher than the maximum allowed latency of 3600 seconds. The Impact score breakdown (callout 2) gives the arithmetic behind the colour: base score 0 plus latency threshold breach 48 equals total 48.

The Raw JSON toggle (callout 3) shows the structured status behind the view, the same document the AI Assistant and the alerts consume. Nothing in TrackMe changes state without a reason string and a score breakdown. The 7-day timeline below is how flapping is spotted over weeks.

Where we are



In this part the secops tenant was created with DSM and its hybrid tracker in one wizard, every data source of the SIEM index families was discovered without a manual list, and every entity received a state, a score and a reason from the first run.

Three things made it work. Auto-discovery: the hybrid tracker finds every index and sourcetype pair in scope, nothing to declare or maintain. Tenant defaults: impact weights and variable delay are set once in the wizard, inherited by every entity and overridable. Explainable state: every colour carries a score, a breakdown and a sentence, the same text an alert uses. Every entity is still medium priority; classification is what policies bring.

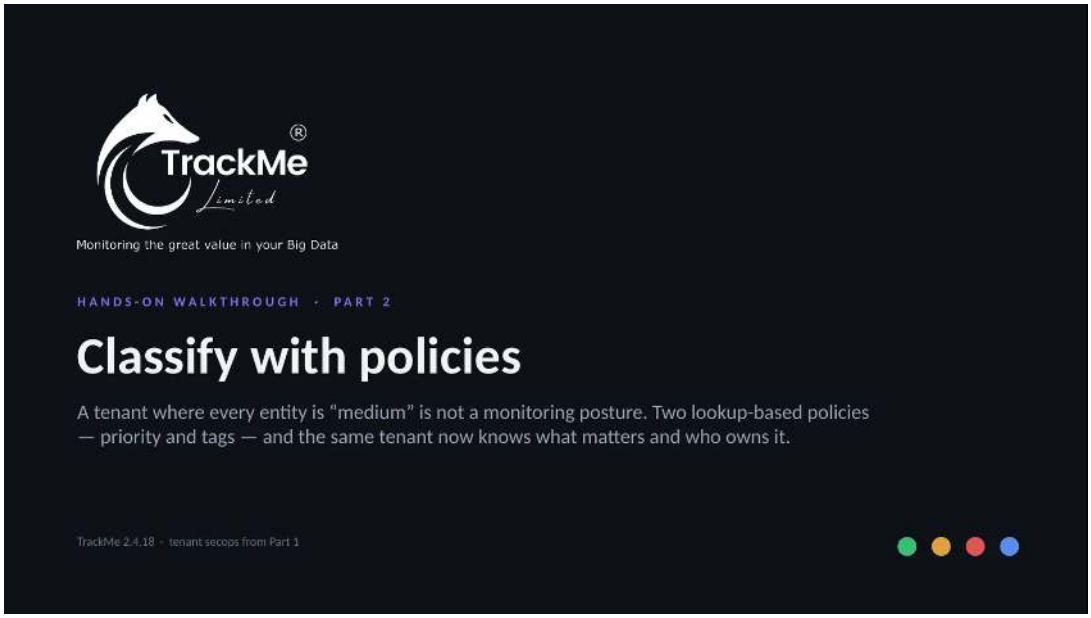
2.1.3 Part 2 — Classify with policies

Two lookup-based policies — priority and tags — simulated before saving, and the same tenant now knows what matters and who owns it.

`Download the deck (PDF) · PowerPoint`

about 15 minutes · 12 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

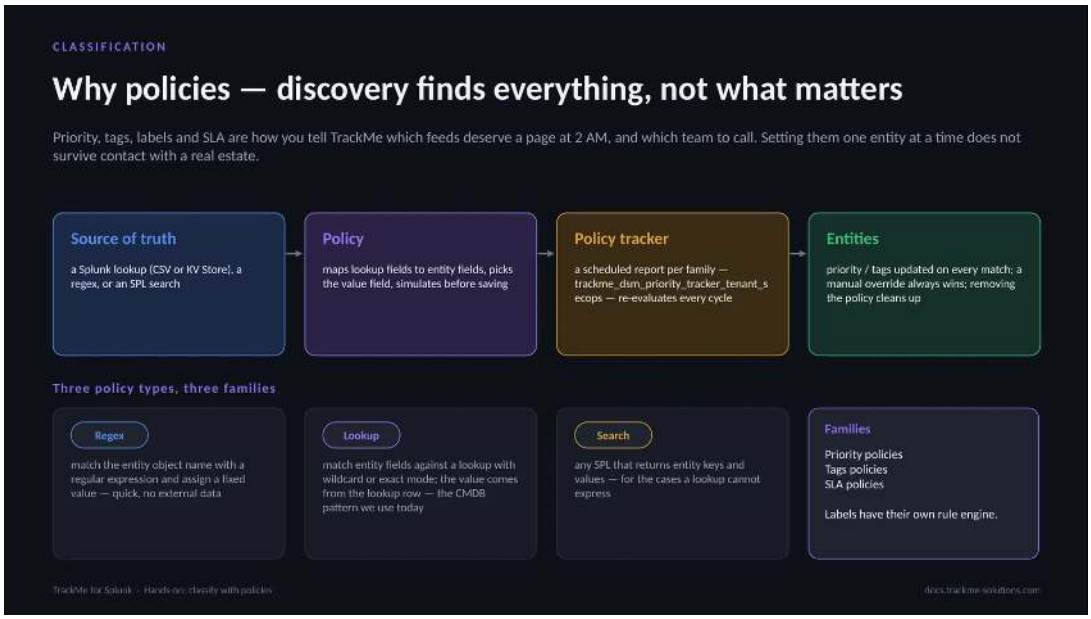
Overview



Part 2 of the hands-on walkthrough starts where Part 1 stopped: the secops tenant has its DSM entities, every one of them at medium priority and none tagged. A tenant where every entity is medium is not a monitoring posture — TrackMe cannot yet tell which feeds matter or who owns them.

This part covers step 4 of the cycle, CLASSIFY, done the way it scales: two lookup-based policies, priority and tags, driven by a CMDB-style lookup rather than set one entity at a time. Once applied, the same tenant knows what matters and which team to call. Captured on TrackMe 2.4.18.

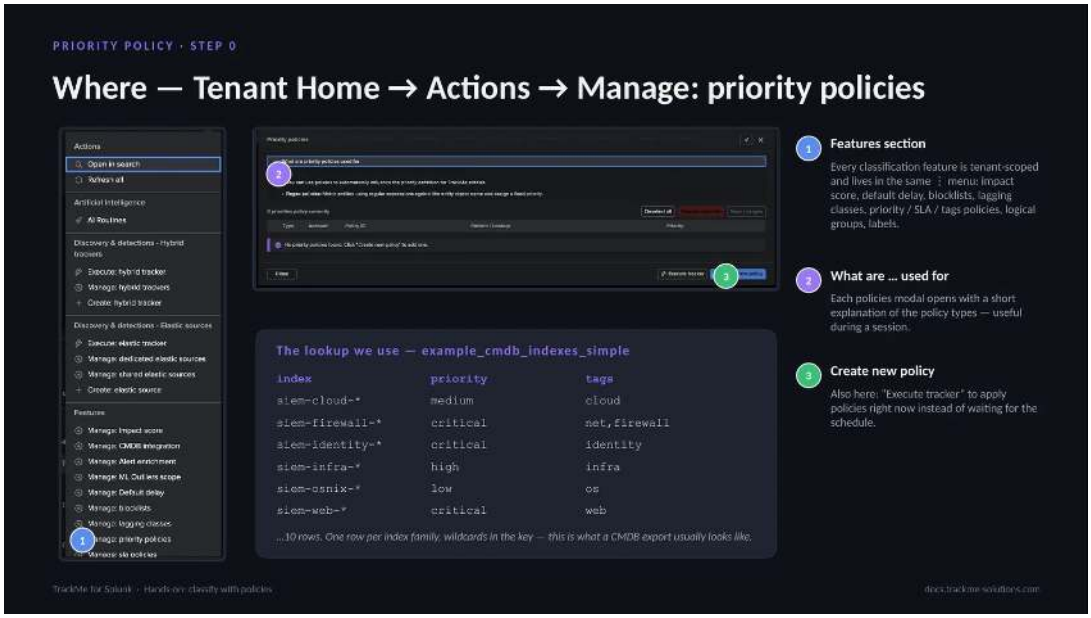
Policy types and policy families



Priority, tags, labels and SLA tell TrackMe which feeds matter and which team to call; setting them one entity at a time does not scale. A policy assigns them by rule: a source of truth (a Splunk lookup in CSV or KV Store, a regex, or an SPL search) feeds a policy that maps lookup fields to entity fields, picks the value field and simulates before saving; a policy tracker, one scheduled report per family such as trackme_dsm_priority_tracker_tenant_secops, re-evaluates every cycle.

Three types (Regex, Lookup, Search) serve three families (Priority, Tags, SLA); labels have their own rule engine. A manual override always wins, removing a policy cleans up, and the higher priority wins when several match.

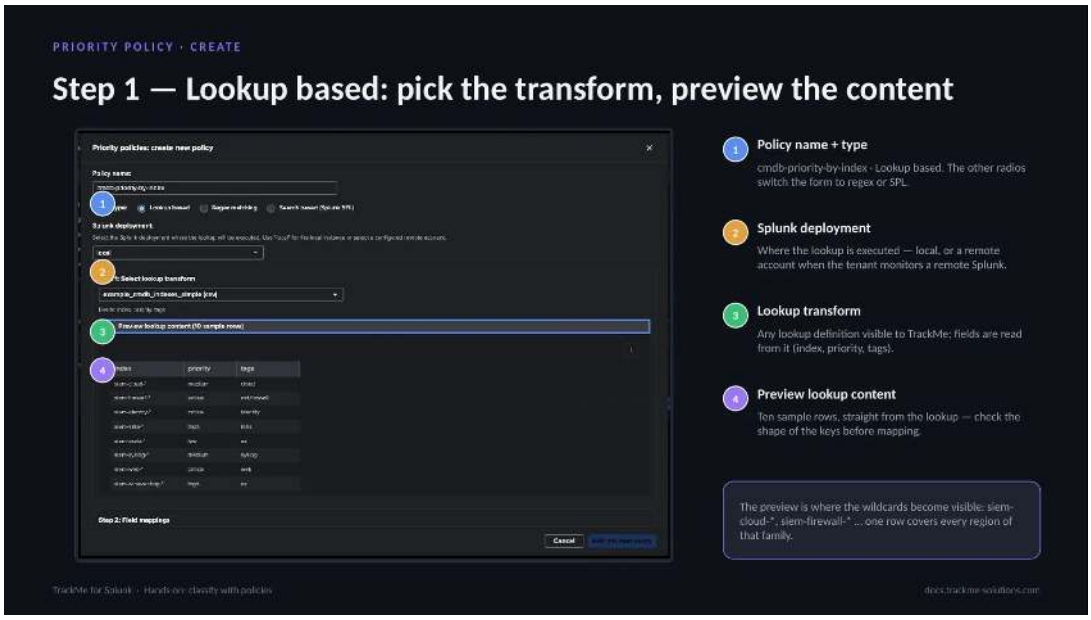
Locating the policies in Tenant Home



Every classification feature is tenant-scoped and sits in the Features section of the ⋮ Actions menu on Tenant Home (callout 1): impact score, default delay, blocklists, lagging classes, priority / SLA / tags policies, logical groups and labels. Each policies modal opens with a "What are … used for" explanation (callout 2). Create new policy starts the wizard; Execute tracker applies the policies immediately, without waiting for the schedule (callout 3).

The lookup used here, example_cmdb_indexes_simple, ships with TrackMe and is shaped like a CMDB export: an index pattern with wildcards, a priority and comma-separated tags. Ten rows, one per index family. A deployment replaces it with its own CSV or KV Store without changing the policy definition.

Choosing the lookup transform

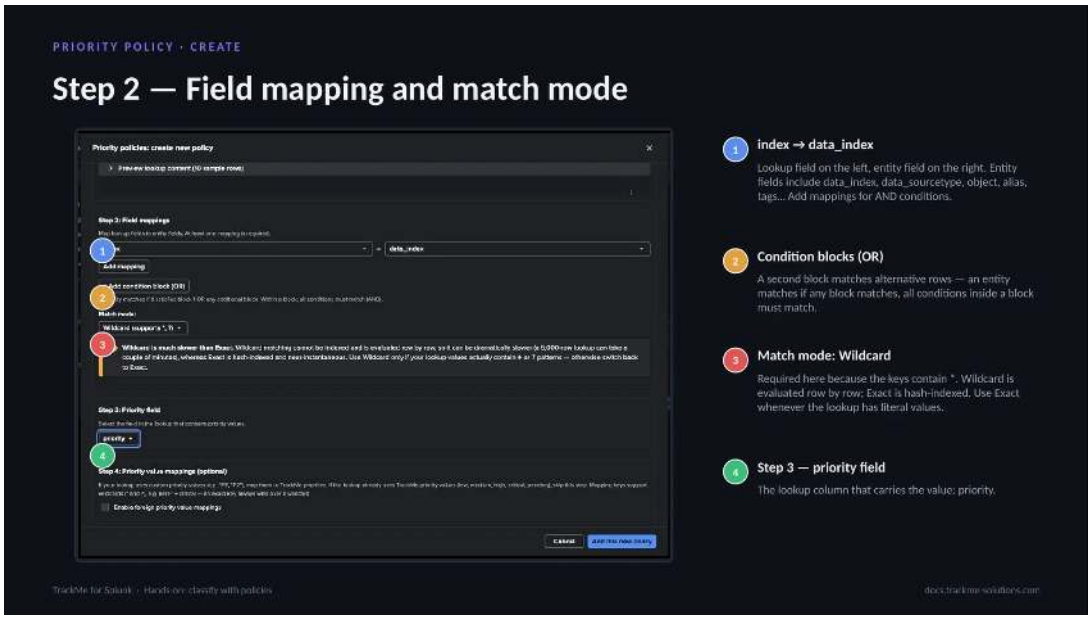


Step 1 of the wizard. Callout 1 names the policy, cmdb-priority-by-index, and selects the type, Lookup based; the other radio buttons switch the form to regex or SPL. Callout 2, Splunk deployment, sets where the lookup runs — local, or a remote account when the tenant monitors a remote Splunk. Callout 3, Lookup transform, lists every lookup definition visible to TrackMe and reads its fields (index, priority, tags); the filter box narrows a long list.

Callout 4, Preview lookup content, shows ten sample rows straight from the lookup. The preview reveals whether the key column contains wildcards such as siem-cloud-*, one row covering every region of a family, which decides the

match mode on the next step.

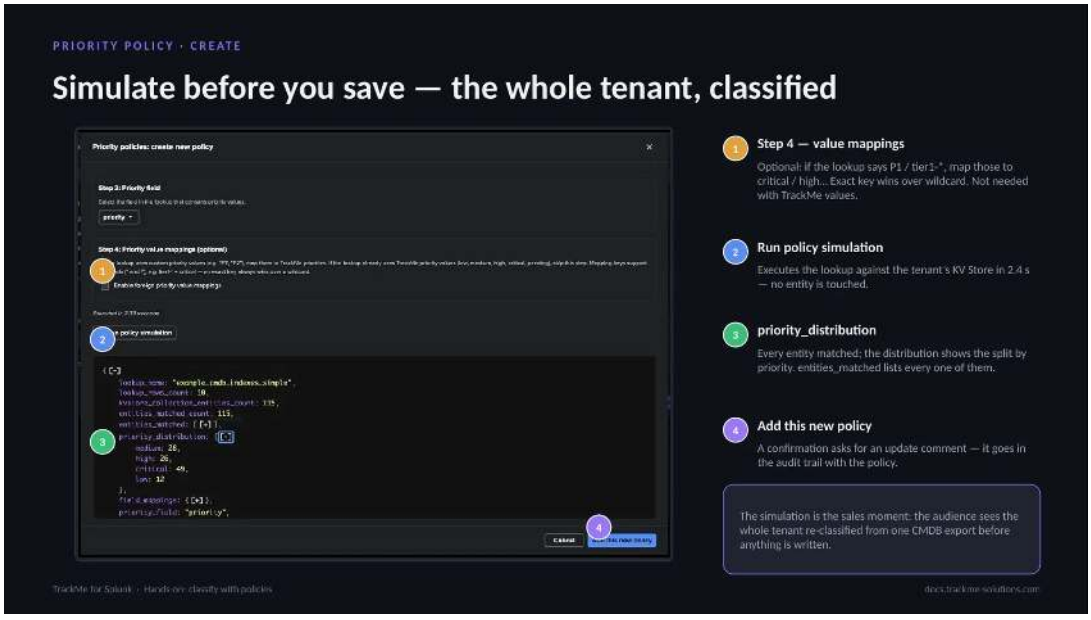
Mapping lookup fields to entity fields



The mapping decides which lookup column is compared with which entity attribute. Callout 1 maps index → data_index; entity fields include data_index, data_sourcetype, object, alias and tags, and additional mappings form AND conditions. Callout 2, Condition blocks (OR), adds a block for alternative rows: an entity matches if any block matches, all conditions in a block must match.

Callout 3 sets Match mode: Wildcard, required because the keys contain *. Wildcard is evaluated row by row while Exact is hash-indexed; a 5,000-row lookup checked this way against tens of thousands of entities takes minutes, so Exact is preferred whenever the lookup holds literal values. Callout 4, Step 3, selects priority as the lookup column carrying the value.

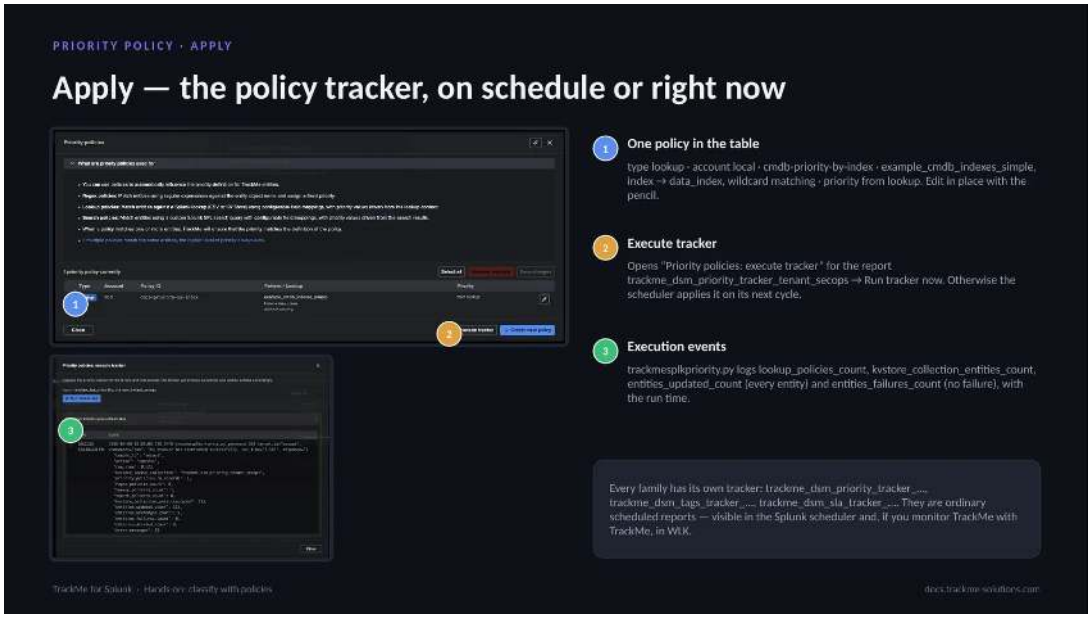
Running the policy simulation



Step 4, value mappings (callout 1), is optional: foreign lookup values (P1, tier1-*) are mapped onto TrackMe priorities, an exact key winning over a wildcard; it is not needed with TrackMe values. Run policy simulation (callout 2) executes the lookup against the tenant's KV Store — 2.4 s here — without touching any entity.

The result (callout 3) is the JSON structure the policy tracker applies: priority_distribution shows the split by priority and entities_matched lists every entity. Every entity matched; a split that looked wrong, such as everything medium, would point at the value field or the match mode. Add this new policy (callout 4) asks for an update comment, recorded in the audit trail with the policy.

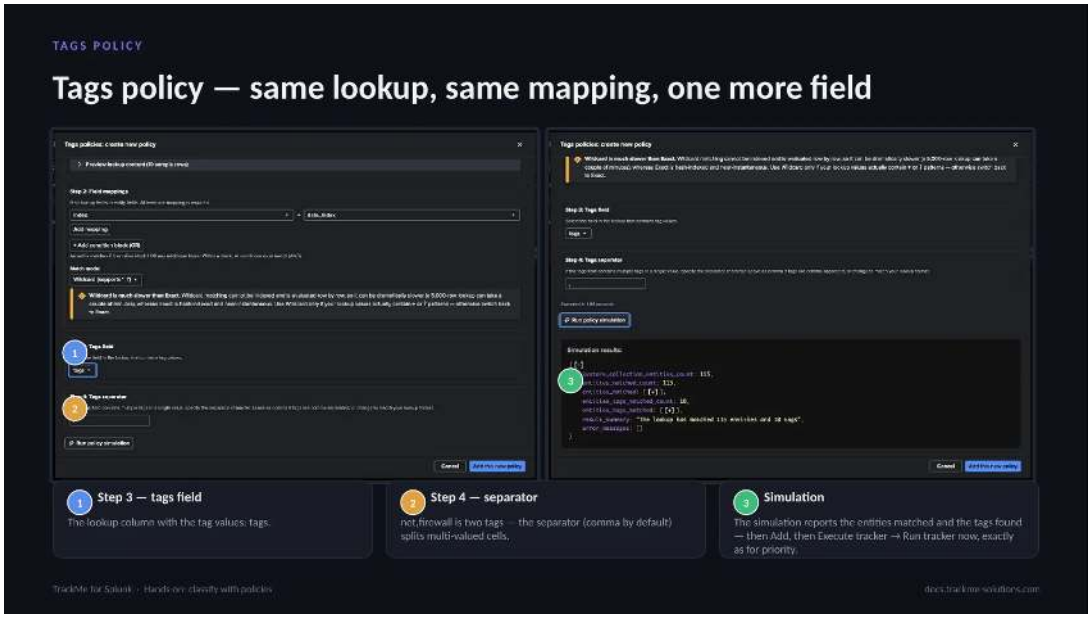
Applying the policy with the tracker



A policy changes nothing by itself; the policy tracker does, on its schedule. The policies table (callout 1) holds one row — type lookup, account local, cmdb-priority-by-index, example_cmdb_indexes_simple, index → data_index, wildcard matching, priority from lookup — editable in place with the pencil. Execute tracker (callout 2) opens "Priority policies: execute tracker" for the report trackme_dsm_priority_tracker_tenant_secops; Run tracker now applies it immediately rather than on the next scheduled cycle.

The execution events (callout 3) come from trackmesplkpriority.py: lookup_policies_count, kvstore_collection_entities_count, entities_updated_count (every entity), entities_failures_count (none) and the run time — the place to check when a policy seems not to apply. Each family has its own tracker; they are ordinary scheduled reports, visible in the Splunk scheduler and in WLK.

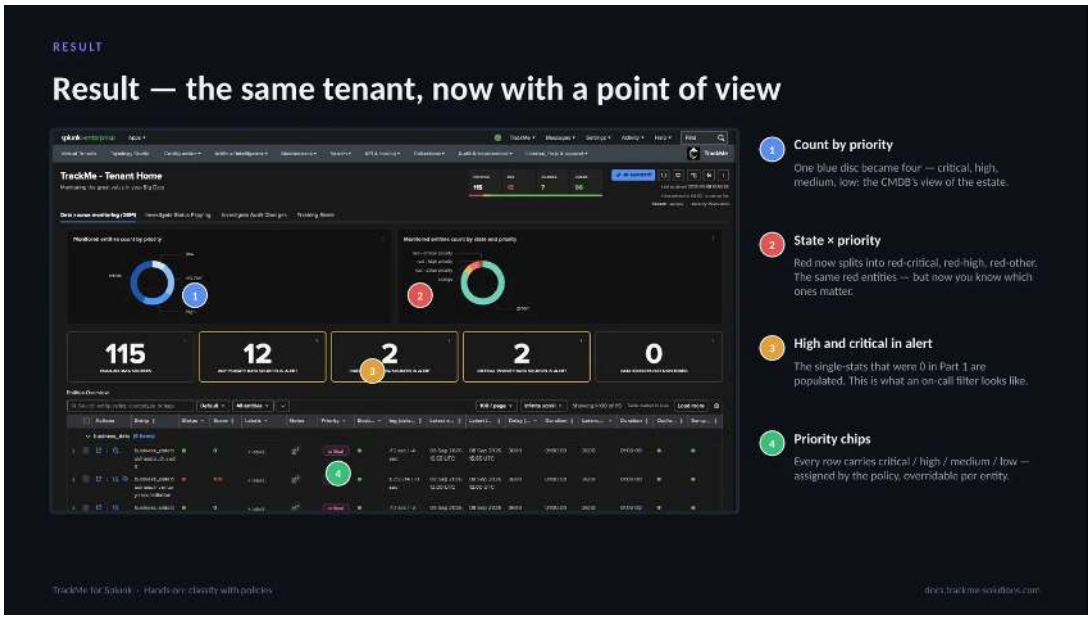
Creating the tags policy



The tags policy is a copy of the priority one: same lookup, same index → data_index mapping, with two different choices. Step 3 (callout 1) selects tags as the lookup column carrying the values. Step 4 (callout 2) sets the separator, comma by default, which splits multi-valued cells — net,firewall becomes two tags. The simulation (callout 3) reports the entities matched and the tags found; then Add, then Execute tracker → Run tracker now, exactly as for priority.

Tags are free-form strings that become a filter and a grouping axis in the entities table, in Virtual Groups, in ML outlier scope filters and in Topology Studio aggregates. Labels are different: a managed, coloured catalogue with its own rule engine.

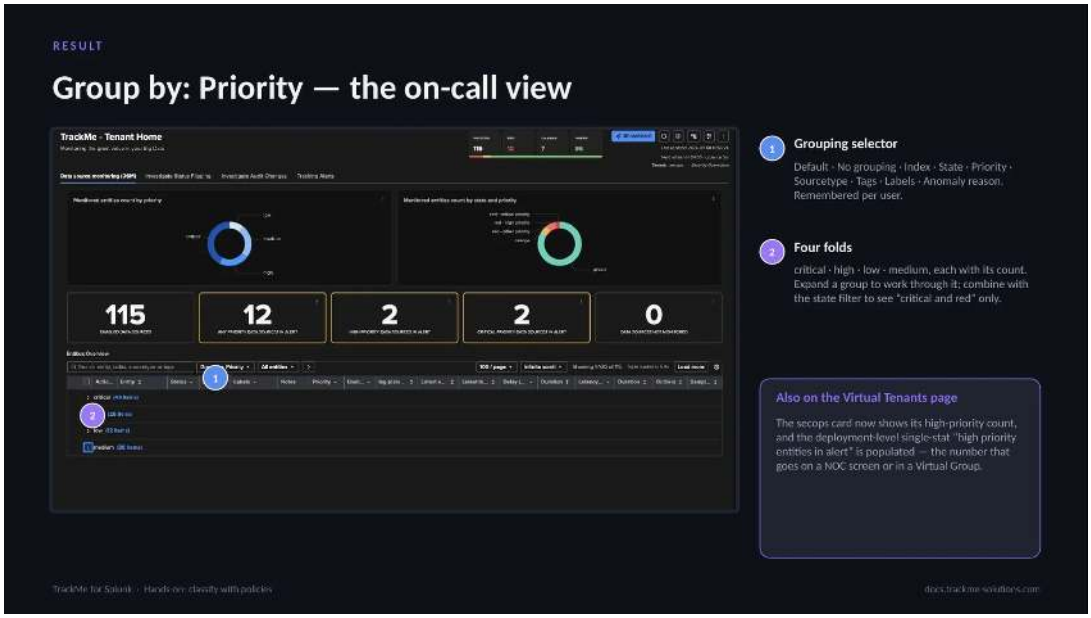
Priority across the tenant summary



Same entities and the same red ones, but now ranked. Count by priority (callout 1) shows one blue disc turned into four — critical, high, medium, low — the CMDB's view of the estate. State × priority (callout 2) splits red into red-critical, red-high and red-other, so the red entities that matter stand out. The single-stats for high and critical in alert (callout 3), which read 0 in Part 1, are populated: this is what an on-call filter looks like.

Every row of the entities table carries a priority chip (callout 4) — critical, high, medium or low — assigned by the policy and overridable per entity. Priority feeds the alerting model: alert on critical and high, watch the rest.

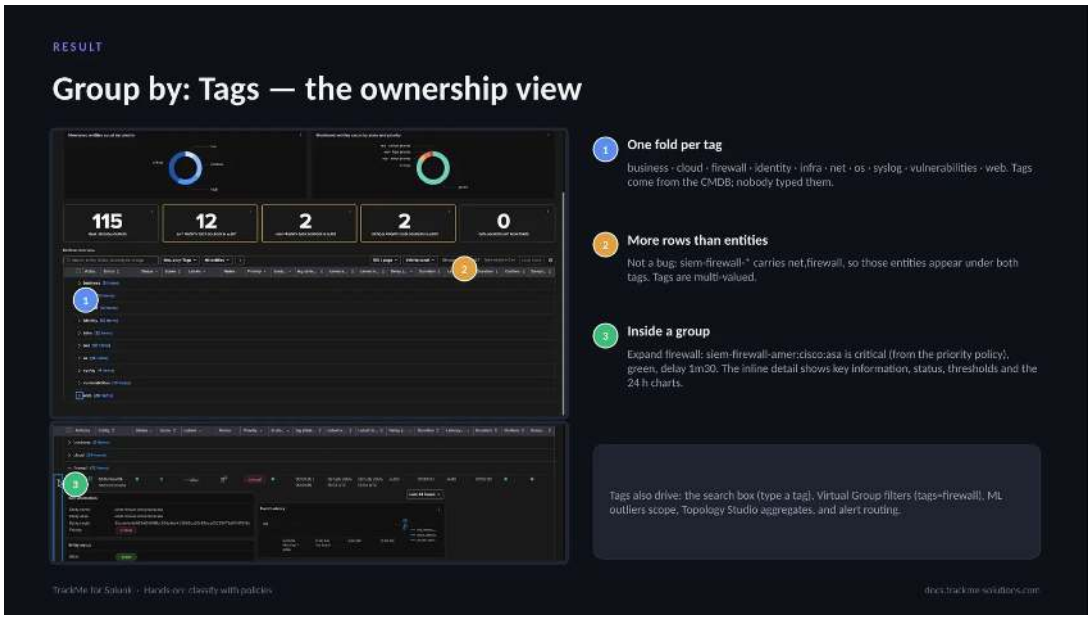
Grouping the entities table by priority



Group by is a per-user display setting on the entities table. The grouping selector (callout 1) offers Default, No grouping, Index, State, Priority, Sourcetype, Tags, Labels and Anomaly reason. Grouped by Priority, the table shows four folds (callout 2) — critical, high, low, medium — each with its count; combined with the state filter it shows "critical and red" only. The critical group contains the business_data inventory feed from Part 1, now critical because business_data is critical in the CMDB.

The Virtual Tenants page reflects the change too: the secops card shows its high-priority count, and the deployment-level single-stat "high priority entities in alert" is populated — the number for a NOC screen or a Virtual Group.

Grouping the entities table by tag

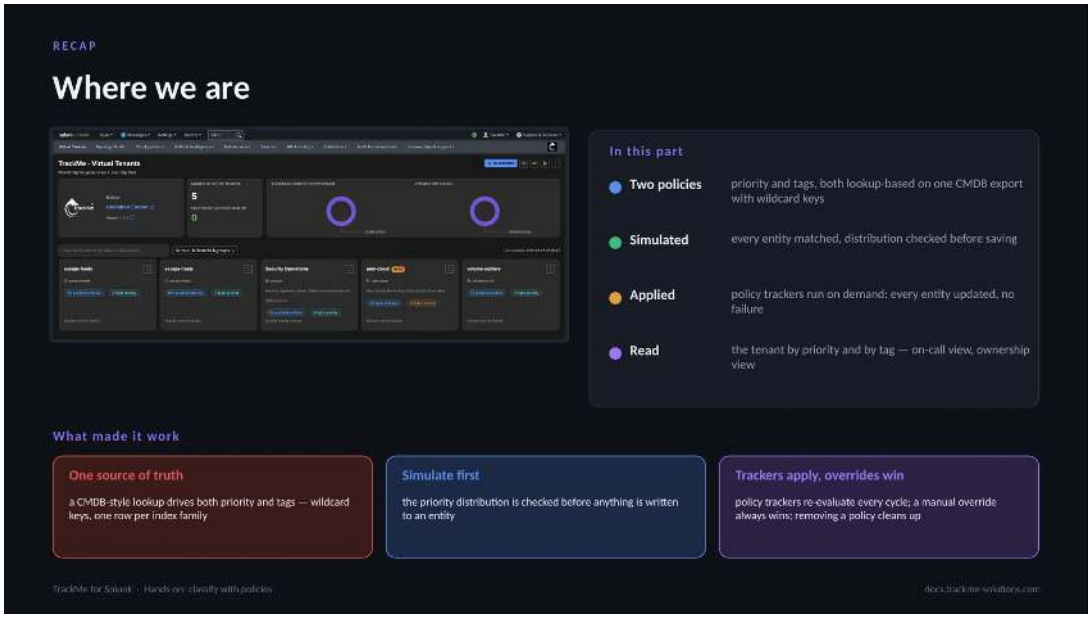


Grouped by Tags, the table shows one fold per tag (callout 1): business, cloud, firewall, identity, infra, net, os, syslog, vulnerabilities, web — all from the CMDB. The table has more rows than entities (callout 2); expected, since siem-firewall-* carries net,firewall and those entities appear under both tags. Tags are multi-valued.

Inside the firewall group (callout 3), siem-firewall-amer:cisco:asa is critical from the priority policy, green, delay 1m30; the inline detail shows key information, status, thresholds and the 24 h charts. Tags are the ownership axis —

the firewall team sees firewall, the identity team sees identity — and also drive the search box, Virtual Group filters (tags=firewall), ML outliers scope, Topology Studio aggregates and alert routing.

Where we are



Classification, step 4 of the cycle, is a policy problem, not a data-entry problem. Two lookup-based policies, priority and tags, run on one CMDB export with wildcard keys. Both were simulated first, the distribution checked before anything was written, then applied by the policy trackers on demand: every entity updated, no failure. The tenant now reads by priority (on-call view) and tag (ownership view).

What makes it work: one source of truth for both families, simulation before saving, and trackers that re-evaluate every cycle while a manual override always wins and a removed policy cleans up. Alert scoping relies on priority: an alert restricted to critical and high pages for the firewall and identity feeds, not the low-priority ones.

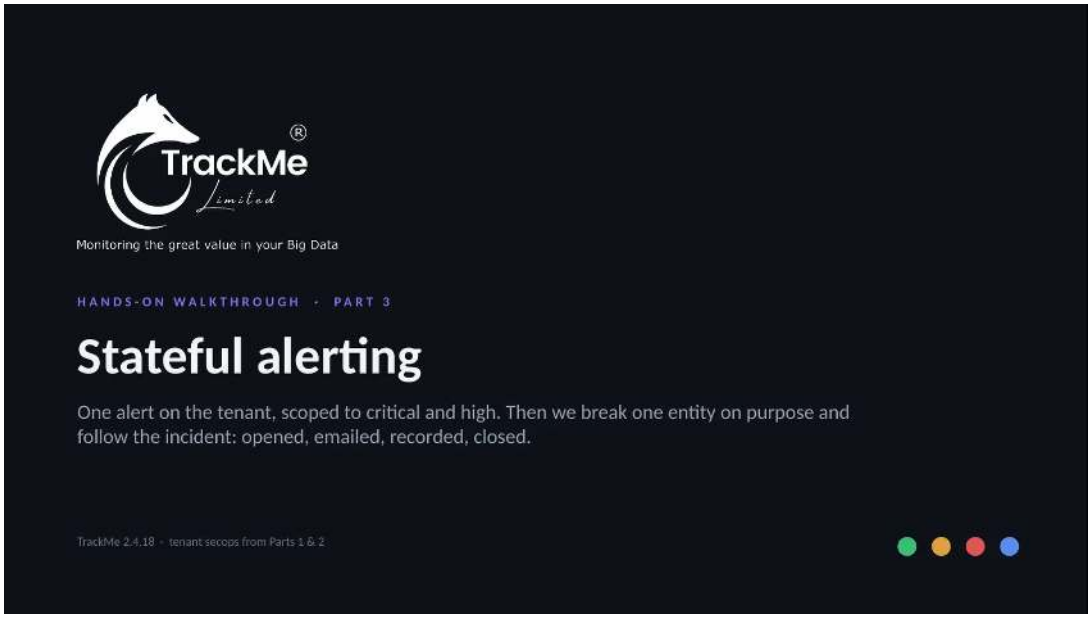
2.1.4 Part 3 — Stateful alerting

One alert on the tenant, scoped to critical and high; then an entity is broken on purpose and the incident is followed: opened, emailed, recorded, closed.

`Download the deck (PDF) · PowerPoint`

about 20 minutes · 16 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

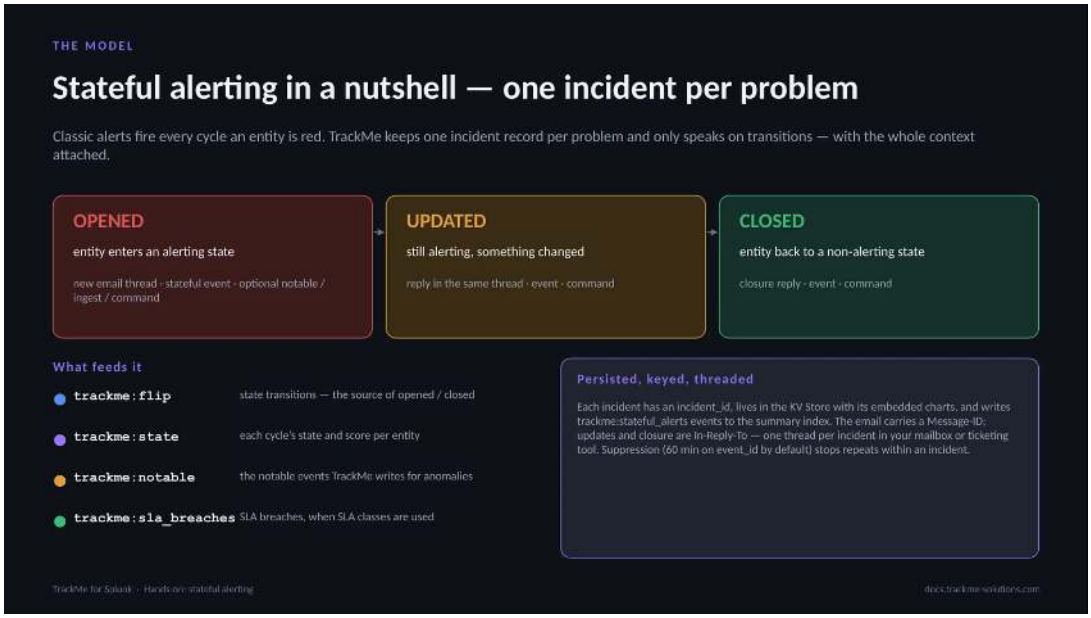
Overview



Part 3 of the hands-on walkthrough covers step 6 of the cycle, ALERT, on the secops tenant classified in Part 2. One stateful alert is created on the tenant, with emails restricted to critical and high priority entities. A healthy critical entity is then forced into red with a manual score influence, and the resulting incident is followed end to end: opened, emailed, recorded and closed.

Every screen in this part was captured live on TrackMe 2.4.18.

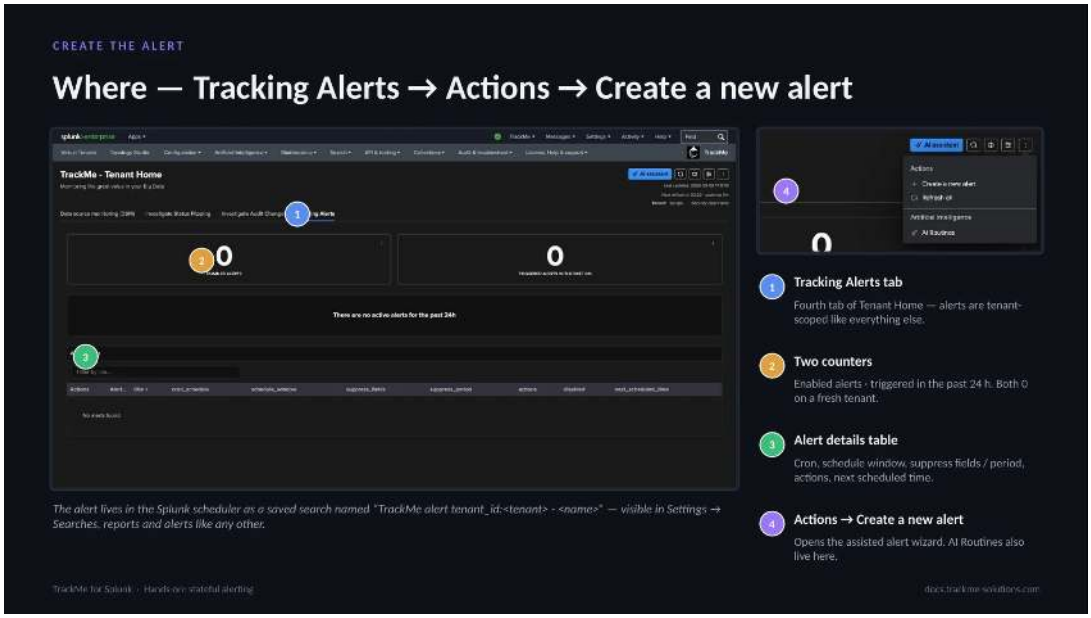
Incident lifecycle and its inputs



Classic alerts fire every cycle an entity is red. TrackMe keeps one incident record per problem and only speaks on transitions. An incident is OPENED when the entity enters an alerting state (new email thread, stateful event, optional notable / ingest / command), UPDATED when it is still alerting and something changed, and CLOSED when it returns to a non-alerting state.

Each incident has an incident_id, lives in the KV Store with its charts, and writes trackme:stateful_alerts events to the summary index. The email carries a Message-ID; updates and closure are In-Reply-To, one thread per incident. Suppression, 60 min on event_id by default, stops repeats within an incident. Inputs: trackme:flip (the source of opened and closed), trackme:state, trackme:notable, trackme:sla_breaches.

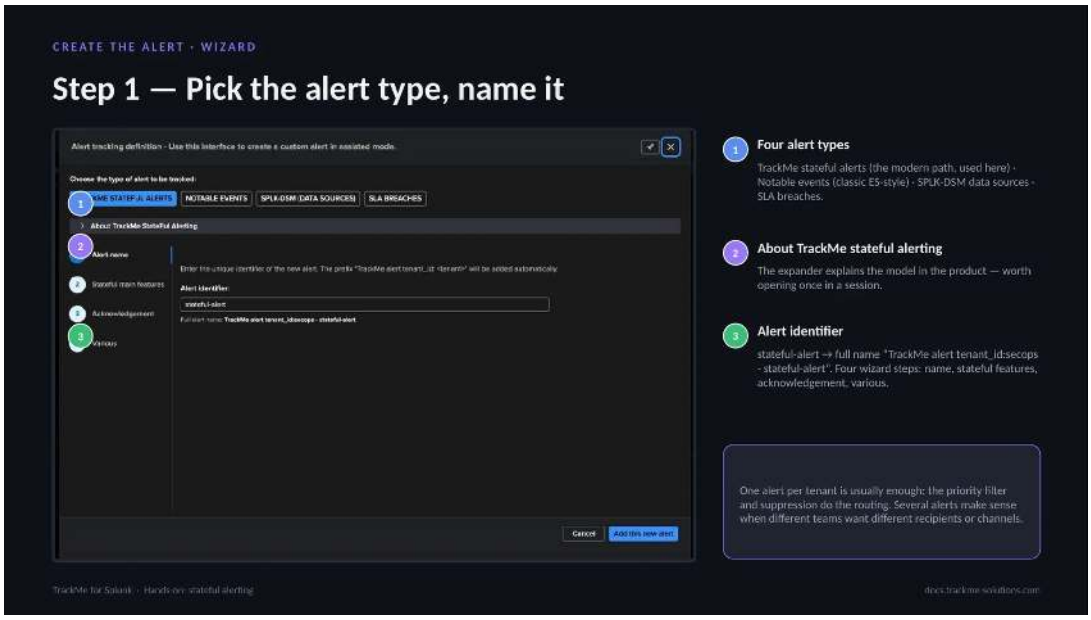
The Tracking Alerts tab



Alerts are created per tenant from the Tracking Alerts tab, the fourth tab of Tenant Home (callout 1). Two counters (callout 2) show enabled alerts and alerts triggered in the past 24 h, both 0 on a fresh tenant. The alert details table (callout 3) lists cron, schedule window, suppress fields and period, actions and next scheduled time. Actions → Create a new alert (callout 4) opens the assisted wizard; AI Routines also live here.

The wizard produces a normal Splunk saved search named "TrackMe alert tenant_id:<tenant> - <name>", visible in Settings → Searches, reports and alerts, with TrackMe alert actions attached — nothing proprietary in the scheduler, which matters where saved searches are managed through CI/CD.

Alert type and identifier

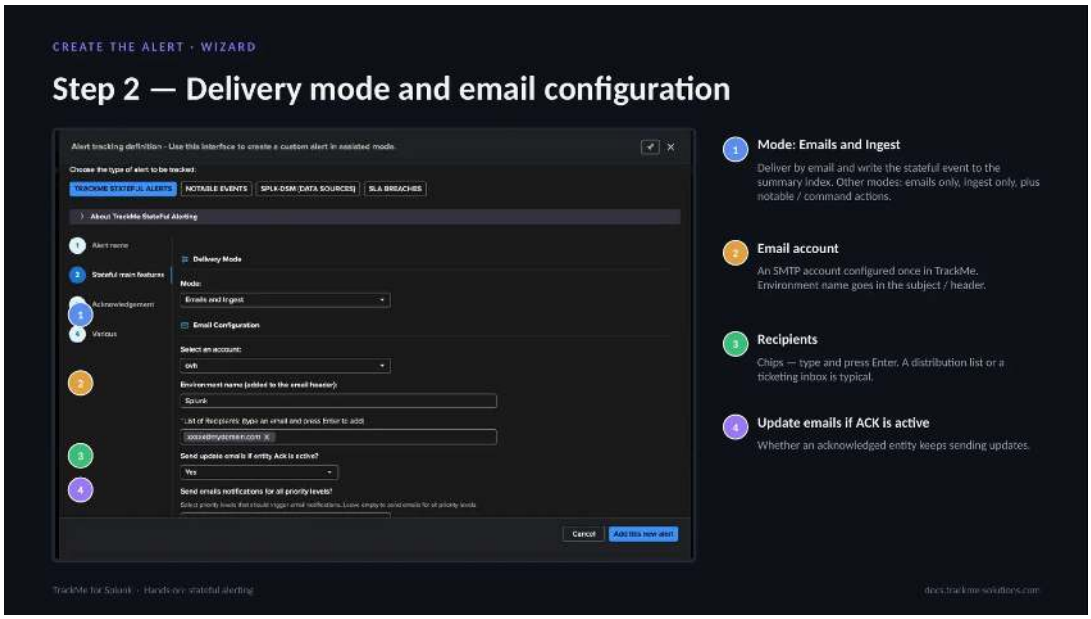


Step 1 offers four alert types (callout 1): TrackMe stateful alerts, the modern path used here; Notable events, the classic ES-style path where TrackMe writes notables and Enterprise Security correlates them; SPLK-DSM data sources; and SLA breaches. The About TrackMe stateful alerting expander (callout 2) explains the model inside the product. The alert identifier (callout 3), stateful-alert, is prefixed automatically with the tenant id, giving "TrackMe alert tenant_id:secops - stateful-alert". The wizard has four steps: name, stateful features, acknowledgement, various.

One alert per tenant is usually enough, because the priority filter and suppression do the routing; several alerts make

sense when different teams want different recipients or channels.

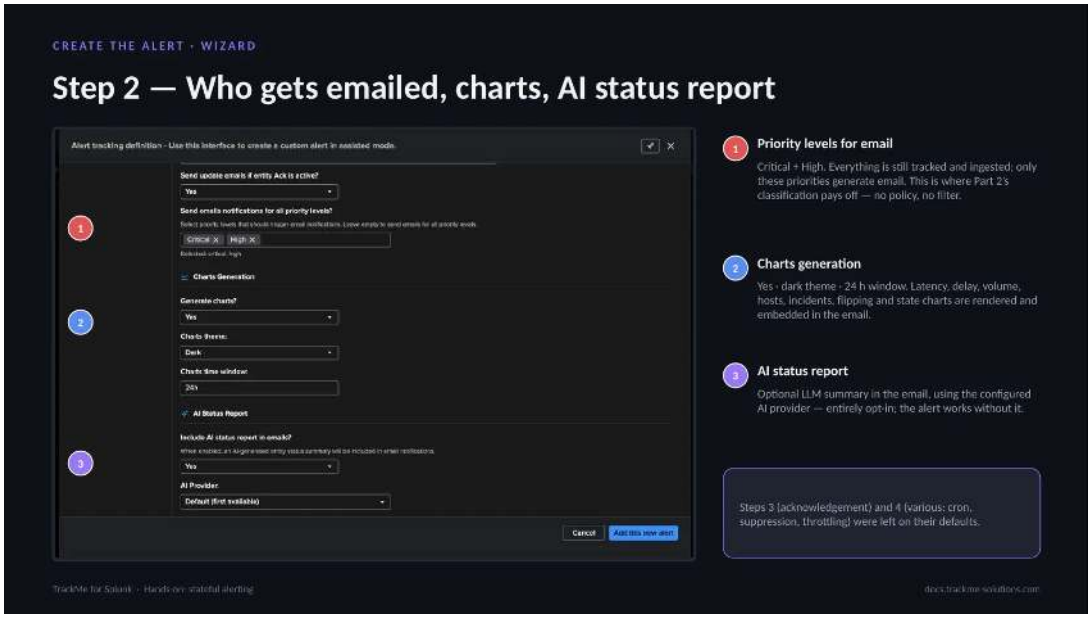
Delivery mode and email account



The mode (callout 1), Emails and Ingest, delivers by email and writes the stateful event to the summary index — the usual choice, since people get the thread and Splunk keeps the record; other modes are emails only, ingest only, plus notable and command actions. The email account (callout 2) is an SMTP account configured once under Configuration; the wizard only selects it, and its environment name goes in the subject and header, separating prod from preprod in a shared inbox.

Recipients (callout 3) are entered as chips, one per Enter; a distribution list or a ticketing inbox is typical. Update emails if ACK is active (callout 4) decides whether an acknowledged entity keeps sending updates.

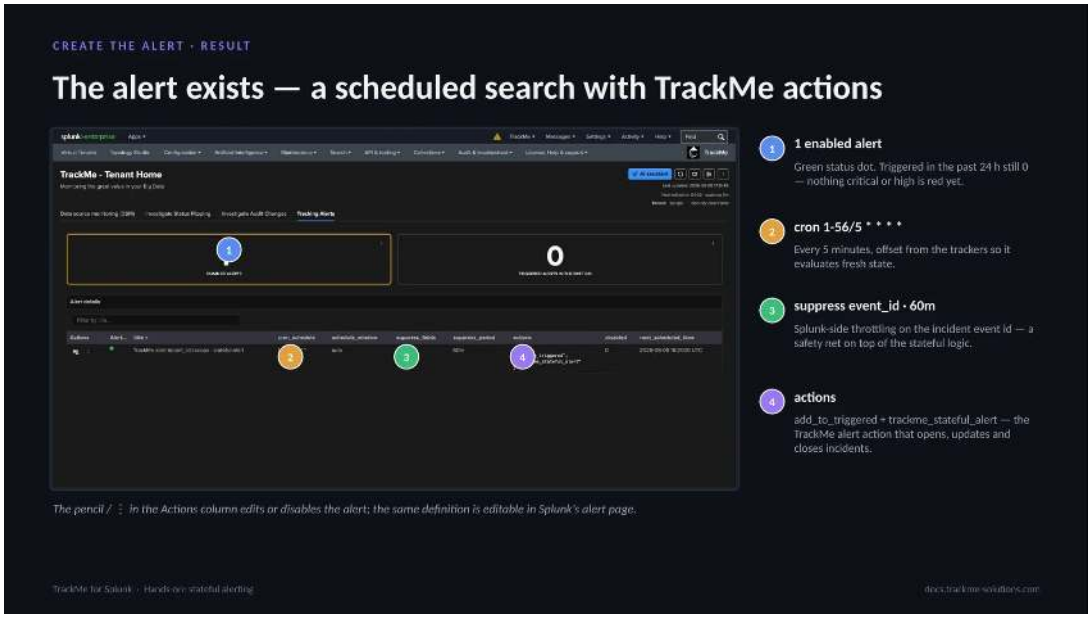
Priority filter, charts and AI report



Priority levels for email (callout 1) is set to Critical + High: everything is still tracked and ingested, but only these priorities generate email. This is where the classification from Part 2 pays off and the key setting for a quiet on-call: low and medium feeds still get their incidents recorded, but nobody is paged for them.

Charts generation (callout 2) is Yes, dark theme, 24 h window: latency, delay, volume, hosts, incidents, flipping and state charts are embedded, giving the recipient 24 hours of context without opening Splunk. AI status report (callout 3) adds an opt-in LLM summary from the configured AI provider. Steps 3 (acknowledgement) and 4 (cron, suppression, throttling) keep their defaults.

The saved search behind the alert



The Tracking Alerts tab now shows 1 enabled alert with a green dot (callout 1); triggered in the past 24 h is still 0, as nothing critical or high is red. The cron is 1-56/5 * * * * (callout 2): every 5 minutes, one minute after the trackers so it evaluates fresh state. Suppression is on event_id for 60m (callout 3), Splunk-side throttling on the incident event id, a safety net over the stateful logic.

The actions (callout 4) are add_to_triggered, keeping Splunk's triggered-alerts list in sync, and trackme_stateful_alert, the TrackMe alert action that opens, updates and closes incidents. The pencil and ⋮ in the Actions column edit or disable the alert, also editable from Splunk's alert page.

Adding a manual score influence



The entity ⋮ menu (callout 1) lists: manually influence the score, false positive, disruption queue, lagging classes,

priority, maintenance. Callout 2 shows a healthy critical entity, siem-firewall-amer:cisco:asa — green, score 0, priority critical from the policy. Add 100 to the score (callout 3), with a comment: audited as a manual_score event; a score_breached anomaly is recorded.

Manual score influence adds or subtracts a value from the impact score; operators use it to force attention on an entity or soften one. Here it produces a deterministic red state on a healthy feed without touching data or thresholds. Being critical, the entity passes the alert's email filter; the AI status report will name the manual override as the cause.

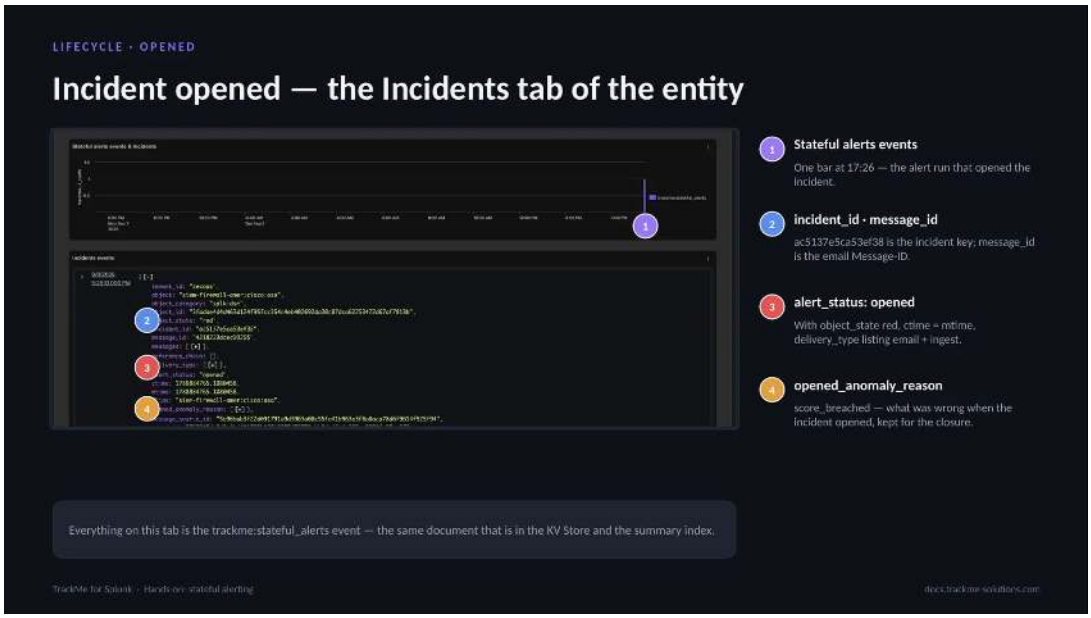
The entity header after the influence



The metrics are still healthy (callout 1): delay 02:59, latency 01:50, thresholds 3600 — nothing wrong with the feed. The state is red (callout 2), latest flip at 16:24 UTC, priority critical. anomaly_reason is score_breached (callout 3): the status reason names the manual influence rather than telling a false delay story. Impact score is 100 (callout 4), entirely the manual component, and score_definition in the Status message tab shows it.

This is the honesty of the scoring model — the reason is always the real one. The next alert run, every 5 minutes, sees a critical entity in red with no open incident and opens one.

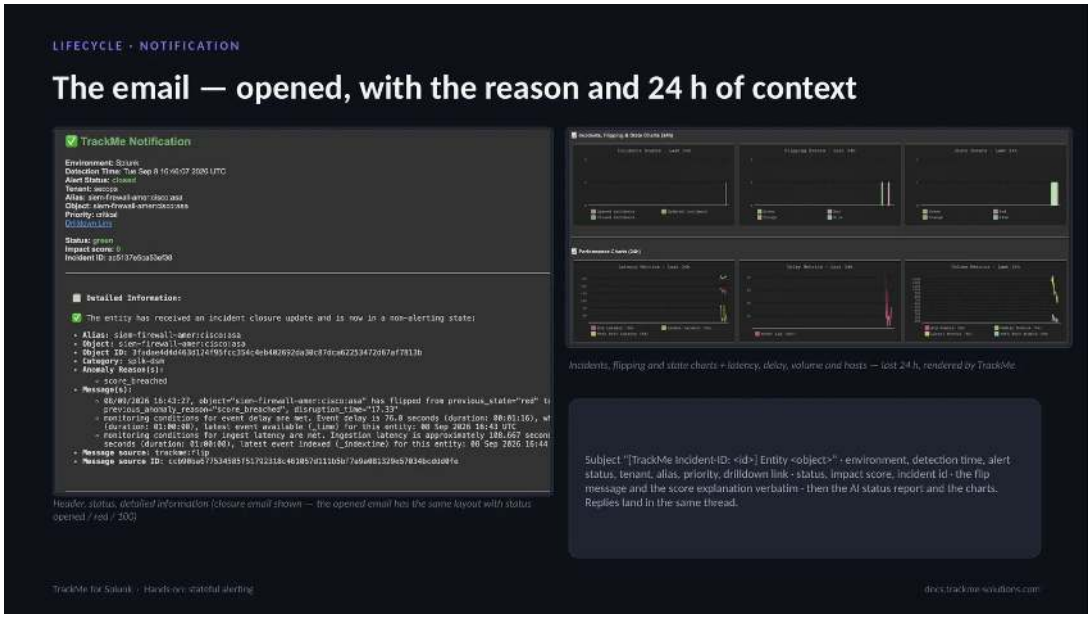
The incident record on the entity



The Incidents tab of the entity shows the incident record as TrackMe stores it. The stateful alerts events chart (callout 1) has one bar at 17:26, the alert run that opened the incident. incident_id ac5137e5ca53ef38 is the incident key and message_id is the email Message-ID (callout 2); message_source_id points at the flip event that caused the incident. alert_status is opened (callout 3), with object_state red, ctime equal to mtime and delivery_type listing email and ingest. opened_anomaly_reason (callout 4) is score_breached — what was wrong when the incident opened, kept for the closure.

Everything on this tab is the trackme:stateful_alerts event, the same document held in the KV Store and in the summary index.

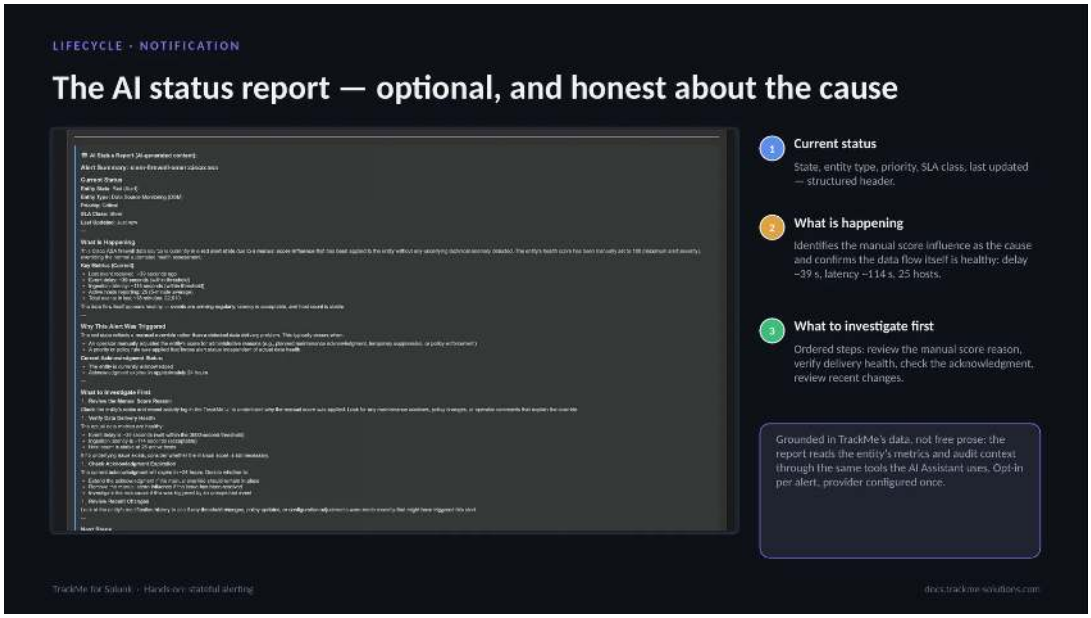
Anatomy of the notification email



The email reads top to bottom: who, what, why, how bad, link to the entity. The subject is "[TrackMe Incident-ID: <id>] Entity <object>". The header carries environment, detection time, alert status, tenant, alias, priority and the drilldown link; the status block gives status, impact score and incident id; detailed information quotes the flip message and score explanation verbatim. Then the AI status report, when enabled, and the 24 h charts rendered by TrackMe: incidents, flipping and state, plus latency, delay, volume and hosts.

The closure email is shown; the opened email has the same layout with status opened, red and impact score 100. Replies land in the same thread, and recipients need no Splunk access.

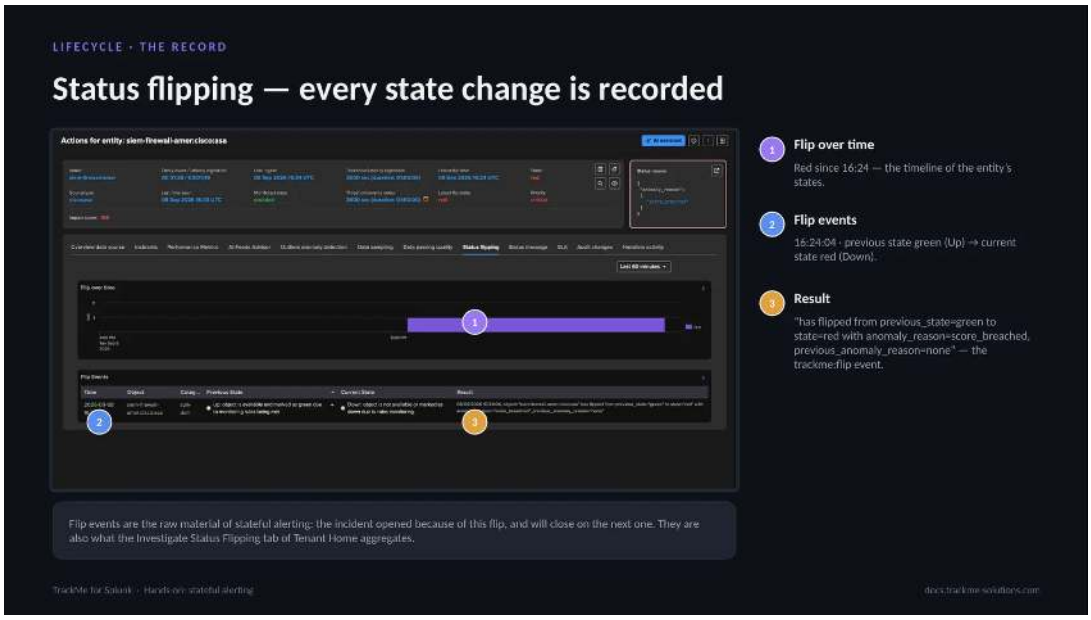
Reading the AI status report



Current status (callout 1) is a structured header: state, entity type, priority, SLA class, last updated. What is happening (callout 2) names the manual score influence as the cause and confirms the data flow is healthy — delay ~39 s, latency ~114 s, 25 hosts. What to investigate first (callout 3) gives ordered steps: review the manual score reason, verify delivery health, check the acknowledgement, review recent changes.

The report is grounded in TrackMe's data rather than free prose: it reads the entity's metrics and audit context through the same tools the AI Assistant uses, hence the correct verdict of a manual override on a healthy feed. Opt-in per alert, provider configured once; the email is complete without it.

Flip events behind the incident

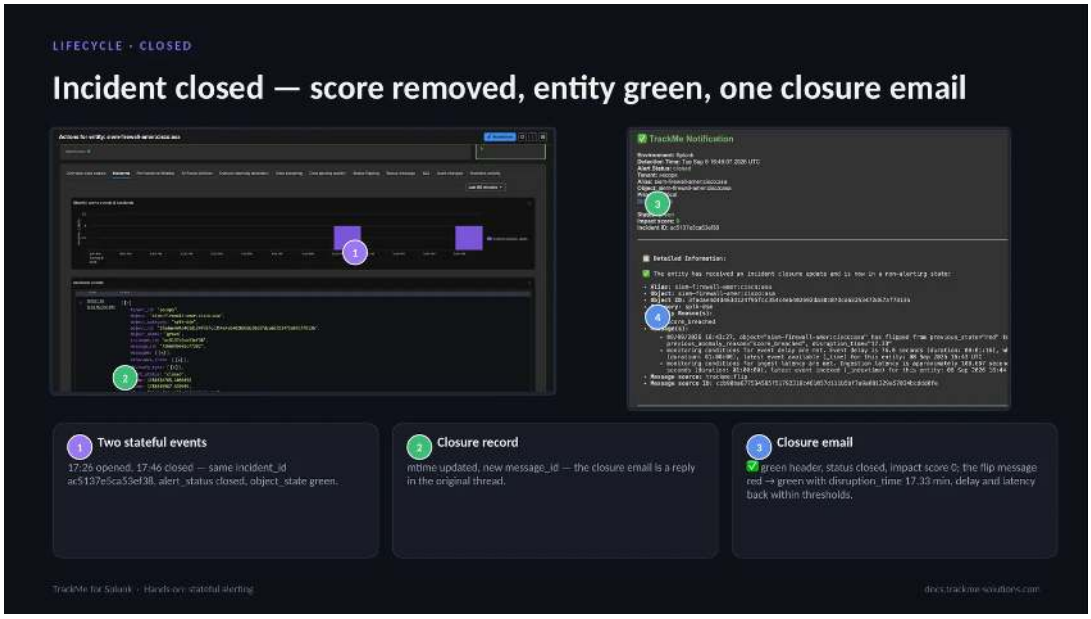


The Status flipping tab records every state change as an event with a history. Flip over time (callout 1) is the timeline of the entity's states, red since 16:24. The flip events list (callout 2) shows 16:24:04, previous state green (Up) to current state red (Down). The result (callout 3) is the trackme:flip event itself: "has flipped from previous_state=green to state=red with anomaly_reason=score_breached, previous_anomaly_reason=none" — the same sentence quoted in the email.

Flip events are the raw material of stateful alerting: the incident opened because of this flip and closes on the next

one. They are also what the Investigate Status Flipping tab of Tenant Home aggregates, which is how flapping entities are found later.

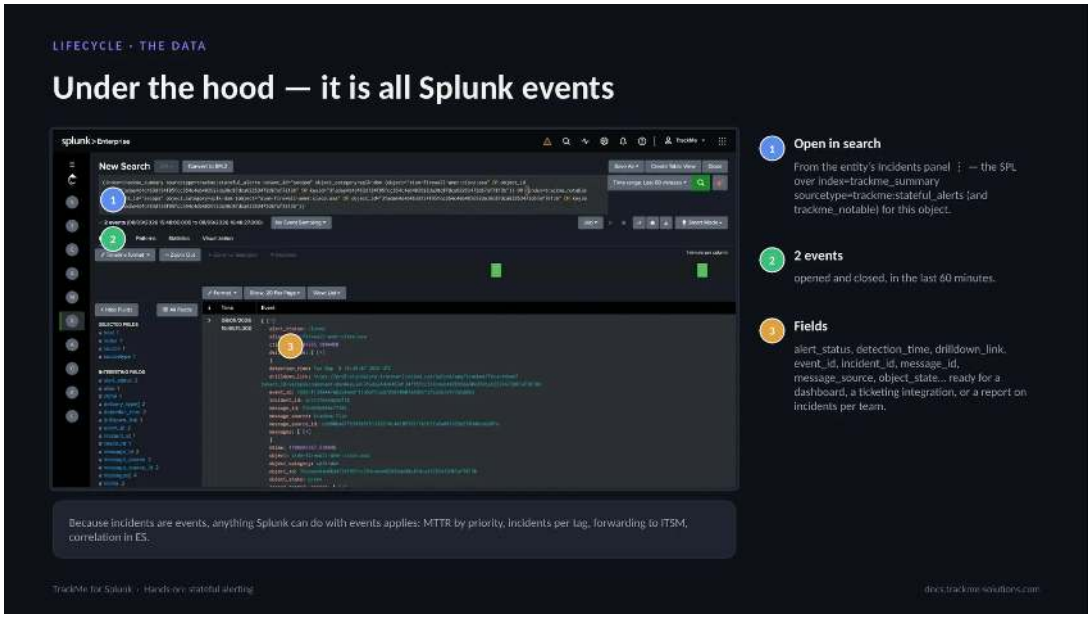
Closure record and closure email



Removing the 100 points turns the entity green on the next tracker run; the alert's next run finds an open incident and closes it. The Incidents tab shows two stateful events (callout 1): 17:26 opened, 17:46 closed, same incident_id ac5137e5ca53ef38, alert_status closed, object_state green. The closure record (callout 2) has an updated mtime and a new message_id, so the closure email is a reply in the original thread.

The closure email (callout 3) has a green header, status closed, impact score 0; it quotes the recovery flip red → green with disruption_time 17.33 min, delay and latency back within thresholds. SLA reporting uses the same disruption time; the AI recovery summary confirms the entity is healthy.

Searching the incident events

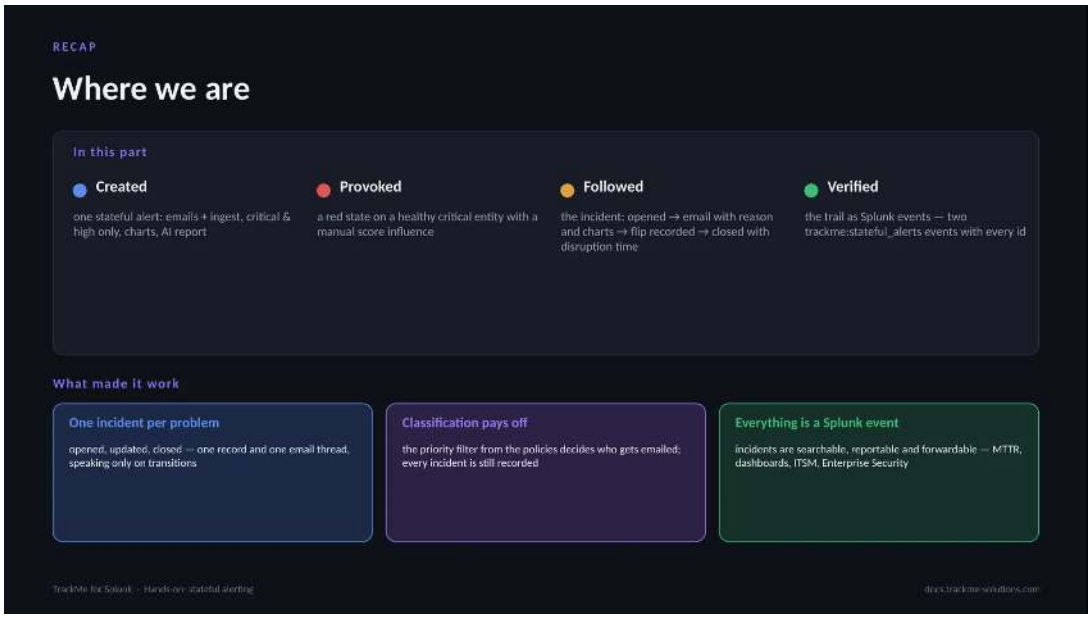


Open in search (callout 1), from the ⋮ of the entity's Incidents panel, runs the SPL over index=trackme_summary sourcetype=trackme:stateful_alerts (and trackme_notable) for this object. It returns 2 events in the last 60 minutes (callout 2): opened and closed. The fields (callout 3) — alert_status, detection_time, drilldown_link, event_id, inci-

dent_id, message_id, message_source, object_state and more — are ready for a dashboard, a ticketing integration or a report on incidents per team.

Because incidents are events, anything Splunk can do with events applies: MTTR by priority, incidents per tag, forwarding to ITSM, correlation in Enterprise Security. There is no black box: one event per transition, with every id needed, and the drilldown_link brings a ticket back to the entity.

Where we are



Discover, classify, alert: the cycle is complete. One stateful alert was created — emails and ingest, critical and high only, charts, AI report — and a red state provoked on a healthy critical entity by score influence. The incident was followed from opened, through the email with reason and charts, the recorded flip, to closed with its disruption time, verified as two trackme:stateful_alerts events with every id.

Key points: one incident per problem — opened, updated, closed, one record and one email thread, speaking only on transitions; the priority filter from the policies decides who is emailed while every incident is recorded; and everything is a Splunk event — searchable, reportable, forwardable to dashboards, ITSM or Enterprise Security.

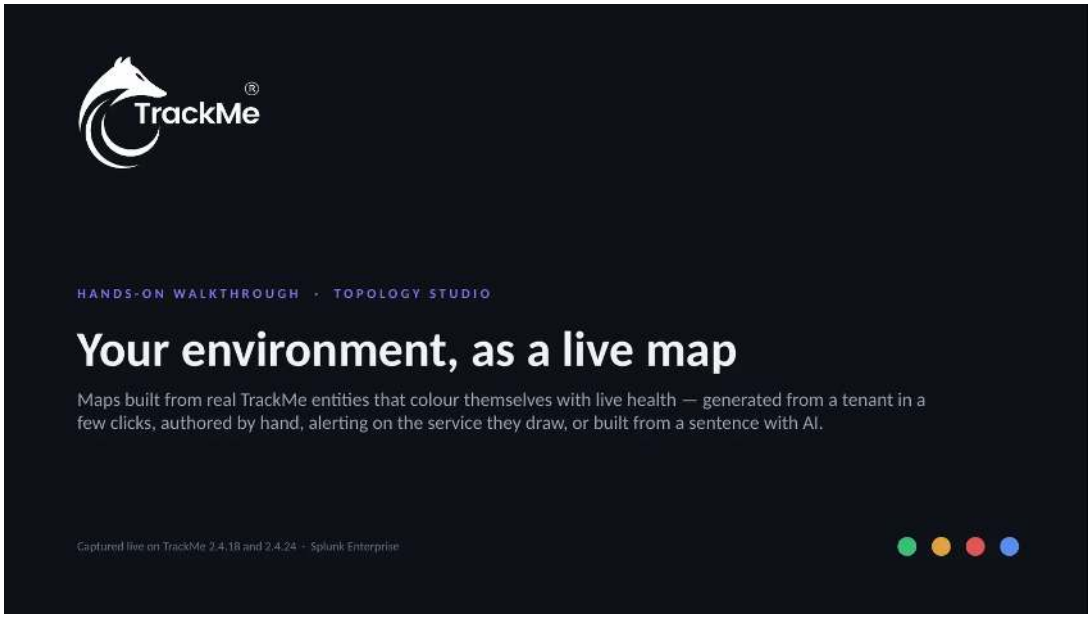
2.1.5 Topology Studio

Maps built from real entities that colour themselves with live health — generated from a tenant, authored by hand, alerting on the service they draw, or built from a sentence with AI; and the maps the findings tenants draw from their own workspace.

`Download the deck (PDF) · PowerPoint`

about 40 minutes · 40 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

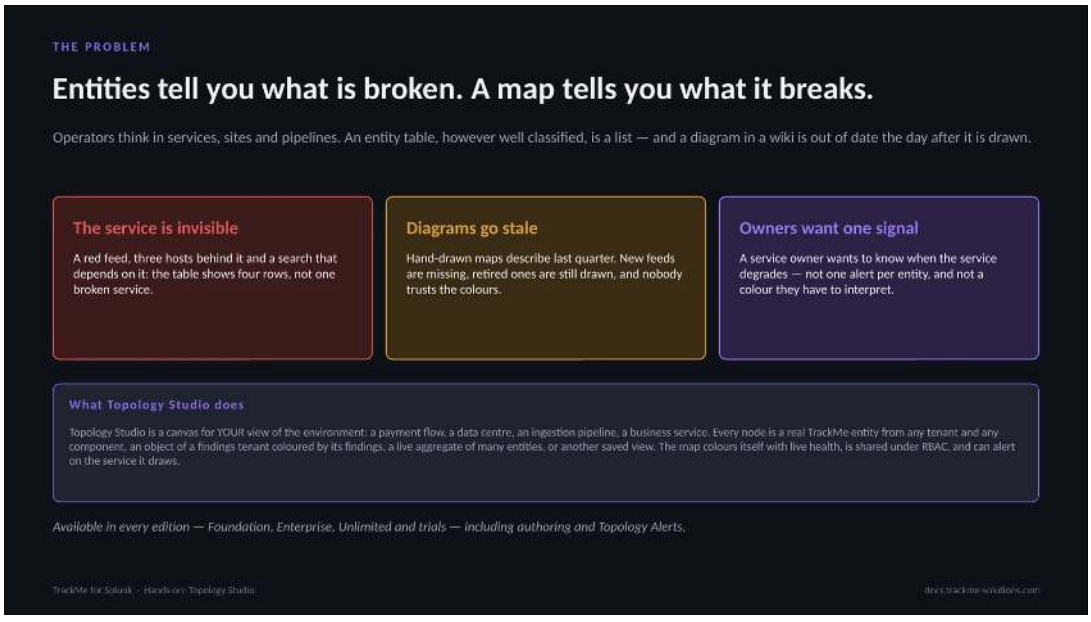
Overview



Topology Studio draws maps from real TrackMe entities, and every node colours itself with live health. A map can be generated from a tenant in a few clicks, authored by hand, made to alert on the service it draws, or built from a sentence with AI. The screens were captured on TrackMe 2.4.18 running on Splunk Enterprise, against a production-like deployment with several tenants, and the maps of the findings tenants on TrackMe 2.4.24.

The walkthrough has four parts: generating a map from a tenant, building one from an empty canvas with aggregates and auto-generated members, alerting on the map, and building one from a sentence with AI. It closes with a few maps that run in production every day, and with the maps the findings tenants draw for themselves.

Why an entity list is not enough

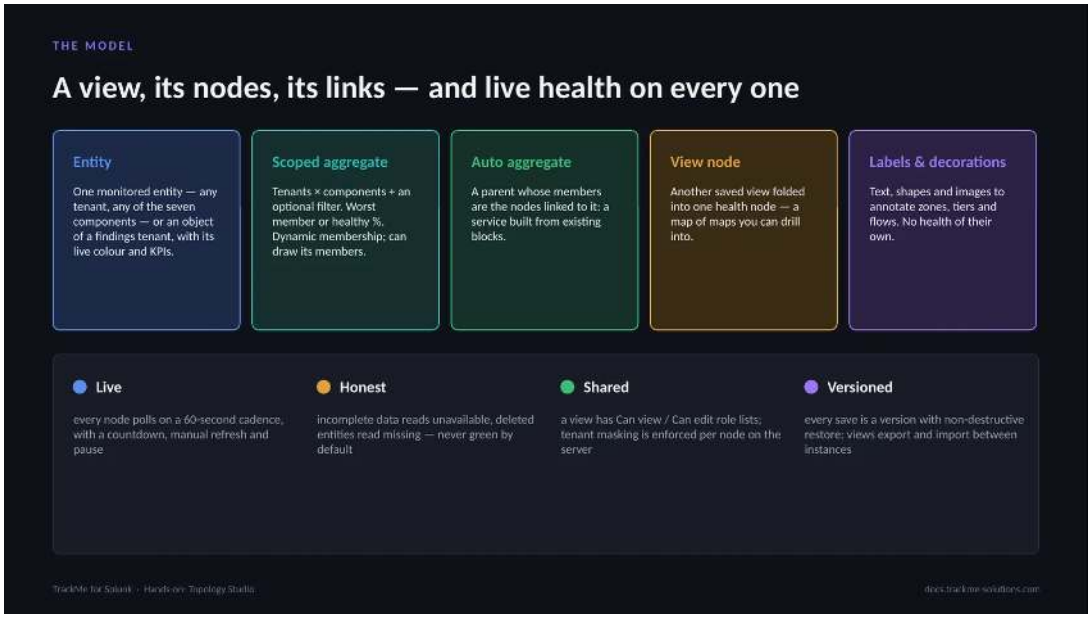


An entity table is a list: a red feed, three hosts behind it and a search that depends on it show as four rows, not one broken service. Hand-drawn diagrams go stale within a week. A service owner wants one signal when the service degrades, not one alert per entity.

Topology Studio is a canvas for the operator's own view of the environment, such as a payment flow or a business service. Every node is a real TrackMe entity from any tenant and component, an object of a findings tenant coloured

by its findings, a live aggregate of many entities, or another saved view. The map colours itself with live health, is shared under RBAC, can alert on the service it draws, and is available in every edition, including trials.

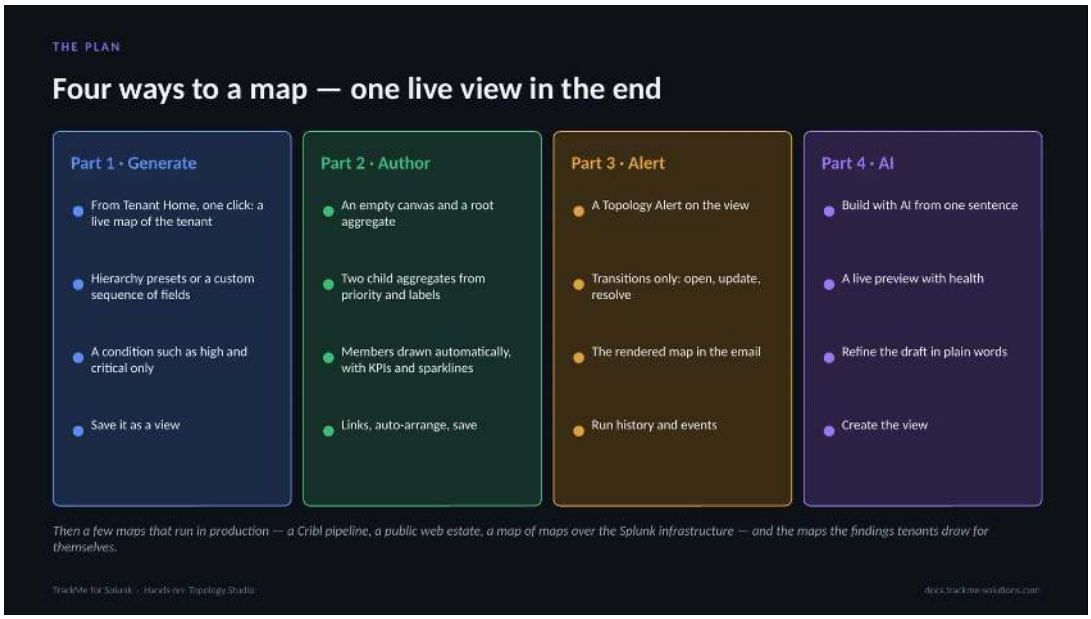
Views, node types and live health



A view is one map: an id, a display name, a category and two role lists, Can view and Can edit. An Entity node is one monitored entity with up to five KPIs, or an object of a findings tenant coloured by its findings. A Scoped aggregate rolls up tenants × components plus an optional filter, with worst-member or healthy-percentage health and dynamic membership. An Auto aggregate is a parent whose members are the nodes linked to it. A View node folds another saved view into one health node. Labels and decorations carry no health.

Every node polls every 60 seconds. Incomplete data reads unavailable and a deleted entity reads missing, never green by default. Tenant masking is enforced per node on the server, and every save is a restorable version.

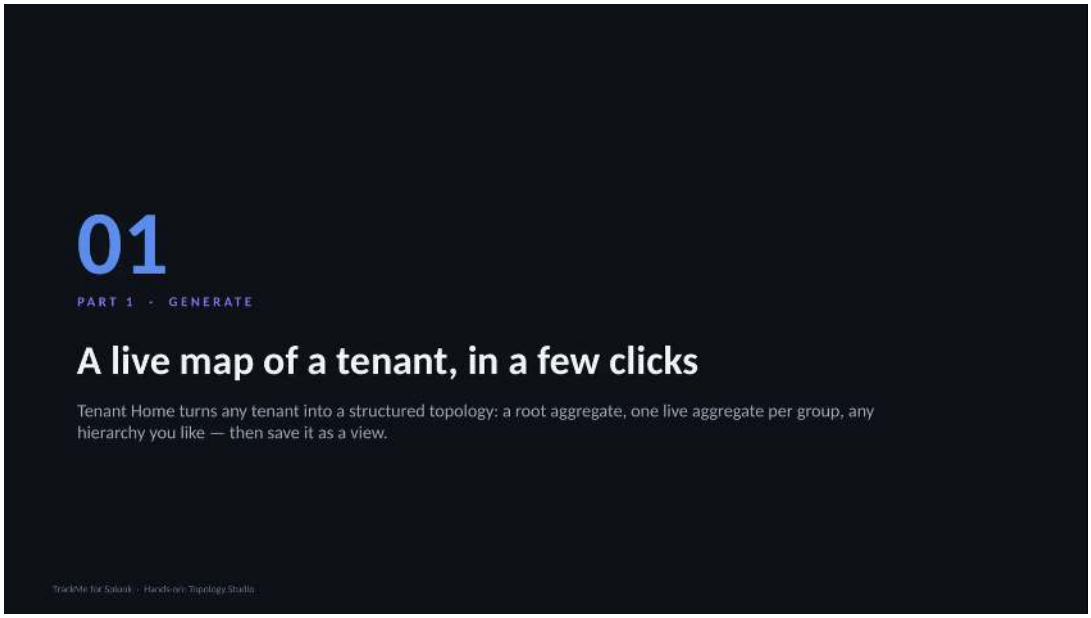
Four routes to a live map



The four parts are not steps of one procedure; each is a way to obtain a live map. Part 1 generates a map from Tenant Home in one click, with a chosen hierarchy and an optional condition. Part 2 authors a map from an empty canvas: a root aggregate, two child aggregates from priority and labels, members drawn automatically with KPIs and sparklines.

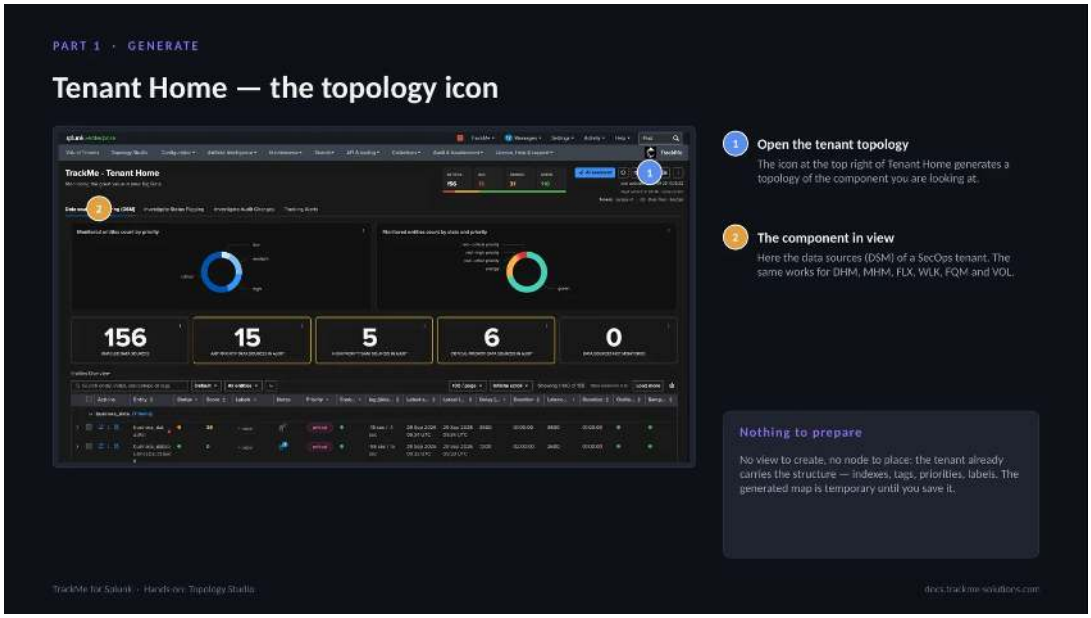
Part 3 attaches a Topology Alert to the view, notifying on transitions only with the rendered map in the email. Part 4 builds a map with AI from one sentence and refines it in plain words. In production follows: a Cribl pipeline, a web estate, a map of maps, and the maps the findings tenants draw.

Part 1 — Generate



The fastest way to a map starts from a tenant that already knows its own structure: indexes, tags, priorities and labels. Tenant Home turns any tenant into a structured topology, with a root aggregate and one live aggregate per group, in any hierarchy. The operator picks the hierarchy, narrows it with a condition and saves it as a view once it is right. The generated map is temporary until it is saved.

Opening the topology from Tenant Home



The topology icon at the top right of Tenant Home (callout 1) generates a topology of the component in view; callout 2 marks that component. Here it is the data sources (DSM) of a SecOps tenant tracking a few hundred data sources; the same works for DHM, MHM, FLX, WLK, FQM and VOL. The icon opens a full-screen panel with three tabs, Topology Studio, Link graph and Network graph; the Topology Studio tab is the one used here.

Nothing needs preparing: no view to create and no node to place, because the tenant already carries the structure in

its indexes, tags, priorities and labels. The generated map stays temporary until it is saved.

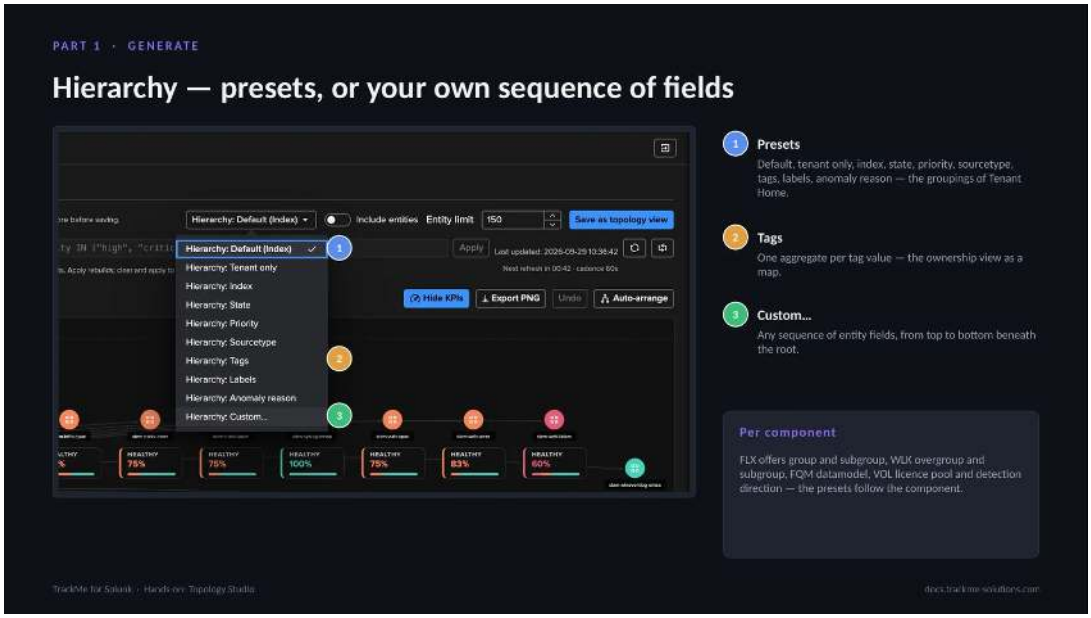
The temporary topology and its controls



The generated map opens as a temporary topology (callout 1): nothing is saved yet, and the banner says how many entities are represented. Hierarchy (callout 2) is one selector, defaulting per component: index for DSM, group for FLX, overgroup for WLK, datamodel for FQM, the tenant root for DHM and MHM. Include entities and Entity limit (callout 3) add individual entity nodes on demand, within a limit; aggregates only is the default. Condition (callout 4) is an optional filter applied to every generated aggregate.

The Root aggregate (callout 5) is the tenant itself with its live healthy percentage; only monitored entities are included. Save as topology view (callout 6) freezes membership and layout while health stays live.

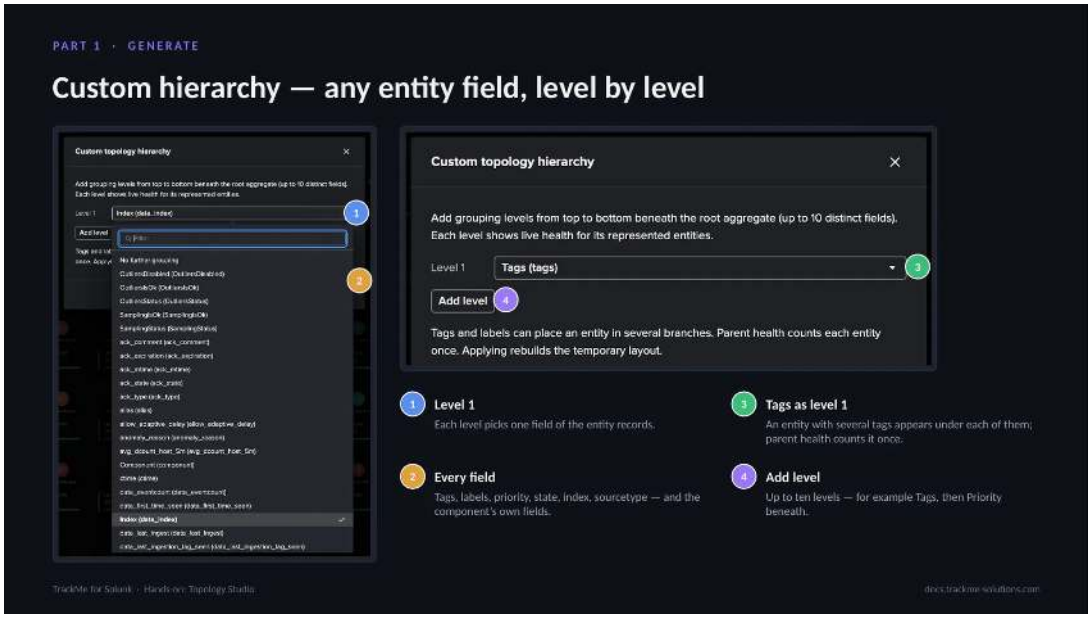
Hierarchy presets per component



The hierarchy selector lists Presets (callout 1): Default, tenant only, index, state, priority, sourcetype, tags, labels and anomaly reason, the same groupings as the Tenant Home table. Choosing Tags (callout 2) gives one aggregate per tag value, the ownership view as a map. Custom… (callout 3) opens a dialog to choose any sequence of entity fields, from top to bottom beneath the root.

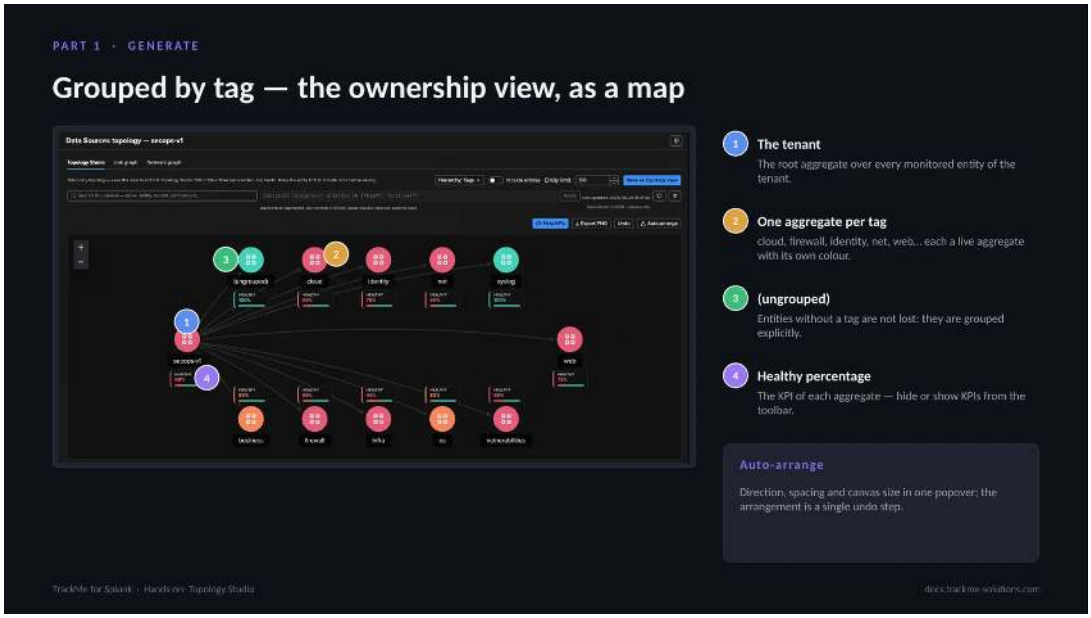
The presets follow the component: FLX offers group and subgroup, WLK overgroup and subgroup, FQM datamodel, and VOL licence pool and detection direction.

Building a custom hierarchy



The custom hierarchy dialog defines grouping levels from top to bottom beneath the root aggregate. Each level (callout 1) picks one field of the entity records; here Tags is the single level. Every field (callout 2) of the records is offered: tags, labels, priority, state, index, sourcetype and the component's own fields. An entity with several tags appears under each of them, while the parent health still counts it once (callout 3). Add level (callout 4) adds up to ten distinct levels, for example Tags with Priority beneath. Applying rebuilds the temporary layout.

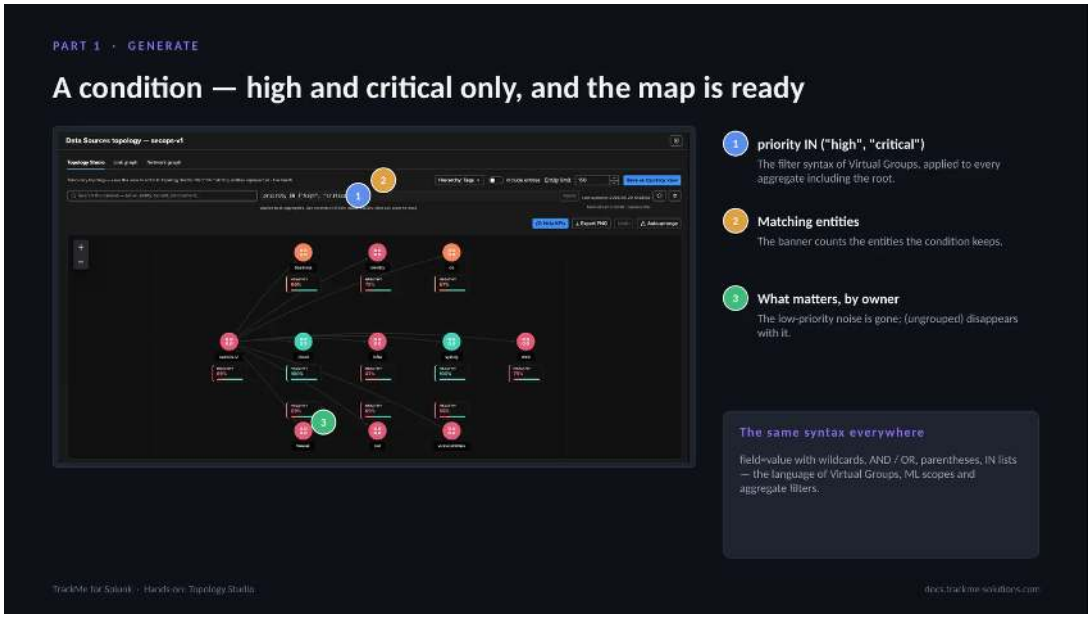
One aggregate per tag



With Tags as the hierarchy, the map shows the tenant as its owners see it. The root aggregate (callout 1) covers every monitored entity of the tenant. Beneath it sits one aggregate per tag (callout 2): cloud, firewall, identity, net, web and so on, each a live aggregate with its own colour. Entities without a tag are not lost; the (ungrouped) node (callout 3) collects them explicitly. The healthy percentage (callout 4) is the KPI of each aggregate; KPIs can be hidden or shown from the toolbar.

Auto-arrange lays the map out from one popover, with direction, spacing from compact to spacious and canvas size; the arrangement is a single undo step.

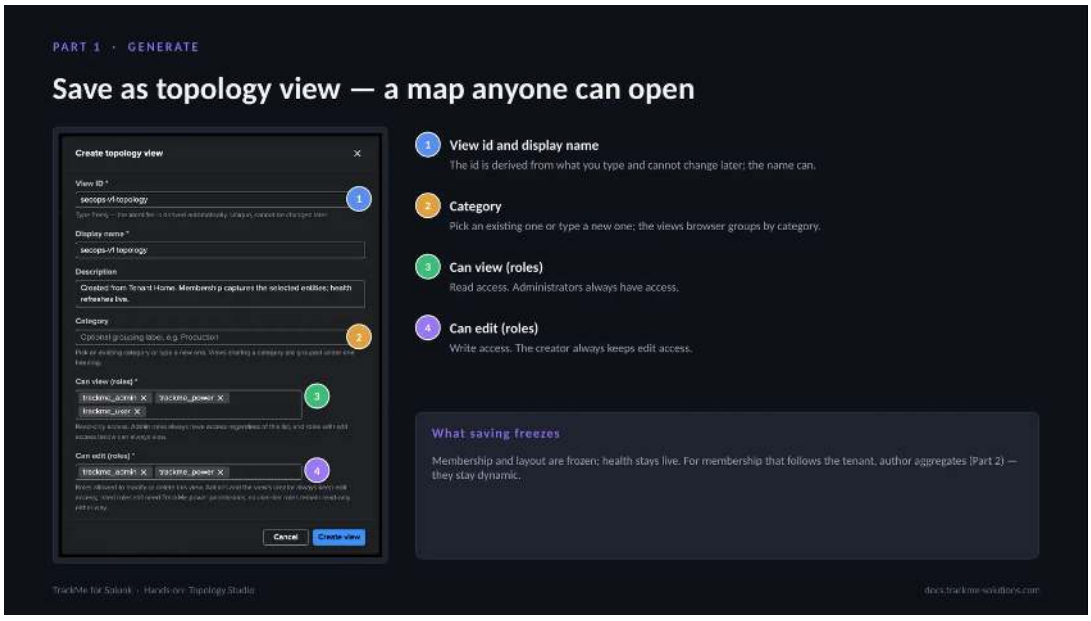
Narrowing the map with a condition



A condition narrows the whole map. The expression priority IN (“high”, “critical”) (callout 1) uses the filter syntax of Virtual Groups and is applied to every aggregate, including the root. The banner (callout 2) counts the matching entities the condition keeps. The result (callout 3) is what matters, by owner: the low-priority noise is gone, and the (ungrouped) node disappears with it. This is the on-call view a SecOps lead would put on a screen; clearing the condition and applying resets the map.

The same syntax is used everywhere in TrackMe: field=value with wildcards, AND / OR, parentheses and IN lists, the language of Virtual Groups, ML scopes and aggregate filters.

Saving the generated map as a view

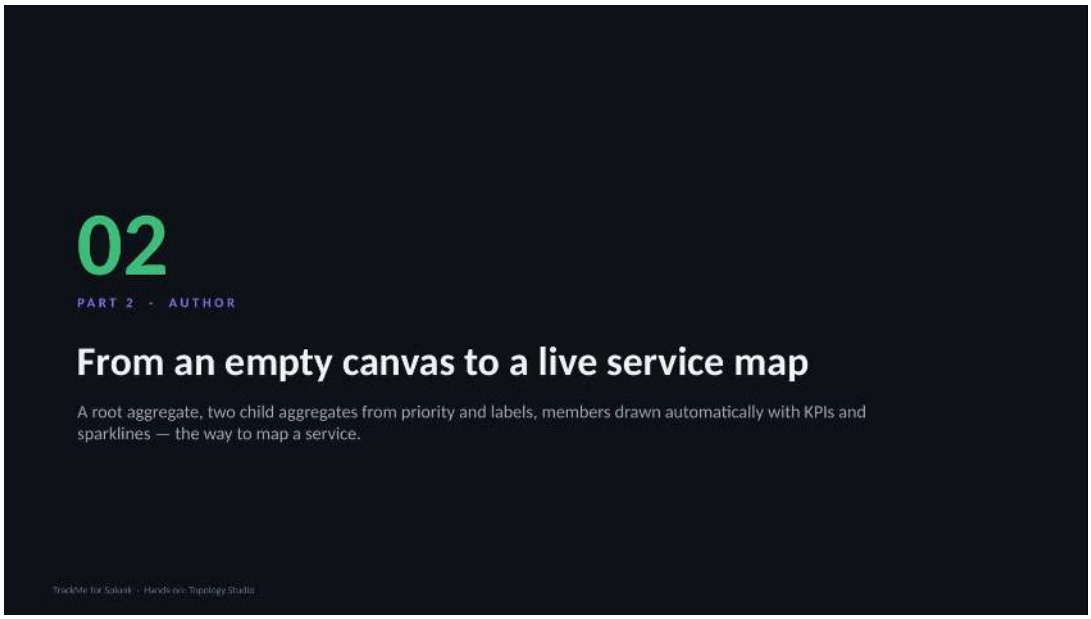


Save as topology view opens the create dialog. View id and display name (callout 1): the id is derived from what is typed and cannot change later, while the name can. Category (callout 2) accepts an existing value or a new one; the views browser groups views by category. Can view (callout 3) lists the roles with read access; administrators always

have access. Can edit (callout 4) lists the roles with write access; the creator always keeps edit access. A description can also be entered.

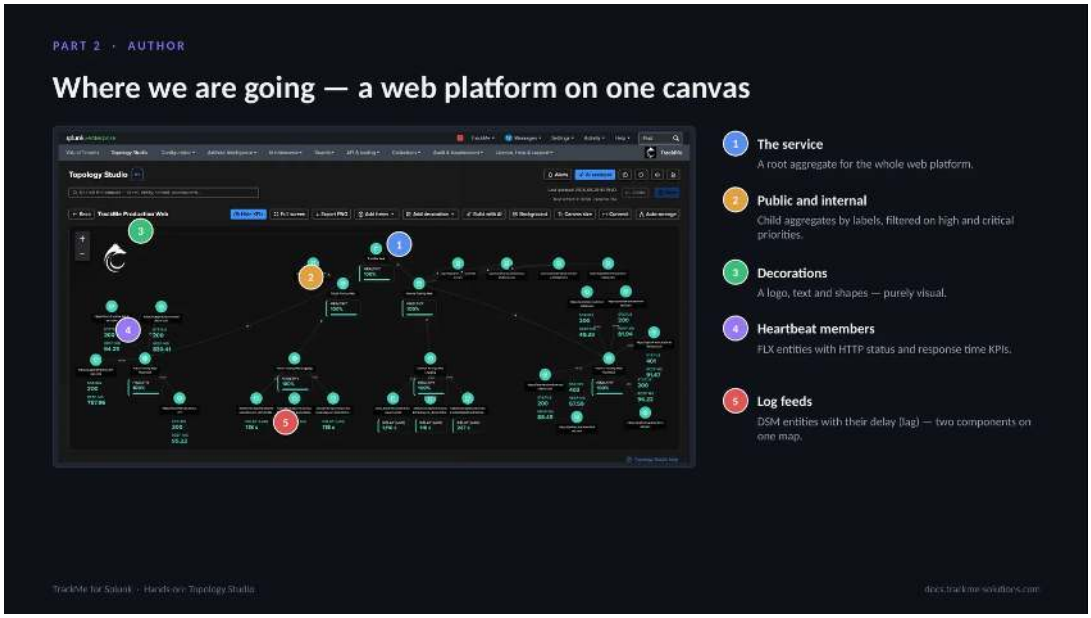
Saving freezes membership and layout while health stays live. For membership that follows the tenant as new entities appear, authored aggregates (Part 2) stay dynamic.

Part 2 — Author



Authoring by hand is the way most service maps are built: start from the service, describe its parts with aggregates, and let the members draw themselves. From an empty canvas, this part creates a root aggregate, two child aggregates built from priority and labels, and members drawn automatically with KPIs and sparklines, then links, arranges and saves the result as a live service map.

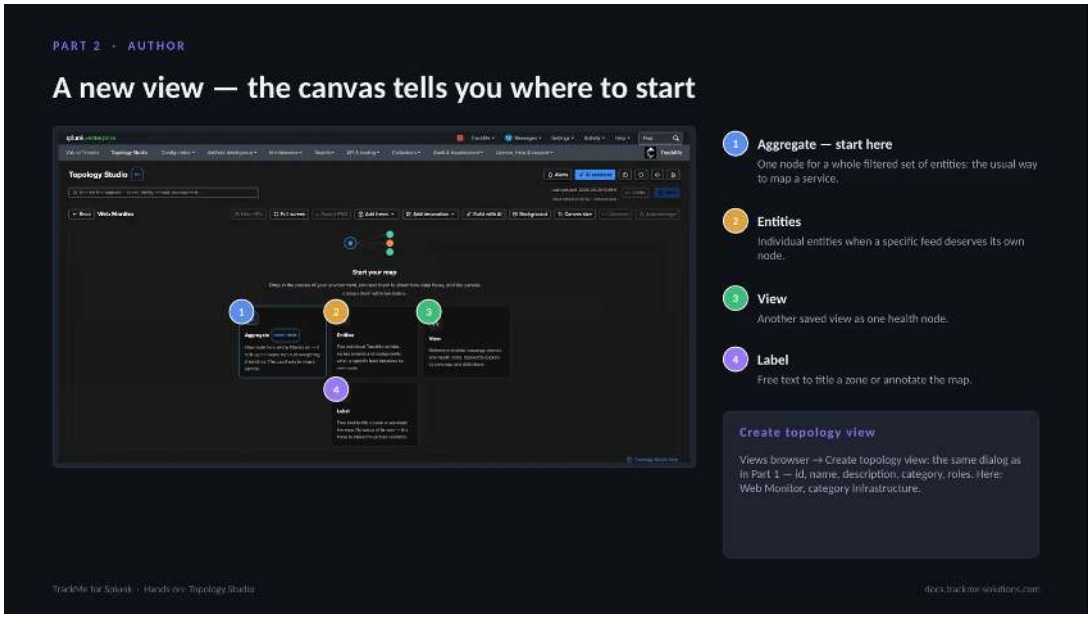
The target: a web platform map



The target is a map that runs in production, the web platform of TrackMe itself, rebuilt as a skeleton from an empty canvas in a few minutes. The service (callout 1) is a root aggregate for the whole web platform. Public and internal (callout 2) are child aggregates by labels, filtered on high and critical priorities. Decorations (callout 3), a logo, text

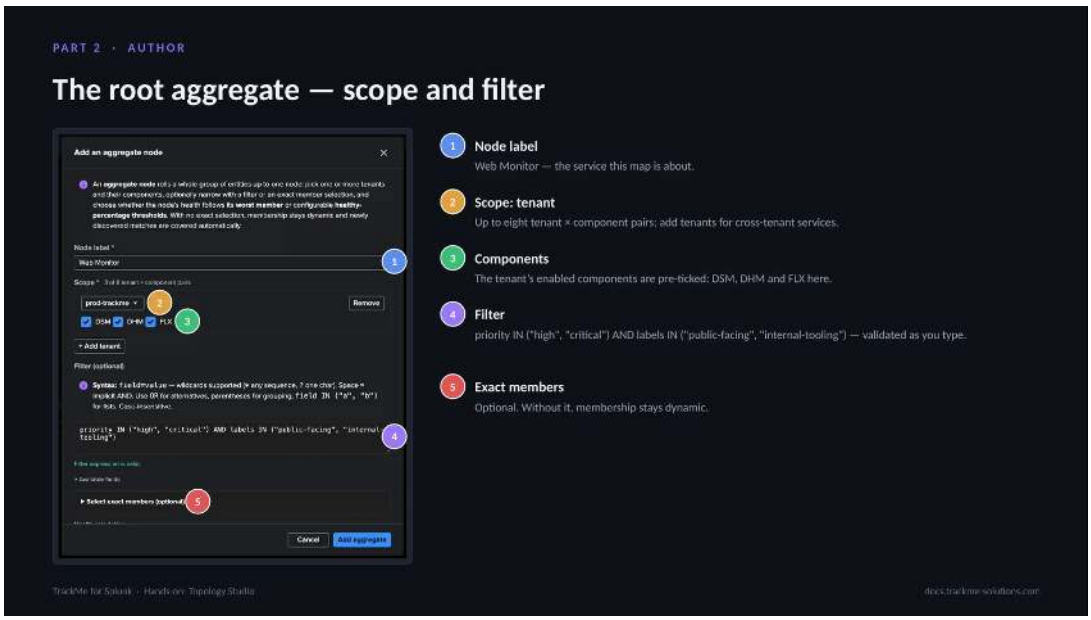
and shapes, are purely visual. Heartbeat members (callout 4) are FLX entities with HTTP status and response time KPIs. Log feeds (callout 5) are DSM entities with their delay (lag), so two components share one map.

Starting an empty view



A view called Web Monitor, category Infrastructure, is created from the views browser with Create topology view, the same dialog as in Part 1: id, name, description, category and roles. An empty view opens on the Start your map guide with four starting points. Aggregate (callout 1) is one node for a whole filtered set of entities, the usual way to map a service and the guide's recommendation. Entities (callout 2) adds individual entities when a specific feed deserves its own node. View (callout 3) adds another saved view as one health node. Label (callout 4) is free text to title a zone or annotate the map.

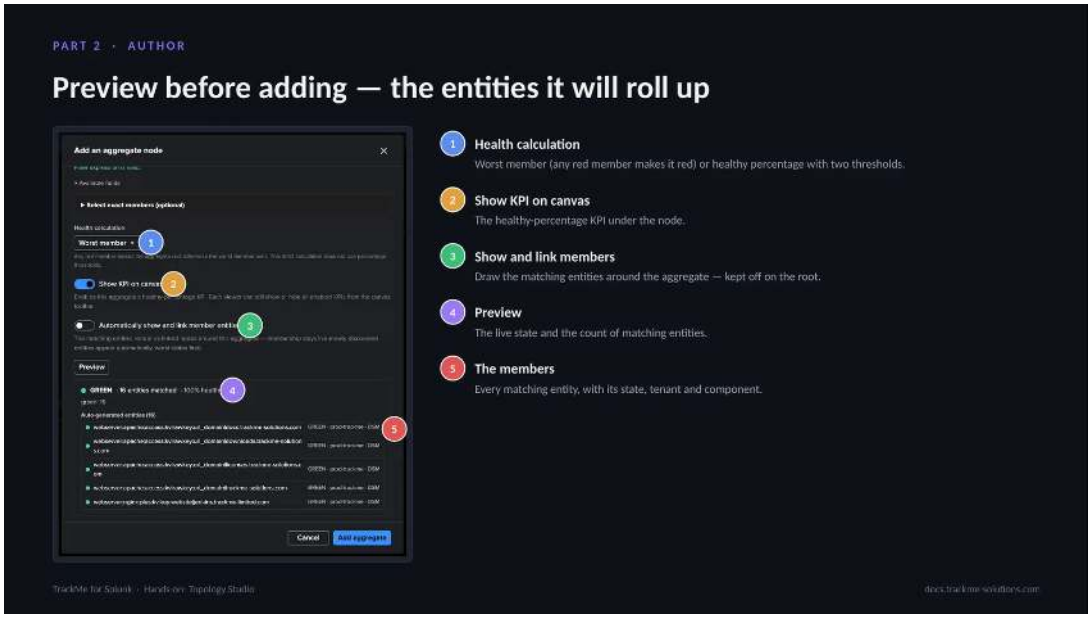
Defining the root aggregate



In the aggregate dialog, Node label (callout 1) is Web Monitor, the service the map is about. Scope: tenant (callout 2) takes up to eight tenant × component pairs; more tenants can be added for cross-tenant services. Components (callout 3) pre-ticks the tenant's enabled components, DSM, DHM and FLX here. Filter (callout 4) uses the same syntax as Part 1, validated as it is typed: priority IN ("high", "critical") AND labels IN ("public-facing", "internal-tooling"). The

labels come from the TrackMe label rules and the priorities from the policies. Exact members (callout 5) is optional; without it, membership stays dynamic and entities that start matching later join the aggregate by themselves.

Health calculation and member preview



Health calculation (callout 1) is either worst member, where any red member makes the aggregate red, or healthy percentage with two thresholds, by default green from 90 percent and orange from 75. Show KPI on canvas (callout 2) displays the healthy-percentage KPI under the node. Show and link members (callout 3) draws the matching entities around the aggregate; it is kept off on the root. Preview (callout 4) shows the live state and the count of matching entities, and the members list (callout 5) names every matching entity with its state, tenant and component, so the aggregate's coverage is known before anything is added to the canvas.

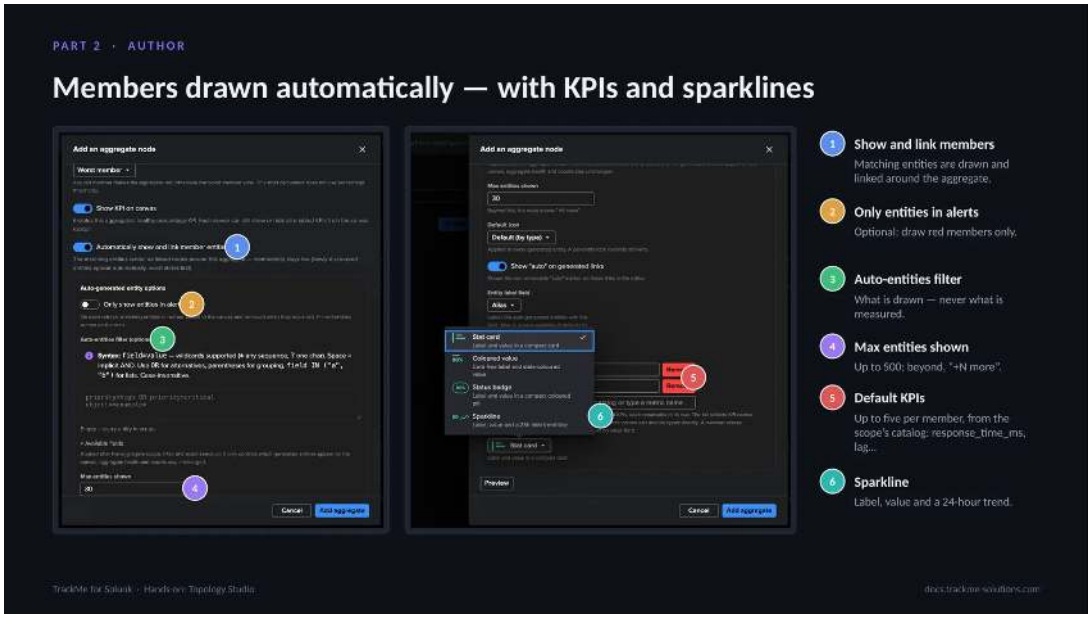
Adding the two child aggregates



The root aggregate Web Monitor (callout 1) sits on the canvas, green, with its healthy percentage. Its KPI (callout 2) reads HEALTHY 100%: every member is green. Add items (callout 3) offers entities, aggregate, auto aggregate, view and label. The Inspector (callout 4) shows aggregate details, live status and a members map for the selected node.

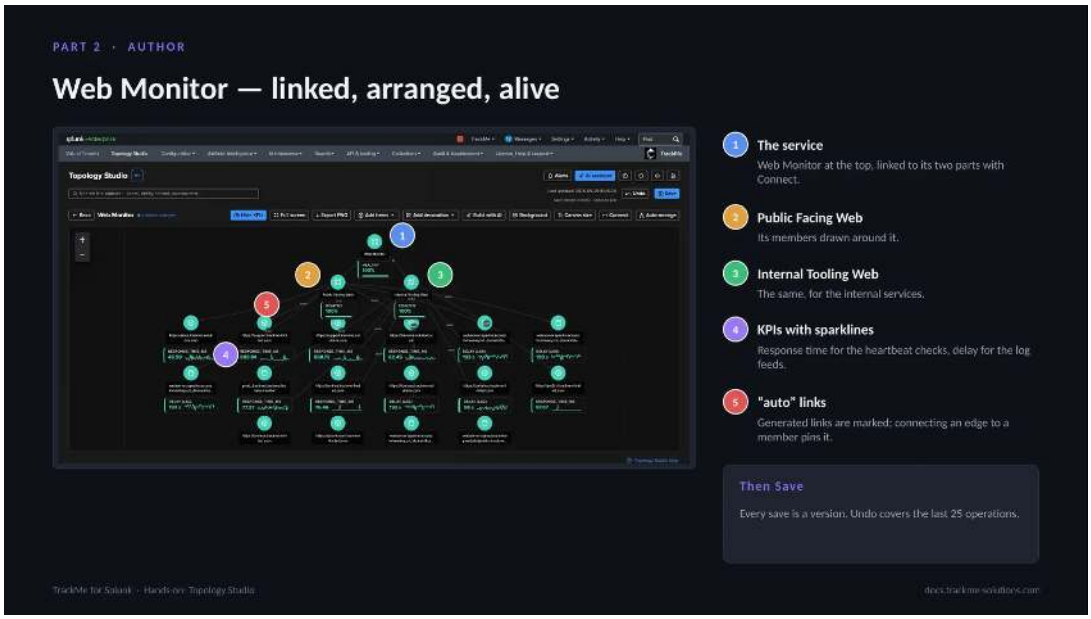
Two children are added with Add items → Add aggregate: Public Facing Web with labels IN (“public-facing”) and Internal Tooling Web with labels IN (“internal-tooling”), each with the same scope and priority filter as the root and its own label. On the children, automatic members are switched on.

Automatic members, KPIs and sparklines



Show and link members (callout 1) draws the matching entities around the aggregate and links them; membership stays live, worst states first. Only entities in alerts (callout 2) draws red members only and removes them when they recover. The Auto-entities filter (callout 3) decides what is drawn, never what is measured. Max entities shown (callout 4) goes up to 500; beyond that the node shows “+N more”. Default KPIs (callout 5) allows up to five per member from the scope’s catalog, here response_time_ms for the heartbeat checks and lag for the log feeds. The Sparkline style (callout 6) shows label, value and a 24-hour trend on every member.

Linking, arranging and saving



Web Monitor (callout 1) sits at the top, linked to its two parts with Connect: toggle it, click the source, then the target. Public Facing Web (callout 2) and Internal Tooling Web (callout 3) each have their members drawn around them. KPIs with sparklines (callout 4) show response time for the heartbeat checks and delay for the log feeds, with a

24-hour trend. Generated links are marked “auto” (callout 5); connecting an edge to a member pins it. Auto-arrange lays out the tree.

Nothing is stored until Save is clicked; every save becomes a version that can be restored, and Undo covers the last 25 operations.

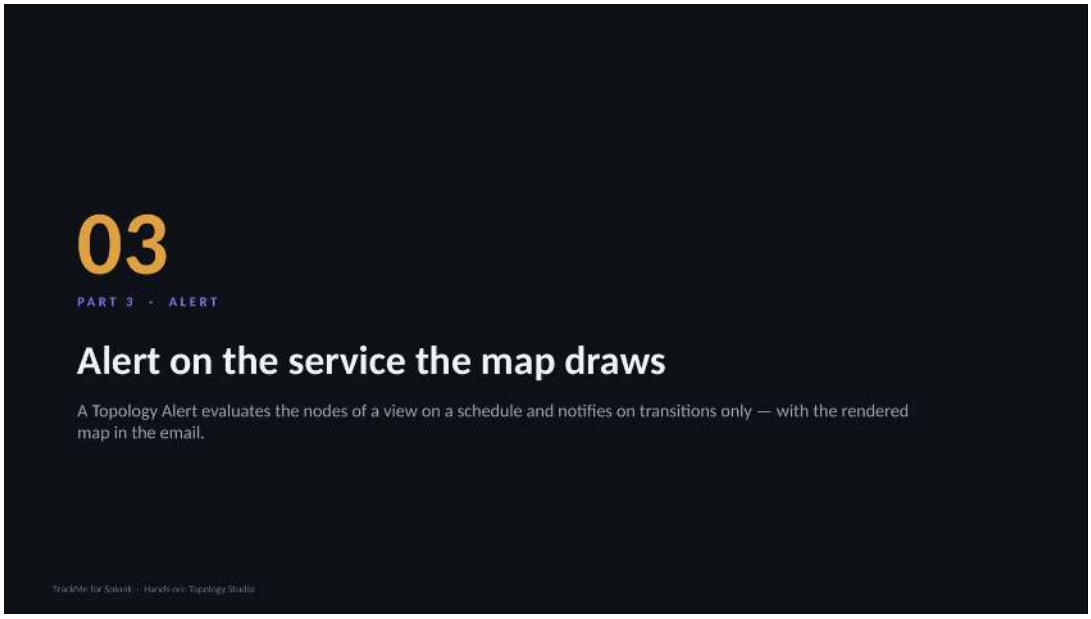
Inspecting a member node



Selecting a generated member (callout 1), one of the heartbeat checks of Public Facing Web, opens the inspector. Entity details (callout 2) give the entity, tenant and component, here an FLX entity. View status, View details and Open in Tenant Home (callout 3) lead from the map to the status message, the entity and its tenant. Live status (callout 4) shows the state and the current value of each KPI.

The inspector is where the map becomes operational: the status is the same status message and score breakdown as in Tenant Home, and the entity can be opened directly in its tenant.

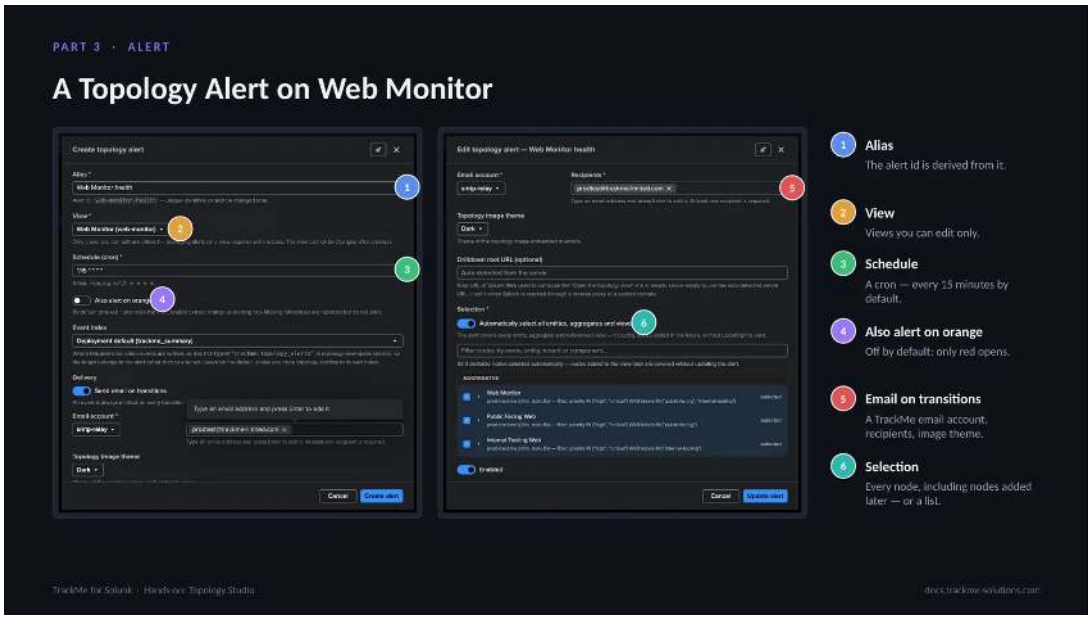
Part 3 — Alert



A map worth watching is a map worth alerting on. A Topology Alert is attached to a view, evaluates the nodes of that view on a schedule and notifies on transitions only: open, update and resolve. It speaks only when the set of unhealthy

items changes, and the email carries the rendered map.

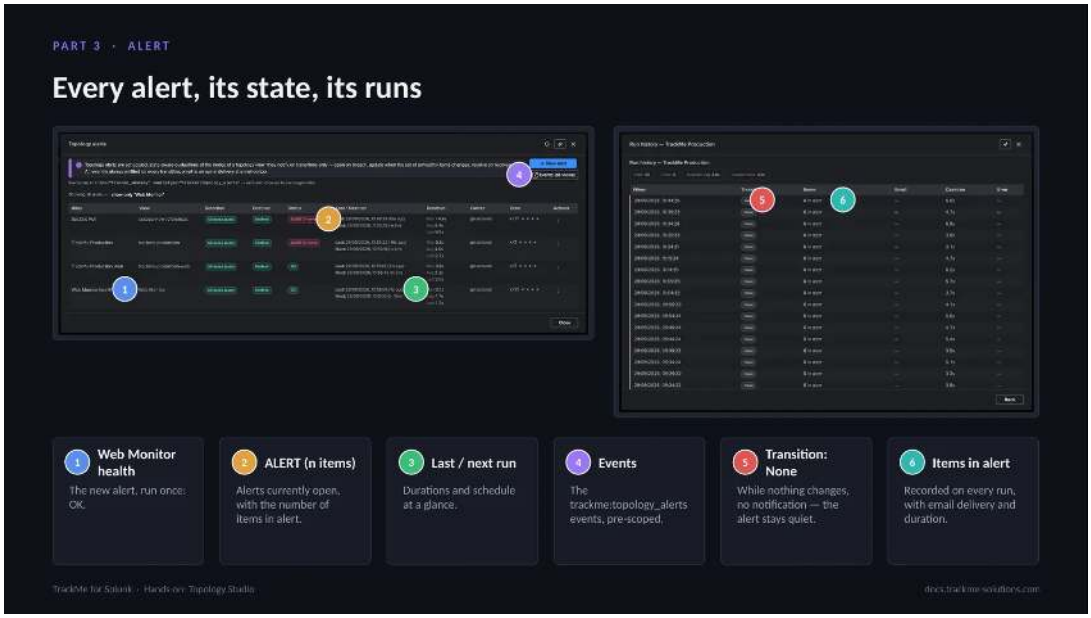
Creating a Topology Alert



From the Alerts button in the view header, New alert opens the alert form. Alias (callout 1): the alert id is derived from it. View (callout 2) lists only views the user can edit. Schedule (callout 3) is a cron, every 15 minutes by default. Also alert on orange (callout 4) is off by default, so only red opens an alert. Email on transitions (callout 5) takes a TrackMe email account, recipients and an image theme, dark or light, with the rendered topology embedded. Selection (callout 6) covers every entity, aggregate and referenced view of the map, including nodes added later, or an explicit list.

Every run writes a trackme:topology_alerts event on each transition; email is an extra channel.

Alert list and run history

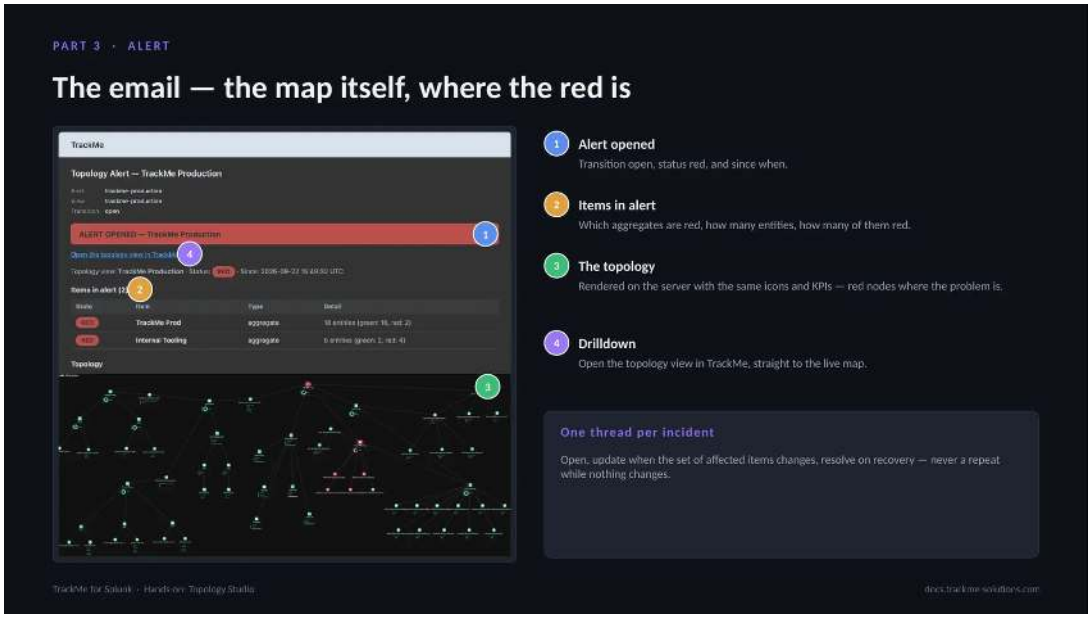


The alerts list covers every view. Web Monitor health (callout 1) is the new alert, run once: OK. ALERT (n items) (callout 2) marks alerts currently open with the number of items in alert; here two alerts are open, one with seven items. Last / next run (callout 3) shows durations and schedule. Events (callout 4) opens the trackme:topology_alerts events, pre-scoped. Transition: None (callout 5) is the key property of an open alert: while the set of affected items

does not change, no notification is sent. Items in alert (callout 6) is recorded on every run, with email delivery and duration.

Row actions include Run now, Alert events, Logs, History and Ask AI.

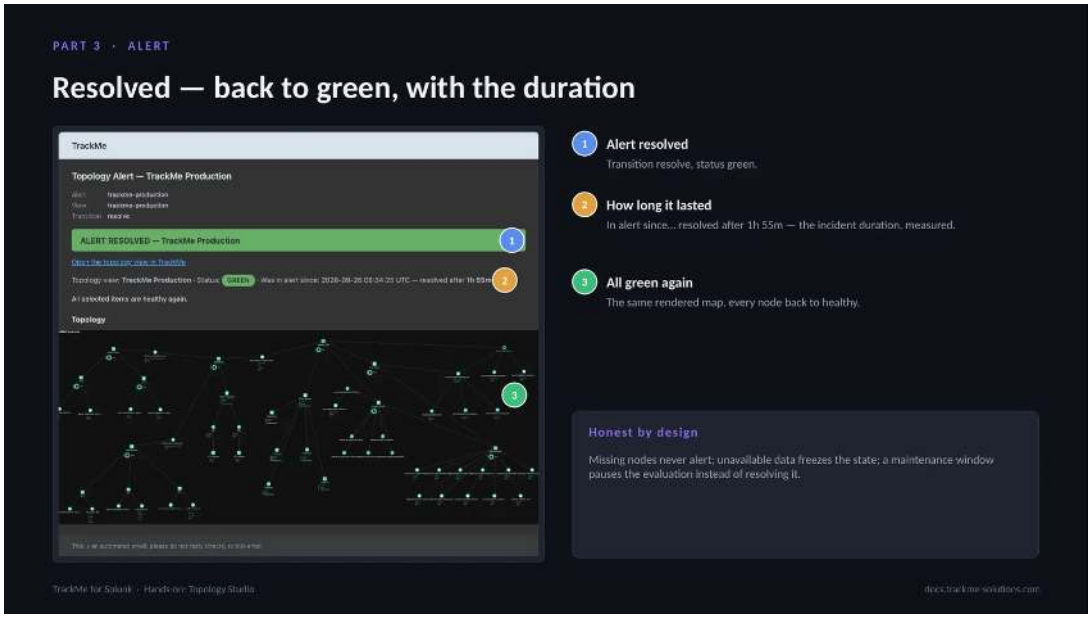
The alert-opened email



The email of a Topology Alert on the TrackMe Production view. Alert opened (callout 1) states the transition open, the status red and since when. Items in alert (callout 2) lists which aggregates are red, how many entities they hold and how many of those are red; when this alert opened, two aggregates were red. The topology (callout 3) is rendered on the server with the same icons and KPIs as the canvas, red nodes where the problem is. Drilldown (callout 4) opens the topology view in TrackMe, straight to the live map.

Each incident is one thread: open, update when the set of affected items changes, resolve on recovery, never a repeat while nothing changes.

The resolution email

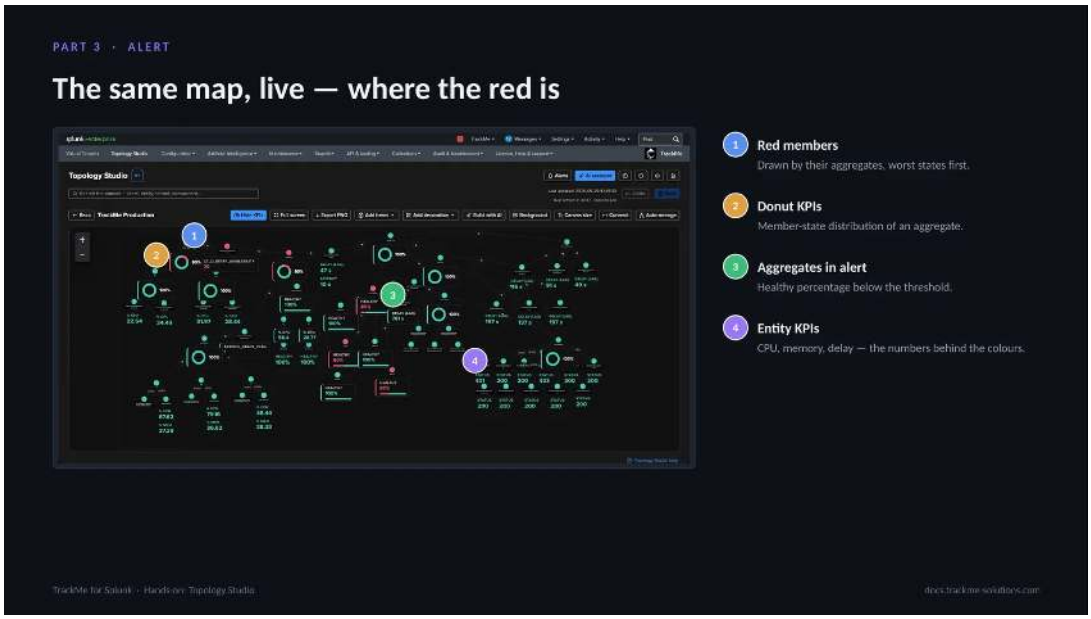


The resolution email of the same alert. Alert resolved (callout 1) states the transition resolve and the status green. How long it lasted (callout 2) reads “in alert since… resolved after 1h 55m”, the measured incident duration. All

green again (callout 3) is the same rendered map with every node back to healthy.

The alert state is honest by design: missing nodes never alert, unavailable data freezes the state, and a maintenance window pauses the evaluation instead of resolving it. An alert is never resolved by a data gap or a maintenance window.

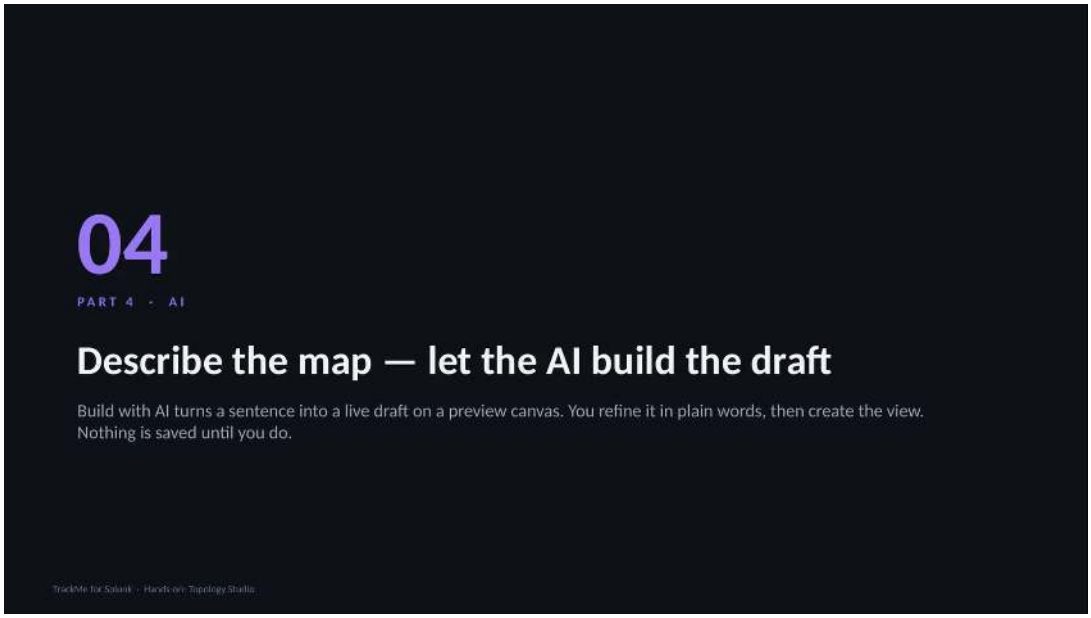
The live view during an alert



The TrackMe Production view while its alert is open: thirty-five nodes over several tenants and components. Red members (callout 1) are drawn by their aggregates, worst states first. Donut KPIs (callout 2) show the member-state distribution of an aggregate. Aggregates in alert (callout 3) have a healthy percentage below the threshold. Entity KPIs (callout 4), CPU, memory and delay, are the numbers behind the colours.

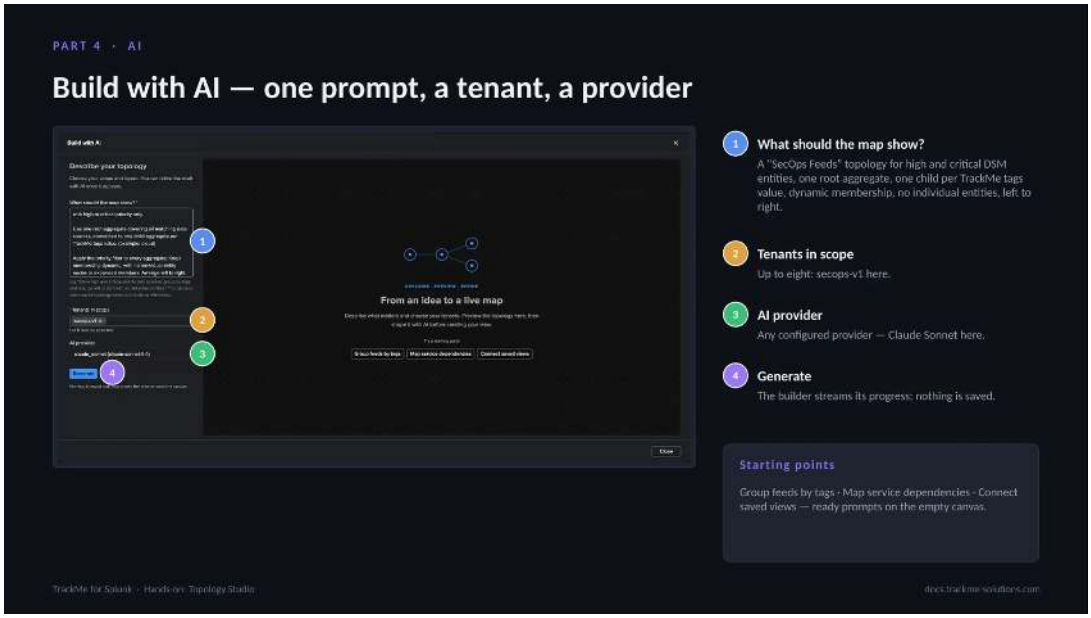
The red is where the email placed it: the live map and the email tell the same story because they are the same map.

Part 4 — AI



Build with AI turns a sentence into a live draft on a preview canvas. A dedicated AI builder agent reads the inventory of the selected tenants and proposes a draft, compiled and validated like a manual save. The draft is refined in plain words and then created as a view; nothing is saved until that point.

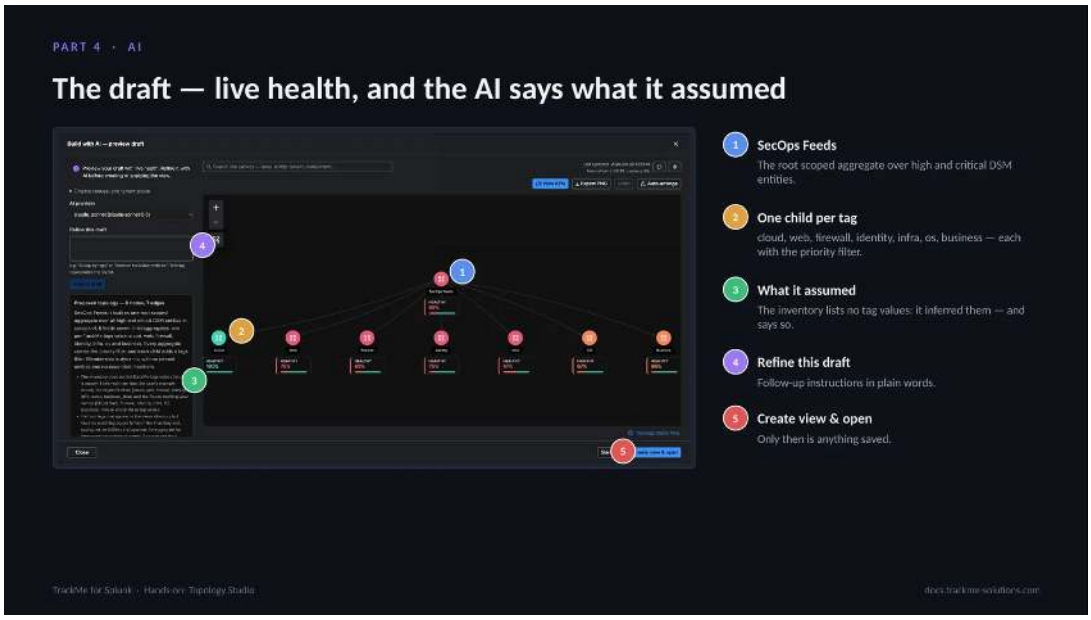
Prompt, tenants and provider



The prompt (callout 1) asks for a "SecOps Feeds" topology for high and critical DSM entities: one root aggregate, one child aggregate per TrackMe tags value, the priority filter on every aggregate, dynamic membership, no individual entities, laid out left to right. Tenants in scope (callout 2) accepts up to eight tenants, secops-v1 here. AI provider (callout 3) is any provider configured in TrackMe, Claude Sonnet in this case. Generate (callout 4) starts the builder, which streams its progress; nothing is saved.

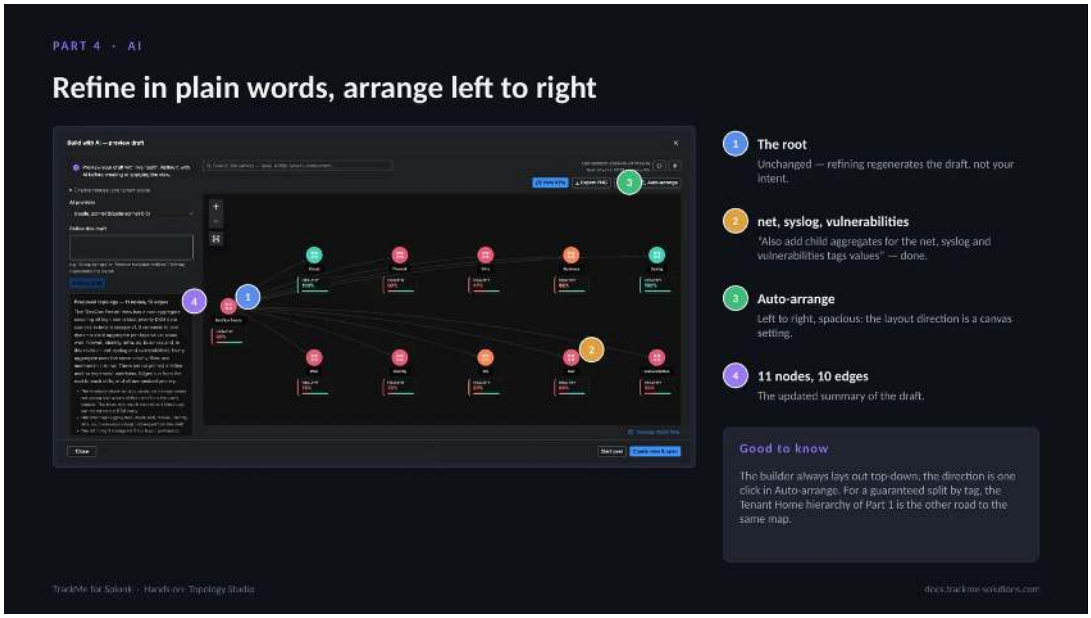
The empty canvas offers ready prompts as starting points: Group feeds by tags, Map service dependencies and Connect saved views.

Reading the first draft



The first draft arrives in about twenty seconds on a live preview canvas with real health. SecOps Feeds (callout 1) is the root scoped aggregate over high and critical DSM entities. One child per tag (callout 2): cloud, web, firewall, identity, infra, os and business, each carrying the priority filter. What it assumed (callout 3): the inventory the builder receives carries priorities and labels but no tag values, so it inferred them from the data source families, left out three tags it could not confirm, and says so in its summary. Refine this draft (callout 4) takes follow-up instructions in plain words. Create view & open (callout 5) is the only step that saves anything.

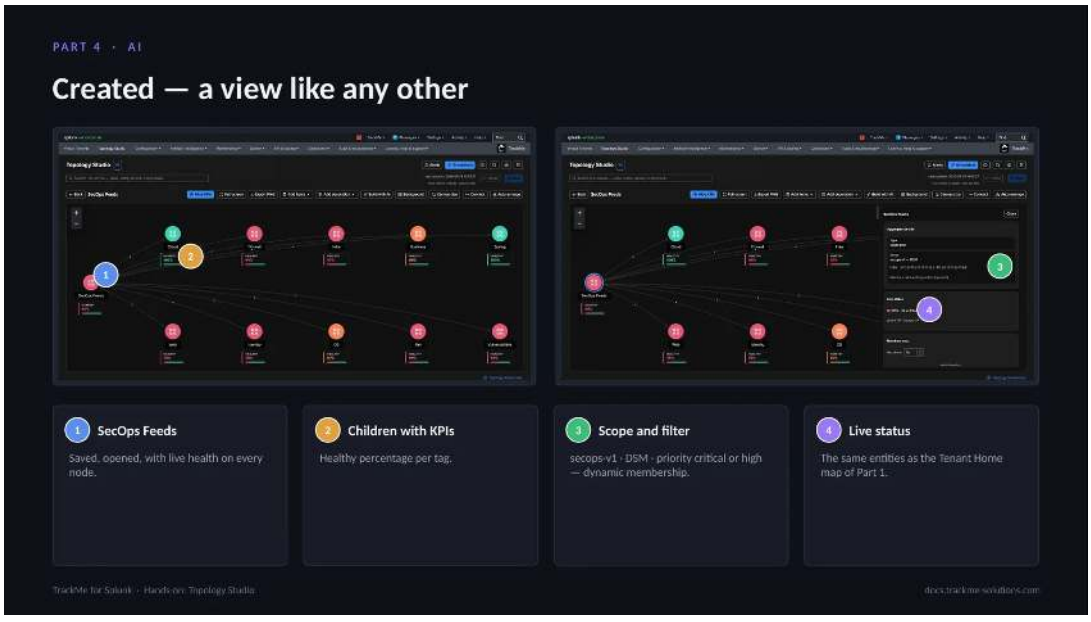
Refining the draft



One refinement adds the missing tags: "Also add child aggregates for the net, syslog and vulnerabilities tags values". The root (callout 1) is unchanged, because refining regenerates the draft, not the intent. The three new children (callout 2) appear; the draft updates in place, and a failed refinement would keep the previous draft. Auto-arrange (callout 3) switches the layout to left to right, spacious: the builder always lays out top-down, and the direction is a canvas setting, one click away. The summary (callout 4) now reads 11 nodes, 10 edges.

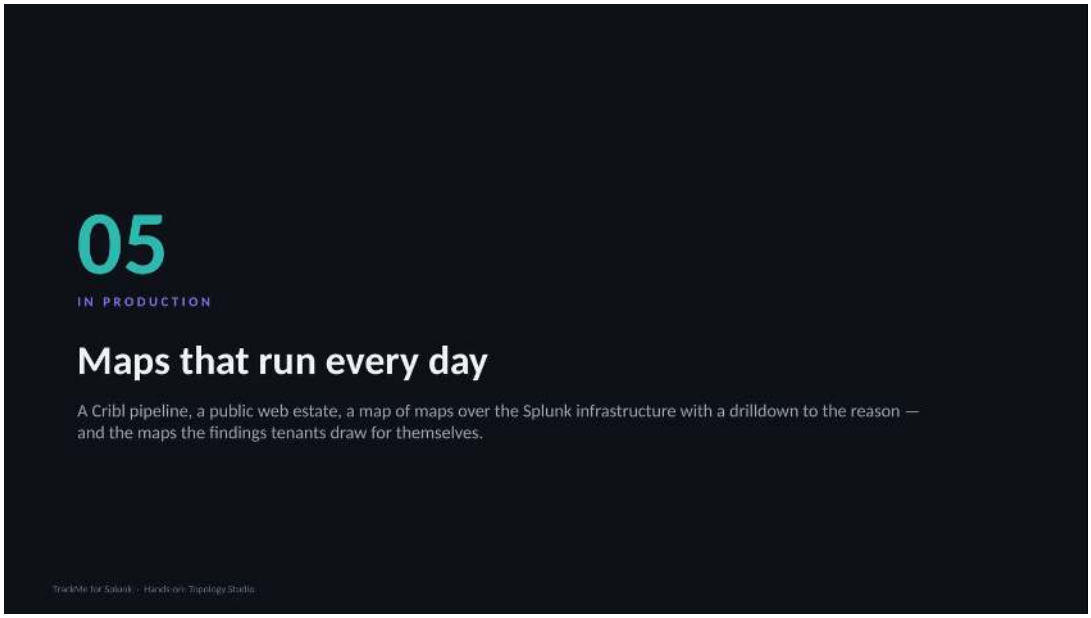
For a guaranteed split by tag, the Tenant Home hierarchy of Part 1 produces the same map: two roads to the same destination.

The created view



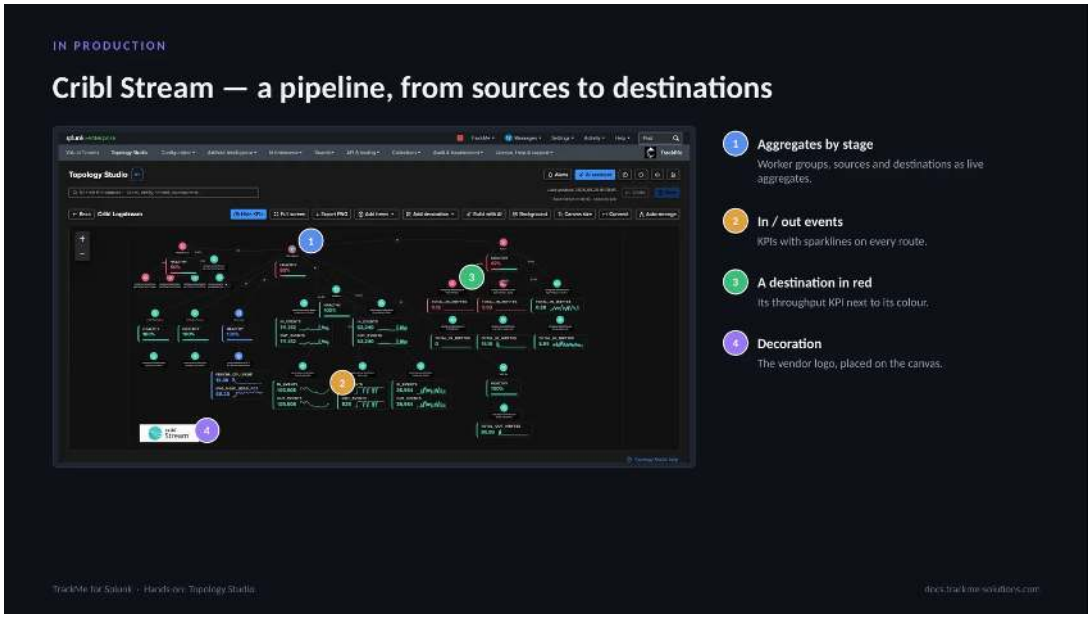
Create view & open saves the draft as a regular view: editable, versioned, shareable and alertable. SecOps Feeds (callout 1) is saved and opened, with live health on every node. Children with KPIs (callout 2) show the healthy percentage per tag. Scope and filter (callout 3) in the root's inspector read secops-v1 · DSM · priority critical or high, with dynamic membership, exactly as the AI wrote them. Live status (callout 4) covers the same entities as the Tenant Home map of Part 1.

In production



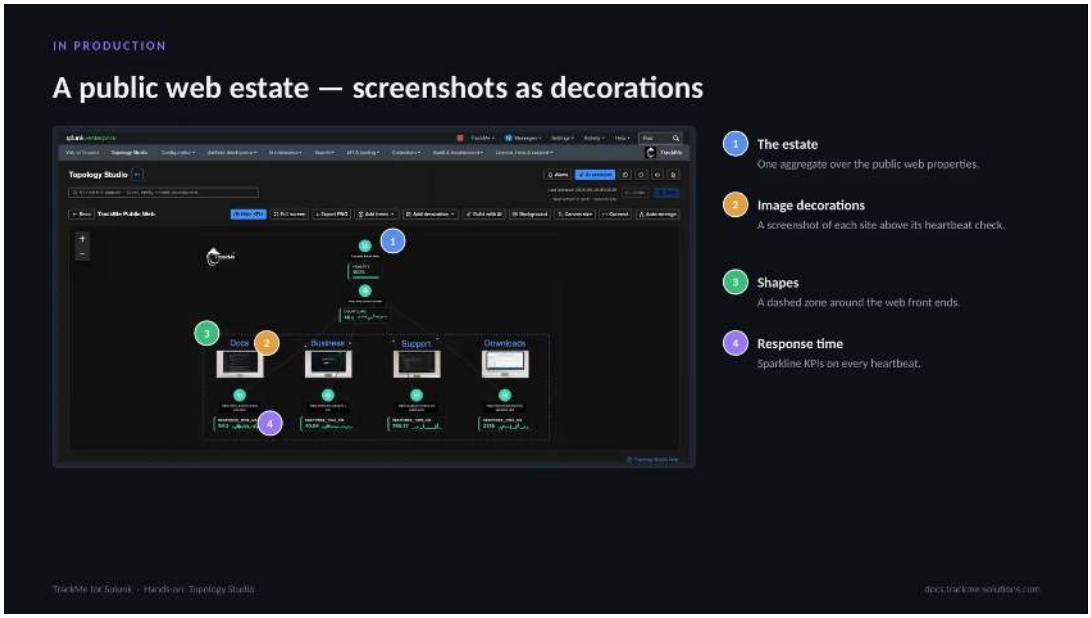
Three maps run every day on the deployment the screens were captured from: a Cribl pipeline mapped from sources to destinations, a public web estate with screenshots as decorations, and a map of maps over the Splunk infrastructure, with a drilldown from the top view to the reason for a red node. The findings tenants then draw their own maps.

A Cribl Stream pipeline map



A Cribl Stream pipeline mapped from TrackMe entities. Aggregates by stage (callout 1) represent worker groups, sources and destinations as live aggregates. In / out events (callout 2) are KPIs with sparklines on every route. A destination in red (callout 3) shows its throughput KPI next to its colour. The decoration (callout 4) is the vendor logo, placed on the canvas. With events in and out and throughput as KPIs, the map reads like the pipeline itself.

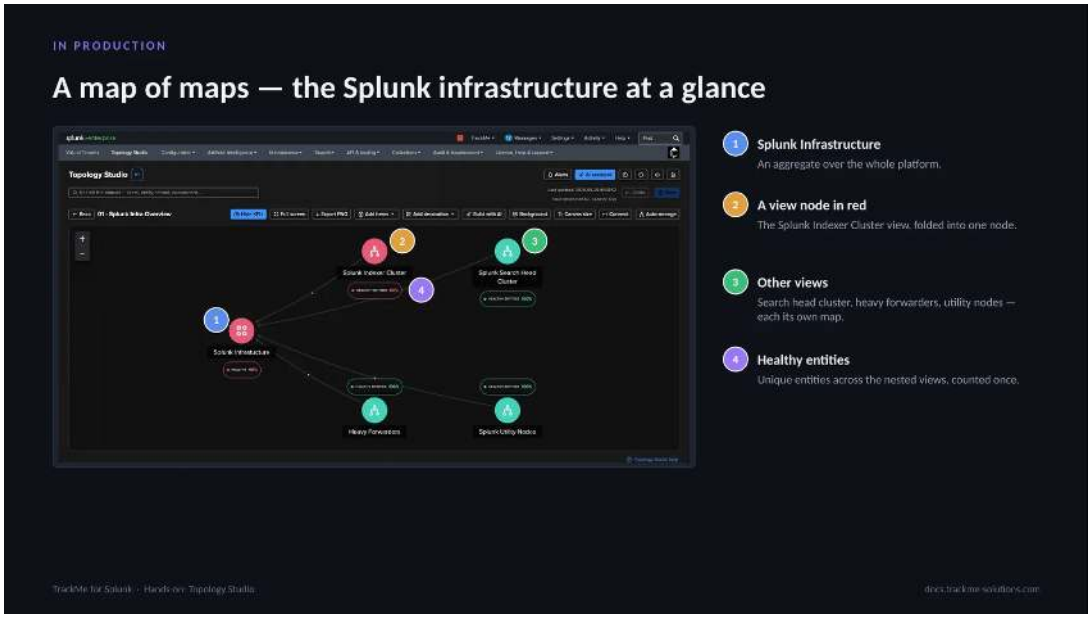
Decorations on a web estate map



The estate (callout 1) is one aggregate over the public web properties. Image decorations (callout 2) place a screenshot of each site above its heartbeat check. Shapes (callout 3) draw a dashed zone around the web front ends. Response time (callout 4) is a sparkline KPI on every heartbeat.

Decorations turn a map into a picture people recognise. They never affect health, and they appear in PNG exports and in alert emails.

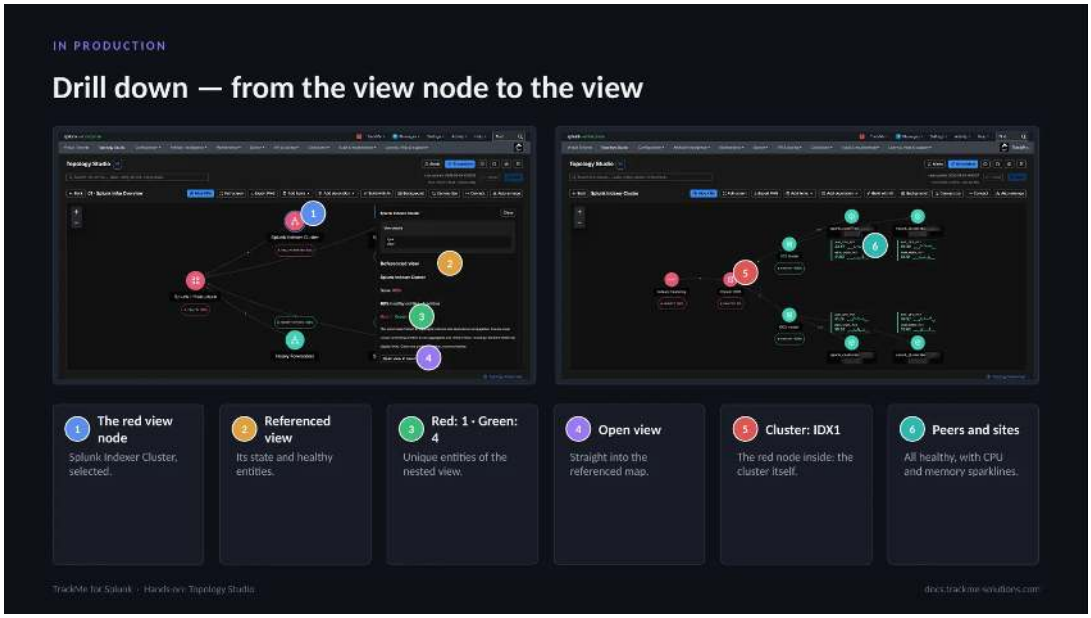
Nesting views: the Splunk infrastructure



The top view of the Splunk infrastructure. Splunk Infrastructure (callout 1) is an aggregate over the whole platform. A view node in red (callout 2) is the Splunk Indexer Cluster view, folded into one node. Other views (callout 3), search head cluster, heavy forwarders and utility nodes, are each their own map. Healthy entities (callout 4) counts unique entities across the nested views, each counted once.

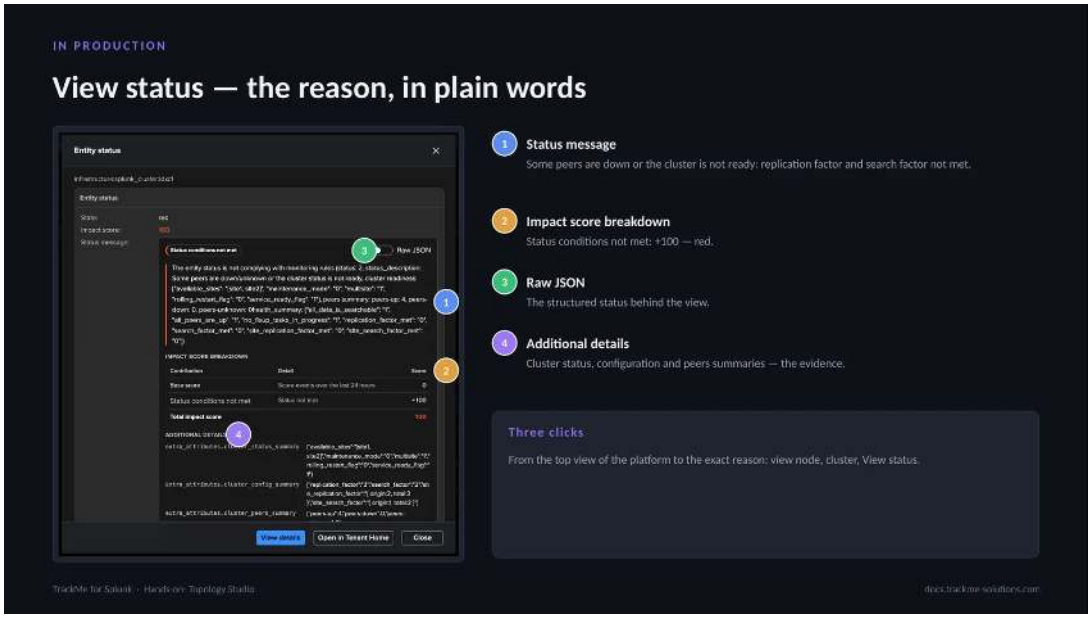
A view node counts the unique underlying entities across nested views and follows their policies; one of the nested views is red at the time of capture.

Drilling into a view node



The drilldown starts by selecting the red view node (callout 1), Splunk Indexer Cluster. Referenced view (callout 2) shows its state and healthy entities. Red: 1 · Green: 4 (callout 3) counts the unique entities of the nested view: one red entity out of five. Open view (callout 4) leads straight into the referenced map. Inside, Cluster: IDX1 (callout 5) is the red node, the cluster itself. Peers and sites (callout 6) are all healthy, with CPU and memory sparklines. The red belongs to the cluster entity, not to any peer.

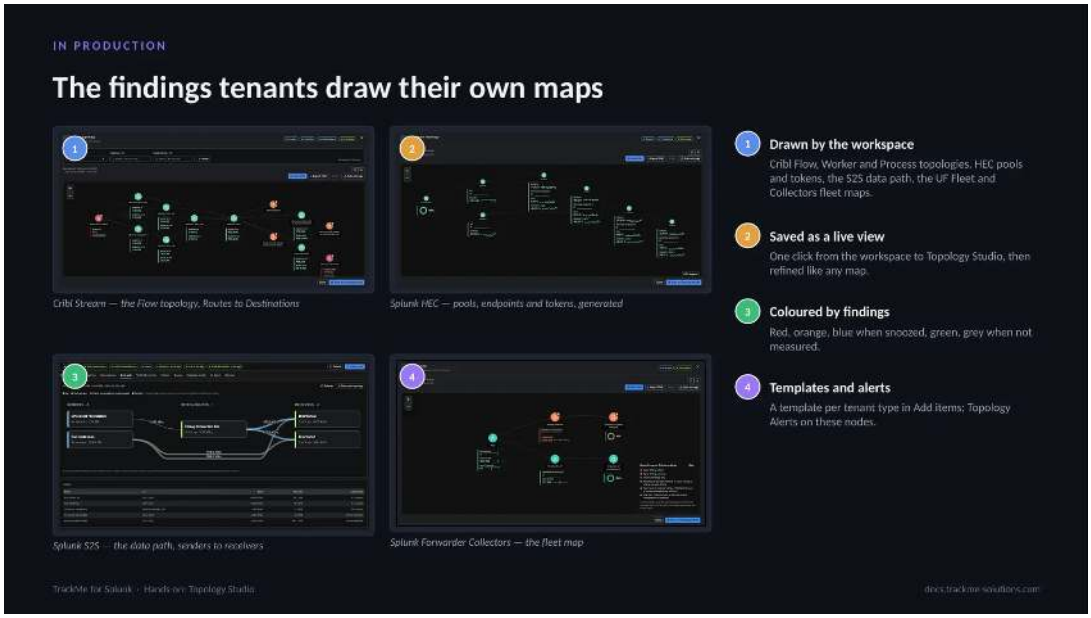
Reading the entity status



View status on the red member opens the entity status. The Status message (callout 1) reads that some peers are down or the cluster is not ready: replication factor and search factor not met. The Impact score breakdown (callout 2) shows Status conditions not met: +100, which makes the entity red. Raw JSON (callout 3) is the structured status behind the view. Additional details (callout 4) hold the cluster status, configuration and peers summaries, the evidence.

From the top view of the platform to the exact reason takes three clicks: view node, cluster, View status. This is the same status message the alert and the AI Assistant use.

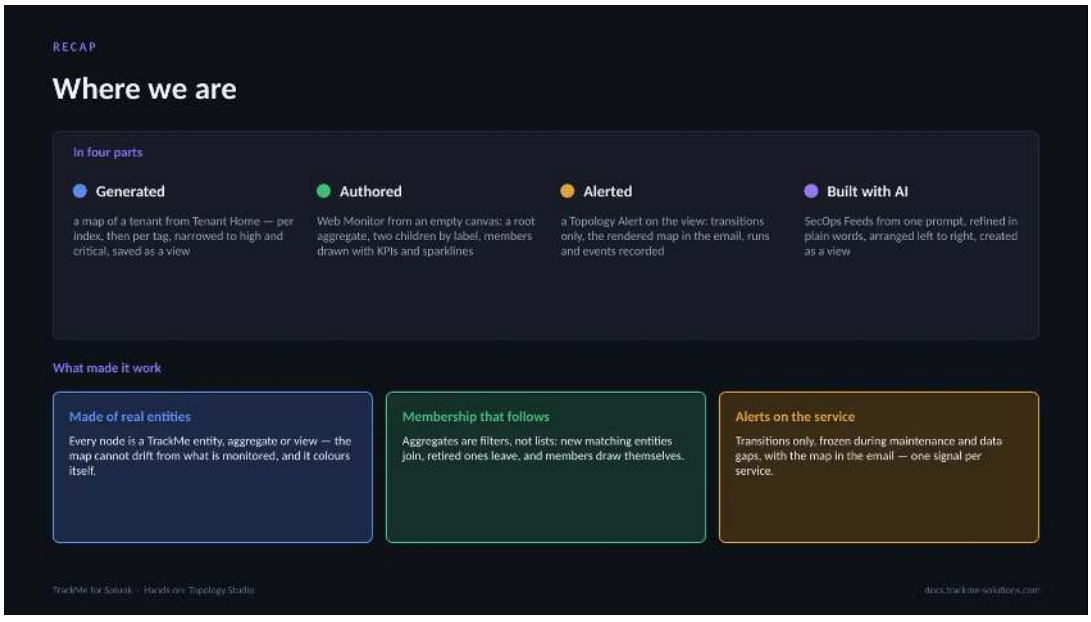
Maps of the findings tenants



Six dedicated tenant types draw their maps for you: the Cribl Stream Flow, Worker and Process topologies (1), the Splunk HEC pools, endpoints and tokens generated in one click (2), the Splunk S2S data path from the senders to the receivers (3), and the Splunk UF Fleet and Splunk Forwarder Collectors fleet maps (4).

Each one is saved from its workspace to Topology Studio as a live view and refined like any map. Its nodes are coloured by their findings: red for a critical finding, orange for another, blue when only snoozed, green when measured without a finding, grey when not measured. Add items offers a template per tenant type, and Topology Alerts work on these nodes too.

Where we are



Four ways to a live map. Generated: a map of a tenant from Tenant Home, per index then per tag, narrowed to high and critical. Authored: Web Monitor from an empty canvas, a root aggregate, two children by label, members with KPIs and sparklines. Alerted: a Topology Alert on the view, transitions only, the rendered map in the email. Built with AI: SecOps Feeds from one prompt, refined in plain words.

What made it work: maps are made of real entities, aggregates or views, so they cannot drift from what is monitored. Aggregates are filters, not lists: matching entities join, retired ones leave. Alerts fire on transitions only, frozen during

maintenance and data gaps: one signal per service.

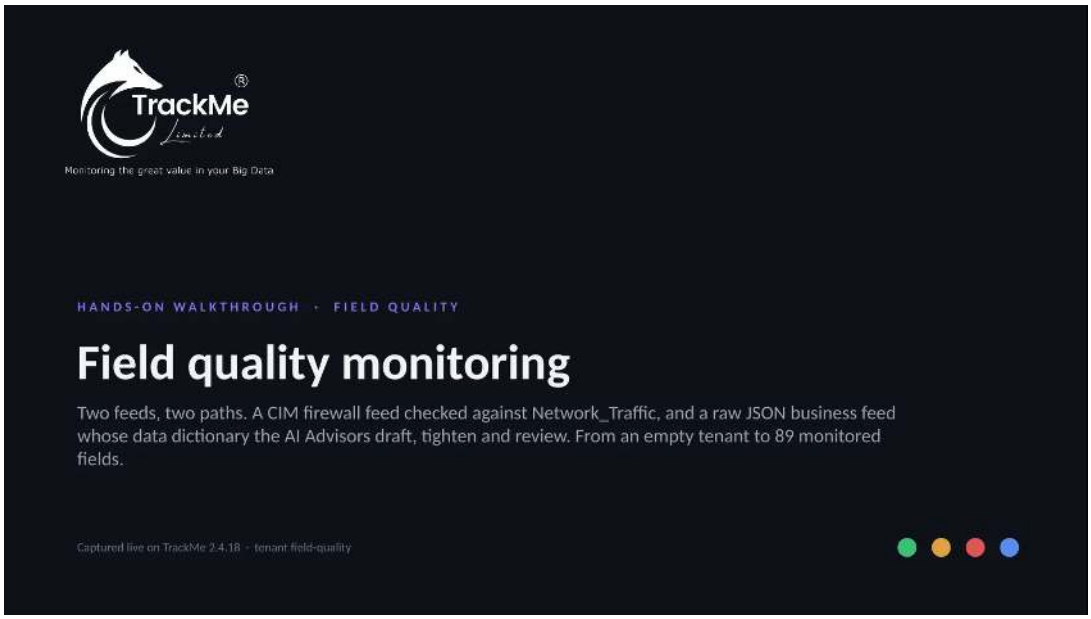
2.1.6 Field Quality Monitoring

Field extraction success and CIM compliance measured per field: CIM data models, AI-generated dictionaries for custom sourcetypes, sampling, and the field-quality advisors.

`Download the deck (PDF) · PowerPoint`

about 40 minutes · 35 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



Field Quality Monitoring (FQM) follows two feeds down two paths. A CIM firewall feed, Cisco ASA logs, is checked against the Network_Traffic data model with a dictionary generated from CIM. A raw JSON business feed, e-commerce orders, has no CIM model: its data dictionary is drafted, tightened and reviewed by the AI Advisors. Both start from a brand new tenant, field-quality, and end with 89 monitored fields.

Every screen was captured on TrackMe 2.4.18. The walkthrough closes with what a mature FQM tenant looks like after several months of history.

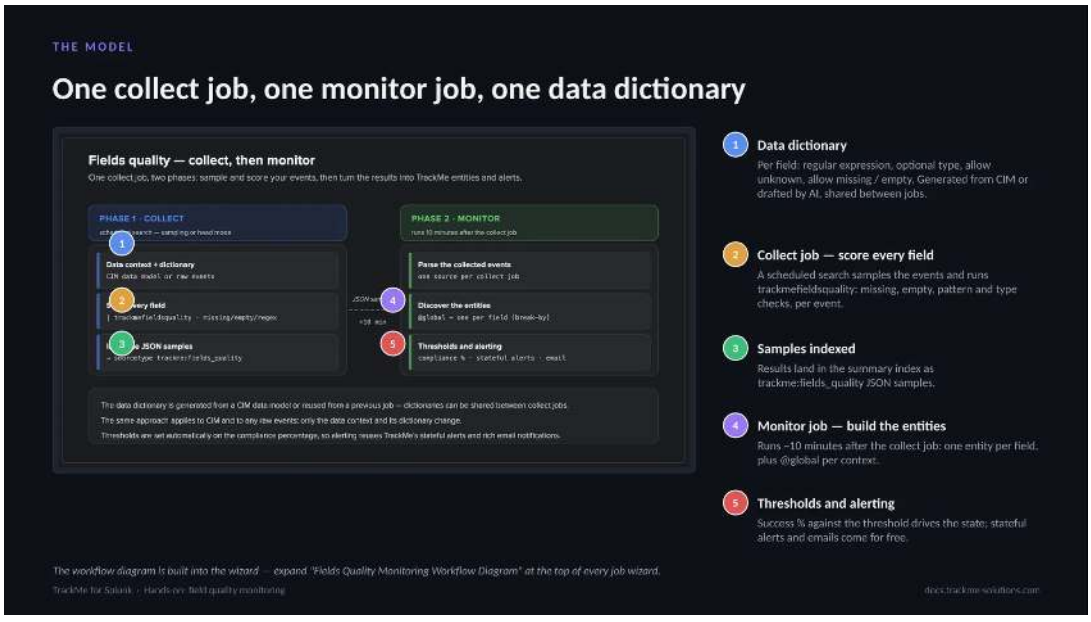
Why availability monitoring is not enough



Data source monitoring sees volume and latency; detections, dashboards and business reports depend on the values inside each event. A field stops parsing when a TA or parser change leaves it empty. A value drifts from the standard, EURO instead of EUR: the field exists, the value is wrong. A field goes missing on one region or one message type; averages hide it, per-field quality shows it.

FQM samples the events of a feed, checks every field against a data dictionary (presence, regular expression, optional type) and turns the result into TrackMe entities: one per field plus one @global per context, each with a threshold, a state, a score, history and alerting.

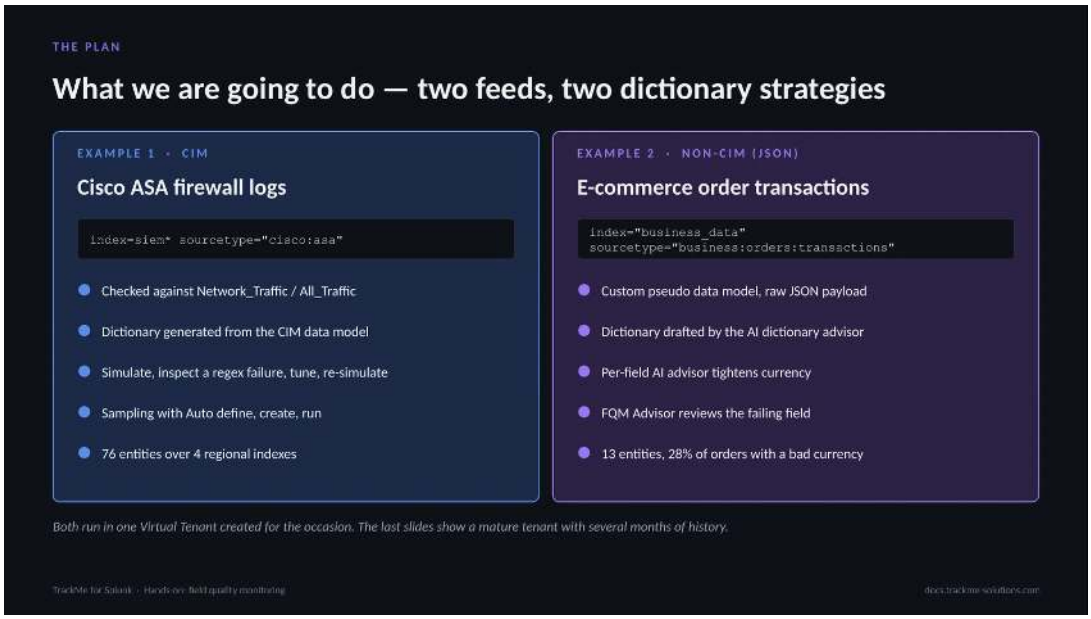
Dictionary, collect job and monitor job



The Data dictionary (callout 1) holds, per field, a regular expression, an optional type, allow unknown and allow missing / empty; it is generated from CIM or drafted by AI. The Collect job (callout 2) is a scheduled search that samples the events and runs trackmefieldsquality against the dictionary. The samples (callout 3) land in the summary index as trackme:fields_quality JSON. The Monitor job (callout 4) runs about 10 minutes later and builds one entity per field plus @global per context. Thresholds and alerting (callout 5): the success % against the threshold drives the state; stateful alerts and emails follow.

Both jobs are created and scheduled by the wizard; the dictionary is reusable across trackers that share a format.

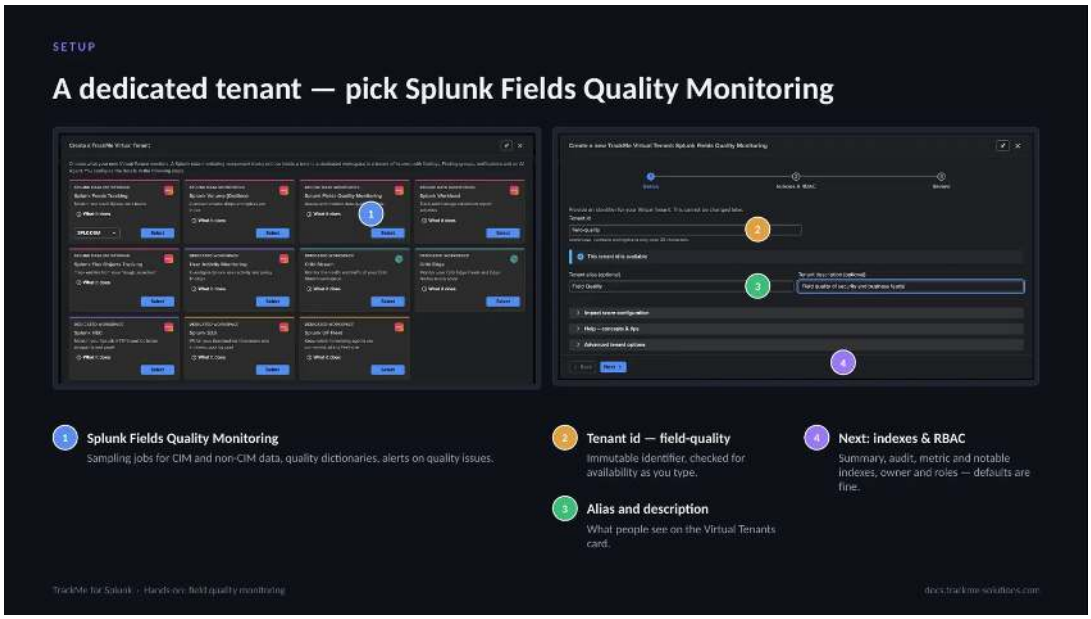
Two feeds, two dictionary strategies



Example 1 is the CIM path: Cisco ASA firewall logs, index=siem* sourcetype="cisco:asa", checked against Network_Traffic / All_Traffic. The dictionary is generated from the CIM data model; the flow is simulate, inspect a regex failure, tune, re-simulate, then sampling with Auto define, create and run. The result is 76 entities over 4 regional indexes.

Example 2 is the non-CIM path: e-commerce order transactions, index="business_data" sourcetype="business:orders:transactions", a custom pseudo data model over a raw JSON payload. The AI dictionary advisor drafts the dictionary, a per-field AI advisor tightens currency, and the FQM Advisor reviews the failing field. The result is 13 entities, with 28% of orders carrying a bad currency. Both run in one Virtual Tenant created for the occasion.

Creating the Field Quality tenant

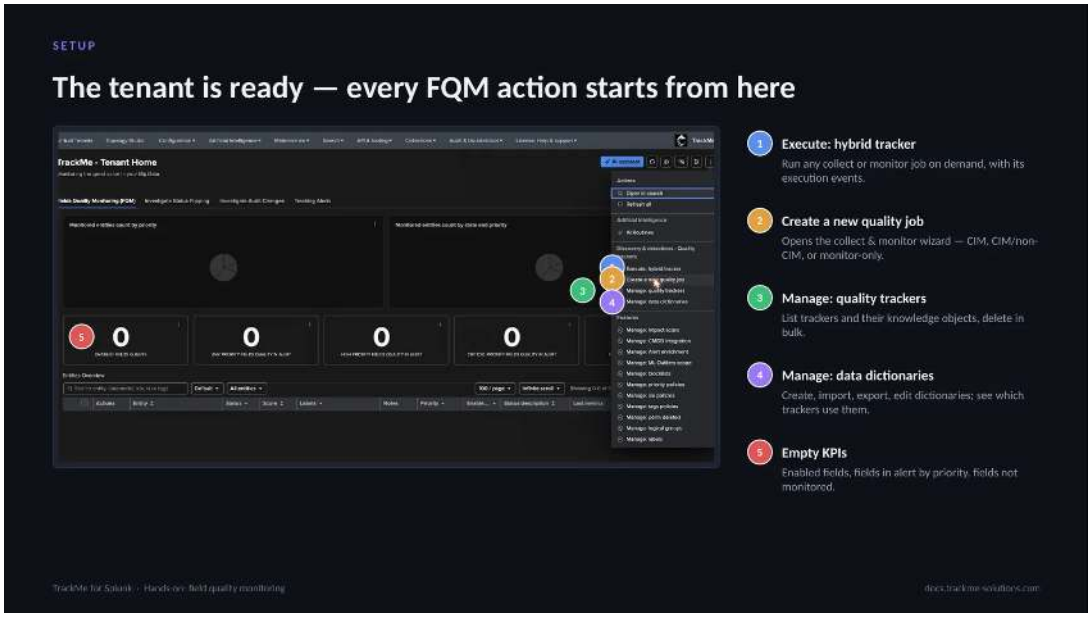


Virtual Tenants → Actions → Create a tracking tenant opens the component chooser. Splunk Fields Quality Monitoring (callout 1) is its own card: sampling jobs for CIM and non-CIM data, quality dictionaries and alerts on quality

issues. The tenant id (callout 2), field-quality, is an immutable identifier checked for availability as it is typed. Alias and description (callout 3) are what people see on the Virtual Tenants card. Next: indexes & RBAC (callout 4) covers the summary, audit, metric and notable indexes, the owner and the roles; the defaults are fine.

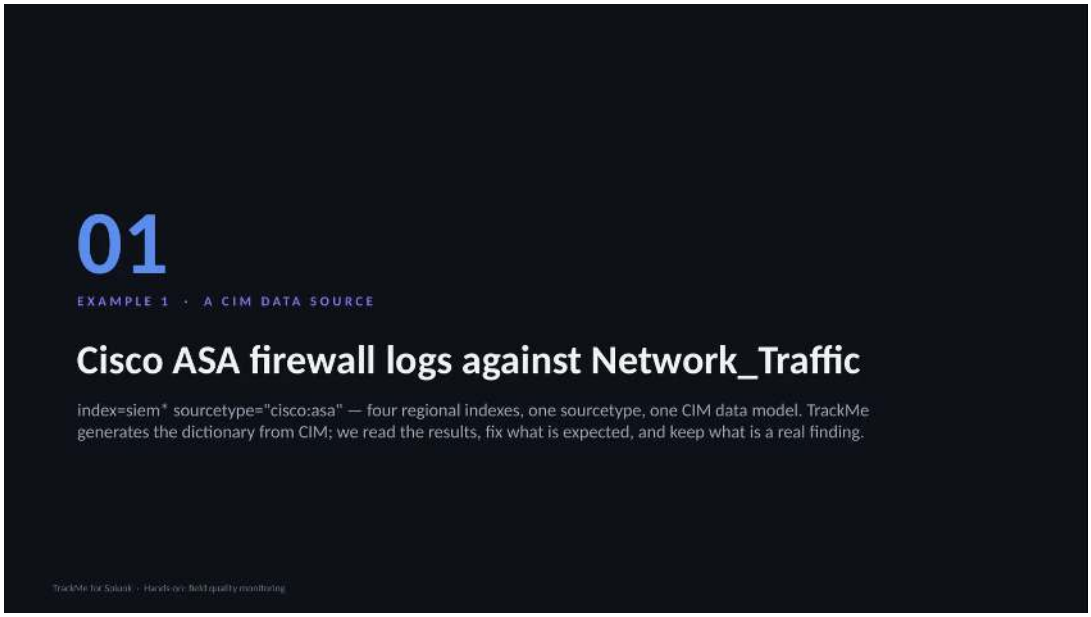
The wizard asks three things, basics, indexes and RBAC, and review; TrackMe then creates the KV collections, lookups and health tracker for the tenant in seconds.

FQM actions in Tenant Home



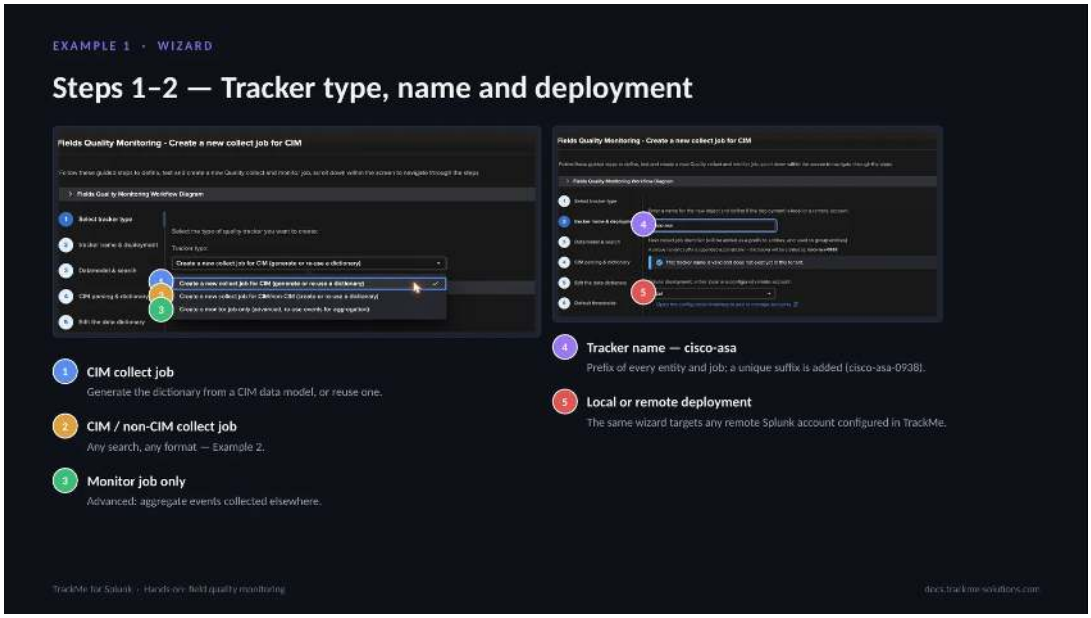
Tenant Home is where FQM lives, and the kebab menu at the top right carries every FQM action. Execute: hybrid tracker (callout 1) runs any collect or monitor job on demand and shows its execution events. Create a new quality job (callout 2) opens the collect & monitor wizard for CIM, CIM/non-CIM or monitor-only jobs. Manage: quality trackers (callout 3) lists trackers and their knowledge objects and deletes them in bulk. Manage: data dictionaries (callout 4) creates, imports, exports and edits dictionaries and shows which trackers use them. The KPIs (callout 5), enabled fields, fields in alert by priority and fields not monitored, are still empty on a new tenant. The first step is Create a new quality job.

Example 1 — A CIM data source



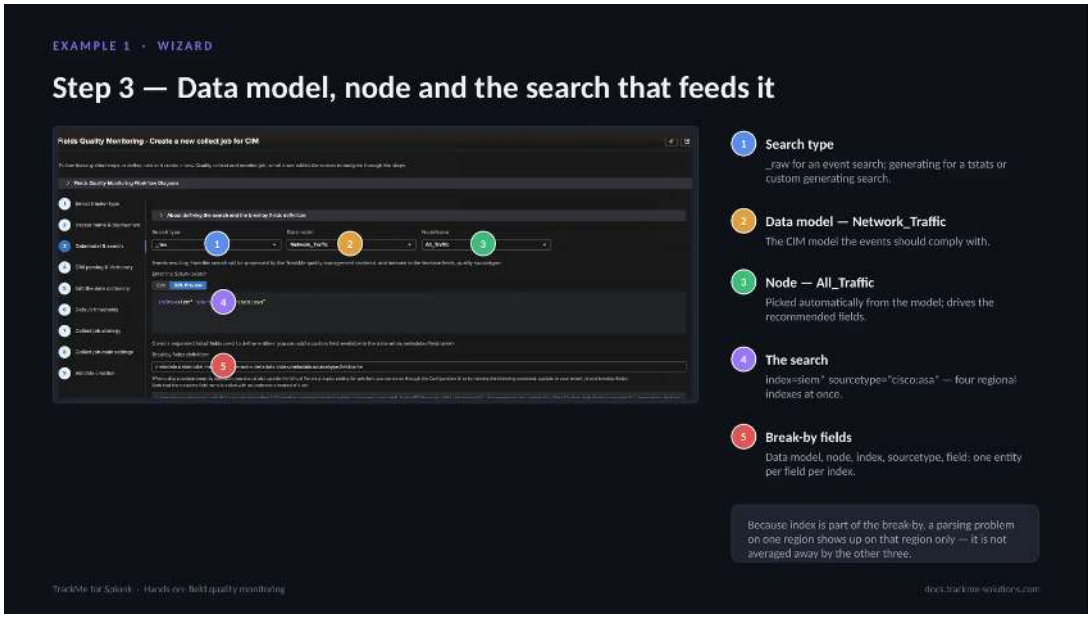
The first example is a CIM source: index=siem* sourcetype="cisco:asa", four regional indexes, one sourcetype, one CIM data model. TrackMe generates the dictionary from CIM, so the interesting part is interpreting the first results: which failures are real CIM compliance problems, and which are simply fields that only exist on some message types. Expected gaps are fixed in the dictionary; real findings are kept.

Choosing the tracker type



Steps 1 and 2 of the wizard offer three tracker types. CIM collect job (callout 1) generates the dictionary from a CIM data model or reuses one. CIM / non-CIM collect job (callout 2) takes any search and any format, used in Example 2. Monitor job only (callout 3) is advanced: it aggregates events collected elsewhere. Tracker name (callout 4), cisco-asa, is the prefix of every entity and job; TrackMe appends a short unique suffix (cisco-asa-0938). Local or remote deployment (callout 5): the same wizard targets the local deployment or any remote Splunk account configured in TrackMe.

Break-by fields and the source search

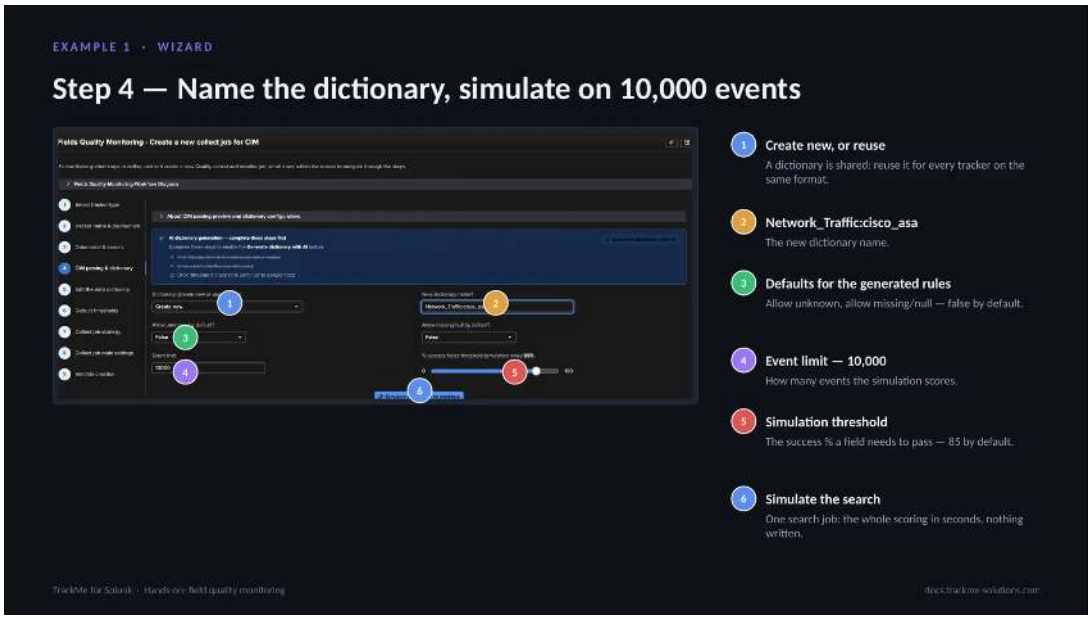


Search type (callout 1) is _raw for an event search or generating for a tstats or custom generating search. Data model (callout 2) is Network_Traffic, the CIM model the events should comply with. Node (callout 3), All_Traffic, is picked automatically from the model and drives the recommended fields. The search (callout 4), index=siem*

sourcetype="cisco:asa", covers four regional indexes at once. Break-by fields (callout 5) are data model, node, index, sourcetype and field: one entity per field per index.

The break-by is the identity of the entities. Because index is part of it, a parsing problem on one region shows up on that region only and is not averaged away by the other three.

Simulation settings



Step 4 names the dictionary and runs a simulation. Create new, or reuse (callout 1): a dictionary is shared and can be reused for every tracker on the same format. The new dictionary is named Network_Traffic:cisco_asa (callout 2). Defaults for the generated rules (callout 3), allow unknown and allow missing/null, are false by default. Event limit (callout 4) sets how many events the simulation scores, 10,000 by default; 100,000 also works well. Simulation threshold (callout 5) is the success % a field needs to pass, 85 by default. Simulate the search (callout 6) runs the real collect logic, scoring in seconds and writing nothing.

The simulation is the loop to iterate on.

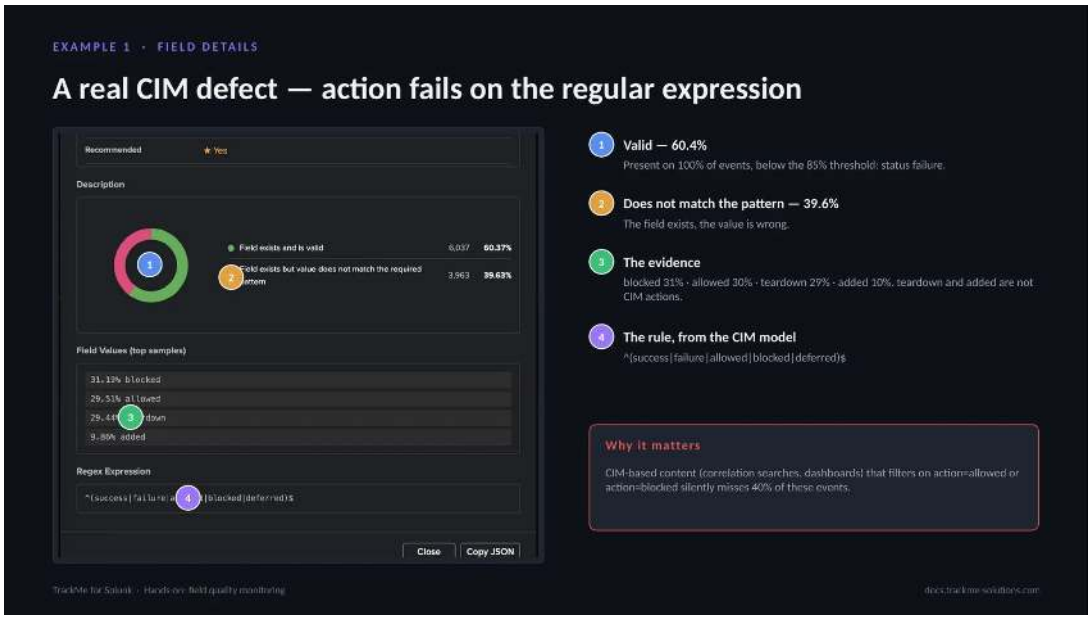
Reading the first simulation



All fields (callout 1): 53 checked, 5 passed, 9.43%, across every field of the Network_Traffic model, most of which are irrelevant for a firewall. Recommended fields (callout 2): 3 of 12, the fields CIM recommends for All_Traffic, and the number that matters, 25%. Recommended only (callout 3) filters the table to those 12 fields, marked in the star column. The table (callout 4) shows, per field and per index, coverage %, success %, totals, failures and distinct values. Inspect (callout 5) opens Field Details for one field.

A low first score is normal on CIM: the model lists every possible field, and a Cisco ASA feed mixes connection builds, teardowns and translations that carry different fields.

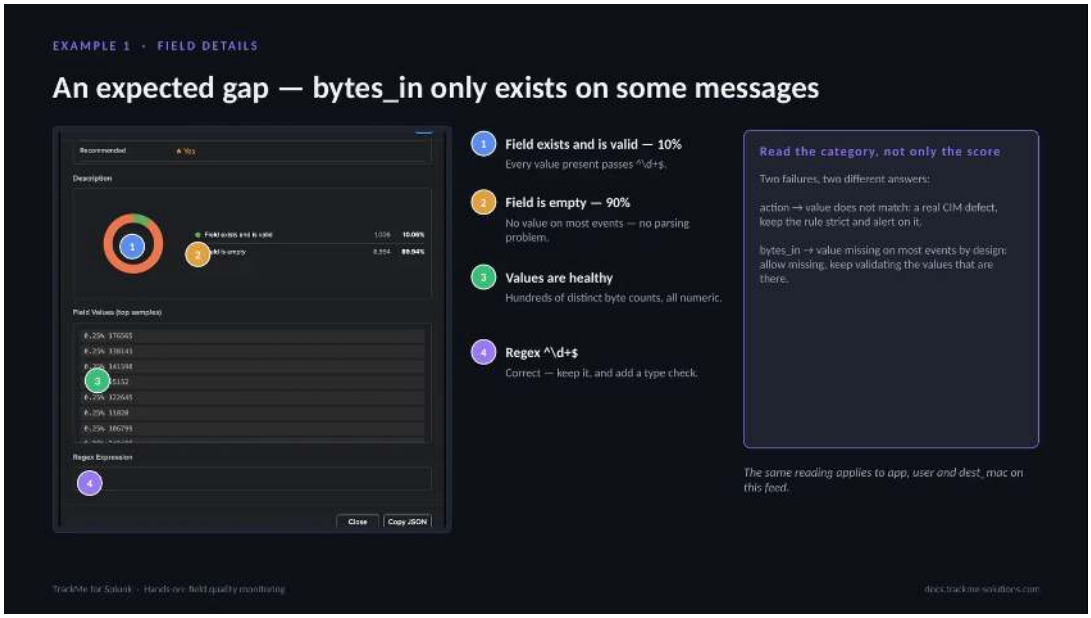
A genuine compliance defect: action



Field Details for action, opened from the simulation with Inspect. Valid (callout 1) is 60.4%: the field is present on 100% of events but below the 85% threshold, so its status is failure. Does not match the pattern (callout 2) is 39.6%: the field exists, the value is wrong. The evidence (callout 3) is the top values: blocked 31%, allowed 30%, teardown 29%, added 10%; teardown and added are not CIM actions. The rule (callout 4), from the CIM model, is ^(success|failure|allowed|blocked|deferred)$.

This is a genuine CIM compliance issue in the field extraction. CIM-based content that filters on action=allowed or action=blocked silently misses 40% of these events, so the rule is kept strict.

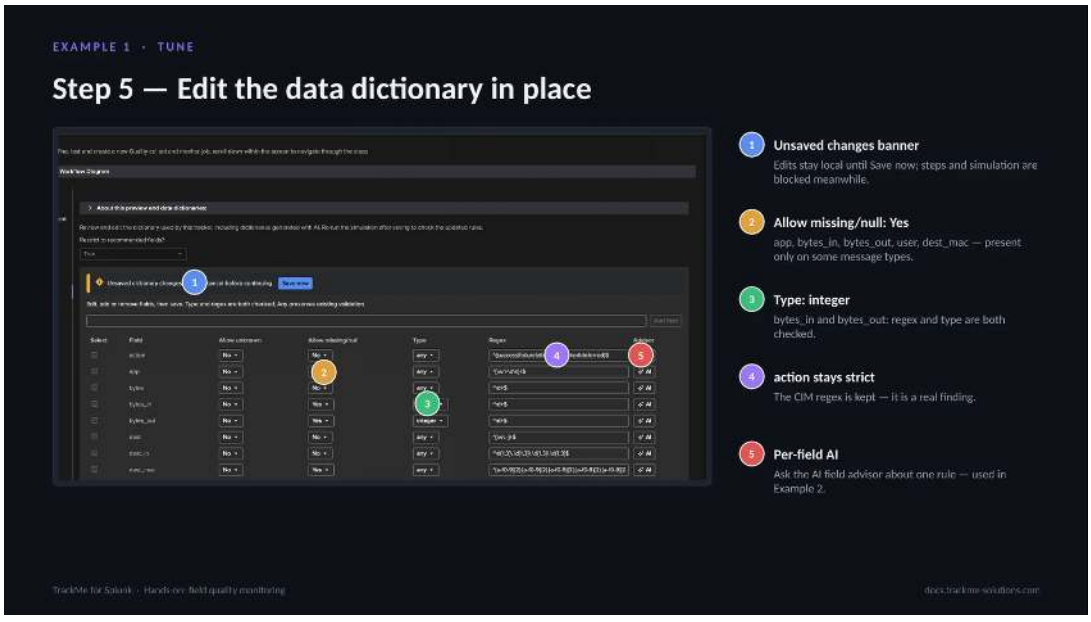
An expected gap: bytes_in



bytes_in reads differently. Field exists and is valid (callout 1) is 10%, and every value present passes ^d+$. Field is empty (callout 2) is 90%: no value on most events, because only connection teardown messages carry byte counts; this is not a parsing problem. Values are healthy (callout 3): hundreds of distinct byte counts, all numeric. The regex ^d+$ (callout 4) is correct; it is kept, and a type check is added.

Two failures call for two answers. action fails because the value does not match: a real CIM defect, so the rule stays strict and alerts. bytes_in fails because the value is missing on most events by design: allow missing, and keep validating the values that are there.

Editing rules in the dictionary editor

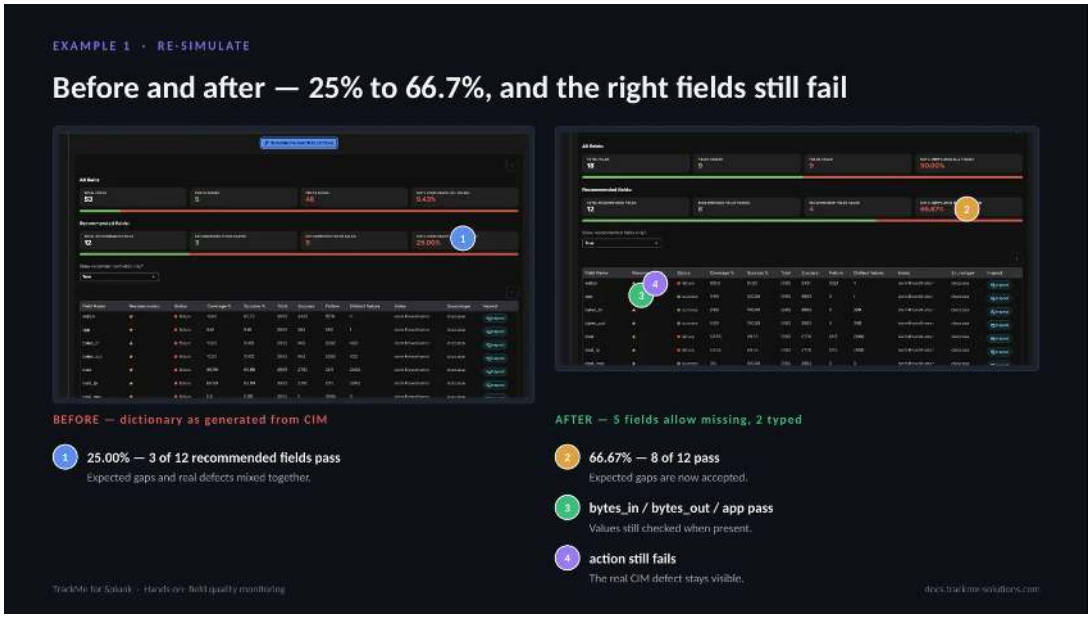


Step 5 edits the dictionary in place. The unsaved changes banner (callout 1) shows that edits stay local until Save now; steps and simulation are blocked meanwhile. Allow missing/null: Yes (callout 2) is set on app, bytes_in, bytes_out, user and dest_mac. Type: integer (callout 3) is set on bytes_in and bytes_out, checking regex and type. action stays strict (callout 4): the CIM regex is kept, a real finding. Per-field AI (callout 5) asks the AI field advisor about one rule.

Every rule is editable, with allow unknown, allow missing or empty, an optional type (integer, numerical, alpha,

alphanumeric, boolean) and the regex. Save writes the whole dictionary at once and checks nobody changed it in between.

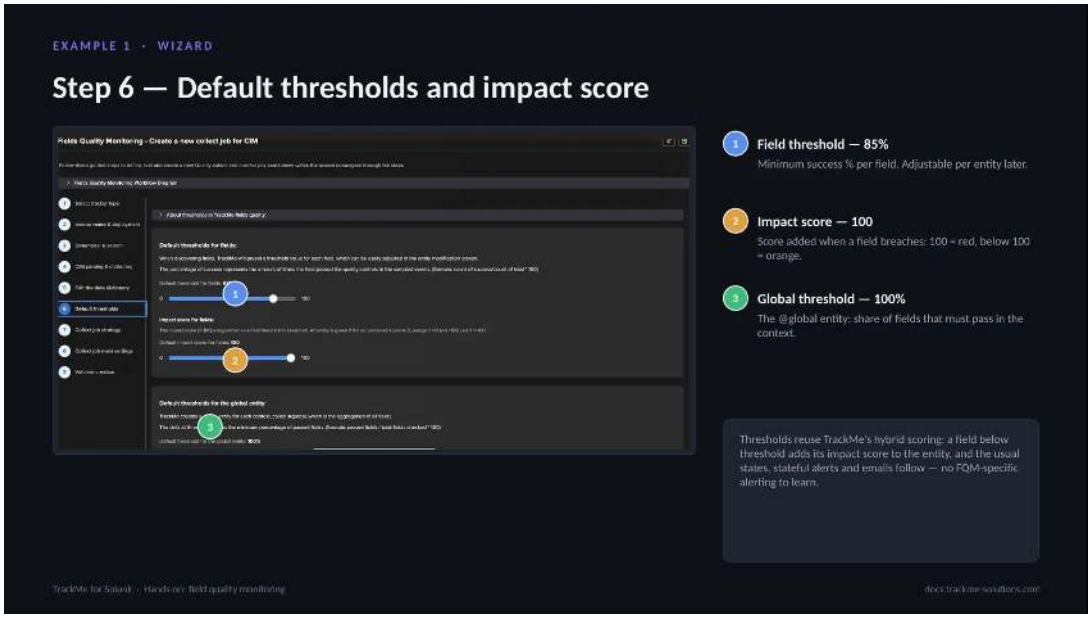
Compliance after tuning



The same data, re-simulated with the new dictionary. Before (callout 1), with the dictionary as generated from CIM, 25.00% of the recommended fields passed, 3 of 12, with expected gaps and real defects mixed together. After (callout 2), with 5 fields allowing missing and 2 typed, 66.67% pass, 8 of 12: the expected gaps are now accepted. bytes_in, bytes_out and app pass (callout 3), their values still checked when present. action still fails (callout 4): the real CIM defect stays visible.

What remains red is meaningful: action with its non-CIM values, and dest, dest_ip and protocol, missing on about 30% of events, worth raising with the TA owner.

Thresholds and impact score

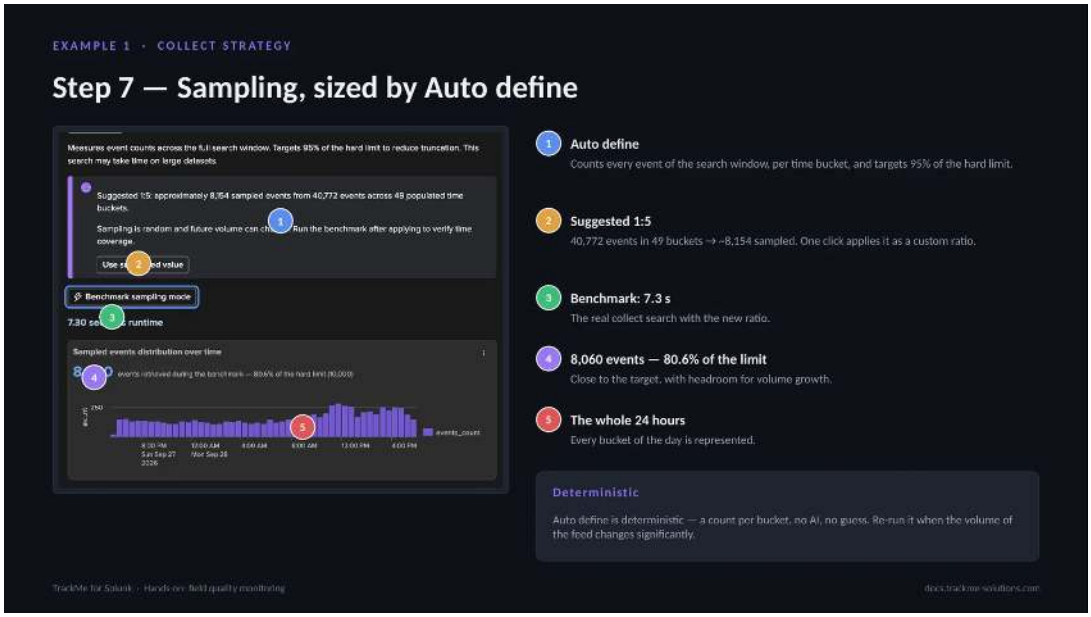


Step 6 sets the default thresholds. Field threshold (callout 1), 85%, is the minimum success % per field, adjustable per entity later. Impact score (callout 2), 100, is the score added when a field breaches: 100 makes the entity red,

a lower value makes it orange. Global threshold (callout 3), 100%, applies to the @global entity, the share of fields that must pass in the context; at 100% any failing field turns it red.

Thresholds reuse TrackMe's hybrid scoring: a field below threshold adds its impact score to the entity, and the usual states, stateful alerts and emails follow, with no FQM-specific alerting to learn. Lowering the impact score makes quality issues show orange rather than red.

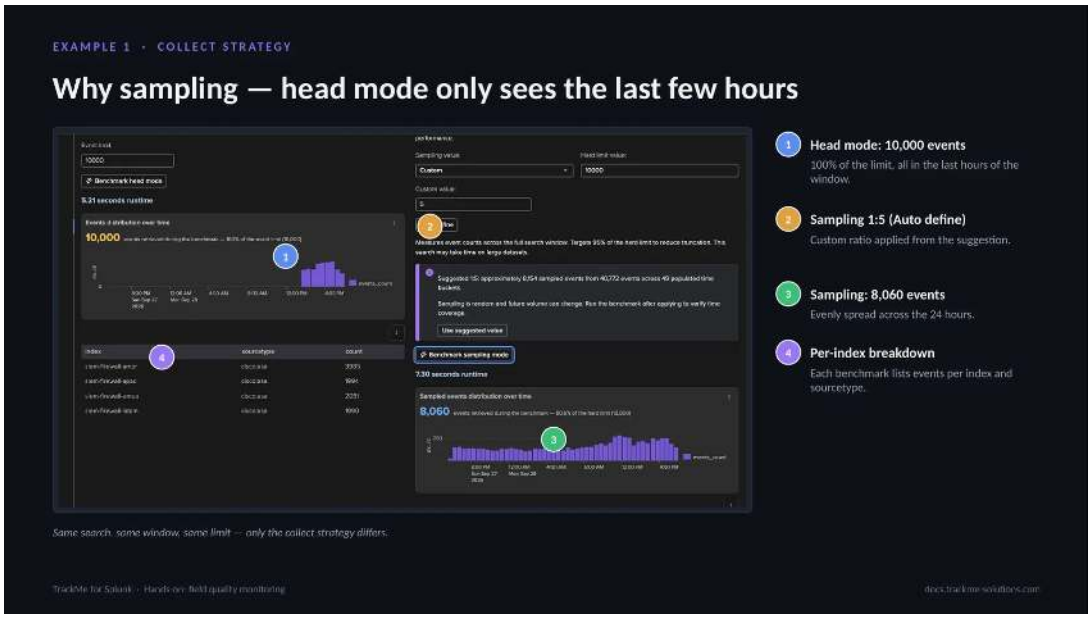
Sizing the sample with Auto define



Step 7 sizes the sample. Auto define (callout 1) counts every event of the search window per time bucket and targets 95% of the hard limit. The suggestion (callout 2) is 1:5: 40,772 events in 49 buckets give about 8,154 sampled events, applied as a custom ratio in one click. Benchmark (callout 3) runs the real collect search with the new ratio in 7.3 s. It retrieves 8,060 events (callout 4), 80.6% of the limit, with headroom for volume growth, spread over the whole 24 hours (callout 5) so every bucket of the day is represented.

Auto define is deterministic, a count per bucket, no guess; it should be re-run when the volume of the feed changes significantly.

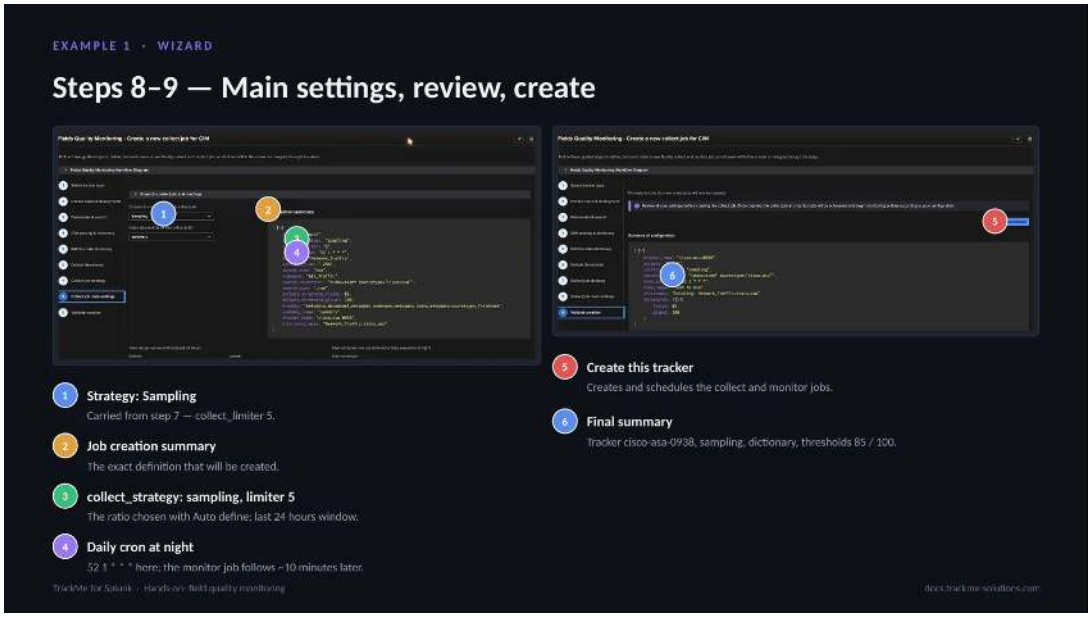
Head mode versus sampling



Same search, same window, same limit; only the collect strategy differs. Head mode (callout 1) returns the first 10,000 events the search yields, which on Splunk means the most recent: 100% of the limit, all inside the last few hours of the window. Sampling 1:5 (callout 2) is the custom ratio applied from the Auto define suggestion. It retrieves 8,060 events (callout 3), evenly spread across the 24 hours. The per-index breakdown (callout 4) lists events per index and sourcetype for each benchmark.

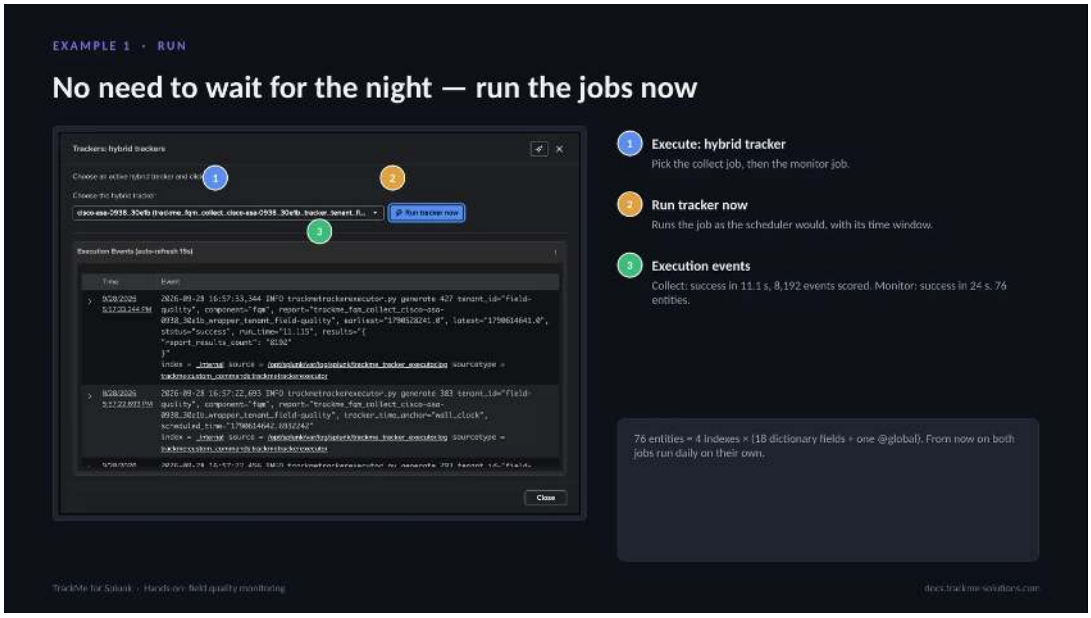
For quality measurement, representative beats recent: a parsing problem at 3 AM is invisible to head mode.

Review and job creation



Step 8 confirms the settings. Strategy: Sampling (callout 1) is carried from step 7 with collect_limiter 5. The job creation summary (callout 2) describes tracker cisco-asa-0938 with sampling, the dictionary and thresholds 85 / 100; it is the exact payload sent to TrackMe. collect_strategy: sampling, limiter 5 (callout 3) is the ratio chosen with Auto define over a last-24-hours window. The daily cron (callout 4) runs at night as recommended, 52 1 * * * here, and the monitor job follows about 10 minutes later. Create this tracker (callout 5) creates and schedules both the collect and monitor jobs, and the final summary (callout 6) shows the configuration that will be created.

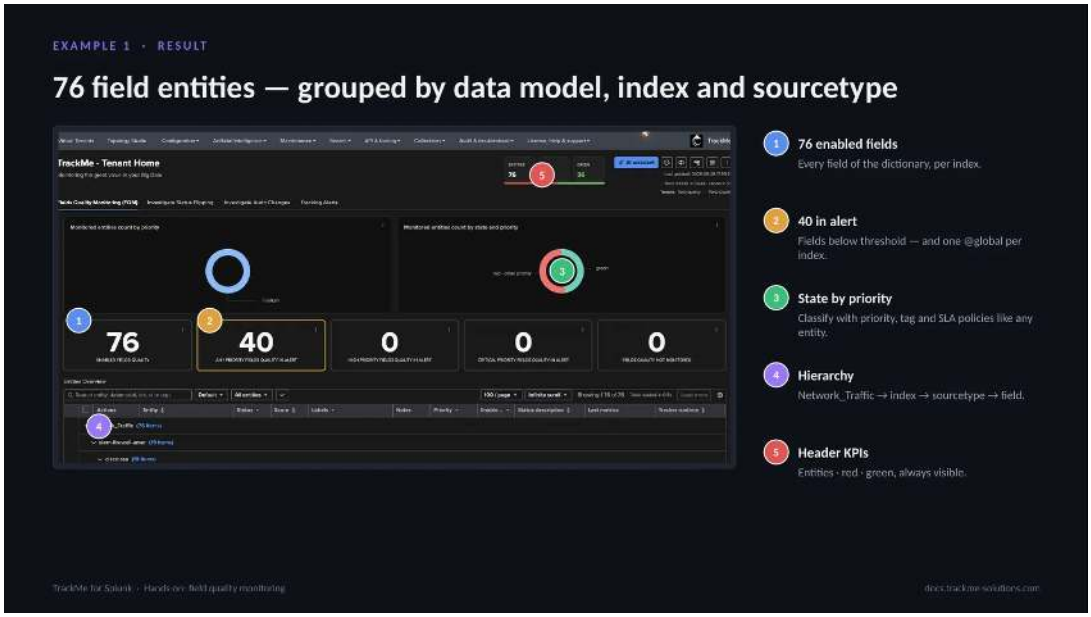
Running the jobs on demand



In production the jobs run on their cron; for a first validation they can be run on demand from the Tenant Home menu. Execute: hybrid tracker (callout 1) picks the collect job, then the monitor job. Run tracker now (callout 2) runs the job as the scheduler would, with its time window. Execution events (callout 3) stream from the executor: the collect job succeeds in 11.1 s with 8,192 events scored, and the monitor job succeeds in 24 s with 76 entities.

76 entities is 4 indexes × (18 dictionary fields + one @global). From now on both jobs run daily on their own.

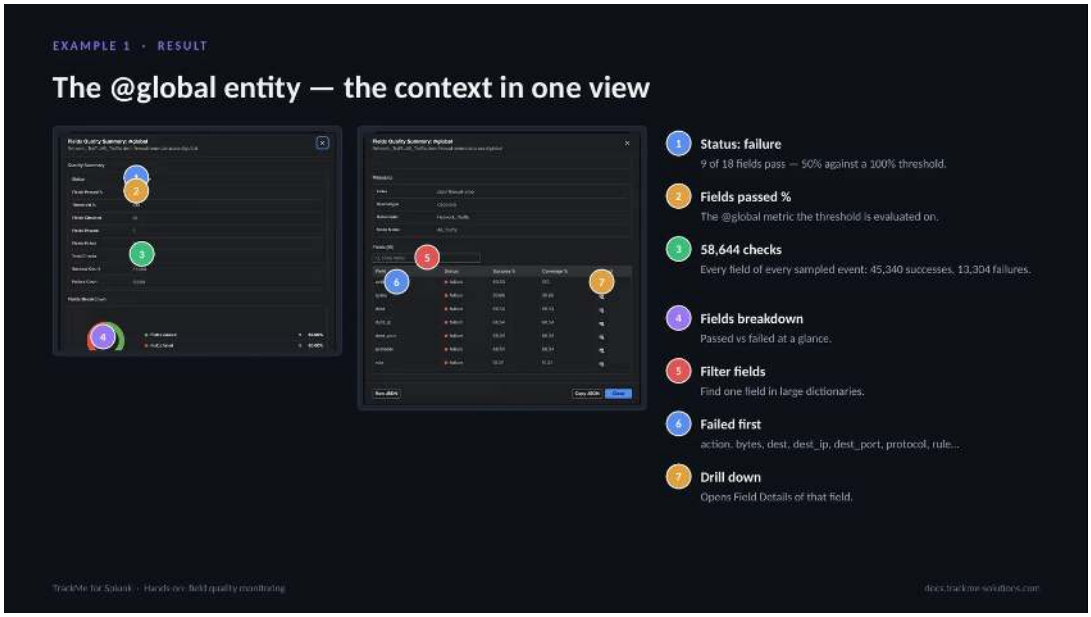
The field entities in Tenant Home



The tenant now holds 76 enabled fields (callout 1), every field of the dictionary, per index. 40 are in alert (callout 2): fields below threshold, plus one @global per index. State by priority (callout 3) classifies them with priority, tag and SLA policies like any entity. The hierarchy (callout 4) follows the break-by: Network_Traffic → index → sourcetype → field. Header KPIs (callout 5) show entities, red and green, always visible.

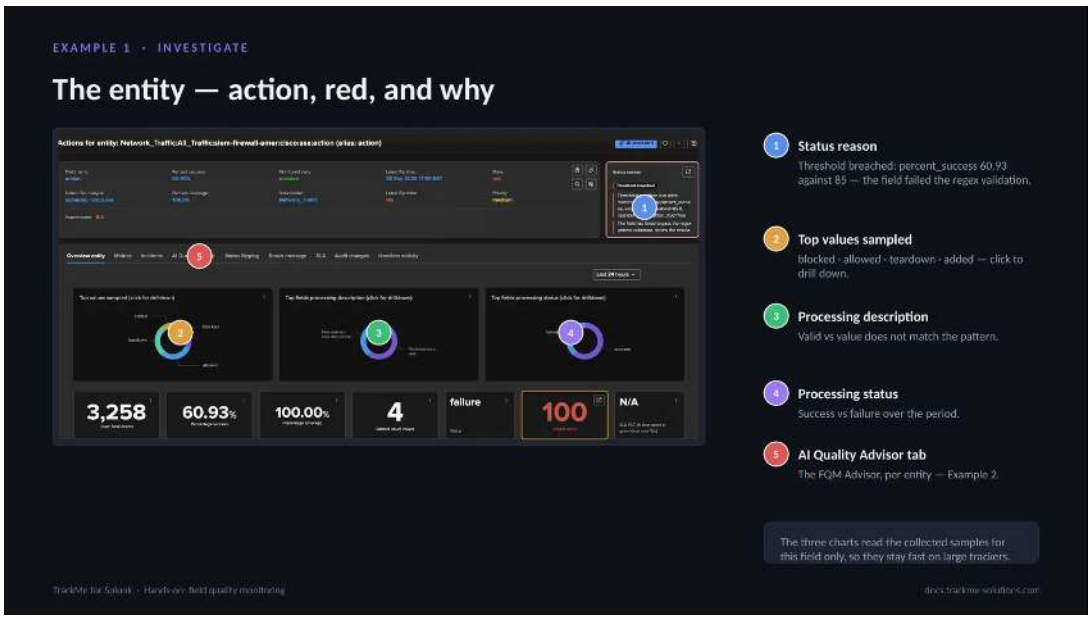
FQM entities are regular TrackMe entities: priority, tags, SLA, labels, logical groups, acknowledgments, maintenance and stateful alerting all apply.

Inspecting the @global entity



@global aggregates every field of one context, here Network_Traffic / All_Traffic / siem-firewall-amer / cisco:asa. Status: failure (callout 1): 9 of 18 fields pass, 50% against a 100% threshold. Fields passed % (callout 2) is the @global metric the threshold is evaluated on. 58,644 checks (callout 3) cover every field of every sampled event: 45,340 successes and 13,304 failures. The fields breakdown (callout 4) shows passed against failed at a glance. Filter fields (callout 5) finds one field in large dictionaries. Failed first (callout 6) lists action, bytes, dest, dest_ip, dest_port, protocol, rule and so on. Drill down (callout 7) opens the Field Details of that field, with a Back to @global link.

The action entity over time



Opening the action entity gives the same evidence as the wizard, now over time. The status reason (callout 1) is written in plain language: threshold breached, percent_success 60.93 against 85, the field failed the regex validation. Top values sampled (callout 2), blocked, allowed, teardown and added, each drill down on click. Processing description (callout 3) splits valid values from values that do not match the pattern. Processing status (callout 4) shows success against failure over the period. The AI Quality Advisor tab (callout 5) hosts the FQM Advisor for this entity, used in Example 2.

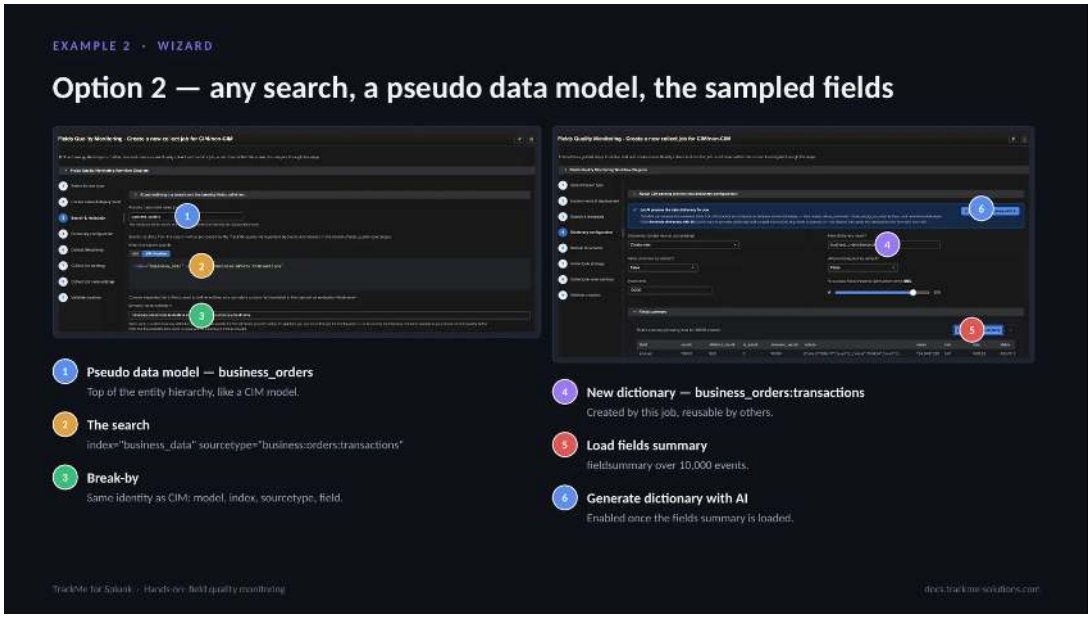
The three charts read the collected samples for this field only, so they stay fast on large trackers.

Example 2 — A non-CIM data source



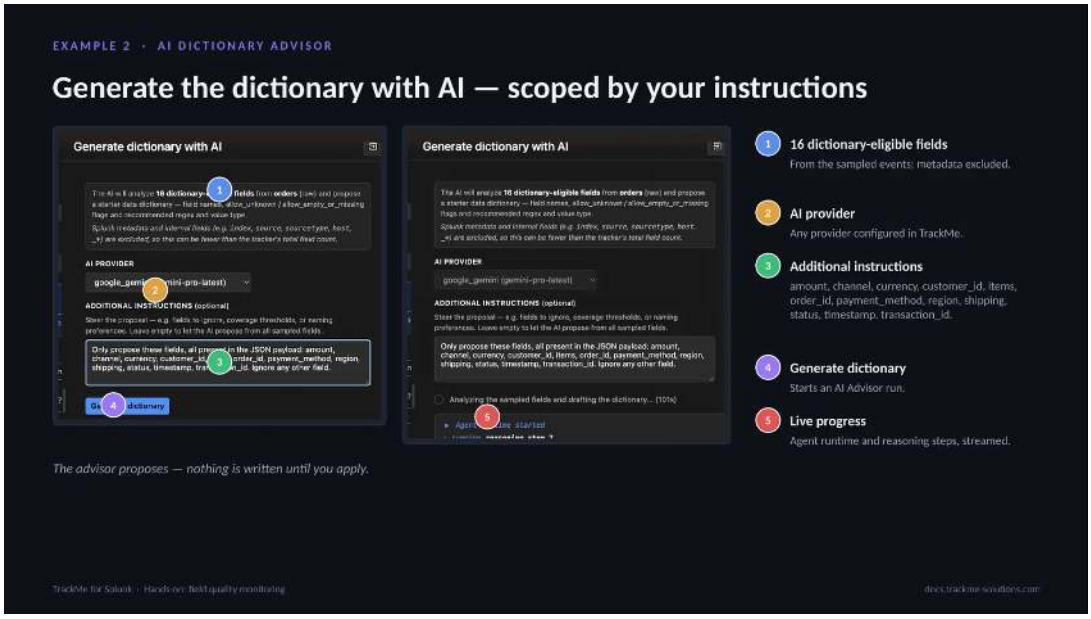
The second example is a business feed with no CIM model behind it: index="business_data" sourcetype="business:orders:transactions", JSON order events. This is where the AI Advisors do the heavy lifting: they turn the sampled fields into a starter dictionary, tighten one rule on request and review the verdict. The analyst stays in control of every change.

Pseudo data model and fields summary



The tracker type is CIM/non-CIM. Pseudo data model (callout 1), business_orders, is the top of the entity hierarchy, like a CIM model, and groups the entities. The search (callout 2) is index="business_data" sourcetype="business:orders:transactions". Break-by (callout 3) keeps the same identity as CIM: model, index, sourcetype and field. New dictionary (callout 4), business_orders:transactions, is created by this job and reusable by others. Load fields summary (callout 5) runs fieldsummary over 10,000 events: the fields found, with counts, distinct values and statistics. Generate dictionary with AI (callout 6) is enabled once the fields summary is loaded; that summary is what the AI dictionary advisor analyses.

Starting the AI dictionary advisor

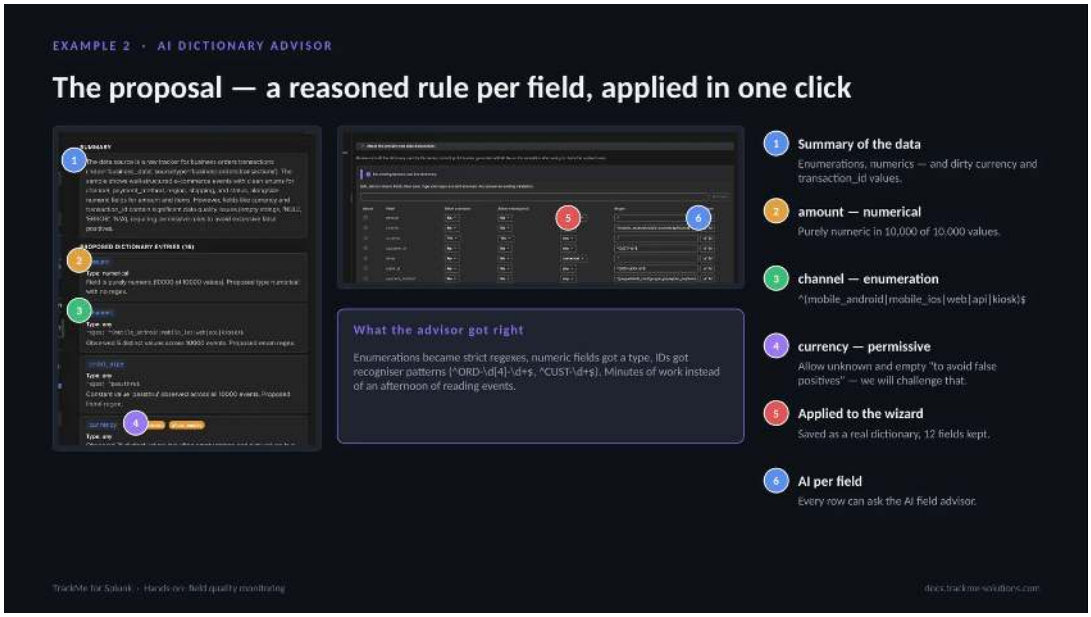


The AI dictionary advisor analyses the 16 dictionary-eligible fields (callout 1) of the sampled events; Splunk metadata and internal fields are excluded. AI provider (callout 2) is any provider configured in TrackMe. Additional

instructions (callout 3) steer the advisor; here they list the business fields of interest: amount, channel, currency, customer_id, items, order_id, payment_method, region, shipping, status, timestamp and transaction_id. Generate dictionary (callout 4) starts an AI Advisor run, whose live progress (callout 5), agent runtime and reasoning steps, is streamed in the panel.

The advisor proposes a complete starter dictionary: field names, allow flags, a recommended regex and a value type. Nothing is written until it is applied.

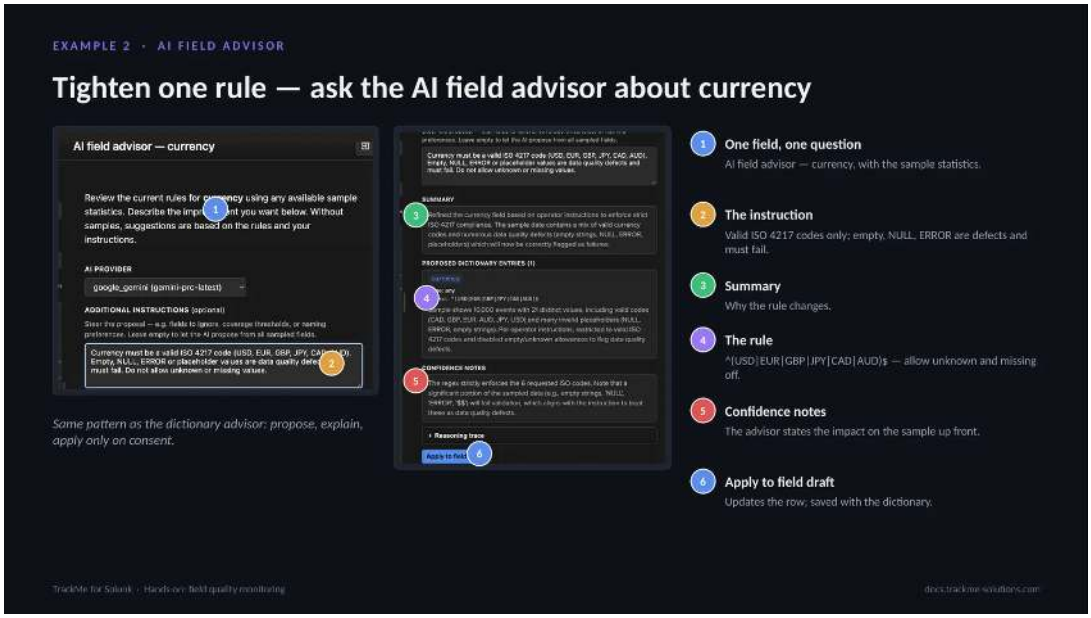
Applying the proposed dictionary



The summary of the data (callout 1) reports clean enumerations for channel, payment_method, region, shipping and status, numeric amount and items, and dirty currency and transaction_id values. amount (callout 2) is typed numerical, numeric in every sampled value. channel (callout 3) becomes the enumeration ^(mobile_android|mobile_ios|web|api|kiosk)$. currency (callout 4) is left permissive, allowing unknown and empty "to avoid false positives". Applied to the wizard (callout 5) saves it as a real dictionary with 12 fields kept; four ingestion pipeline fields are removed in the editor. AI per field (callout 6) lets every row ask the AI field advisor.

Enumerations became strict regexes, numeric fields got a type and IDs got recogniser patterns; each entry explains its rule.

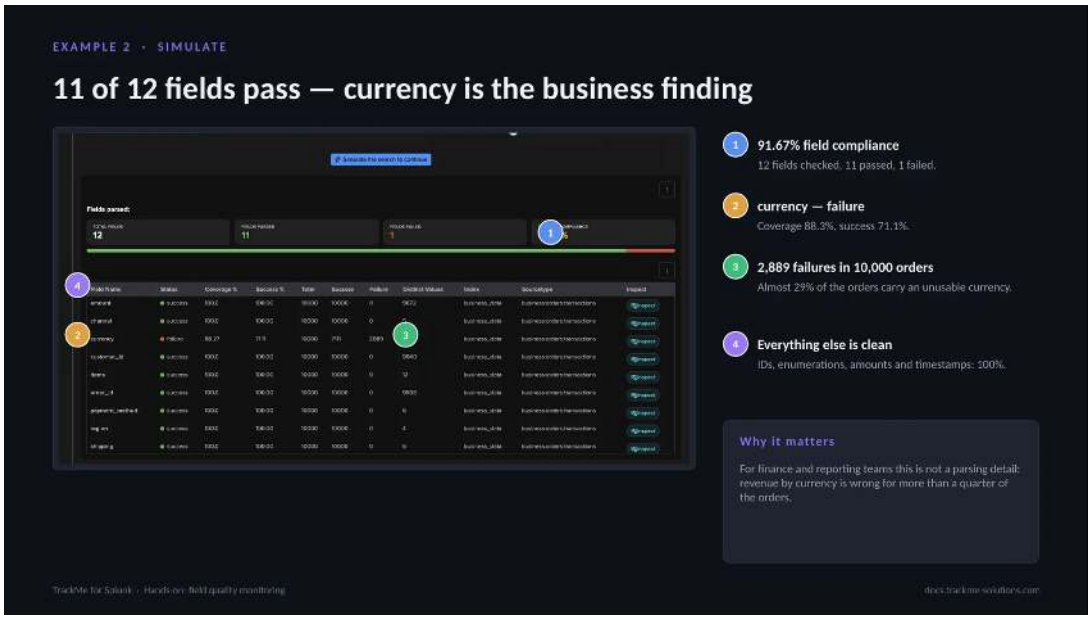
Tightening the currency rule



The permissive currency rule is the wrong call for a quality monitor: the dirty values are precisely what should be caught. The AI field advisor (callout 1) reviews one field, currency, with the sample statistics. The instruction (callout 2): valid ISO 4217 codes only; empty, NULL and ERROR are defects and must fail. The summary (callout 3) explains why the rule changes. The rule (callout 4) becomes ^(USD|EUR|GBP|JPY|CAD|AUD)$ with allow unknown and missing off. Confidence notes (callout 5) state the impact on the sample up front: a significant share will fail. Apply to field draft (callout 6) updates the row, saved with the dictionary.

Same pattern as the dictionary advisor: propose, explain, apply only on consent.

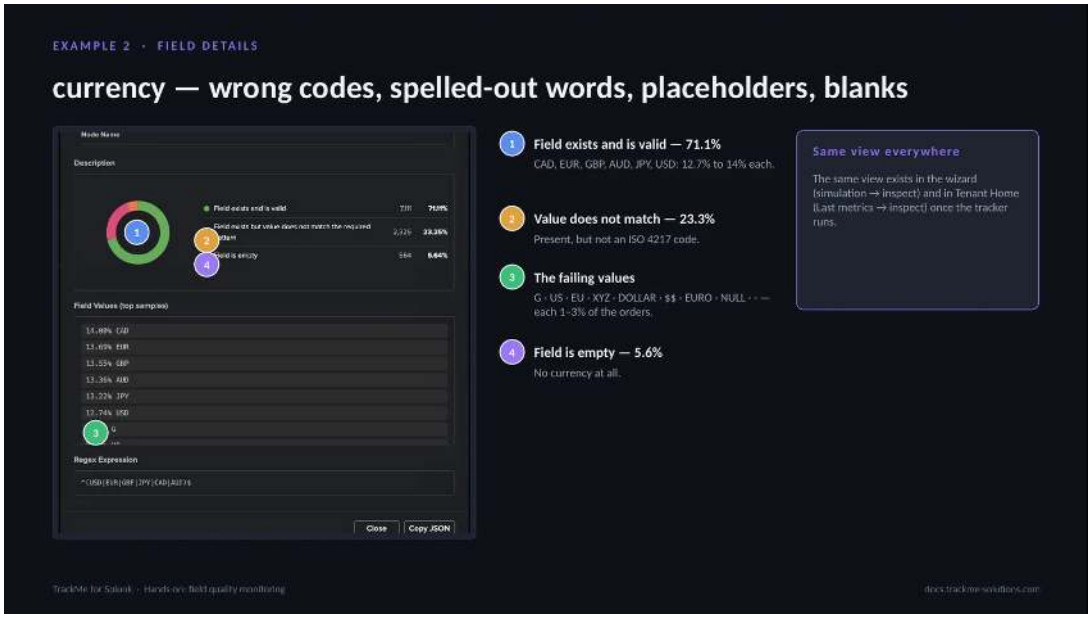
Simulation with the tightened rule



With the tightened rule, the simulation isolates exactly one problem. Field compliance (callout 1) is 91.67%: 12 fields checked, 11 passed, 1 failed. currency (callout 2) fails with coverage 88.3% and success 71.1%. 2,889 failures in 10,000 orders (callout 3): almost 29% of the orders carry an unusable currency. Everything else (callout 4) is clean: IDs, enumerations, amounts and timestamps at 100%.

For finance and reporting teams this is not a parsing detail: revenue by currency is wrong for more than a quarter of the orders. The failure is a genuine data quality issue in the source application.

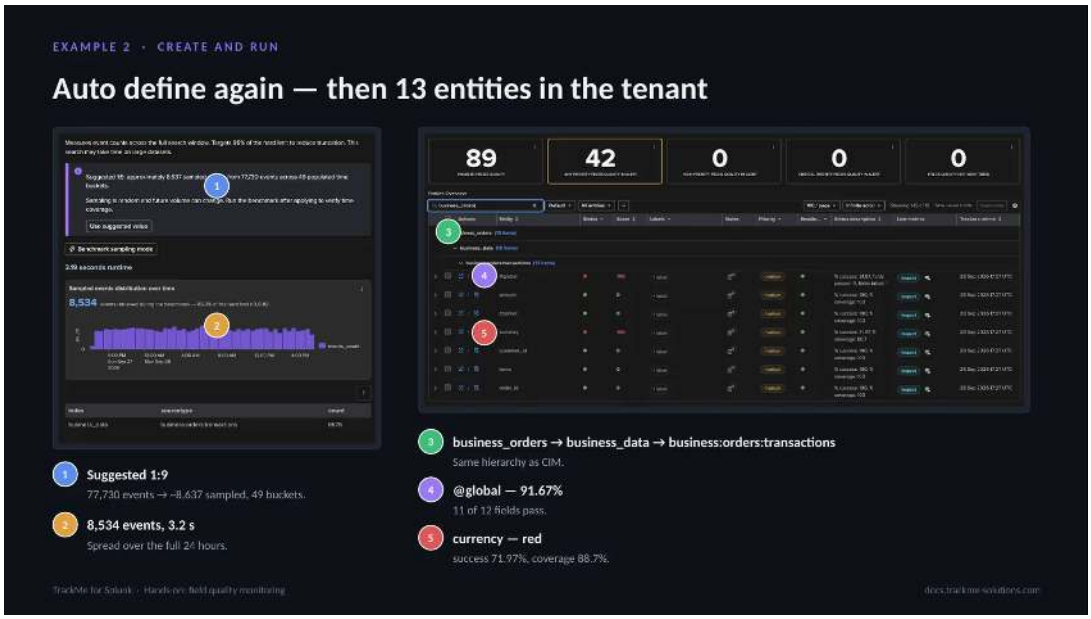
The failing currency values



Field Details for currency. Field exists and is valid (callout 1) is 71.1%: CAD, EUR, GBP, AUD, JPY and USD at 12.7% to 14% each. Value does not match (callout 2) is 23.3%: present, but not an ISO 4217 code. The failing values (callout 3) are G, US, EU, XYZ, DOLLAR, $$, EURO, NULL and -, each 1–3% of the orders: truncated codes, spelled-out words, symbols and placeholders. Field is empty (callout 4) is 5.6%, no currency at all.

The top values are the evidence to forward to the application team. The same view exists in the wizard (simulation → inspect) and in Tenant Home (Last metrics → inspect) once the tracker runs.

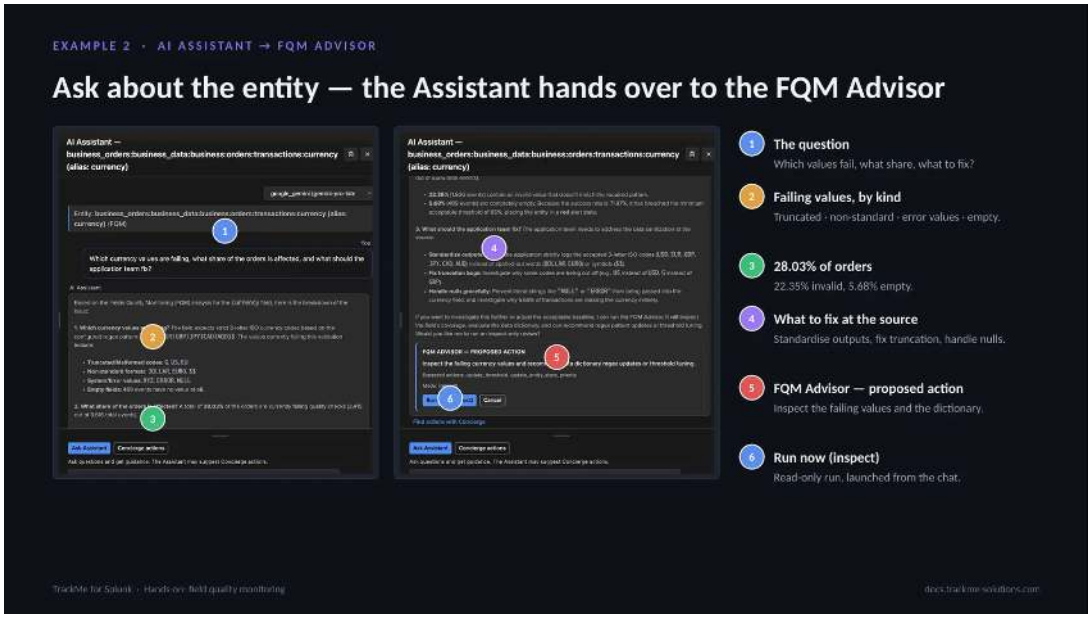
Creating and running the orders tracker



Step 7 again sizes the sample. Auto define suggests 1:9 (callout 1): 77,730 events in 49 buckets give about 8,637 sampled events. The benchmark (callout 2) retrieves 8,534 events in 3.2 s, spread over the full 24 hours. After creation and a first run, the hierarchy (callout 3) reads business_orders → business_data → business:orders:transactions, the same as CIM. @global (callout 4) is at 91.67%, 11 of 12 fields pass. currency (callout 5) is red with success 71.97% and coverage 88.7%.

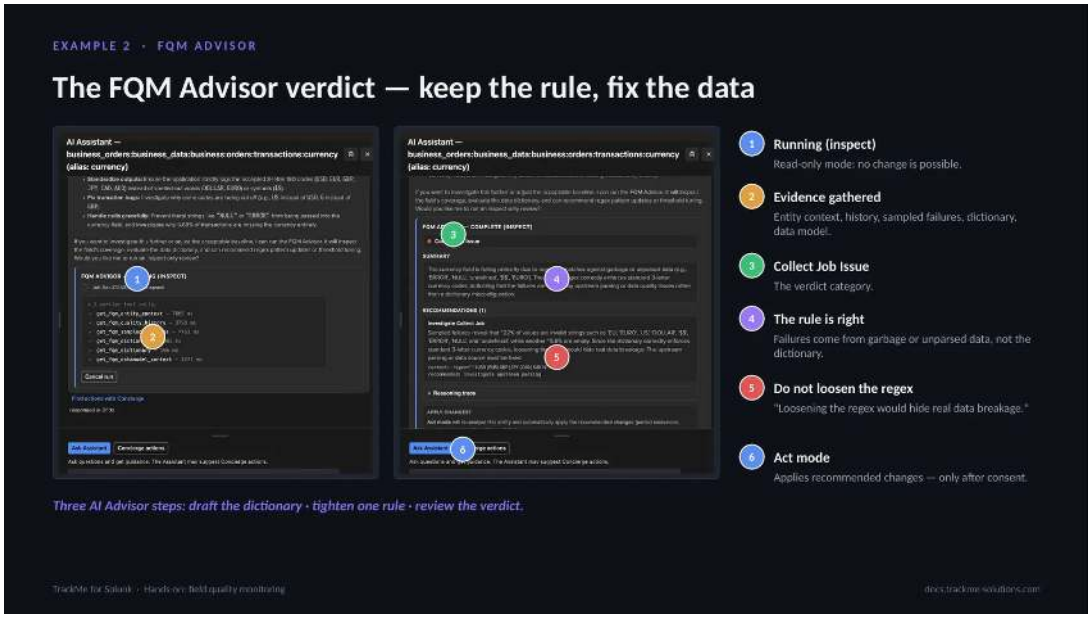
The tenant now has 13 new entities: 12 fields and the @global. currency and @global are red, everything else green.

From the AI Assistant to the FQM Advisor



On the currency entity, the AI Assistant is asked which values fail, what share of orders is affected and what to fix (callout 1). It answers from the entity data. Failing values, by kind (callout 2): truncated, non-standard, error values and empty. 28.03% of orders (callout 3) are affected: 22.35% invalid and 5.68% empty. What to fix at the source (callout 4): standardise outputs, fix truncation, handle nulls. The Assistant then proposes the right specialist, an FQM Advisor proposed action (callout 5) to inspect the failing values and the dictionary, as an action card. Run now (inspect) (callout 6) launches a read-only run from the chat in one click.

The advisor's inspect verdict



The FQM Advisor runs as an agent with read tools. Running (inspect) (callout 1) is read-only mode: no change is possible. Evidence gathered (callout 2) covers the entity context, quality history, sampled failures, the dictionary and the data model; the live feed shows each call. The verdict category (callout 3) is Collect Job Issue. The rule is right (callout 4): the failures come from garbage or unparsed data, not from the dictionary. Do not loosen the regex (callout 5), because "loosening the regex would hide real data breakage." Act mode (callout 6) applies recommended changes, only after consent.

Three AI Advisor steps cover the example: draft the dictionary, tighten one rule, review the verdict.

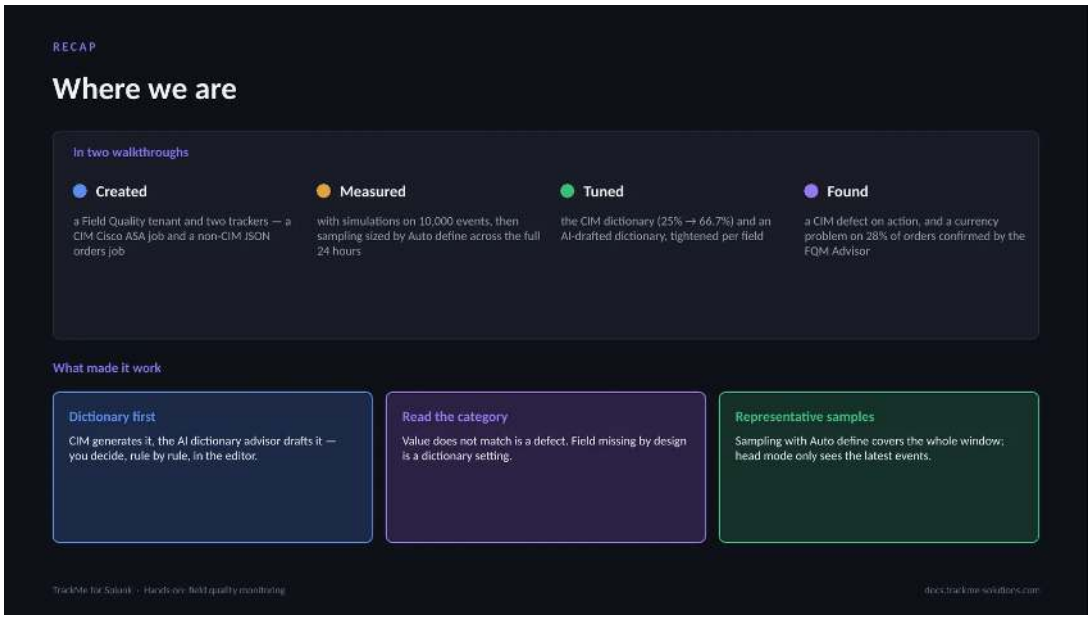
A mature tenant and its topology



The same model on an established tenant with several business and CIM trackers. 214 enabled fields (callout 1) span business and CIM trackers, with months of history. 97 in alert (callout 2) are quality issues to route to owners. Grouped by data model (callout 3): business, Network_Traffic and Web. The Field Quality topology (callout 4) shows tenant → data model with live health. business (callout 5) is 61% healthy, aggregated from every field below it; Web is at 62%. Network_Traffic (callout 6) is at 0%. The topology can be saved as a Topology Studio view.

Every entity keeps its history, flips, incidents and SLA like any TrackMe entity, and stateful alerts notify the owners.

Where we are



Two feeds, two dictionary strategies. Created: a Field Quality tenant and two trackers, CIM Cisco ASA and non-CIM JSON orders. Measured: simulations on 10,000 events, then Auto define sampling across the full 24 hours. Tuned: the CIM dictionary from 25% to 66.7%, and an AI-drafted dictionary tightened per field. Found: a CIM defect on action, and a currency problem on 28% of orders confirmed by the FQM Advisor.

What made it work: dictionary first, CIM generates it, the AI dictionary advisor drafts it, the analyst decides. Read the category: value does not match is a defect, field missing by design is a dictionary setting. Representative samples: Auto define covers the whole window, head mode only the latest events.

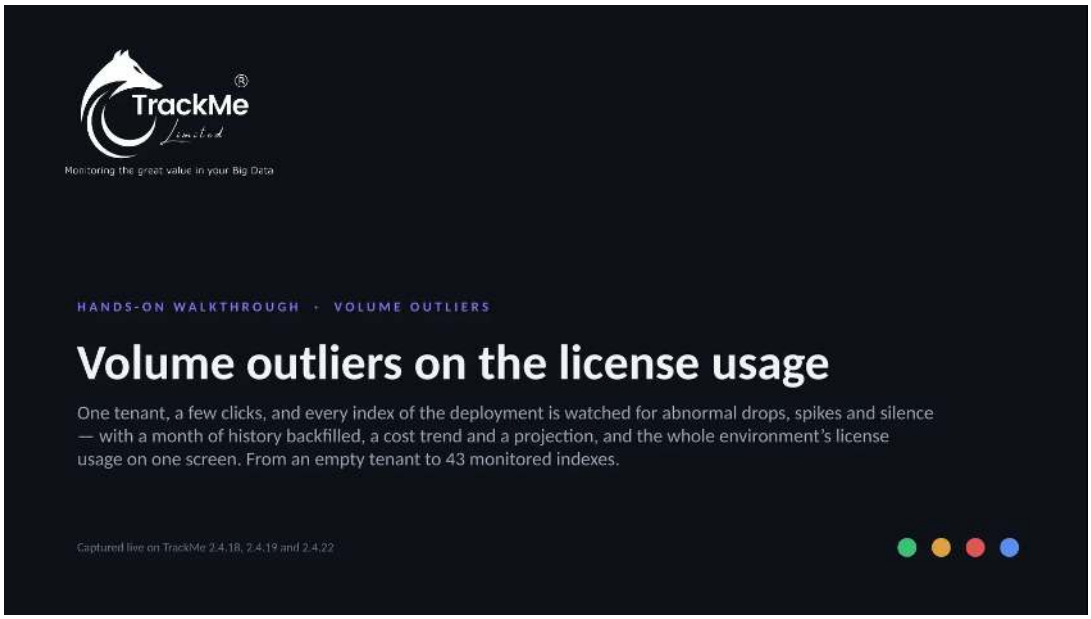
2.1.7 Volume Outliers

Licensed volume per index as entities: rolling volume, daily licensed volume verified against the license usage log, daily changes and the month projection, ML outliers on drops and spikes, and the ML Advisor.

`Download the deck (PDF) · PowerPoint`

about 35 minutes · 43 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview

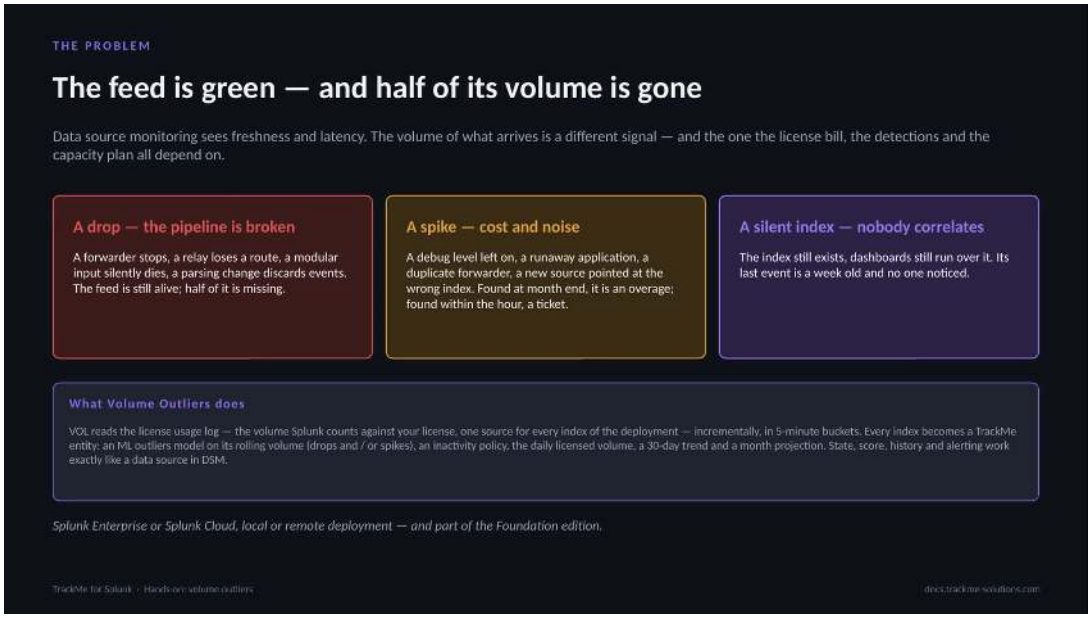


Volume Outliers (VOL) is the seventh TrackMe component, introduced in 2.4.18: the standardised, first-class form of the volume detection TrackMe has always offered. One tenant watches every index of a deployment for abnormal drops, spikes and silence, backfills its history at creation, and adds a cost trend, a month projection and the whole environment's license usage. Parts 1 to 3 were captured on TrackMe 2.4.18, from an empty tenant on a Splunk Enterprise deployment with 43 indexes in one license pool; the license usage view of Part 4 on TrackMe 2.4.19.

The path runs from the tenant creation and the backfill, through reading the results and tuning the detection, to the global license usage view from scopes to chargeback, alerting, the AI Assistant and an AI ML Advisor review of a volume model.

Parts 2 and 4 were refreshed on TrackMe 2.4.22: binary units with IEC labels, a collection that waits for a late source, the closed-day reconciliation against the license usage log, and the Daily changes and Investigate an index views.

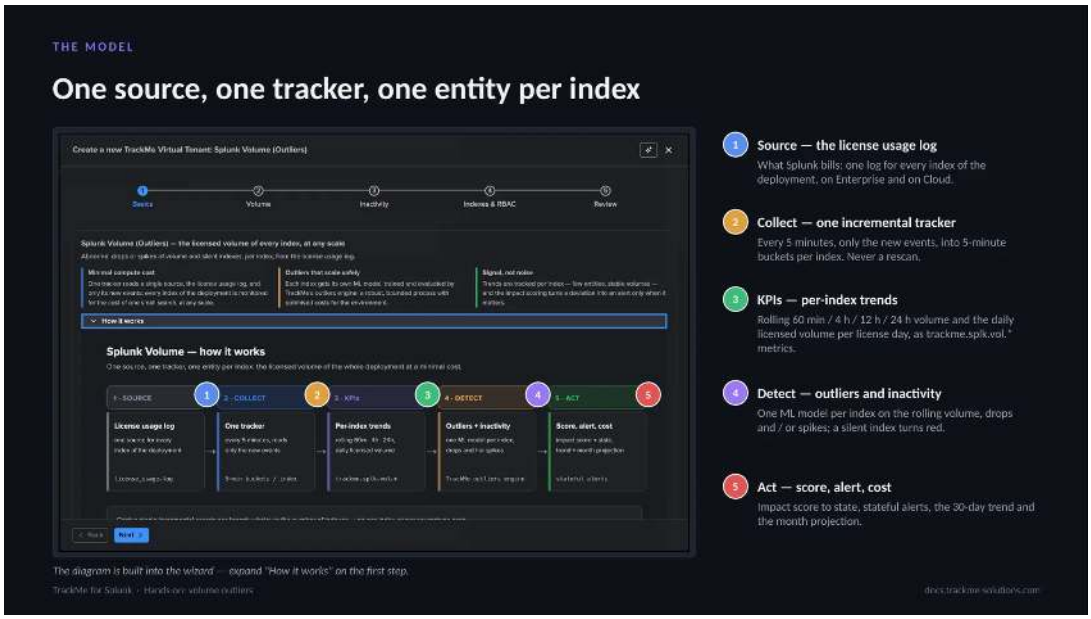
Why volume needs its own signal



Availability monitoring answers whether data arrives on time, not whether the amount is normal. A broken forwarder, a lost route or a parser change leaves the feed green while half of the data is missing; a debug level left on is an overage at month end; a silent index still exists while its last event is a week old.

VOL watches the volume itself, from the license usage log Splunk bills, read incrementally in 5-minute buckets. Every index becomes a TrackMe entity with an ML outliers model on its rolling volume, an inactivity policy, the daily licensed volume, a 30-day trend and a month projection, on Enterprise or Cloud, in the Foundation edition.

Source, tracker, KPIs, detection, action



Callouts 1 to 5 follow the pipeline. The source is the license usage log: what Splunk bills, one log for every index, on Enterprise and on Cloud. One incremental tracker collects every 5 minutes, only the new events, into 5-minute buckets per index, never rescanning. The KPIs are per-index trends: rolling 60 min, 4 h, 12 h and 24 h volume and the daily licensed volume, as trackme.splk.vol.* metrics.

Detection runs one ML model per index on the rolling volume, drops and / or spikes, and a silent index turns red. Action is the impact score to state, stateful alerts, the 30-day trend and the month projection. The same diagram sits under How it works in the wizard.

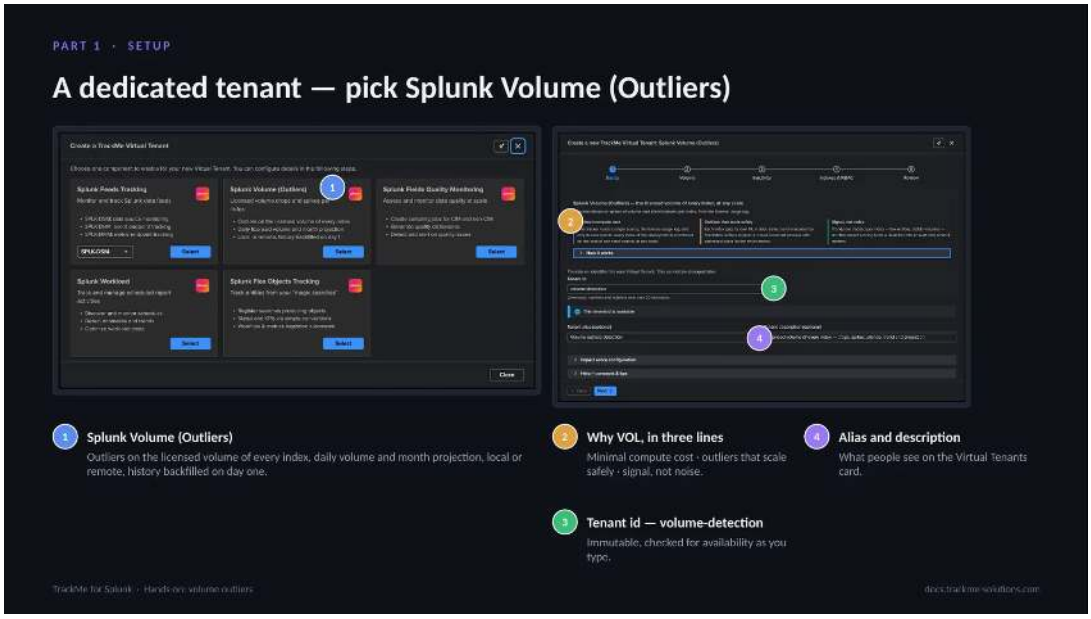
The four parts



One Virtual Tenant covers the whole deployment. Part 1, Setup, creates the tenant through the wizard (basics, volume, inactivity, RBAC), replays 30 days of history at creation, and follows day one from pending to complete. Part 2, Read, covers the VOL table (volume, yesterday, trend, projection), an index with its Volume & cost tab, rolling volume and hourly profile, and the outliers model with its bounds.

Part 3, Tune, sets drops, spikes or both per index and in bulk, the inactivity policy for business hours versus nights, and the default KPIs and outliers scope. Part 4, Operate, covers the global license usage view (scopes, working days, days over quota, costs and chargeback), SPLK-VOL alerts with stateful emails, the AI Assistant and the AI ML Advisor review of a volume model.

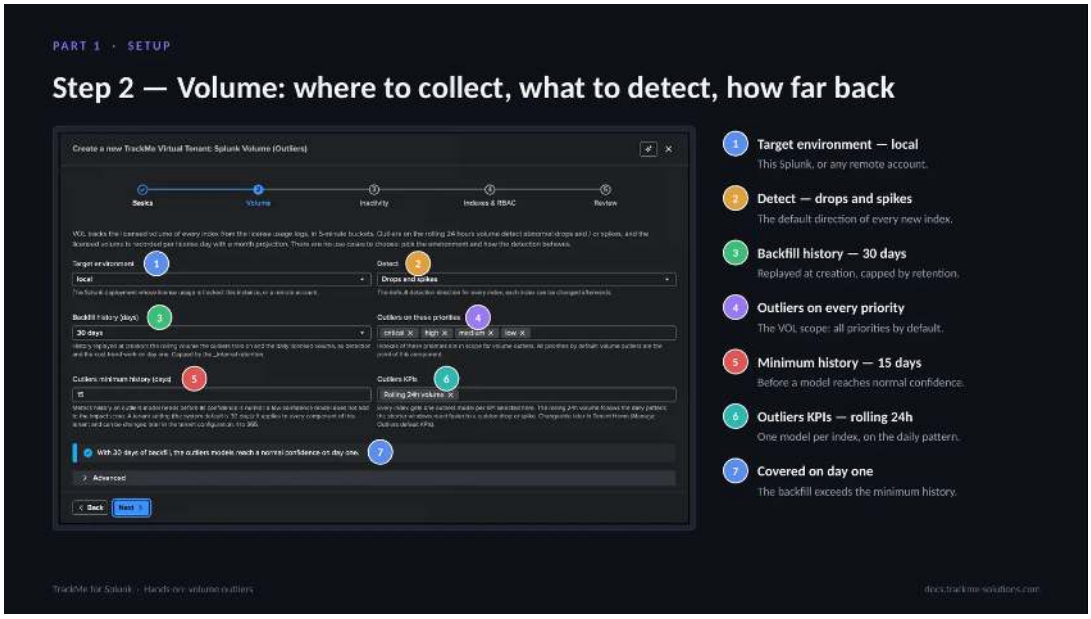
Choosing the Volume tenant type



The tenant chooser opens from Virtual Tenants, Actions, Create a tracking tenant, and the Splunk Volume (Outliers) card sits second in it (callout 1). The wizard has five steps: basics, volume, inactivity, indexes & RBAC, review. The first step recalls the three arguments of the component (callout 2), minimal compute cost, outliers that scale safely, signal rather than noise, and what VOL provides: outliers on the licensed volume of every index, daily volume and month projection, local or remote, history backfilled on day one.

The tenant id (callout 3) is immutable, checked for availability as it is typed, twenty characters at most; volume-detection is used here. The alias and description (callout 4) appear on the Virtual Tenants card.

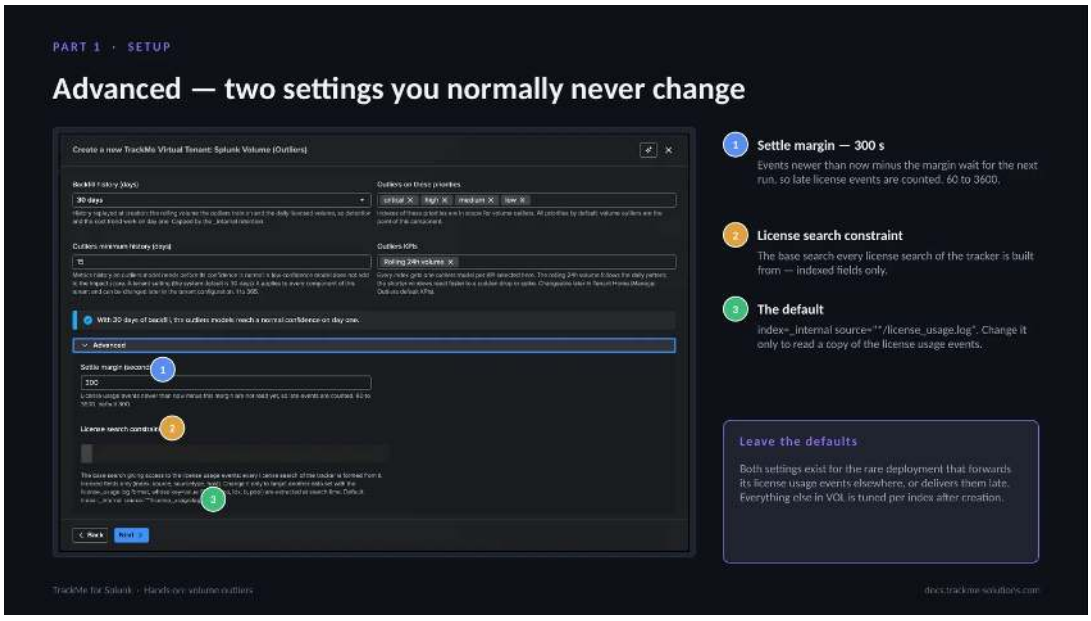
The Volume step and its defaults



The Volume step is the whole configuration and its defaults are the recommended values. Target environment (callout 1) is this Splunk or any remote account configured in TrackMe. Detect (callout 2) sets the default direction of every new index, drops and spikes, changed per index later. Backfill history (callout 3) is 30 days replayed at creation, capped by the retention of the license usage log.

The outliers scope (callout 4) covers every priority by default, because volume outliers are the point of this component. Minimum history (callout 5) is 15 days before a model reaches normal confidence; the notice (callout 7) confirms the backfill covers it. The default outliers KPI (callout 6) is the rolling 24h volume.

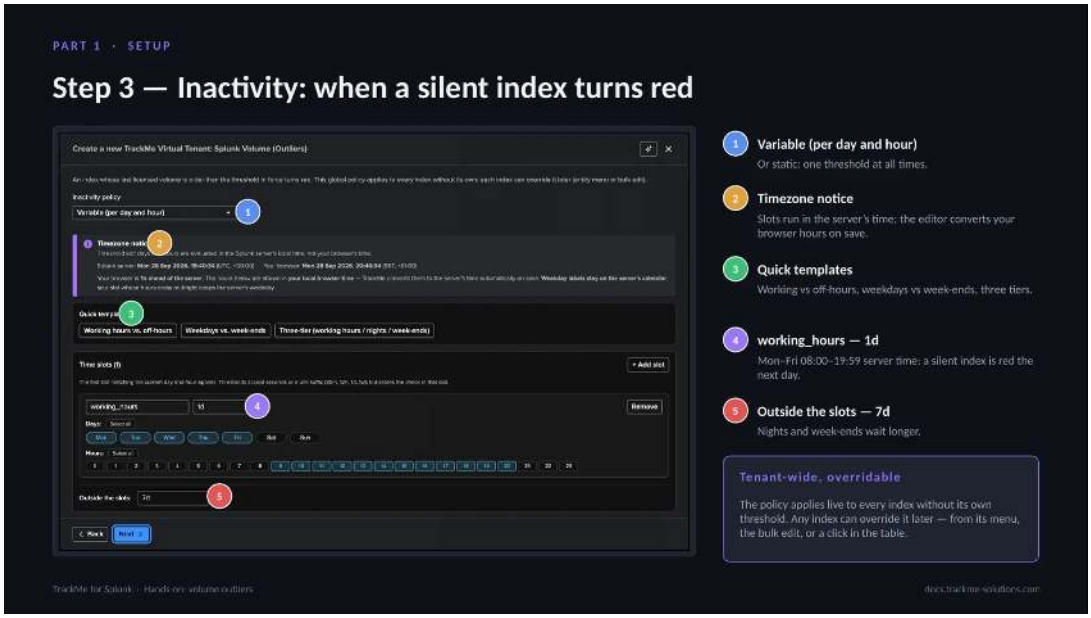
Settle margin and license search constraint



Expanding Advanced reveals two technical settings that normally stay at their defaults. The settle margin (callout 1), 300 s by default and adjustable from 60 to 3600, keeps the tracker behind the newest license events: events newer than now minus the margin wait for the next run, so late license events are still counted in their bucket.

The license search constraint (callout 2) is the base search every license search of the tracker is built from, indexed fields only. The default (callout 3) is index=_internal source="*/license_usage.log"; it changes only to read a copy of the license usage events. Both exist for the rare deployment that forwards its license usage events elsewhere or delivers them late.

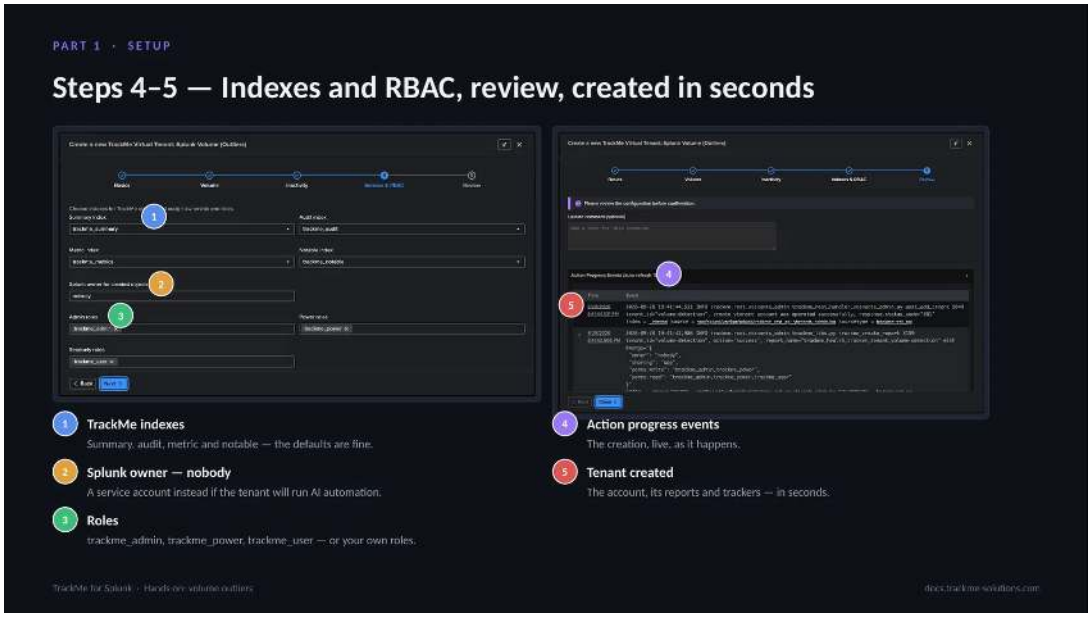
The inactivity policy by day and hour



Inactivity is the second detection: after how long without licensed volume an index turns red. The policy is variable by day and hour (callout 1), or static with one threshold. The timezone notice (callout 2) shows the server and browser clocks side by side: slots run in the server's time and the editor converts browser hours on save. Quick templates (callout 3) cover working versus off-hours, weekdays versus week-ends and three tiers.

The default working_hours slot (callout 4), Monday to Friday 08:00 to 19:59 server time, uses 1d: a silent index is red the next day. Outside the slots (callout 5) the threshold is 7d. The policy applies live to every index without its own threshold.

Indexes, RBAC and creation

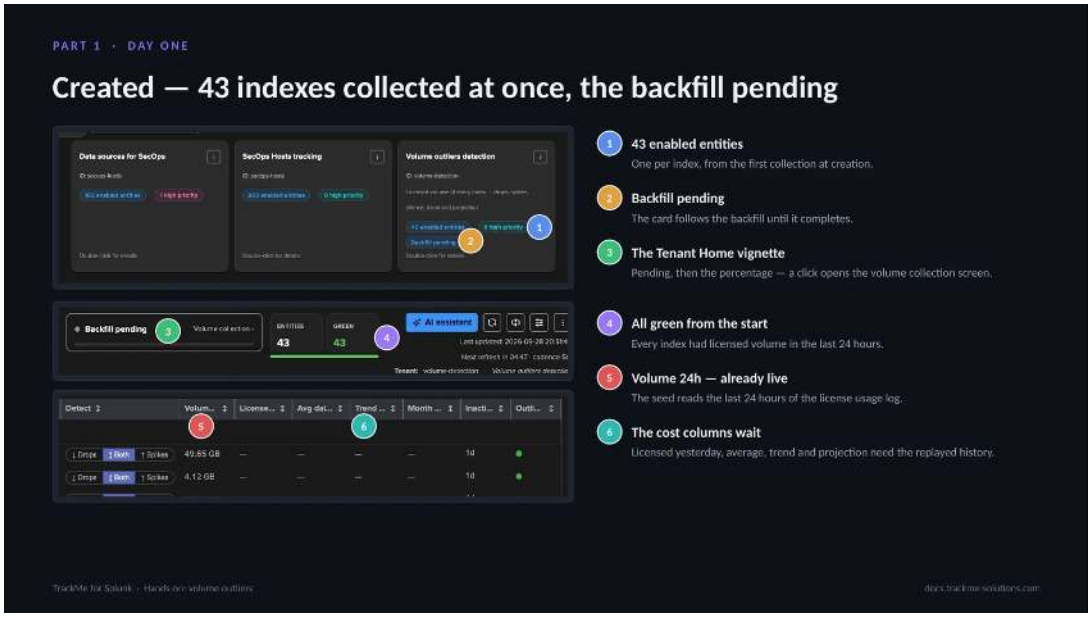


The last two steps are shared by every tenant type. Indexes & RBAC sets the TrackMe indexes (callout 1): sum-

mary, audit, metric and notable, where the defaults are fine. The Splunk owner (callout 2) is nobody by default; a service account is the better choice if the tenant will run AI automation. Roles (callout 3) default to trackme_admin, trackme_power and trackme_user, or any custom roles.

The review step is followed by Done, and the action progress events (callout 4) show the creation live as it happens: the tenant account, its reports and its trackers are created in seconds (callout 5), and the first collection runs straight away.

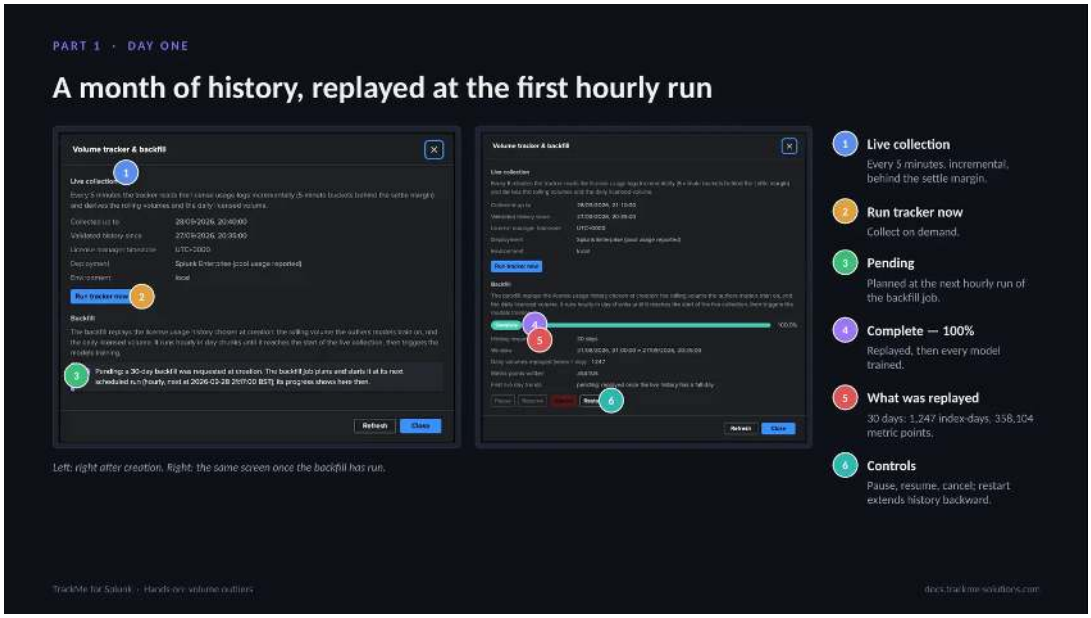
First collection and the pending backfill



The creation runs the first collection: the tracker seeds the last 24 hours of the license usage log and one entity per index appears, 43 enabled entities here (callout 1), each with its rolling volume. The backfill job plans itself at its first hourly run; until then the Virtual Tenants card reads Backfill pending (callout 2) and follows the backfill until it completes.

The Tenant Home header shows a matching vignette (callout 3): pending, then the percentage; a click opens the volume collection screen. Every index is green from the start (callout 4) because each had licensed volume in the last 24 hours. Volume 24h is already live (callout 5); the cost columns (callout 6) wait for the replayed history.

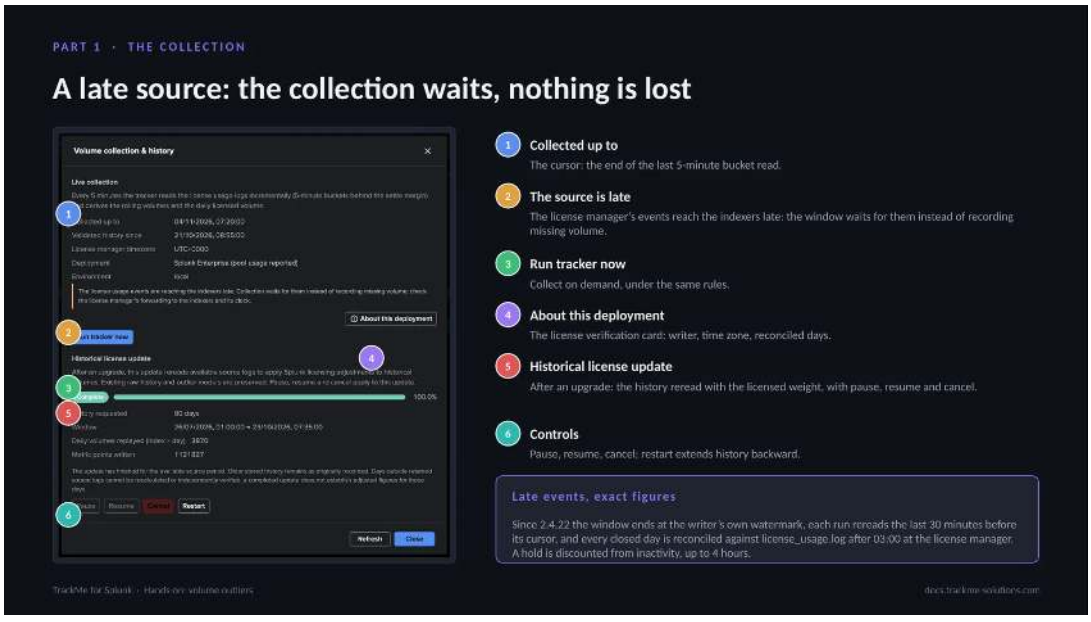
The volume collection screen



The volume collection screen explains both jobs, shown right after creation on the left and after the backfill on the right. Live collection (callout 1) runs every 5 minutes, incrementally, behind the settle margin; Run tracker now (callout 2) collects on demand. The backfill is Pending (callout 3) until the backfill job plans it at its next hourly run, then Complete at 100% (callout 4): the history is replayed day by day through the same code as the live collection, with historical timestamps.

The replay summary (callout 5) reports 30 days, 1,247 index-days and 358,104 metric points. The controls (callout 6) pause, resume and cancel; a restart extends the history backward. On completion every model is trained.

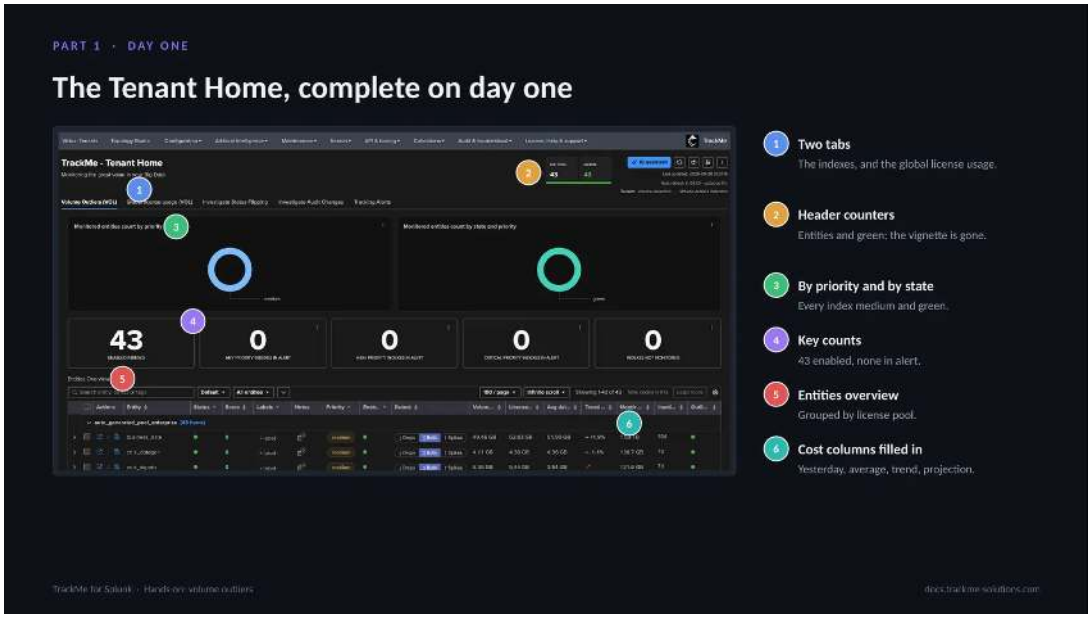
A late source and the collection



The Volume collection & history screen on a deployment whose license manager delivers its events late. Collected up to names the cursor (callout 1); the notice (callout 2) says that the license usage events are reaching the indexers late and that the collection waits for them instead of recording missing volume. Run tracker now collects on demand (callout 3) and About this deployment opens the license verification card (callout 4). The Historical license update (callout 5) is the post-upgrade replay that applied the licensed weight to the stored history, with its pause, resume, cancel and restart controls (callout 6).

Since 2.4.22 the window ends at the writer's own watermark, each run rereads the last 30 minutes before its cursor, and every closed day is reconciled against license_usage.log after 03:00 at the license manager; a hold is discounted from inactivity, up to four hours.

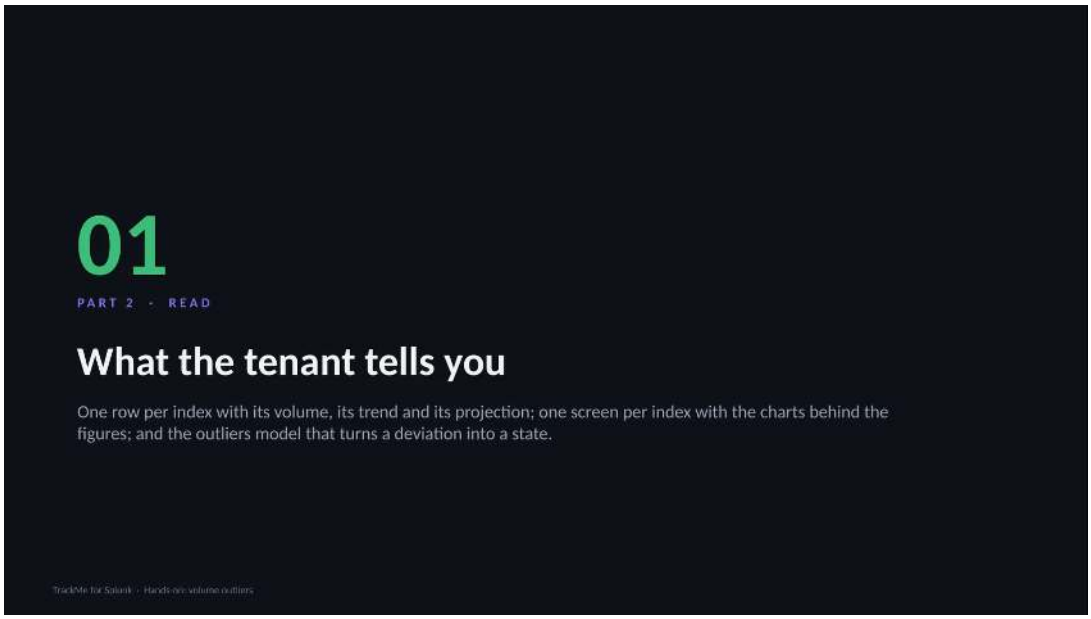
Tenant Home after the backfill



Once the backfill has completed, the Tenant Home is the finished product. Two tabs (callout 1) hold the Volume Outliers indexes and the global license usage. The header counters (callout 2) show the entities and the green count, and the backfill vignette is gone. The charts by priority and by state (callout 3) show every index medium and green, and the key counts (callout 4) read 43 enabled and none in alert.

The entities overview (callout 5) is grouped by license pool, and the cost columns (callout 6), yesterday, average, trend and projection, are now filled from the replayed month.

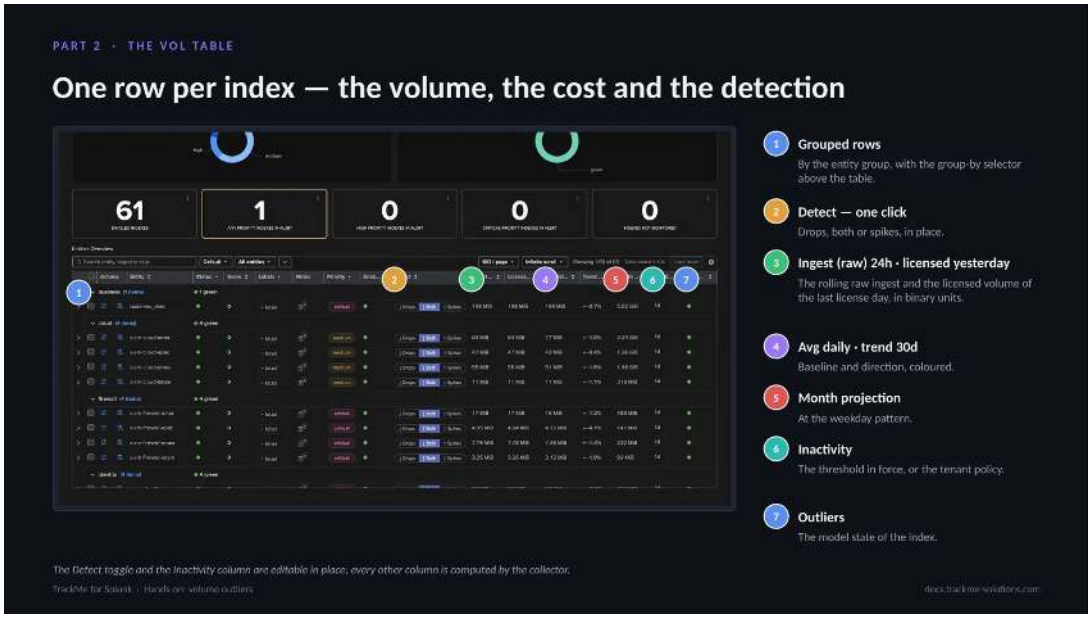
Part 2 — Read



Part 2 reads what the tenant shows. The VOL table has one row per index with its volume, its trend and its projection. The entity screen of an index has the charts behind the figures: the daily licensed volume, the rolling volume windows and the hourly profile.

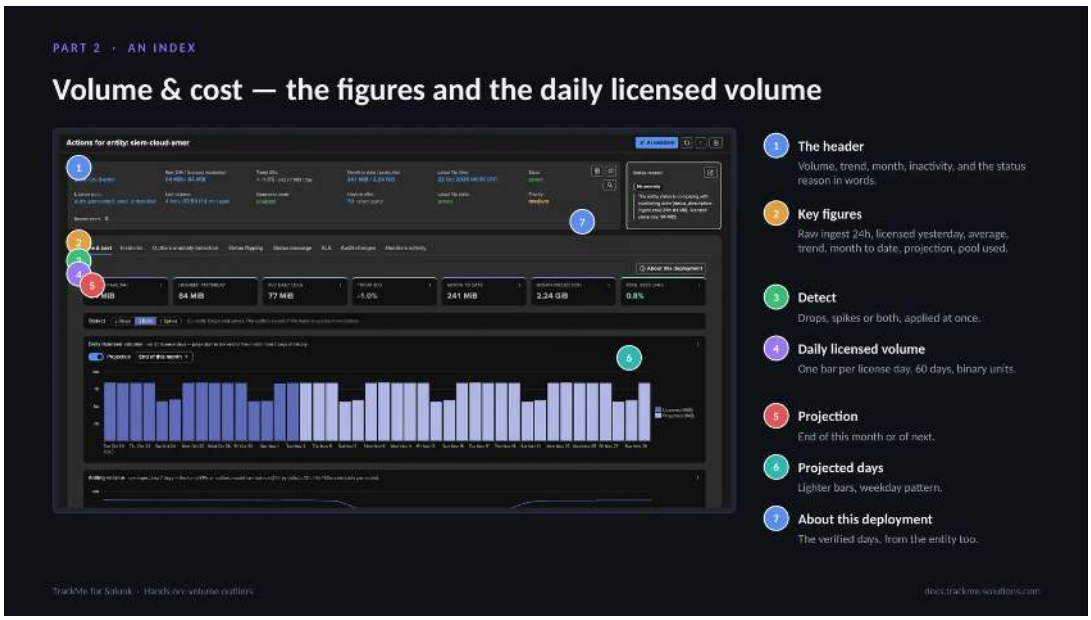
Behind each state sits the outliers model that turns a deviation into a state, with its bounds, its direction and its confidence.

Reading the VOL table columns



One row per index, grouped by the entity group with the group-by selector above the table (callout 1). The Detect toggle switches the direction in place (callout 2). Ingest (raw) 24h is the rolling raw ingest and Licensed yesterday the licensed volume of the last license day, both in binary units (callout 3); Avg daily and the 30-day trend give the baseline and its direction (callout 4), the month projection follows the weekday pattern (callout 5), Inactivity shows the threshold in force or the tenant policy (callout 6) and Outliers the model state (callout 7). Everything but the Detect toggle and the Inactivity column is computed by the collector every five minutes.

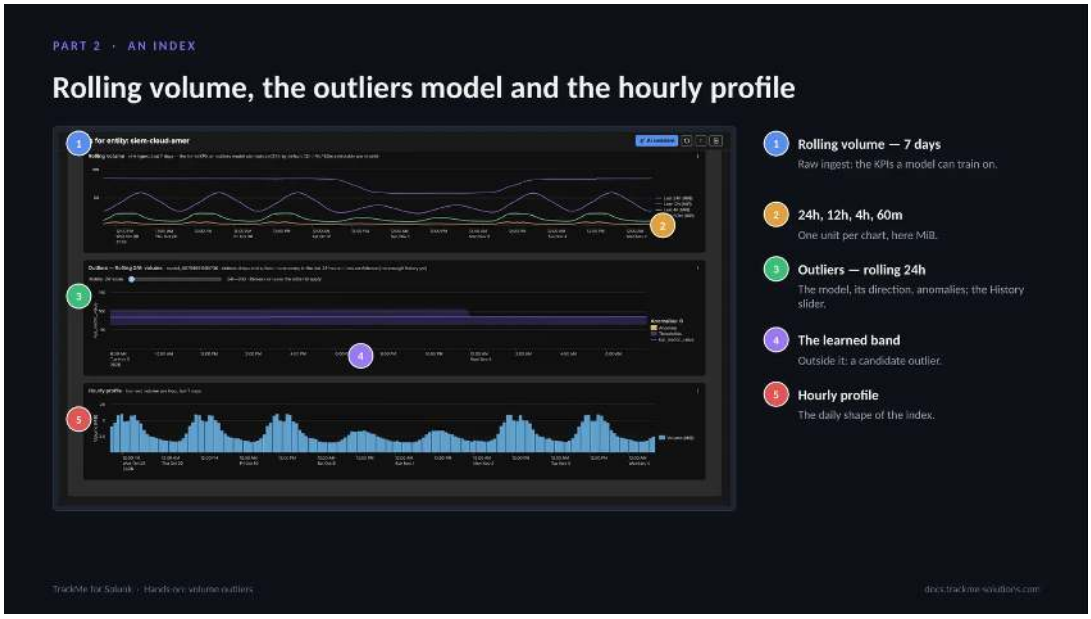
The Volume & cost tab



The entity opens on Volume & cost. The header gives the volume, the trend, the month, the inactivity and the status reason in words (callout 1). The key figures (callout 2) are the raw ingest of the last 24 hours, licensed yesterday, the 30-day average, the trend, month to date, the month projection and the pool used; the Detect control applies the direction at once (callout 3). The daily licensed volume chart (callout 4) draws one bar per license day over 60 days,

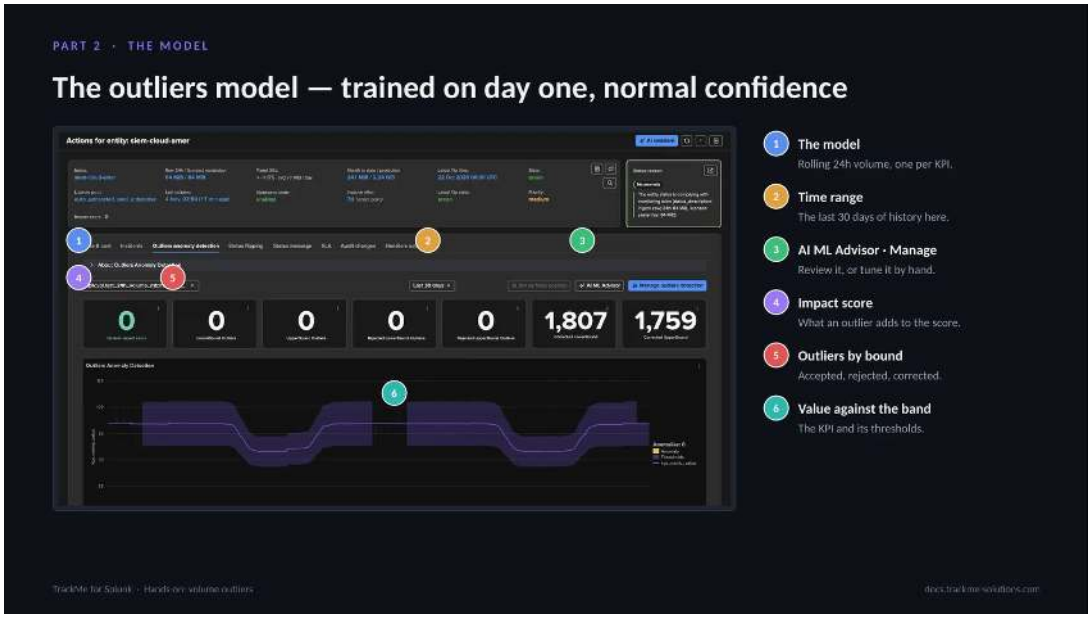
in binary units, with the projection switch (callout 5) and the projected days as lighter bars (callout 6). About this deployment (callout 7) opens the verified days from the entity too.

Rolling volume and hourly profile charts



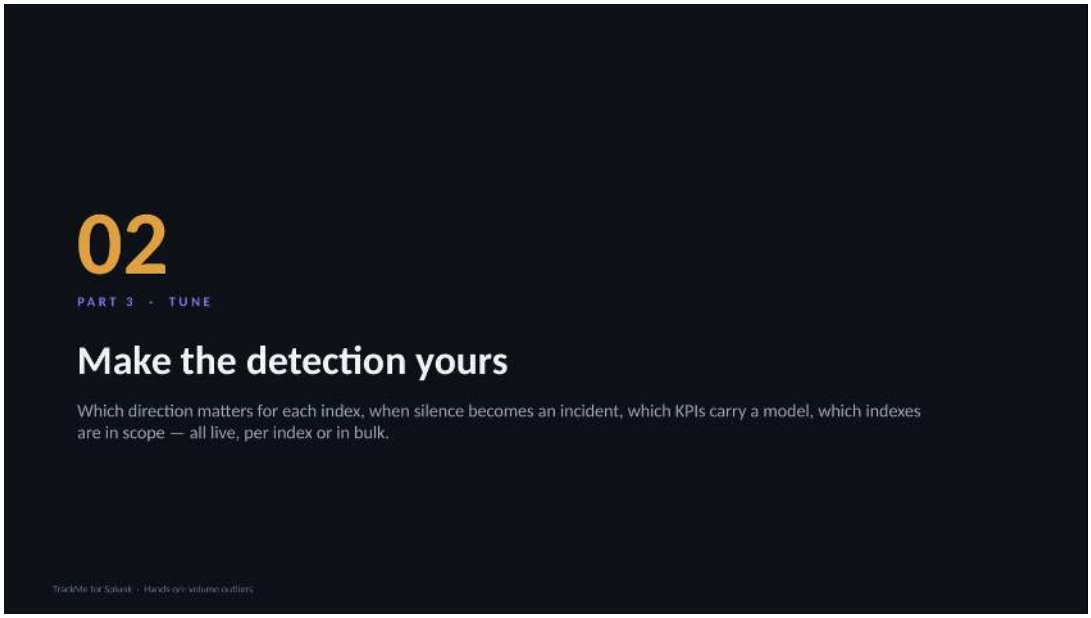
The rolling volume chart (callout 1) draws the last 7 days of the raw ingest, the KPIs a model can train on: 24 hours, 12 hours, 4 hours and 60 minutes, one unit per chart (callout 2). The outliers chart (callout 3) shows the model on the rolling 24-hour volume with its direction and anomalies, and the History slider widens the window; a value outside the learned band is a candidate outlier (callout 4). The hourly profile (callout 5) draws the daily shape of the index, which explains the band.

The outliers model of an index



The Outliers anomaly detection tab carries the model of the index on the rolling 24-hour volume (callout 1) over the time range picked, here the last 30 days (callout 2). AI ML Advisor and Manage outliers detection (callout 3) review it or tune it by hand. The counters give the impact score of an outlier (callout 4) and the outliers by bound, accepted, rejected and corrected (callout 5); the chart draws the KPI against its thresholds (callout 6). Because the history was backfilled, the model reached normal confidence on day one.

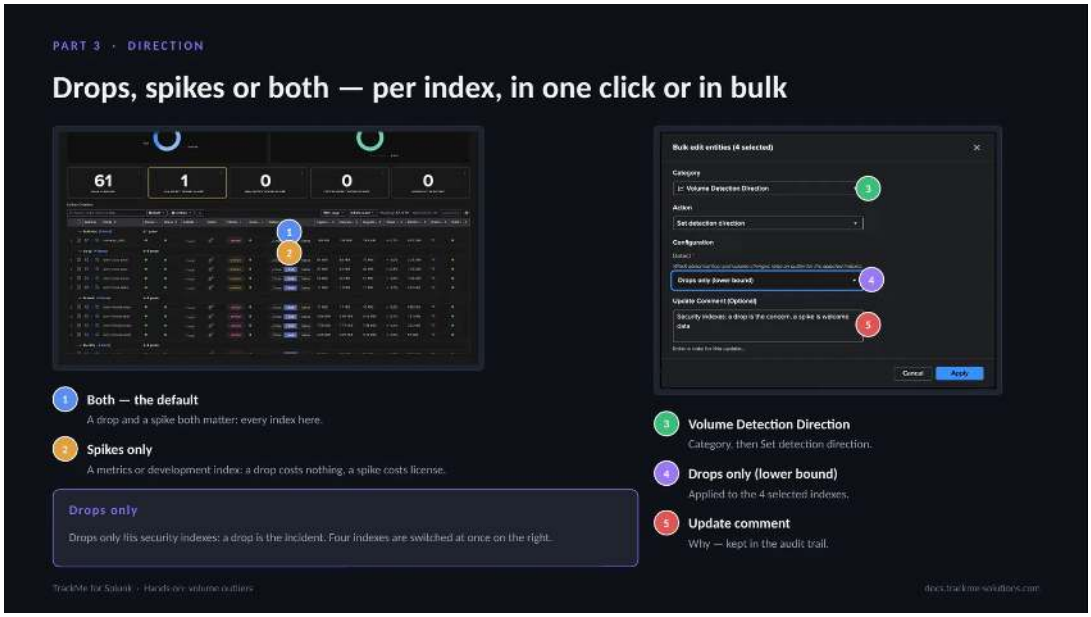
Part 3 — Tune



Part 3 tunes the detection. The defaults are sensible; these are the settings operators actually use: which direction matters for each index, when silence becomes an incident, which KPIs carry a model, and which indexes are in scope.

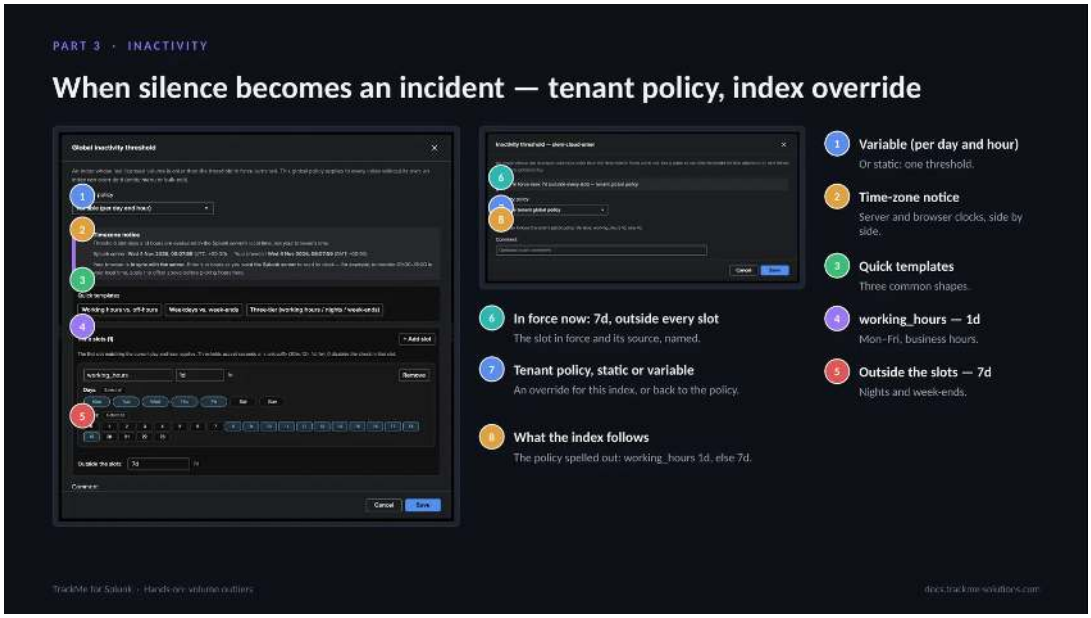
Every setting applies live, per index or in bulk.

Setting the detection direction



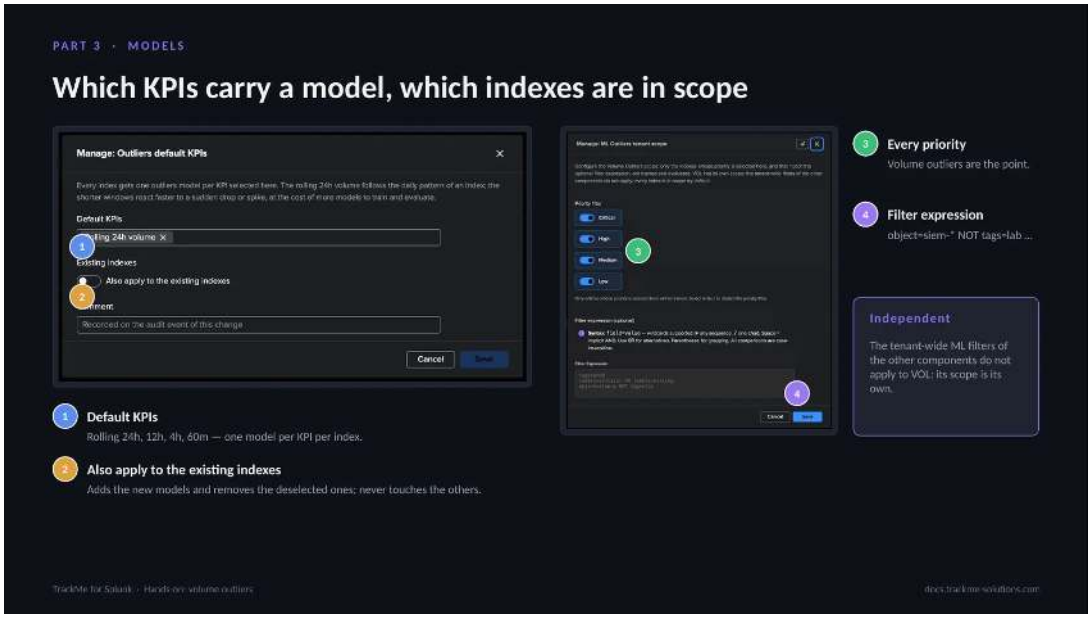
The Detect toggle in the table switches one index (callouts 1 and 2): both directions is the default, a metrics or development index is a spike concern, a security index a drop concern. The bulk edit category Volume Detection Direction (callout 3) applies Drops only, Spikes only or both to a selection (callout 4), with an update comment kept in the audit trail (callout 5). The direction is stored on the index's outliers model and takes effect immediately.

Tenant inactivity policy and index overrides



Manage: Global inactivity threshold edits the tenant policy: variable per day and hour, or static (callout 1), with a time-zone notice that shows the server and browser clocks side by side because the slots are evaluated on the server (callout 2), quick templates (callout 3), the working_hours slot at one day (callout 4) and the threshold outside every slot, seven days (callout 5). The per-index modal names the slot in force and its source before anything changes (callout 6), offers the tenant policy, a static or a variable override (callout 7) and spells out what the index follows (callout 8).

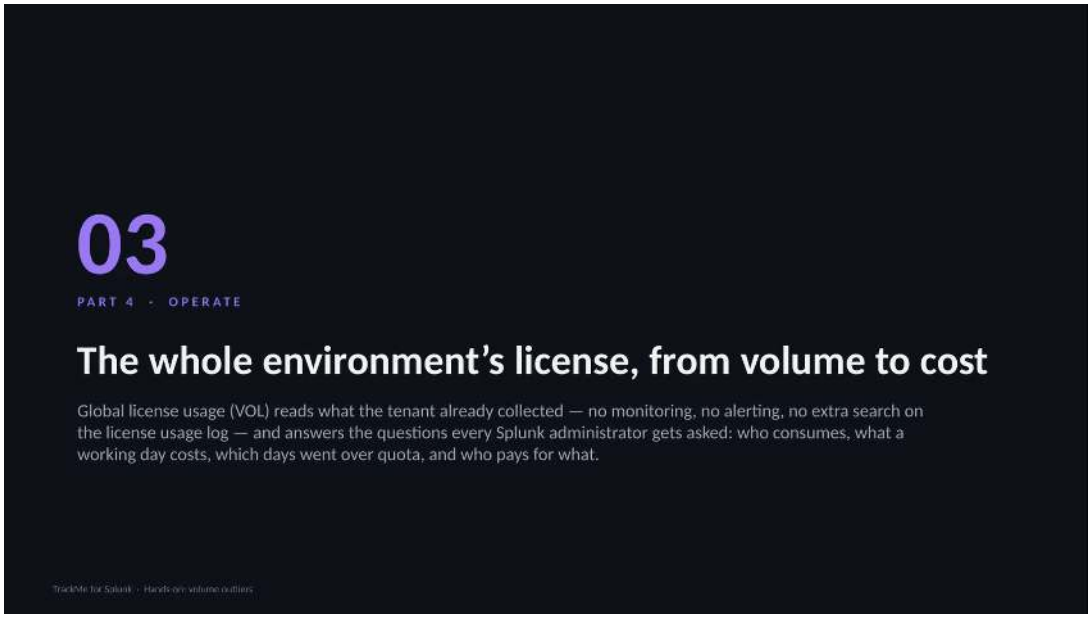
Default KPIs and the outliers scope



Manage: Outliers default KPIs (callout 1) sets the rolling KPIs every new index gets a model on: rolling 24h, 12h, 4h and 60m, one model per KPI per index, the 24-hour volume by default. The shorter windows react faster, at the cost of more models. Also apply to the existing indexes (callout 2) adds the new models and removes the deselected ones, never touching the others.

Manage: ML Outliers tenant scope sets which indexes are trained and evaluated. Every priority is in scope by default (callout 3), and an optional filter expression (callout 4), such as object=siem-* NOT tags=lab, narrows on object, priority, tags, labels and pool. The other components' ML filters do not apply to VOL.

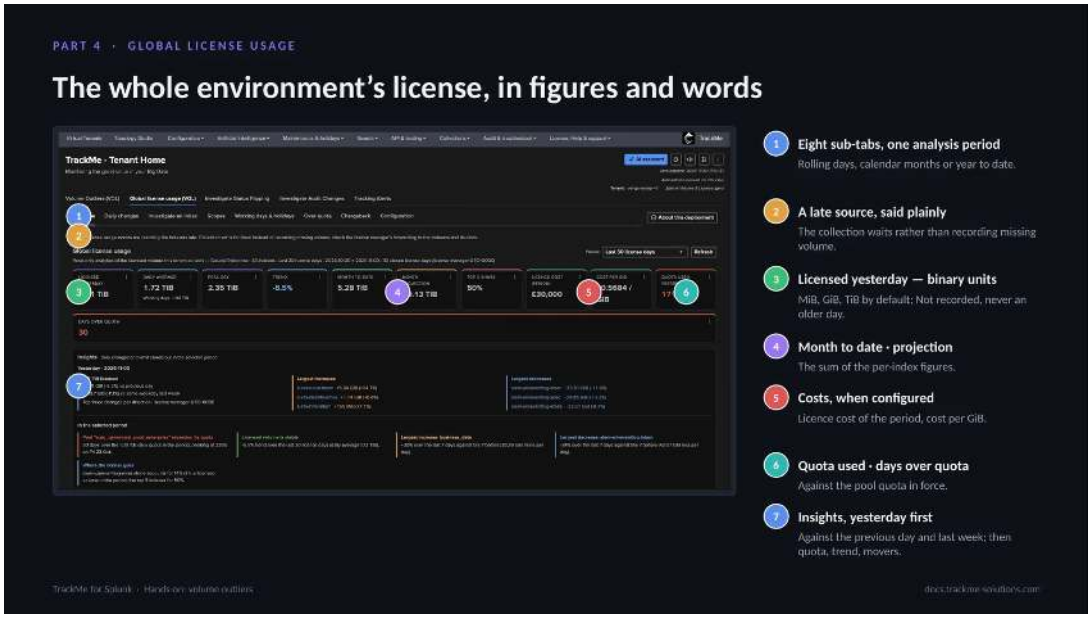
Part 4 — Operate



Part 4 covers the global license usage view, alerting, and the AI Advisors. Global license usage (VOL) takes the whole environment's licence from volume to cost.

It reads what the tenant already collected, with no monitoring, no alerting and no extra search on the license usage log, and answers the questions every Splunk administrator gets asked: who consumes, what a working day costs, which days went over quota, and who pays for what.

Global license usage overview



The Overview of the Global license usage view. Eight sub-tabs share one analysis period (callout 1): Overview, Daily changes, Investigate an index, Scopes, Working days and holidays, Over quota, Chargeback and Configuration, on rolling days, calendar months or the year to date. On this deployment the license manager delivers its events late, and the banner says that the collection waits for them rather than recording missing volume (callout 2). Licensed yesterday is the last closed license day in binary units, MiB, GiB and TiB by default, or Not recorded (callout 3); month to date and the projection sum the per-index figures (callout 4). With costs configured, the licence cost of the period and the cost per GiB appear (callout 5); with a quota in force, the quota used yesterday and the days over quota (callout 6). The insights start with yesterday against the previous day and the same weekday last week (callout 7).

The figures verified against the license log



About this deployment leads with the verification state. Daily figures successfully reconciled means that TrackMe's totals and per-index figures match the detailed Usage records of license_usage.log (callout 1); the days where Splunk's own RolloverSummary reports more or less than those records are listed with the difference in MiB and in percent (callout 2), then the range of days matched against the license manager (callout 3). The Deployment card names the platform, the Usage writer and the monitored indexes (callout 4); the Collection card the collected-through time, the license day frame and the unit labels (callout 5).

Technical details and verification dates lists the license source and one row per day (callout 6) with its status and revision (callout 7), both byte totals, the last successful reconciliation and the next attempt (callout 8).

Daily consumption and month-end projection

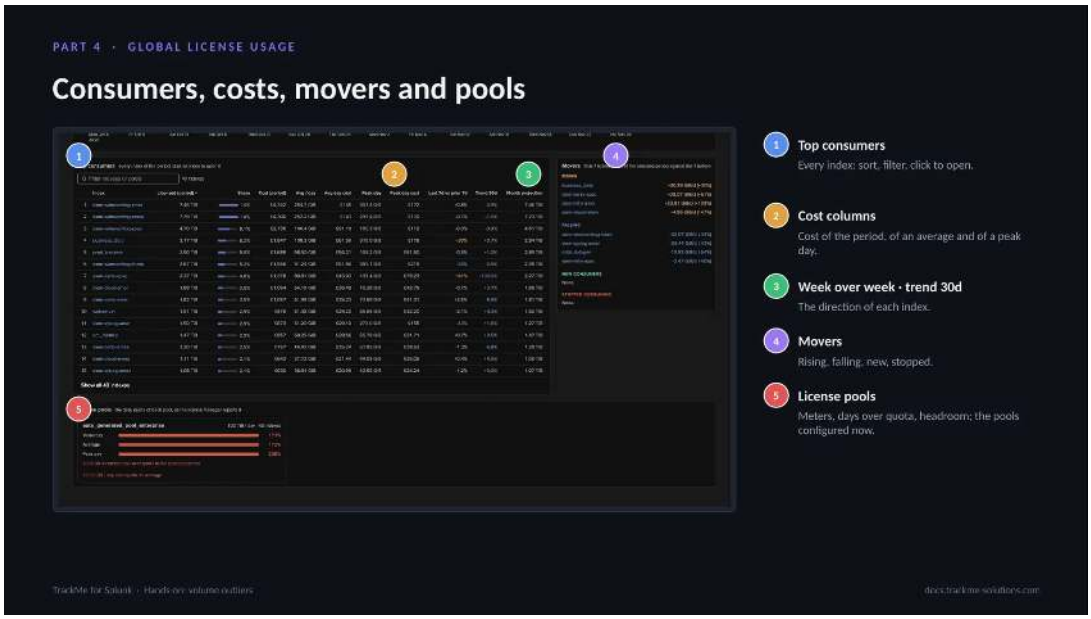


The first chart stacks the daily consumption per index (callout 3), each closed license day by consumer. The top consumers slider (callout 1) selects 5 to 100 series, with the rest grouped together, and the daily quota line (callout 2) shows the day the pool went over its quota at a glance.

The second chart shows the whole environment with the projection switch (callout 4): to the end of this month or of next month, the same rule as the entity chart. The projected days (callout 5) appear as lighter bars following the

weekday pattern.

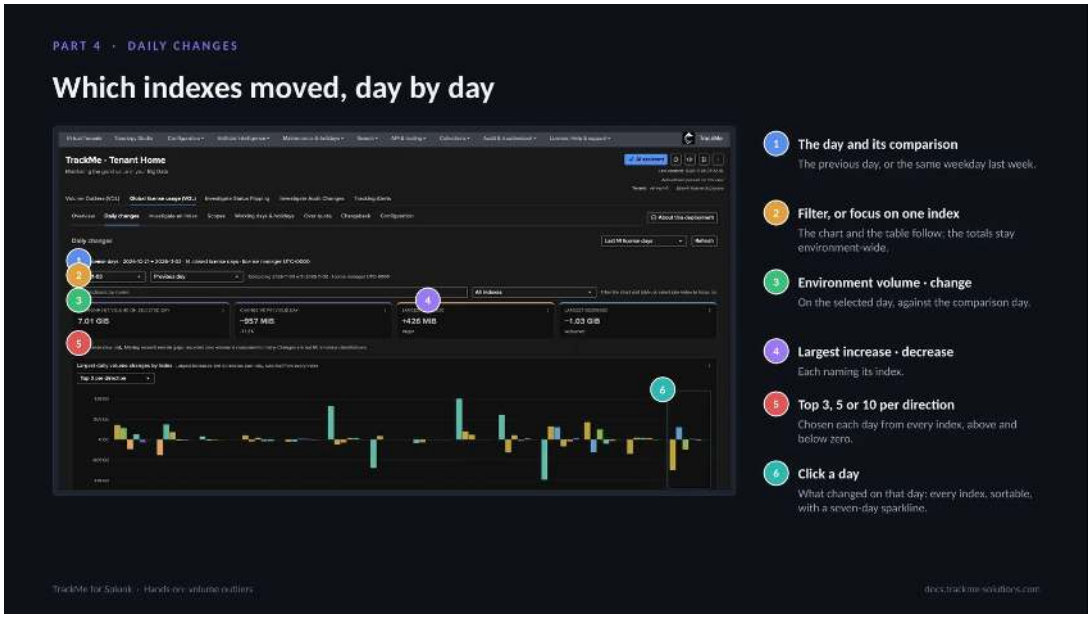
Consumers, costs, movers and pools



The top consumers table (callout 1) lists every index of the period; it sorts, filters and opens an index on click. With costs on, the cost columns (callout 2) give the cost of the period, of an average day and of a peak day, and the week-over-week and 30-day trend columns (callout 3) give each index's direction.

The movers card (callout 4) turns the week-over-week comparison into rising, falling, new and stopped indexes: here siem-osnix-apac rose by 44 %. The license pool card (callout 5) gives the capacity view: meters for yesterday, the average and the peak day, the days over quota and the headroom.

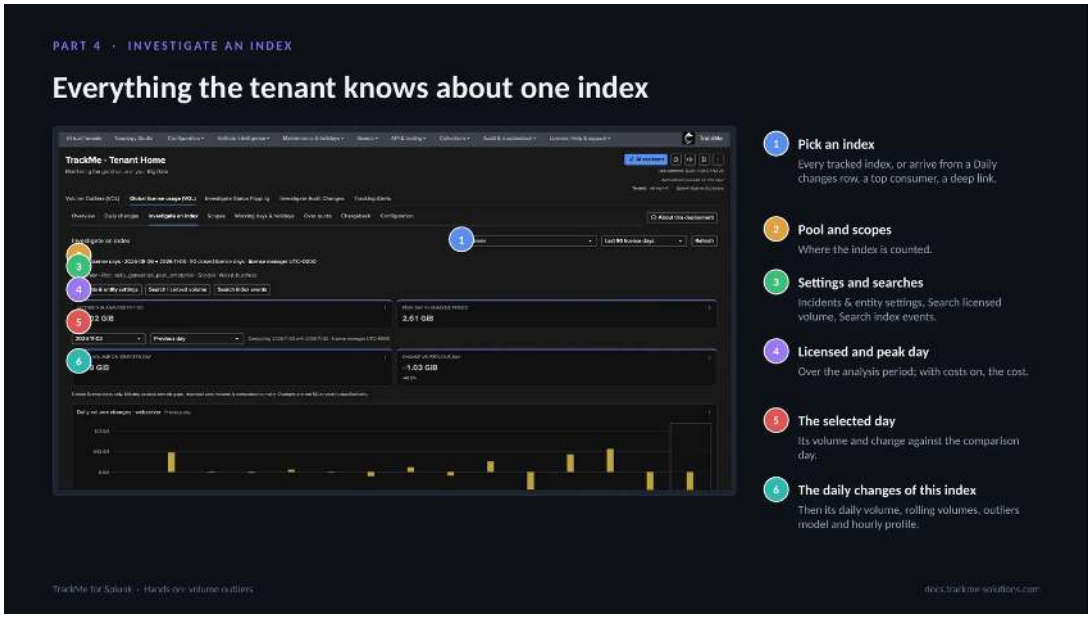
Daily changes



Daily changes compares each closed license day with the previous day or with the same weekday a week earlier (callout 1). Filter indexes narrows the chart and the table by name, and the index selector focuses them on one index (callout 2). The cards give the environment volume on the selected day and its change (callout 3), then the largest increase and the largest decrease, each naming its index (callout 4). The chart ranks the top 3, 5 or 10 increases and decreases of each day, chosen from every index (callout 5); clicking a day opens the What changed on table with

every index, sortable, with a seven-day sparkline (callout 6). Missing days stay gaps and a recorded zero is compared as zero.

Investigate an index



Investigate an index gathers what the tenant knows about one index, read-only. The selector lists every tracked index (callout 1); a row of Daily changes, a top consumer or a deep link arrives here too. The index's pool and scopes (callout 2) and the Incidents & entity settings, Search licensed volume and Search index events buttons (callout 3) sit above the licensed volume and peak day of the analysis period, with the cost when costs are on (callout 4). The selected day is compared with its comparison day (callout 5), followed by the daily changes of this index alone, its daily volume with the projection, the rolling volumes, the outliers model and the hourly profile (callout 6).

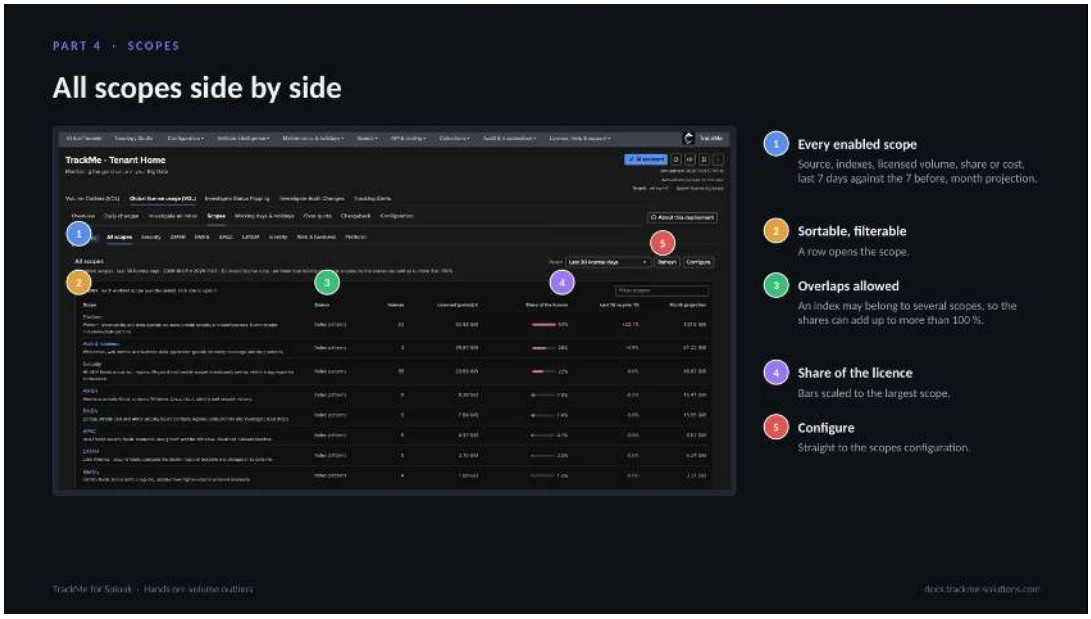
Named scopes



A scope is a set of indexes analysed together. The second row holds All scopes and one tab per enabled scope, a filterable selector once they no longer fit (callout 1). The scope's source is index patterns or the CIM datamodel macros, read on this search head or through a remote account on the Enterprise Security tier (callout 2). Its tab gives the licensed volume, the share of the licence or the cost of the period (callout 3), the daily average with the

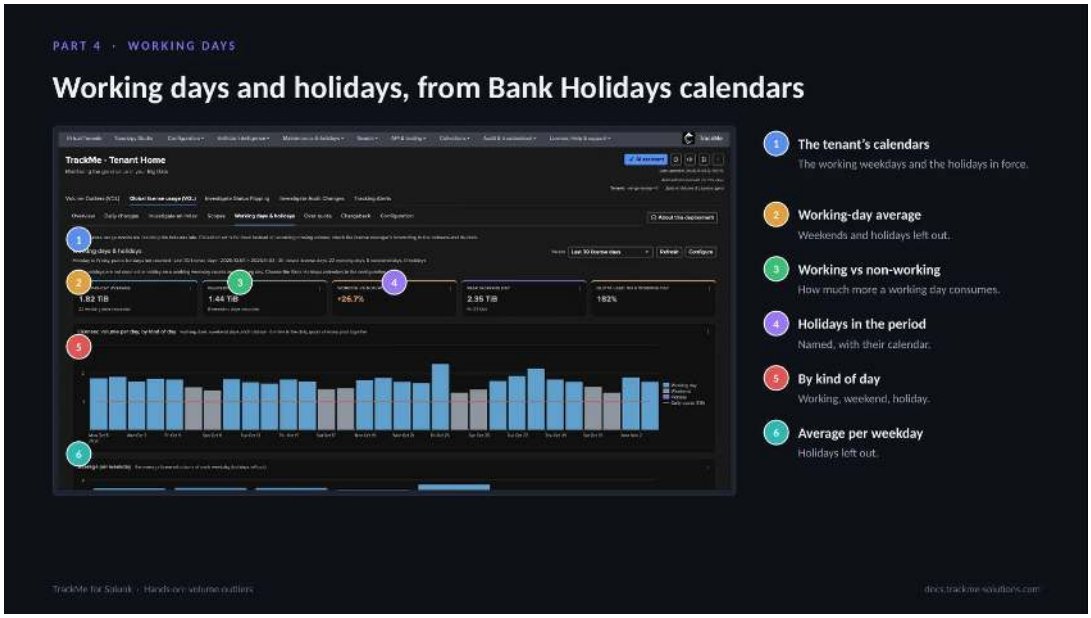
working-day average (callout 4), the peak day, the last 7 days against the 7 before, the month projection, and the daily volume stacked by index with the scope's own projection below (callout 5).

All scopes side by side



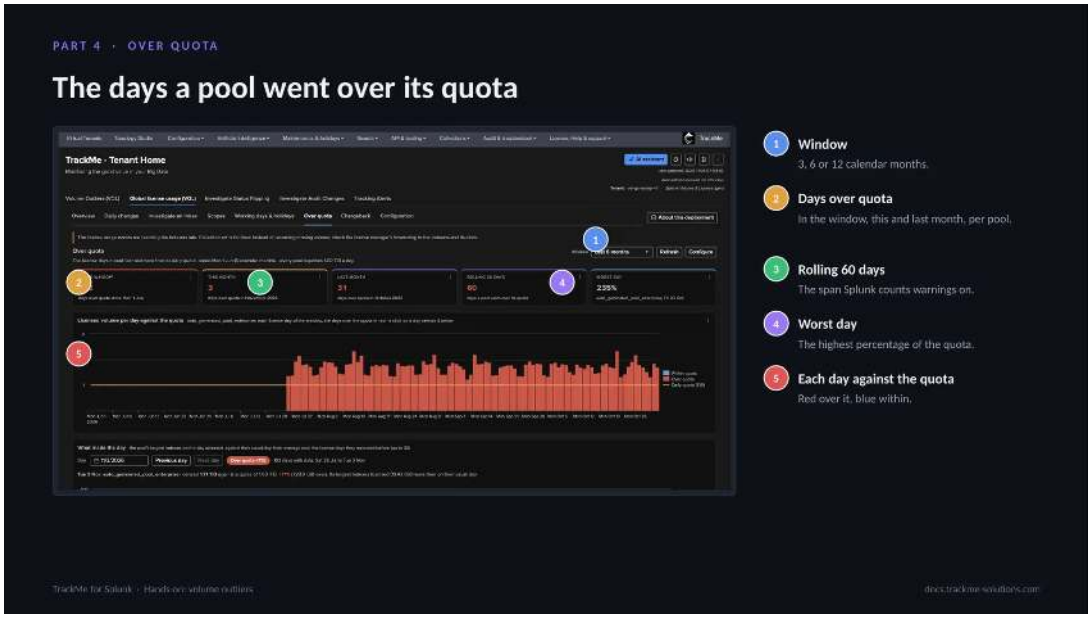
All scopes compares every enabled scope side by side (callout 1): its source, its indexes, the licensed volume of the period, the share of the licence or the cost (callout 2), the last 7 days against the 7 before and the month projection. The headers sort and the filter narrows by name; a row opens the scope. An index may belong to several scopes, so the shares can add up to more than 100 % (callout 3); the share bars are scaled to the largest scope (callout 4). Configure opens the scopes configuration (callout 5). The view is computed from the per-index search the Overview already runs, so it carries no daily average or peak.

Working days and holidays



The Working days & holidays sub-tab separates working days, weekends and holidays. The tenant's calendars and working weekdays are named (callout 1); the working-day average leaves weekends and holidays out (callout 2) and the view says how much more a working day consumes than a non-working one (callout 3); the holidays of the period are listed with their calendar (callout 4). The daily chart colours each license day by kind of day (callout 5) and the average per weekday leaves the holidays out (callout 6).

Days over quota



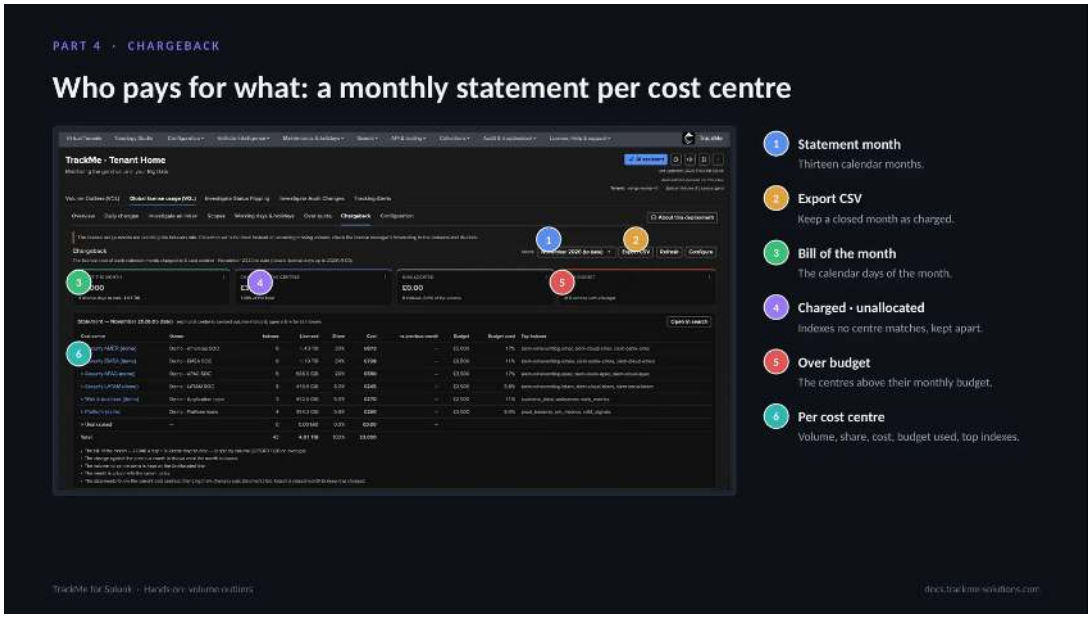
Over quota needs a quota in force: the license manager's pool size on Splunk Enterprise, or a daily quota configured per pool. The window covers 3, 6 or 12 calendar months (callout 1); the cards count the days over quota in the window, this month and last month, per pool (callout 2), the rolling 60 days Splunk counts warnings on (callout 3) and the worst day as a percentage of the quota (callout 4). The chart draws each day against the quota, red over it and blue within (callout 5). These are days over quota, not Splunk license warnings.

What made the day



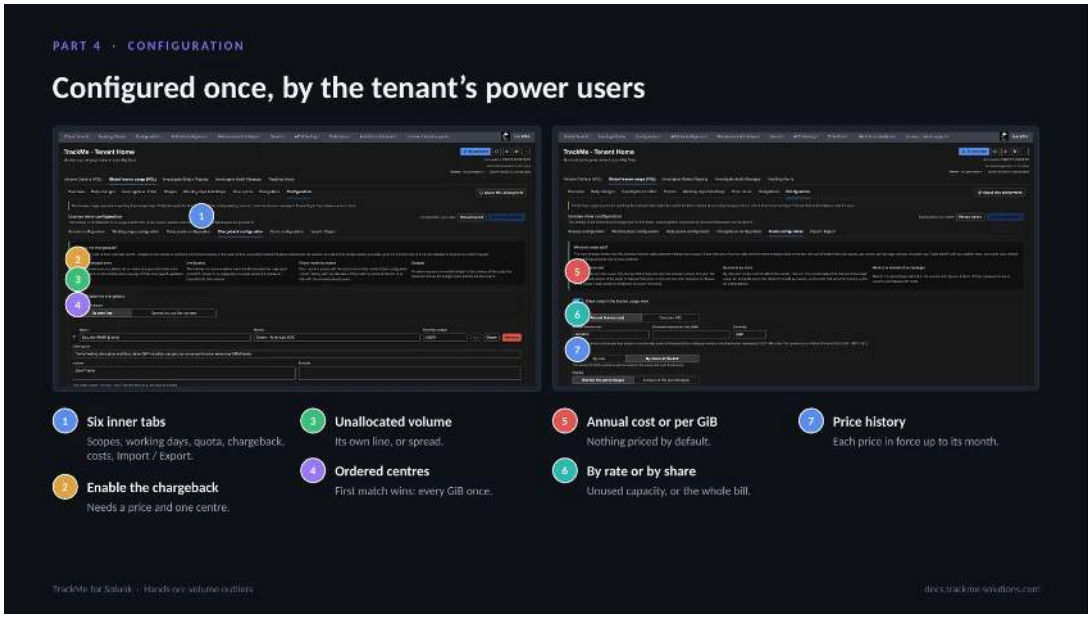
What made the day compares the pool's largest indexes on the day picked with their usual day, their average over the license days they recorded before. The day selector defaults to the latest day over quota (callout 1), Previous day and Next day step through the window (callout 2), and the badge says by how much the day was over or within its quota (callout 3). The chart draws that day against the usual day per index (callout 4), the table reads the same comparison in figures with the share of the day (callout 5), and the list of days over quota opens any of them (callout 6).

Chargeback statement



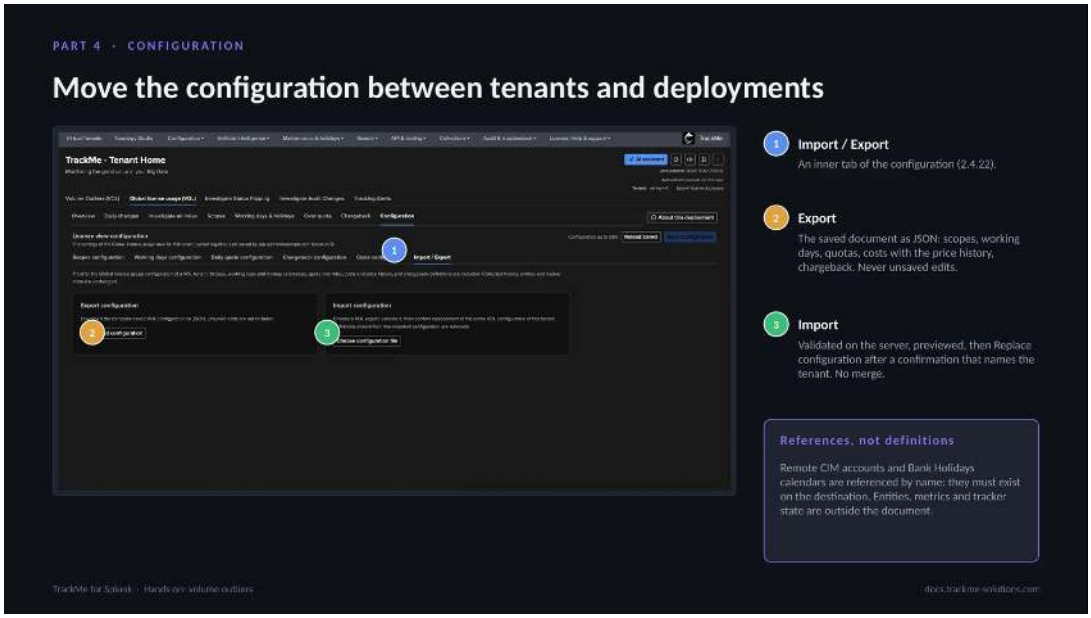
The Chargeback sub-tab turns the licence into a monthly statement per cost centre. The statement month covers thirteen calendar months (callout 1) and Export CSV keeps a closed month as charged (callout 2). The cards give the bill of the month (callout 3), the volume charged to the centres and the unallocated volume kept apart (callout 4) and the centres over their monthly budget (callout 5). The statement lists each centre's volume, share, cost, budget used and top indexes (callout 6), followed by the cost per cost centre month by month. An index belongs to the first centre that matches it, so every GiB is charged once.

Configuring the license view



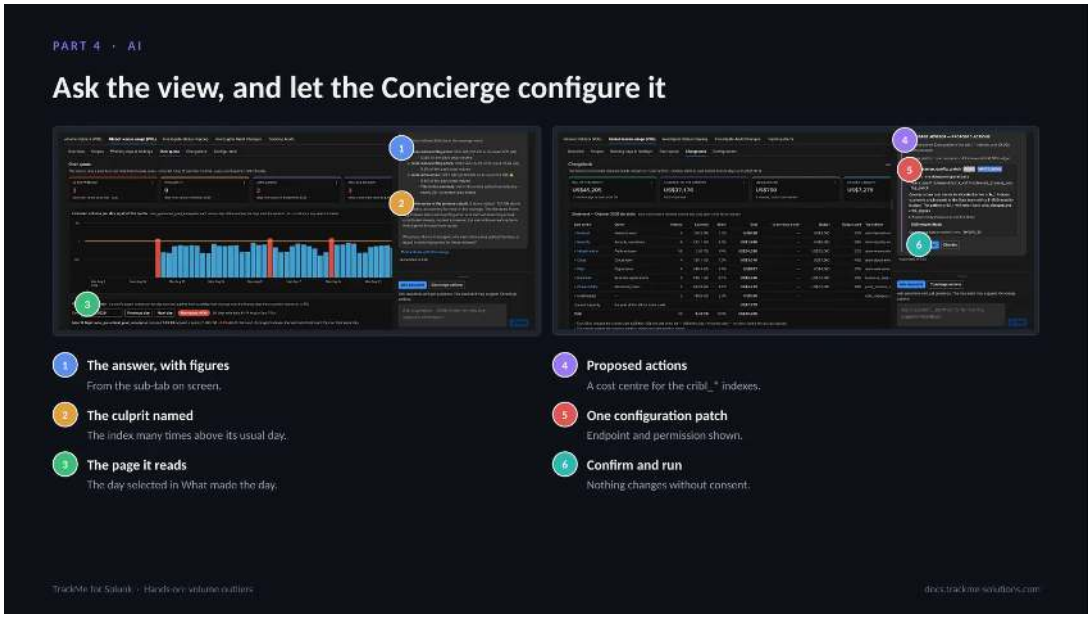
The Configuration sub-tab holds six inner tabs (callout 1): scopes, working days, daily quota, chargeback, costs and Import / Export, saved together with a revision so two people editing at once never overwrite each other. The chargeback section is enabled with a switch (callout 2), keeps the unallocated volume as its own line or spreads it (callout 3) and holds the ordered cost centres with an owner, a monthly budget and include and exclude patterns, previewing the indexes each resolves to (callout 4). The costs section holds an annual licence cost with the licensed volume per day or a cost per GiB (callout 5), the allocation by rate or by share of the bill (callout 6) and the price history, so a renewal never reprices past months (callout 7).

Moving the configuration



Import / Export, an inner tab of the configuration (callout 1), moves the whole document between tenants and deployments. Export downloads the saved configuration as a JSON file: scopes, working days, quotas, costs with the price history and chargeback, never unsaved edits (callout 2). Import reads such a file, validates it on the server, shows the summary and the JSON, and Replace configuration replaces the destination's document after a confirmation that names the tenant, with no merge (callout 3). Remote CIM accounts and Bank Holidays calendars are referenced by name and must exist on the destination.

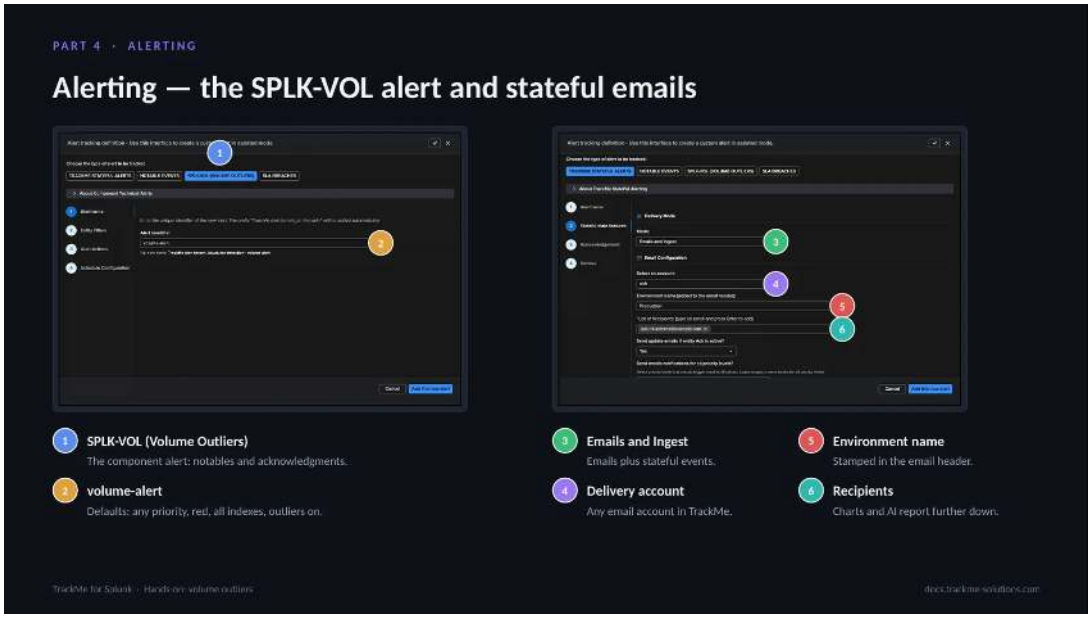
The AI Assistant on the license view



While the license view is open, the Tenant Home AI Assistant works on it. Asked what made the pool go over its quota on 14 September, it answers with the figures of the sub-tab on screen (callout 1) and names the culprit (callout 2), siem-infra-emea, from the day selected in What made the day (callout 3).

Asked to add a cost centre for the cribl_* indexes, it proposes the change through the Concierge (callout 4): one configuration patch with its endpoint and permission (callout 5), applied only on Confirm and run (callout 6), audited, and reloaded by the view.

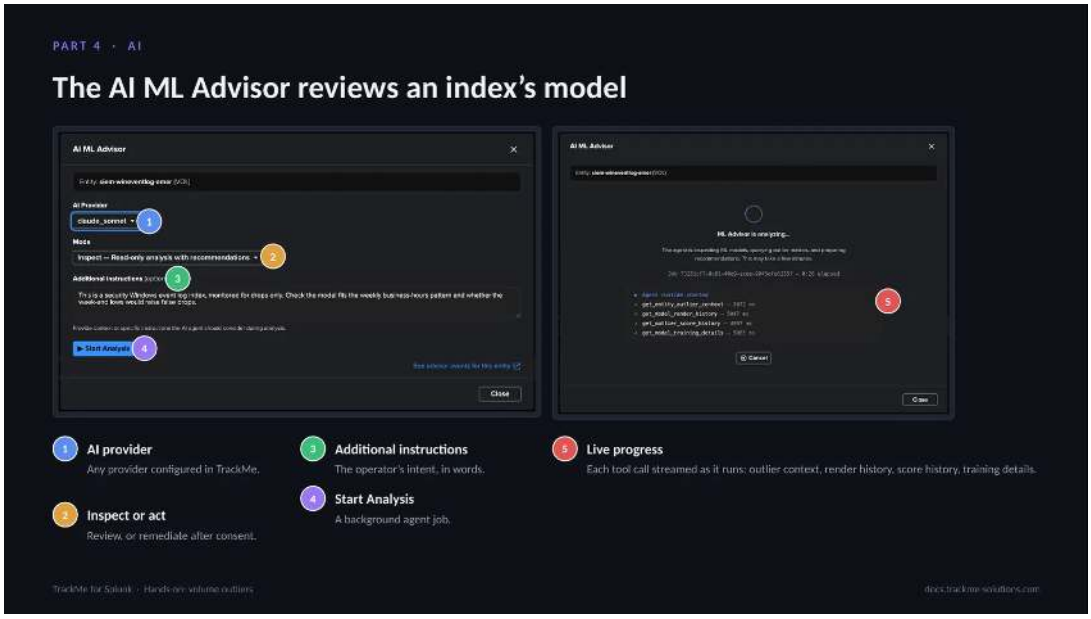
Configuring alerting for red indexes



Volume Outliers alerts like any TrackMe component. The SPLK-VOL (Volume Outliers) alert type (callout 1) generates the notable events and acknowledgments for red indexes; its defaults (callout 2) are any priority, red, all indexes, outliers on. The outliers trigger stays on because the outliers are the state of this component; the same alert covers inactivity.

A TrackMe stateful alert, volume-alert here, sends the opened, updated and closed emails. Emails and Ingest (callout 3) delivers emails plus stateful events; the delivery account (callout 4) is any email account in TrackMe; the environment name (callout 5) is stamped in the email header; the recipients (callout 6) sit further down with the charts and the optional AI status report.

Launching the AI ML Advisor

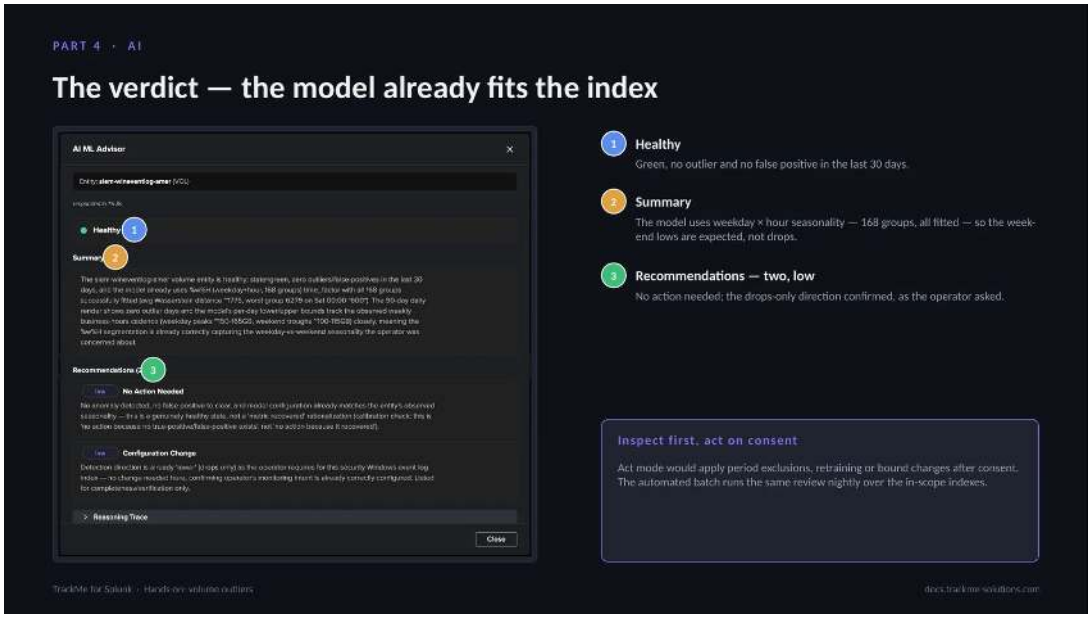


From the Outliers tab of an index, the AI ML Advisor runs as a background agent job with read tools: the entity, its model rules, the outlier and score history, the training details. The AI provider (callout 1) is any provider configured in TrackMe. Inspect or act (callout 2) chooses between a review that changes nothing and a remediation applied after consent.

Additional instructions (callout 3) carry the operator's intent in words: this security index is monitored for drops only;

does the model fit the weekly business-hours pattern, and would the week-end lows raise false drops? Start Analysis (callout 4) launches the job, and the live progress feed (callout 5) streams each tool call as it runs.

Reading the advisor verdict



The verdict is Healthy (callout 1): green, with no outlier and no false positive in the last 30 days. The summary (callout 2) answers with evidence from the model itself: it uses weekday × hour seasonality, 168 groups, all fitted, so the week-end lows are expected rather than drops, and its bounds track the weekday peaks and the week-end troughs. The two recommendations (callout 3) are low: no action needed, the drops-only direction confirmed.

The advisor explains why nothing changes rather than rationalising a recovered metric; the reasoning trace shows every step. Act mode would apply period exclusions, retraining or bound changes after consent. The automated batch runs the same review nightly over the in-scope indexes.

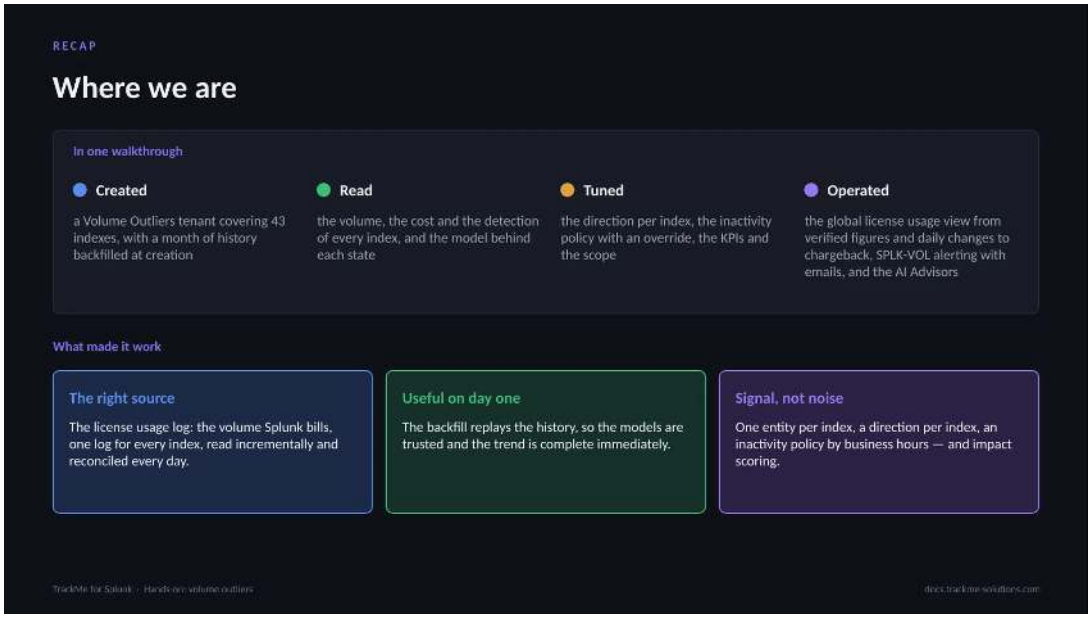
Running cost of the component



Collection is one bounded search per tenant every 5 minutes, reading only the new license usage events, whatever the number of indexes: no per-index and no per-sourcetype scan. Metrics cost about 216 KB per index per day of metric-index licensing; the backfill writes the same series over the chosen depth, and the wizard shows the estimate.

Models are one per index on the rolling 24 hours; the VOL scope and the default KPIs control how many. Live and backfill never overlap, the window waits for a late source and every closed day is reconciled against the license usage log, and a failed search never advances the cursor, so missing volume is never recorded as a drop. The rolling values come from the buckets TrackMe holds, not from its own metrics.

Where we are



One tenant watches the licensed volume of every index. The backfill makes it useful on day one; the table, the entity screen and the model explain every figure, in binary units and verified against the license usage log every day; the direction per index, the inactivity policy with its overrides, the default KPIs and the scope make the detection the operator's own. The Global license usage view, from the verified figures and the daily changes to the scopes, the working days, the days over quota, the costs and the chargeback, SPLK-VOL alerting with stateful emails and the AI Advisors take the component into operations.

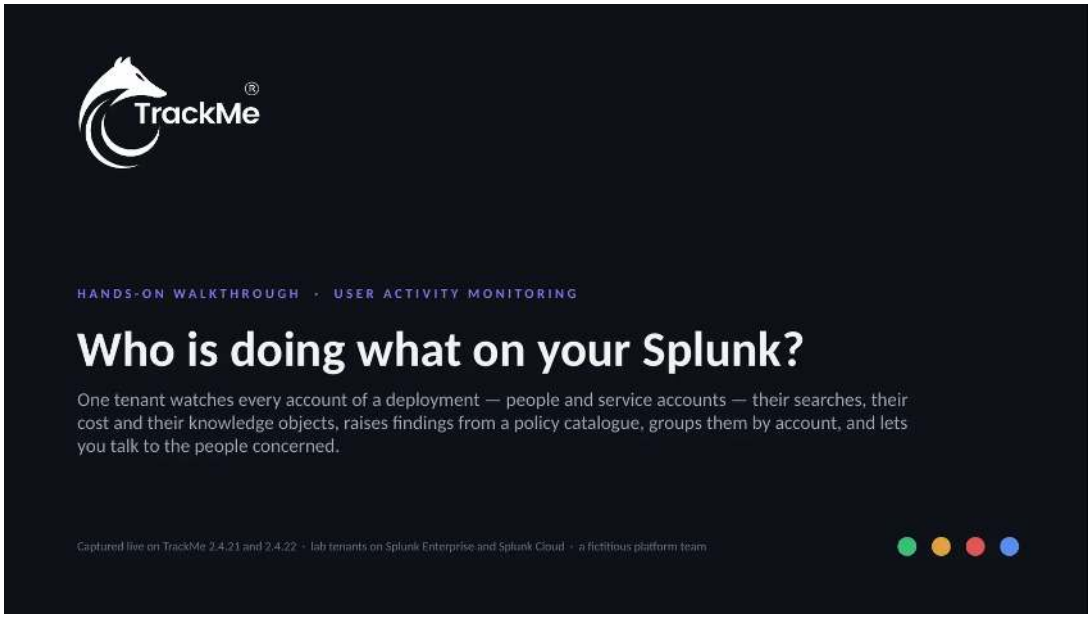
2.1.8 User Activity Monitoring

Who is doing what on your Splunk: one tenant watches every account through the audit trail, the introspection and the REST API catalogs, raises findings from sixteen built-in policies and your custom policies, groups them by account, lets you investigate, assign and email the people concerned, and maps which indexes are read, by whom, and what the unread data costs.

`Download the deck (PDF) · PowerPoint`

about 48 minutes · 48 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

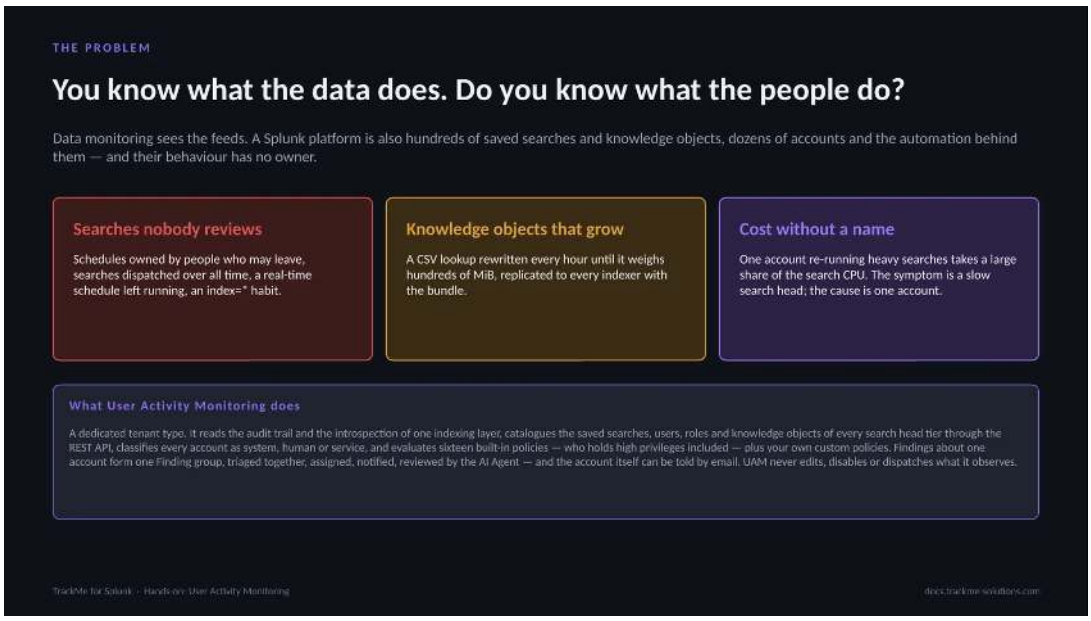
Overview



User Activity Monitoring (UAM) is a dedicated tenant type of TrackMe, generally available since 2.4.19. One tenant watches every account of a deployment: their searches, their cost and the knowledge objects they own. It raises findings from a policy catalogue, groups them by account, and lets operators talk to the people concerned.

Everything shown here was captured live on TrackMe 2.4.21 and 2.4.22: on a tenant watching two search head tiers of one indexing layer, on a second lab tenant for the screens that need a longer history, and on a Splunk Cloud tenant with seven thousand accounts. The accounts that appear, alice.moreau, jane.doe, ben.okafor, svc-siem and others, are fictitious names of a small platform team. Five parts follow: setup, reading and grouping, communicating, tuning and operating, and Data usage.

The other half of the platform



Data monitoring sees the feeds. A Splunk platform is also hundreds of saved searches and knowledge objects, dozens of accounts and the automation behind them, and their behaviour has no owner. Three cards name the symptoms: searches nobody reviews, knowledge objects that grow (a CSV lookup rewritten every hour until it weighs hundreds of MiB) and cost without a name (one account taking a large share of the search CPU).

UAM reads the audit trail and the introspection of one indexing layer, catalogues every search head tier through the

REST API, classifies every account as system, human or service, and evaluates sixteen built-in policies, who holds high privileges included, plus custom policies. It never edits, disables or dispatches what it observes.

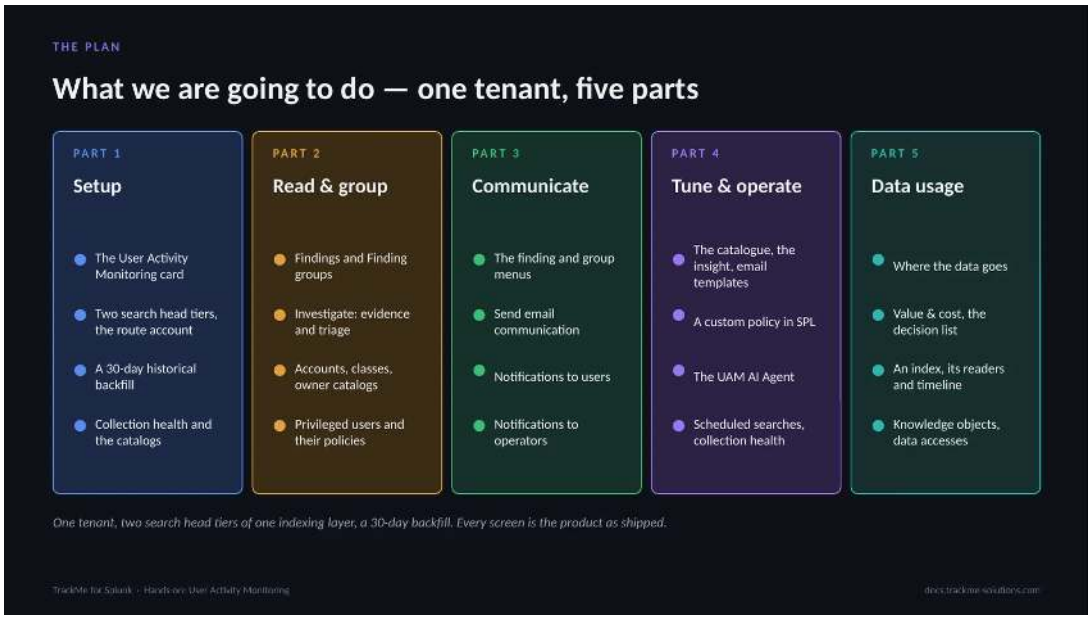
From sources to Finding groups



Five stages make the model. Sources: the audit trail, the introspection, and the REST API of every search head tier. Reports: catalogs per tier, three indexing-layer trackers (Audit activity, Resource usage, Account signals), policies every fifteen minutes and a bounded backfill. Evidence: compact trackme:uam:* records in the tenant indexes. Classes: system, human or service. Finding groups: the sixteen built-in and the custom policies raise findings, and an account with its saved searches forms one Finding group.

On top of the findings sit the triage, assignment, notifications, emails to the account concerned, findings actions and the UAM AI Agent. An account is the one Splunk reports in its audit trail, not a verified person, and collection never drops data: what a run cannot do carries over.

Five parts, one tenant



The walkthrough has five parts. Setup creates the tenant from the User Activity Monitoring card, with two search head tiers, the route account and a 30-day backfill, then checks the collection health. Read and group covers the

findings and Finding groups, the investigation, the accounts with their classes and owner catalogs, and the privileged users.

Communicate covers the menus, the email communication and the notifications. Tune and operate covers the policy catalogue, a custom policy in SPL, the UAM AI Agent and the scheduled searches. Data usage shows where the data goes, value and cost, an index in depth, the knowledge objects and an account's data accesses. Every screen is the product as shipped.

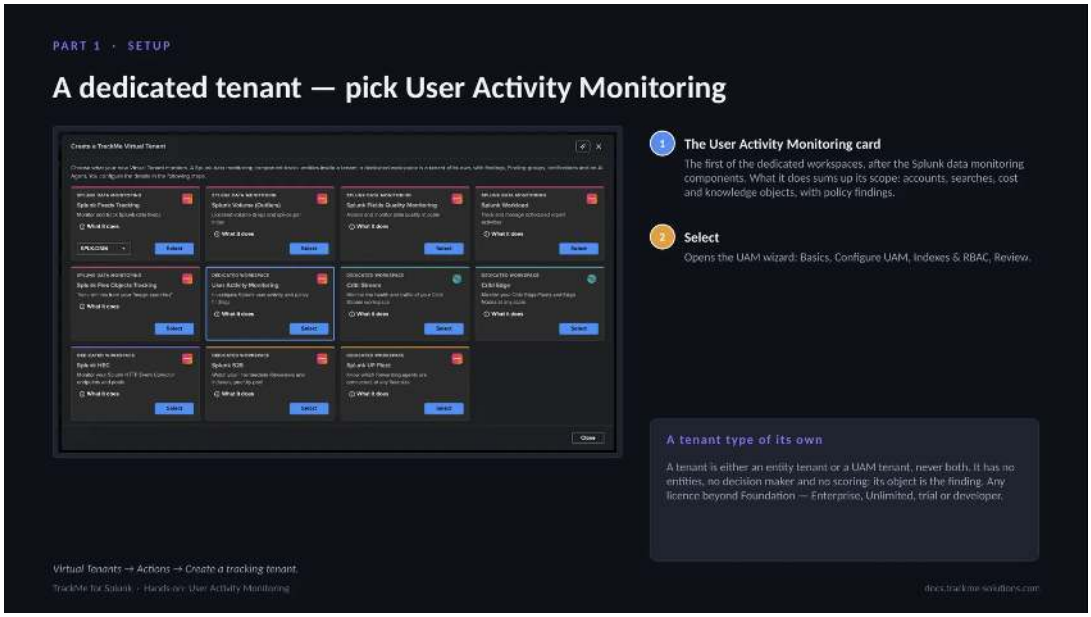
Part 1 — Setup



The first part creates the tenant and follows its first collection: the User Activity Monitoring card, the search head tiers and the route account, the historical backfill, then the tenant card, the status strip and the collection health on day one.

The wizard has four steps. Within minutes of the creation the catalogs of every tier are published and the first activity records arrive, and Collection health shows every report at work.

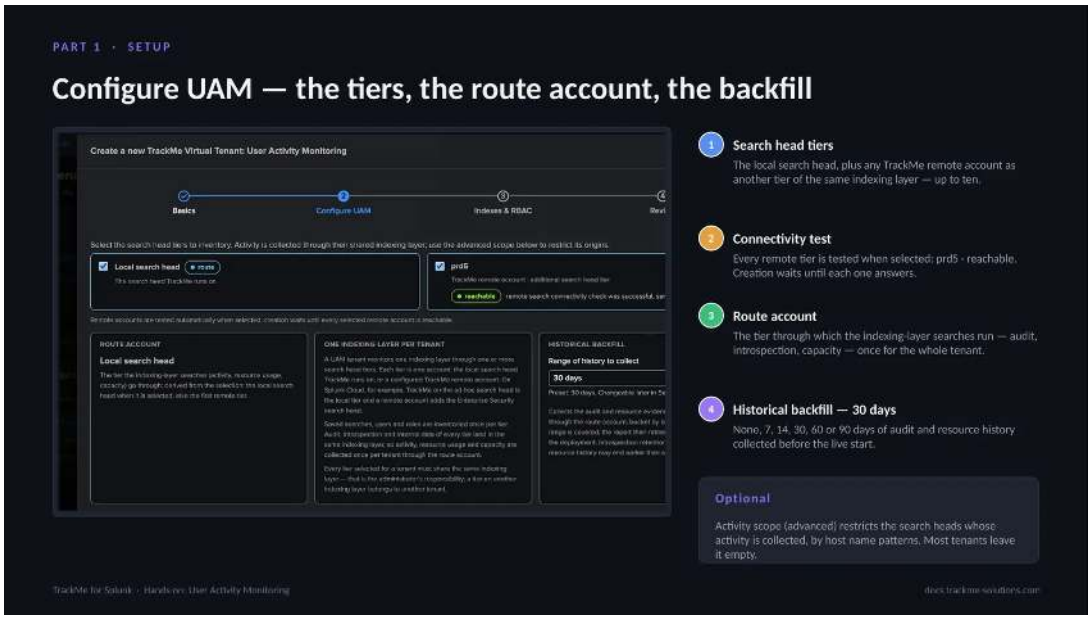
Choosing the tenant type



Virtual Tenants, Actions, Create a tracking tenant opens the chooser. The User Activity Monitoring card is the first of the dedicated workspaces, after the Splunk data monitoring components; What it does sums up its scope: the accounts, searches, cost and knowledge objects of one indexing layer, with policy findings grouped by account. Select opens the UAM wizard: Basics, Configure UAM, Indexes & RBAC, Review.

A tenant is either an entity tenant or a UAM tenant, never both. A UAM tenant has no entities, no decision maker and no scoring: its object is the finding. It requires any licence beyond Foundation: Enterprise, Unlimited, trial or developer.

Tiers, route account and backfill



Configure UAM shapes the tenant. Search head tiers lists the local search head plus any TrackMe remote account as another tier of the same indexing layer, up to ten. Every remote tier is tested when selected (here prd5, reachable), and creation waits until each one answers. The route account is the tier through which the indexing-layer searches run (audit, introspection, capacity), once for the whole tenant.

The historical backfill collects none, 7, 14, 30, 60 or 90 days of audit and resource history before the live start; 30 days are chosen here, so that classification and the first findings are meaningful on day one. The optional activity scope restricts the collected search heads by host name patterns.

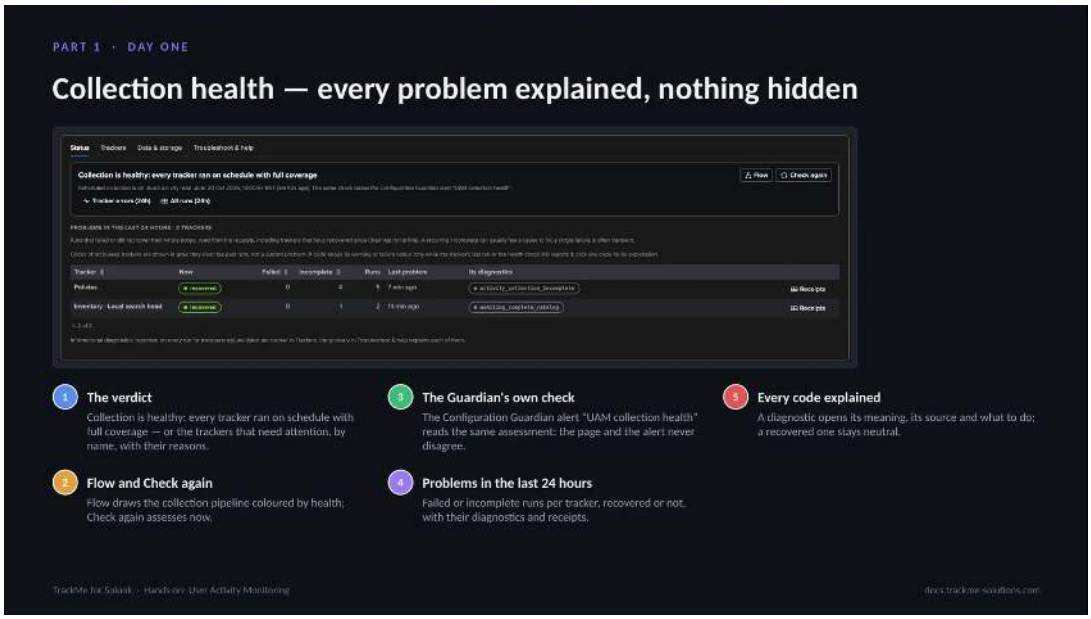
Status at a glance



On the Virtual Tenants page, the tenant card shows the open Finding groups and findings (10 groups, 36 findings), the high or critical ones, and the collection status, here the backfill at 16 %. Selecting the card opens the workspace; a double-click opens a read-only overview.

Inside the workspace, the status strip summarises the collection: the tiers with the route marked, the age of the inventory, Audit activity, Resource usage, Account signals and policies runs, the checkpoint lag and the coverage. Refresh and Collect now sit on the right; Collect now runs every report on demand.

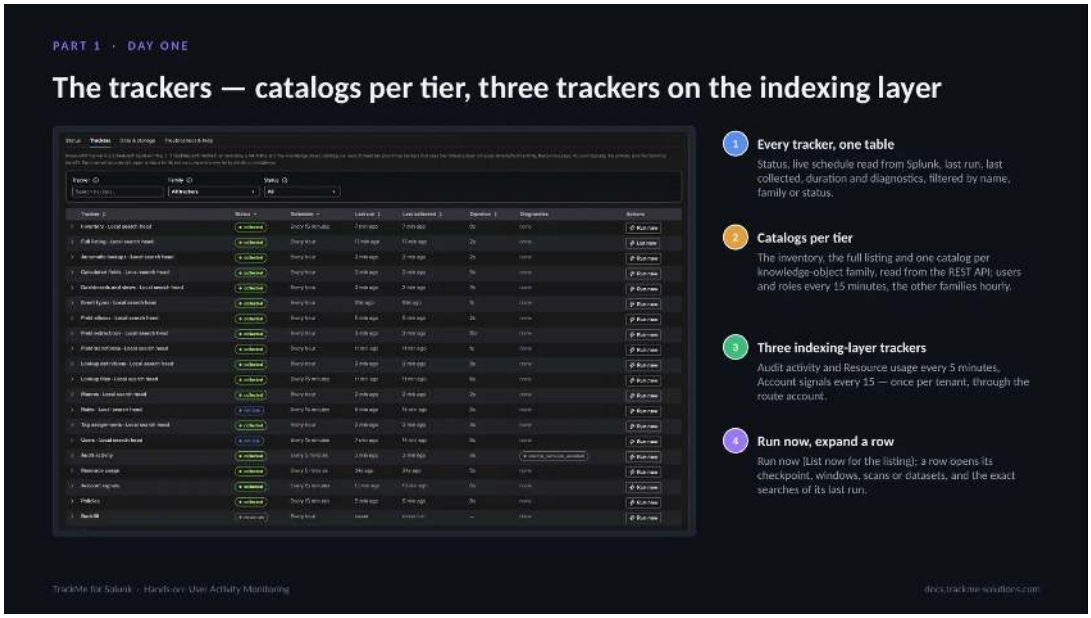
Day one in Collection health



Collection health answers one question: is UAM collecting what it should, and if not, why and what to do. The Status view gives the verdict, here Collection is healthy: every tracker ran on schedule with full coverage, or names the trackers that need attention with their reasons. Flow and Check again sit beside it. The Configuration Guardian alert UAM collection health reads the same assessment, so the page and the alert never disagree.

Problems in the last 24 hours lists the failed or incomplete runs of each tracker, recovered or not, with their diagnostics and receipts. Every diagnostic opens its meaning, its source and what to do; a code that only describes a recovered run stays neutral. The other views are Trackers, Data & storage and Troubleshoot & help.

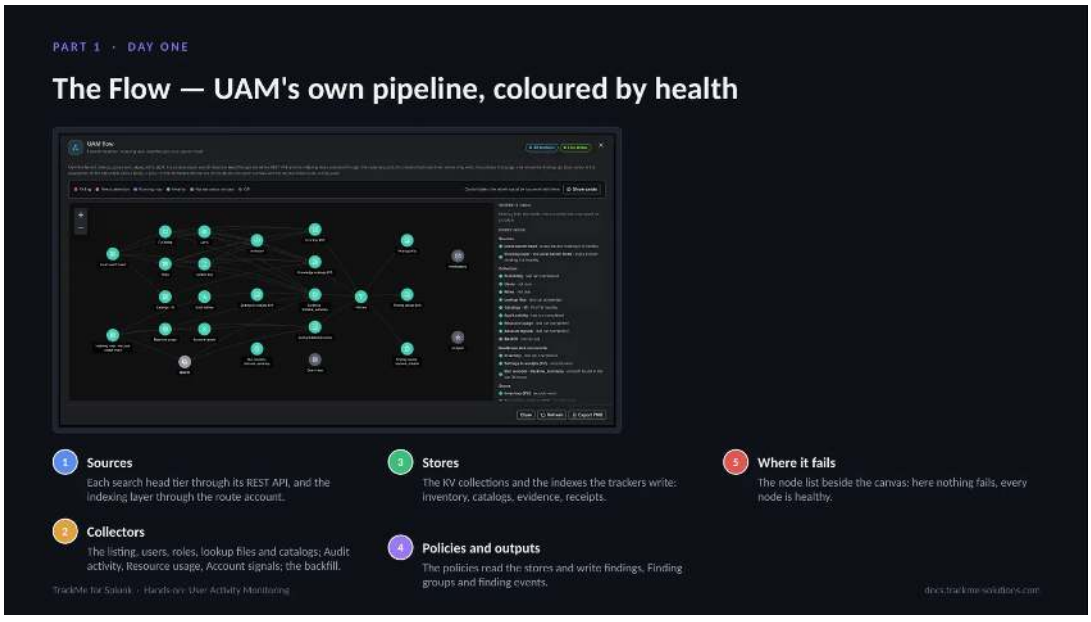
Every tracker in one table



The Trackers view lists every UAM report with its status, its live schedule read from Splunk, last run, last collected, duration and diagnostics, filtered by name, family or status. Per search head tier come the inventory, the full listing and one catalog per knowledge-object family, read from the REST API: users and roles every 15 minutes, the other families hourly, each published on its own so one slow family never blocks the others.

Once per tenant, through the route account, three trackers read the indexing layer: Audit activity and Resource usage every 5 minutes, Account signals every 15. Run now dispatches a tracker (List now for the full listing), and a row expands to its checkpoint, windows, scans or datasets and the exact searches of its last run.

The collection pipeline

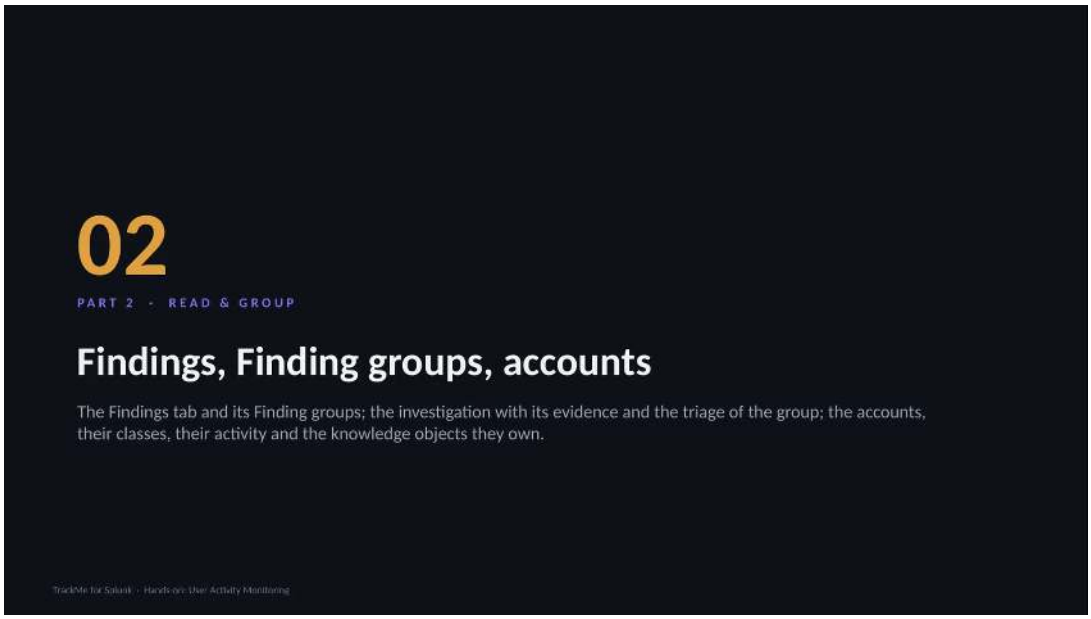


Flow, opened from the Status view, draws how the tenant collects, stores and judges, left to right: the sources (each search head tier through its REST API, and the indexing layer through the route account), the collectors, the stores they write (the KV collections and the indexes) and the policies with the findings, Finding groups and finding events they produce.

Each node is coloured by the same assessment as the Status view: failing, needs attention, running now, healthy, not run yet or off. The list beside the canvas names where it fails; here nothing fails and every node is healthy. The view

exports as a PNG.

Part 2 — Read and group



The second part is about reading what the tenant found: the Findings tab and its Finding groups, the investigation with its evidence and the triage of the group, then the accounts, their classes, their activity and the knowledge objects they own.

Findings about the same account and the saved searches it owns form one Finding group, decided on together.

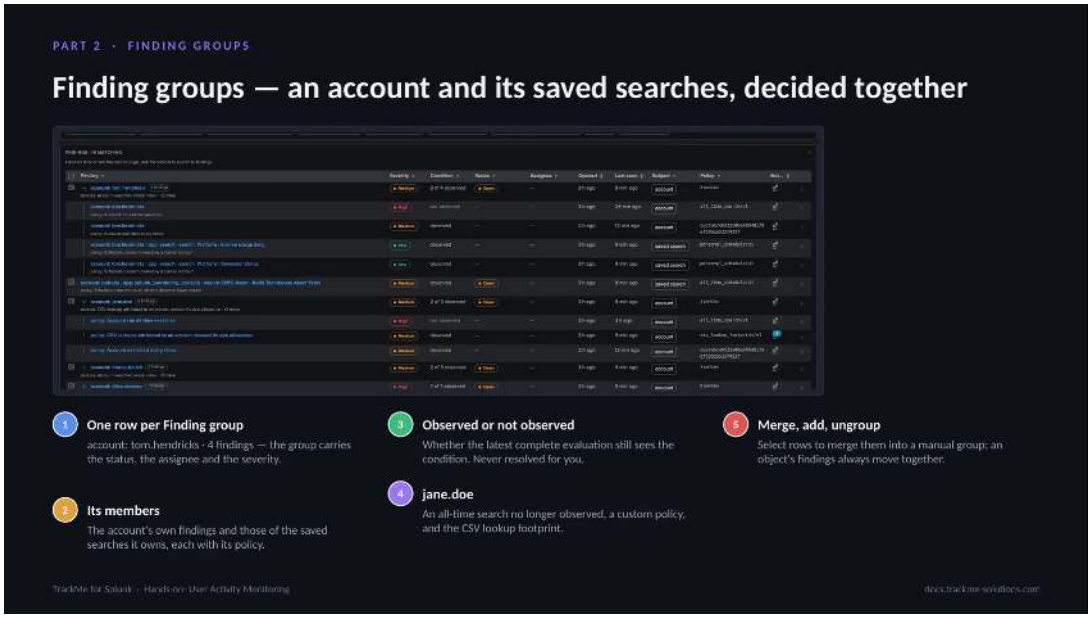
The findings backlog



The Findings tab is the backlog. The tiles count Finding groups: here 9 open groups holding 12 findings, 7 of them high or critical, with the groups new in the range, assigned to the current user, resolved and dismissed. Finding transitions charts the reobserved, unobserved and updated findings over the last 24 hours.

By severity shows where the weight is across low, medium and high. By policy shows which policies raised what: accounts that ran all-time searches, scheduled searches owned by a human account and repeated resource-intensive searches. Below the charts come the filters and one row per Finding group.

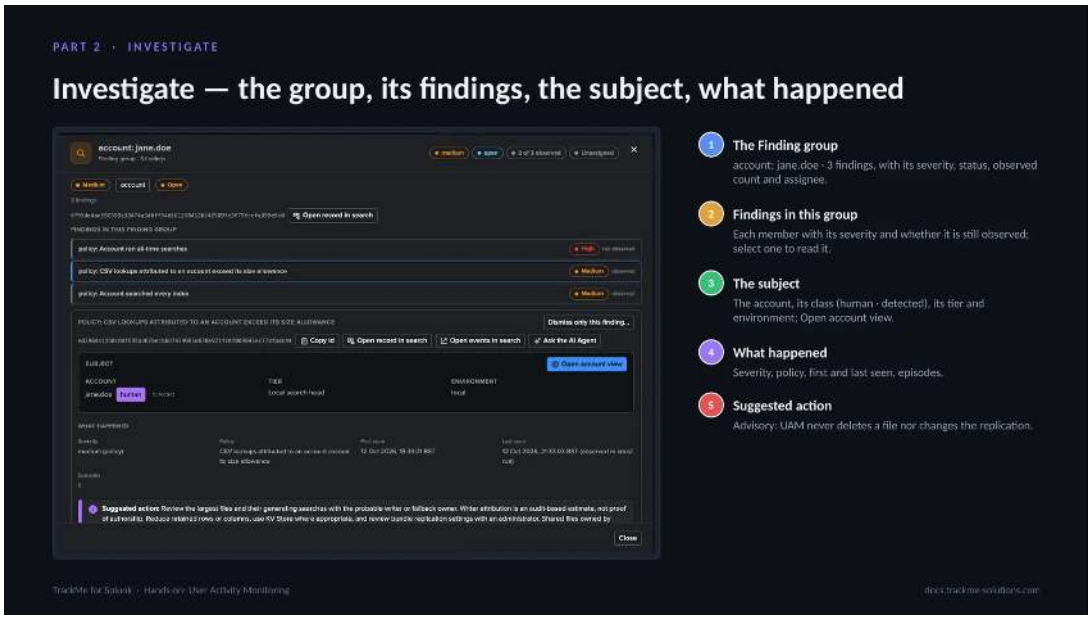
One decision per account



All findings about the same account and the saved searches it owns automatically form one Finding group. Each row is a group, for example account: tom.hendricks with 4 findings; the group carries the status, the assignee and the severity, and expands to its members, each with its policy and its own evidence.

The condition column says whether the latest complete evaluation still observes the condition; a finding that is no longer observed stays listed until a person resolves it. The jane.doe group combines an all-time search no longer observed, a custom policy and the CSV lookup footprint. Selecting rows merges them into a manual group; an object's findings always move together.

The investigation view

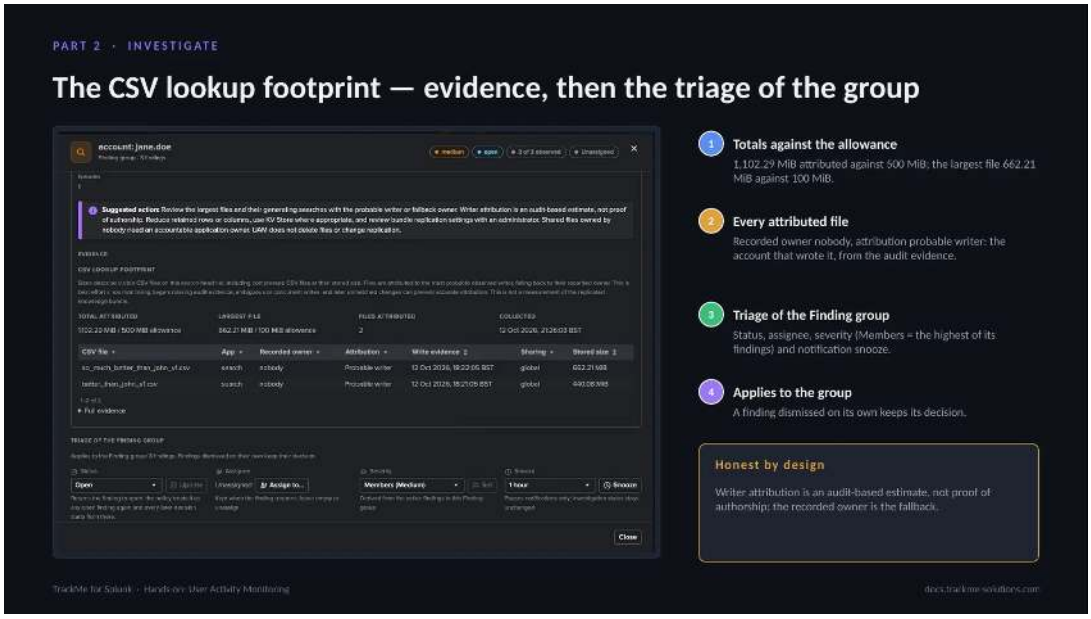


Investigate opens the whole case. The header names the Finding group, account: jane.doe with 3 findings, with its severity, status, observed count and assignee. Findings in this group lists each member with its severity and whether it is still observed; selecting one shows it below.

For the CSV lookup footprint finding, the subject section shows the account, its class (human, detected), its tier and environment, with Open account view. What happened gives the severity, the policy, the first and last seen times and

the episodes. The suggested action is advisory: UAM never deletes a file nor changes the replication. Copy id, Open record in search, Open events in search and Ask the AI Agent complete the view.

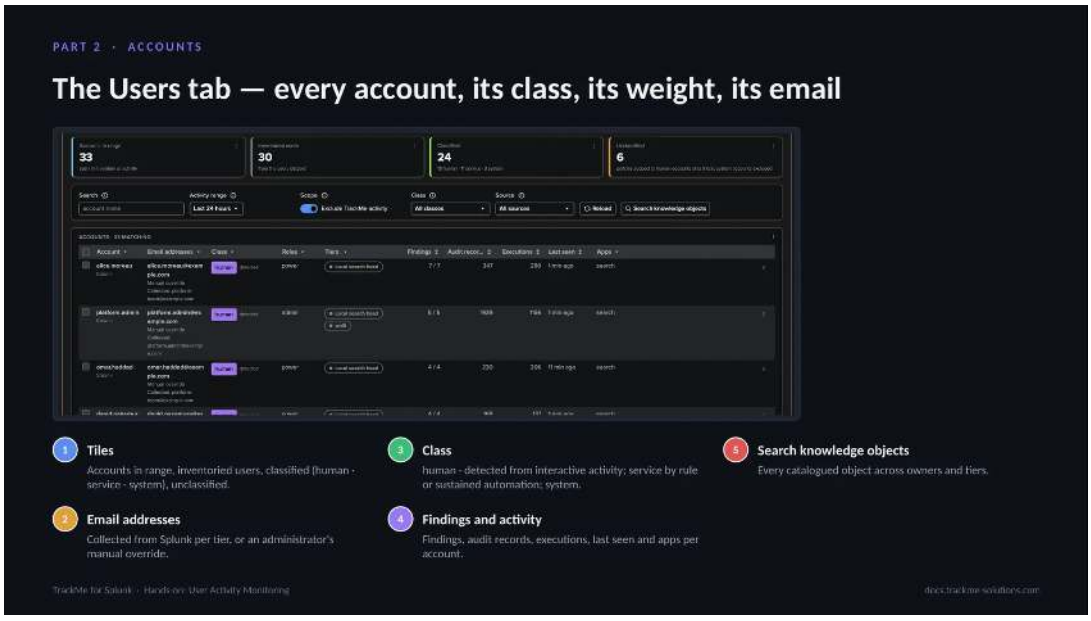
Evidence and triage of the group



The CSV lookup footprint policy came with 2.4.19. The evidence totals 1,102.29 MiB of CSV lookups attributed to jane.doe against a 500 MiB allowance, with a largest file of 662.21 MiB against 100 MiB. Every attributed file is listed: the recorded owner is nobody, and the attribution is probable writer, the account that wrote the file according to the audit evidence. Writer attribution is an audit-based estimate, not proof of authorship; the recorded owner is the fallback.

Below the evidence, the triage applies to the whole Finding group: status, assignee, severity (Members means the highest of its findings) and notification snooze. A finding dismissed on its own keeps its decision.

Every account in one table

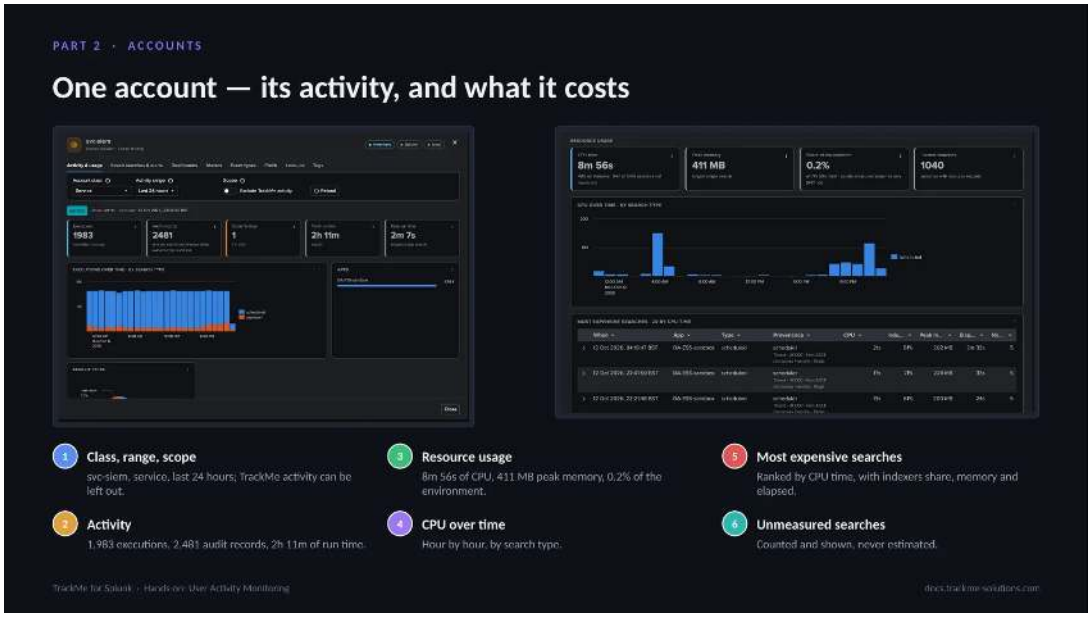


The Users tab is the account-centric view. The tiles count the accounts in range, the inventoried users, the classified accounts (human, service, system) and the unclassified ones. Each row shows the email address, collected from

Splunk per tier or set by an administrator as a manual override; that address is where notifications to users and email communications are sent.

The class column shows human (detected from interactive activity), service (by rule or sustained automation) or system. Findings, audit records, executions, last seen and apps complete the row. Search knowledge objects opens a search over every catalogued object across owners and tiers.

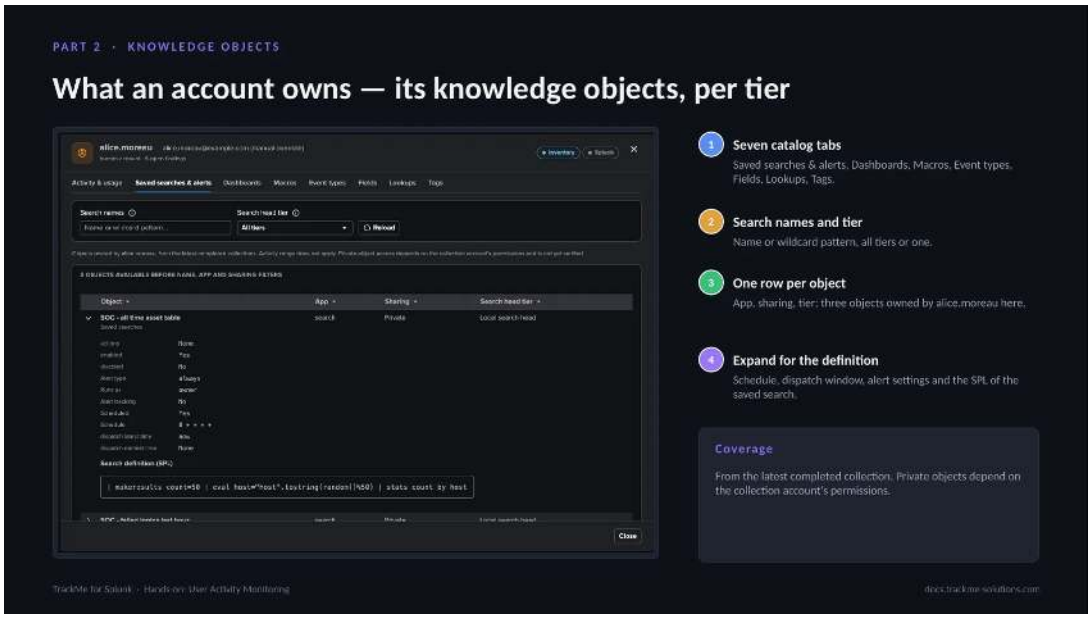
Activity and cost of one account



The account view gathers everything about one account, here svc-siem, a service account, over the last 24 hours. The top shows its class, the activity range and the scope, then the tiles: 1,983 executions, 2,481 audit records, one open finding and 2h 11m of run time, with the executions over time by search type and the app it works in.

Further down, the resource usage tells what the account costs: 8m 56s of CPU time, 411 MB peak memory and 0.2% of the environment, with CPU over time and the most expensive searches ranked by CPU time. Searches the resource records could not measure are counted and shown, never estimated. Seven catalog tabs list what the account owns.

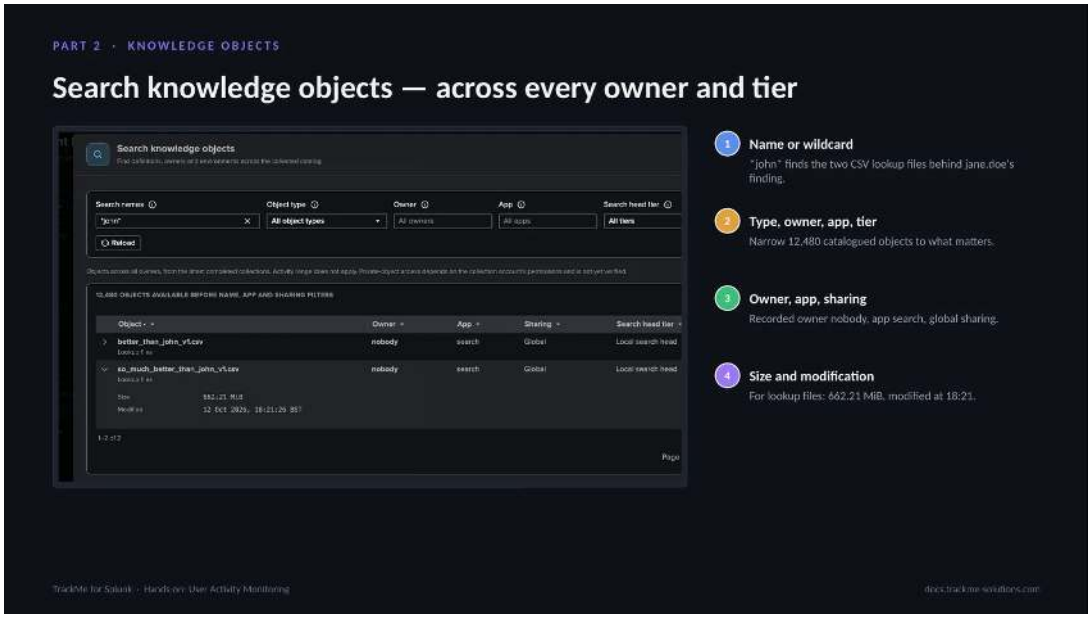
What an account owns



The seven catalog tabs of the account view (Saved searches & alerts, Dashboards, Macros, Event types, Fields, Lookups, Tags) answer the question every offboarding asks: what does this account own? A name or wildcard pattern and a search head tier narrow the list; here three objects are owned by alice.moreau, each with its app, sharing and tier.

Expanding a saved search shows its definition: schedule, dispatch window, alert settings and SPL. The data comes from the latest completed catalog collection, not from a live call, and private objects depend on the permissions of the collection account.

Finding objects across owners



Search knowledge objects runs one search across the catalogued objects of every tier and owner. The name *john* finds the two CSV lookup files behind jane.doe's footprint finding; the object type, owner, app and search head tier narrow the 12,480 catalogued objects.

Each row shows the owner, app and sharing, here recorded owner nobody, app search, global sharing. For lookup files, the expanded row adds the size and the last modification: 662.21 MiB, modified at 18:21. The files are owned by nobody and were written by jane.doe, which is exactly the case the writer attribution resolves.

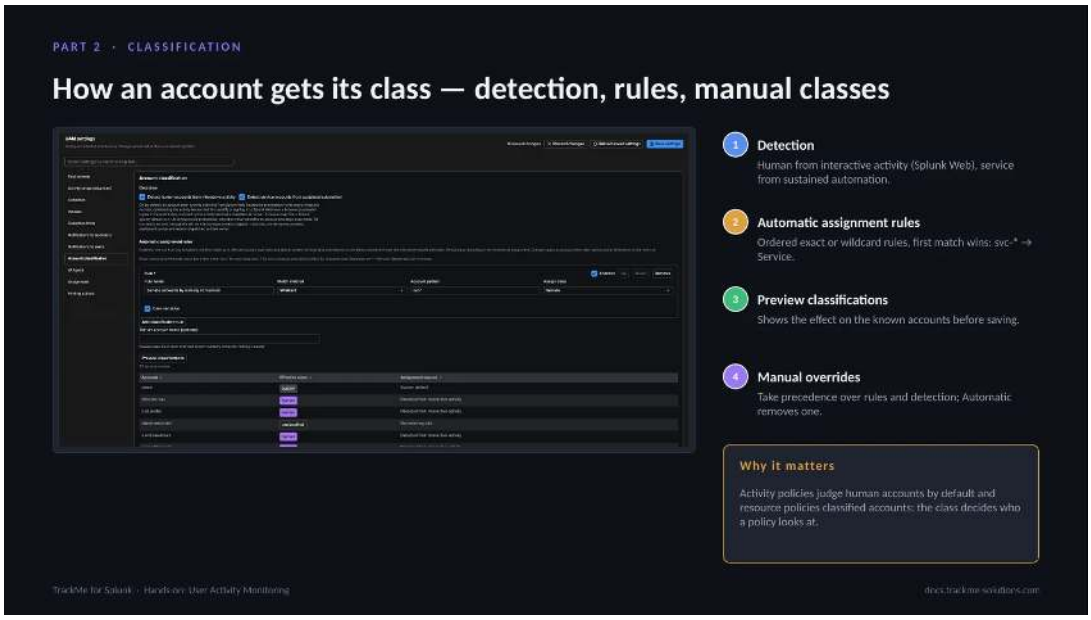
The raw audit evidence



The Activity tab is the raw evidence: one record per search and window rather than one line per audit event. The time range and scope sit on top; Exclude TrackMe activity leaves out the searches run in the trackme app. Evidence switches between the audit records and the executions correlated per search id, and the account and app filters narrow the view, here to alice.moreau.

The chart shows the records over time by account source, with history from the backfill. The table lists the 500 most recent records with search id, outcome, run time and evidence state; Inspect opens a record. This is the material every activity policy is evaluated on.

Account classification

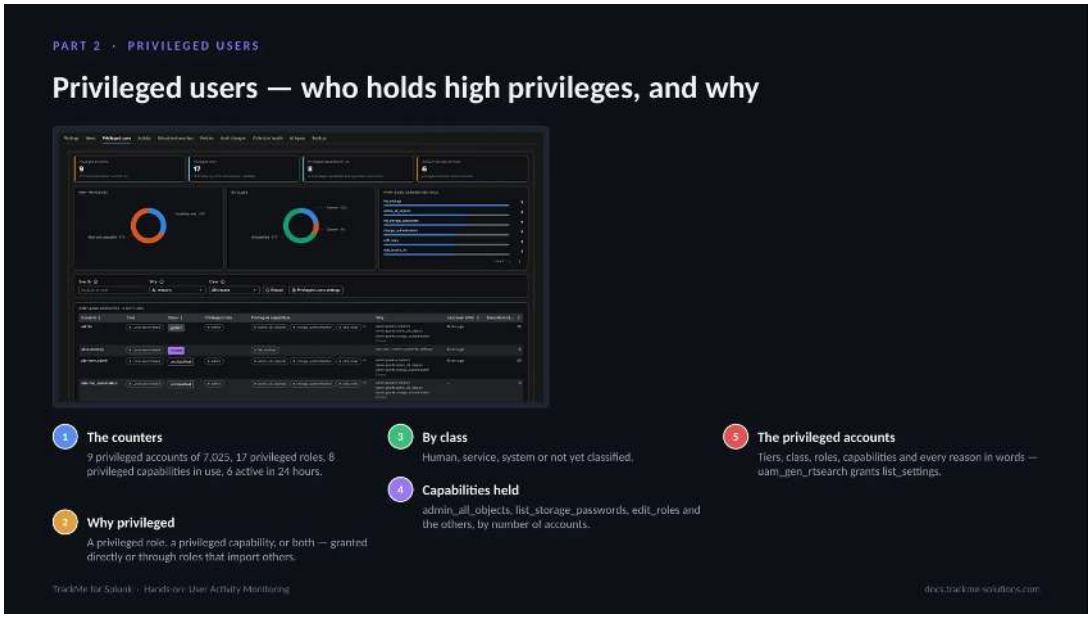


Settings, Account classification decides who each policy looks at: activity policies judge human accounts by default, resource policies classified accounts. Detection classifies an account as human from interactive activity in Splunk Web and as service from sustained automation.

Automatic assignment rules are ordered exact or wildcard rules where the first match wins; the rule svc-* to Service classifies every service account that will ever be created. Preview classifications shows the effect on the known

accounts before saving. Manual overrides take precedence over rules and detection, and Automatic removes one. Precedence is always manual class, Splunk system defaults, the first matching rule, then detection.

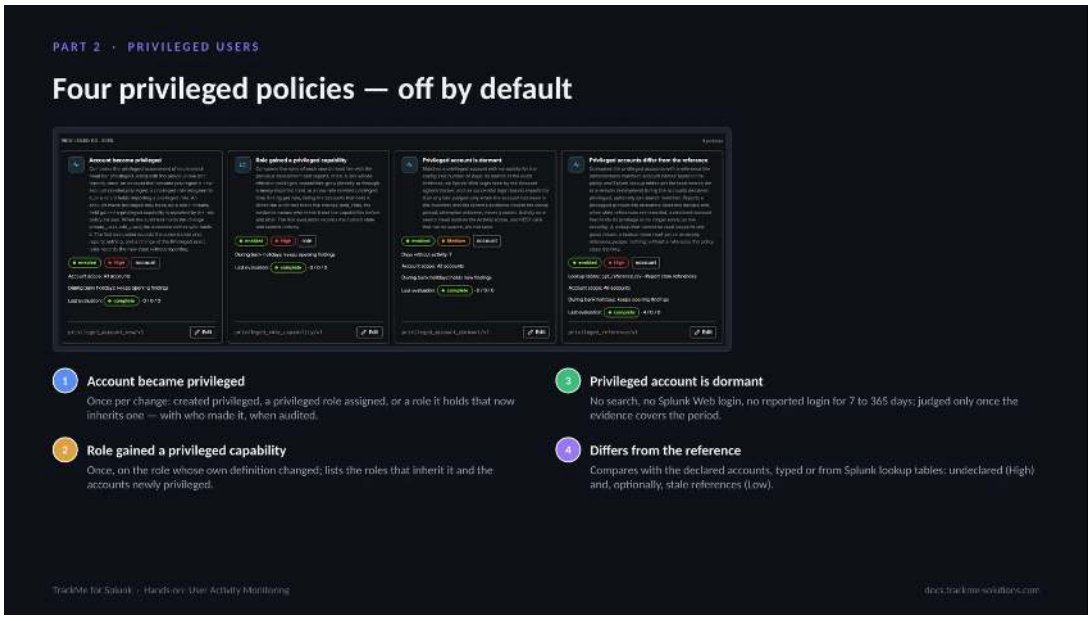
Who holds high privileges



The Privileged users tab reads the users and roles catalogs UAM already collects and makes no call to Splunk. The counters show 9 privileged accounts of 7,025, 17 privileged roles, 8 privileged capabilities in use and 6 privileged accounts active in the last 24 hours. Charts show why accounts are privileged (a role, a capability or both), their classes and the accounts holding each privileged capability.

An account is privileged when one of its roles, or a role that role imports however indirectly, is a privileged role or grants a privileged capability. The table lists every privileged account with its tiers, class, roles, capabilities and each reason in words, for example uam_gen_rtsearch grants list_settings.

Four policies on privileges

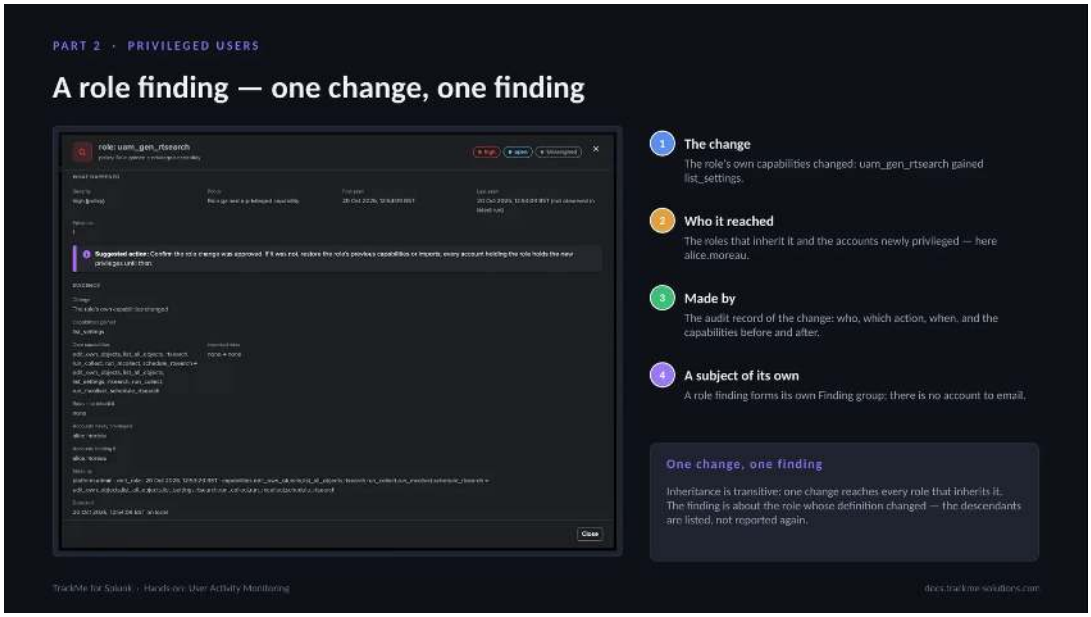


Four privileged policies build on that assessment, all off by default. Account became privileged reports once per change an account created privileged, given a privileged role, or holding a role that now inherits one, with who made

the change when Splunk audited it. Role gained a privileged capability reports once, on the role whose own definition changed, the roles that inherit it and the accounts newly privileged.

Privileged account is dormant reports a privileged account with no search, no Splunk Web login and no reported login for 7 to 365 days, once the evidence covers the period. Privileged accounts differ from the reference compares the accounts with the declared privileged accounts, typed names and Splunk lookup tables: undeclared at high severity and, optionally, stale references at low severity.

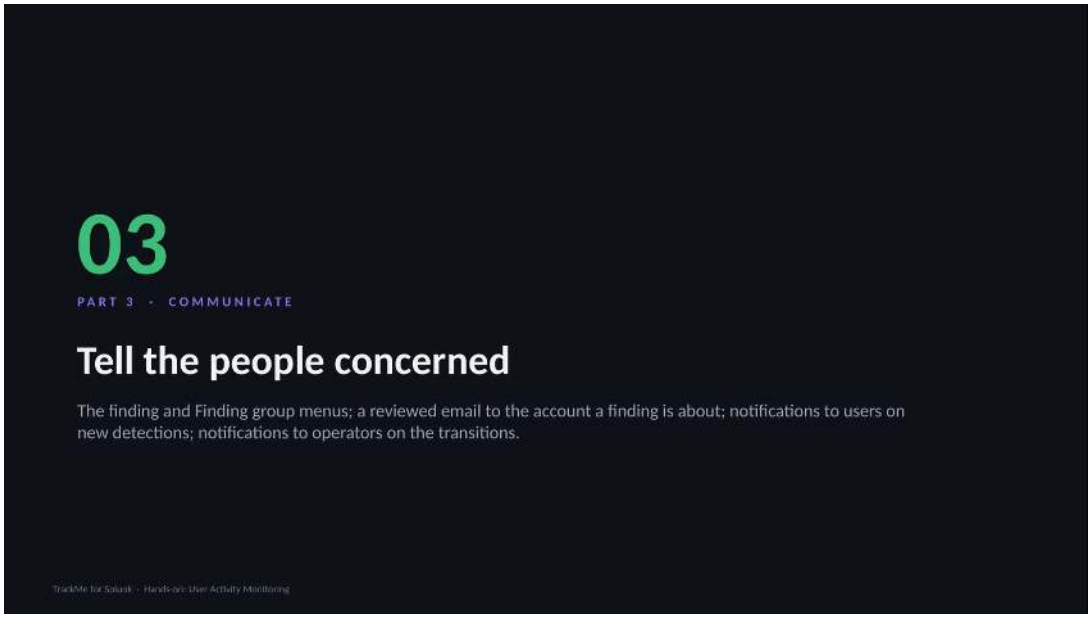
One change, one finding



A role finding opened in Investigate. The role's own capabilities changed: uam_gen_rtsearch gained list_settings, a privileged capability. The finding lists the roles that inherit the change and the accounts newly privileged through it, here alice.moreau, and the audit record of who made the change, the action, when, and the capabilities before and after.

Inheritance is transitive: one change reaches every role that inherits it, and the finding is about the role whose definition changed, its descendants listed rather than reported again. A role finding forms its own Finding group, and there is no account to email.

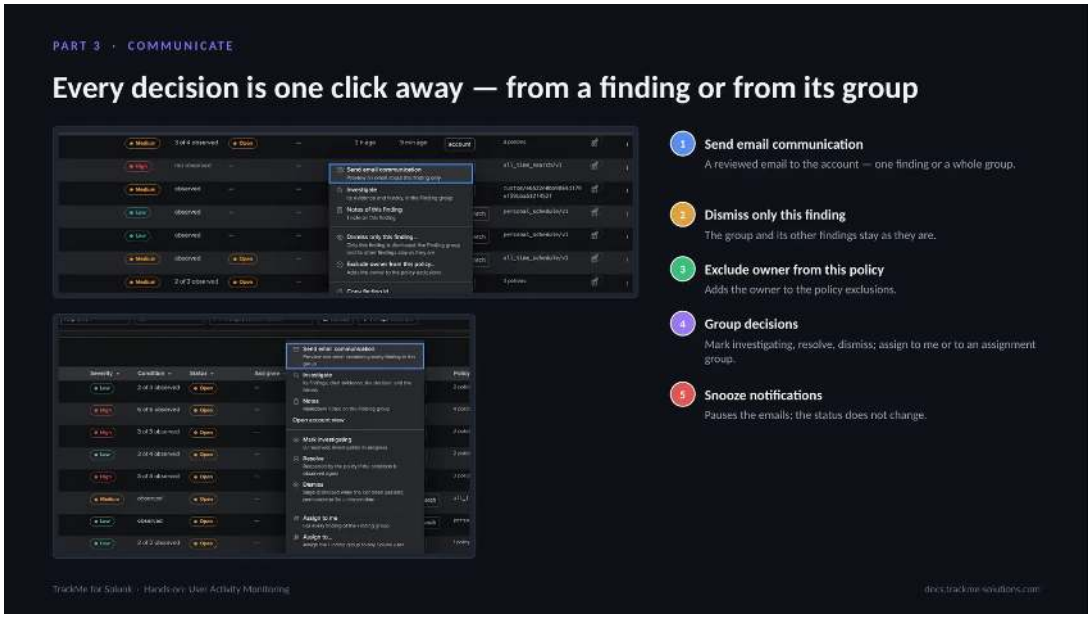
Part 3 — Communicate



The third part is about talking to the people behind the accounts. It covers the finding and Finding group menus, a reviewed email to the account a finding is about, notifications to users on new detections, and notifications to operators on the transitions of the findings.

A one-off communication is reviewed before it leaves; automatic notifications to users work independently of the operators' notifications.

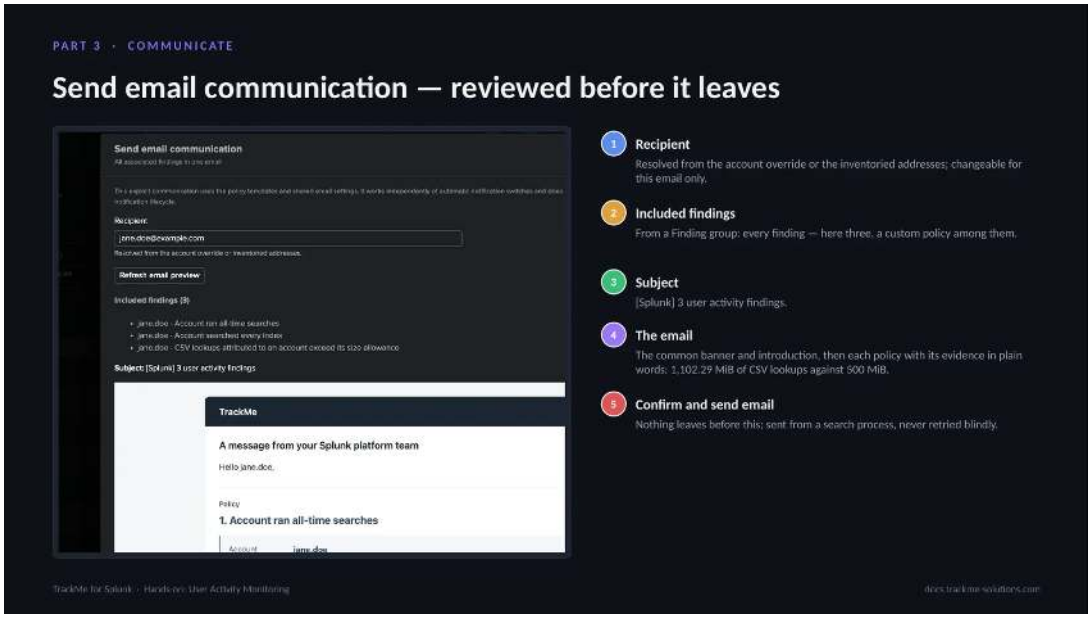
The finding and group menus



Two menus carry the decisions. The menu of a single finding offers Send email communication, Investigate, Notes, Dismiss only this finding (the group and its other findings stay as they are), Exclude owner from this policy (which adds the owner to the policy exclusions), Copy finding id, and Open record and events in search.

The Finding group menu offers Send email communication for every finding of the group, Investigate, Notes and Open account view, then the group decisions: mark investigating, resolve, dismiss, assign to me or to an assignment group. Snooze notifications pauses the emails without changing the status.

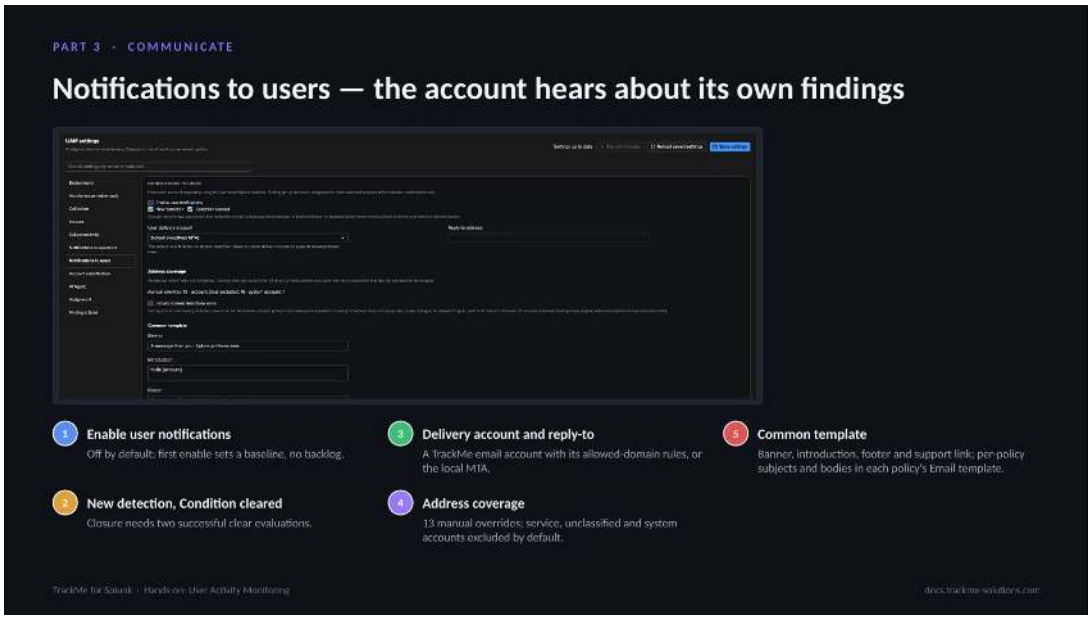
A reviewed email to the account



Send email communication prepares a private preview of one email for the account concerned. The recipient is resolved from the account override or the inventoried addresses and can be changed for this email only. From a Finding group, every finding is included, here three, a custom policy among them, under the subject [Splunk] 3 user activity findings.

The email carries the common banner and introduction, then each policy with its evidence in plain words, such as 1,102.29 MiB of CSV lookups against 500 MiB. Nothing leaves before Confirm and send email; the email is sent from a search process and never retried blindly. It changes neither the findings nor their notification lifecycle.

Automatic emails to account owners

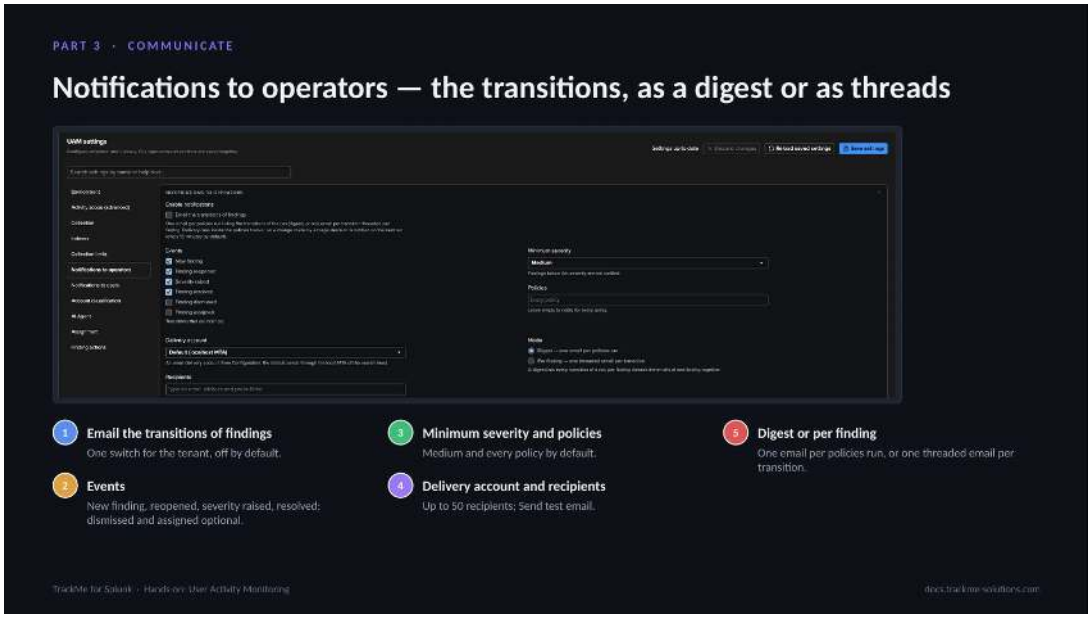


Notifications to users email each account about its own findings at its inventoried address. Enable user notifications is off by default, and the first enable sets a baseline so that no backlog is sent. Events are New detection and Condition cleared; a closure needs two successful clear evaluations.

A delivery account (a TrackMe email account with its allowed-domain rules, or the local MTA) and a reply-to address send them. Address coverage shows 13 manual overrides; service, unclassified and system accounts are excluded by

default. The common template holds the banner, introduction, footer and support link, while each policy has its own subject and body. Limits protect the mailboxes: at most 50 per run and 3 per user per hour.

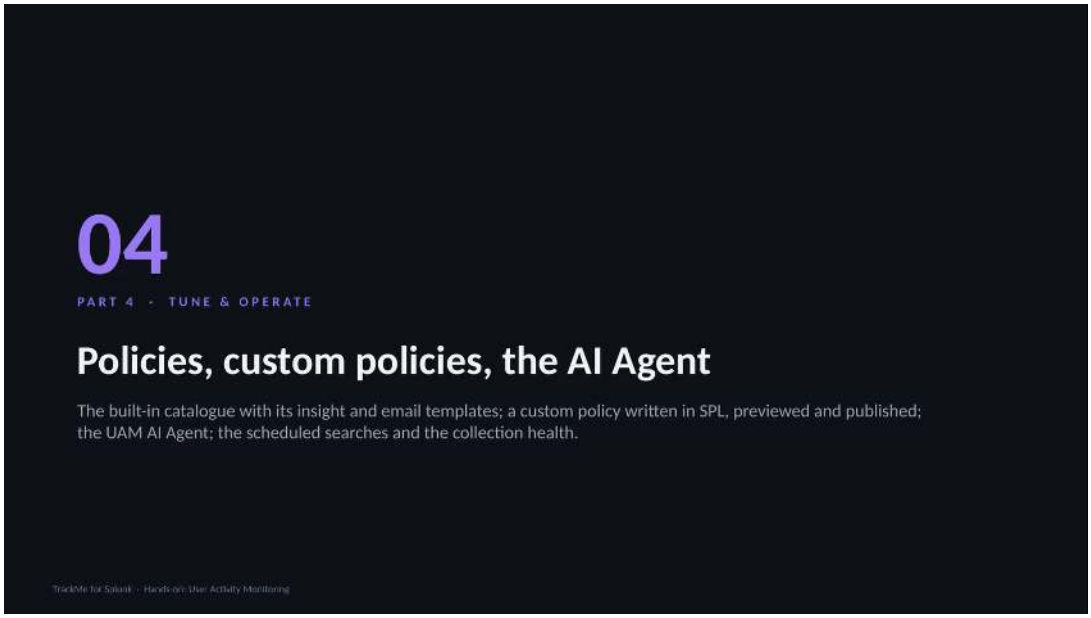
Emails to the operators



Notifications to operators email the transitions of the findings, behind one switch for the tenant that is off by default. The events are new finding, reopened, severity raised and resolved, with dismissed and assigned optional. A minimum severity (medium by default) and a policy list (every policy by default) narrow them.

A delivery account sends them to up to 50 recipients, and Send test email checks the delivery. The mode is either a digest, one email per policies run, or one threaded email per finding transition. Emails link straight to the finding in its investigation.

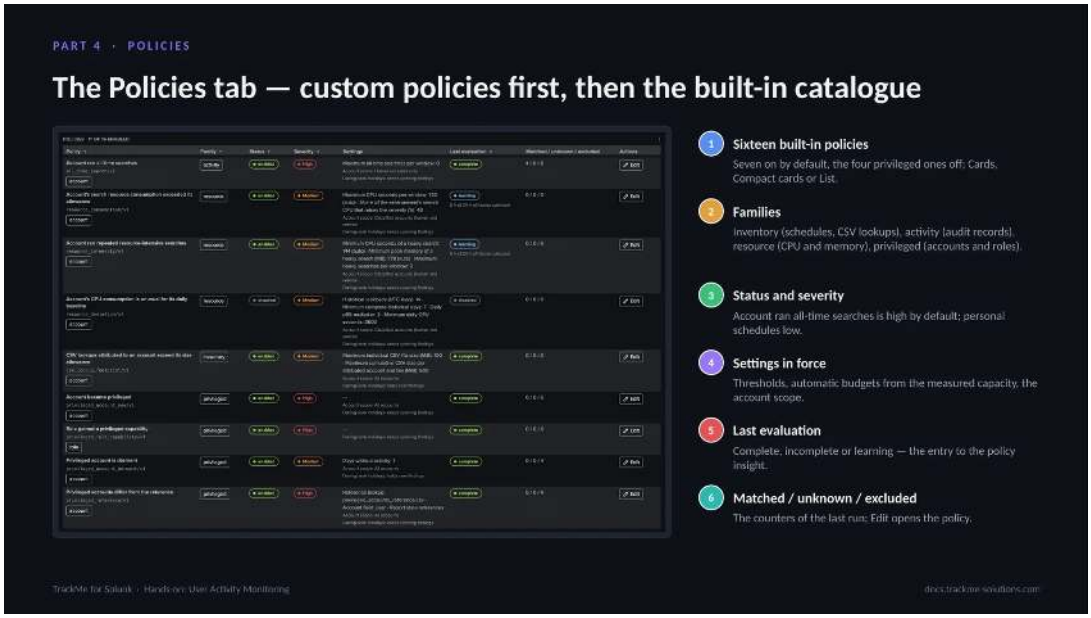
Part 4 — Tune and operate



The fourth part covers tuning and operating: the built-in policy catalogue with its insight and email templates, a custom policy written in SPL, previewed and published, the UAM AI Agent, the scheduled searches and the collection health.

The catalogue ships with defaults that fit a busy platform; a rule that is not in it can be written as a custom policy.

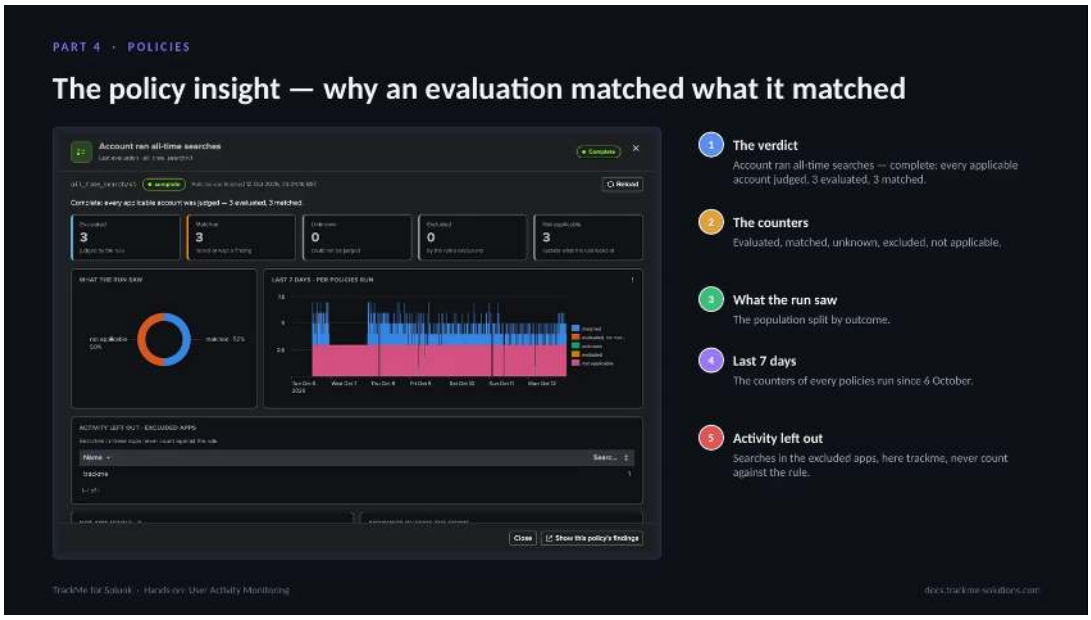
The policy catalogue



The Policies tab lists the custom policies of the tenant first, then the sixteen built-in policies, seven of them on by default and the four privileged ones off, as cards, compact cards or a list. The families are inventory (schedules and CSV lookups), activity (audit records), resource (CPU and memory) and privileged (accounts and roles).

Each row shows the status and severity (Account ran all-time searches is high by default, personal schedules low), the settings in force (thresholds, automatic budgets from the measured capacity, the account scope), the last evaluation (complete, incomplete or learning) and the matched, unknown and excluded counters; Edit opens the policy. Resource policies learn for 24 hours before they judge anyone, and a policy never resolves a finding.

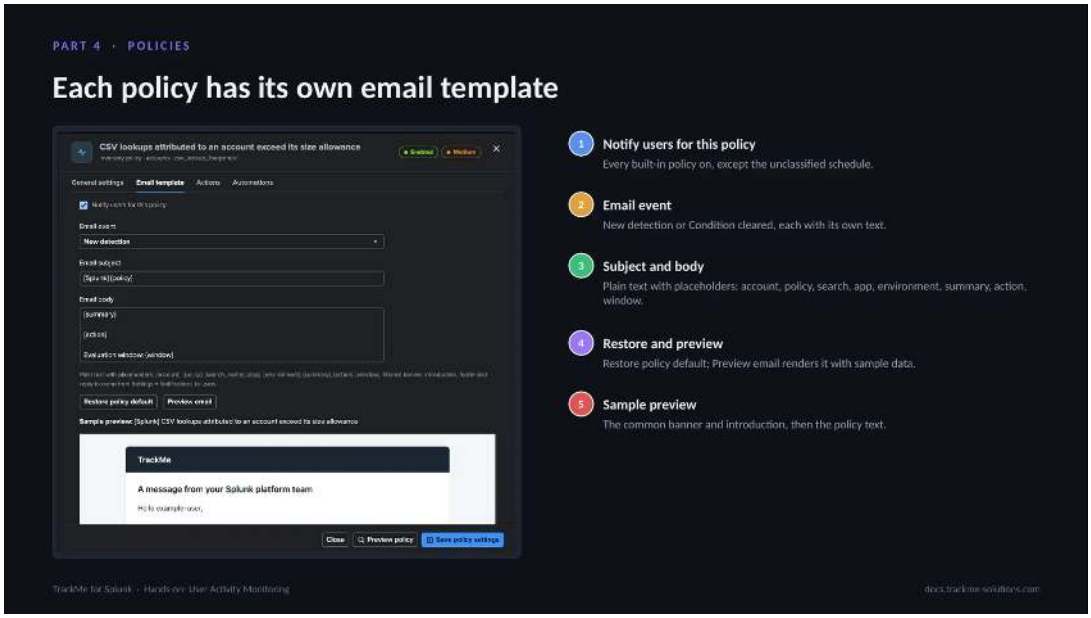
Explaining one evaluation



The policy insight opens from the Last evaluation pill. For Account ran all-time searches the verdict reads Complete: every applicable account was judged, 3 evaluated and 3 matched. The counters show evaluated, matched, unknown, excluded and not applicable.

What the run saw splits the population by outcome, and the last 7 days chart repeats the counters for every policies run since 6 October. Activity left out lists the excluded apps, here trackme, whose searches never count against the rule. Show this policy's findings jumps to the findings the policy raised.

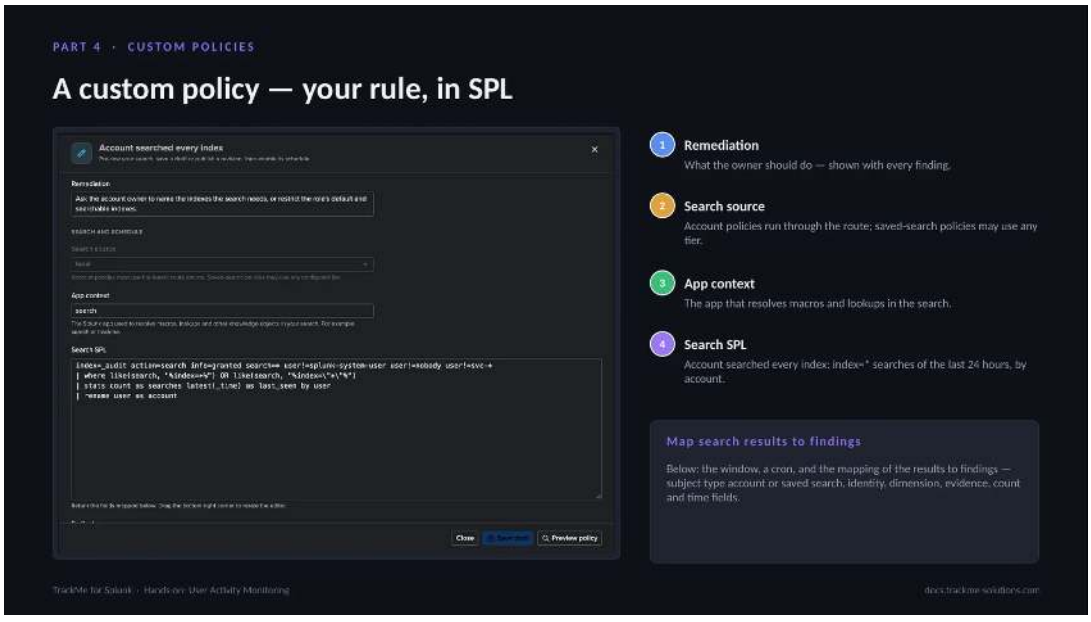
Policy email templates



The edit modal of a policy has four tabs: General settings, Email template, Actions and Automations. Email template holds the words sent to the account. Notify users for this policy is on for every built-in policy except the unclassified schedule.

Each email event (New detection or Condition cleared) has its own subject and body, in plain text with placeholders such as account, policy, search, app, environment, summary, action and window. Restore policy default reverts the text, and Preview email renders it with sample data under the common banner and introduction; a test recipient receives a synthetic test email.

Writing a custom policy

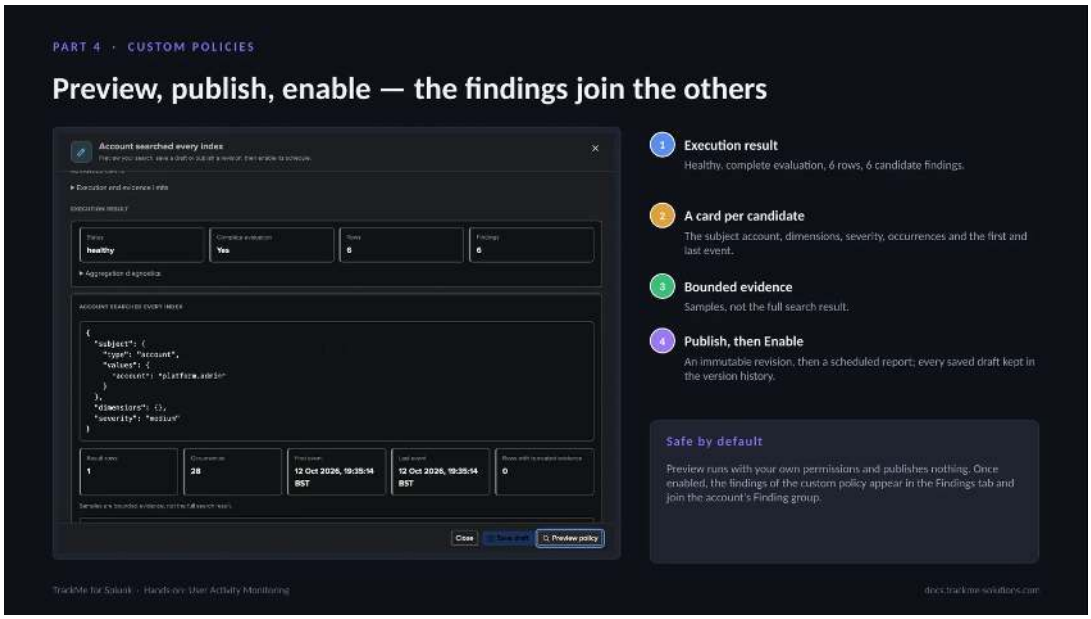


When the catalogue lacks a rule, a custom policy adds it: an SPL search plus a mapping of its results to findings. This one, Account searched every index, flags accounts that searched index=* in the last 24 hours. The remediation text is shown with every finding. The search source is the route for account policies, while saved-search policies may use any tier, and the app context resolves the macros and lookups of the search.

Below the SPL, the policy defines its window, a cron, and the mapping of the results to findings: subject type (account or saved search), identity, dimensions, evidence, count and time fields. A tenant holds up to 100 definitions and 25

enabled policies.

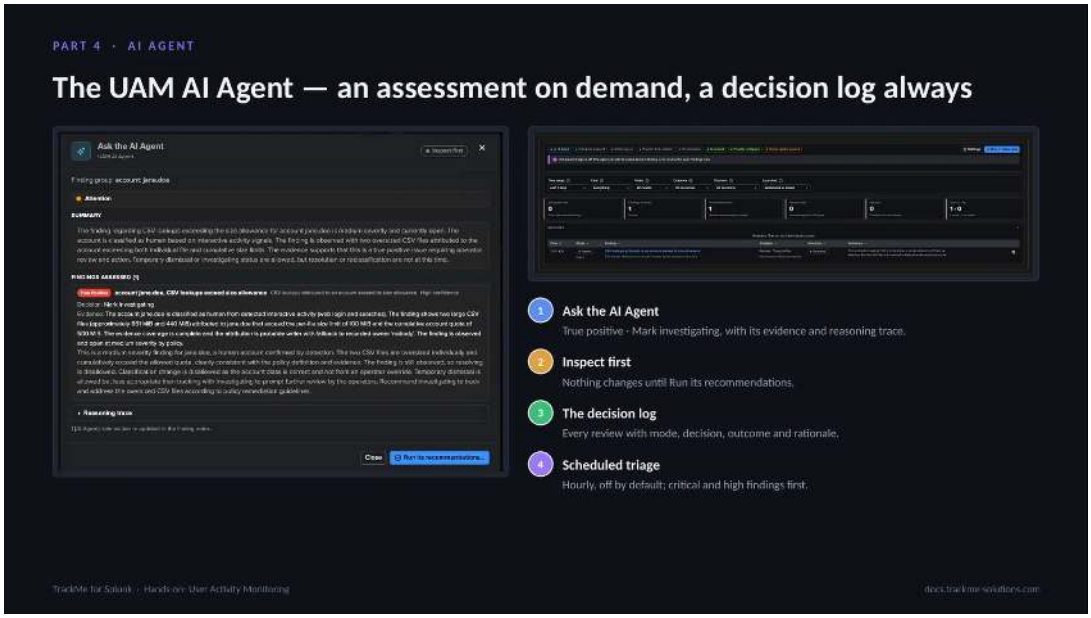
Preview, publish and enable



Preview runs the search as the author, asynchronously, and publishes nothing. The execution result reads healthy, complete evaluation, 6 rows and 6 candidate findings. Each candidate is a card with its subject, dimensions, severity, occurrences and first and last event; the evidence is a bounded sample, not the full search result.

Publish freezes an immutable revision, and Enable installs its scheduled report; every saved draft stays in the version history. Once enabled, the findings of the custom policy appear in the Findings tab and join the Finding groups of the accounts concerned, next to the built-in policies.

Assessments and the decision log

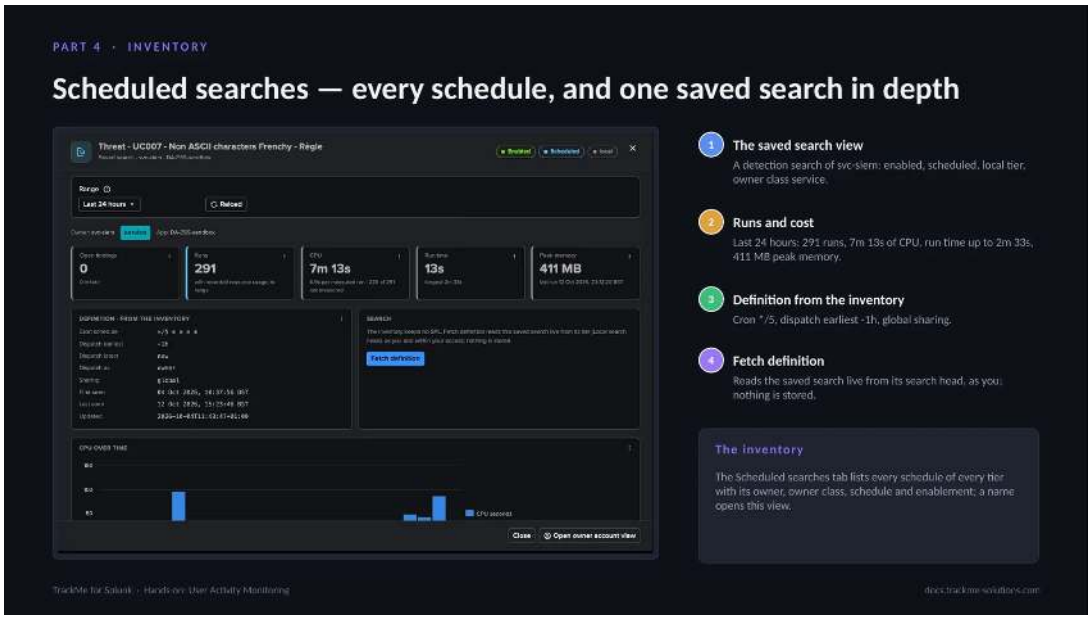


Ask the AI Agent on a finding reads the evidence of the Finding group and returns an assessment, here a true positive of medium severity with its evidence and reasoning trace, and recommends a decision, Mark investigating. In inspect mode nothing changes until Run its recommendations is selected.

The AI Agent tab logs every review, asked or scheduled, with its mode, decision, outcome and rationale. Scheduled triage runs hourly and is off by default, critical and high findings first; in act mode it applies only the decisions the

tenant allows and never overrides an operator decision.

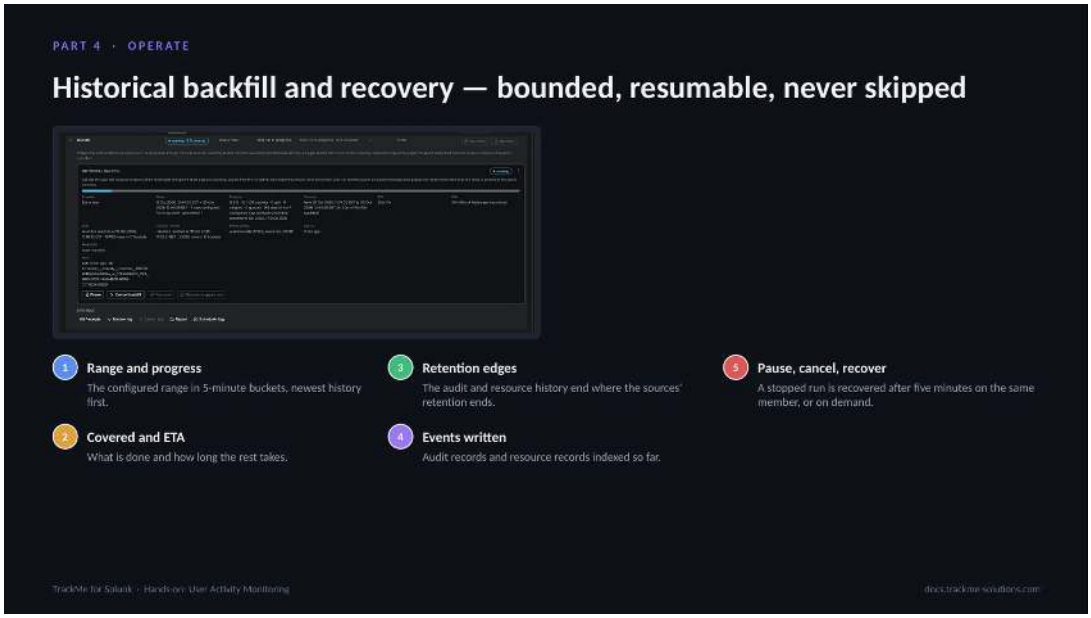
The scheduled-search inventory



The Scheduled searches tab lists every schedule of every tier with its owner, owner class, schedule and enablement; a name opens the saved search view. Here, a detection search of the svc-siem service account is enabled, scheduled on the local tier and shared globally.

Over the last 24 hours it ran 291 times for 7m 13s of CPU, with a longest run of 2m 33s and a peak memory of 411 MB. The definition from the inventory shows a cron of */5 and a dispatch earliest of -1h. Fetch definition reads the saved search live from its search head with the caller's permissions; nothing is stored.

Backfill and recovery

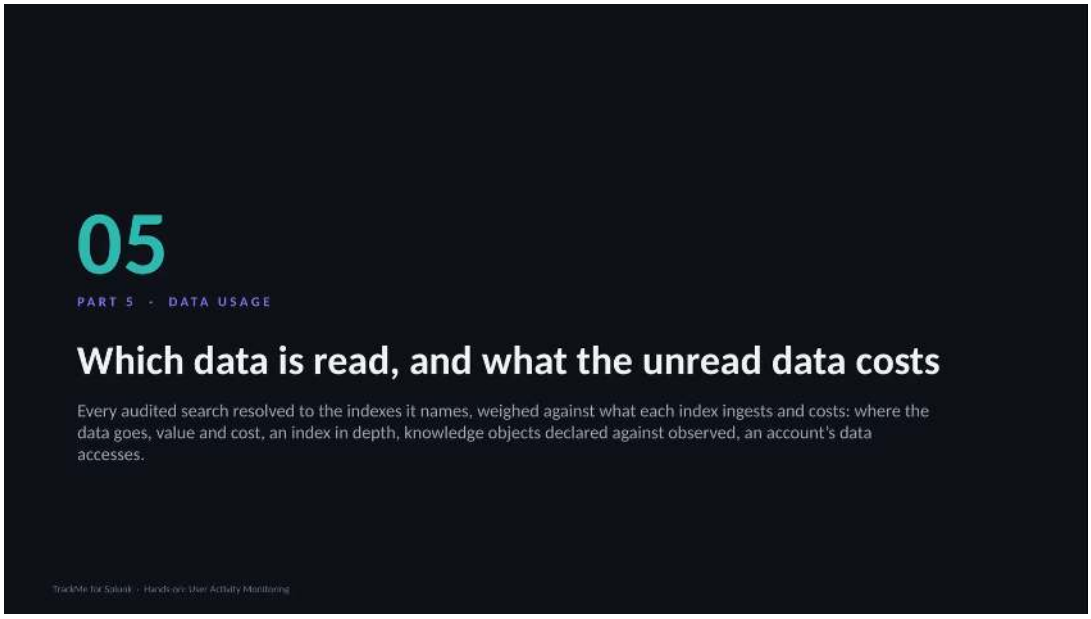


The backfill shows the configured range in 5-minute buckets collected newest first, with what is covered and the estimated time for the rest. The retention edges show where the audit and resource history end because the sources' retention ends there, and the counters show the audit records and resource records written so far.

Pause and Cancel backfill control the run; a stopped run is recovered after five minutes on the same member, or on demand. The backfill never collects a span twice, and a bucket that fails is held, split and resumed, never skipped; it

retires itself once the range is covered. The Configuration Guardian check uam_collection_health surfaces whatever stays wrong and clears itself.

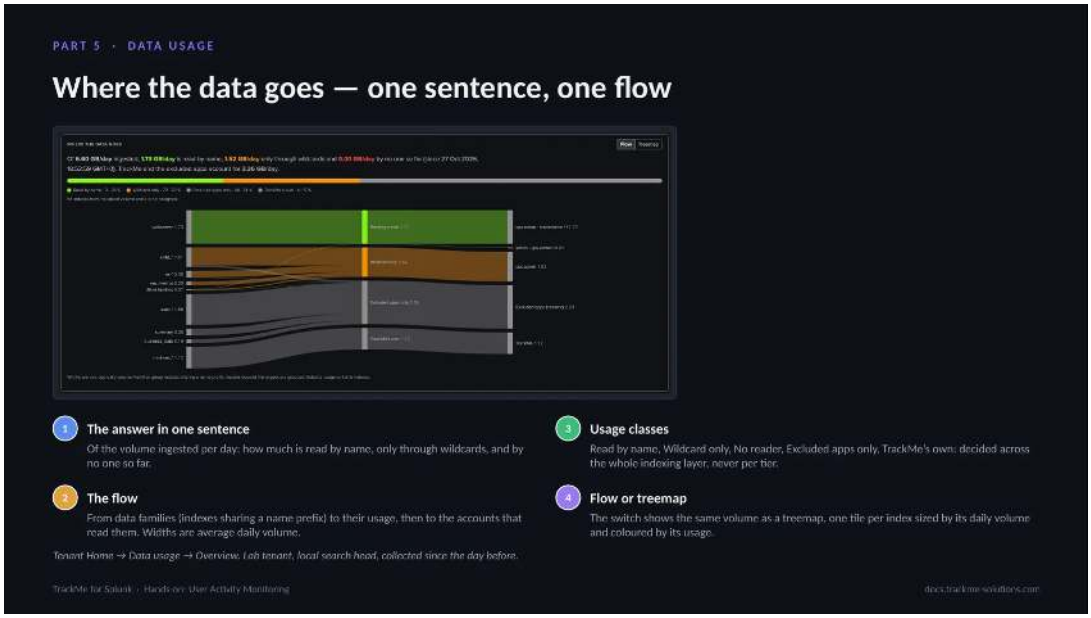
Part 5 — Data usage



Data usage, new in TrackMe 2.4.22, resolves every audited search to the indexes it names, through its macros, event types, saved searches and data models, and weighs that against what each index ingests and costs, read from linked VOL and DSM tenants.

One rule runs through every view: an absence of reads is a claim, made only when the evidence supports it. The part covers where the data goes, value and cost, an index in depth, knowledge objects declared against observed, and an account's data accesses.

From ingestion to readers



The Overview opens on one sentence: of 6.60 GB per day ingested, 1.73 GB is read by name, 1.52 GB only through wildcards and none by no one so far, since the collection is a day old. The flow runs from data families, indexes sharing a name prefix, to their usage and then to the accounts that read them; widths are average daily volume.

The usage classes (Read by name, Wildcard only, No reader, Excluded apps only, TrackMe's own) are decided across the whole indexing layer, never per tier. A wildcard such as index=cribl_* is a reference, not a read. The Flow and Treemap switch shows the same volume as tiles sized by daily volume.

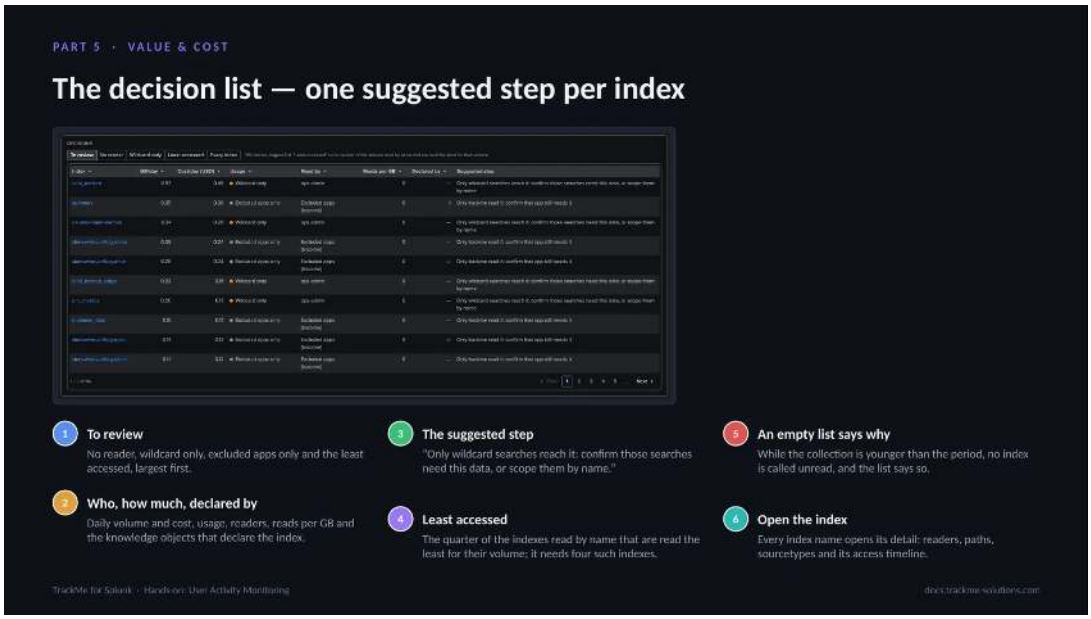
Weighing volume against reads



Value & cost opens with the coverage card: the collection start and the date from which an index can be called unread. Ingestion by usage shows the daily volume as a treemap coloured by usage class. Volume against reads places one dot per index on log scales, with a shaded corner for the indexes larger than the median and read fewer than ten times.

Cost of disuse prices what is not read by name: here 1.52 GB per day, 1.29 USD per day, reached only by wildcards. The cost comes from the linked VOL tenant's costs when they are enabled; otherwise no cost is shown rather than a guessed one.

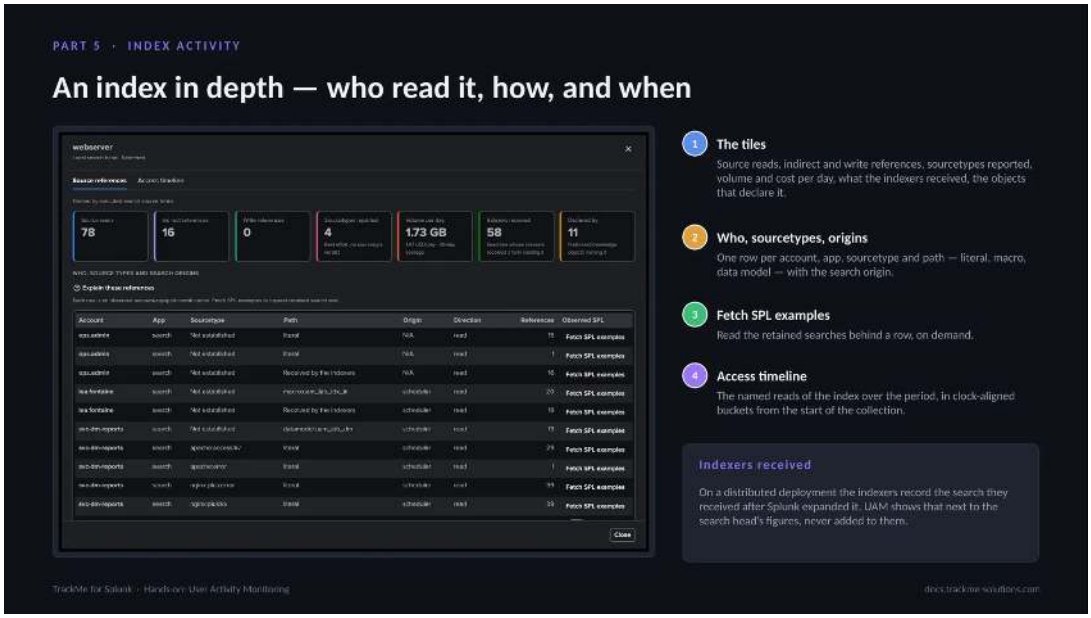
From the map to decisions



The Decisions list gives one row per index, largest first, with its daily volume and cost, usage, readers, reads per GB, the knowledge objects that declare it and a suggested step, such as confirming that wildcard searches need the data or scoping them by name.

To review gathers the indexes with no reader, those only wildcards or excluded apps reach, and the least accessed: the quarter of the indexes read by name that are read the least for their volume, ranked only when at least four such indexes exist. An empty list says why, and every index name opens its detail.

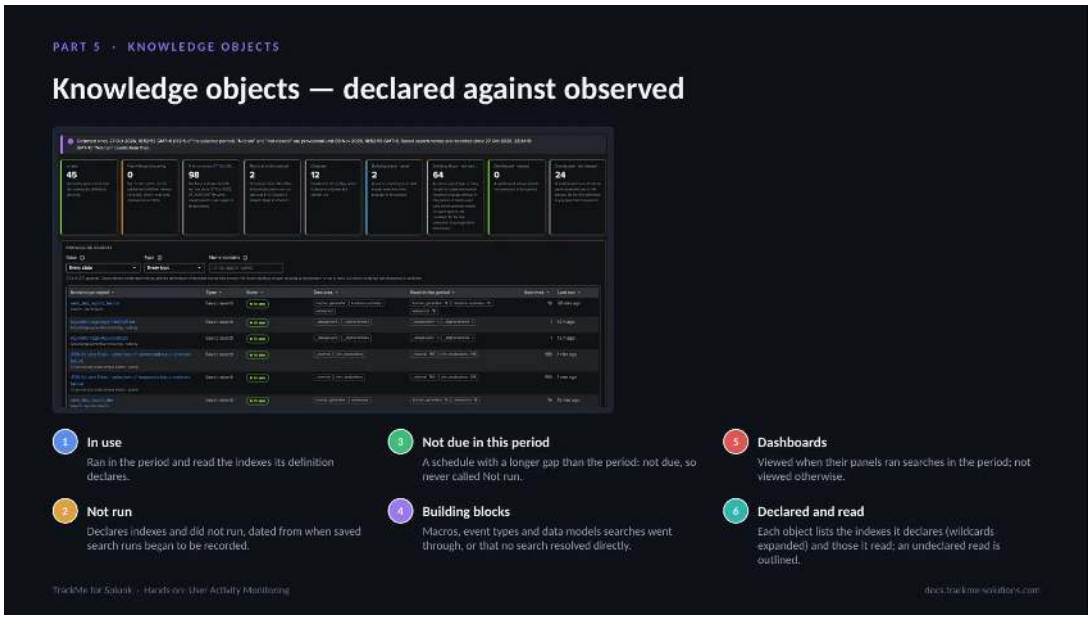
One index, its readers and paths



The webserver detail opens on Source references: 78 source reads, 16 indirect references, 4 sourcetypes reported, 1.73 GB per day at 1.47 USD, 58 searches the indexers received and 11 knowledge objects that declare it. One row per account, app, sourcetype and path shows who read it and how: by literal term, through a macro or through a data model, with Fetch SPL examples on demand.

Sourcetypes are reported on a best-effort basis and never receive a verdict. On a distributed deployment, the indexers' record of the expanded search sits beside the search head's figures, never added to them. The Access timeline tab charts the named reads in clock-aligned buckets.

Declarations checked against runs

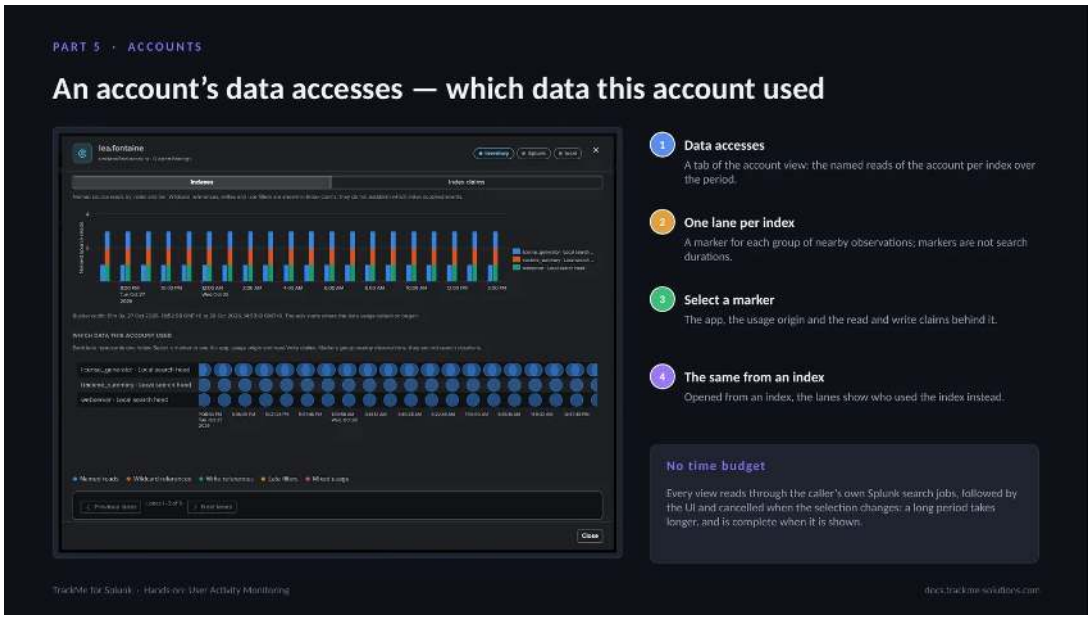


The Knowledge objects view sets each published saved search, macro, event type, dashboard and data model against what ran. Tiles count the states: 45 in use, 98 not run, 2 not due in this period, 12 disabled, building blocks used or

not seen, and dashboards viewed or not.

Each object lists the indexes its definition declares, wildcards expanded across the index catalog, next to the indexes its searches read; an undeclared read is outlined. Not run is dated from when saved search runs began to be recorded, and a schedule with a longer gap than the period is not due, never not run.

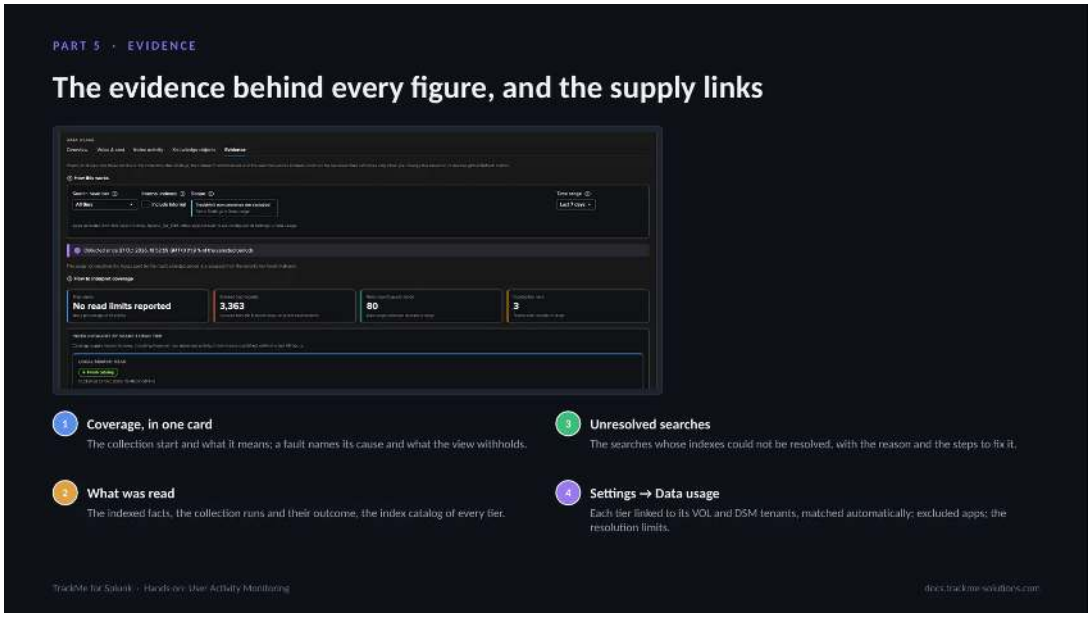
An account's indexes over time



The account view of lea.fontaine has a Data accesses tab: the named reads of the account per index over the period, then one lane per index with a marker for each group of nearby observations. Markers are not search durations. Selecting one shows the app, the usage origin and the read and write claims; opened from an index, the same lanes show who used it.

Every Data usage view reads through the caller's own Splunk search jobs, followed by the UI and cancelled when the selection changes or the view closes. There is no time budget: a long period takes longer and is complete when it is shown.

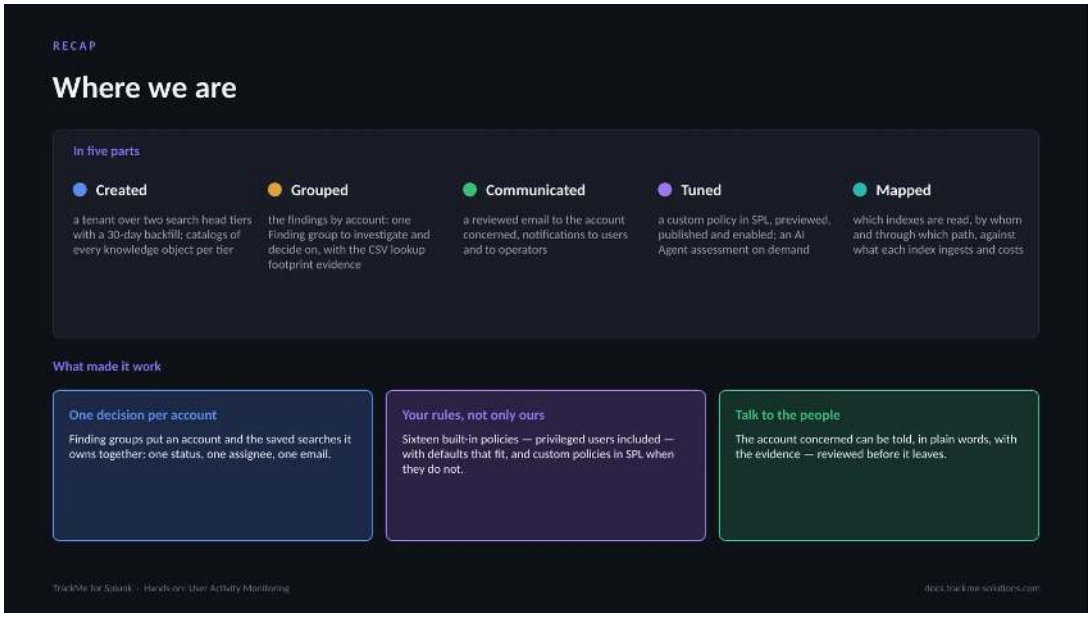
Evidence and supply links



Evidence shows what supports the figures: one coverage card with the collection start and any fault with its cause, the indexed fact records, the collection runs and their outcome, the index catalog of every tier, and the searches whose indexes could not be resolved, each with its reason and the steps to fix it.

In Settings, Data usage links each search head tier to the VOL tenant that gives its volumes and to the DSM tenants that give its incoming data, matched automatically on the same route. It also sets the excluded apps and the resolution limits. A link grants no access: every read repeats the caller's visibility.

Where we are



In five parts, a tenant was created over two search head tiers with a 30-day backfill and catalogs of every knowledge object per tier. Its findings were grouped by account into Finding groups and investigated, with the CSV lookup footprint evidence, and the accounts holding high privileges were reviewed with their policies. A reviewed email went to the account concerned, next to the notifications to users and to operators. A custom policy was written in SPL, previewed, published and enabled, and the AI Agent gave an assessment on demand. Data usage mapped which indexes are read, by whom and through which path, against what each index ingests and costs.

Three things make it work: one decision per account through Finding groups, rules beyond the built-in catalogue through custom policies, and plain-language communication with the people behind the accounts.

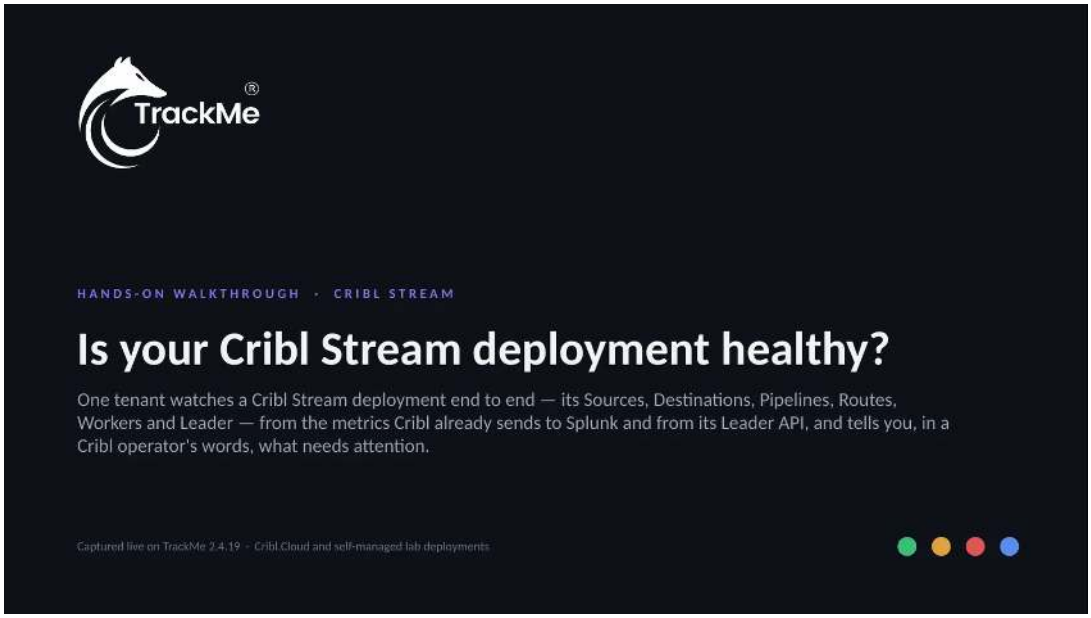
2.1.9 Cribl Stream

Monitor a Cribl Stream deployment from the metrics it already sends to Splunk and its Leader API: create the tenant, triage Finding groups, investigate traffic outliers against what each object learned, observe Workers, the Leader and the data paths, and operate the tenant with policies, notifications, the AI Agent, Topology Studio and the Configuration Guardian.

`Download the deck (PDF) · PowerPoint`

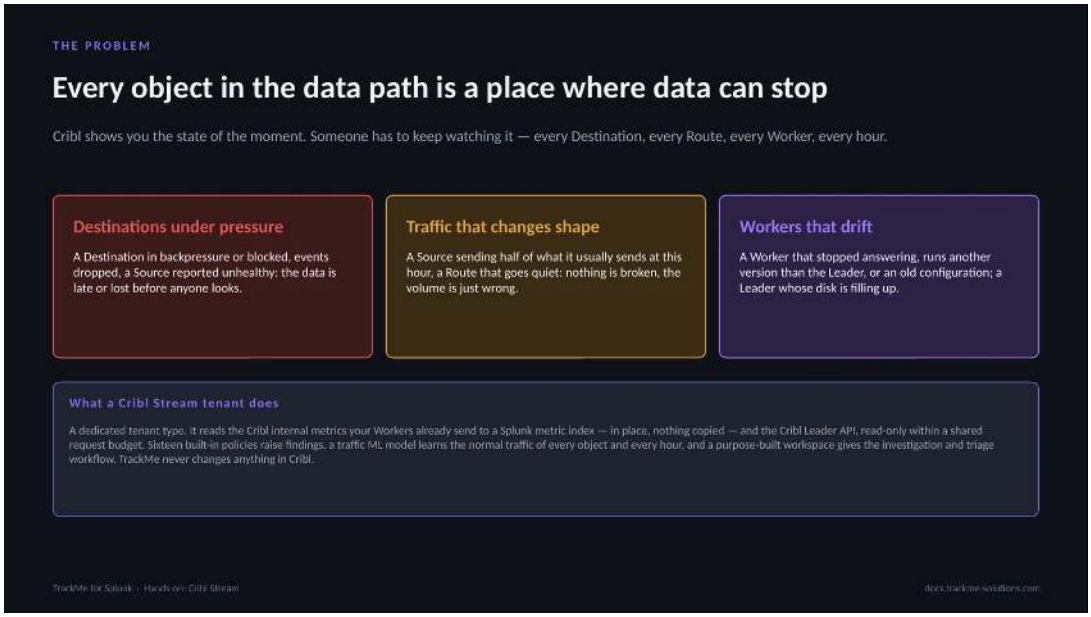
about 35 minutes · 33 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



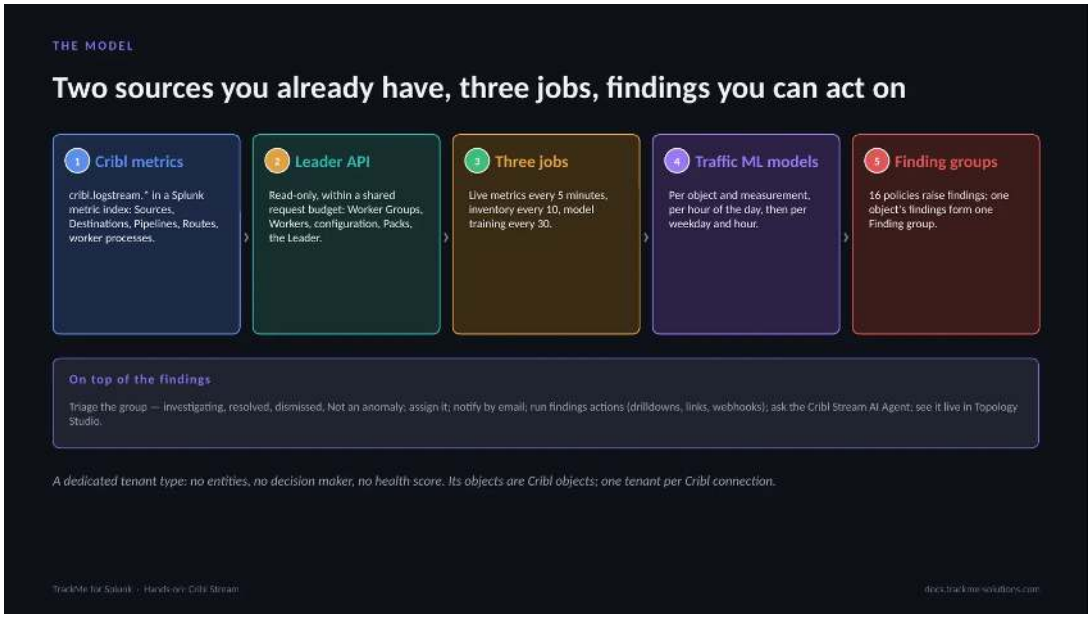
This walkthrough covers the Cribl Stream tenant, generally available in TrackMe 2.4.19. Everything was captured live on lab deployments: a Cribl.Cloud workspace and a self-managed Leader with five Workers. In four parts: create the tenant, read and triage the findings, observe the deployment and its traffic models, and operate the tenant with policies, notifications, the AI Agent and Topology Studio.

Why a dedicated Cribl Stream tenant



Cribl Stream sits in the middle of the data path. Every Source, Route, Pipeline and Destination is a place where data can slow down, back up, be dropped or stop, and every Worker can disconnect or drift from its Worker Group. The Cribl UI shows the state of one moment; a Cribl Stream tenant keeps watching, raises findings and gives you the triage workflow.

How a Cribl Stream tenant works



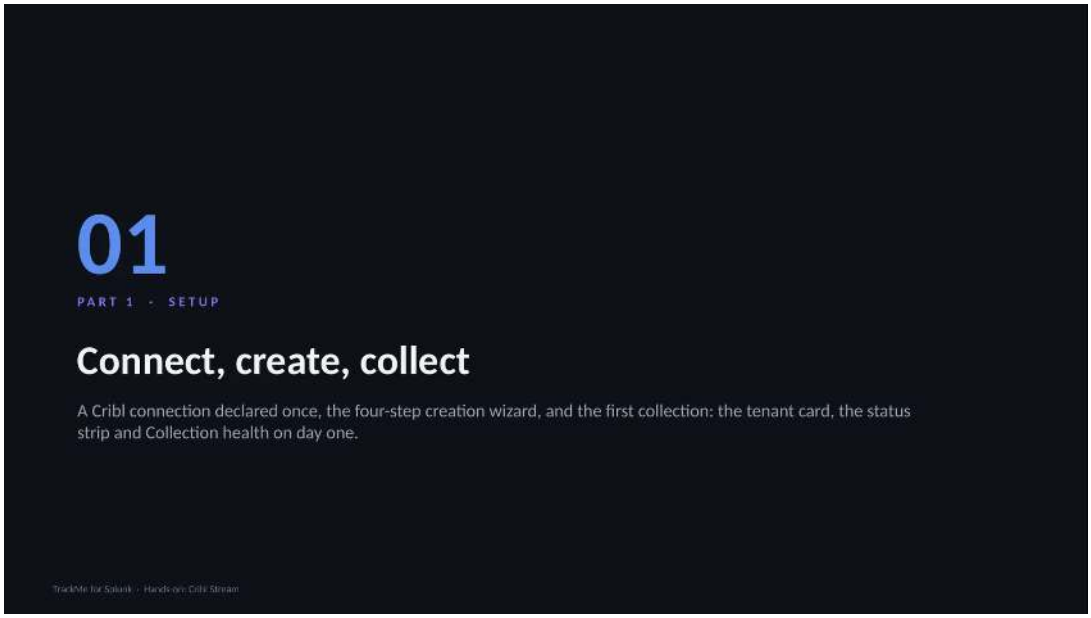
The live metrics job reads the newest five-minute interval of the Cribl metrics in Splunk every five minutes and evaluates the policies; the inventory job reads the configuration and the Workers from the Leader within its request budget; the model training job keeps a traffic ML model per object and measurement. Policies turn what is measured into findings, grouped per object into Finding groups.

The parts of the walkthrough



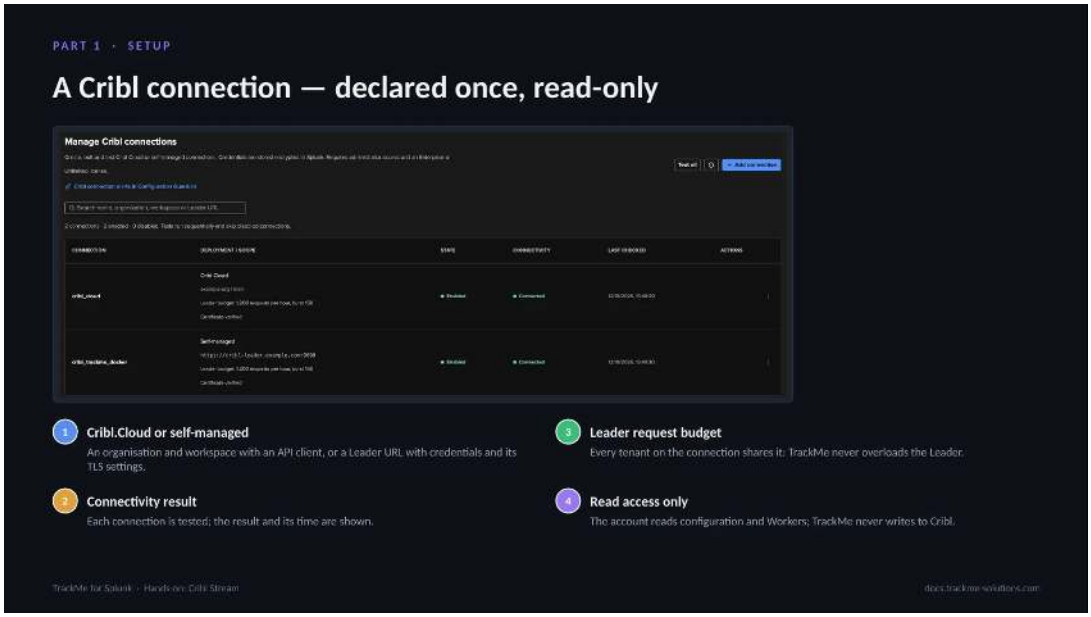
Four parts. Setup: the Cribl connection, the creation wizard, the first collection. Findings: the backlog, Finding groups, the investigation and the triage, assignment and webhooks. Observe: traffic, the traffic ML models, the machines and the data paths. Operate: inventory and exclusions, policies, notifications, the AI Agent, Topology Studio and the collection health.

Part 1 — Setup



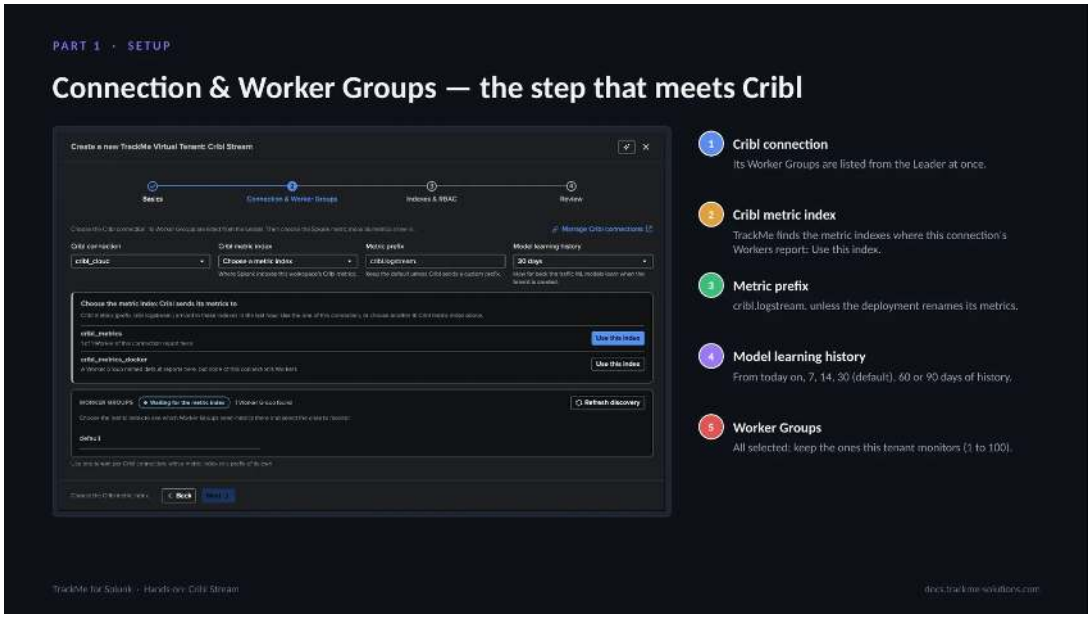
A Cribl connection is declared once and serves Cribl Stream and Cribl Edge tenants. The wizard does the rest, and the first collection starts within minutes. Configuration, Cribl connections holds one connection per Cribl.Cloud workspace or self-managed Leader, read-only. The creation wizard has four steps: Basics, Connection & Worker Groups, Indexes & RBAC and Review.

Declaring the Cribl connection



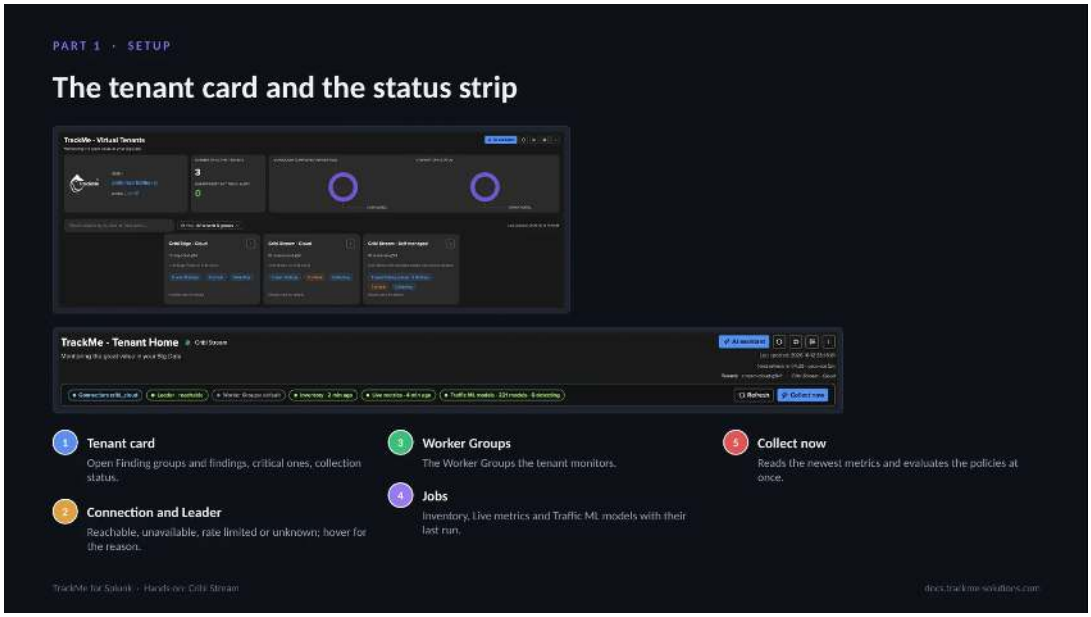
Configuration, Cribl connections. A connection is a Cribl.Cloud workspace with an API client ID and secret, or a self-managed Leader with a user name and password and its TLS settings. The Leader request budget is shared by every tenant that uses the connection. The account only needs read access: TrackMe never writes to Cribl.

Choosing the connection and Worker Groups



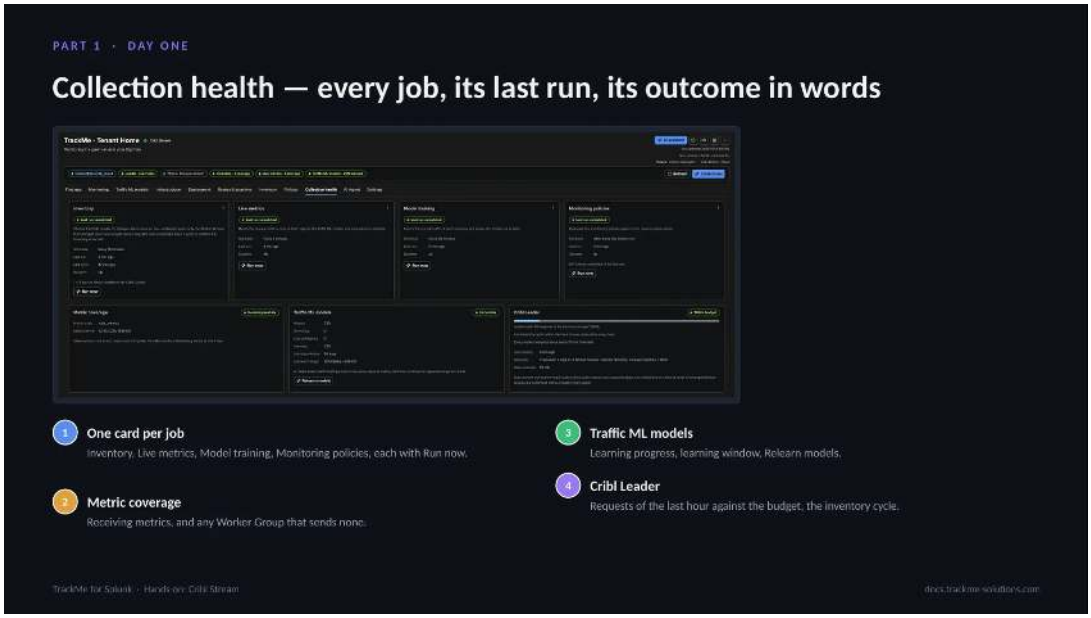
Actions, Create a tracking tenant, Cribl Stream card. Step 1 is the identifier and alias. Step 2 is where TrackMe meets the deployment: the connection, the metric index that TrackMe proposes from where this connection's Workers report, the metric prefix, how much history the traffic ML models learn from at creation, and the Worker Groups. With history in the metric index the models can detect on the first day. Steps 3 and 4 are the indexes, roles and review.

Status at a glance



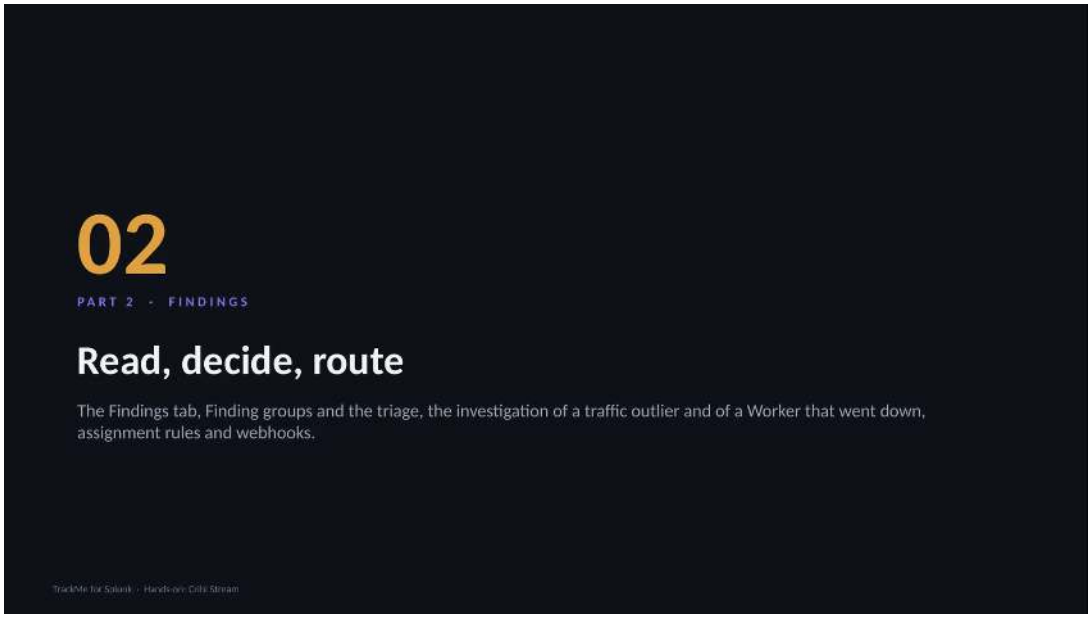
On Virtual Tenants the tenant card shows the open Finding groups and findings, the critical ones and the collection status. Inside the workspace, the status strip says at a glance that the tenant is working: the connection, the Leader, the Worker Groups, the last run of each job and how many traffic ML models are detecting; Refresh, and for a TrackMe administrator of the tenant Collect now.

Collection health on day one



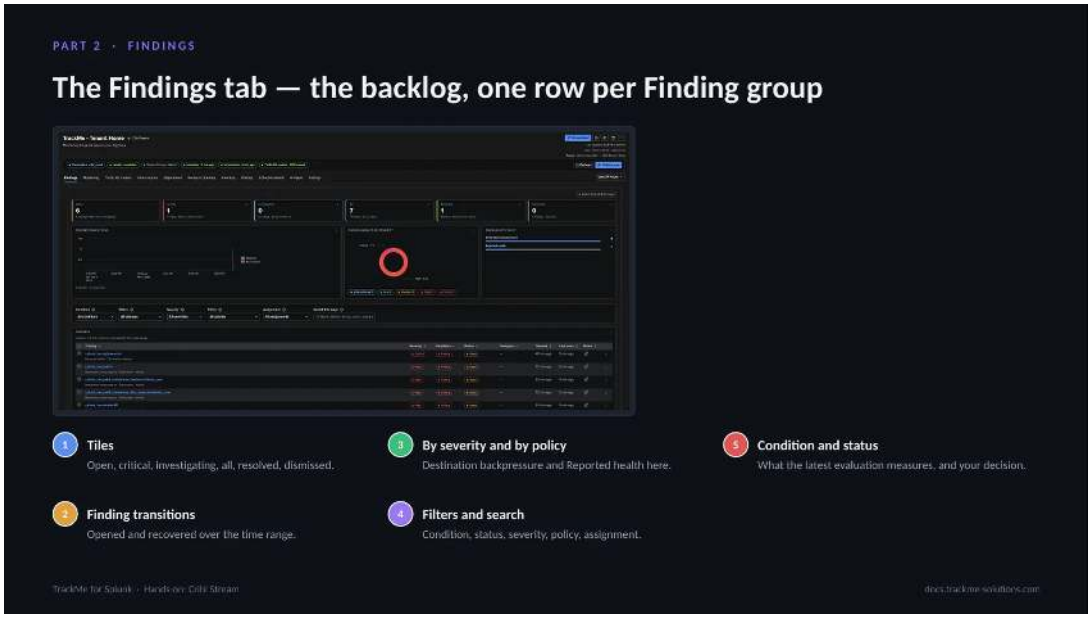
Collection health answers: is TrackMe collecting what it should? One card per job with its last run, its outcome in words, View last run and Run now; the metric coverage; the traffic ML models with their learning progress; and the Cribl Leader card with the requests against the budget and the inventory cycle.

Part 2 — Findings



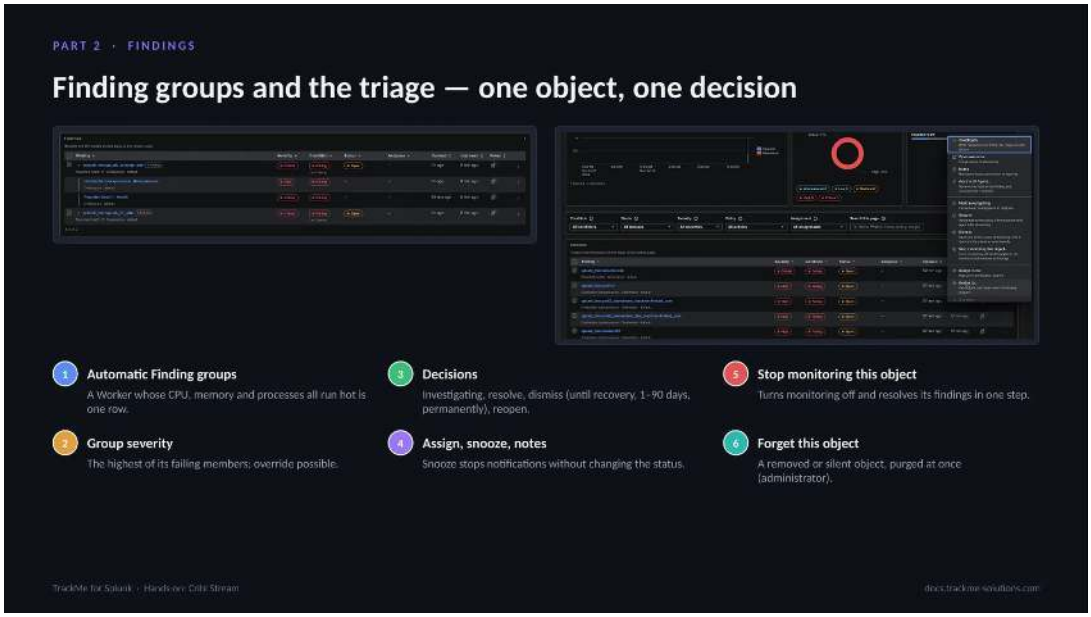
A finding is one policy applied to one measurement of one object. One object's findings form one Finding group: one row, one decision. Each finding shows its condition, what the newest evaluation measures, and its status, the decision taken. Triage, assignment rules and findings actions are shared with the other findings tenants.

The findings backlog



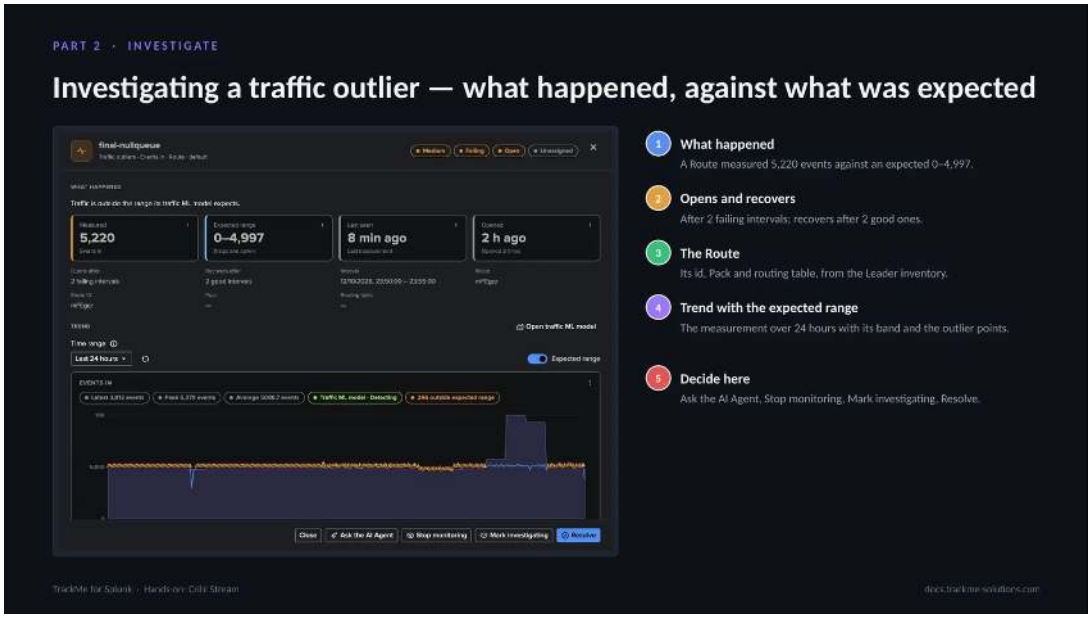
The Findings tab is the backlog and the tab the workspace opens on. Each finding shows its condition — Failing, Recovered, Awaiting data — and its status, the decision: Open, Investigating, Resolved, Dismissed. Severity uses the TrackMe scale from Informational to Critical. A measured recovery resolves an open finding by itself; a new failure reopens it.

One decision per object



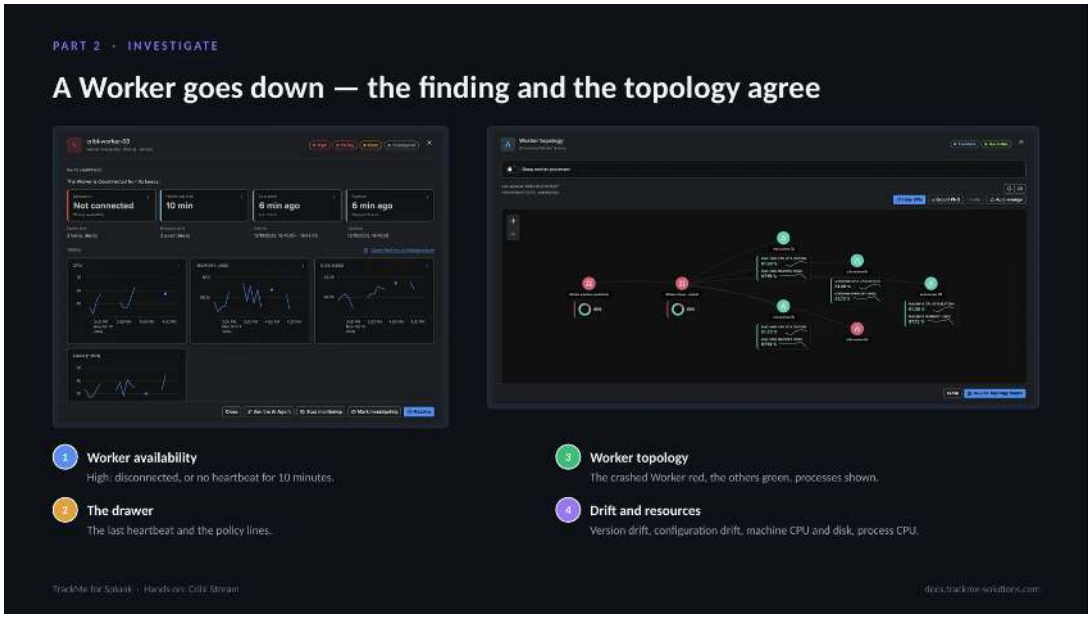
One object can fail several policies at once. TrackMe groups its findings automatically into a Finding group, so you make one decision: status, assignee, snooze, dismissal and severity override live on the group. Groups can also be merged, edited or ungrouped by hand. The row menu carries Ask the AI Agent, the decisions, and two Cribl-specific actions: Stop monitoring this object and Forget this object.

Investigating a traffic outlier



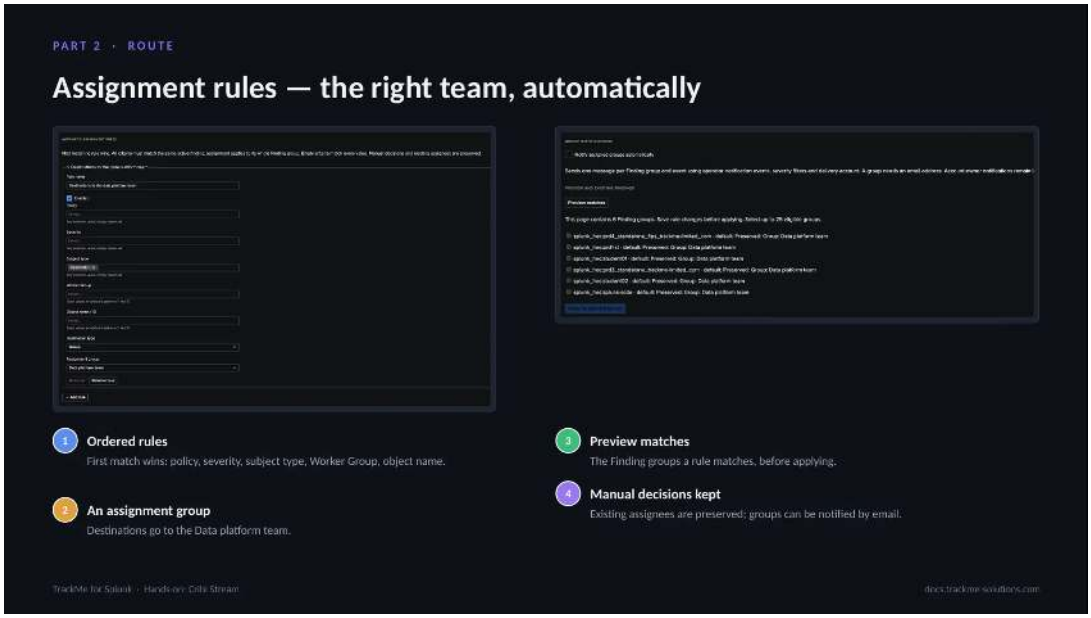
Selecting a row opens the investigation drawer: what happened in plain words, the measurement against its expected range, the open and recovery rules, the object in the inventory, the trend chart with the band, notes and history. For a Worker or the Leader the chart shows the policy threshold and recovery lines instead of a band.

When a Worker goes down



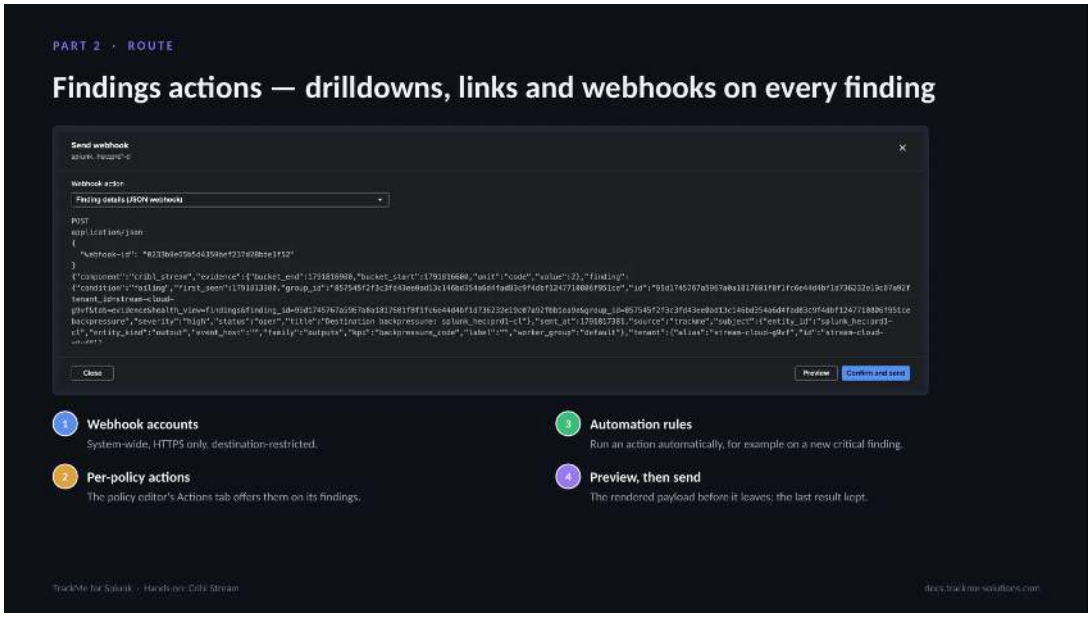
From the Leader, TrackMe knows every Worker and how it should look. A Worker that stopped answering opens a High Worker availability finding. The Worker topology of the Infrastructure tab shows the same story: the crashed Worker in red with its processes. Version and configuration drift, machine CPU and disk and process CPU follow the same pattern.

Routing findings to a team



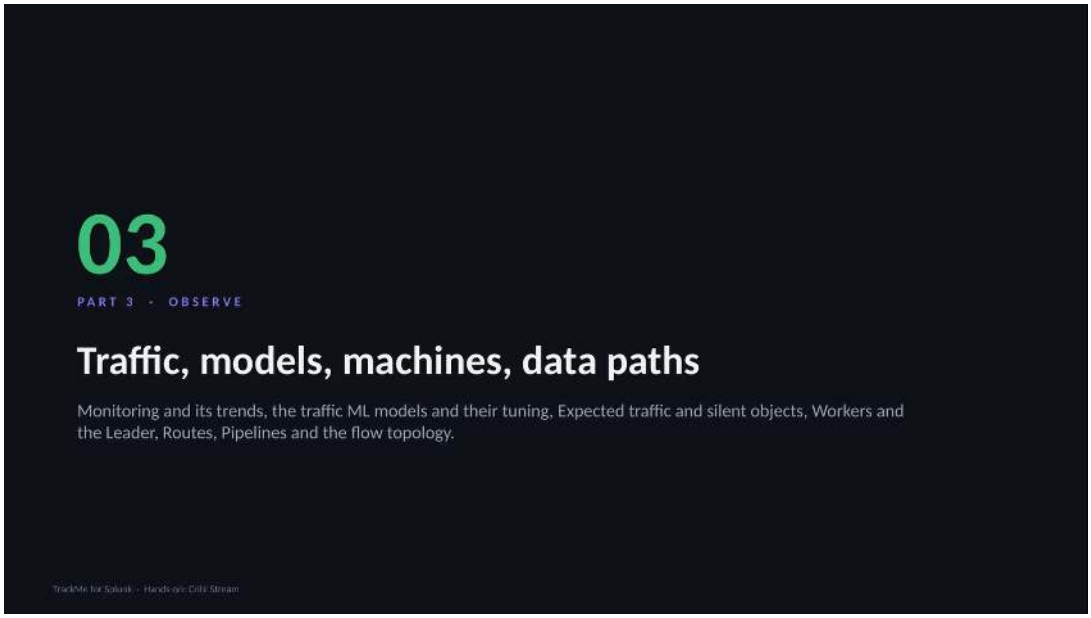
Settings, Assignment. Assignment groups and ordered rules route new findings to the right team. Preview matches lists the Finding groups a rule matches, here already assigned to the Data platform team and preserved, with Apply to selected groups for the existing ones. Assignment groups carry a team email, so an assigned group can be notified automatically.

Drilldowns, links and webhooks



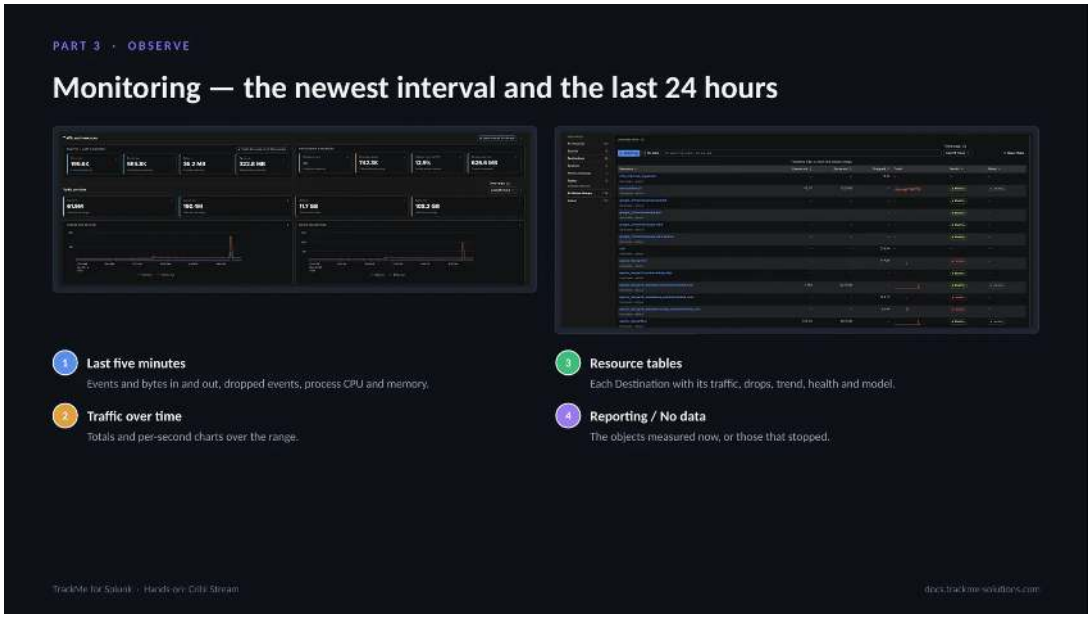
Findings actions add drilldown searches, links and webhooks to the findings, run by hand or by automation rules, through system-wide Webhook accounts. The preview shows the payload before it is sent. The concepts are shared with the other findings tenants. An automation rule can run an action without anyone opening the finding, for example posting every new critical finding to a chat channel.

Part 3 — Observe



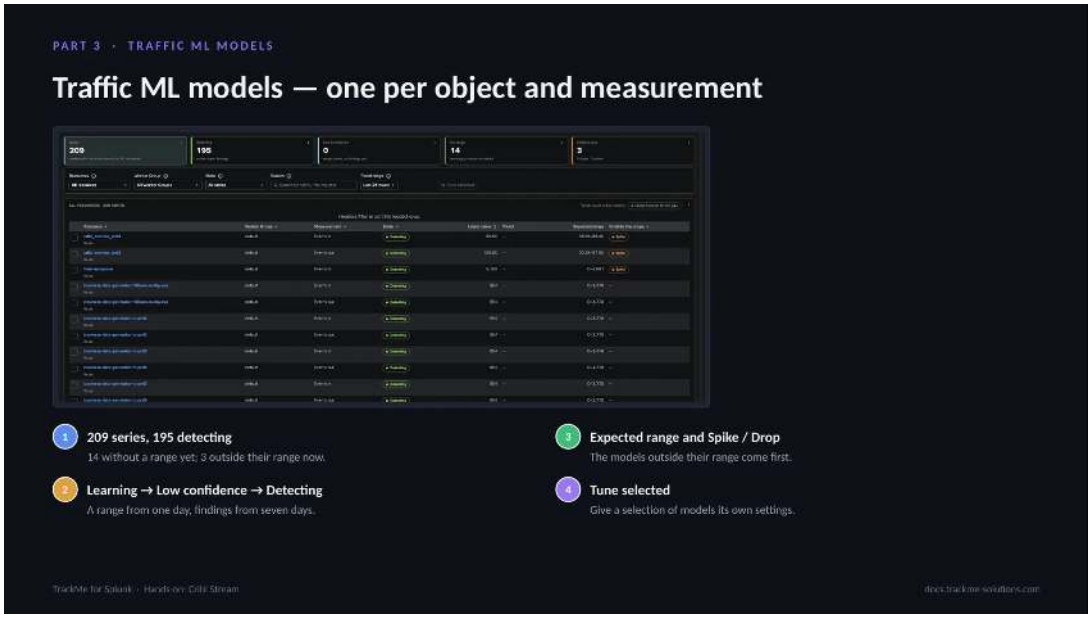
Every figure has a menu with View chart data and Open in Search, a plain editable search that reproduces it. Monitoring, the traffic ML models, Expected traffic and silent objects, the Infrastructure tab with Workers and the Leader, and Routes & pipelines with the flow topology are covered in turn.

Traffic and resources



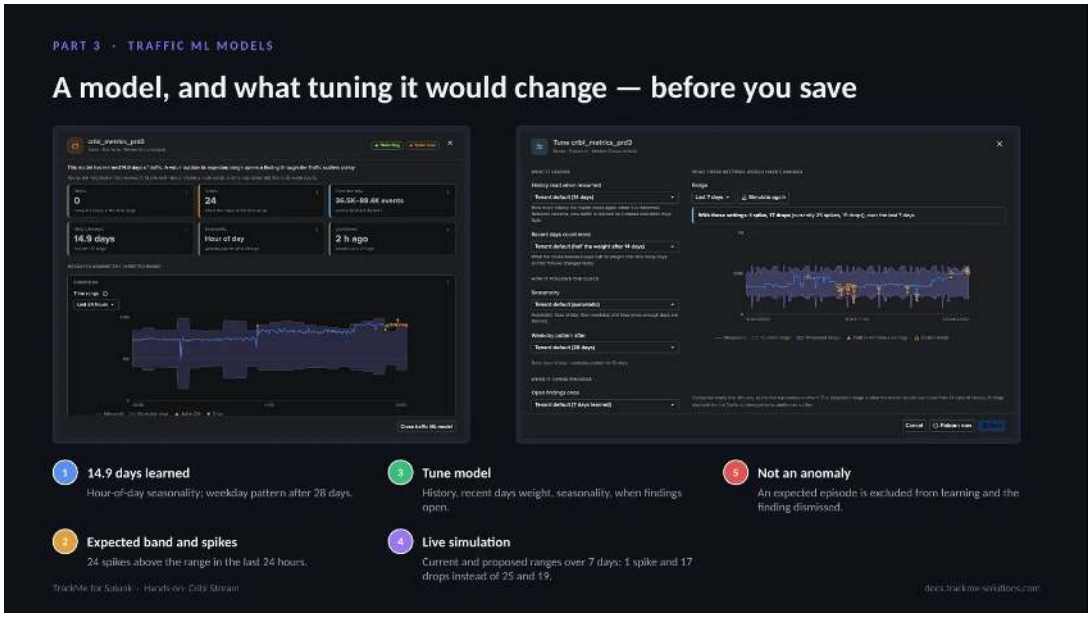
Monitoring shows the newest five-minute interval of every object and the traffic over the selected range. The resource tables list each Source, Destination, Pipeline, Route and worker process with its latest values, its health, its traffic ML model and a trend; a row opens its charts with the expected range and, for a Source or Destination, its live status from the Leader.

One model per object and measurement



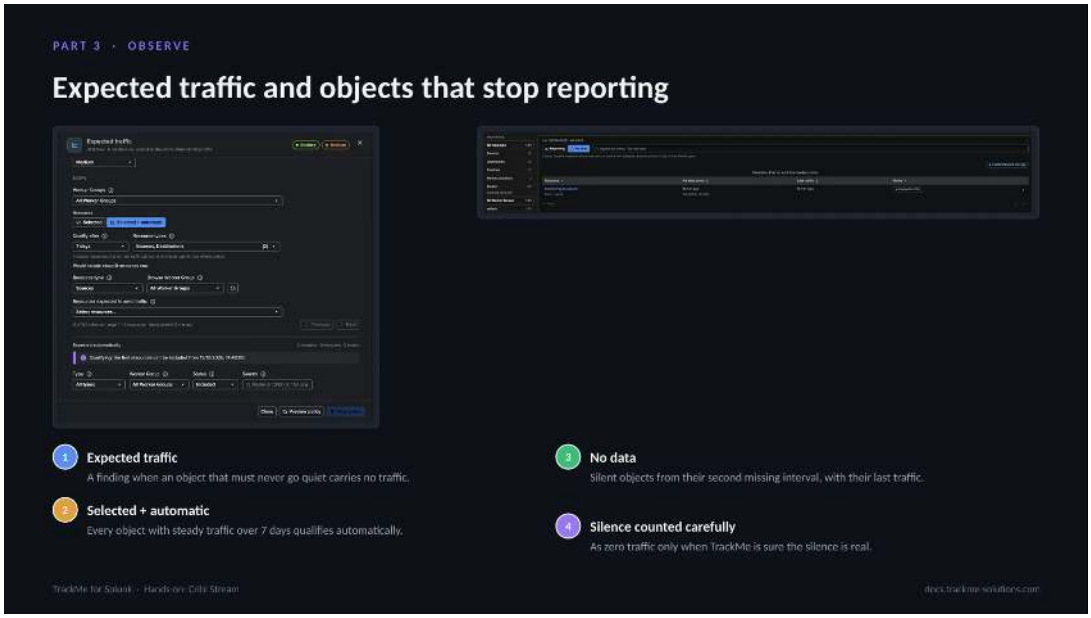
Every Source, Destination, Pipeline and Route gets its own traffic ML model. A model learns per hour of the day, then per weekday and hour after four weeks, with recent days counting more, and keeps learning from the newest intervals: there is no periodic full retrain. Learning means no range yet; Low confidence draws the range without findings; Detecting opens findings through the Traffic outliers policy.

A model and its tuning



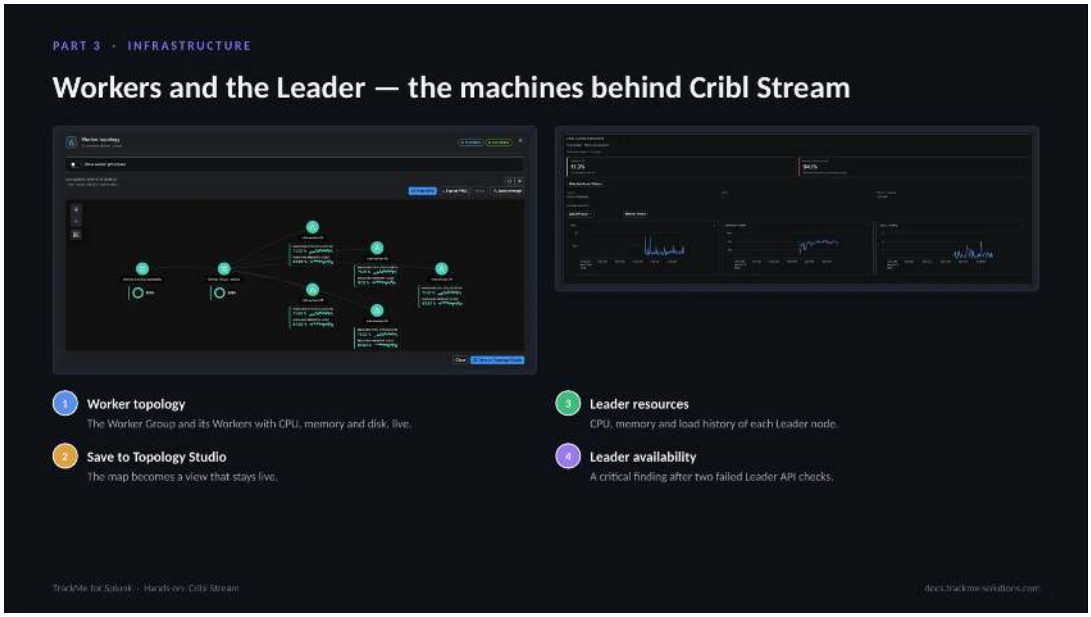
A model shows what it learned, its drops and spikes, its expected range, the history learned and its seasonality. Tune model gives a model or a selection its own settings, and the tuning screen simulates the current and proposed bands over 24 hours, 7 or 30 days, with the drops and spikes each would have raised, before anything is saved. Not an anomaly excludes an expected episode from what the model learns.

Expected traffic and silent objects



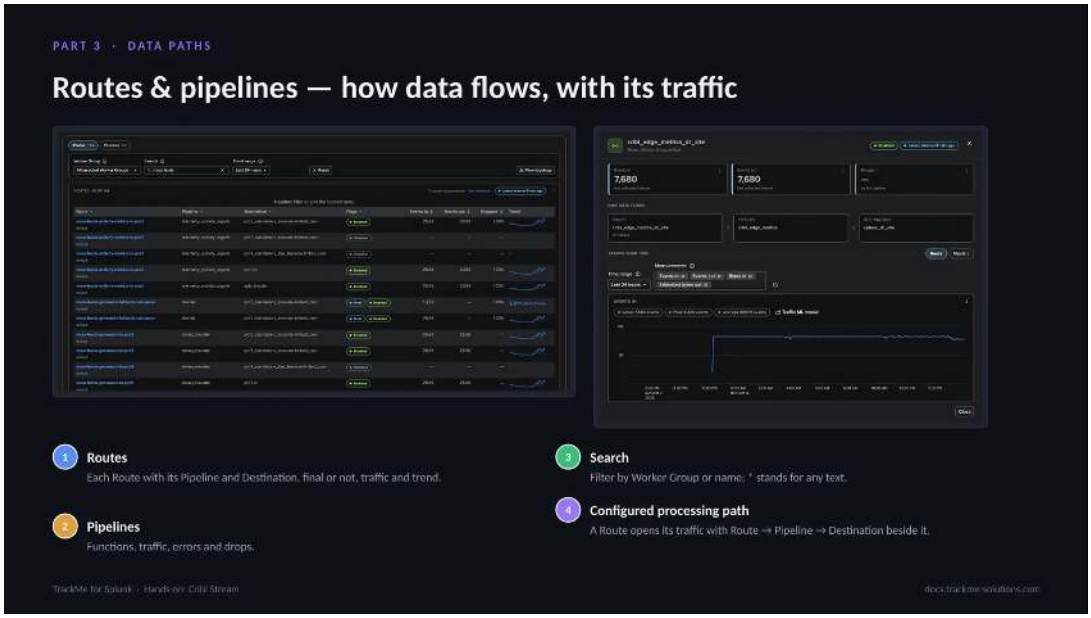
Some objects must never go quiet. Expected traffic opens a finding when an expected object carries no traffic; in Selected plus automatic mode TrackMe also expects every object that carried traffic steadily over the qualifying window. TrackMe remembers every object it measured: a silent object appears under No data, and Traffic outliers counts its silence as zero traffic only when the inventory, the Worker Group and the model confirm it is real.

Workers and the Leader



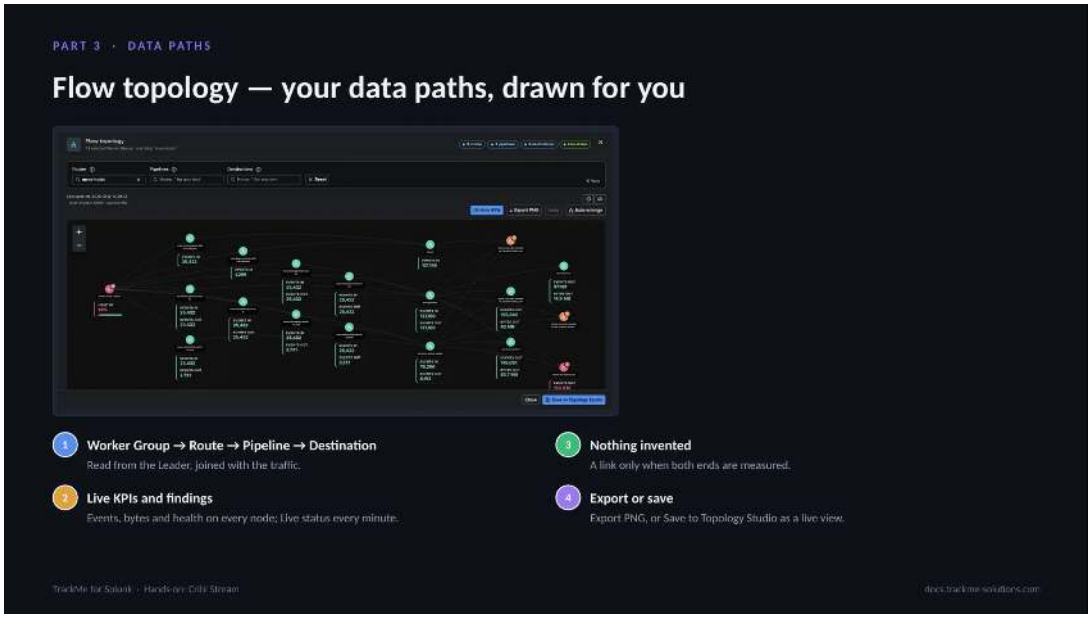
Infrastructure shows the machines: the Leader nodes with their resources and history, every Worker with its heartbeat, CPU, memory, load and disk, and every worker process. Worker topology draws them as a live map, ready to save to Topology Studio. TrackMe checks the Leader API at every live run: two failed checks open a critical Leader availability finding.

Routes and pipelines



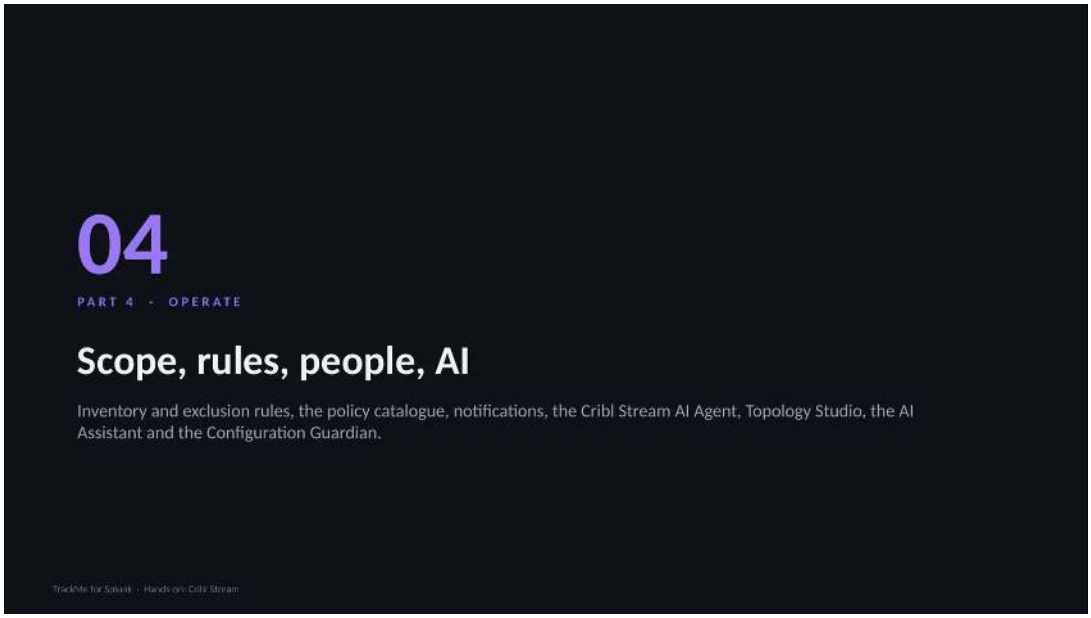
Routes & pipelines shows how data flows: every Route with its Pipeline and Destination and its measured traffic, and every Pipeline with its functions, errors and drops. A Route or Pipeline opens its traffic over time with its configured processing path beside it. Filters select a Worker Group or a name, with * standing for any text.

Data paths drawn as flows



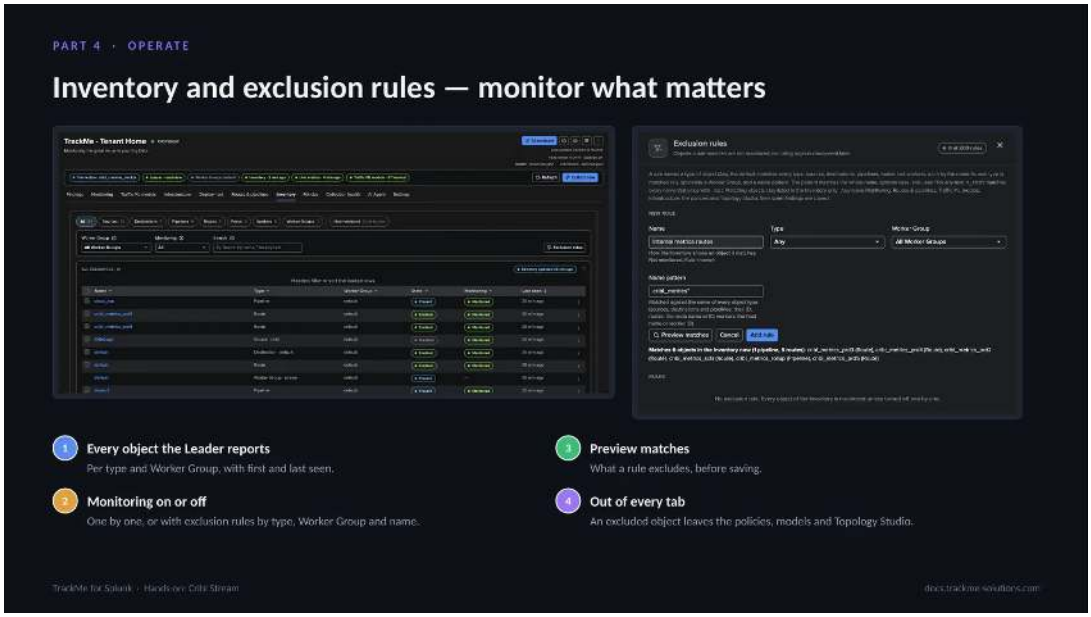
Flow topology draws the Routes the table selects as flows, left to right. You configure nothing: TrackMe reads each Route's Pipeline and Destination from the Leader and joins them with the traffic it measures. A Destination fed by several Routes shows once, so you see which Routes depend on a Destination under backpressure. It covers up to 200 Routes at a time.

Part 4 — Operate



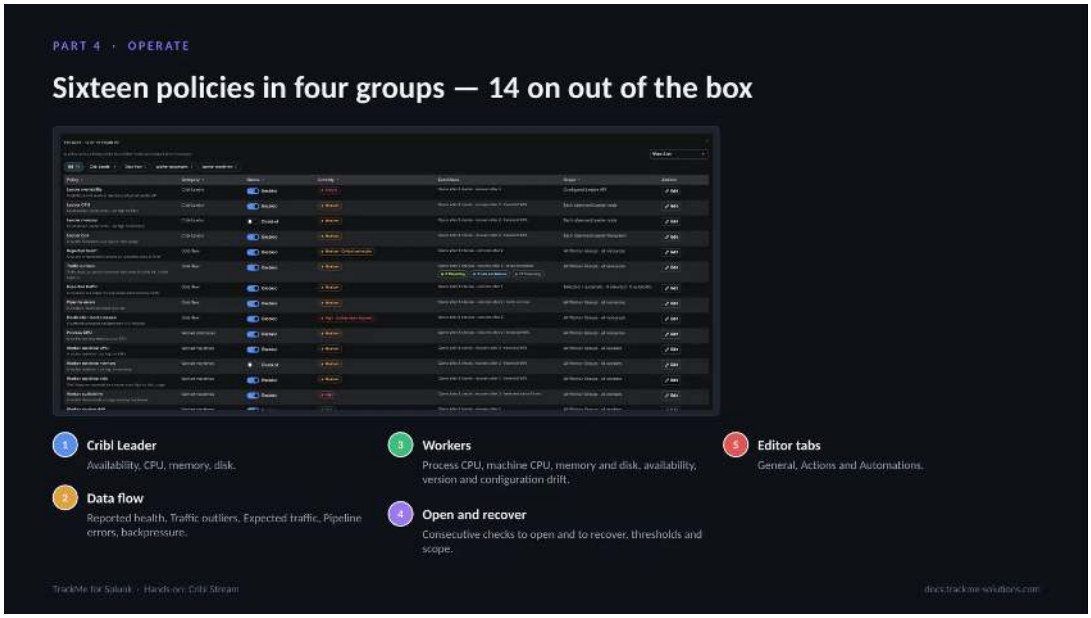
The Inventory and exclusion rules decide what is monitored; the sixteen policies, notifications, the Cribl Stream AI Agent, Topology Studio, the AI Assistant with the Concierge, and the Configuration Guardian complete the operation of the tenant. Policies, exclusion rules and settings are changed by a TrackMe administrator of the tenant; every role of the tenant reads them.

Choosing what to monitor



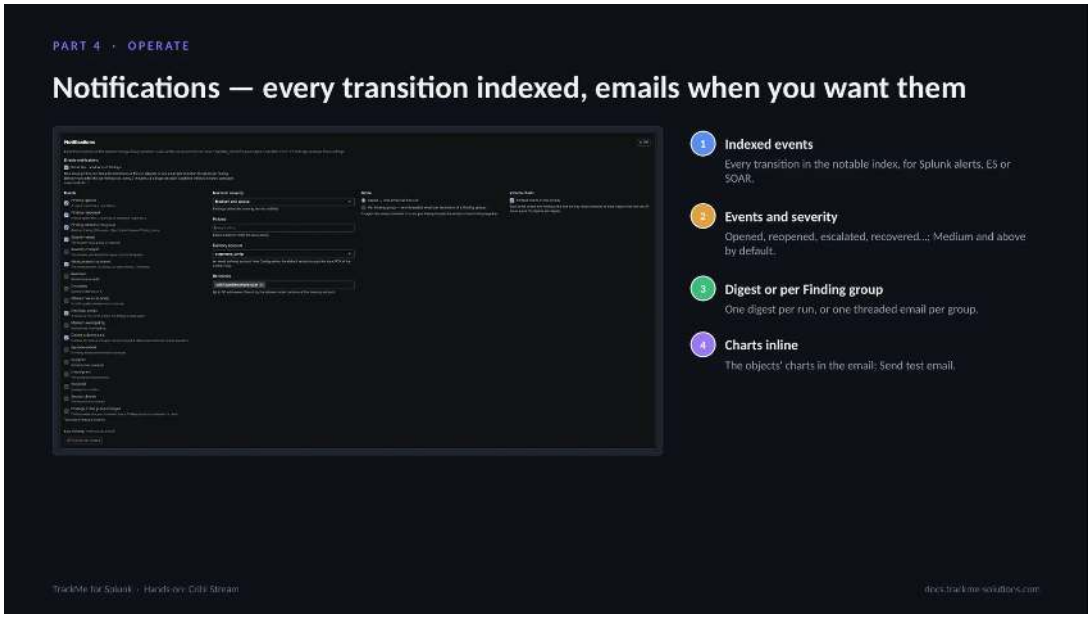
The Inventory lists every object the Leader reports, with its monitoring state. Turn monitoring off for objects you do not care about — a test Route, a retired Destination, a lab Worker — one by one or with exclusion rules. An object turned off appears only in the Inventory: it leaves the other tabs, the policies, the model training and Topology Studio, and its open findings close.

The policy catalogue



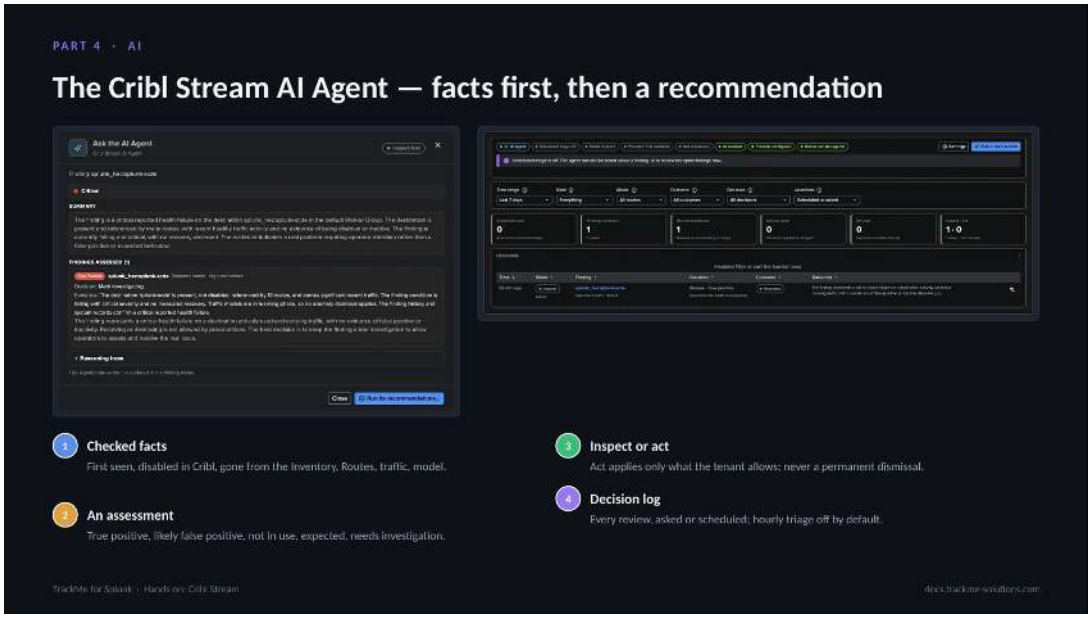
The 16 built-in policies in four groups. Each has its enablement, its severity, how many consecutive checks open and recover a finding, its thresholds and its scope, with a preview of what it would select. Leader memory and Worker machine memory are off by default because the memory the operating system reports as used includes its file cache.

Notifications and indexed events



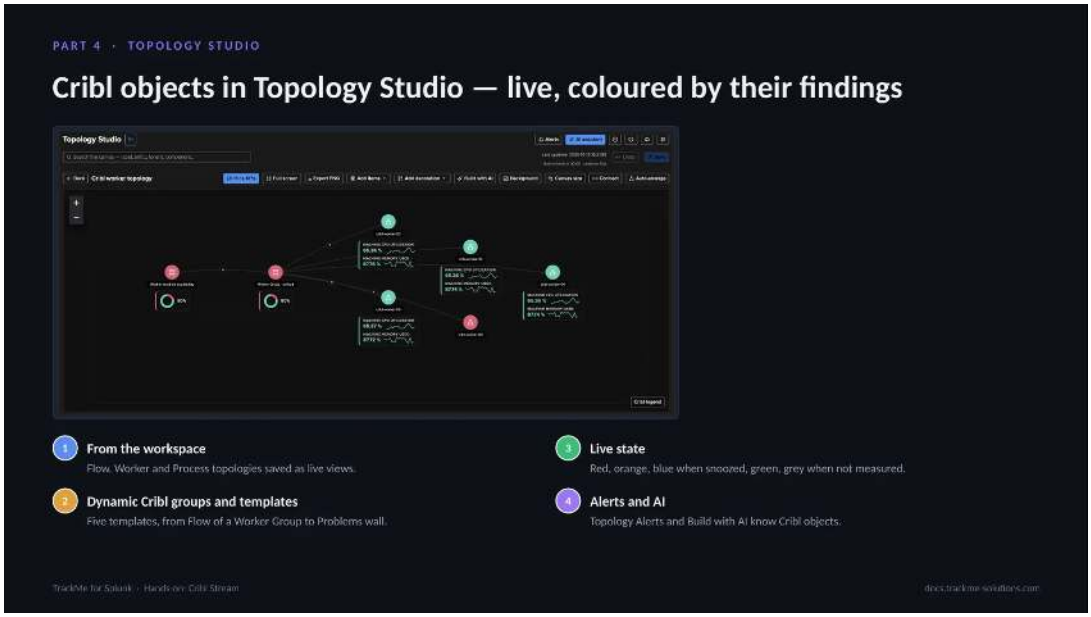
Every transition of a finding or Finding group is written to the notable index, whatever the notification settings. Email notifications are off by default; a TrackMe administrator of the tenant chooses the events, the minimum severity, the policies, the delivery account and up to 50 recipients, and the digest or per-group mode.

The Cribl Stream AI Agent



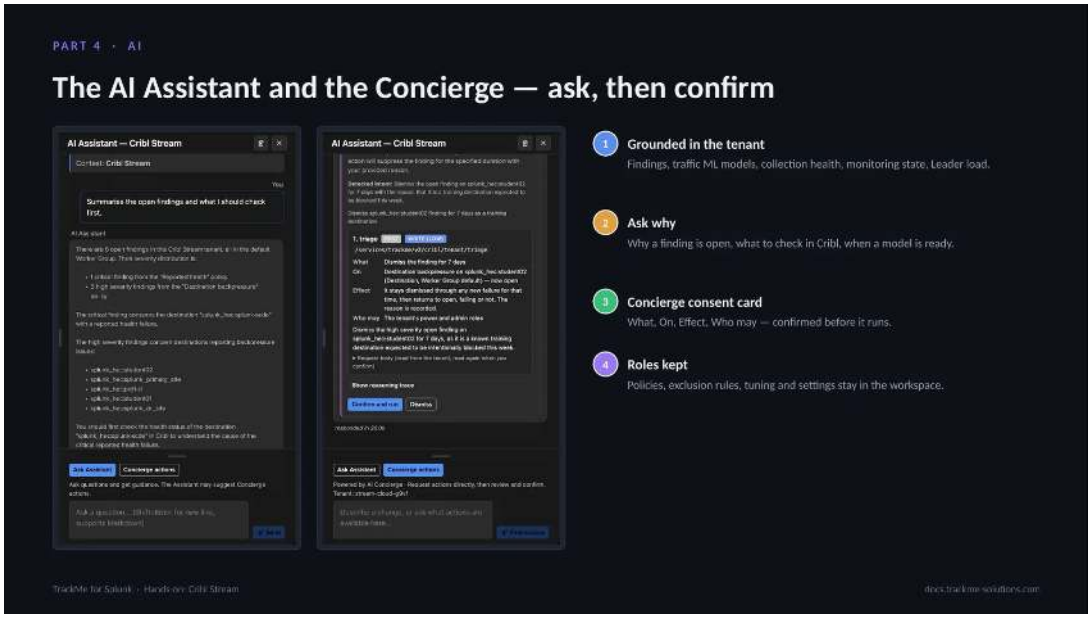
The Cribl Stream AI Agent reviews findings from facts TrackMe checks: when the object was first seen, whether it is disabled in Cribl or gone from the Inventory, which Routes reference it, its traffic and its model. It recommends a decision and keeps one current AI Agent note. Act applies by default only Mark investigating, Dismiss for a time and Not an anomaly; Resolve and Stop monitoring need a proof.

Cribl objects in Topology Studio



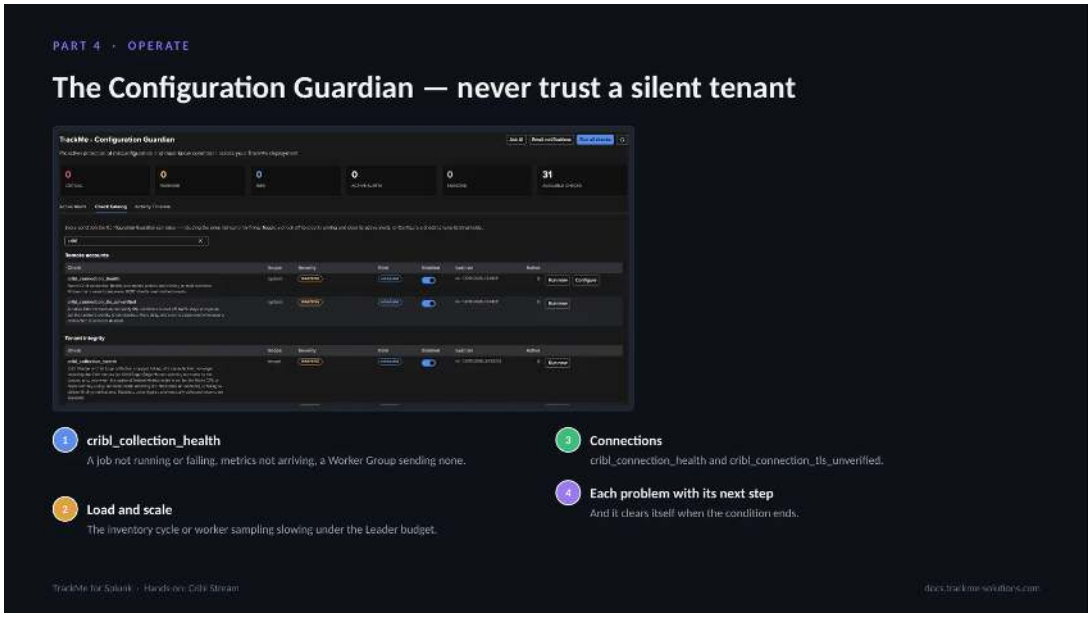
Cribl objects are first-class citizens of Topology Studio: Sources, Destinations, Pipelines, Routes, Workers and worker processes, dynamic Cribl groups that follow a selection, and five templates. Nodes carry their KPIs and 24-hour sparklines, and their colour follows their findings. From the workspace, the Flow, Worker and Process topologies are saved as live views. In the Studio, Add entities and dynamic Cribl groups pick objects, and Topology Alerts and Build with AI know Cribl objects.

AI Assistant and Concierge



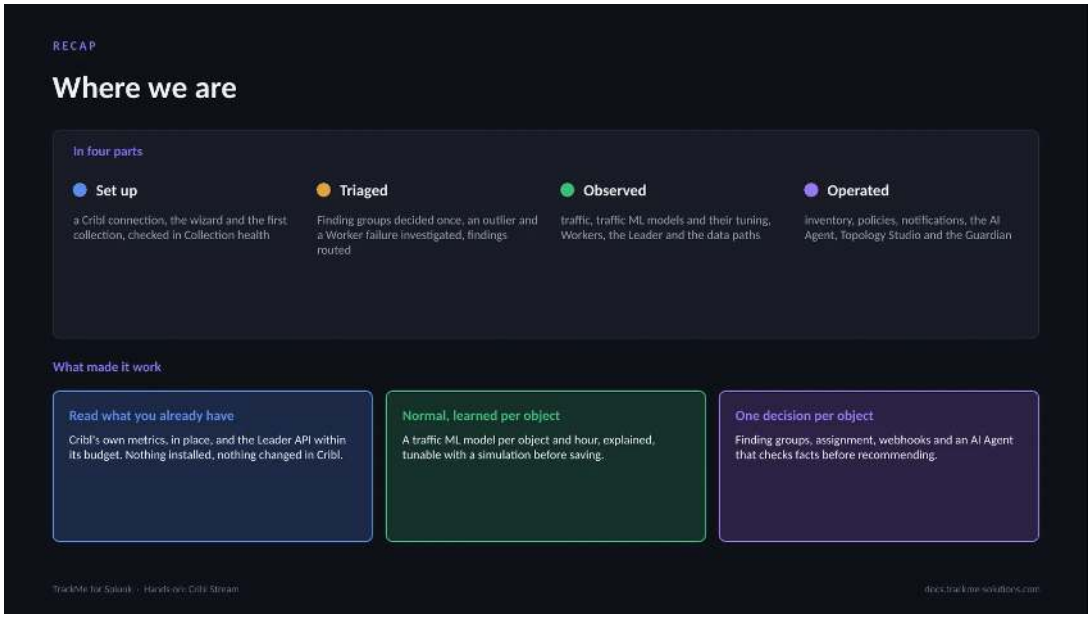
The AI Assistant of a Cribl Stream tenant is grounded in its findings, models, collection health, monitoring state and Leader load. When you ask for a change it can make, it proposes an AI Concierge consent card that you confirm before anything runs: a decision on a finding, Not an anomaly, monitoring on or off for named objects, a collection job, or relearning a model. Each action keeps the role it requires.

Collection health and the Guardian



Collection health keeps you from trusting a silent tenant. The Configuration Guardian check cribl_collection_health raises a warning when a job is not running or keeps failing, when metrics stop arriving, when the collection slows down under the Leader budget, when many objects stop together, or when notification emails fail. The Guardian also checks the Cribl connections themselves.

Where we are



A tenant created on a Cribl connection, its findings grouped per object and triaged, a traffic outlier investigated against its learned range, a Worker failure seen in the finding and the topology, findings routed to a team and to a webhook, the deployment observed from its traffic to its machines and data paths, and the tenant operated with policies, notifications, the AI Agent, Topology Studio and the Guardian.

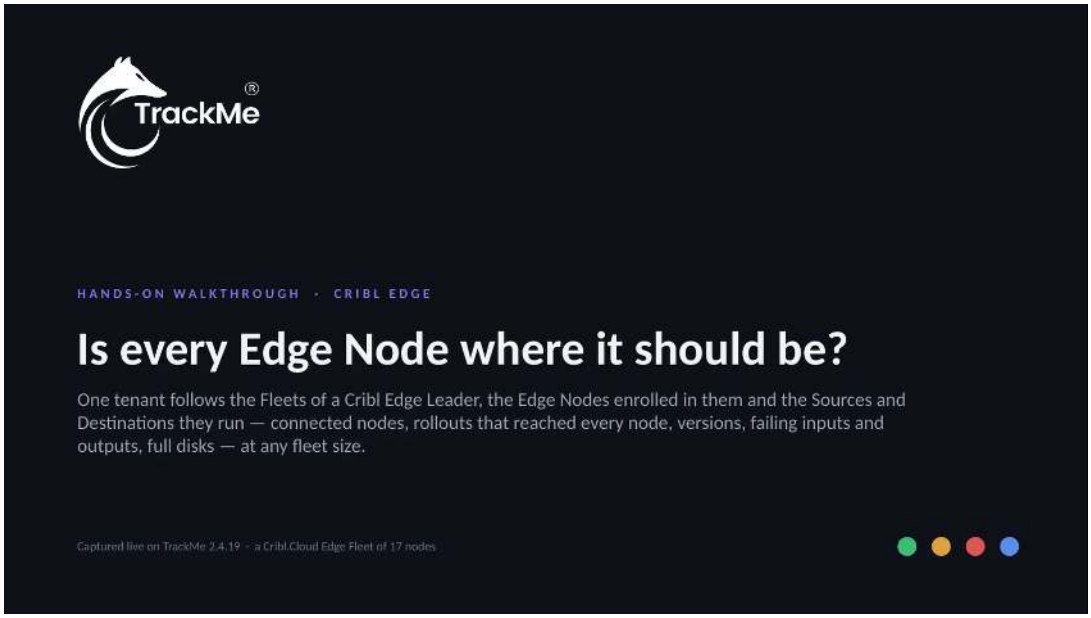
2.1.10 Cribl Edge

Follow the Fleets of a Cribl Edge Leader, the Edge Nodes enrolled in them and the Sources and Destinations they run, from one bulk read of the Leader at any fleet size: connected nodes, rollouts and version drift, failing inputs and outputs, any node opened up close with its System activity read live, the nodes that must stay connected selected from Splunk lookups, bank holidays per policy, and the Cribl Edge AI Agent with its custom instructions.

`Download the deck (PDF) · PowerPoint`

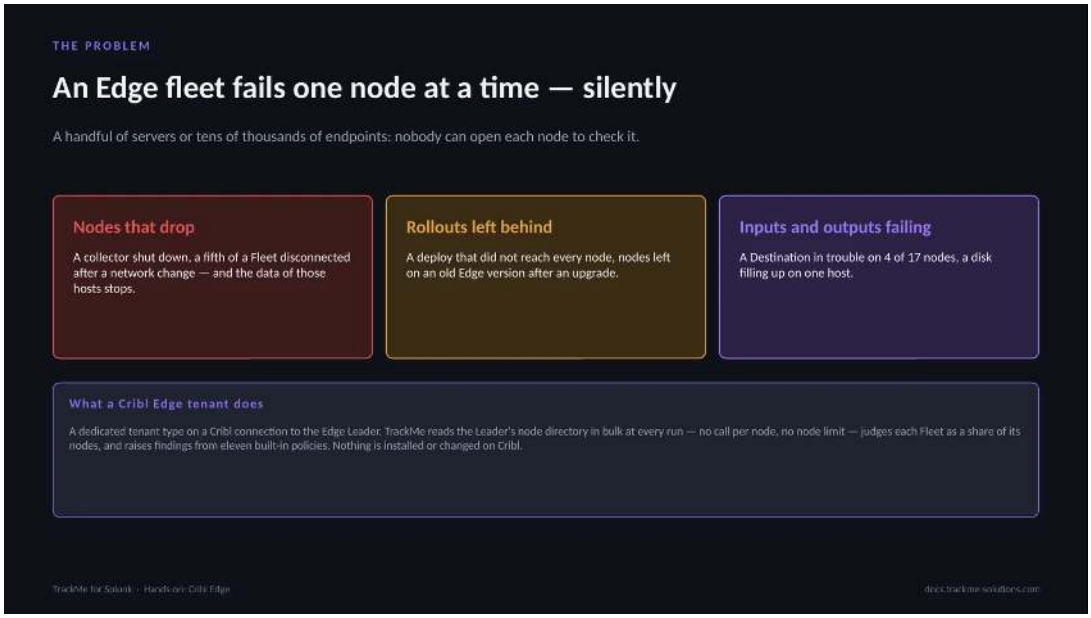
about 30 minutes · 30 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



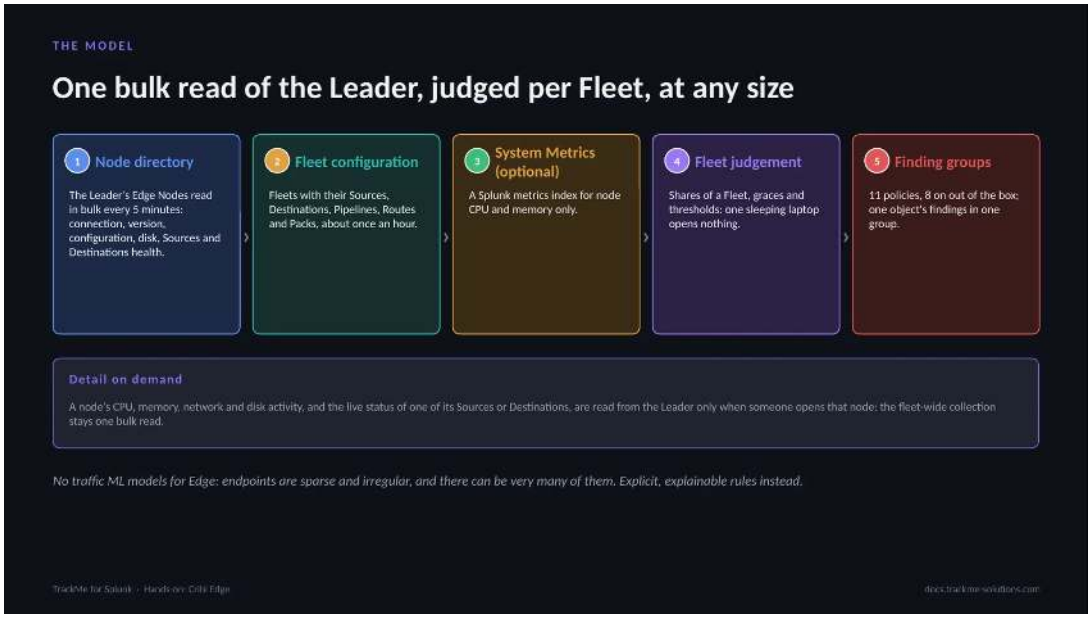
This walkthrough covers the Cribl Edge tenant, generally available in TrackMe 2.4.19. Everything was captured live on a Cribl.Cloud Edge Fleet of 17 nodes. In four parts: create the tenant, follow the Fleets and the Edge Nodes, read the findings, and operate the tenant.

Why a dedicated Cribl Edge tenant



An Edge operator asks the same questions every day: are my nodes connected, did my last deploy reach every node, is every node on the same Edge version, which Source or Destination is failing and on how many nodes, which host is running out of disk. The Cribl Edge tenant answers them with findings raised by built-in policies, with the same triage as the Cribl Stream tenant.

How a Cribl Edge tenant works



Every five minutes TrackMe reads the Leader's list of Edge Nodes page by page and keeps it as one published node directory: no call per node, no rotation, every node current within one interval. The Fleet configuration is read about once an hour. Policies judge each Fleet as a share of its nodes, select nodes that must never drop, and watch disk, and CPU and memory when a System Metrics index is set. There is no machine learning for Edge.

The parts of the walkthrough



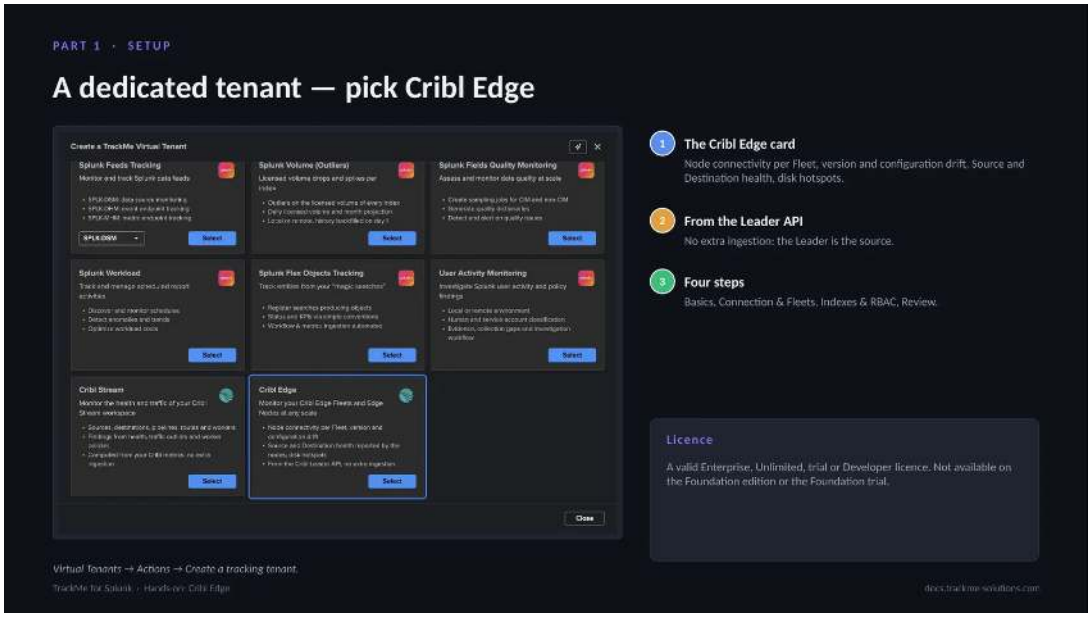
Four parts. Setup: the wizard and the workspace. Fleets and nodes: the Fleets tab, a Fleet, the Edge Nodes and one node up close. Findings: the backlog, a disconnected node, the nodes that must stay connected and version drift. Operate: policies and bank holidays, collection health, settings, Topology Studio, the AI Assistant and the Cribl Edge AI Agent.

Part 1 — Setup



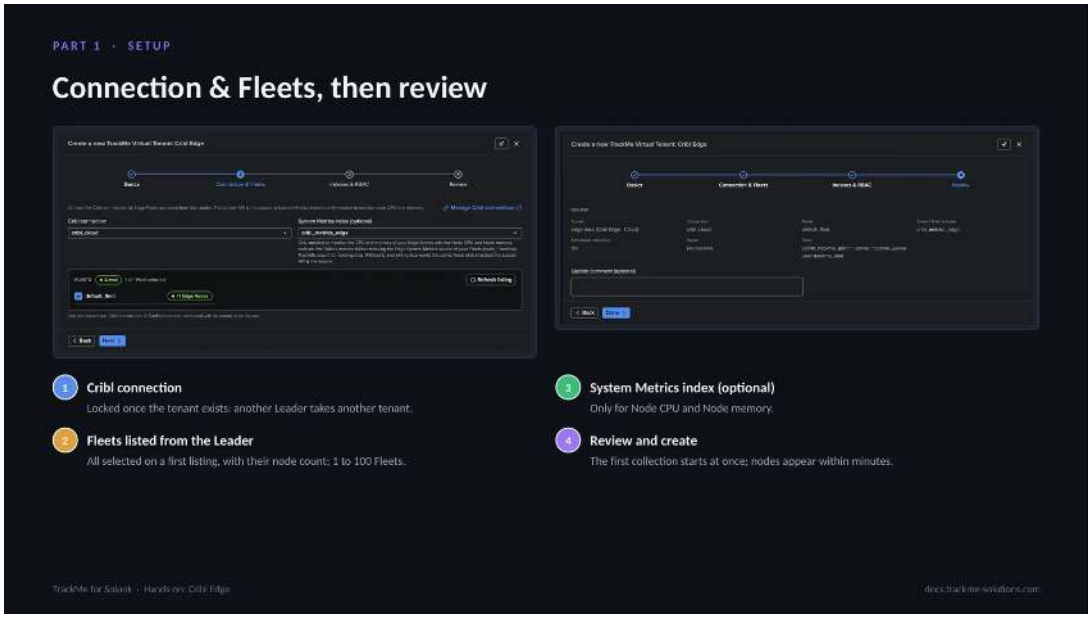
A Cribl Edge tenant uses the same Cribl connections as Cribl Stream; a Cribl.Cloud workspace running both products takes one tenant of each on the same connection. The wizard has four steps: Basics, Connection & Fleets, Indexes & RBAC and Review. The first collection starts at once, and the Fleets and Edge Nodes appear in the workspace within a few minutes.

Choosing the Cribl Edge card



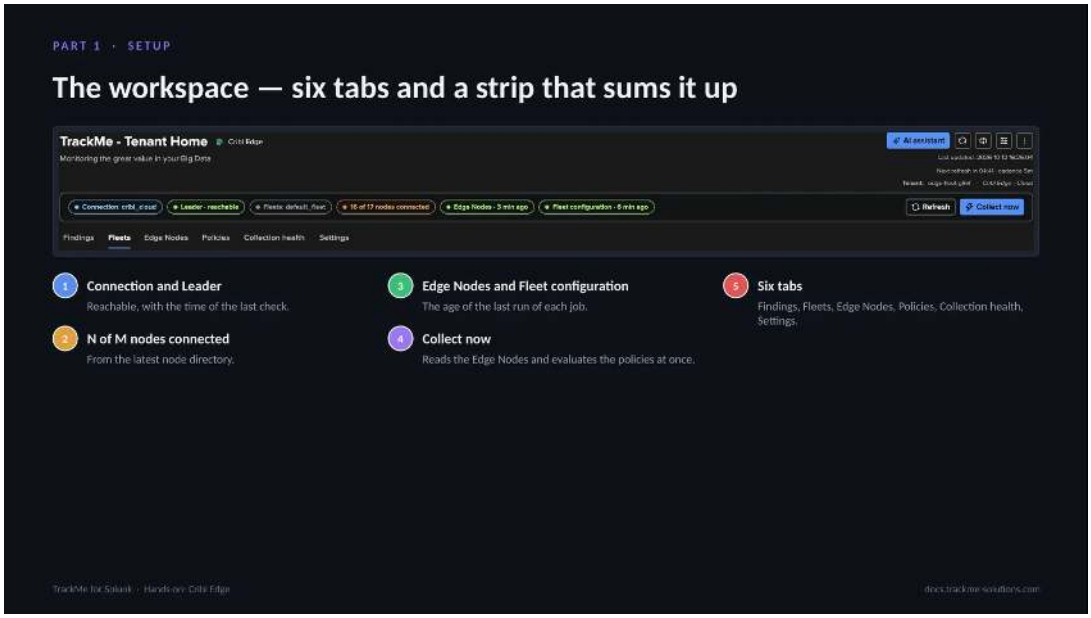
Virtual Tenants, Actions, Create a tracking tenant, Cribl Edge card. A tenant is a Cribl Edge tenant or a Cribl Stream tenant, never both. Basics is the identifier, alias and description. The card summarises what the tenant follows: node connectivity per Fleet, version and configuration drift, Source and Destination health reported by the nodes, and disk hotspots, all from the Cribl Leader API without extra ingestion.

Connection, Fleets and review



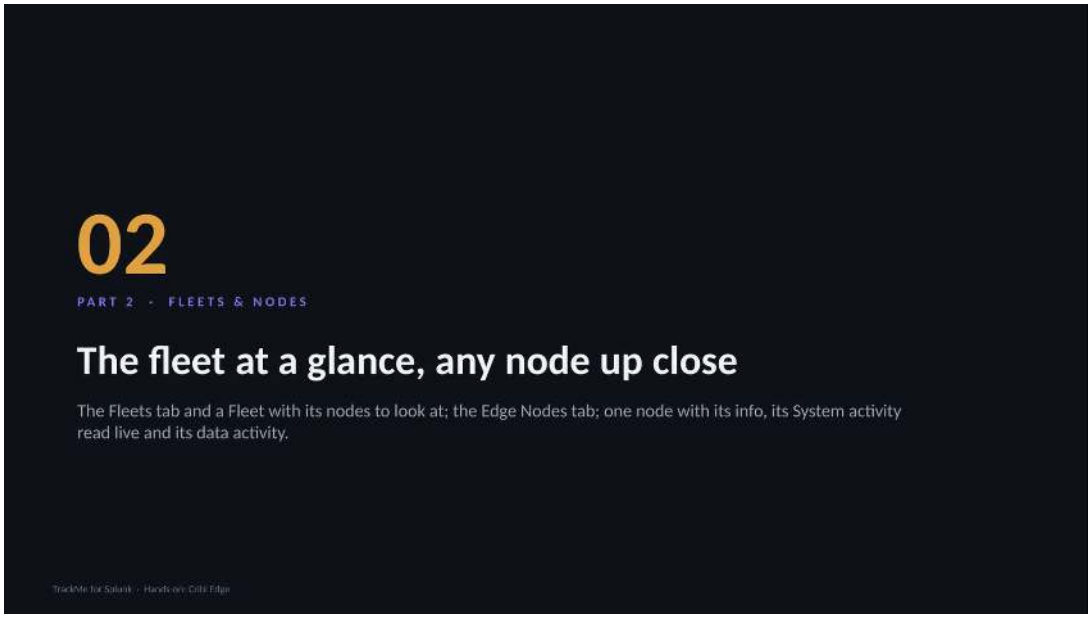
Choose the connection; TrackMe lists the Fleets from the Leader, each with its number of nodes, and a Subfleet can be monitored with its parent or on its own. The System Metrics index is optional: without it everything else works the same, node disk included. When the Leader cannot list the Fleets, the step says why and what to do.

The workspace and its status strip



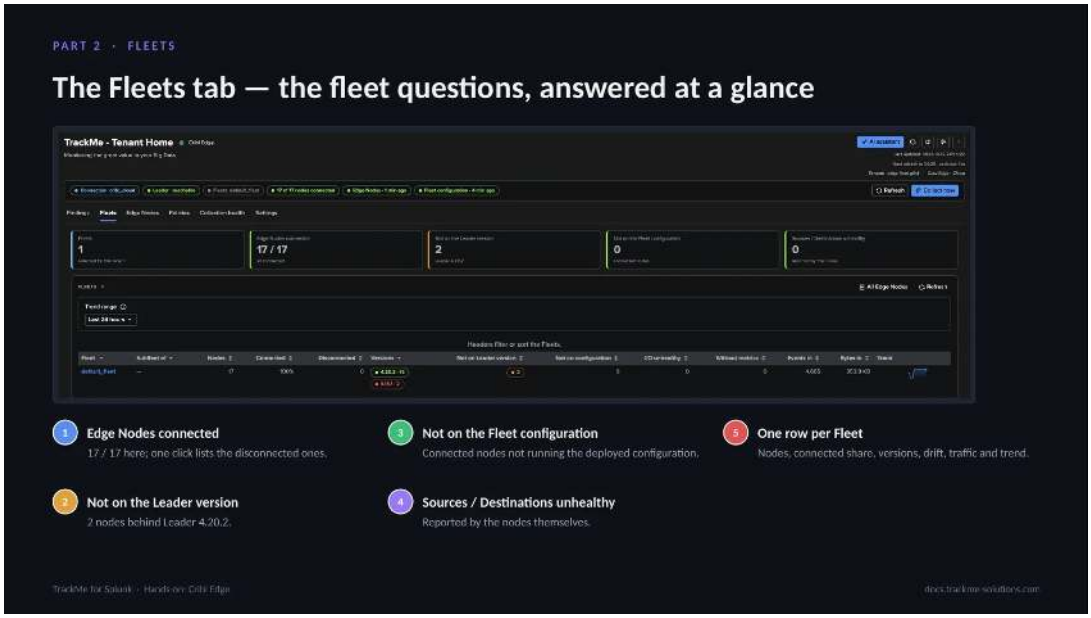
The Cribl Edge workspace is built with the same components as the Cribl Stream workspace, so the two look and behave the same. The status strip sums the tenant up; Collect now is for TrackMe administrators of the tenant. The six tabs are Findings, Fleets, Edge Nodes, Policies, Collection health and Settings. Every table is sorted, filtered and paged by the server.

Part 2 — Fleets and nodes



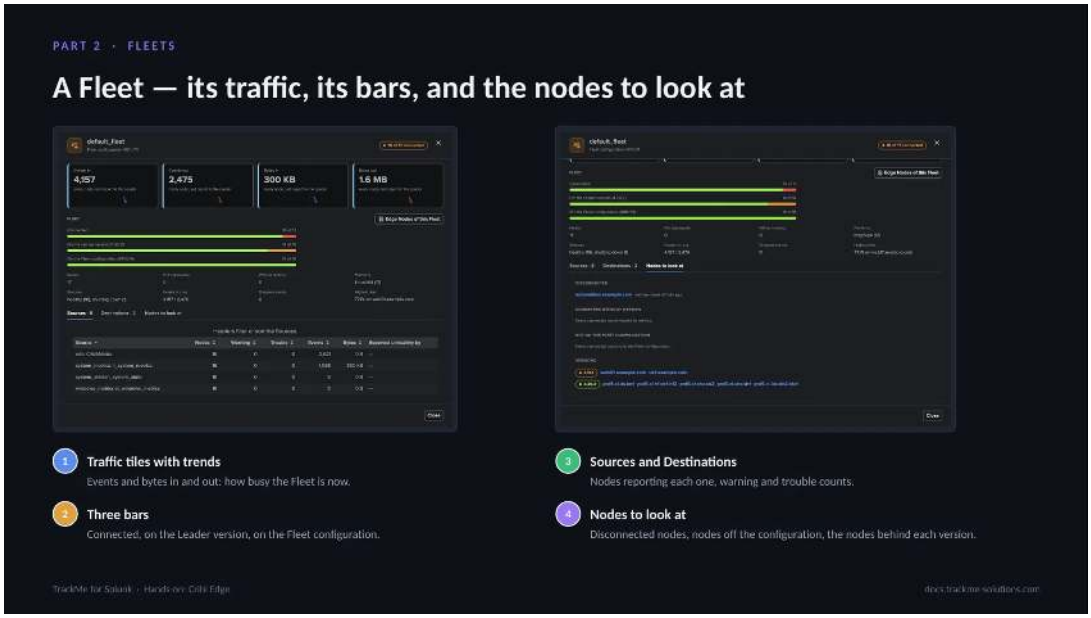
The Fleets tab answers the fleet questions at a glance; a Fleet opens with its traffic, its bars and the nodes to look at; the Edge Nodes tab lists every node; and a node opens with its Node info, its System activity read live and its Data activity.

The fleet at a glance



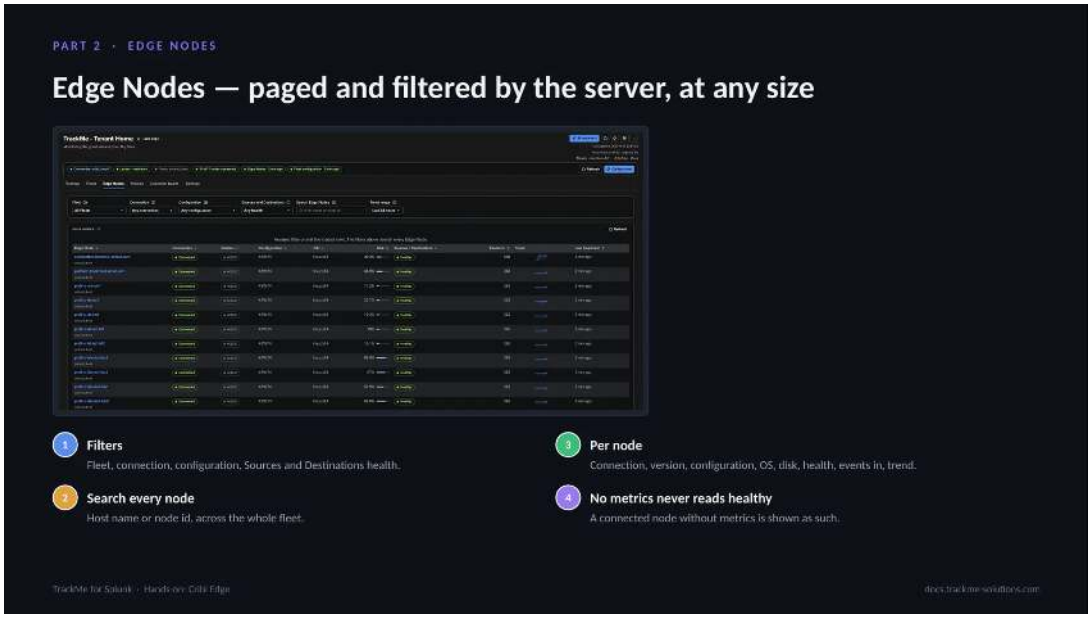
The tiles answer the fleet questions; the table has one row per Fleet with its Subfleet parent, connected share, versions running — green on the Leader version, amber otherwise, red when newer than the Leader or two minor versions behind — drift counts, unhealthy Sources and Destinations, traffic and trend. Versions are coloured green on the Leader version, amber otherwise, and red when newer than the Leader or two minor versions behind.

Inside a Fleet



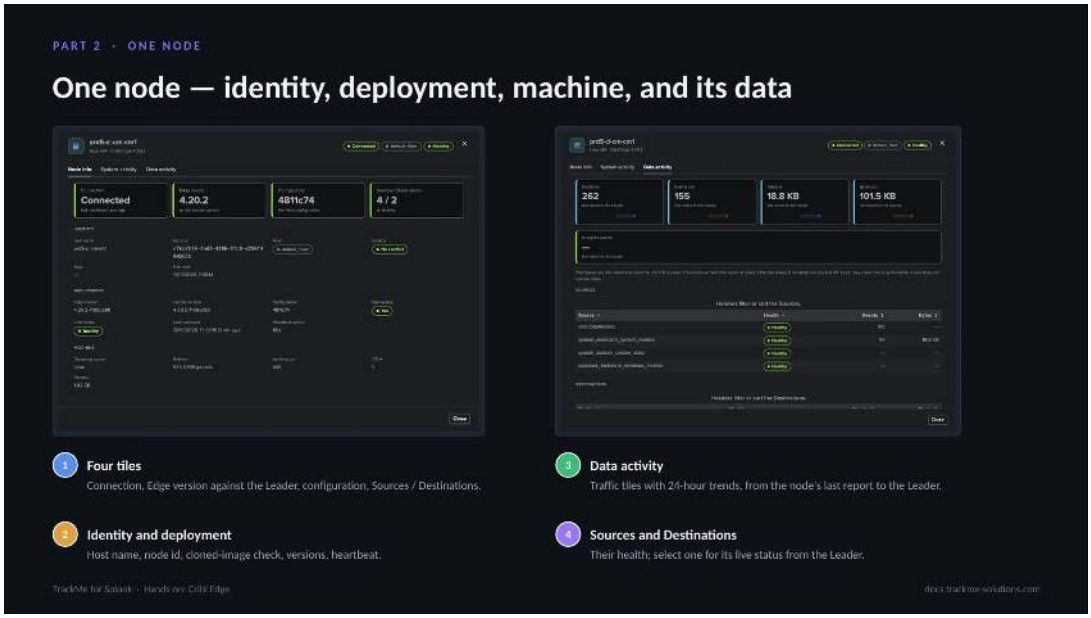
Selecting a Fleet opens it: traffic tiles, the connected, Leader-version and configuration bars, its facts, and three tabs. Sources and Destinations per object with the nodes reporting it unhealthy; Nodes to look at lists the disconnected nodes with their last heartbeat and the nodes behind each version. Every node name opens the node.

Every Edge Node, paged by the server



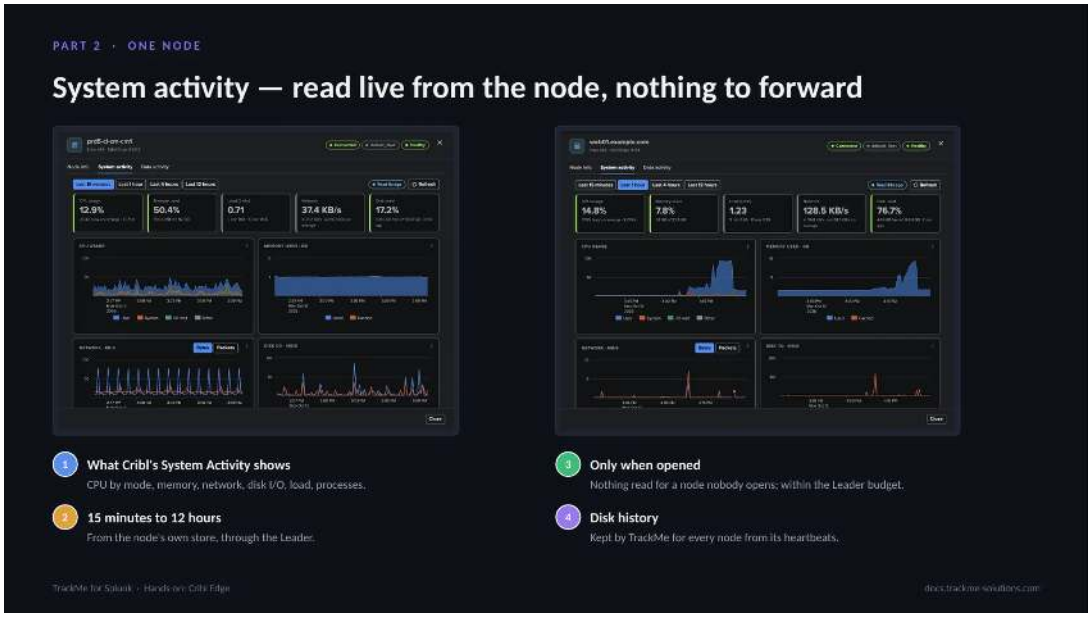
Every Edge Node of the selected Fleets, paged and filtered by the server so the tab stays fast with any number of nodes. A node is disconnected when the Leader says so or when it has sent no heartbeat for five periods, at least 10 minutes. Filters for Fleet, connection, configuration and Sources and Destinations health apply to every node, and the search finds a host name or node id across the whole fleet.

One node: info and data activity



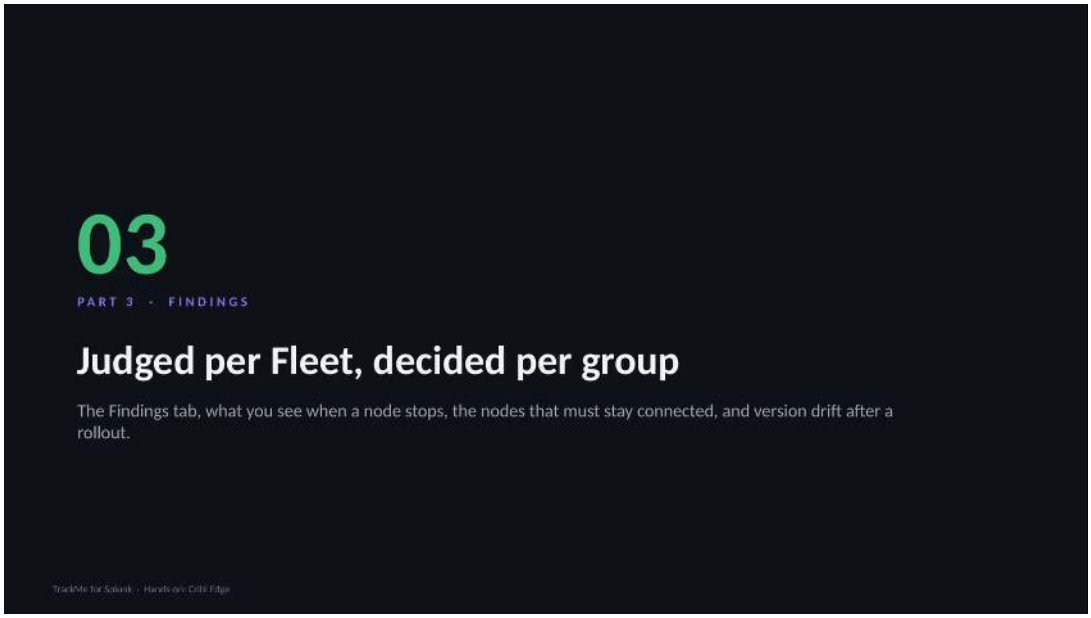
A node opens on three tabs. Node info: connection, Edge version compared with the Leader's, configuration compared with the Fleet's target, and Identity, Deployment and Machine facts. Data activity: the node's traffic with trends and its Sources and Destinations; selecting a health reads that object's live status on that node, as Cribl's Status tab shows it.

System activity read live from the node



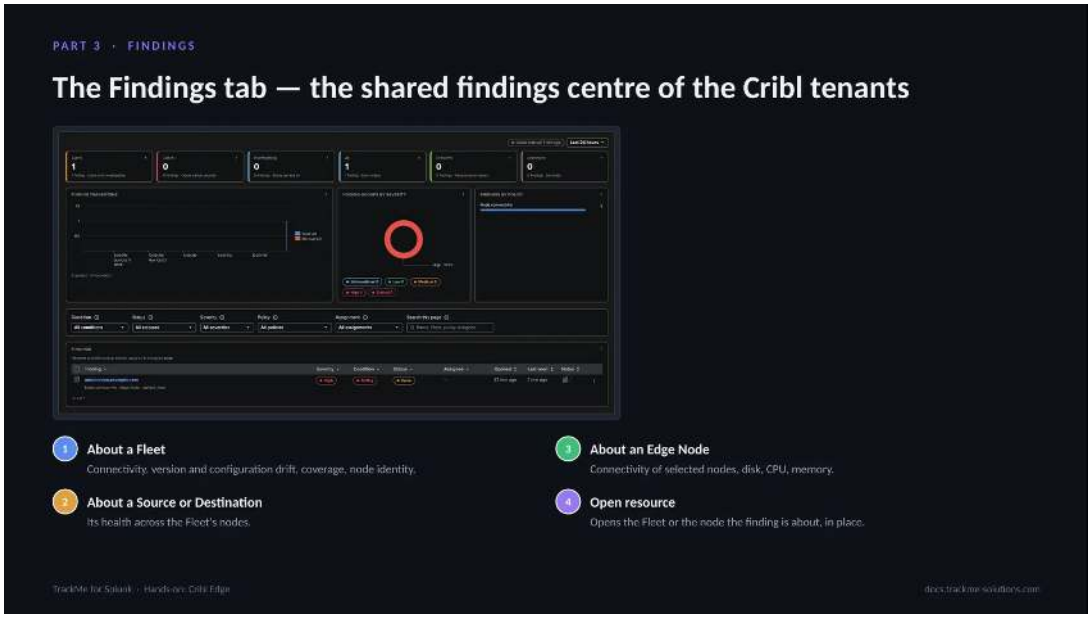
System activity shows what Cribl's own System Activity page shows for the node, read live through the Leader from the node's own store of its system metrics. It does not use the System Metrics index and needs nothing forwarded to Splunk. Nothing is read for a node nobody opens, so the fleet-wide collection stays one bulk read.

Part 3 — Findings



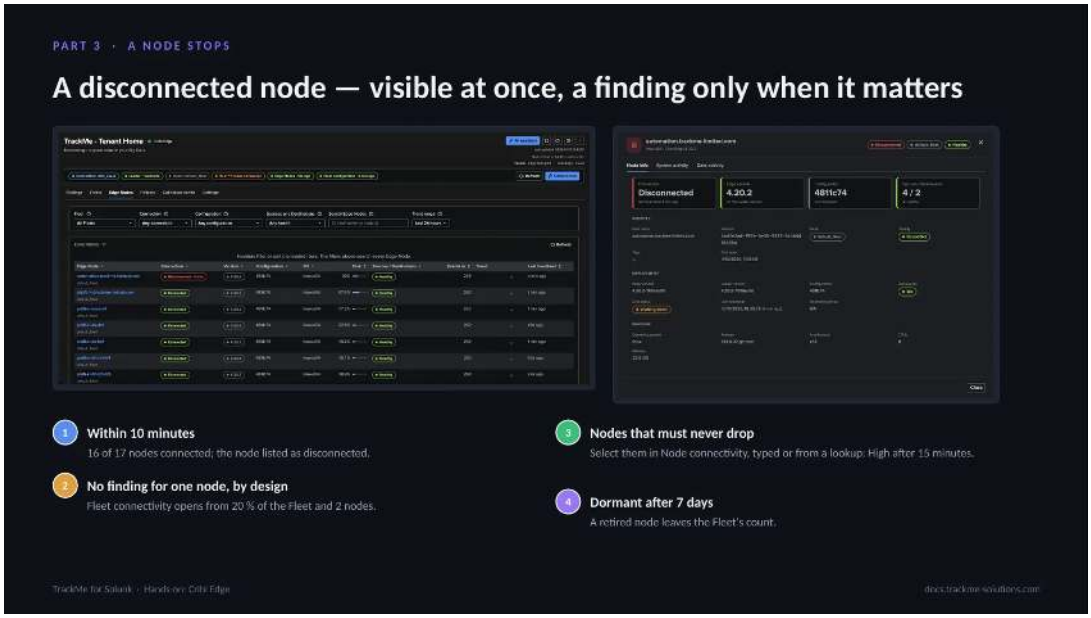
Findings work exactly as on Cribl Stream: Finding groups, triage, notifications, assignment and findings actions. A finding is about a Fleet, a Source or Destination of a Fleet, an Edge Node or the Leader. Each object gathers its findings in one automatic Finding group, the unit that is triaged and notified.

The findings backlog



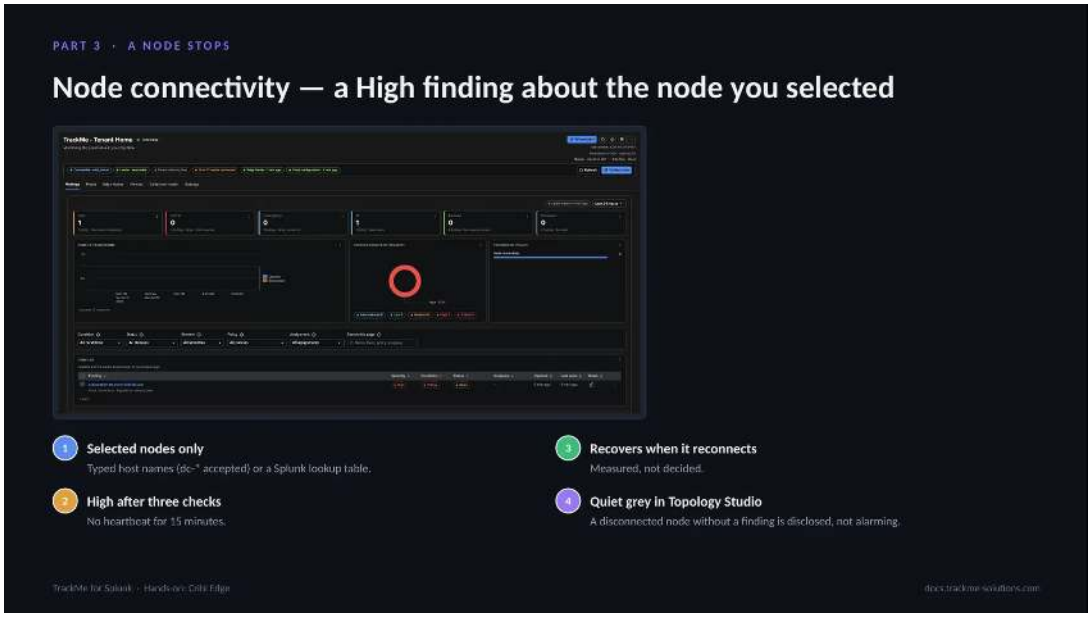
A Cribl Edge finding is about one object, one policy and one measurement: a Fleet, a Source or Destination of a Fleet, an Edge Node, or the Leader. Each object gathers its findings in one automatic Finding group. A finding recovers by itself when the condition clears. Edge has no Stop monitoring and no Not an anomaly: narrow the policy's scope instead.

When a node disconnects



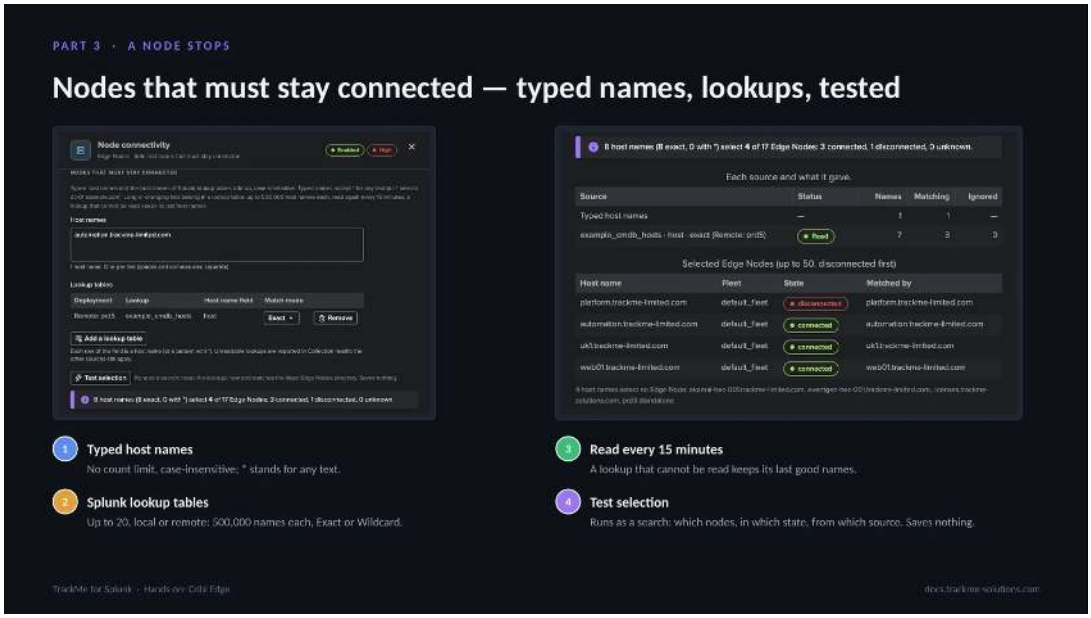
Take a Fleet of 17 nodes, one of which stops. Within 10 minutes it is disconnected in the next directory. No finding opens for that one node by default: Fleet connectivity opens when at least 20 percent of the active nodes and at least two are disconnected. For servers and collectors that must never drop, select them in Node connectivity, by host name or from a Splunk lookup table.

Nodes that must never drop



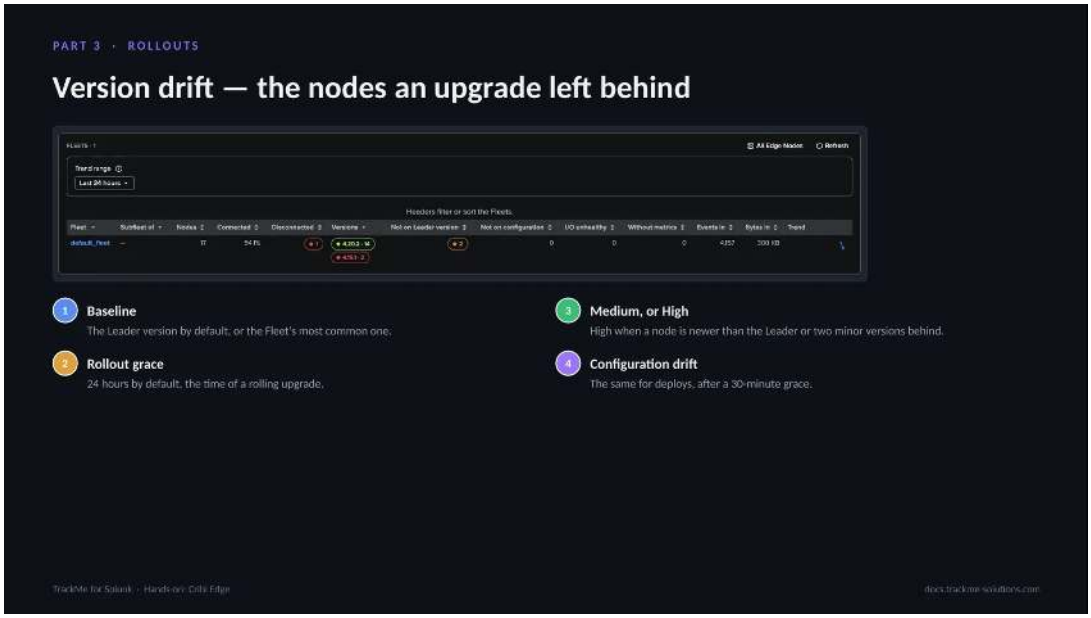
A node selected in Node connectivity opens a High finding about that node after three checks without a heartbeat, and recovers when it reconnects. In Topology Studio a disconnected node without a finding is drawn quiet grey and does not count in a group's healthy share. The Findings tab lists it with the host name it was selected by; its Finding group is triaged, notified and assigned like every other.

Nodes that must stay connected



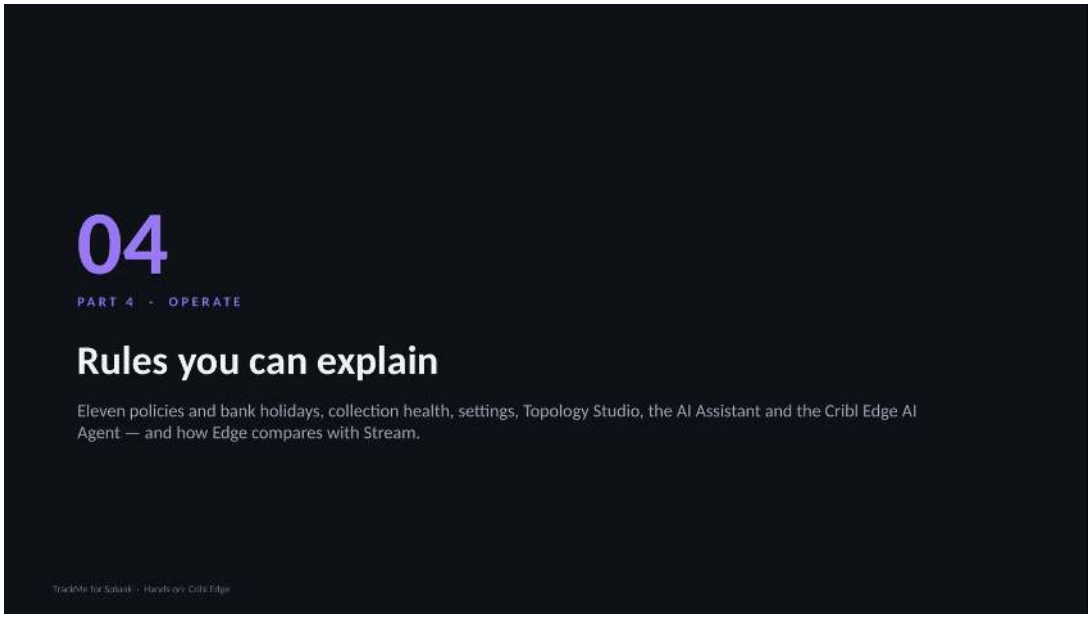
Node connectivity selects the nodes that must stay connected from two sources that add up. Typed host names have no count limit and accept a star for any text. Long or changing lists belong in Splunk lookup tables, such as a CMDB extract, on the local search tier or a remote deployment: up to twenty per policy and 500,000 host names each, matched exactly or with wildcards, read again every 15 minutes. A lookup that cannot be read keeps its last good names, and Collection health says so. Test selection runs as a search and shows which nodes the names select, in which state and from which source, before anything is saved.

Version drift after a rollout



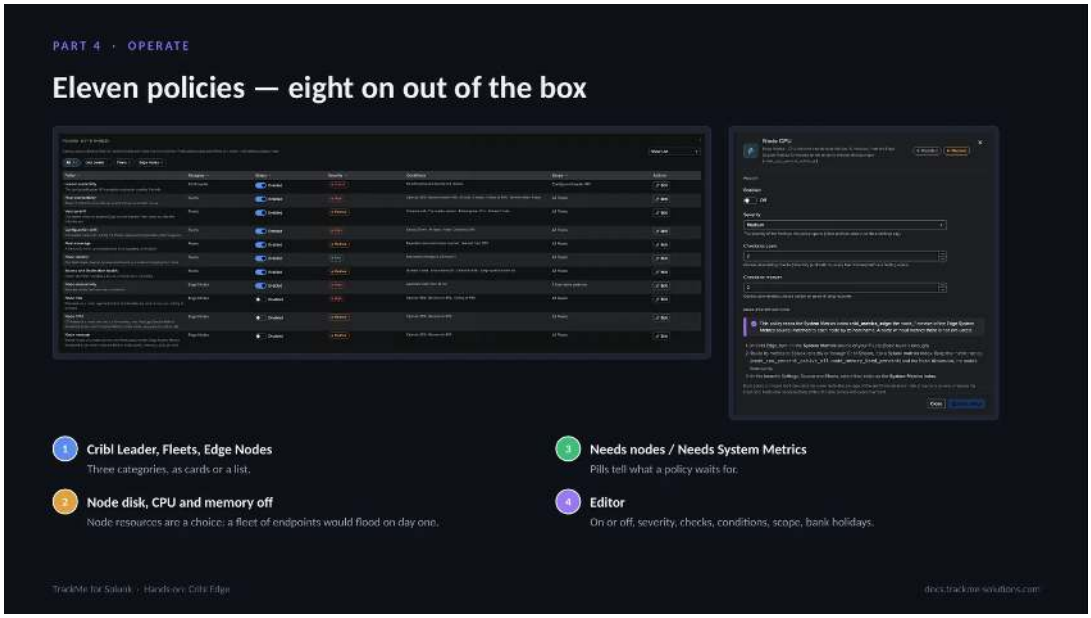
Version drift compares every connected node with a baseline. Once nodes stay off it for the rollout grace, a finding opens on the Fleet: Medium, raised to High for the cases Cribl does not support. Configuration drift works the same way for deployments. The Fleet's Nodes to look at names the nodes behind each version.

Part 4 — Operate



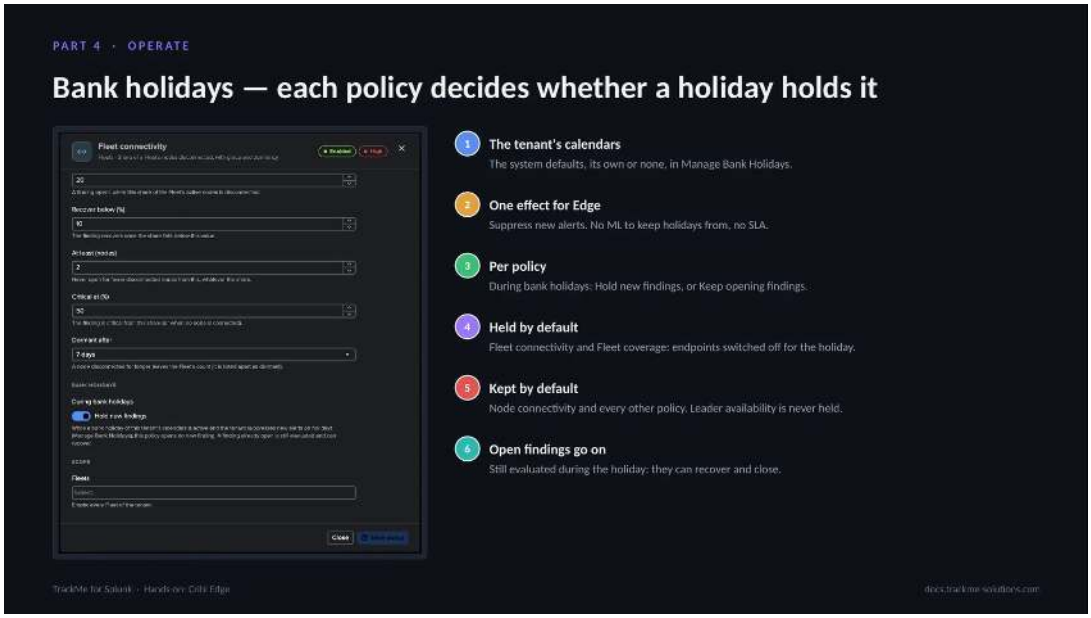
Eleven built-in policies, eight on out of the box, two collection jobs with the Leader request budget, one Save for every setting, Edge objects and groups in Topology Studio, the AI Assistant, and the Cribl Edge AI Agent with its custom instructions. The comparison with Cribl Stream closes the part.

The policy catalogue



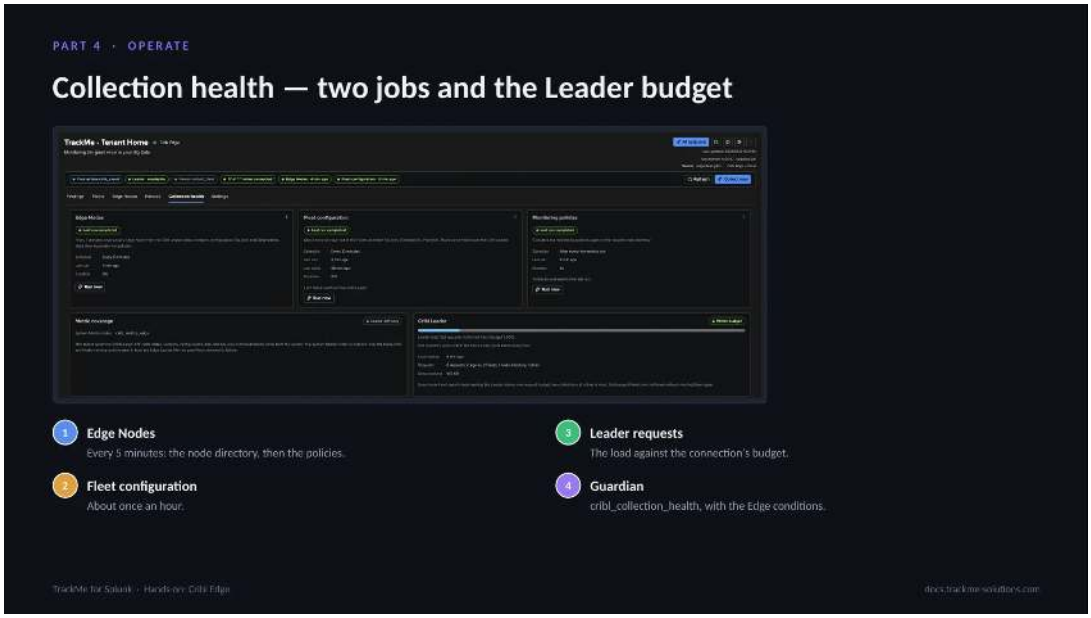
The catalogue of eleven policies in three categories. Two pills tell what a policy is waiting for: Needs nodes on Node connectivity until you select the nodes, and Needs System Metrics on Node CPU and Node memory until the tenant has a System Metrics index. Node disk, CPU and memory are off by default.

Bank holidays, per policy



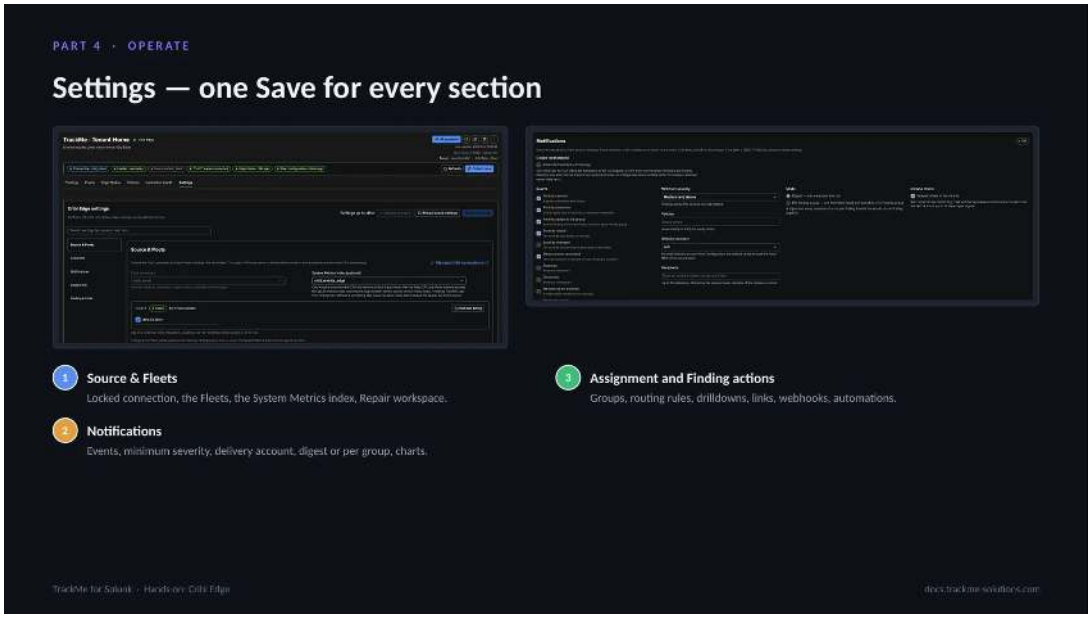
A Cribl Edge tenant uses the bank holiday calendars like every tenant: the system defaults, its own selection, or none. One effect applies to Edge, Suppress new alerts: while a holiday is active, the policies set to hold open no new finding. Each policy decides in its editor. By default Fleet connectivity and Fleet coverage hold, because laptops and desktops switched off for the holiday would otherwise open findings; Node connectivity and every other policy keep opening findings, and Leader availability is always judged. A finding already open is still evaluated and can recover.

Collection health and the Leader budget



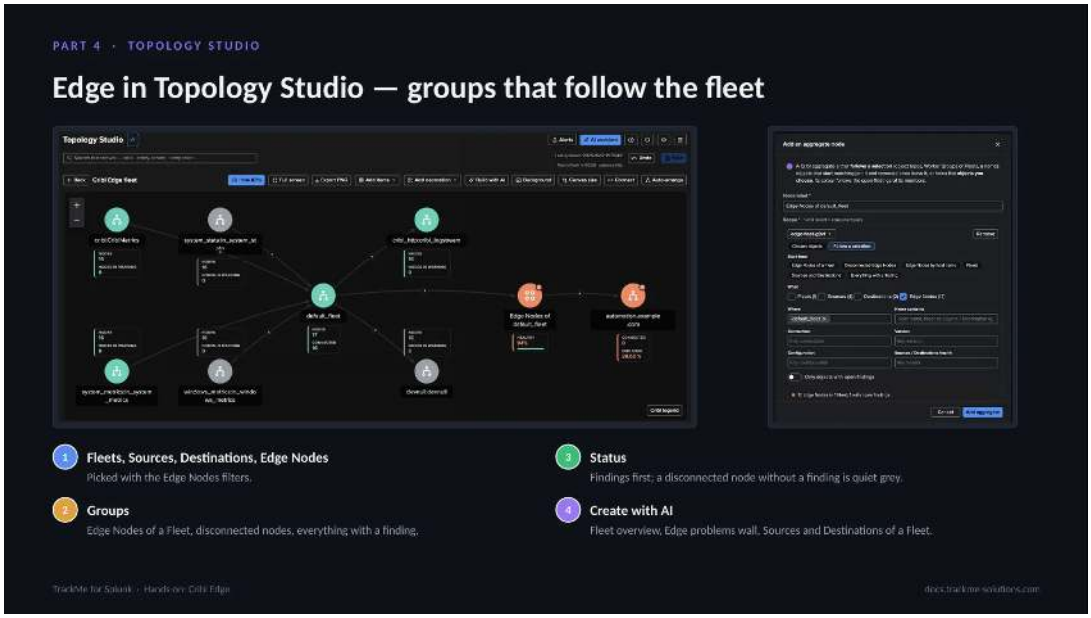
One card per job with its last run, duration, next run and Run now, the Leader requests against the budget and the System Metrics index. When collection does not run as configured, the tab lists the problems and the Configuration Guardian raises Cribl Edge collection needs attention, including the Edge conditions: many nodes without metrics, or no System Metrics arriving.

Settings and notifications



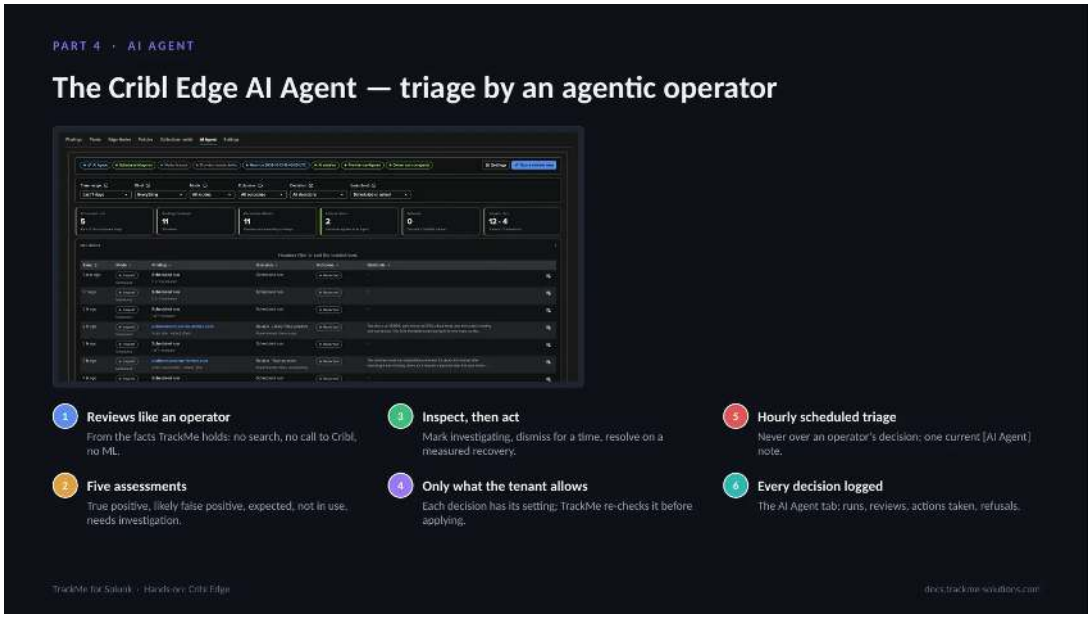
Settings gathers every section with one Save. Changing the Fleets starts a new source and closes the findings of the previous one; the System Metrics index can change at any time. Notifications work as for Cribl Stream, and every transition is also an indexed event.

Edge in Topology Studio



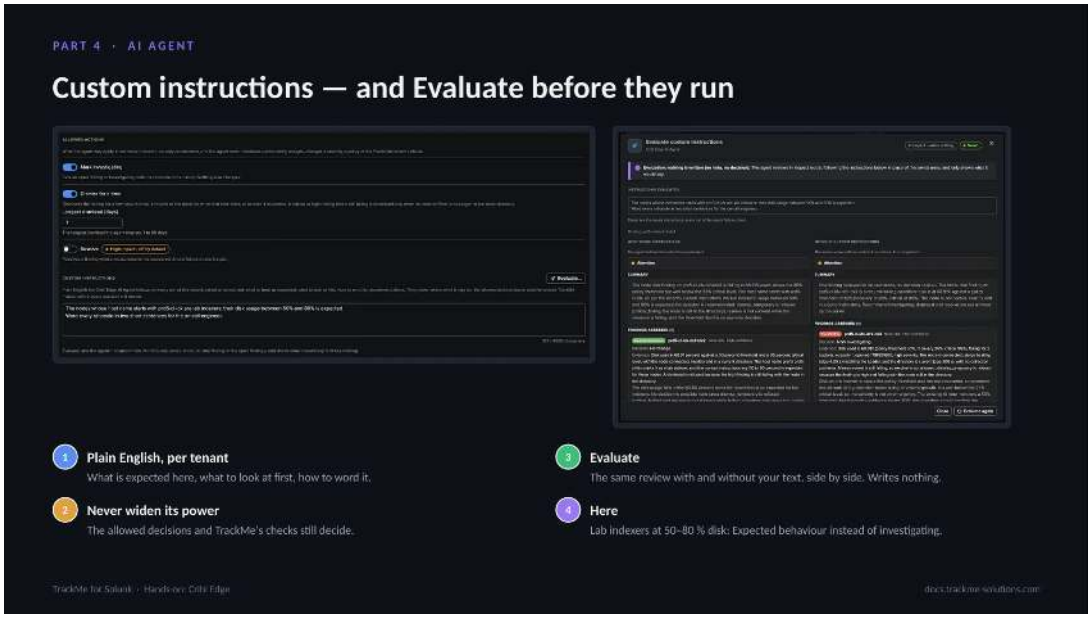
Cribl Edge objects are Topology Studio objects. Groups are the recommended way to show nodes: they follow their nodes as they come and go, at any fleet size, and show their health and healthy share. Topology Alerts work on Edge objects and groups. A view built from an Edge tenant shows the Fleet, its Sources and Destinations and a group of Edge Nodes with its healthy share; a stopped node is drawn quiet grey. Topology Alerts work on Edge objects and groups.

The Cribl Edge AI Agent



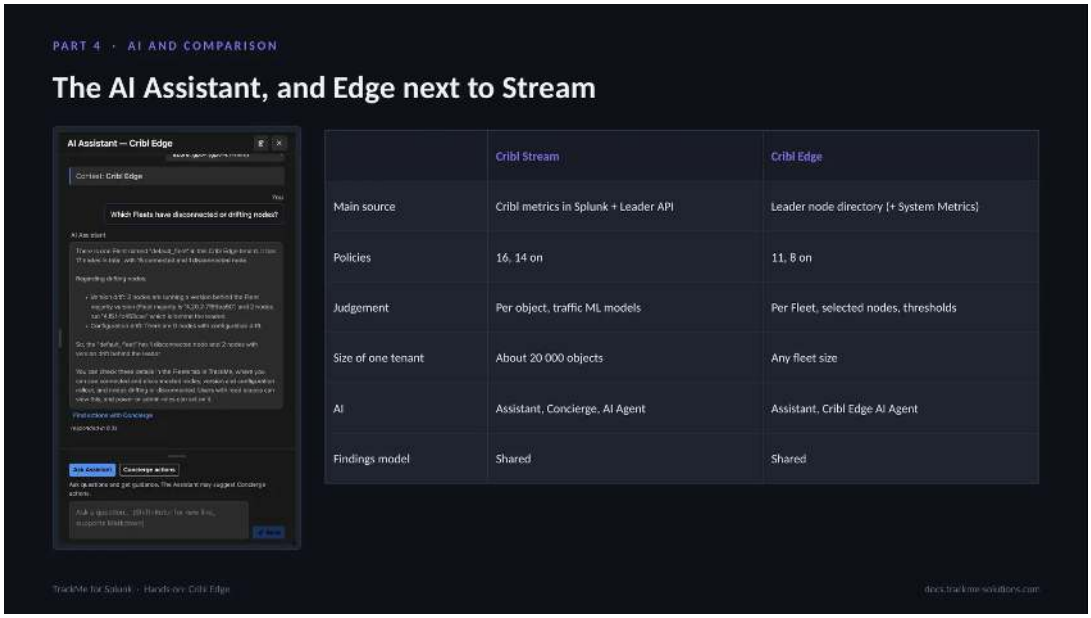
The Cribl Edge AI Agent reviews Edge findings the way an operator would, from facts TrackMe already holds: the finding and its group, the Fleet and the node in the published directory, whether they are still there, and Collection health. It never runs a search, never calls Cribl and uses no machine learning. It assesses each finding and recommends a decision; in act mode it applies only the decisions the tenant allows, and TrackMe checks their conditions again before applying them. Scheduled triage runs hourly and never overrides an operator; the AI Agent tab logs every run and decision.

Custom instructions and Evaluate



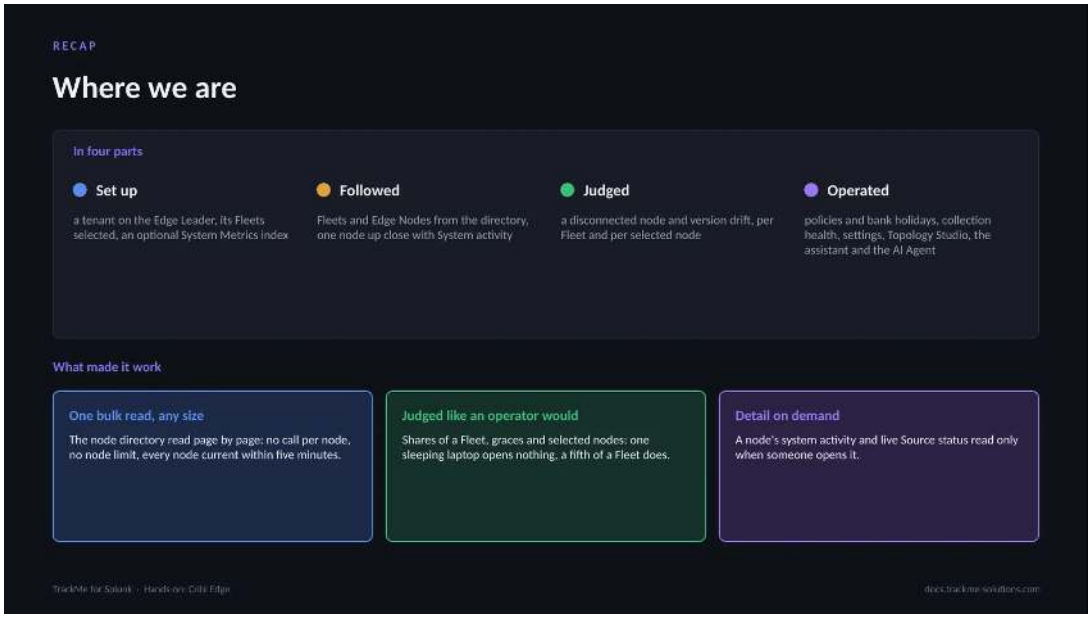
In Settings, AI Agent, a TrackMe administrator of the tenant chooses the decisions the agent may apply and writes custom instructions in plain English: what to treat as expected, what to look at first, how to word the recommendations. Every run follows them, and they never widen what the agent may do. Evaluate runs the agent in inspect mode with the text and shows the same review with and without it, side by side, writing nothing. Here, lab indexers between 50 and 80 percent disk are declared expected: the same Node disk finding reads Expected behaviour instead of a true positive.

The AI Assistant and Edge compared with Stream



On an Edge tenant, the AI Assistant knows the workspace, the policies and their settings, the Fleets, the node summary, the Leader version and the age of the node directory, and reads the open findings. Edge and Stream share the findings model and each has its own AI Agent; they differ in sources, judgement and machine learning.

Where we are



A Cribl Edge tenant on a Cribl connection, its Fleets and nodes followed from one bulk read of the Leader, any node opened up close with its System activity read live, a disconnected node and version drift seen the way the policies judge them, the nodes that must stay connected selected by name or from a lookup, and the tenant operated with its policies and bank holidays, collection health, settings, Topology Studio, the AI Assistant and the Cribl Edge AI Agent.

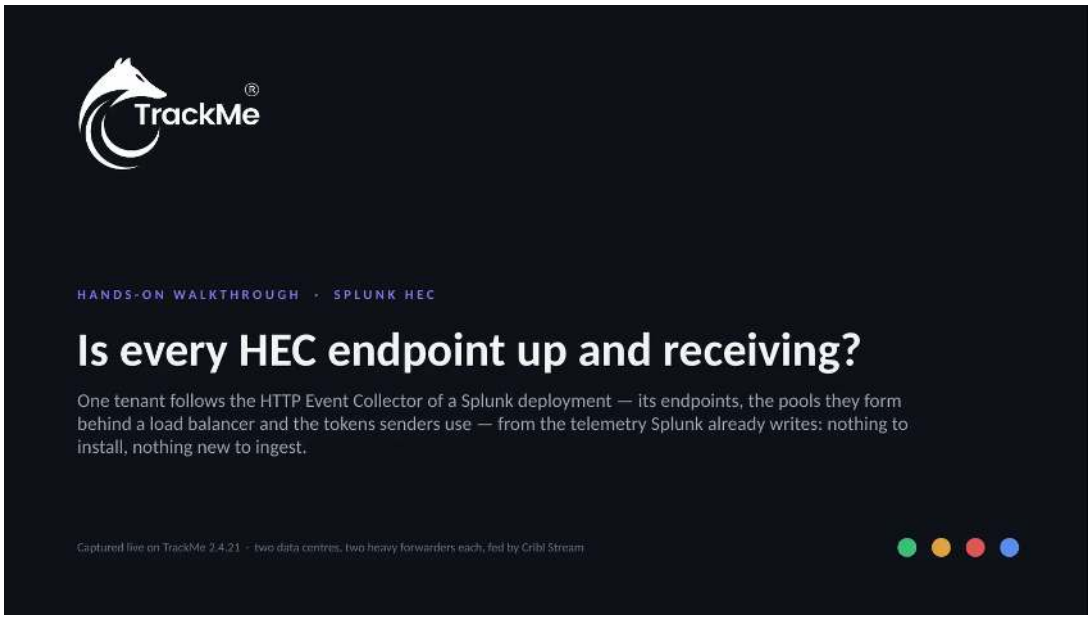
2.1.11 Splunk HEC

Follow the HTTP Event Collector of a Splunk deployment from the telemetry Splunk already writes: endpoints found from their heartbeats, pools of heavy forwarders judged as a whole, tokens known only by name, one endpoint's ingestion pipeline and a pool's traffic distribution; a heavy forwarder stopped on purpose and the two findings it opens, how each is evaluated, the data centres in Topology Studio and the Splunk HEC AI Agent; then the eighteen policies, token silence, the traffic models and their tuning, collection health and the optional splunkd API.

`Download the deck (PDF) · PowerPoint`

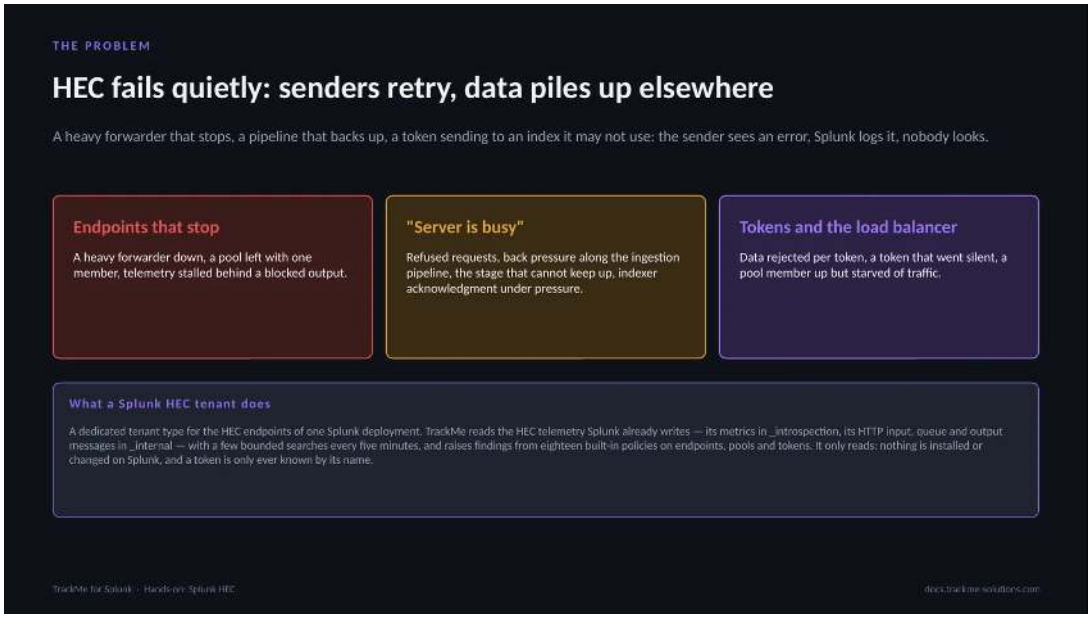
about 30 minutes · 32 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



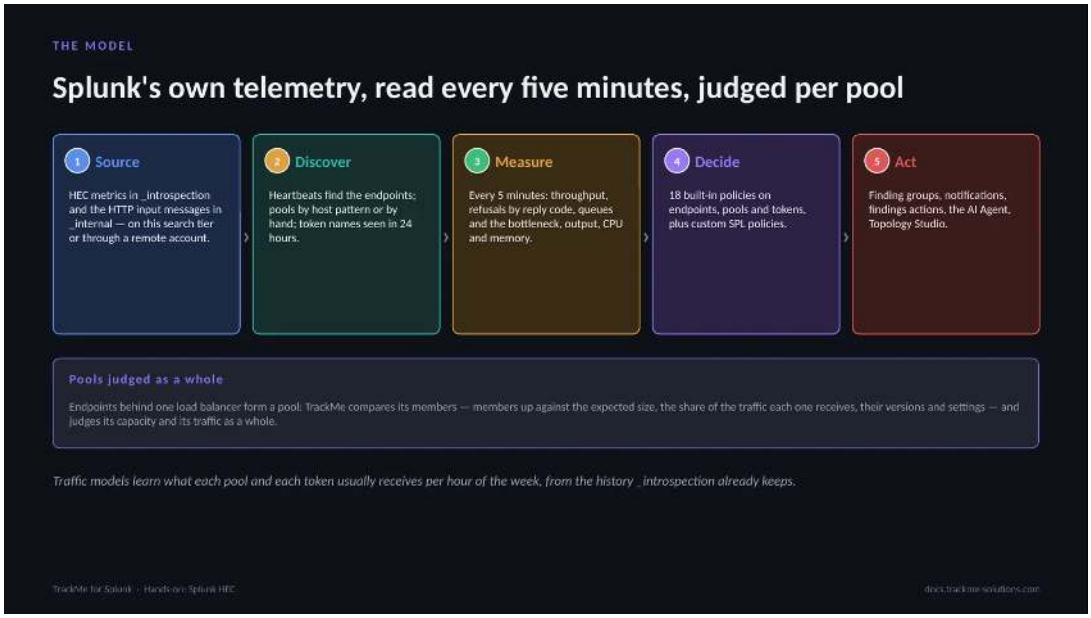
This walkthrough covers the Splunk HEC tenant, generally available in TrackMe 2.4.21. Everything was captured live on a deployment with two data centres: two pools of two heavy forwarders each, every pool receiving its own HEC token from Cribl Stream senders. In four parts: create the tenant, follow the endpoints, pools and tokens, read the findings of a heavy forwarder stopped on purpose, and operate the tenant.

Why a dedicated Splunk HEC tenant



A Splunk administrator asks the same HEC questions every day: are all the HEC endpoints up and receiving, are senders being refused and why, where is the bottleneck when HEC answers Server is busy, which tokens send data Splunk rejects, did a business-critical token stop sending, and does the load balancer spread the traffic over every member of the pool. The Splunk HEC tenant answers them with findings raised by eighteen built-in policies on endpoints, pools and tokens. It only reads: nothing is installed or changed on Splunk, and a token is only ever known by its name.

How a Splunk HEC tenant works



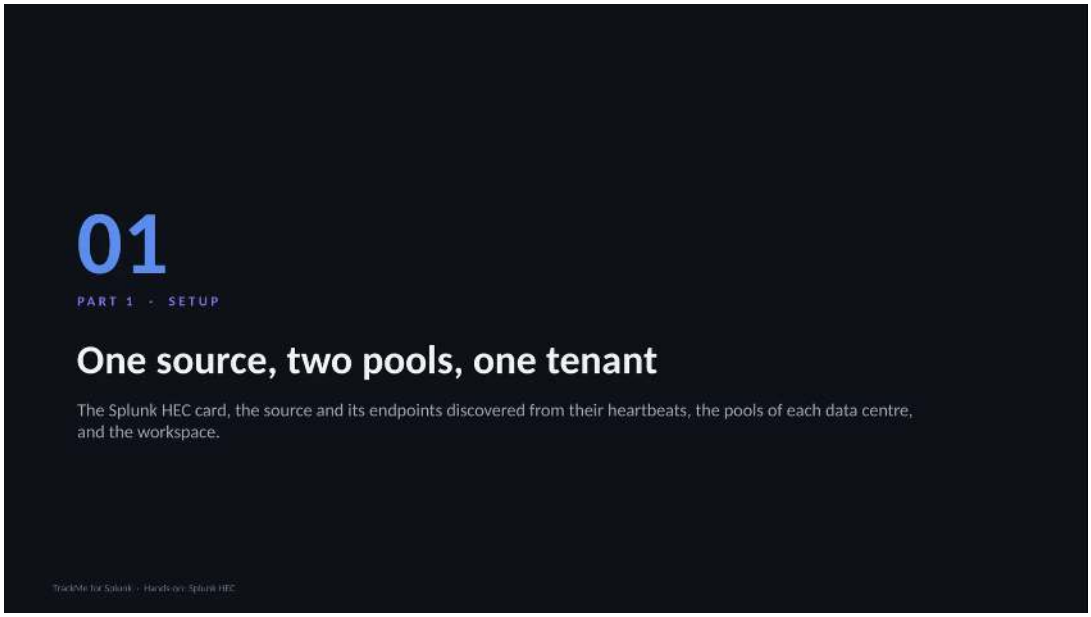
Every HEC instance writes its HEC metrics every minute, traffic or not: that heartbeat finds the endpoints and says whether they are up. Every five minutes TrackMe reads the last settled window of what Splunk indexed — throughput, refusals by reply code, queues and the bottleneck, output and load per endpoint and token — with a few bounded searches whatever the number of endpoints. Endpoints behind one load balancer form a pool, judged as a whole, and traffic models learn what each pool and token usually receives from the history _introspection already keeps.

The four parts of the walkthrough



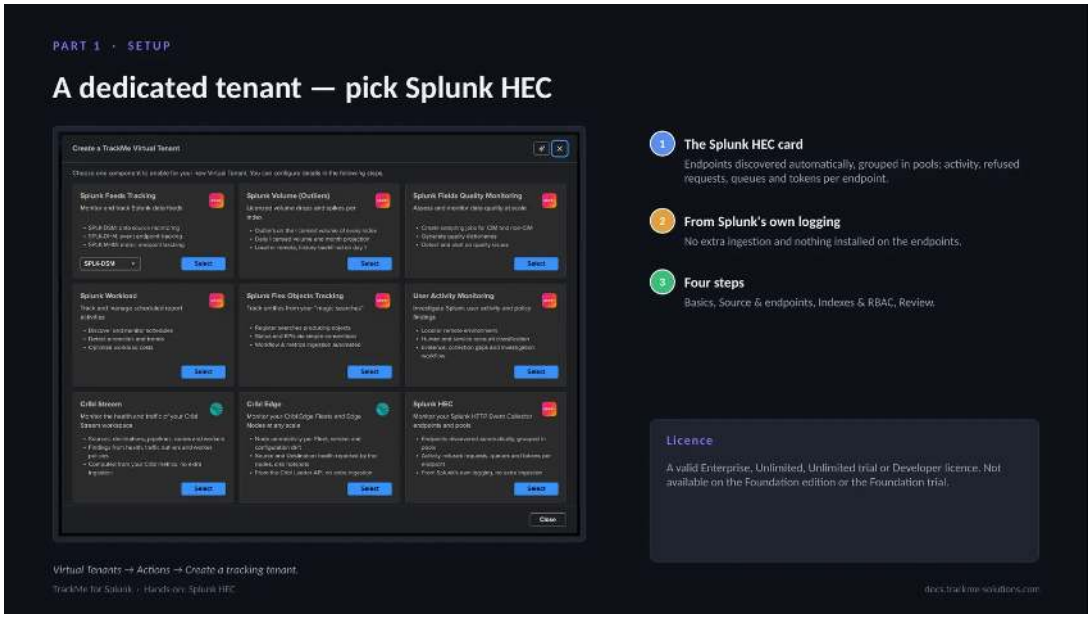
Setup covers the wizard, the pools and the workspace. Endpoints, pools and tokens covers the overview, the endpoints and one endpoint up close, a pool and how it spreads its traffic, and the tokens. Findings follows a heavy forwarder stopped on purpose: what opens, how it is evaluated, the topology and the AI Agent. Operate covers the policies, the tokens that must keep sending, the traffic models and their tuning, collection health and the splunkd API.

Part 1 — Setup



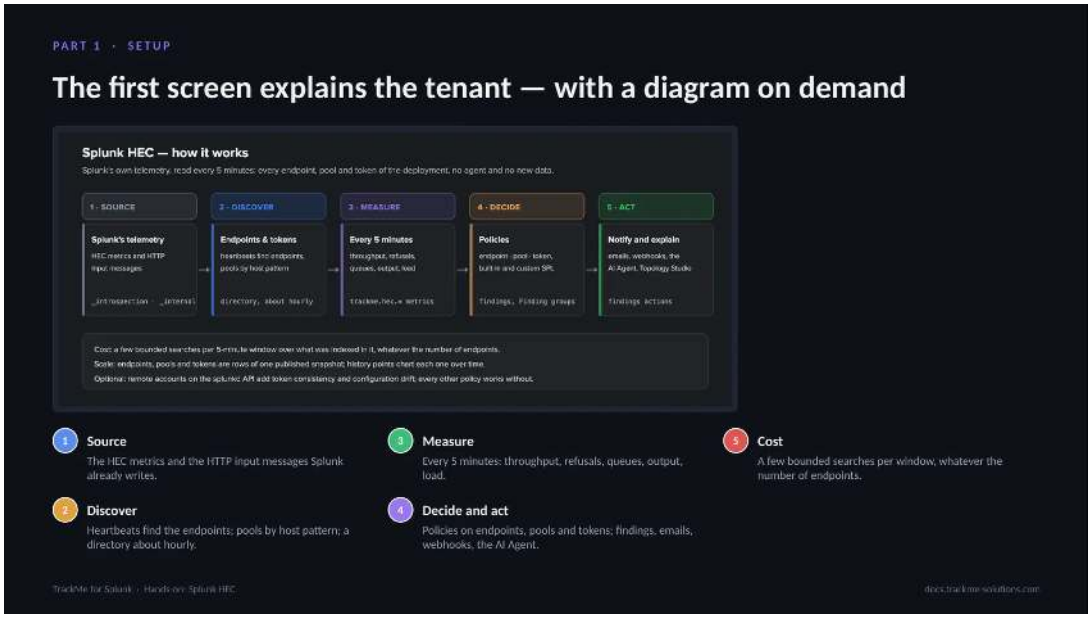
A Splunk HEC tenant reads one deployment: its local search tier, or another deployment through a TrackMe remote account, a Splunk Cloud stack included. This part covers the Splunk HEC card, the source and its endpoints discovered from their heartbeats, the pools of each data centre, and the workspace.

Choosing the Splunk HEC tenant type



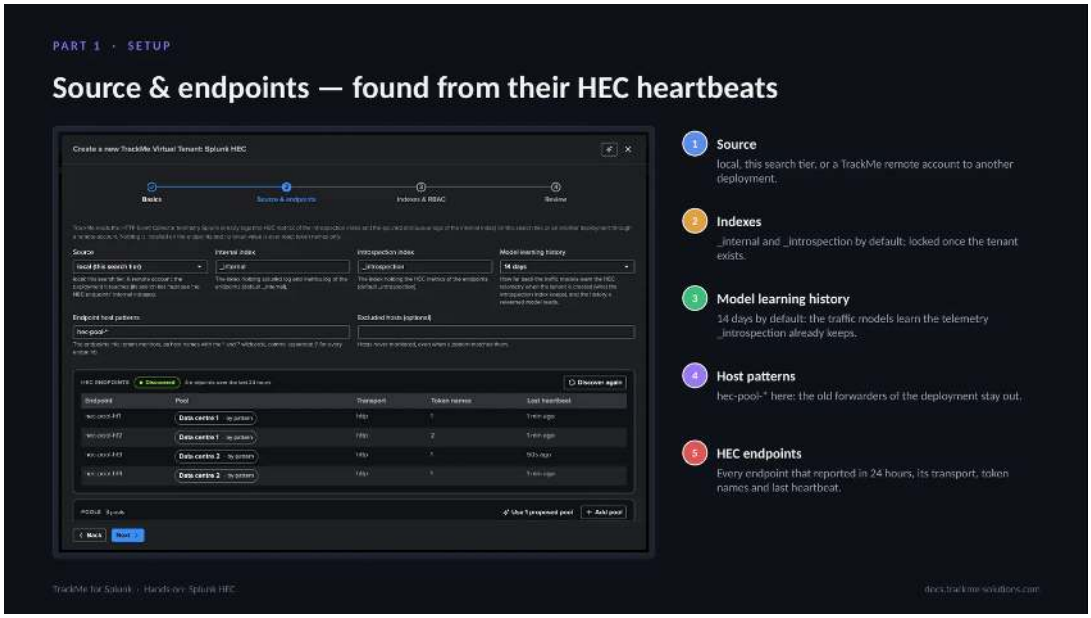
From Virtual Tenants, Actions, Create a tracking tenant, the Splunk HEC card monitors the HTTP Event Collector endpoints and pools of a deployment from Splunk's own logging, with no extra ingestion and nothing installed on the endpoints. The wizard has four steps: Basics, Source & endpoints, Indexes & RBAC, and Review. A Splunk HEC tenant has no entities and no state colour: it is a findings tenant. It needs a valid Enterprise, Unlimited, Unlimited trial or Developer licence.

The How it works diagram



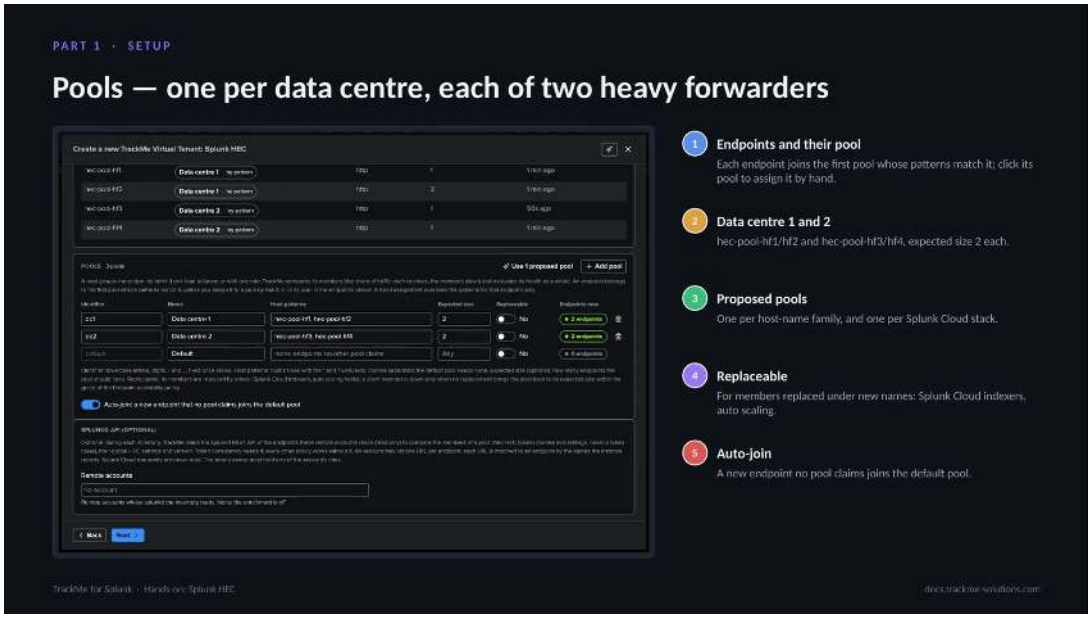
Basics holds the tenant identifier, an alias and a description, under a card that says what the tenant does. How it works draws the five stages: the source, the HEC metrics and HTTP input messages Splunk already writes; discovery of the endpoints from their heartbeats, with pools by host pattern; measurement every five minutes; the policies that decide; and the findings, emails, webhooks and AI Agent that act. The cost stays at a few bounded searches per window, and the optional splunkd API enrichment adds two configuration policies.

Source, indexes and discovered endpoints



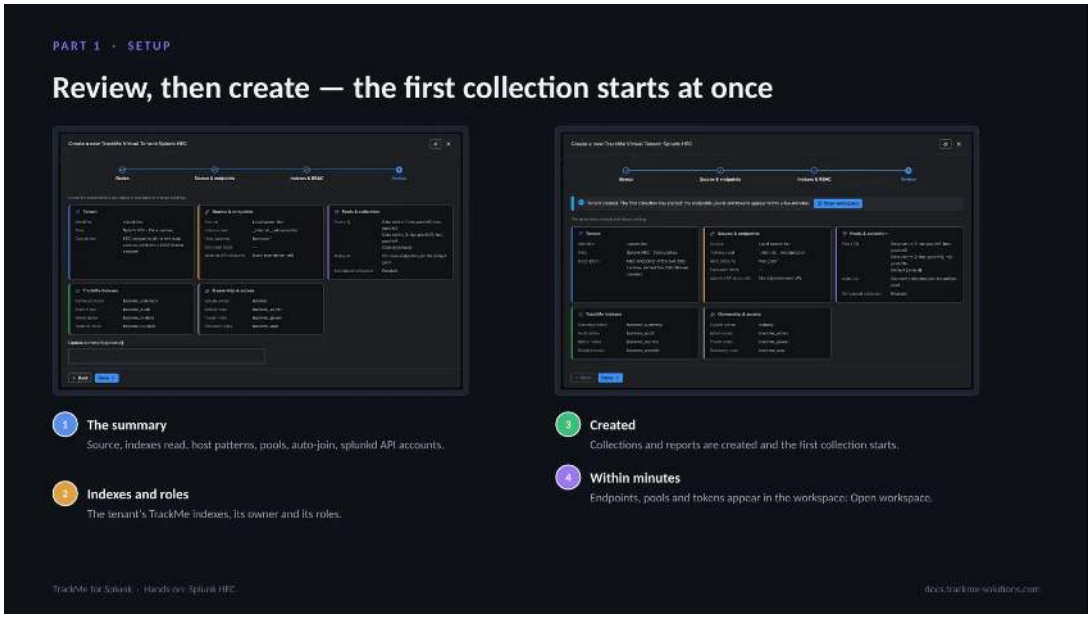
The source is local, this search tier, or a TrackMe remote account to another deployment. The internal and introspection indexes are locked once the tenant exists. The Model learning history, 14 days by default, is how far back the traffic models learn the telemetry _introspection keeps. Host patterns narrow the tenant to the endpoints it follows — hec-pool-* here, which leaves the older forwarders of the deployment out. TrackMe lists every endpoint that reported in the last 24 hours with its transport, token names and last heartbeat.

One pool per data centre



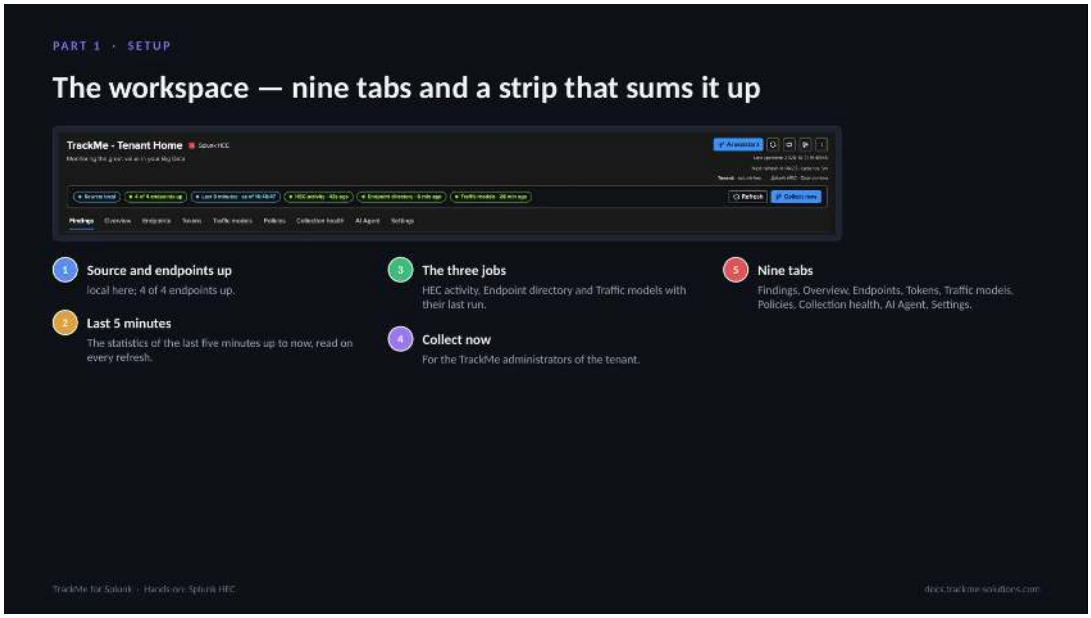
A pool groups the endpoints behind one load balancer or with one role. Each data centre has its own pool, Data centre 1 with hec-pool-hf1 and hec-pool-hf2 and Data centre 2 with hec-pool-hf3 and hec-pool-hf4, with an expected size of two each. An endpoint joins the first pool whose patterns match it, or the pool it is assigned to by hand. The discovery proposes one pool per host-name family and one per Splunk Cloud stack; Replaceable suits members replaced under new names, and Auto-join sends a new endpoint to the default pool.

Review and creation



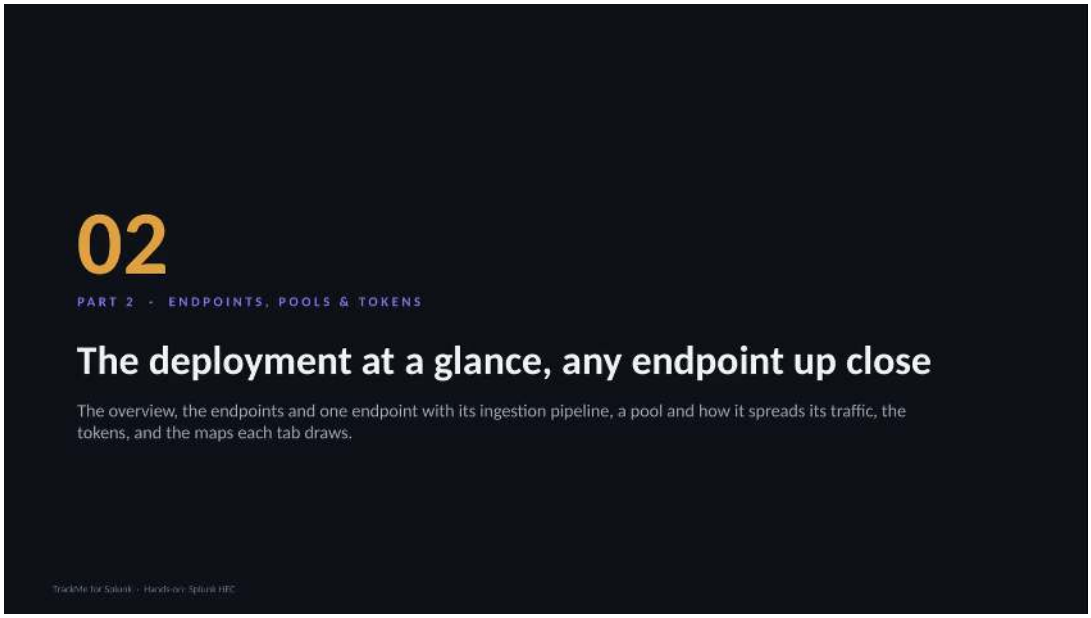
The review summarises the source, the indexes read, the host patterns, the pools, auto-join and the splunkd API accounts, with the tenant's TrackMe indexes, owner and roles. Creating the tenant creates its collections and reports and starts the first collection at once: the endpoints, pools and tokens appear in the workspace within minutes. The HEC activity is read and the policies evaluated every five minutes, the endpoint directory rebuilt about once an hour and the traffic models trained twice an hour.

The workspace and its status strip



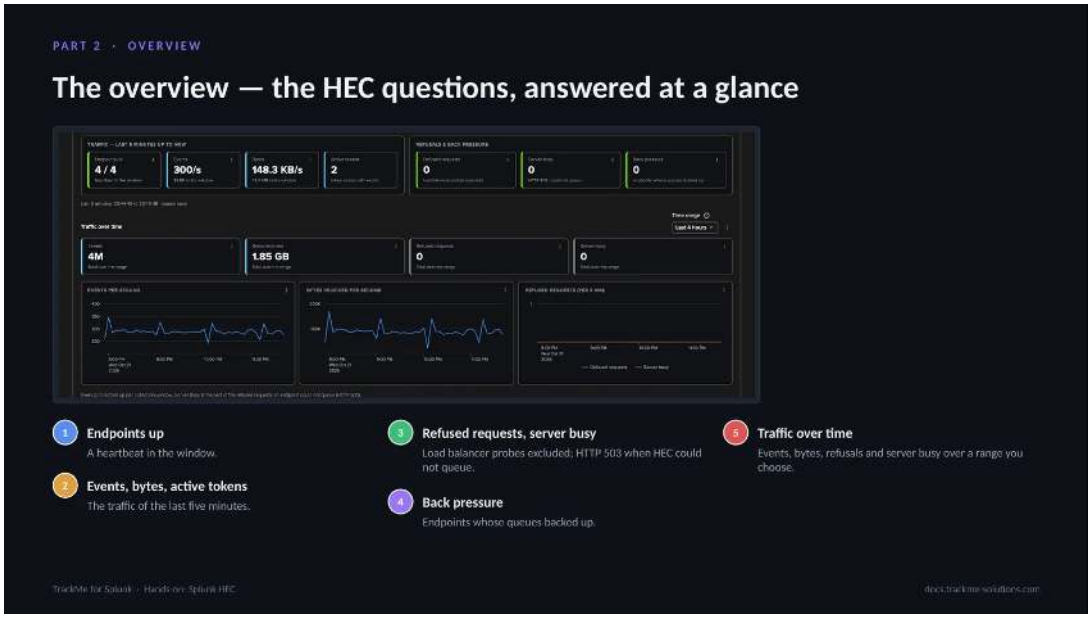
The Splunk HEC workspace is built with the same components as the Cribl workspaces. The status strip shows the source and the endpoints up, the Last 5 minutes badge, the last run of HEC activity, Endpoint directory and Traffic models, Refresh and, for the TrackMe administrators of the tenant, Collect now. Nine tabs follow: Findings, Overview, Endpoints, Tokens, Traffic models, Policies, Collection health, AI Agent and Settings. On every refresh TrackMe reads the last five minutes up to now and lays them over the collected rows.

Part 2 — Endpoints, pools & tokens



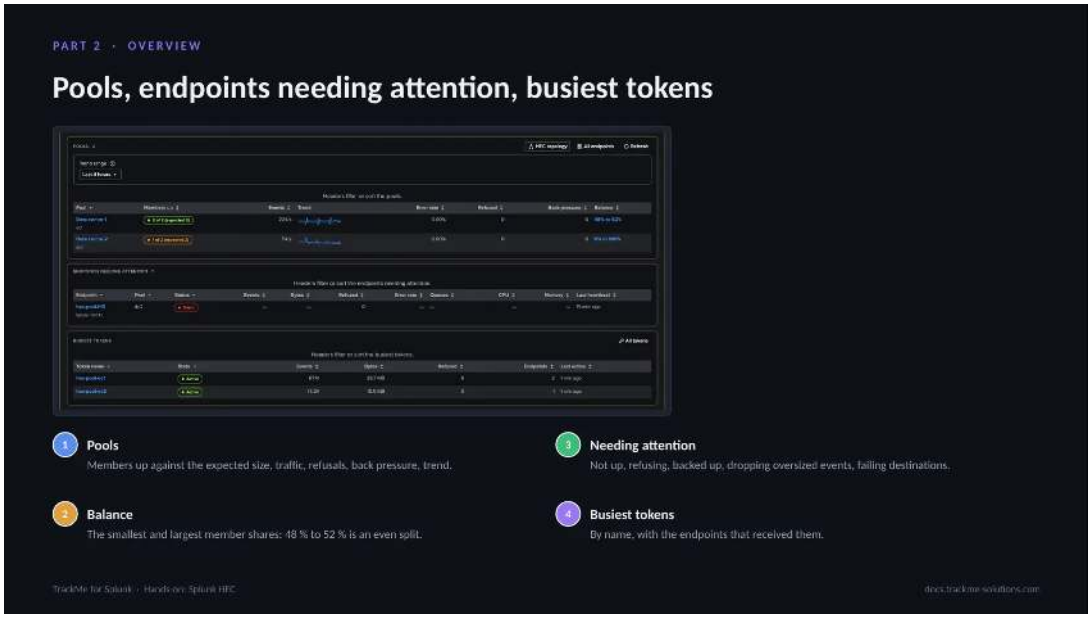
This part covers the deployment at a glance and any endpoint up close: the overview with its tiles and traffic, the pools and the endpoints needing attention, the Endpoints tab, one endpoint with its ingestion pipeline and output, a pool and how it spreads its traffic over its members, the tokens by name, and the live maps each tab draws.

Tiles and traffic over time



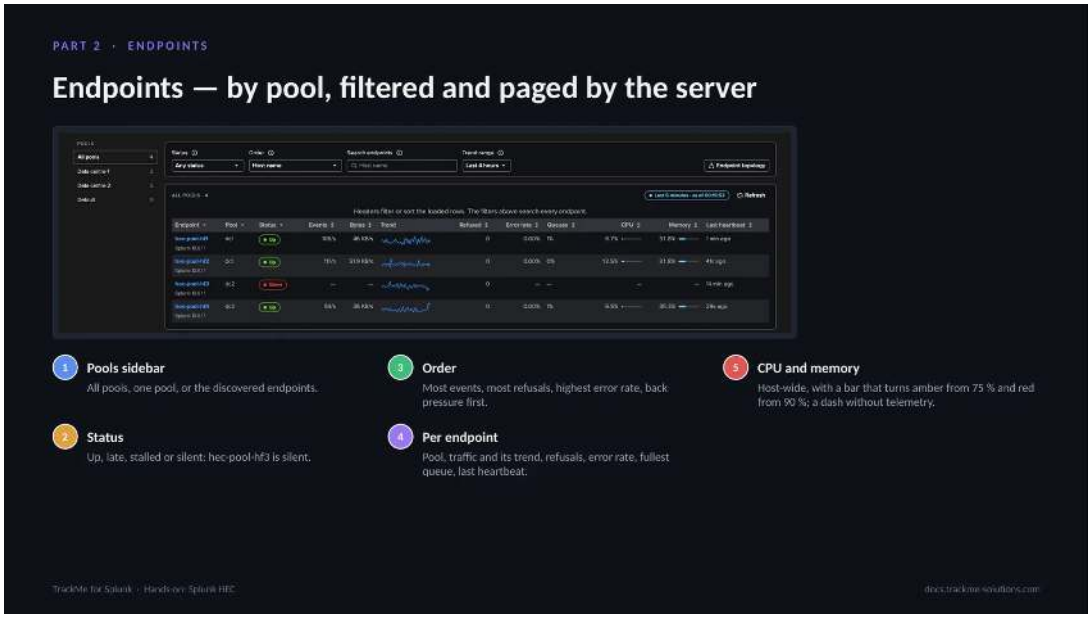
The overview answers the HEC questions at a glance: endpoints up with a heartbeat in the window, events, bytes and active tokens, refused requests with load balancer probes excluded, server busy replies when HEC could not queue a request, and endpoints whose queues backed up. Refused requests, server busy, back pressure and stalled open the Endpoints tab sorted or filtered on them. Traffic over time charts events, bytes, refusals and server busy replies over a chosen range.

Pools and the endpoints needing attention



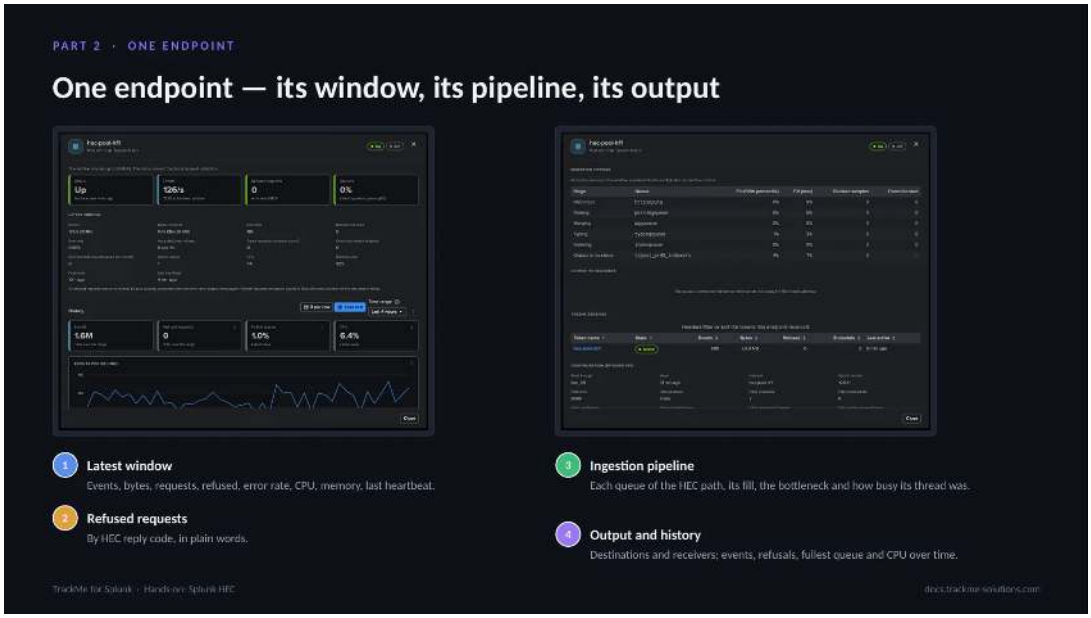
Below the charts, each pool shows its members up against the expected size, its traffic, refusals, back pressure, trend and balance — the smallest and largest member shares, 48 to 52 per cent for an even split. Data centre 2 has one of its two members up: hec-pool-hf3 was stopped, and it is listed as silent among the endpoints needing attention, with no CPU or memory figure. The busiest tokens follow, by name, with the endpoints that received them. A pool name or its balance opens the pool drilldown, and HEC topology draws every pool and its endpoints on a live map.

The Endpoints tab



Every endpoint, by pool, filtered and paged by the server: the sidebar selects all pools, one pool or the discovered endpoints, and the table filters by status and sorts by traffic, refusals, error rate or back pressure. An endpoint is up with a heartbeat in the window, late without one this window but one in the last five minutes, stalled when its output was in trouble and its own telemetry waits behind it, and silent otherwise, as hec-pool-hf3 is here, its trend falling while hec-pool-hf4 takes the traffic. Each row also shows the host's CPU and memory, every process on the host included, with a bar that turns amber from 75 per cent and red from 90; a stopped endpoint shows a dash. Endpoint topology draws the endpoints the tab selects.

One endpoint and its ingestion pipeline



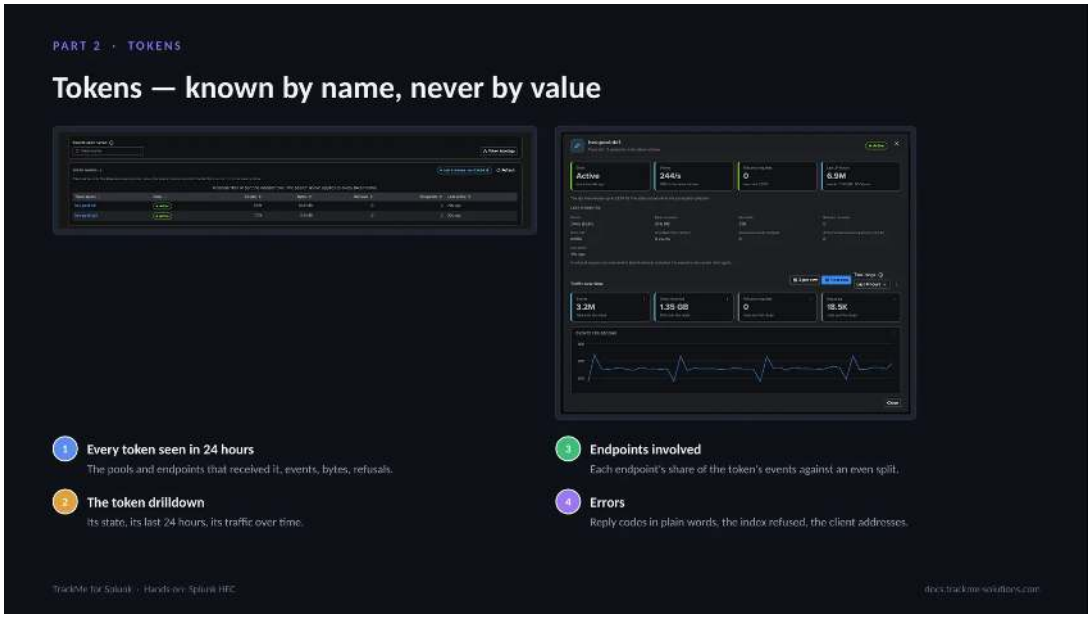
The endpoint drilldown shows the latest window — events, bytes, requests, refused requests, error rate, CPU, memory and last heartbeat — then the refused requests by reply code. The ingestion pipeline lists each queue of the HEC path with its fill, blocked samples, the bottleneck and how busy its thread was: when a stage cannot keep up its queue fills, then the queues before it, until HEC refuses requests. The output to receivers, the tokens received and the history complete it.

A pool and its traffic distribution



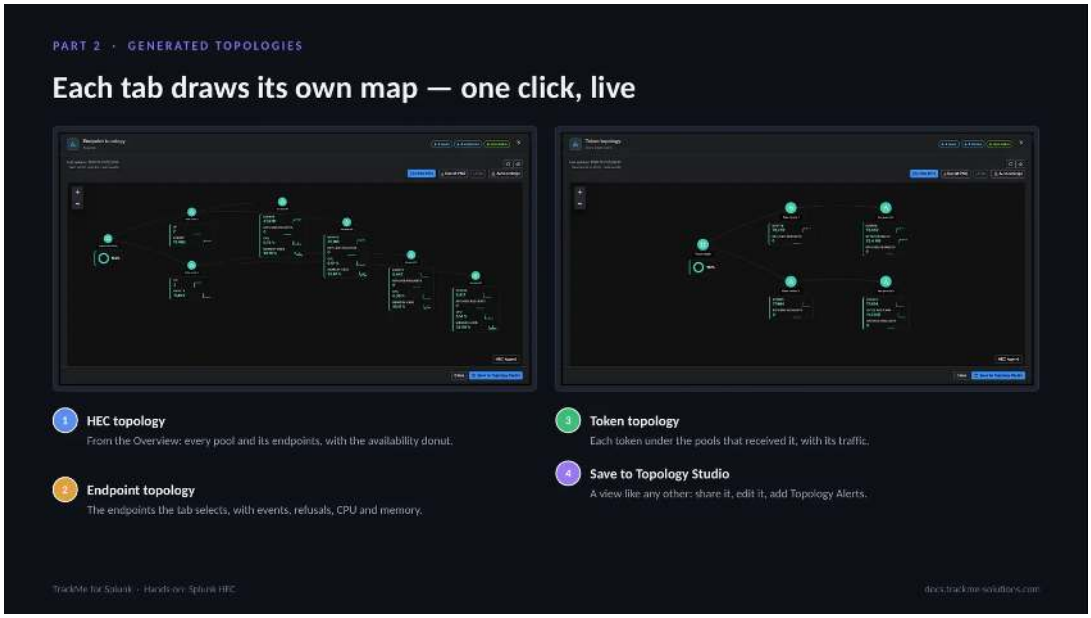
The pool drilldown shows how the members share what the pool receives, against an even split over the members up. Balance is 100 for an even split and 0 when one member takes everything; the tiles add the members carrying traffic, the smallest share against the fair share and the members up that received nothing. Over the last four hours of Data centre 2, the share of hec-pool-hf3 falls to zero while it was stopped, then comes back.

Tokens by name, never by value



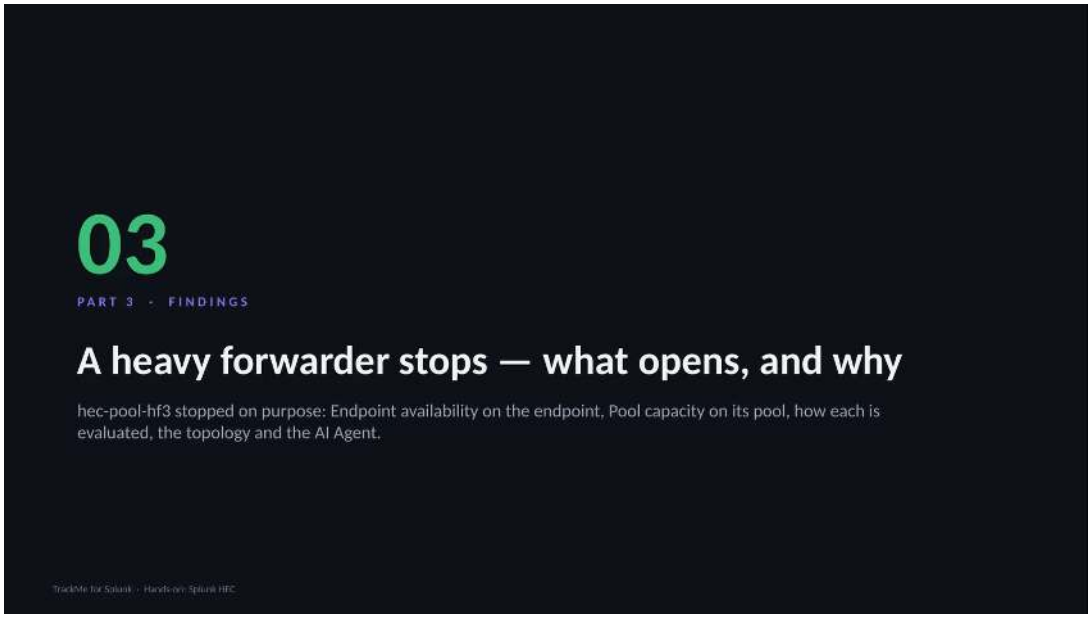
The Tokens tab lists every token name seen in the last 24 hours with the pools and endpoints that received it, its events, bytes and refused requests. A token opens its drilldown: its state, its last 24 hours, its traffic over time, the endpoints involved with each one's share of the token's events against an even split, and the errors in plain words. TrackMe never reads, stores, logs or shows a token value. Here each data centre has its own token, received only by the heavy forwarders of its pool.

Generated topologies



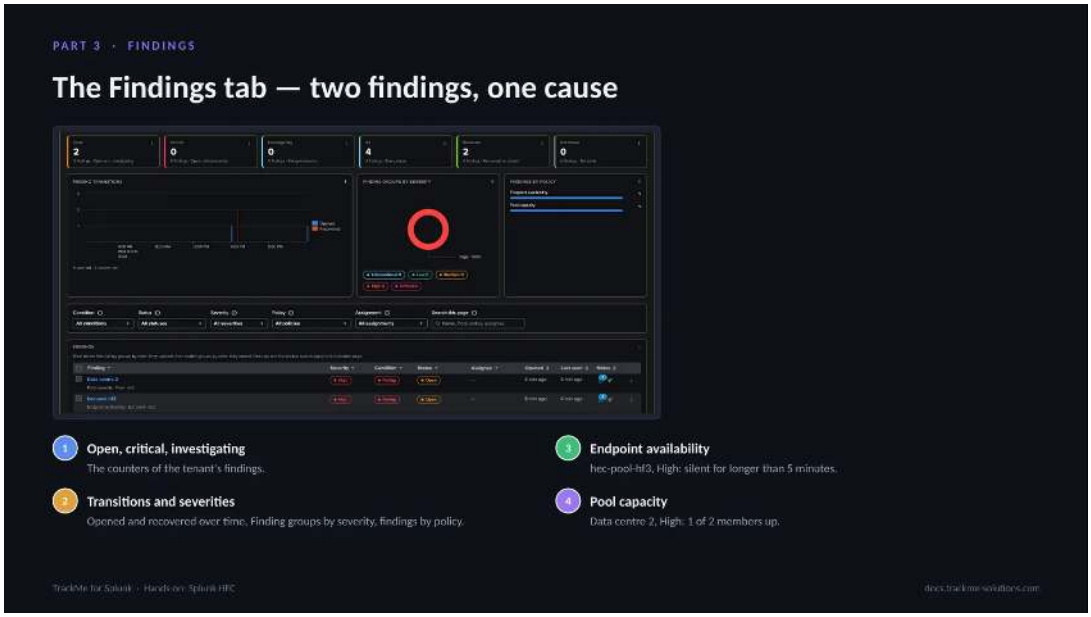
Each workspace tab draws a live map of what it shows, on the Topology Studio canvas. HEC topology, from the Overview, draws every pool and its endpoints with the availability donut. Endpoint topology draws the endpoints the Endpoints tab selects, each with its events, refused requests, CPU and memory. Token topology draws each token under the pools that received it, with its traffic. Every node shows its live status and its twenty-four-hour sparklines, refreshed every minute. Save to Topology Studio keeps the map as a view like any other: share it, edit it, add Topology Alerts.

Part 3 — Findings



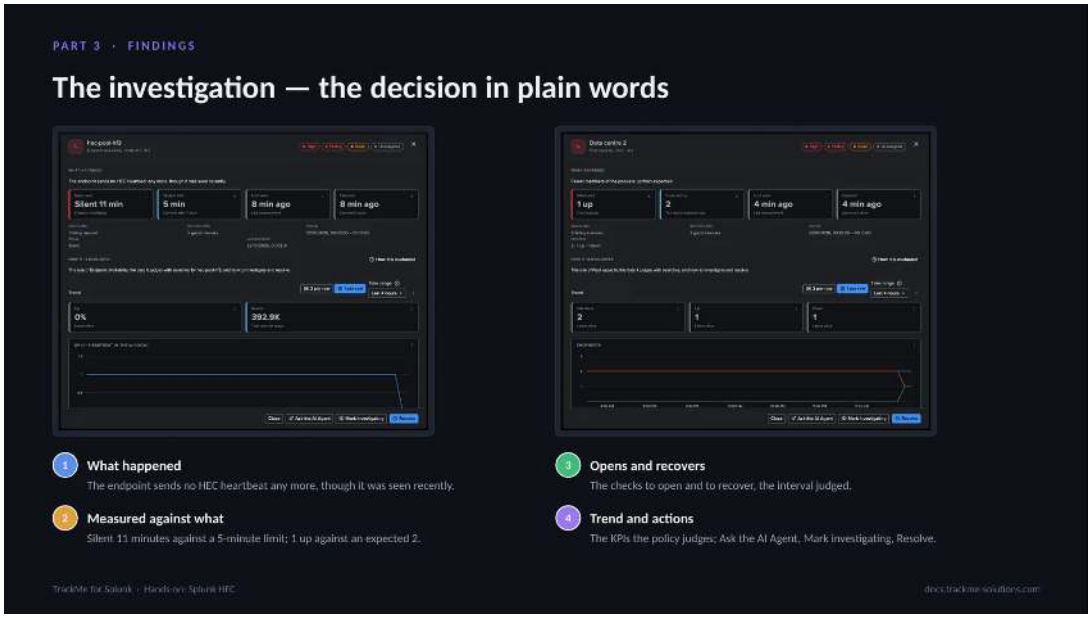
hec-pool-hf3, one of the two heavy forwarders of Data centre 2, was stopped on purpose. This part follows what opened — Endpoint availability on the endpoint and Pool capacity on its pool — how each is evaluated, the topology and the Splunk HEC AI Agent. Findings work as on the Cribl tenants: Finding groups, triage, notifications, assignment and findings actions.

Two findings from one stopped forwarder



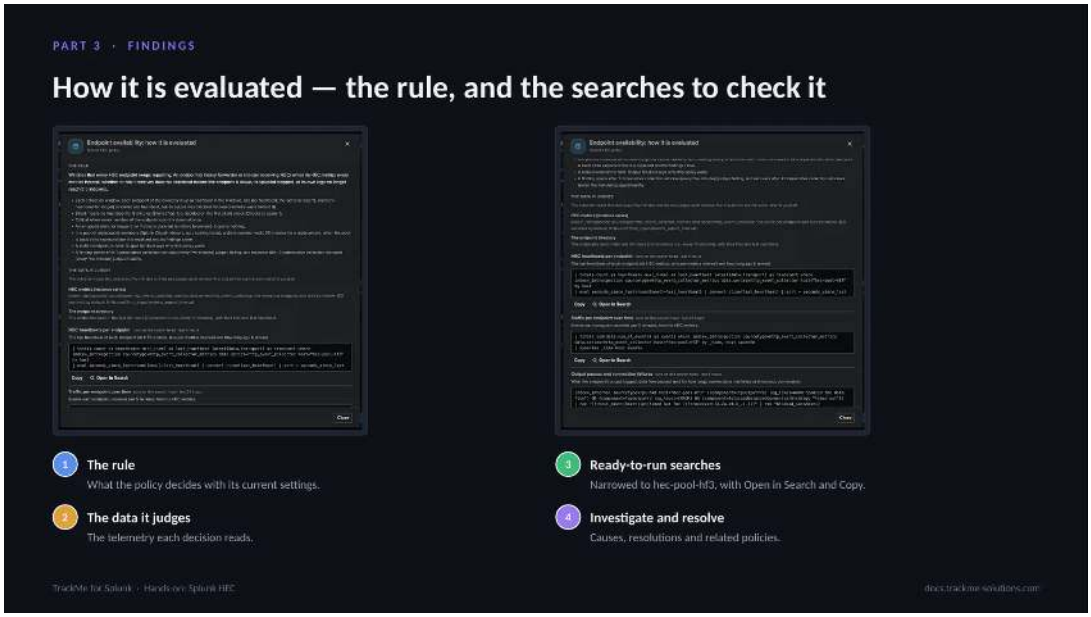
The Findings tab counts the open, critical and investigating findings, charts the transitions over time, the Finding groups by severity and the findings by policy, and lists the findings. Endpoint availability opened on hec-pool-hf3 at the first silent check, High; Pool capacity opened on Data centre 2 after two checks, High, critical only below half the expected size. Each object gathers its findings in one Finding group, the unit that is triaged and notified.

The investigation of each finding



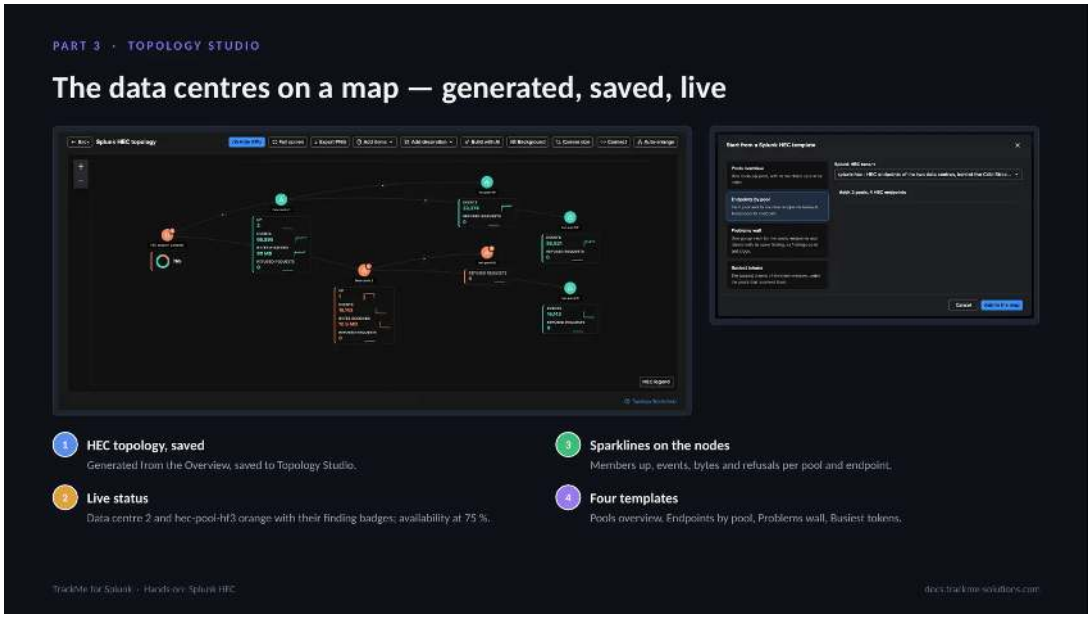
The investigation gives the decision in plain words — the endpoint sends no HEC heartbeat any more, though it was seen recently; fewer members of the pool are up than expected — the measurement and what it was compared with, the checks to open and to recover, the interval judged and the trend of the KPIs the policy judges. Ask the AI Agent, Mark investigating and Resolve sit at the bottom, and Open resource opens the endpoint, pool or token in place.

How a decision is evaluated



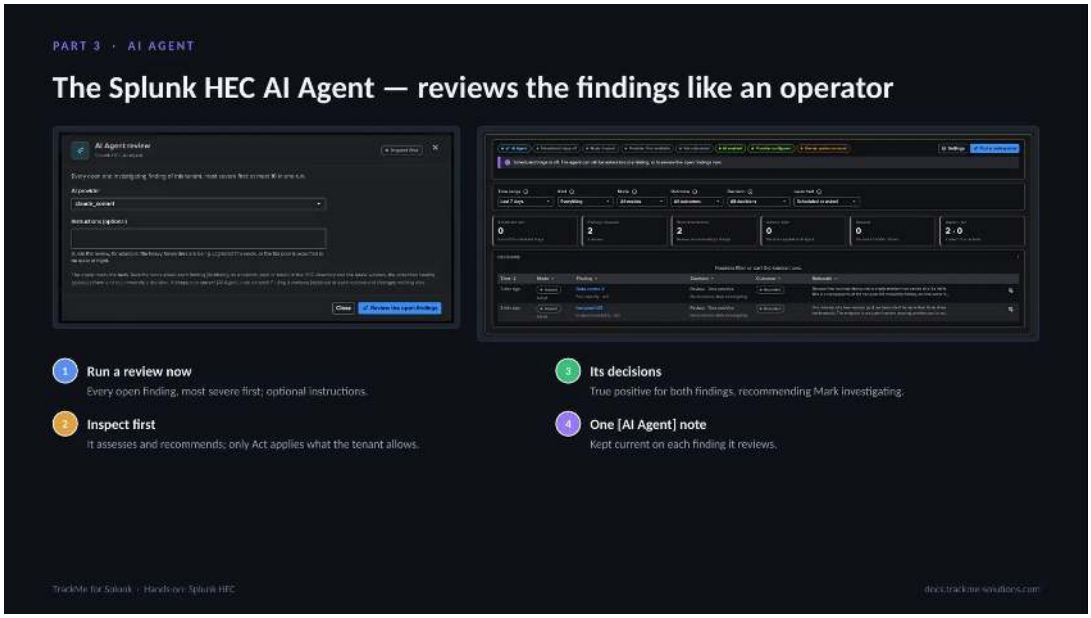
How it is evaluated explains the rule of the policy with its current settings, the data it judges and how to investigate and resolve. Its ready-to-run searches are built from the tenant's own source and, opened from a finding, narrowed to its endpoint or token — here host hec-pool-hf3 — each with Open in Search and Copy. An administrator checks the decision against Splunk's own data in one click.

The data centres in Topology Studio



Splunk HEC pools, endpoints and tokens are Topology Studio objects. This view is the HEC topology generated from the Overview and saved to Topology Studio: Data centre 2 and hec-pool-hf3 are orange with their finding badges while Data centre 1 and its members are green, and the endpoint availability fell to seventy-five per cent. Each node carries its sparklines: members up, events, bytes and refused requests. The templates — Pools overview, Endpoints by pool, Problems wall and Busiest tokens — draw other maps; groups follow their objects as they come and go, and Topology Alerts and Build with AI work on HEC tenants.

The Splunk HEC AI Agent



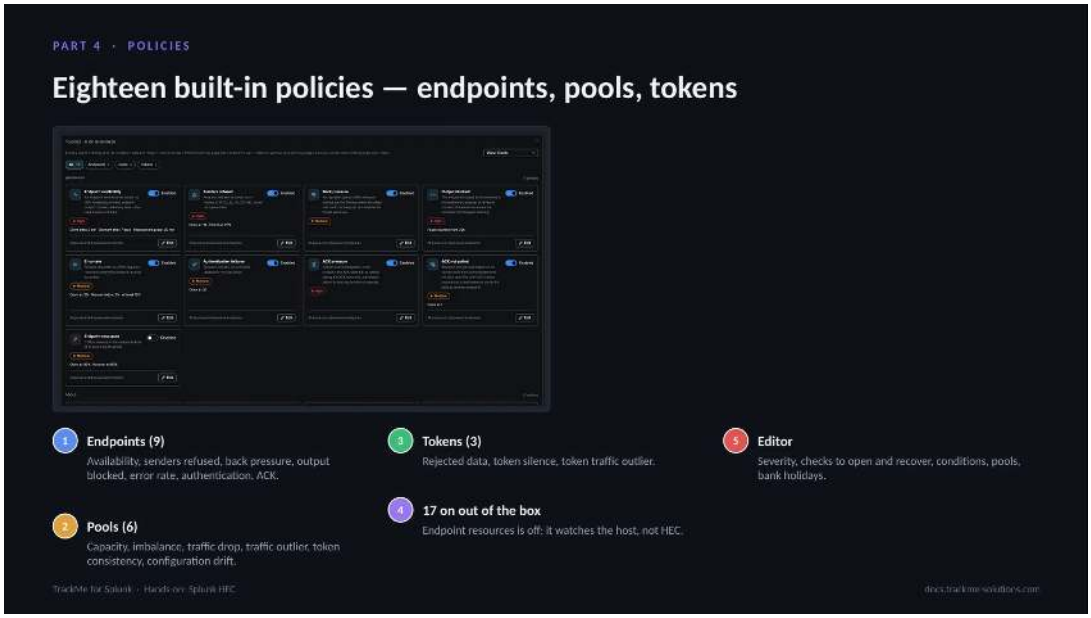
Run a review now reviews every open finding, most severe first, with optional instructions. The agent reads the facts TrackMe holds — the finding and its history, the endpoint or pool in the directory and the latest window, collection health and its own past decisions — and runs no search. It inspects first: here it judged both findings true positives and recommended marking them investigating, recorded without applying anything. Act applies only what the tenant allows, and scheduled triage is off by default.

Part 4 — Operate



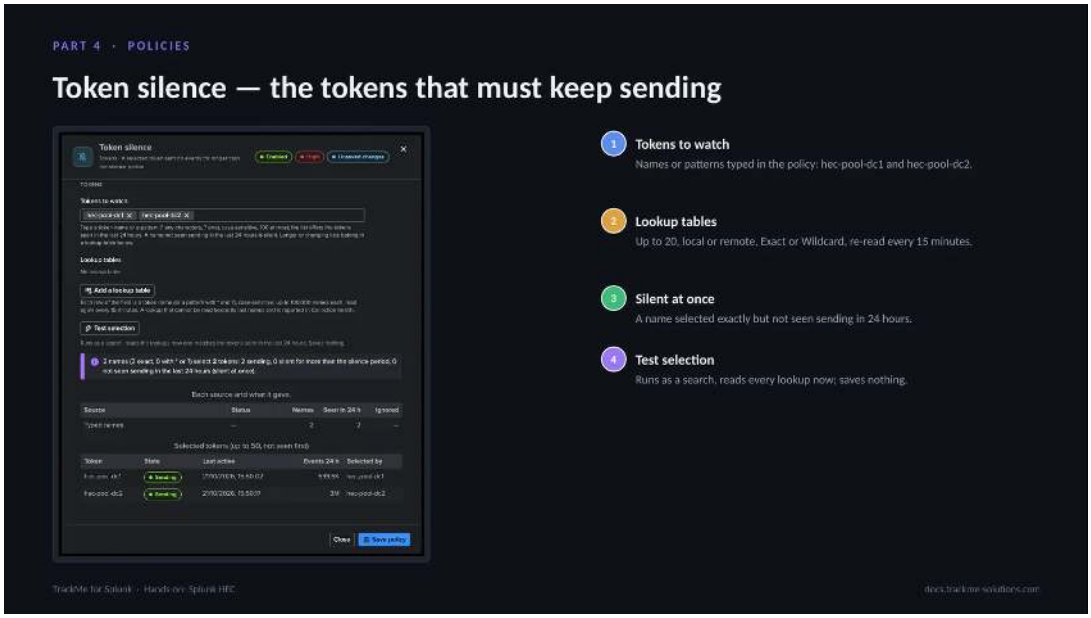
This part covers operating the tenant day to day: the eighteen built-in policies and their editor, the tokens that must keep sending selected by name or from lookups, the traffic models and their tuning, collection health with its run budgets, and the optional splunkd API enrichment of the endpoints.

The built-in policy catalogue



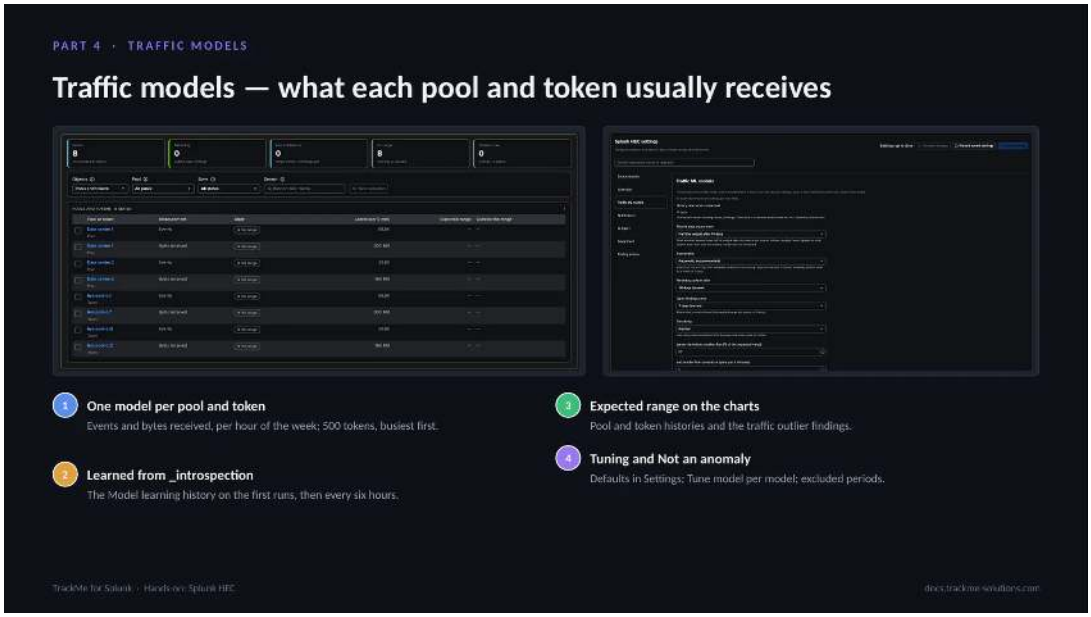
Eighteen built-in policies in three categories: nine on endpoints (availability, senders refused, back pressure, output blocked, error rate, authentication failures, ACK pressure, ACK not polled and endpoint resources), six on pools (capacity, imbalance, traffic drop, traffic outlier, token consistency and configuration drift) and three on tokens (rejected data, token silence and token traffic outlier). Seventeen are on out of the box. A TrackMe administrator of the tenant edits each one's severity, checks, conditions, pools and bank holiday behaviour, and writes custom SPL policies.

Selecting the tokens that must keep sending



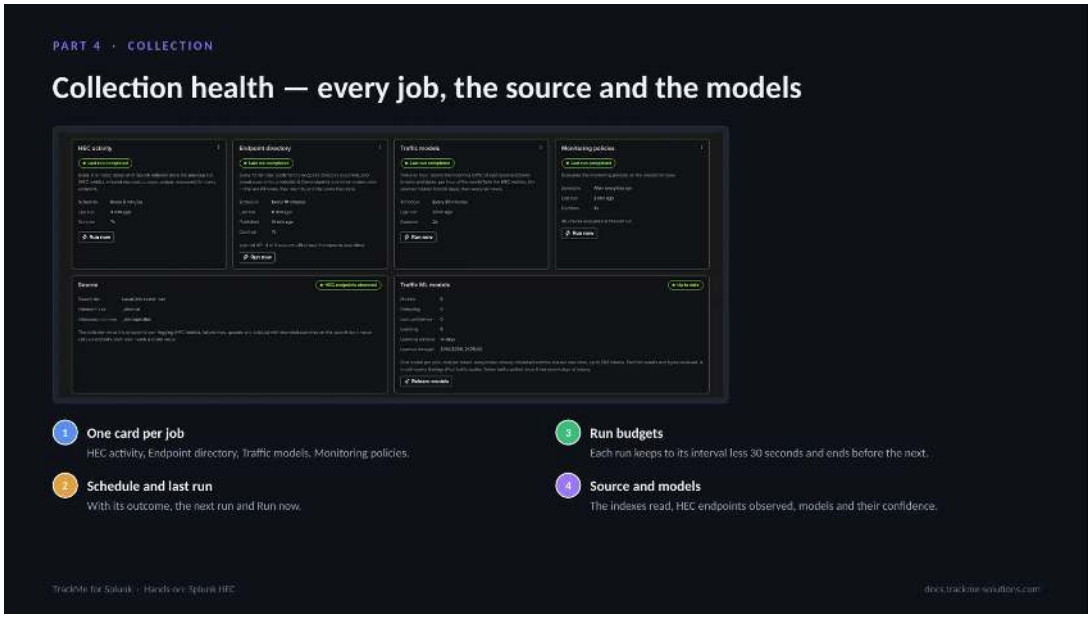
Token silence opens a finding when a selected token sends no event for longer than the silence period, 15 minutes by default. Tokens are typed in the policy — hec-pool-dc1 and hec-pool-dc2 here — or read from up to 20 Splunk lookup tables, local or remote, in Exact or Wildcard mode. A name selected exactly and not seen sending in the last 24 hours is silent at once. Test selection runs as a search, reads every lookup now and saves nothing.

Traffic models and their tuning



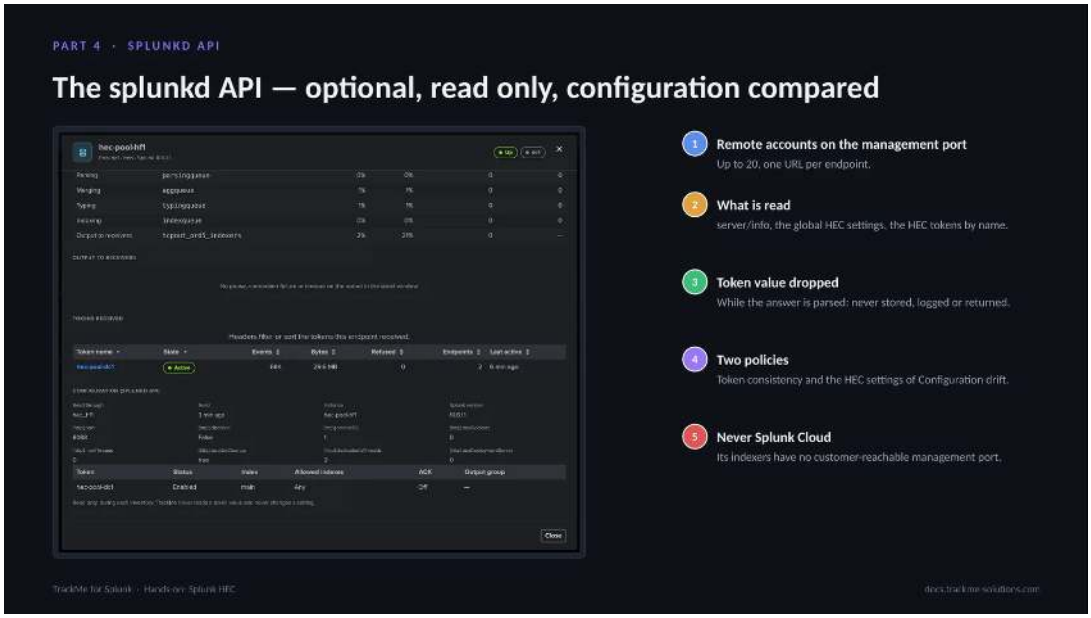
One model per pool and per token, up to 500 tokens busiest first, learns the events and bytes received per hour of the week with the arithmetic of the Cribl Stream traffic models, from the Model learning history on the first runs. The expected range is drawn on the pool and token charts, and Pool traffic outlier and Token traffic outlier open findings once a model detects. Settings, Traffic ML models holds the defaults; Tune model gives a model its own settings and excluded periods, and Not an anomaly excludes an expected episode.

Collection health



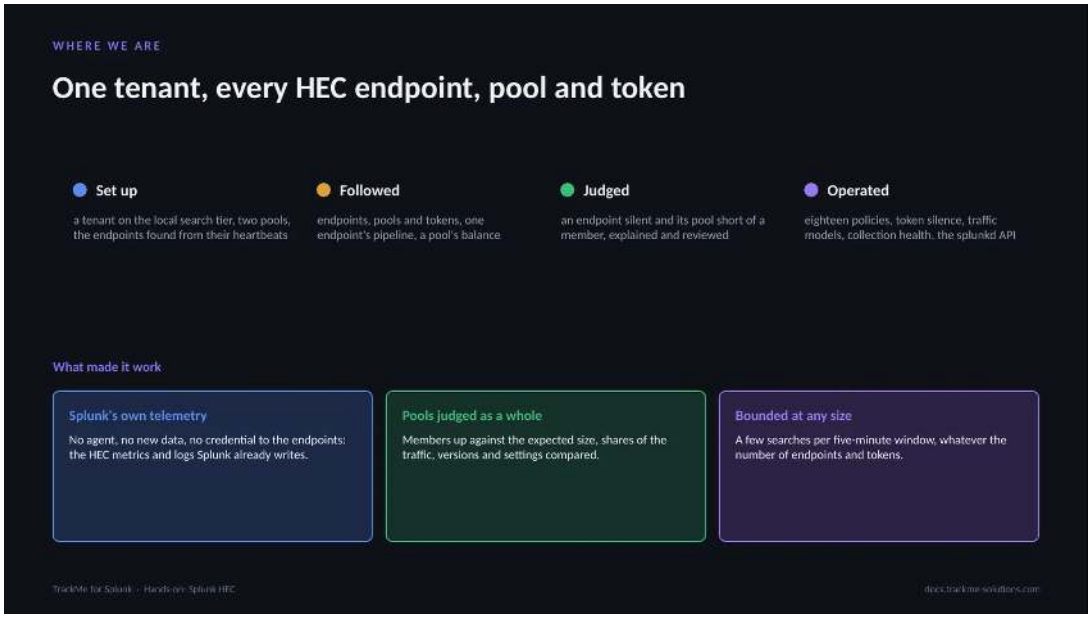
One card per job — HEC activity, Endpoint directory, Traffic models and Monitoring policies — with its schedule, last run, outcome and Run now. Each run keeps to its interval less 30 seconds and ends before the next scheduled run. The Endpoint directory card says what the splunkd API read, the Source panel shows the search tier and indexes read, and the Traffic ML models panel the models and their confidence. When collection does not run as configured, the tab lists the problems and the Configuration Guardian raises an alert.

The optional splunkd API enrichment



With TrackMe remote accounts bound on the endpoints' management port, the endpoint directory reads server/info, the global HEC settings and the HEC tokens by name at each rebuild. The token value is dropped while the answer is parsed: never stored, logged or returned. The endpoint drilldown shows the instance's HEC settings and its tokens with their status, index, allowed indexes and ACK. Token consistency and the HEC settings part of Configuration drift use it; Splunk Cloud is never read.

Where we are



A Splunk HEC tenant was set up on a deployment with two data centres: its endpoints followed from their heartbeats, its pools compared member by member and its tokens by name. A heavy forwarder was stopped, and the two findings it opened were explained, reviewed by the AI Agent and recovered. Three choices make it work: Splunk's own telemetry with no agent and no new data, pools judged as a whole, and a few bounded searches per window at any size.

2.1.12 Splunk S2S

Follow the Splunk-to-Splunk receiving side of a deployment from the telemetry Splunk already writes: the indexer cluster found without configuration as one pool per site and its manager account for planned work, the heavy forwarder tier found as an automatic pool and made a configured one, what is blocked now, a pool's load distribution, a receiver's pipeline queue by queue and the data path from the senders to the indexers; a heavy forwarder stopped on purpose, the three findings it opens, how each is evaluated, the notification email and the Splunk S2S AI Agent; then the twenty-two policies, the traffic models and collection health.

`Download the deck (PDF) · PowerPoint`

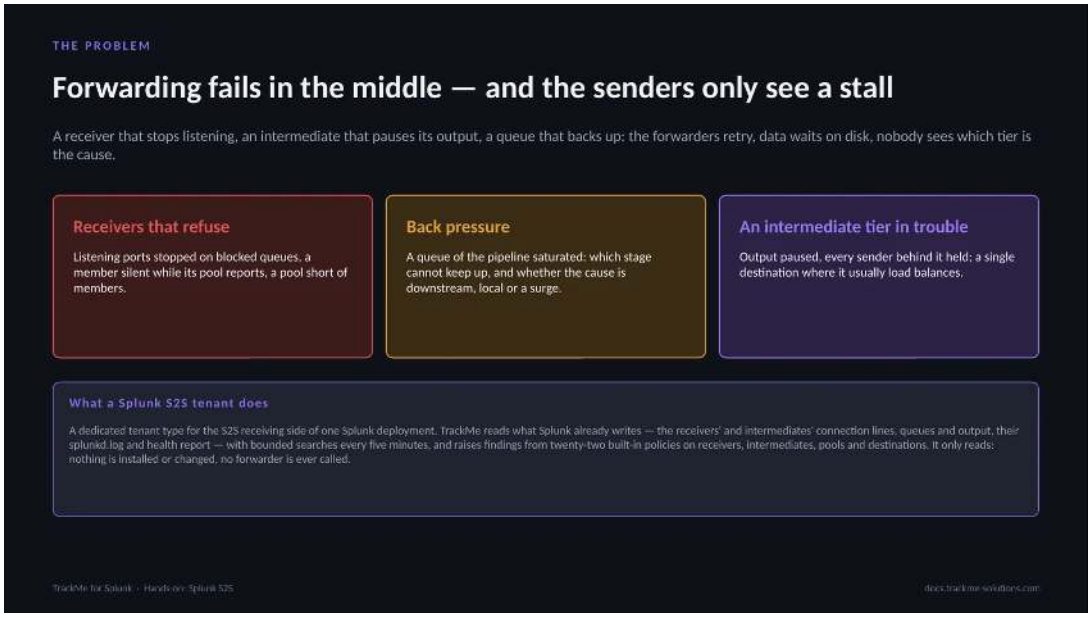
about 30 minutes · 26 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



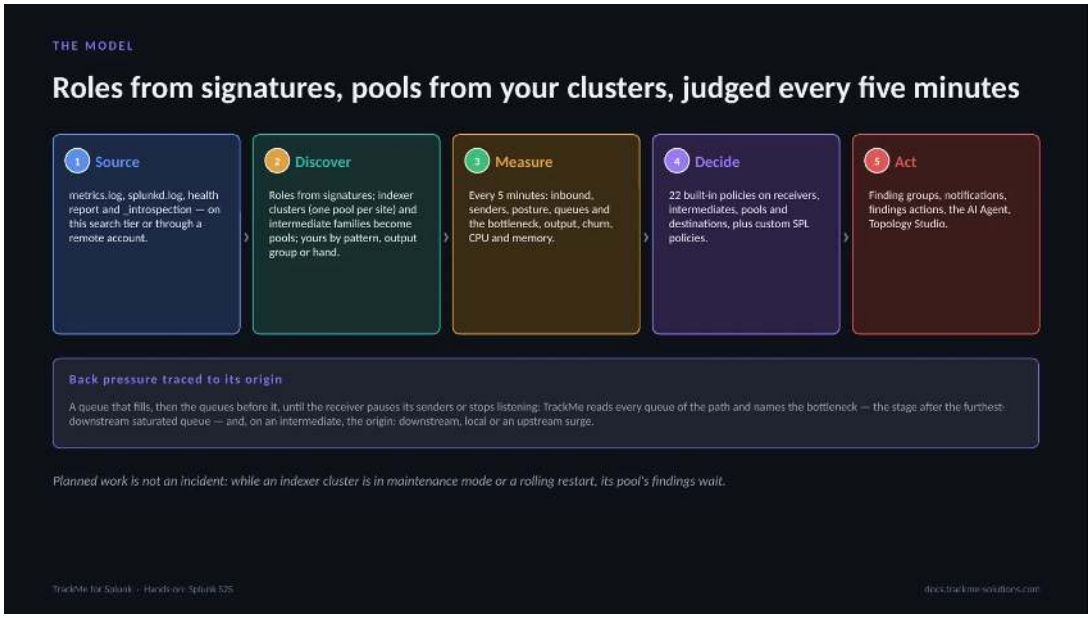
This walkthrough covers the Splunk S2S tenant, generally available in TrackMe 2.4.22. Everything was captured live on a deployment with a four-peer indexer cluster on two sites, a tier of four heavy forwarders, and universal forwarders sending to that tier. In four parts: create the tenant, follow the pools, the hosts and the data path, read the findings of a heavy forwarder stopped on purpose, and operate the tenant.

Why a dedicated Splunk S2S tenant



A Splunk administrator asks the same forwarding questions every day: is every receiver accepting and keeping up, is the intermediate tier blocked, backed up or silent, which tier is the bottleneck when senders are held back, and is the cause downstream, local or an upstream surge. How much flows into each pool, from how many senders, with which TLS and acknowledgement posture? The Splunk S2S tenant answers them with findings raised by twenty-two built-in policies. It only reads the telemetry Splunk already writes: nothing is installed or changed on a forwarder or an indexer.

How a Splunk S2S tenant works



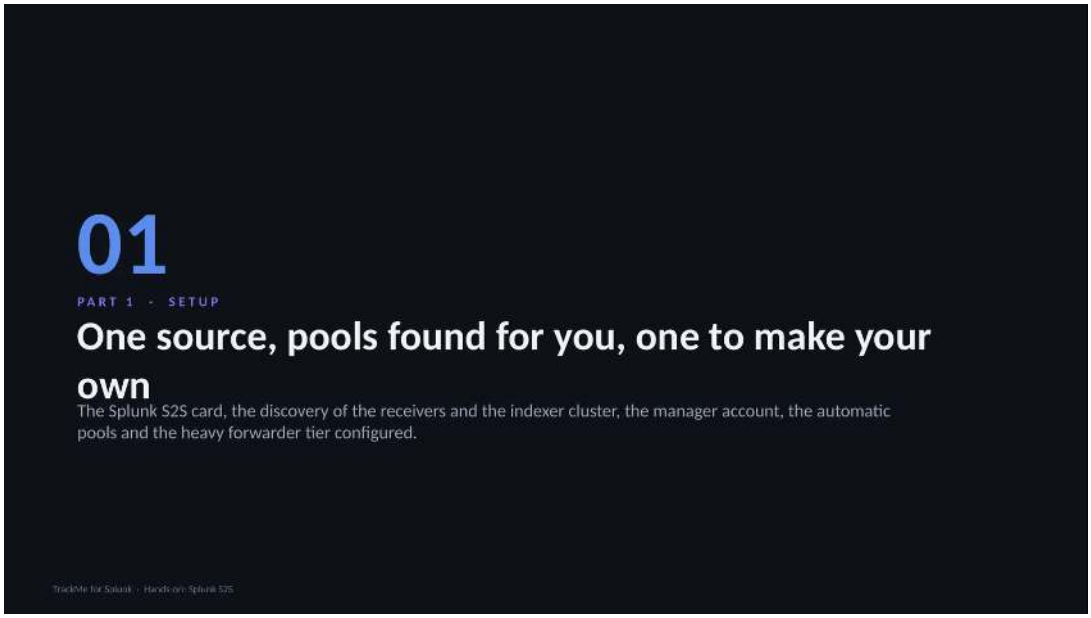
Roles are read from signatures over a day: a host that receives and indexes is a receiver, a host that receives and forwards is an intermediate forwarder. Each indexer cluster the search head is attached to becomes a pool by itself, one per site of a multisite cluster; the intermediate forwarders no pool claims form one automatic pool per host-name family, and other hosts are grouped by host patterns, output groups or hand assignment. Every five minutes the live job reads the last settled index-time window, and the policies decide. Back pressure is traced to its origin: the stage that cannot keep up, and whether the cause is downstream, local or an upstream surge. Planned work is not an incident: an indexer cluster in maintenance mode or a rolling restart holds the pool's findings.

The plan



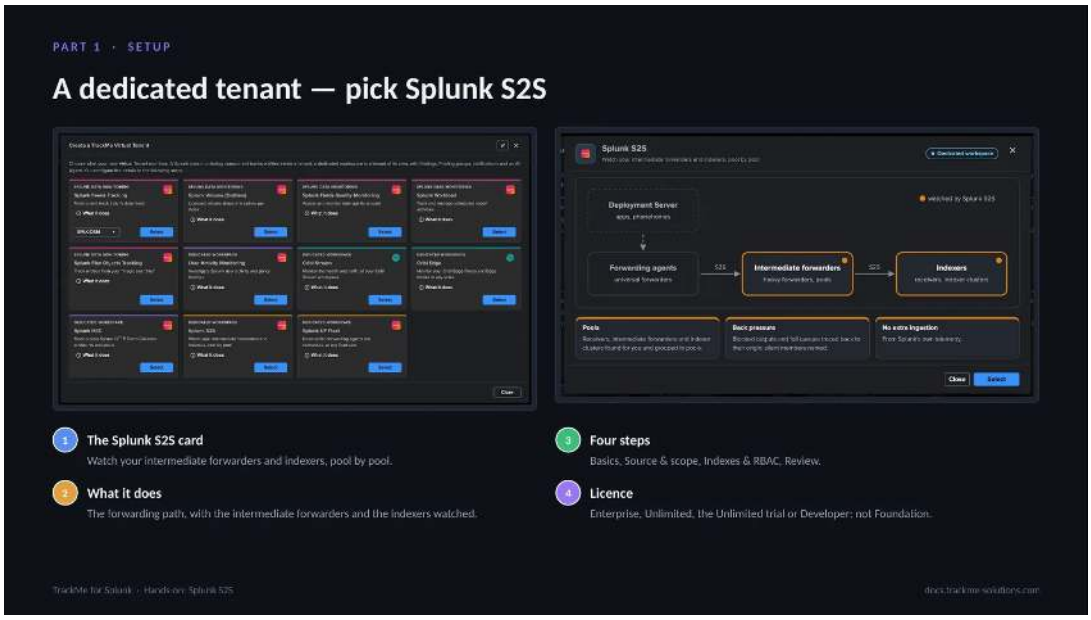
Four parts. Setup: the wizard, the indexer cluster's site pools and their manager account, the automatic pools and the heavy forwarder tier made a configured pool. Pools, hosts and the data path: the Pools tab and Blocked now, a pool's distribution, a receiver's pipeline and the data path. Findings: a heavy forwarder stopped on purpose, what opens and how it is evaluated, the AI Agent and the notification email. Operate: the policies, the traffic models and collection health.

Part 1 — Setup



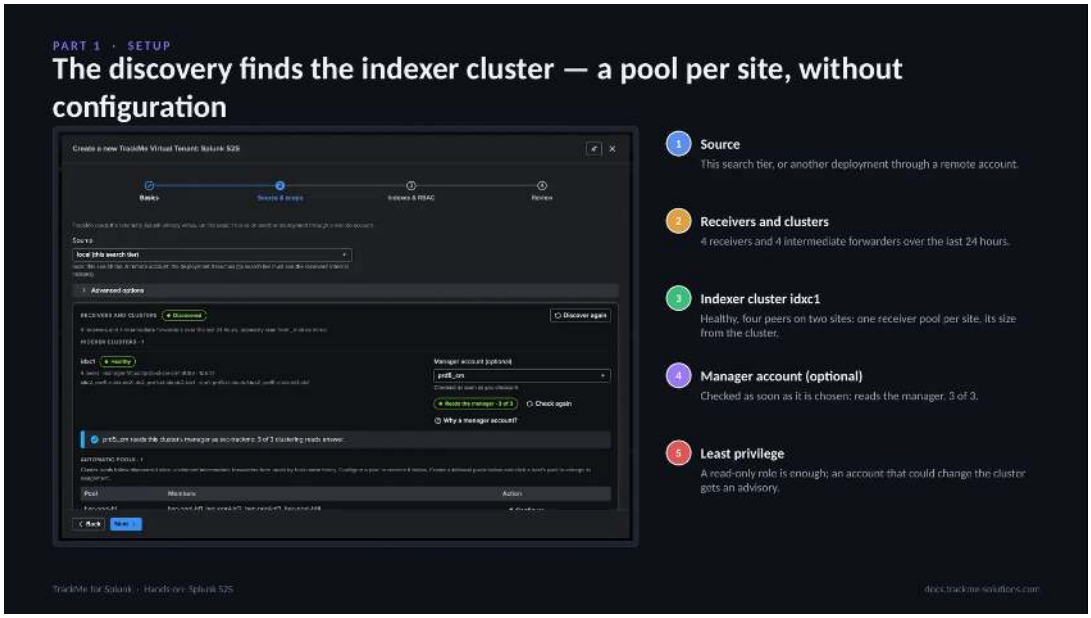
Part one: setup. A Splunk S2S tenant reads one deployment: its local search tier, or another deployment through a TrackMe remote account.

A dedicated tenant — pick Splunk S2S



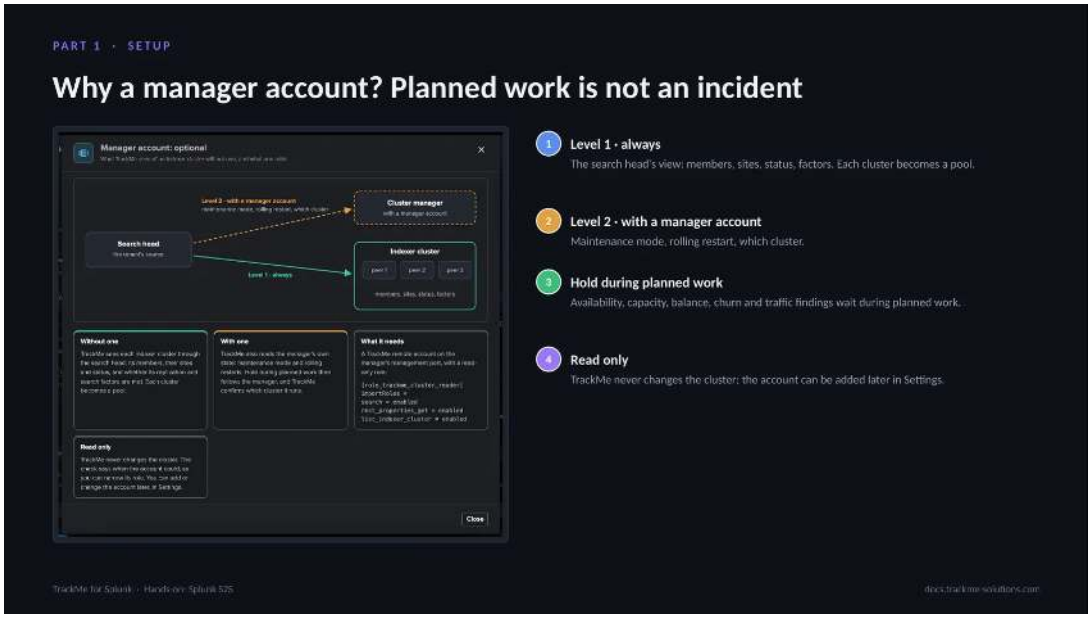
Virtual Tenants, Actions, Create a tracking tenant, then the Splunk S2S card. What it does draws the forwarding path: the forwarding agents send over S2S to the intermediate forwarders, which forward to the indexers, and Splunk S2S watches the intermediates and the indexers. A Splunk S2S tenant has no entities and no state colour: it is a findings tenant, like the Splunk HEC, Cribl and User Activity Monitoring tenants.

The discovery finds the indexer cluster



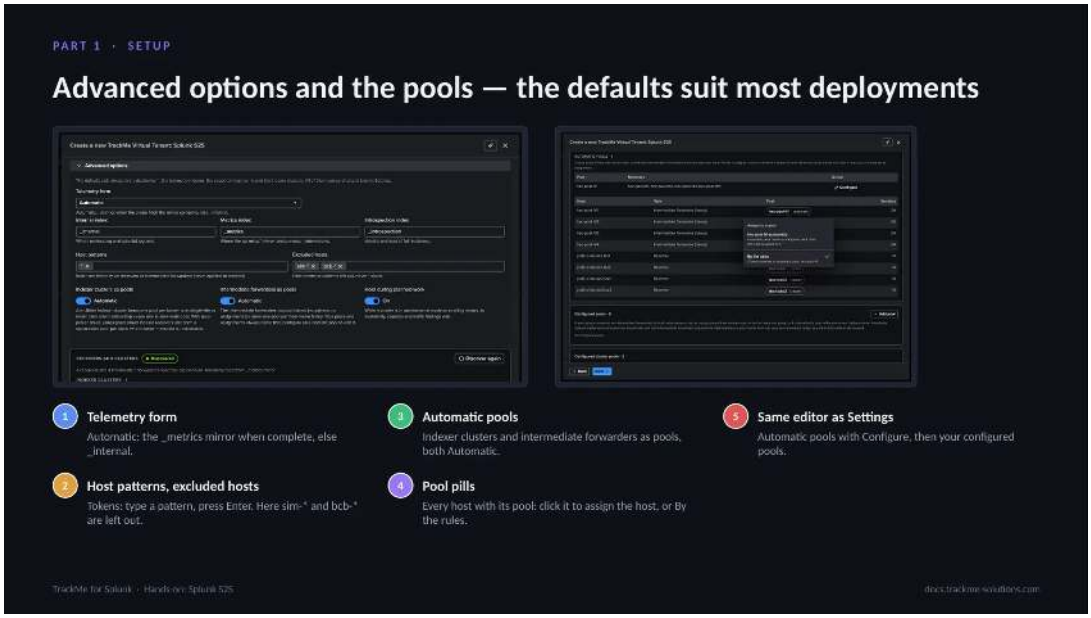
Source and scope. The discovery reads the receivers and intermediates of the last day, and each indexer cluster the search head is attached to: here idxc1, four peers on two sites, which forms one receiver pool per site. The optional manager account lets the tenant read the manager's maintenance mode and rolling restarts. It is checked as soon as it is chosen: here a least-privilege service account reads the manager, 3 of 3 clustering reads. An account that could change the cluster gets an advisory with the read-only role to create: search, rest_properties_get and list_indexer_cluster.

Why a manager account? Planned work is not an incident



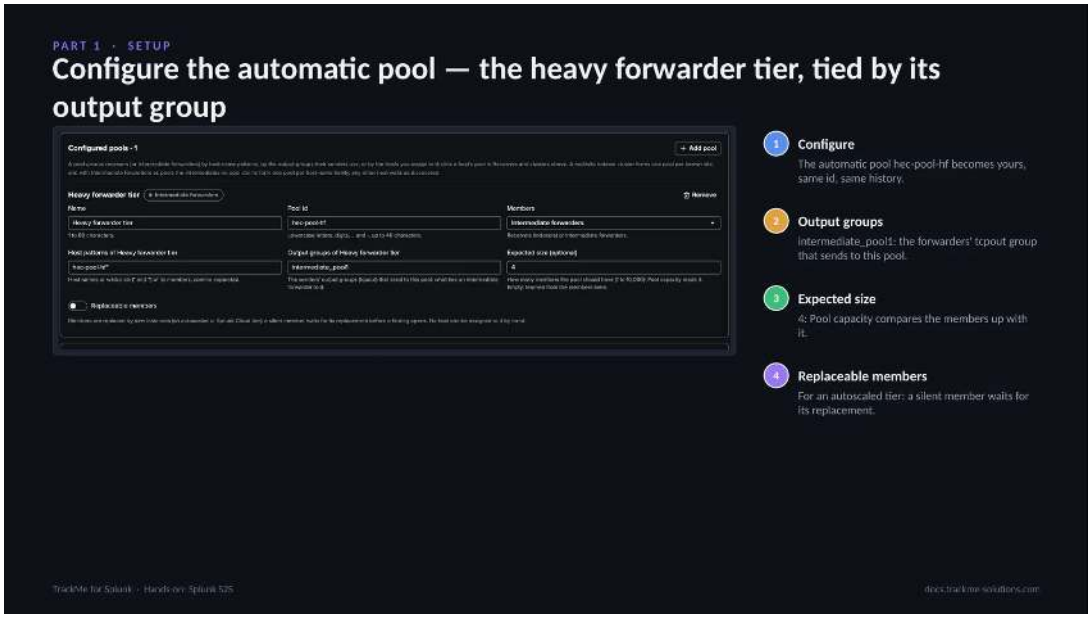
The search head always sees its indexer clusters: their peers, their replication and search factors, whether the data is searchable. A manager account adds the manager's own state, so a rolling restart or a maintenance window holds the pool's findings instead of opening them: an open finding stays open, and a new one waits until the planned work ends. Without it, the tenant works from the search head's view only, and the Review step says so.

Advanced options and the pools



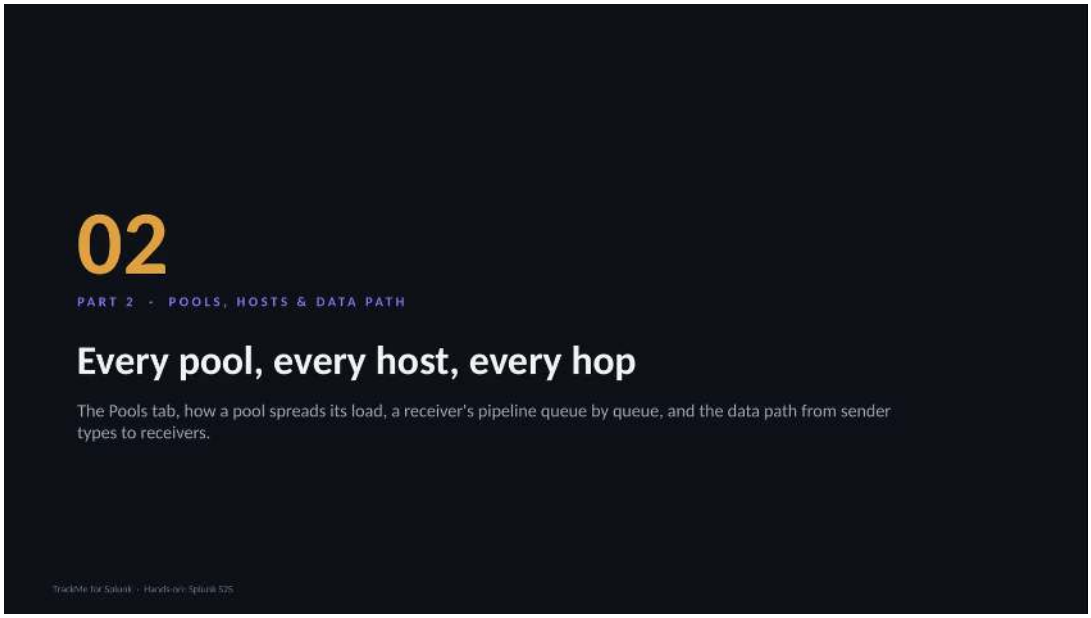
Advanced options stay folded unless a setting needs a fix: the telemetry form, the host patterns and excluded hosts, typed as tokens, and the automatic pools of indexer clusters and intermediate forwarders. Below the clusters, the step shows the pools the tenant starts with: the indexer cluster's two site pools and the automatic pool hec-pool-hf of the four heavy forwarders. Every host carries its pool pill: clicking it assigns the host to another pool, and By the rules gives it back. Nothing needs configuring at creation.

Configure the automatic pool



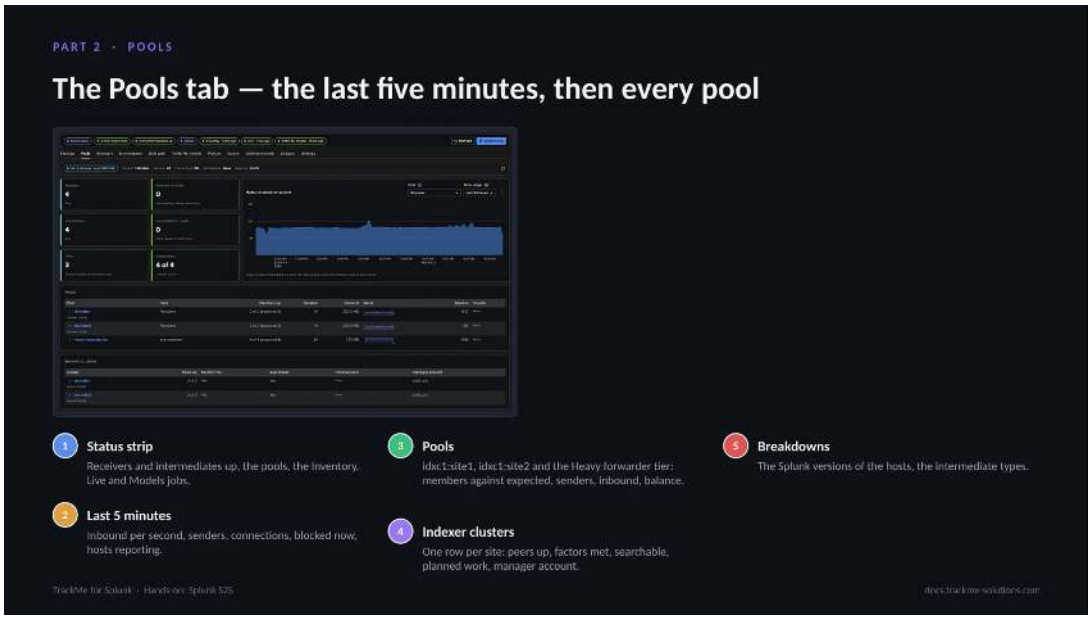
Settings, Pools. The indexer cluster and the heavy forwarder tier are pools from the first run. Configure on the automatic pool hec-pool-hf makes it a configured pool with the same id, so its findings and history continue: here it is named Heavy forwarder tier, given the output group its senders use, which ties the tier to the data path, and its expected size, four. A pool change applies from the next inventory run; Run now on the inventory, then Collect now, shows it at once.

Part 2 — Pools, hosts and data path



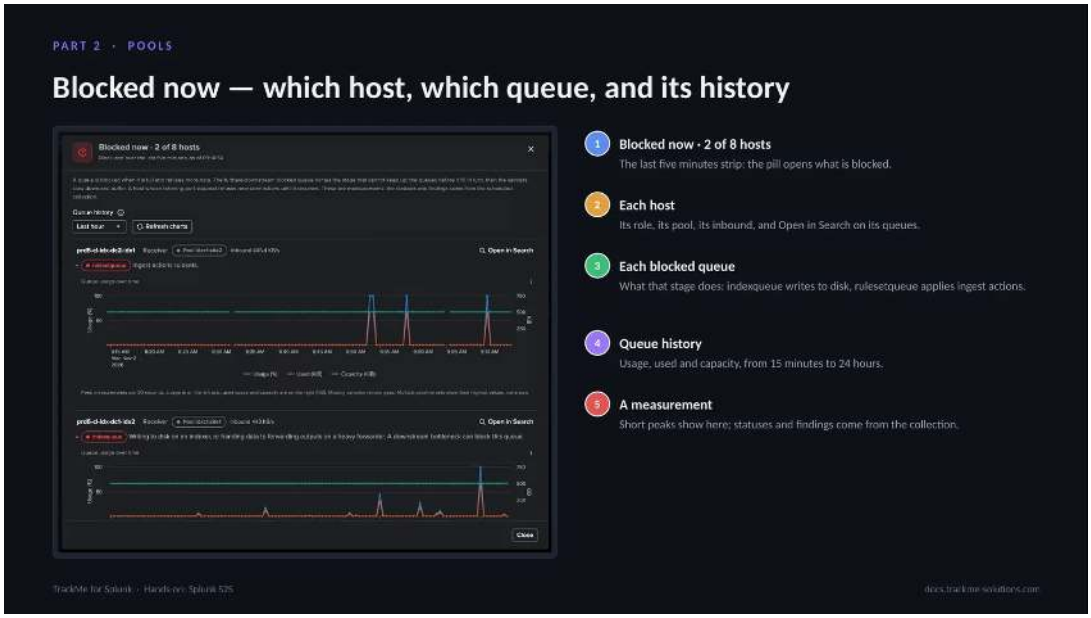
Part two: the workspace. The pools, the receivers and the intermediates, and the data path between them.

The Pools tab



The Pools tab opens on the last five minutes: inbound per second, senders, connections, blocked now and hosts reporting. Then come the host tiles, the pools with their members up against the expected size, their senders, inbound volume, trend and balance: the two site pools idxc1:site1 and idxc1:site2 and the Heavy forwarder tier. The indexer clusters list one row per site with its peers, factors, planned work and manager account. A pool row expands to its members, an indexer cluster pool to its peers.

Blocked now



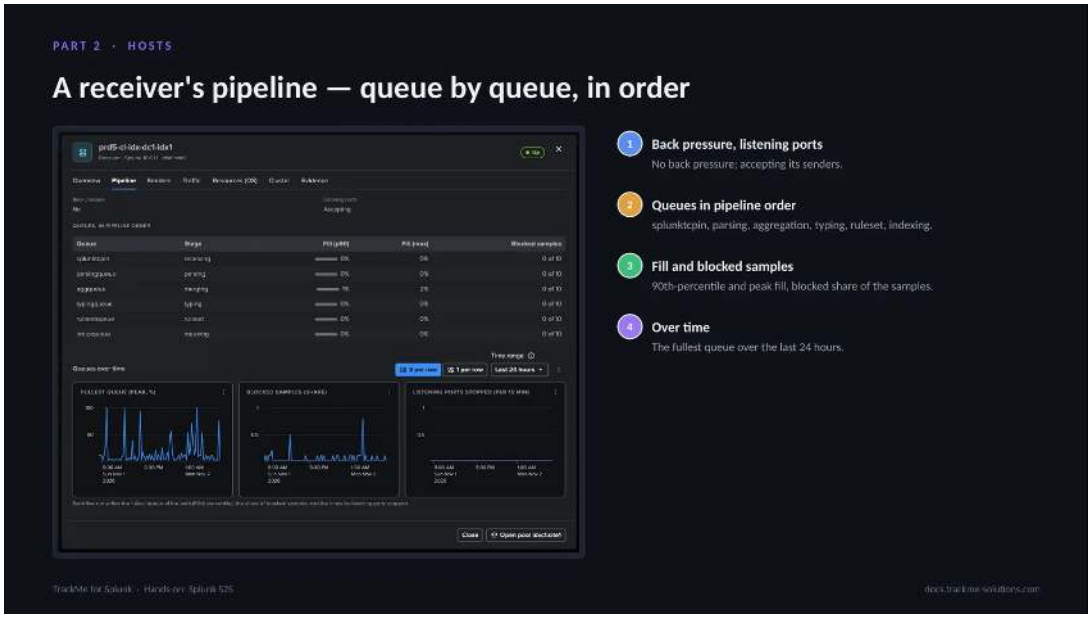
The Blocked now pill of the last five minutes opens what is blocked: each host with its role, its pool and its inbound, every blocked queue with what that stage of the pipeline does, a stopped listening port, Open in Search on the host's queues, and the queue history of each blocked queue, from 15 minutes to 24 hours. Here an indexer of each site shows short peaks of a queue: the index queue on one, the ruleset queue on the other. These are measurements: the statuses and findings come from the scheduled collection.

How the pool spreads its load



The pool modal of the Heavy forwarder tier, Distribution tab. One bar per active member's share of the pool, the fair share, 25 % for four members, and the band of the Receiver imbalance setting. A load balancer that starves one member shows here first, and the policy opens when a member stays below the band. The largest and smallest members are listed with their distance from the fair share.

A receiver's pipeline — queue by queue



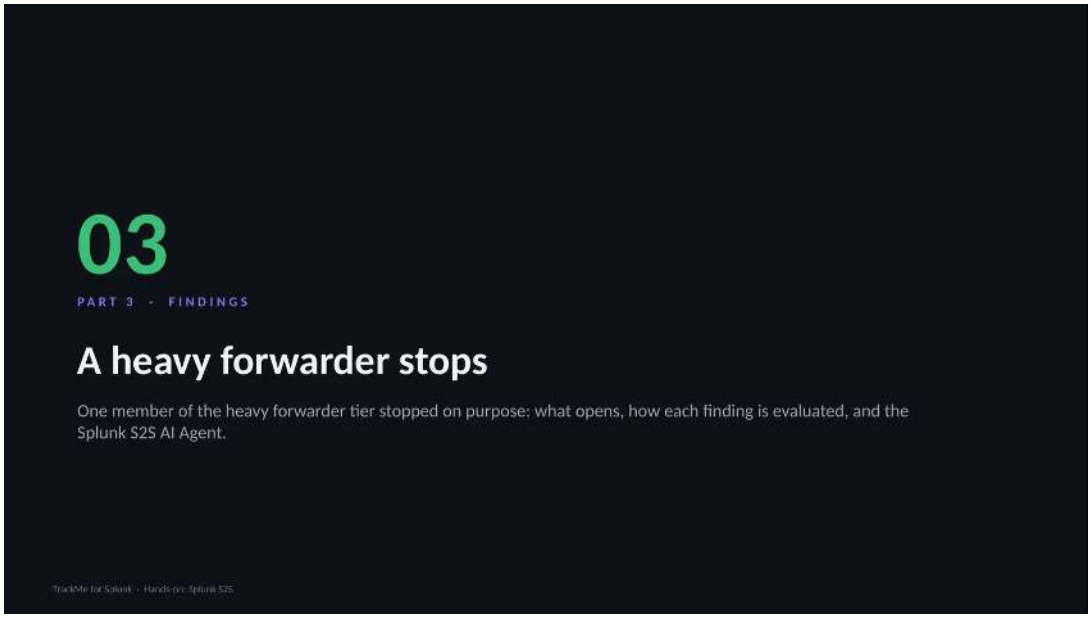
The host modal, Pipeline tab. Each queue of the receiver path, in order, with its fill and its blocked samples. A queue saturated at 80 % or blocked in 30 % of its samples is back pressure: the finding names the stage that cannot keep up, and whether its thread was busy or waiting on the next stage. The chart below follows the fullest queue over time.

The data path



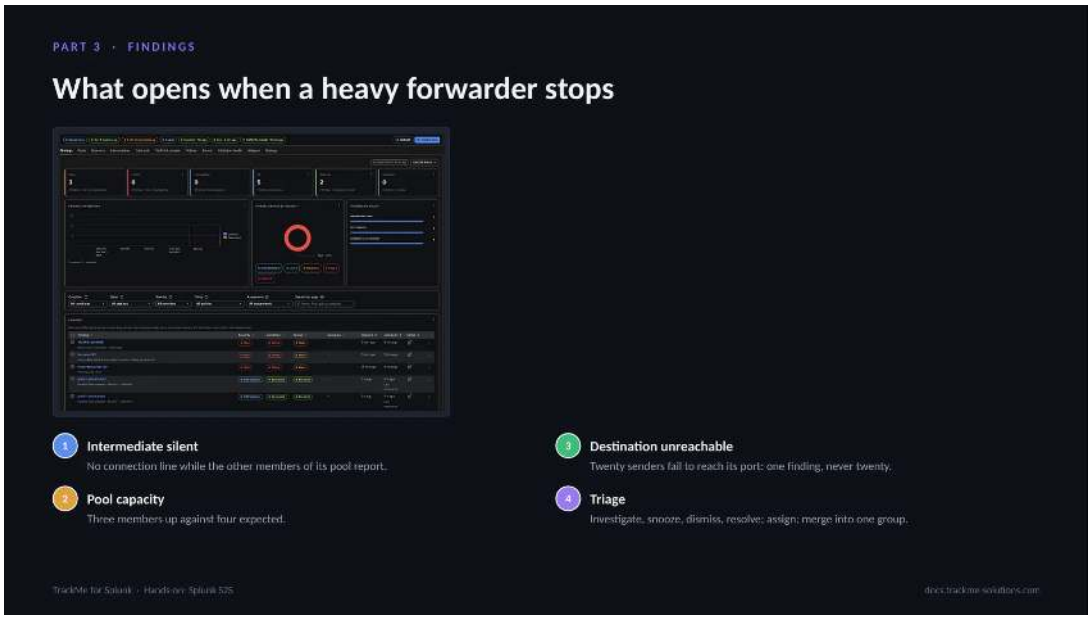
The Data path tab draws the forwarding path from what the receivers report: the universal forwarders into the Heavy forwarder tier, the tier into the two site pools of the indexer cluster, and the full instances sending straight to the indexers. Edges are drawn by volume. A Flows table states every edge in words. An intermediate is tied to its receivers by its output group, never by guessing from a destination address. The topology can be saved to Topology Studio.

Part 3 — Findings



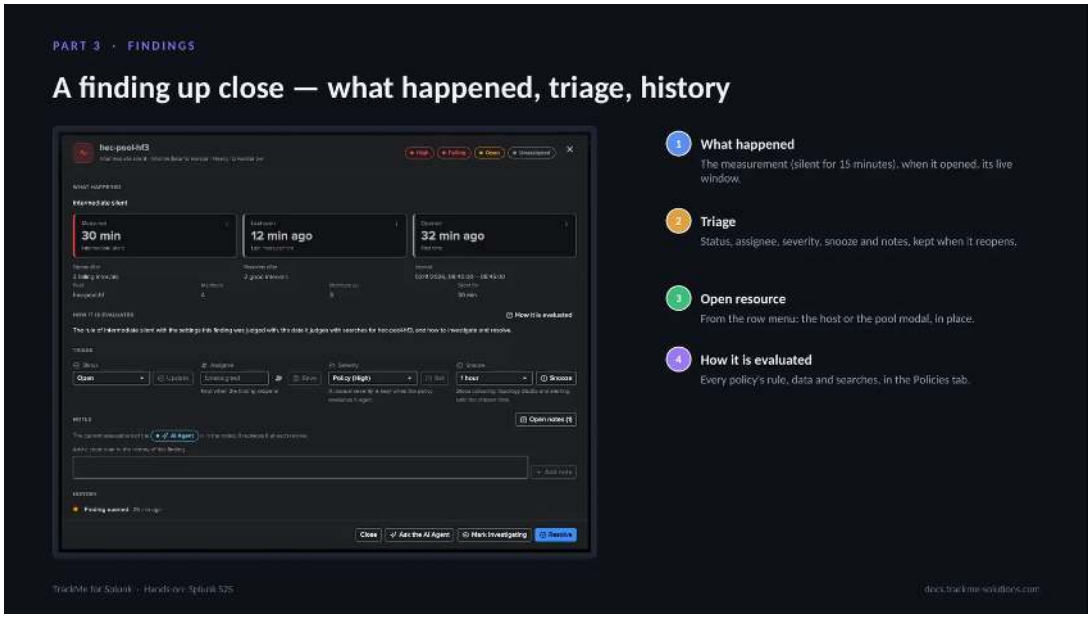
Part three: findings. One heavy forwarder of the tier, hec-pool-hf3, was stopped on purpose for about thirty-five minutes.

What opens when a heavy forwarder stops



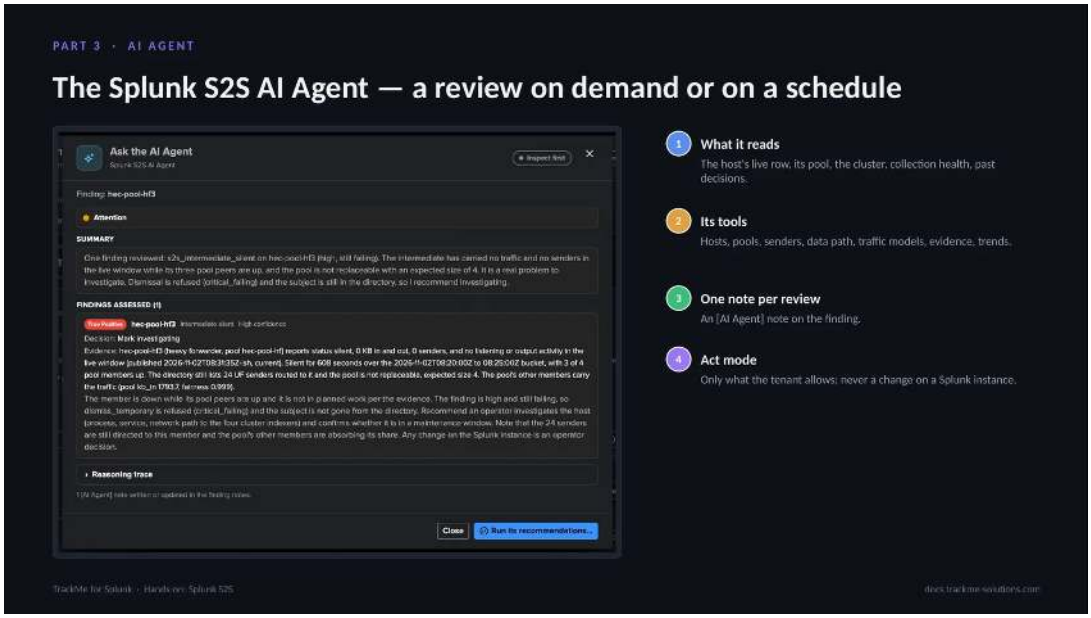
The Findings tab. The stopped heavy forwarder first opens Pool capacity on the Heavy forwarder tier (three members up against four expected), then Intermediate silent on the heavy forwarder and Destination unreachable on its receiving port, measured on the 23 universal forwarders that failed to reach it for 15 minutes or more: one finding on the destination, never one per sender. Each object has its own Finding group; they can be merged by hand when one incident is worked as one. The forwarders load balance to the three other members, so no sender is held: there is no back pressure cascade.

A finding up close



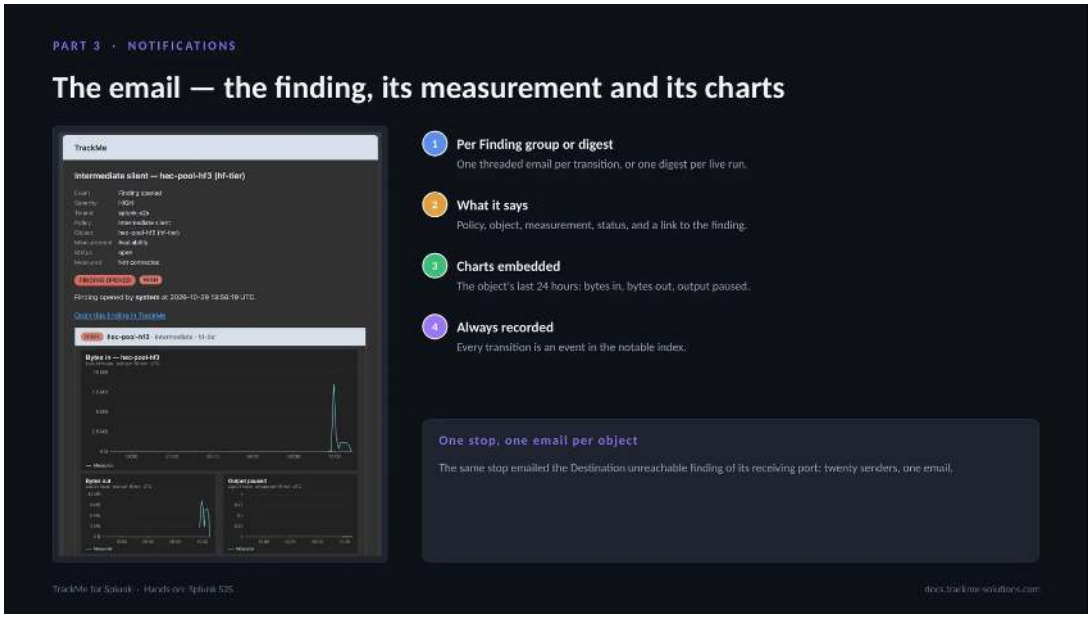
A finding opens with what happened (here the heavy forwarder silent for 10 minutes), when it opened and its live window, its pool with its members up, then its triage — status, assignee, severity, snooze — its notes and its history. Open resource, in the row menu, opens the host or the pool modal in place. How each policy is evaluated — the rule as configured, the data it reads with ready-to-run searches on the tenant's own source, and how to investigate and resolve — is one click away. When the heavy forwarder restarts, the findings recover after two healthy live windows.

The Splunk S2S AI Agent



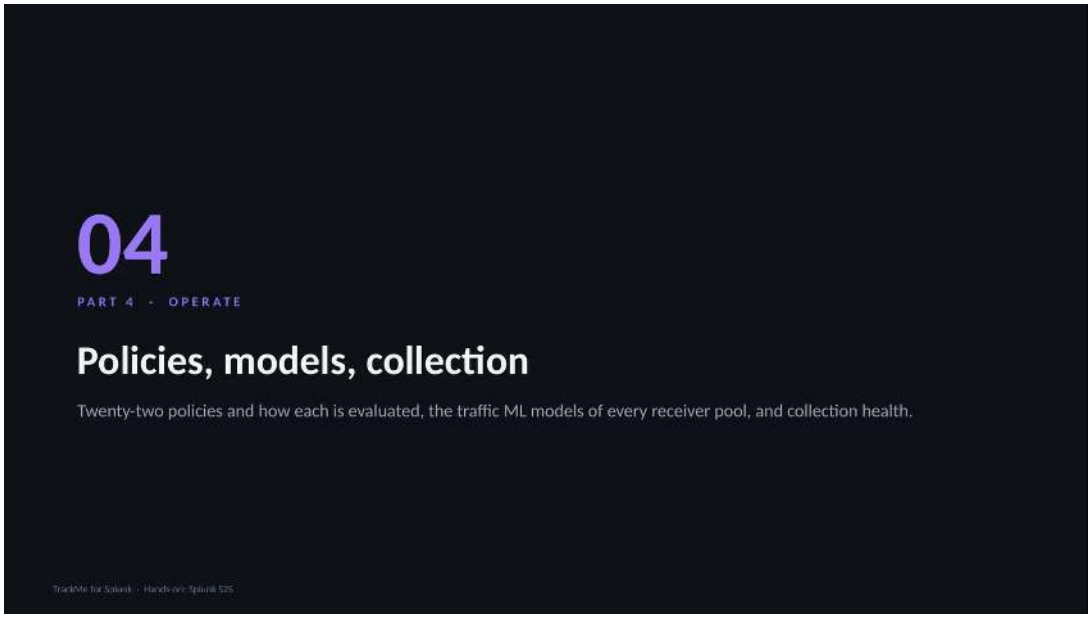
Ask the AI Agent from a finding or a Finding group. The Splunk S2S AI Agent reads the host's live row, its pool, the cluster, collection health and past decisions, and may read further through its tools: hosts, pools, senders, the data path, the traffic models, the evidence and the trends. It knows the pipeline model, the bottleneck and cascade reading, planned cluster work and the TLS and acknowledgement posture. Each review leaves an [AI Agent] note on the finding. Scheduled triage is off by default: hourly, up to ten findings per run, and in act mode only the decisions the tenant allows, never a change on a Splunk instance.

The notification email



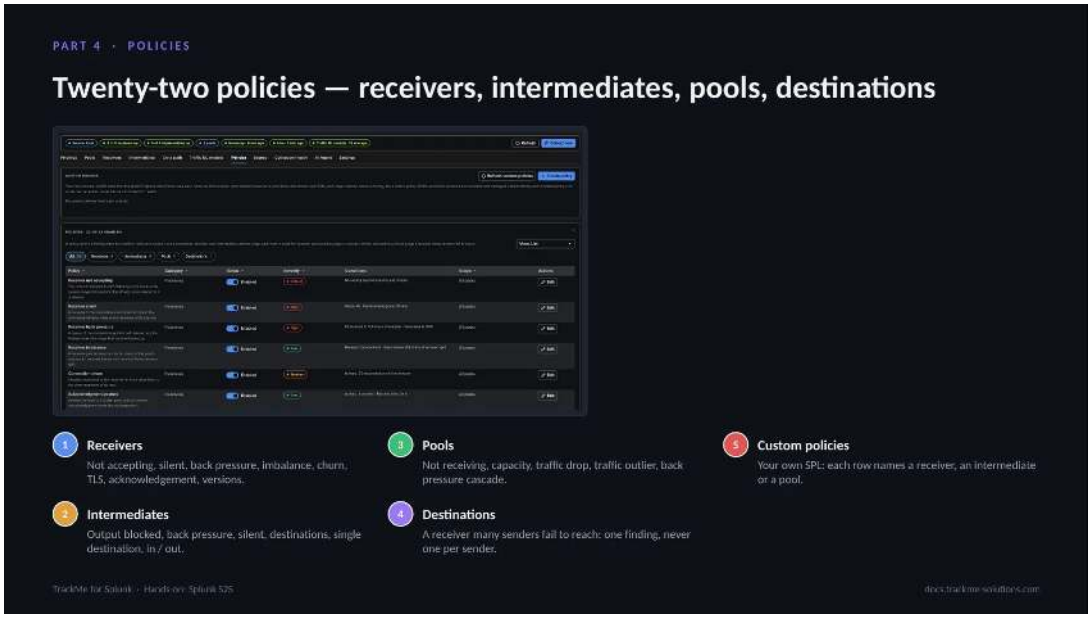
Settings, Notifications sends the transitions of the findings from a minimum severity, for every policy or chosen ones, through an email delivery account: one threaded email per transition of a Finding group, or one digest per live run. Here, the email of the silent heavy forwarder: the policy, the object, its measurement and a link back to the finding, with the intermediate's last 24 hours embedded. The same stop emailed the Destination unreachable finding of its port. Every transition is also an event in the tenant's notable index, whatever the notification settings.

Part 4 — Operate



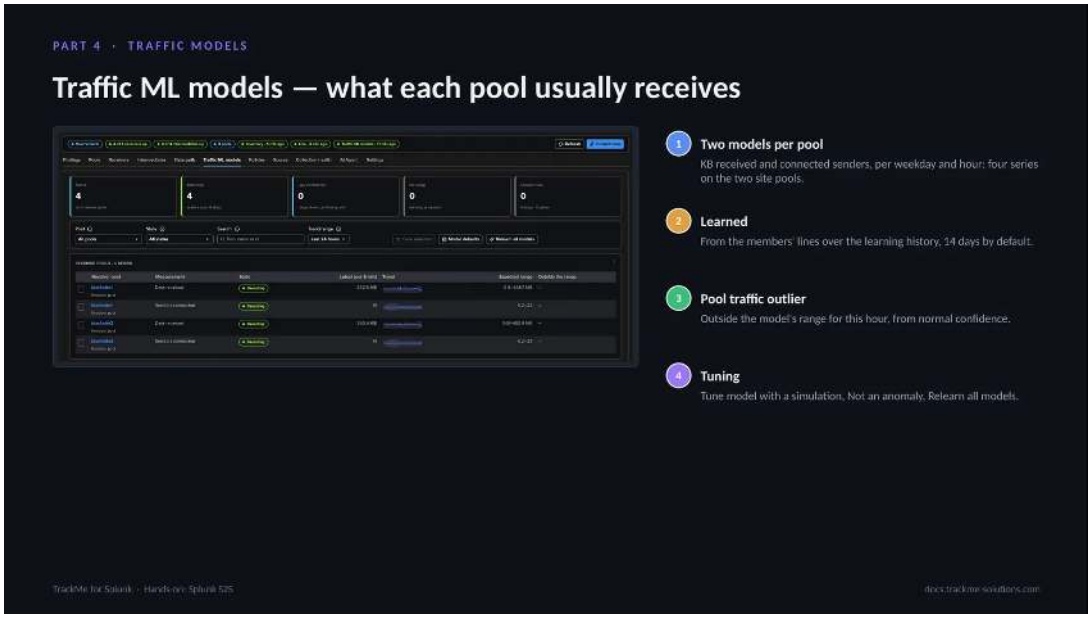
Part four: operate the tenant. The policies, the traffic models and collection health.

Twenty-two policies



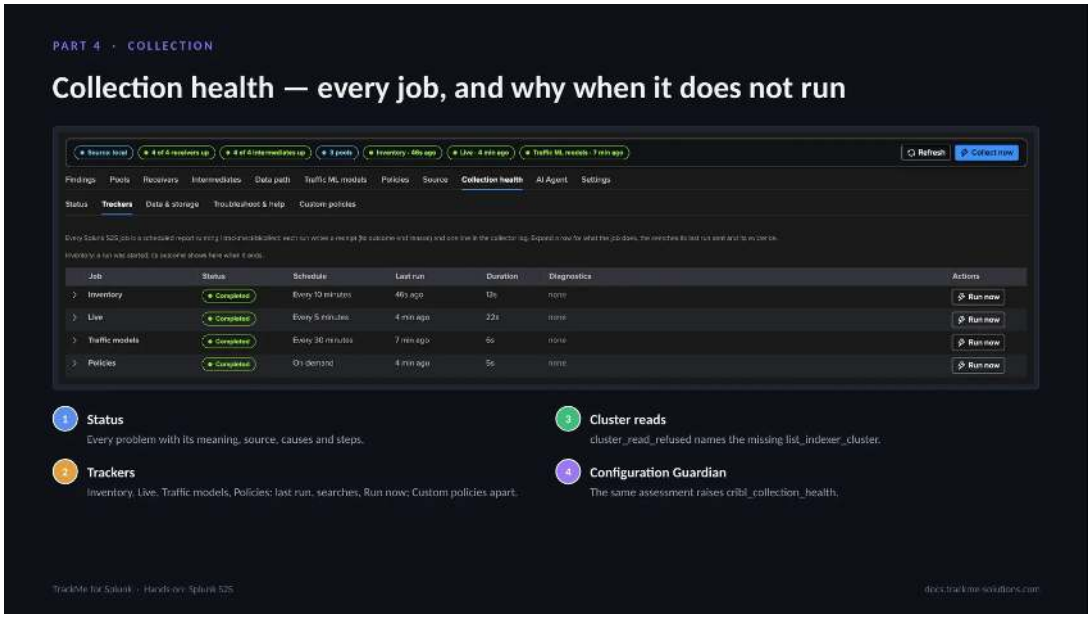
The policy catalogue, by category: receivers, intermediates, pools and destinations. Every policy has its severity, its checks to open and to recover, a scope by pool, a bank holiday hold and How it is evaluated. The ones to know first: Receiver not accepting, Intermediate output blocked and Back pressure cascade, all critical; Pool capacity, the back pressure policies and Destination unreachable, high. Above the catalogue, TrackMe administrators of the tenant write custom SPL policies: each result row names a receiver, an intermediate or a pool.

Traffic ML models



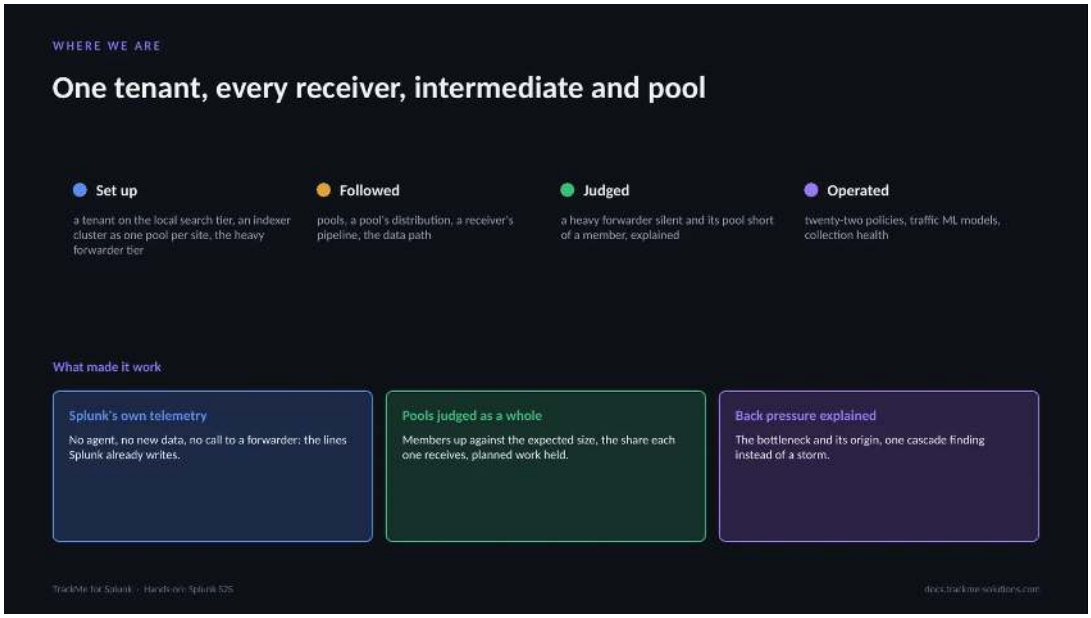
Every receiver pool has two traffic models, KB received and connected senders, learned per weekday and hour over the model learning history, 14 days by default, and refreshed twice an hour: here four series on the two site pools. Pool traffic outlier opens when a pool leaves its range for the hour, from normal confidence. Tune model simulates past windows; Not an anomaly on a finding teaches the model; administrators set the model defaults and relearn all models. Pool traffic drop, the same hour's median over seven days, needs no model.

Collection health



Collection health shows whether the tenant collects as configured: every problem with its meaning, source, causes and steps, and each job — inventory, live, traffic models, policies — with its last run, the searches it sent and Run now; the runs of each custom policy have their own view. The cluster reads need the read-only list_indexer_cluster capability: without it Splunk answers nothing, and Collection health names the missing capability. The Configuration Guardian raises the same assessment.

Where we are



A Splunk S2S tenant set up on a deployment with a four-peer indexer cluster on two sites and a heavy forwarder tier: its cluster found without configuration as one pool per site, its tier an automatic pool made a configured one. The pools, what is blocked, a pool's distribution, a receiver's pipeline and the data path followed. A heavy forwarder stopped, the findings it opened explained and emailed. And the policies, traffic models and collection health that run the tenant. What made it work: Splunk's own telemetry, pools judged as a whole, and back pressure traced to its origin.

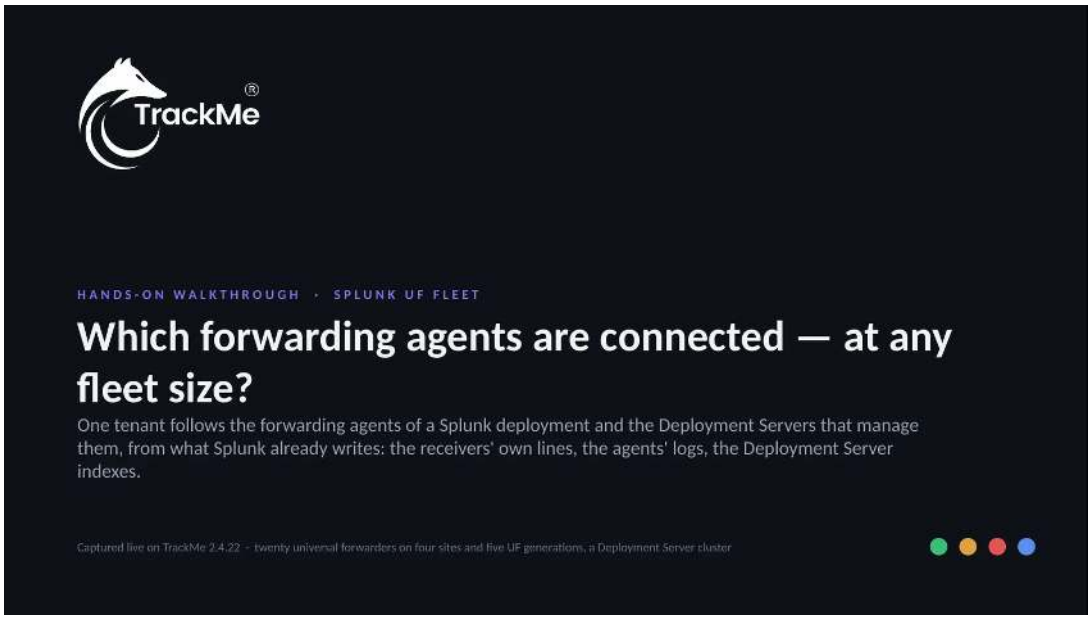
2.1.13 Splunk UF Fleet

Follow every forwarding agent of a Splunk deployment and the Deployment Servers that manage them, from what the receivers, the agents and the Deployment Servers already log: the Deployment Server API accounts, the scope and the discovery, enrichment from a CMDB lookup, groups proposed from the server classes and the enrichment, the agents that must stay connected and the notifications; the fleet, the agents, one agent up close and the Deployment Servers' pace; then four agents of one site stopped on purpose, a mass disconnect per cohort, an expected agent missing, a receiver the agents fail to reach and the Splunk UF Fleet AI Agent.

`Download the deck (PDF) · PowerPoint`

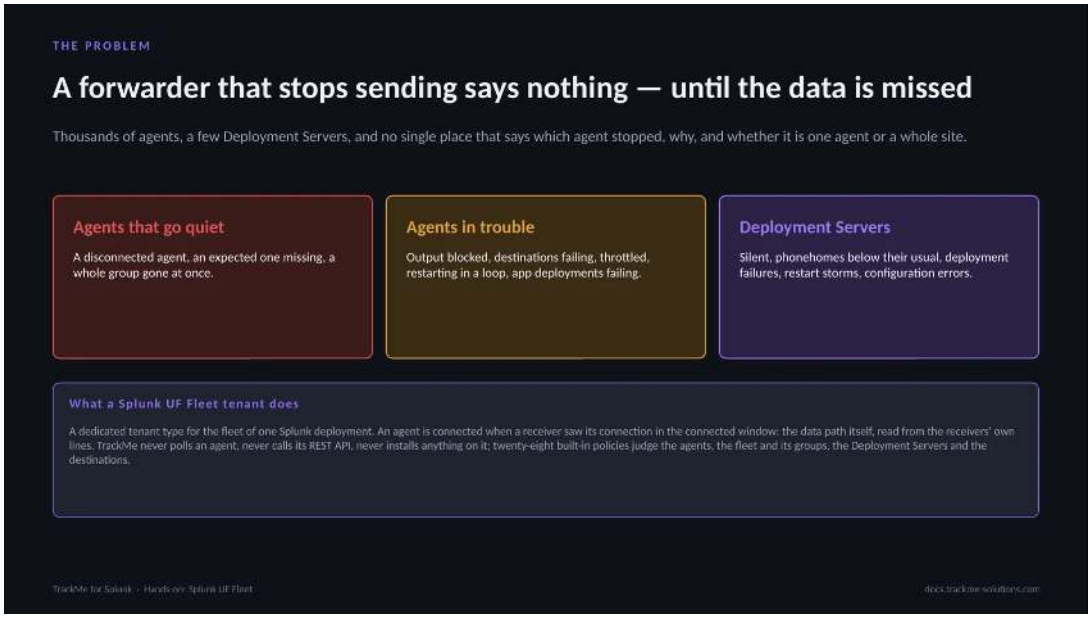
about 30 minutes · 26 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



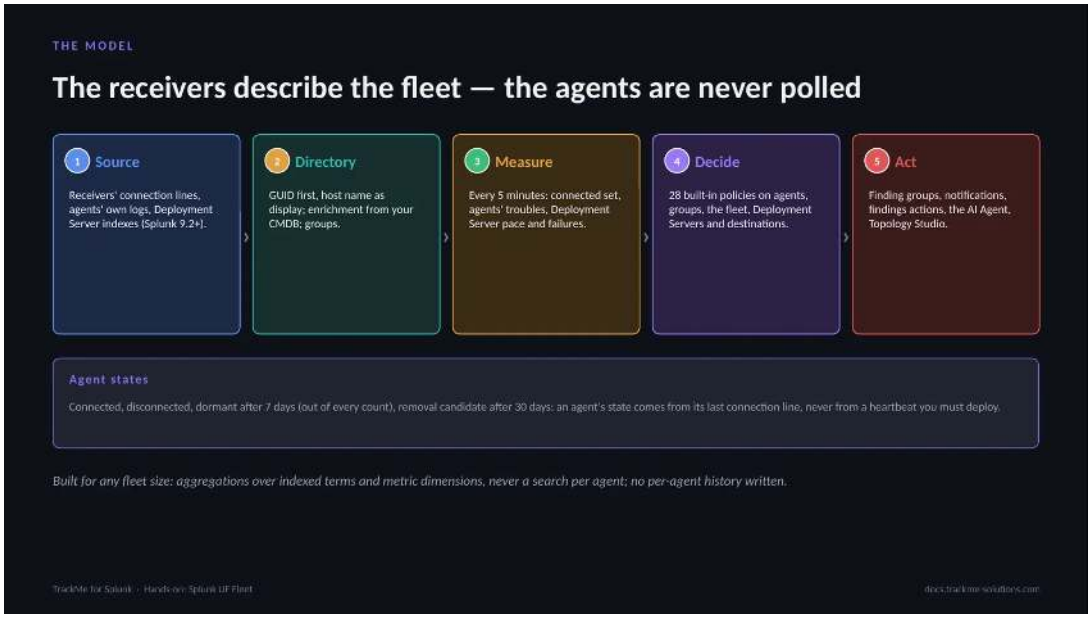
This walkthrough covers the Splunk UF Fleet tenant, generally available in TrackMe 2.4.22. Everything was captured live on a fleet of twenty universal forwarders named after their site and service — London, Paris, New York and a development site — on five UF generations from 9.4 to 10.6, managed by a cluster of two Deployment Servers. In four parts: create the tenant, configure it, follow the fleet and its Deployment Servers, and read the findings of agents stopped on purpose.

Why a dedicated Splunk UF Fleet tenant



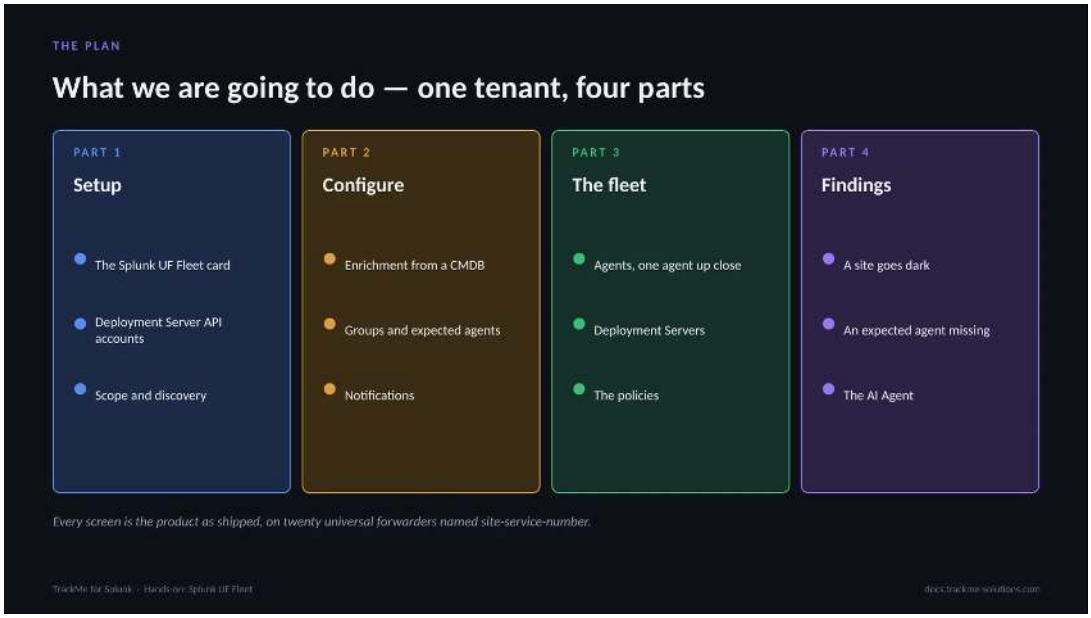
A Splunk administrator asks the same fleet questions every day: how many agents do I have, on which versions and platforms, which are connected, disconnected or dormant, and which of the agents that must stay connected are not. Which agents are blocked, throttled, or failing their destinations or their app deployments? Is each Deployment Server answering its clients at its usual pace? The Splunk UF Fleet tenant answers them with findings raised by twenty-eight built-in policies. It only reads: nothing is installed on an agent, and no agent is ever polled.

How a Splunk UF Fleet tenant works



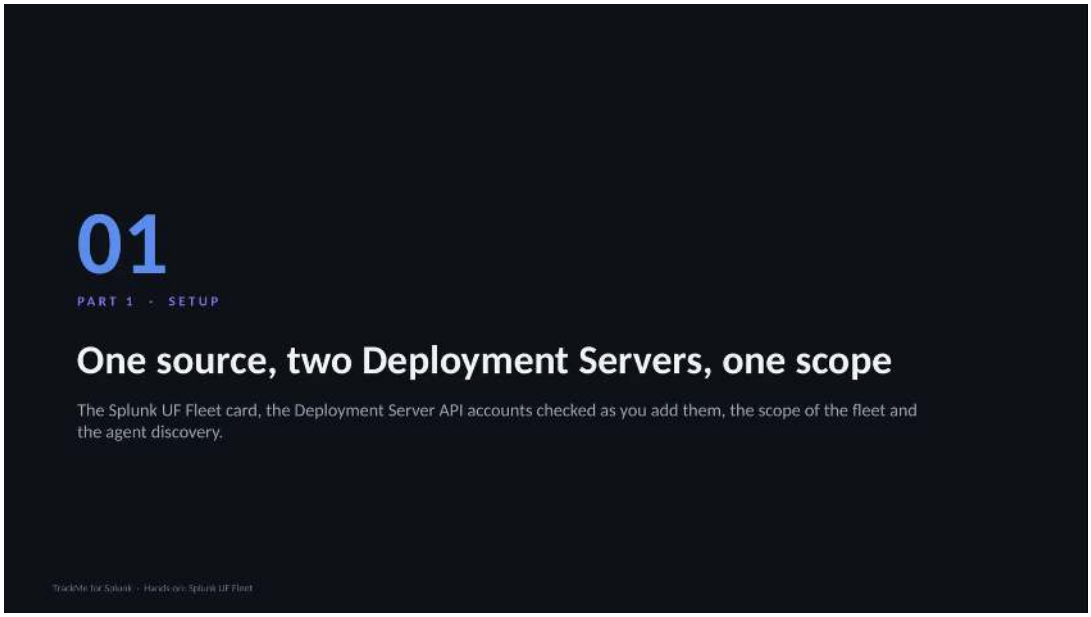
The receivers write a connection line for every sender, the agents forward their own logs, and the Deployment Servers write their phonehomes, their clients and their app events. TrackMe merges the three into one agent directory, reads the connected set every five minutes, and judges the agents, the groups, the fleet and the Deployment Servers. A mass disconnection is one finding on the group, never hundreds of agent findings.

The plan



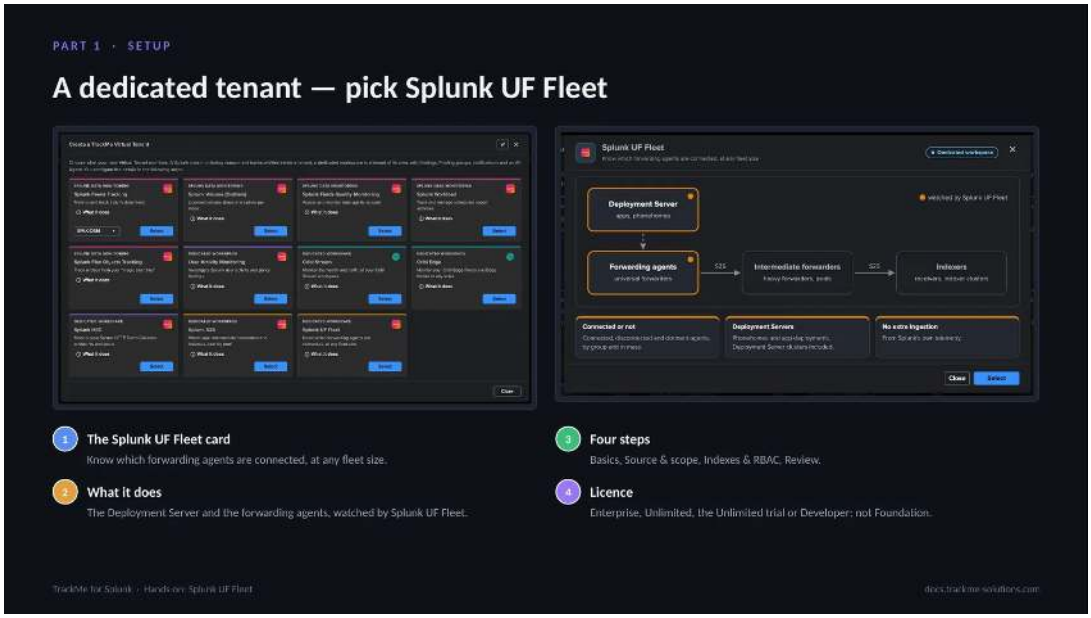
Four parts. Setup: the wizard, the Deployment Server API accounts and the scope. Configure: enrichment from a CMDB, groups proposed from the Deployment Servers and the enrichment, the expected agents and the notifications. The fleet: the Fleet tab, the agents, one agent up close, the Deployment Servers and the policies. Findings: agents stopped on purpose, a site's mass disconnect, an expected agent missing and the AI Agent.

Part 1 — Setup



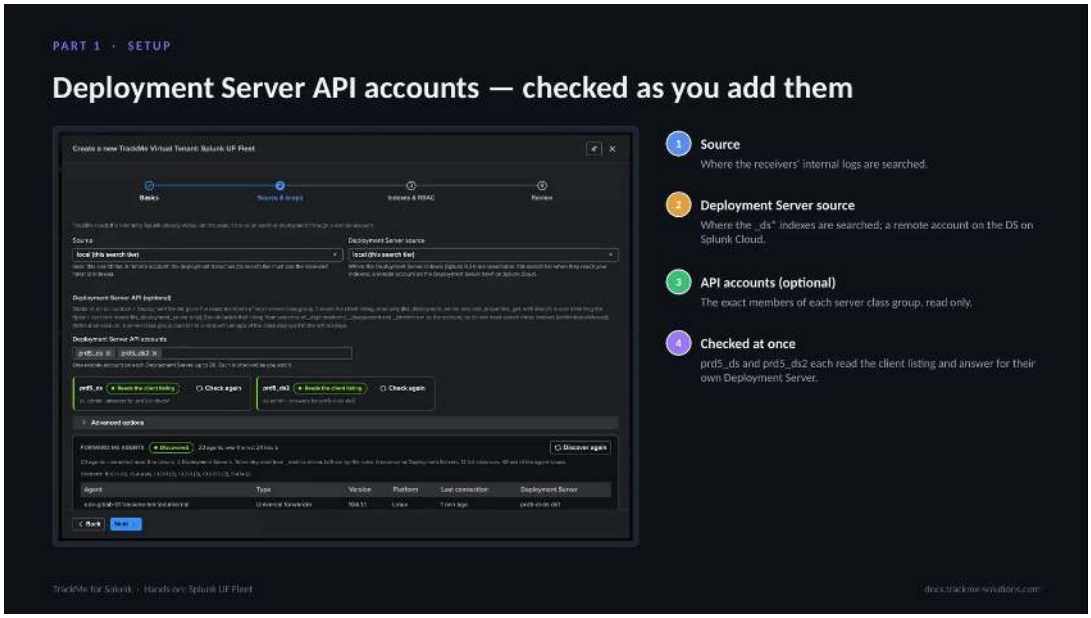
Part one: setup. A Splunk UF Fleet tenant reads one deployment: its local search tier, or another deployment through a TrackMe remote account.

A dedicated tenant — pick Splunk UF Fleet



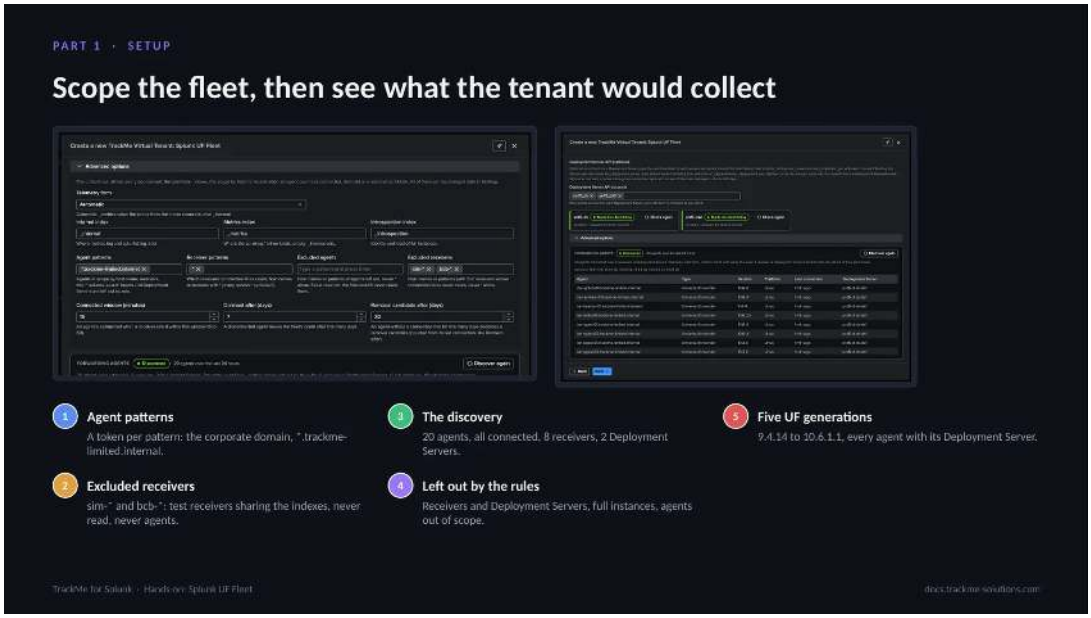
Virtual Tenants, Actions, Create a tracking tenant, then the Splunk UF Fleet card. What it does draws the forwarding path, with the Deployment Server and the forwarding agents watched. The receiving side, the intermediate forwarders and the indexers, has its own tenant type, Splunk S2S; when both read the same source, Topology Studio links them.

Deployment Server API accounts



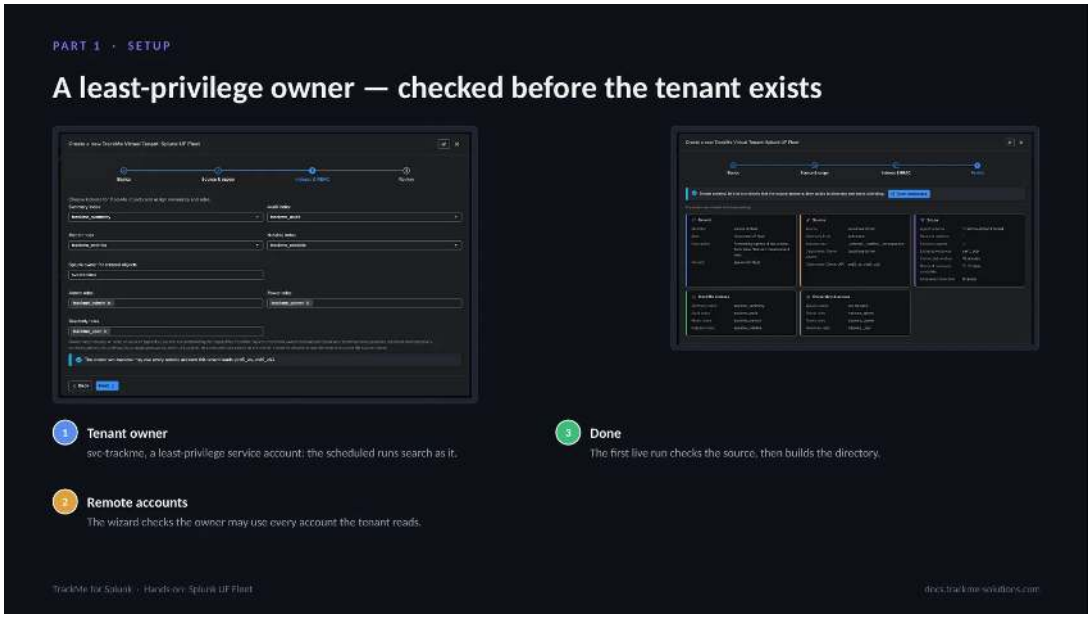
Source and scope. The Deployment Server API accounts are optional: an account on a Deployment Server gives the exact members of each server class group, from its client listing. Its user needs list_deployment_server and rest_properties_get, never an admin role, and its role must be allowed to search the Deployment Server indexes: Splunk builds the client listing from them, as the account. Each account is checked as it is added: here prd5_ds and prd5_ds2 each read the client listing of their own Deployment Server. Without one, a server class group counts the clients with an app of the class deployed in the last 60 days.

Scope the fleet, then see what the tenant would collect



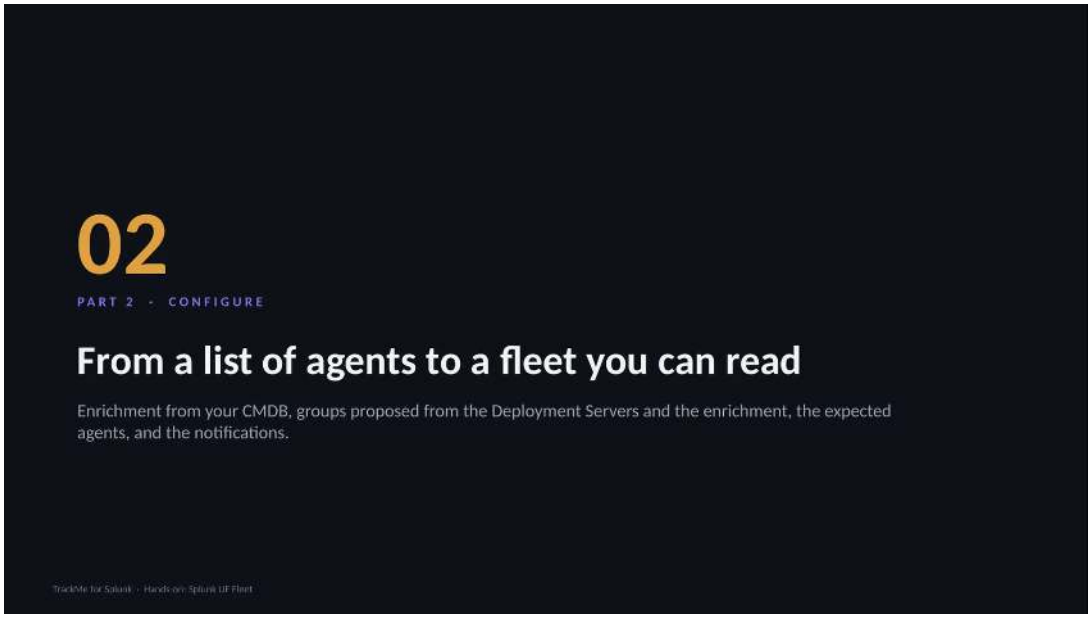
Advanced options: the agent patterns scope the tenant to the corporate domain, here *.trackme-limited.internal, and the excluded receivers leave out two families of test receivers sharing the same indexes; each pattern is a token, typed and added with Enter. Then the connected window and the dormancy and removal delays. The discovery runs the tenant's own rules with your rights over the last day: here twenty universal forwarders on five UF generations, all connected, and what the rules left out.

A least-privilege owner



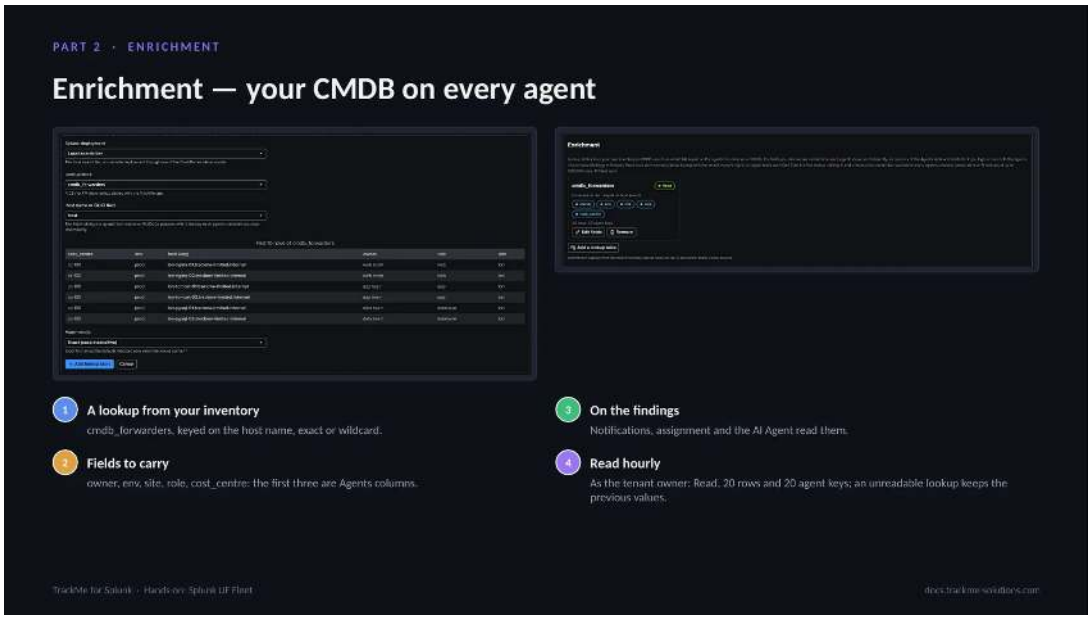
Indexes and RBAC, then Review and Done. The owner must be able to use every remote account the tenant reads, the Deployment Server API accounts included; otherwise the wizard names the fix before the save. The Review lists the source, the scope, the Deployment Server API accounts, the indexes and the ownership. The first live run starts at once.

Part 2 — Configure



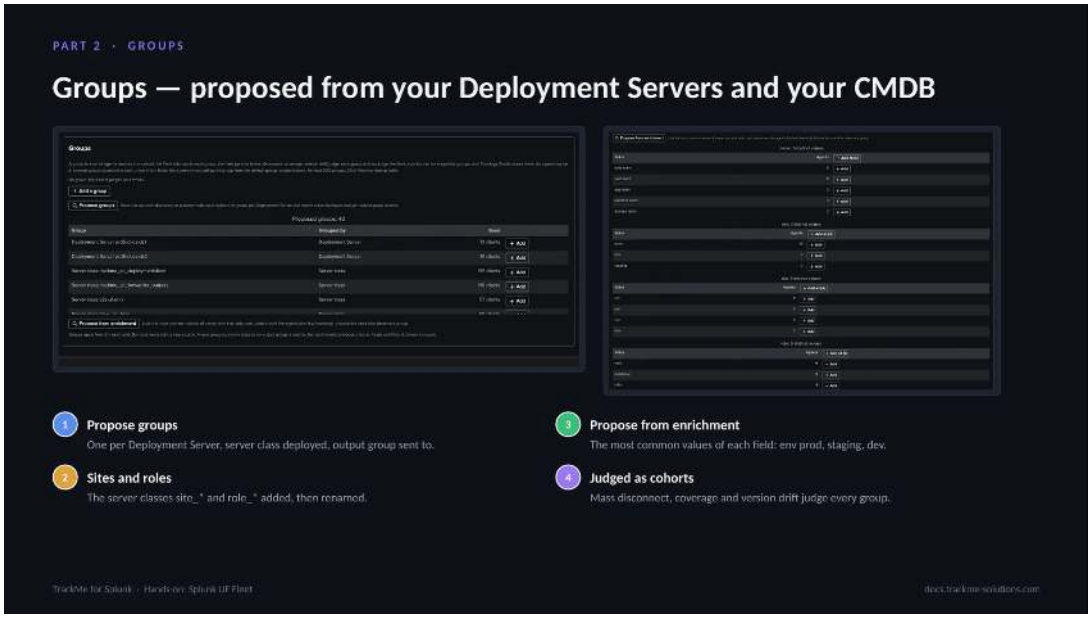
Part two: configure the tenant. Three settings turn a list of agents into a fleet: enrichment, groups and expected agents. Notifications then deliver its findings.

Enrichment — your CMDB on every agent



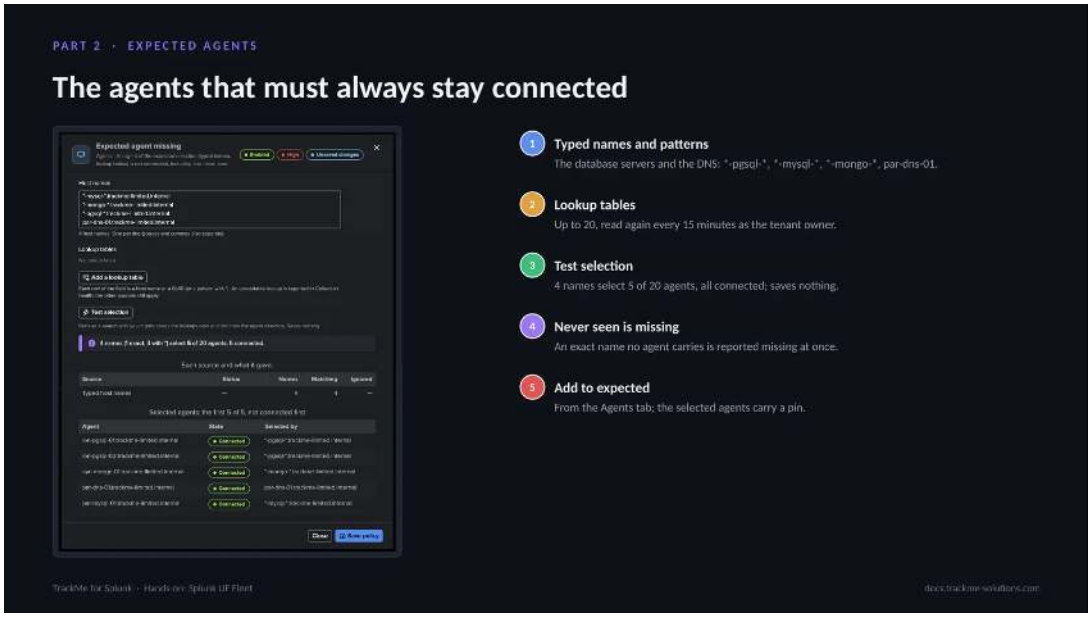
Settings, Enrichment. Up to five lookup tables of up to 500,000 rows, local or through a remote account, carry fields from your own inventory onto each agent: here the owner, the environment, the site, the role and the cost centre from a CMDB lookup, read by the last inventory, 20 rows for 20 agents. Enrichment applies from the next inventory and never starts a new source.

Groups — proposed from your Deployment Servers and your CMDB



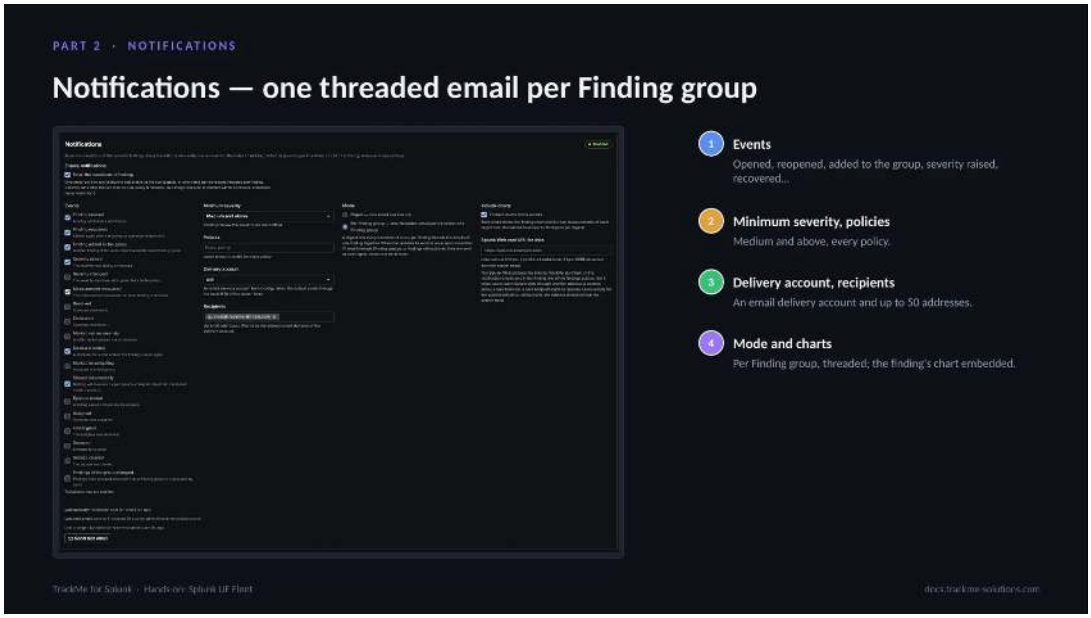
Settings, Groups. A group reads a set of agents as one cohort: the Fleet tab counts it, the fleet policies judge it, a policy can be scoped to it, and Topology Studio draws it. Groups are proposed from the Deployment Server classes and from the enrichment values: here four sites and five roles from the server classes, and three environments from the CMDB. A new group is pending until the next collection; Run now on the inventory shows it at once.

The agents that must always stay connected



The expected agents are selected in the Expected agent missing policy: typed host names or GUIDs with wildcards, and lookup tables. Here the database servers and the Paris DNS server. Test selection runs as a search with your rights and shows the selected agents by state. Dormancy never closes an expected agent's finding.

Notifications — one threaded email per Finding group



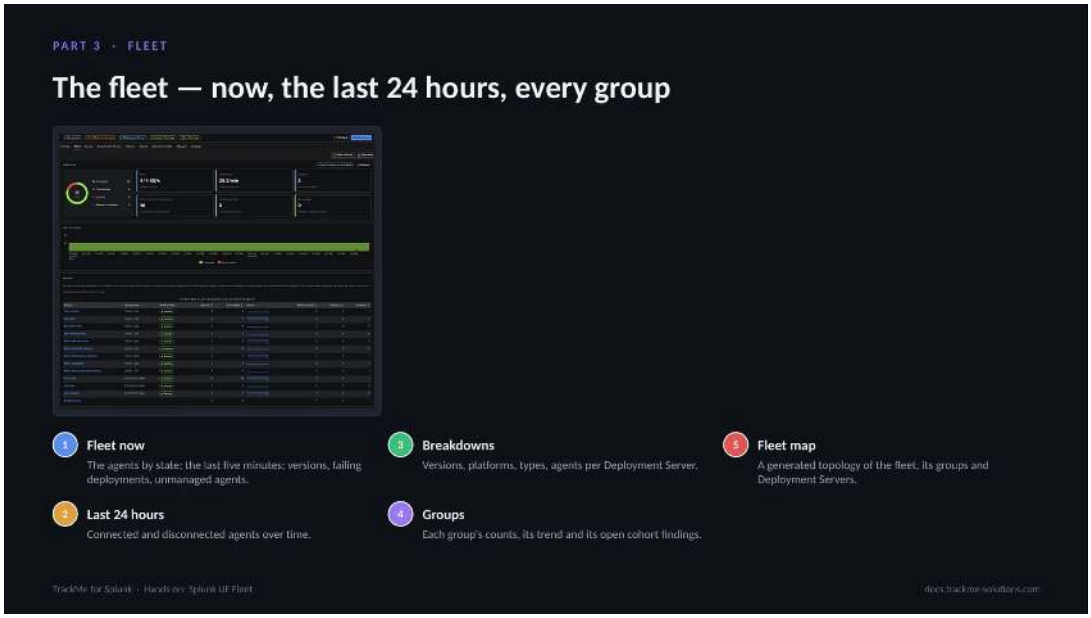
Settings, Notifications: the events notified, the minimum severity, the policies, the email delivery account and the recipients, and the mode — one digest per live run, or one threaded email per transition of a Finding group — with the finding's chart and key measurements embedded. Every transition is also written as an event to the tenant's notable index, whatever these settings.

Part 3 — The fleet



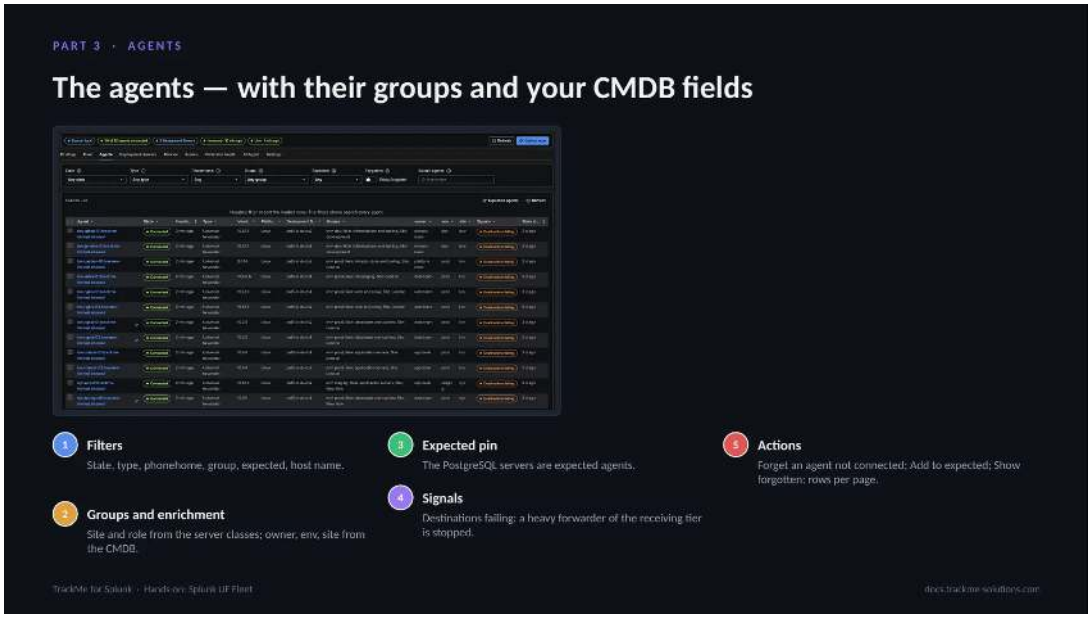
Part three: the workspace. The fleet, its agents, its Deployment Servers and its policies.

The fleet — now, the last 24 hours, every group



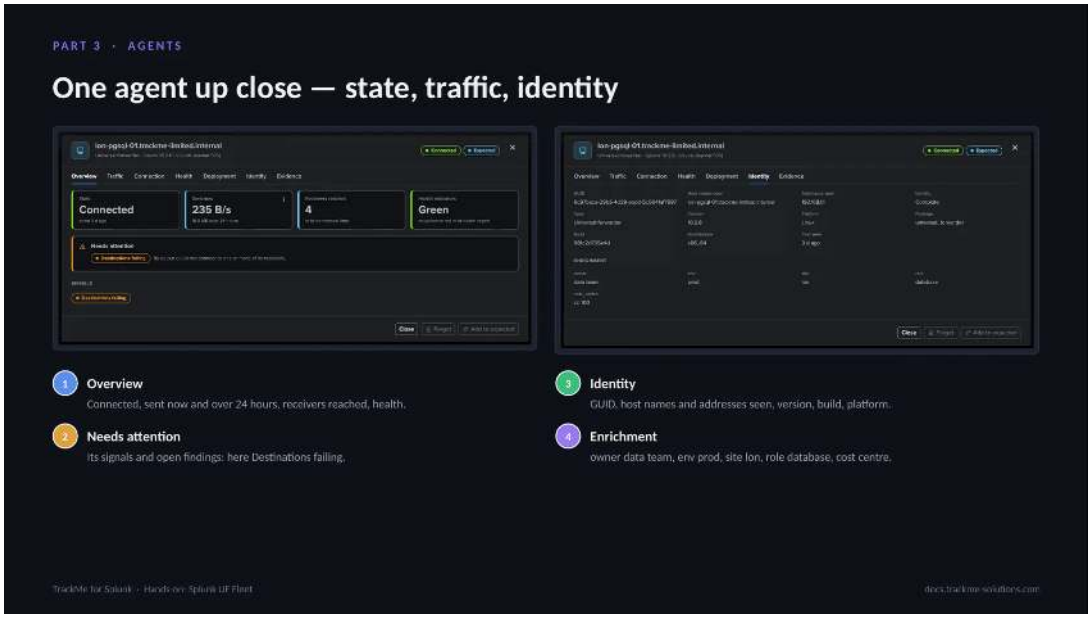
The Fleet tab shows every figure once: the agents by state, the last five minutes (data sent, phonehomes, the receivers that saw the fleet), the fleet's management (the agents not on the most common version, the deployments failing, the agents served by no Deployment Server), the last 24 hours, the breakdowns by version, platform, agent type and Deployment Server, each opening the Agents tab filtered, the groups with their membership state, counts and open cohort findings, the recently disconnected agents, and the Fleet map, a generated topology of the fleet, its groups and its Deployment Servers. Here the Paris site is down: 16 of 20 agents connected.

The agents — with their groups and your CMDB fields



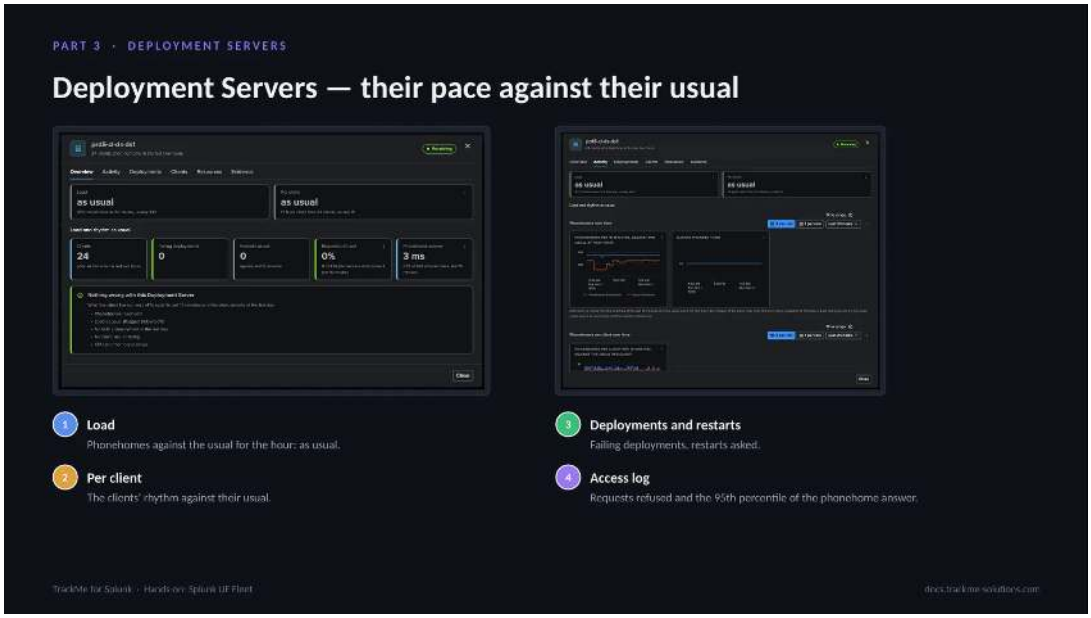
The Agents tab, paged on the server, with filters on state, type, phonehome, group, expected and host name; forgotten agents are listed only with Show forgotten. Each agent with its state, heartbeat, type, version, platform, Deployment Server, groups, the first three enrichment fields and its signals: here Destinations failing on every agent, while a heavy forwarder of the receiving tier is stopped. A signal pill opens its facts and evidence; Forget, for an agent not connected, and Add to expected act on the selection, and the footer sets the rows per page.

One agent up close — state, traffic, identity



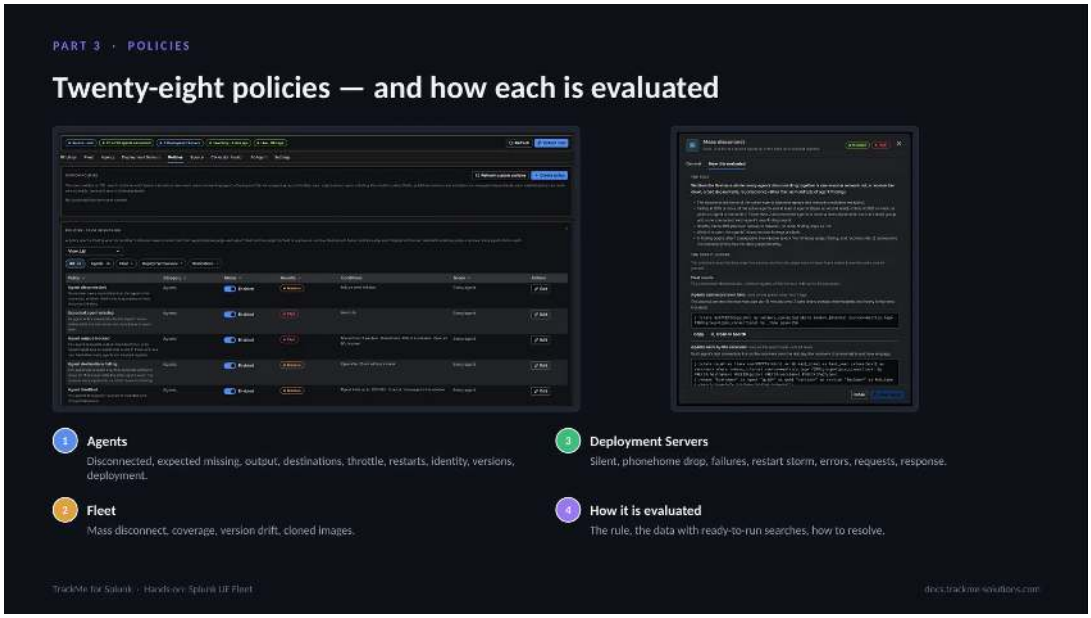
The agent modal: Overview, Traffic, Connection, Health, Deployment, Identity and Evidence. Its Overview shows what needs attention, here Destinations failing; Forget and Add to expected are disabled with their reason for a connected, expected agent. The Identity tab shows the host names and GUIDs the agent was seen with, and the enrichment fields from your CMDB. Its charts read Splunk's own lines on demand: no per-agent history is written, whatever the fleet size.

Deployment Servers — their pace against their usual



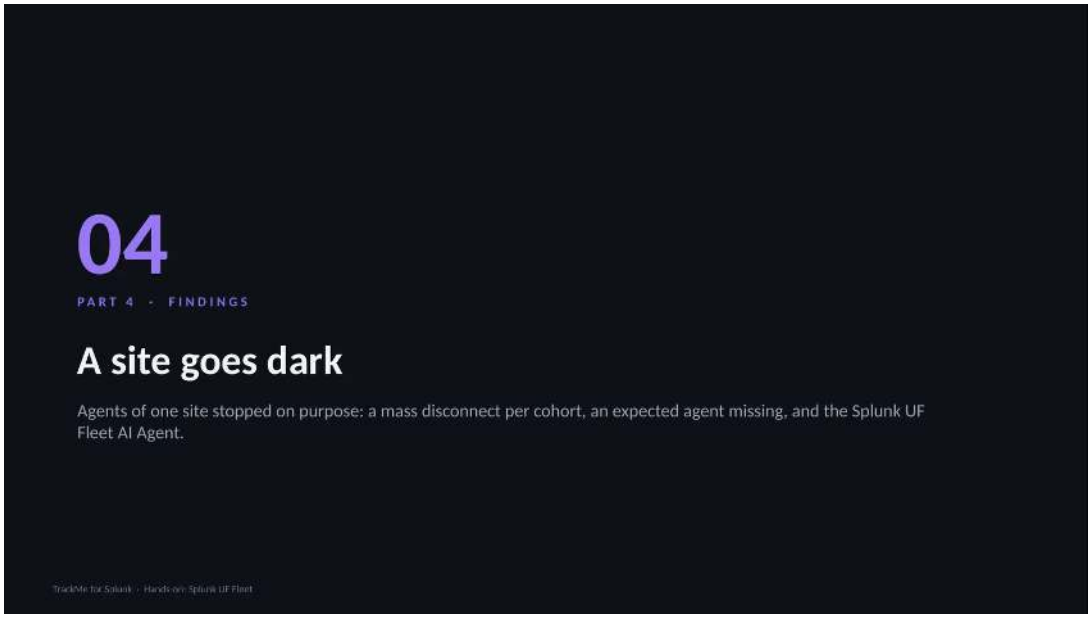
The Deployment Server is a first-class subject: its clients, phonehomes, pace against its usual, failing deployments, restarts asked, CPU and memory. Load and Per client read together: load up with a stable rhythm means more clients; both up, clients phoning home more often; load down with a stable rhythm, clients that left.

Twenty-eight policies — and how each is evaluated



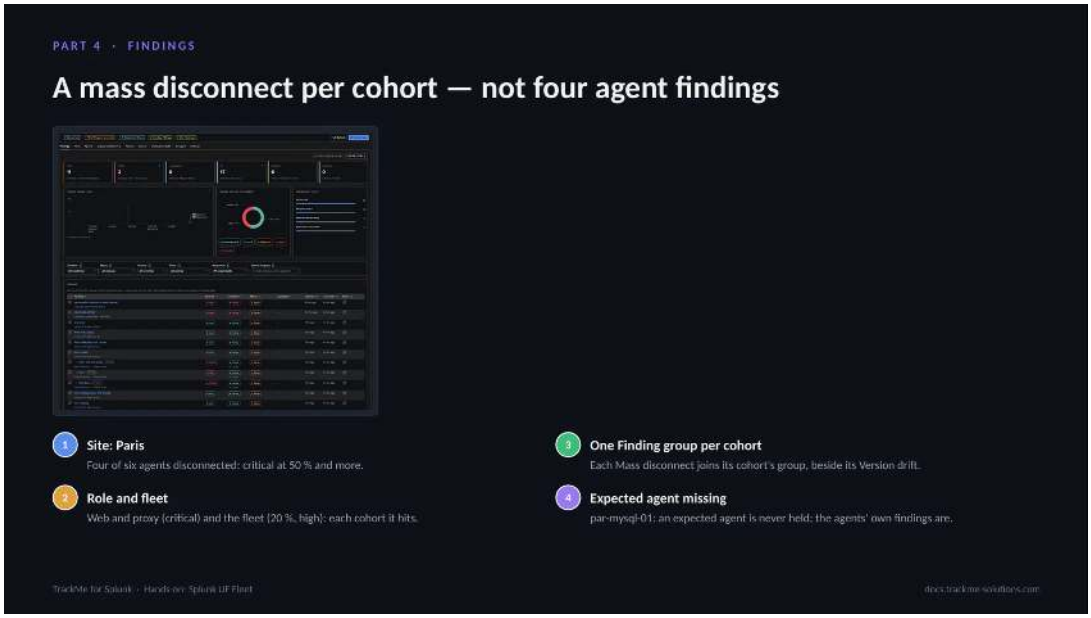
The policy catalogue, by category: agents, the fleet, Deployment Servers and destinations. Every policy has its severity, its checks to open and to recover, a scope by group, a bank holiday hold and How it is evaluated. Mass disconnect opens when twenty per cent of a group, and at least two agents, disconnect together, critical at half; the agents' own findings are held while it is open. Above the catalogue, TrackMe administrators of the tenant write custom SPL policies.

Part 4 — Findings



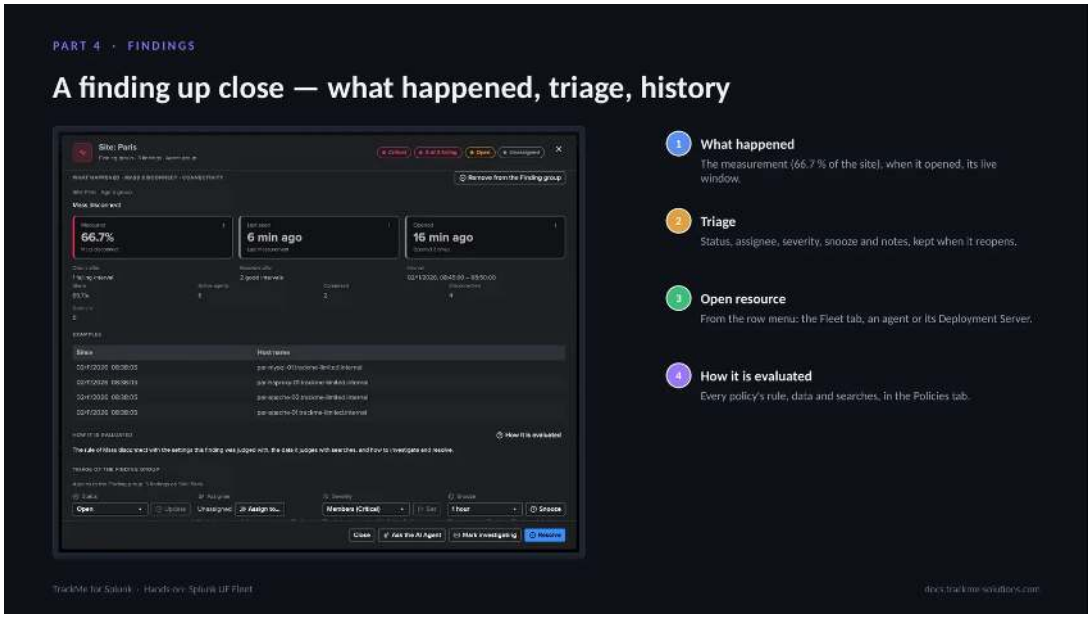
Part four: findings. Four agents of the Paris site were stopped on purpose, one of them an expected database server.

A mass disconnect per cohort — not four agent findings



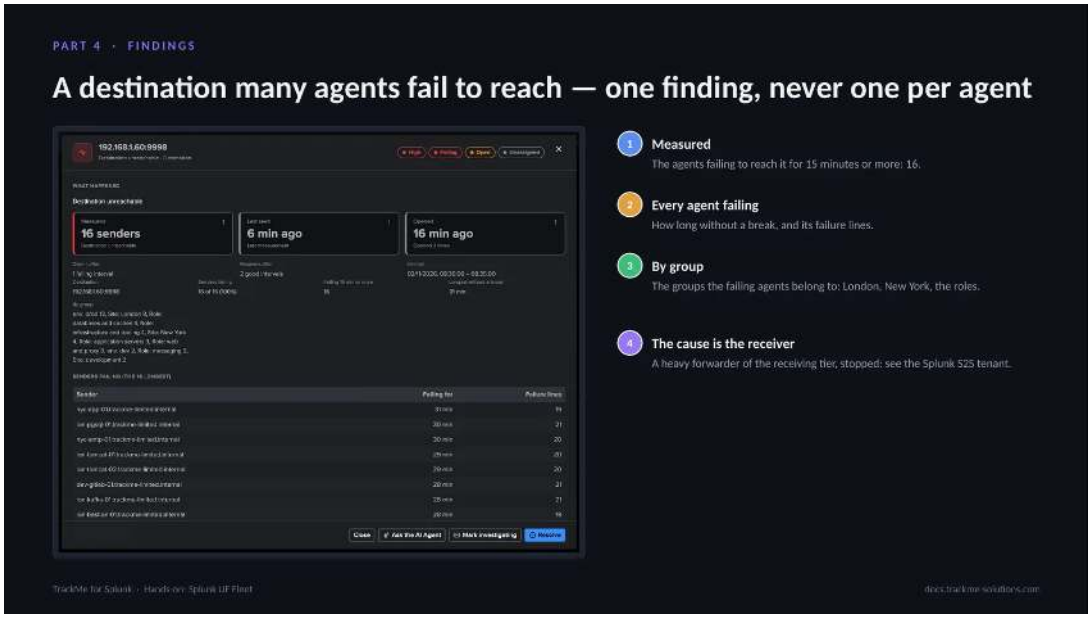
Four agents of the Paris site stopped together. Mass disconnect opens on each cohort they are a large share of: the Paris site group, critical as two thirds of the site is down, the web and proxy role group, critical too, and the fleet at twenty per cent, high; each joins the Finding group of its cohort, beside its Version drift finding. The agents' own disconnection findings are held, never opened one by one. par-mysql-01 is an expected agent: Expected agent missing opens for it whatever the cohort.

A finding up close



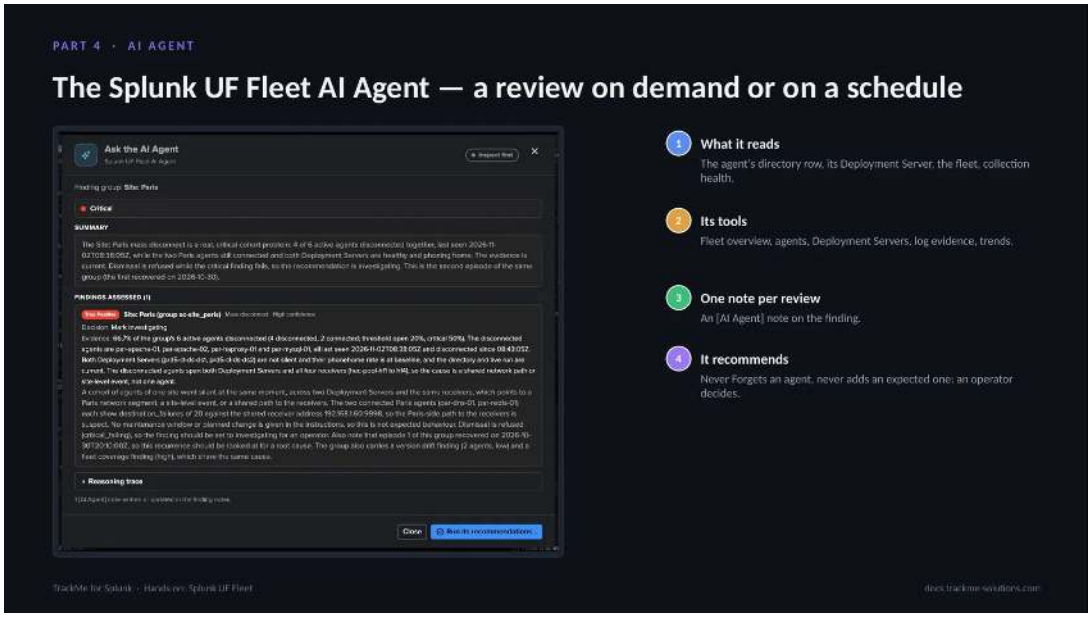
A finding opens with what happened (here 66.7 per cent of the Paris site disconnected), when it opened and its live window, then its triage — status, assignee, severity, snooze — its notes and its history. Open resource, in the row menu, opens the Fleet tab, an agent or its Deployment Server in place. How each policy is evaluated — the rule as configured, the data it reads with ready-to-run searches, and how to investigate and resolve — is in the Policies tab. When the agents reconnect, the mass disconnect recovers below ten per cent for two runs; an agent still disconnected then opens its own finding.

A destination many agents fail to reach



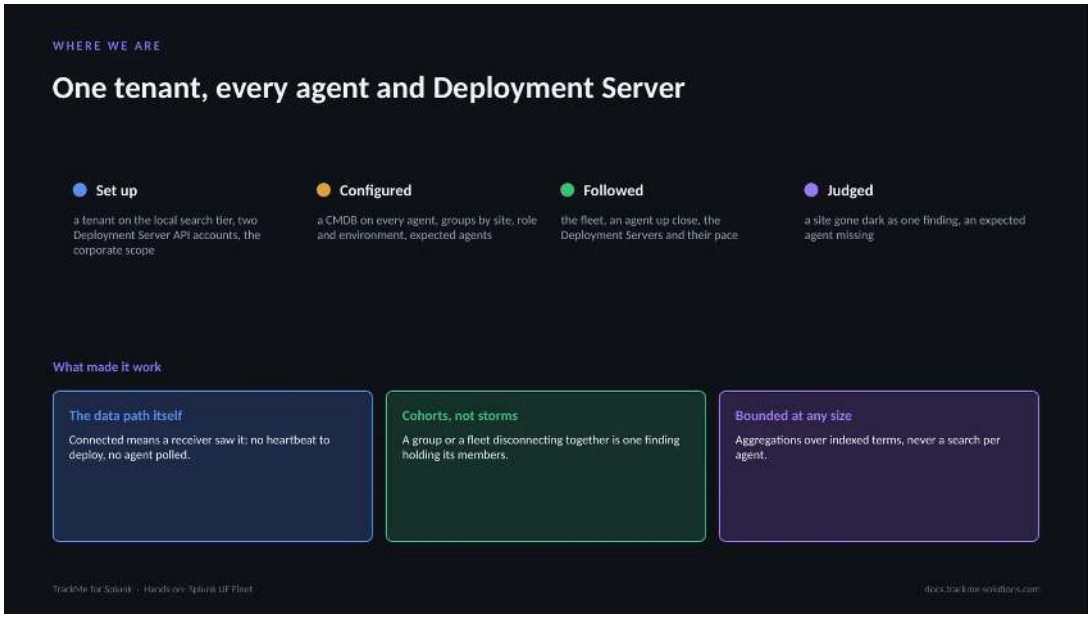
A heavy forwarder of the receiving tier was stopped at the same time. The agents that send to it failed to reach its port: one Destination unreachable finding opened on the receiver, measured on the 16 agents failing for 15 minutes or more, with each agent, how long it failed, its failure lines and the groups the failing agents belong to, never one finding per agent. The receiving side of the same stop is in the Splunk S2S walkthrough.

The Splunk UF Fleet AI Agent



Ask the AI Agent from a finding or a Finding group. The Splunk UF Fleet AI Agent reads the finding, the agent or group, the Deployment Servers and collection health, and may read further through its tools. Each review leaves an [AI Agent] note on the finding. Scheduled triage is off by default: hourly, up to ten findings per run. In act mode it applies only what the tenant allows, and never changes anything on an agent or a Deployment Server.

Where we are



A Splunk UF Fleet tenant set up on twenty universal forwarders and a Deployment Server cluster, enriched from a CMDB, grouped by site, role and environment, its critical agents expected. The fleet, the agents and the Deployment Servers followed. A site gone dark read as a mass disconnect per cohort, an expected agent missing, and the AI Agent's review.

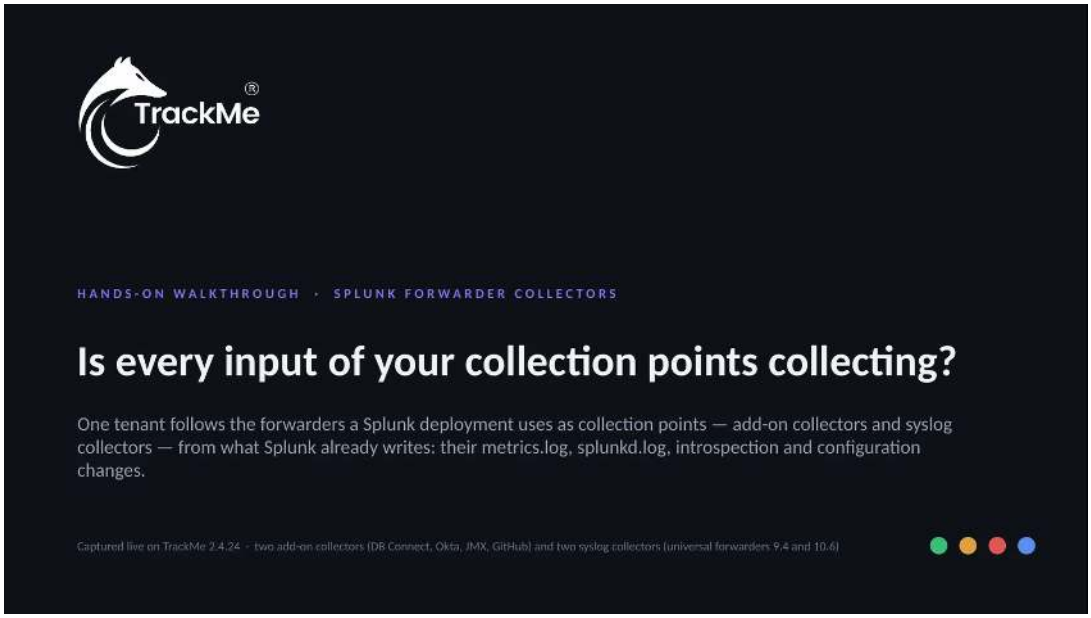
2.1.14 Splunk Forwarder Collectors

Follow the collection points of a Splunk deployment and every input they run, from what the collectors already write: the source and the optional collector API accounts, the scope and the discovery, groups and their minimum, two syslog collectors declared, the inputs that must stay enabled and the notifications; the collectors, their inputs and processes, one collector up close, the inventory and the policies; then a syslog collector stopped on purpose, a group below its minimum, one Finding group per object and the Fleet map.

`Download the deck (PDF) · PowerPoint`

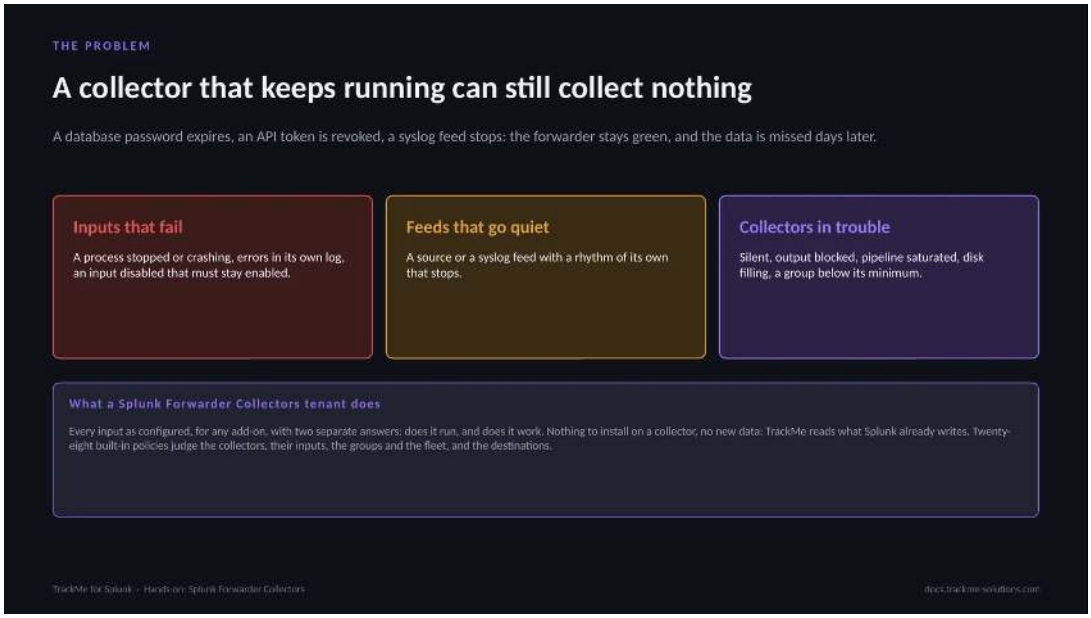
about 30 minutes · 30 slides. Each slide is followed by a description of what it shows — read top to bottom, this page is the walkthrough.

Overview



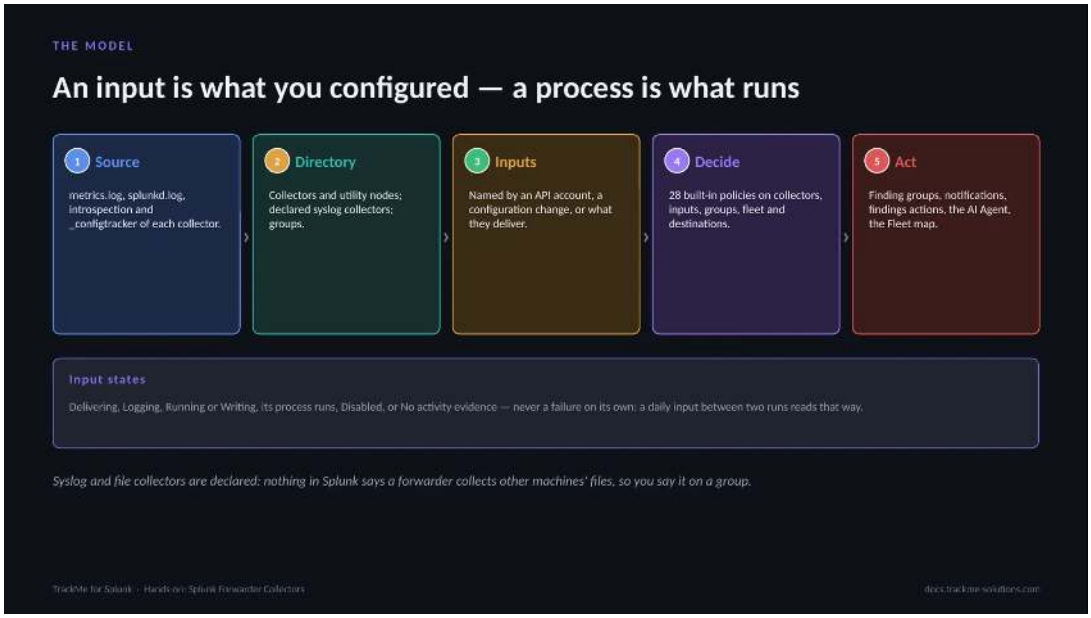
This walkthrough covers the Splunk Forwarder Collectors tenant, generally available in TrackMe 2.4.24. Everything was captured live on a lab of four collection points: two heavy forwarders running the DB Connect, Okta, JMX and GitHub add-ons, and two universal forwarders on Splunk 9.4 and 10.6 that monitor the files an rsyslog daemon writes for forty devices. In four parts: create the tenant, configure it, follow the collectors and their inputs, and read the findings of an outage caused on purpose.

Why a dedicated tenant for collection points



A collection point runs what the rest of the platform depends on: DB Connect, cloud and API add-ons, scripted and modular inputs, or the files of a syslog daemon. A database password expires, an API token is revoked, a syslog feed stops, and the forwarder stays green while the data is missed. A failing add-on may keep logging as usual, so running proves nothing about working. Splunk S2S judges how data moves, Splunk UF Fleet the agents and Splunk HEC its endpoints: none of them says whether an input is collecting. This tenant does.

How the tenant reads a collector



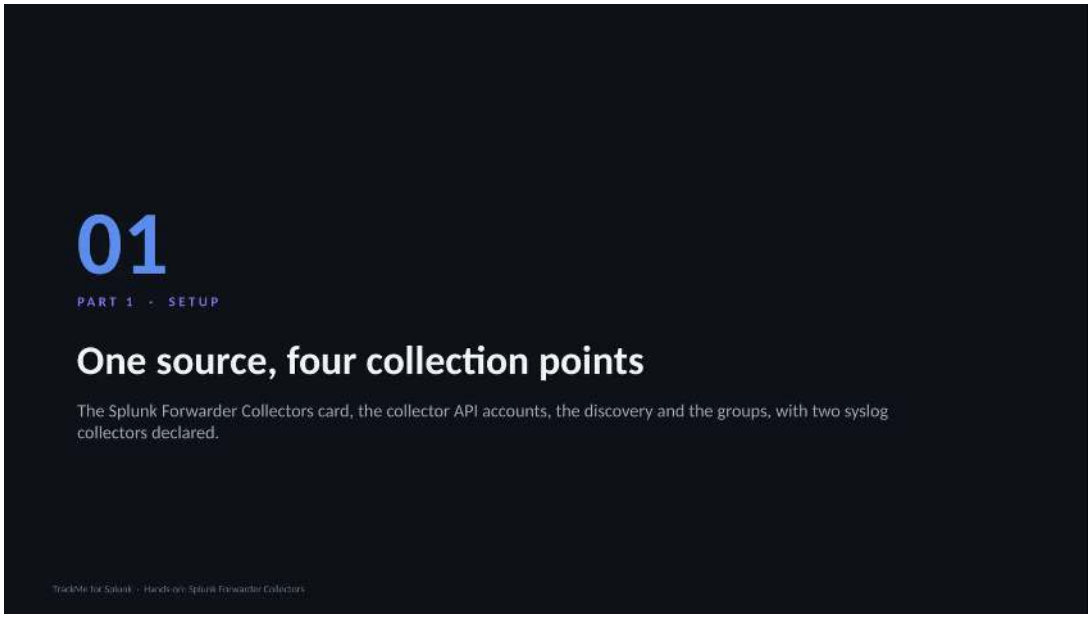
Each collector writes its metrics.log and splunkd.log to _internal, its processes and disks to _introspection, and its configuration changes to _configtracker. TrackMe builds a directory of collectors, lists each input an operator configured, and reads every five minutes what each one delivered, logged and ran. One script runs every stanza of its modular input, so a process names no input: inputs and processes stay apart. Nothing is installed on a collector and no new data is ingested.

The plan



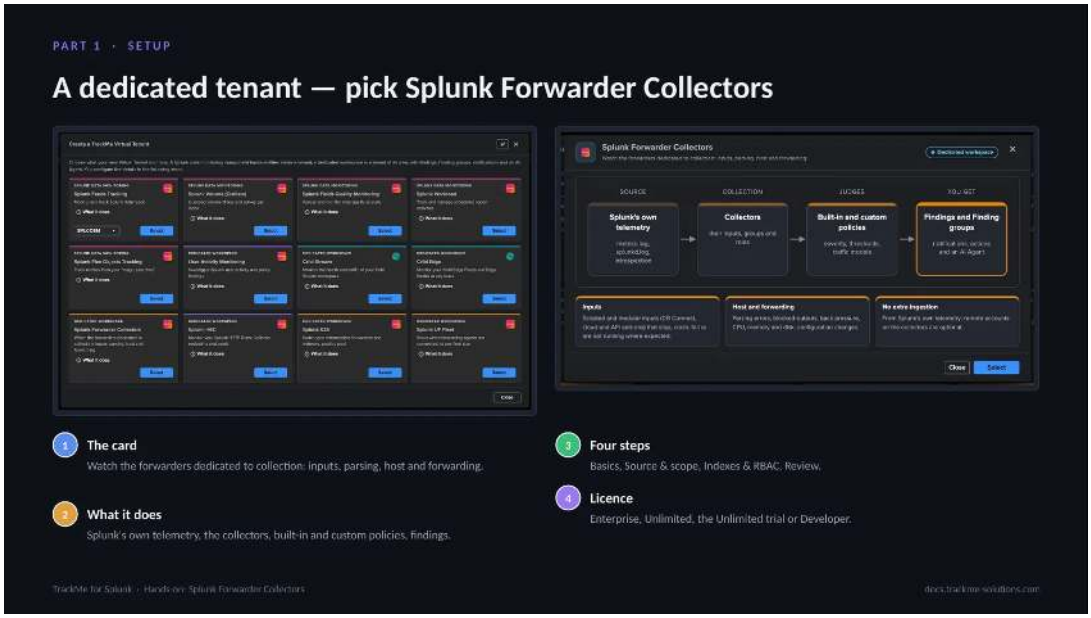
Four parts. Setup: the wizard, the optional collector API accounts, the discovery and the groups, with two syslog collectors declared. Configure: the groups and their minimum, the inputs that must stay enabled and the notifications. Follow: the collectors, their inputs and processes, one collector up close, the inventory and the policies. Findings: the forwarder of one syslog collector stopped on purpose, the findings that open and how they group, then the Fleet map of the fleet and its groups.

Part one



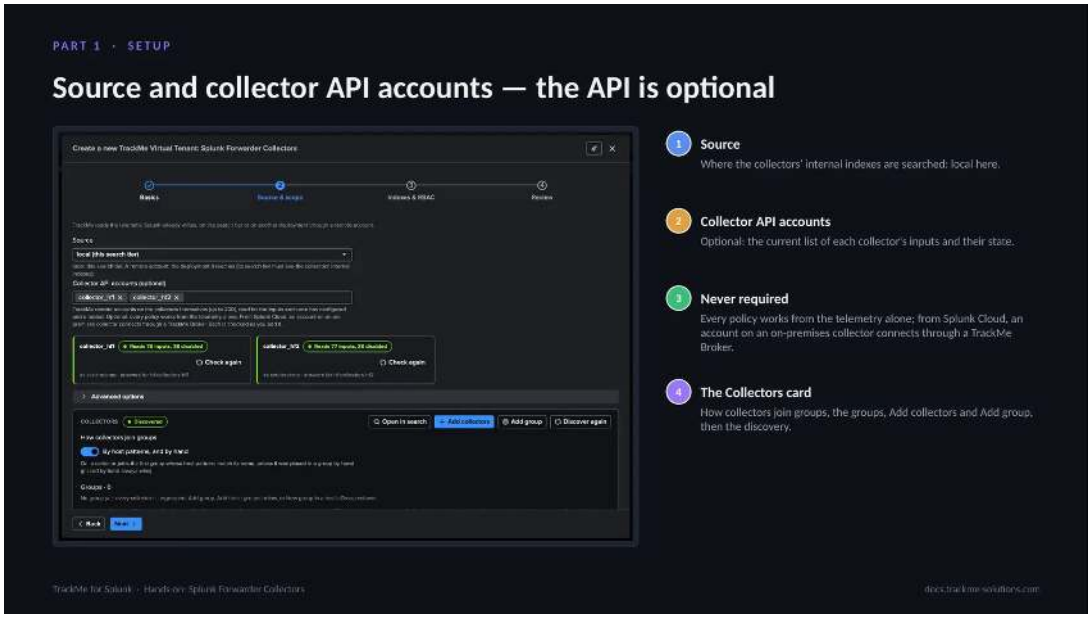
Part one covers the setup. A Splunk Forwarder Collectors tenant reads one deployment: its local search tier, or another Splunk deployment through a TrackMe remote account. The wizard walks through the source, the optional collector API accounts, the scope and the discovery, the groups it proposes from the collectors' host names, the collectors declared as syslog or file collectors, then the indexes, the RBAC and the tenant owner, checked before the tenant is created.

Creating the tenant



The tenant is created from Virtual Tenants, Actions, Create a tracking tenant, then the Splunk Forwarder Collectors card. Its What it does dialog draws the flow: Splunk's own telemetry, the collectors with their inputs, groups and roles, the built-in and custom policies, and the findings with notifications, findings actions and an AI Agent. Three cards sum up what is judged: the inputs, the host and the forwarding. No extra ingestion is needed: everything comes from what the collectors already write.

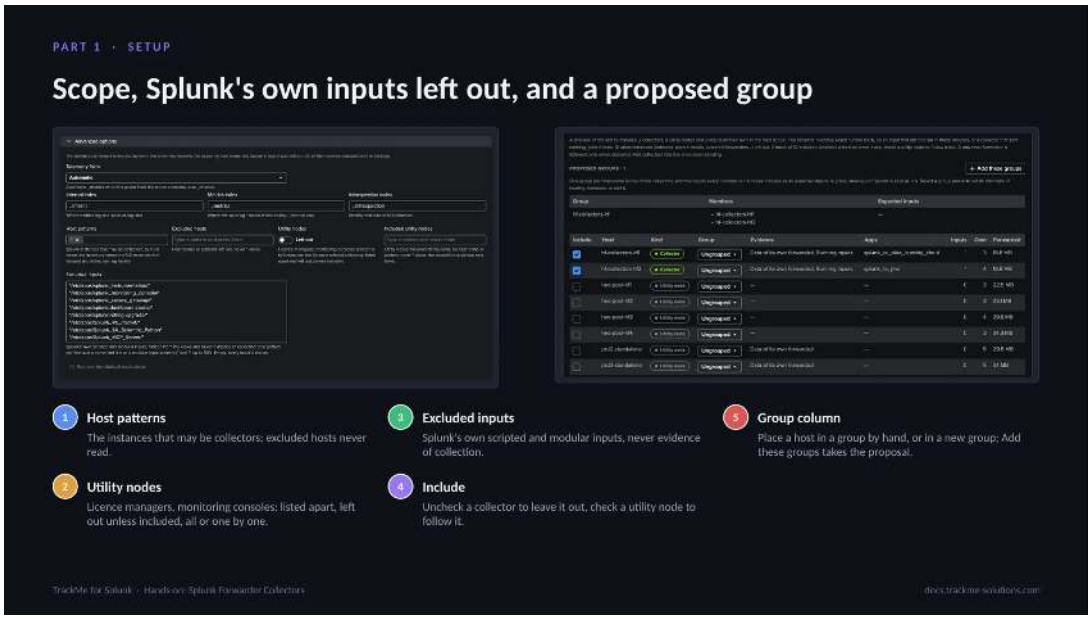
Source and API accounts



The source is the search tier that sees the collectors' _internal, _introspection and _configtracker indexes. A TrackMe remote account on each collector adds the current list of its inputs, enabled or not, read from the collector's own REST API. These accounts are optional: every policy works from the telemetry alone, and from Splunk Cloud an account on an on-premises collector connects through a TrackMe Broker. The wizard checks each account as it is added.

Below, the Collectors card holds how collectors join groups, the groups of the draft, Add collectors and Add group, then the discovery.

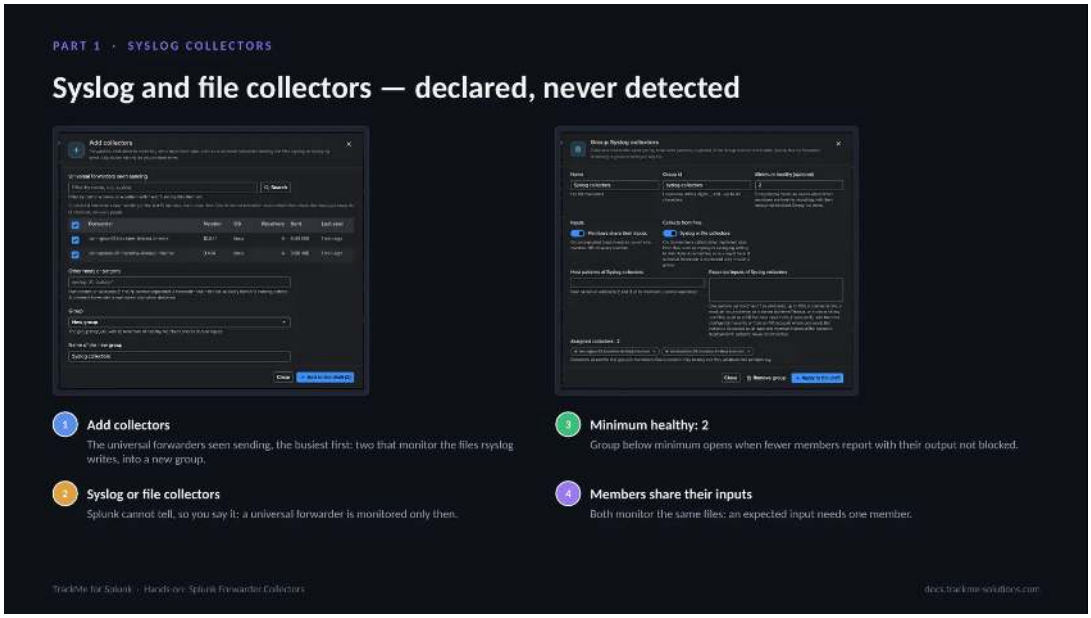
Scope and discovery



Advanced options hold the telemetry form, the three index names, the host patterns and the excluded hosts, the switch for utility nodes such as license managers and monitoring consoles, the utility nodes followed one by one, and the inputs to leave out, Splunk's own inputs by default. The defaults suit almost every deployment.

The discovery runs two short searches over the last 15 minutes and proposes one group per host-name family of the collectors it found. Its Include boxes say which hosts the tenant follows: unchecking a collector adds it to the excluded hosts, checking a utility node follows it. The Group column places a host in a group by hand.

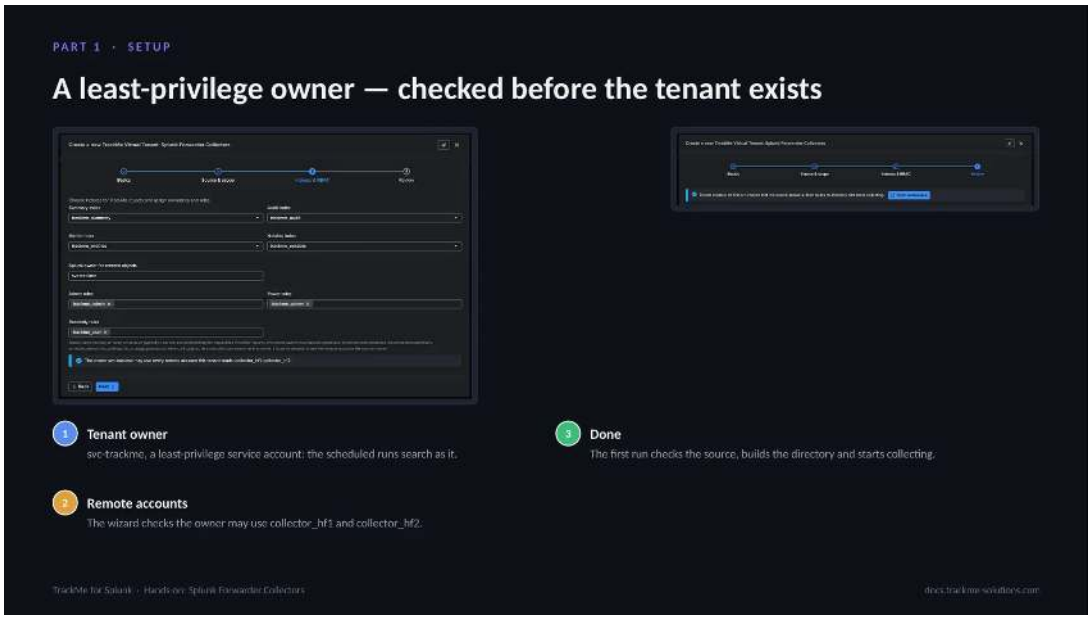
Declaring syslog collectors



Nothing in Splunk says a forwarder collects other machines' data from files, so the operator declares it with Add collectors, in the wizard, in Settings and in the Inventory. The dialog lists the universal forwarders the receivers saw sending in the last 15 minutes, the busiest first; the operator checks the ones that collect files, or types host names or patterns, and picks a group.

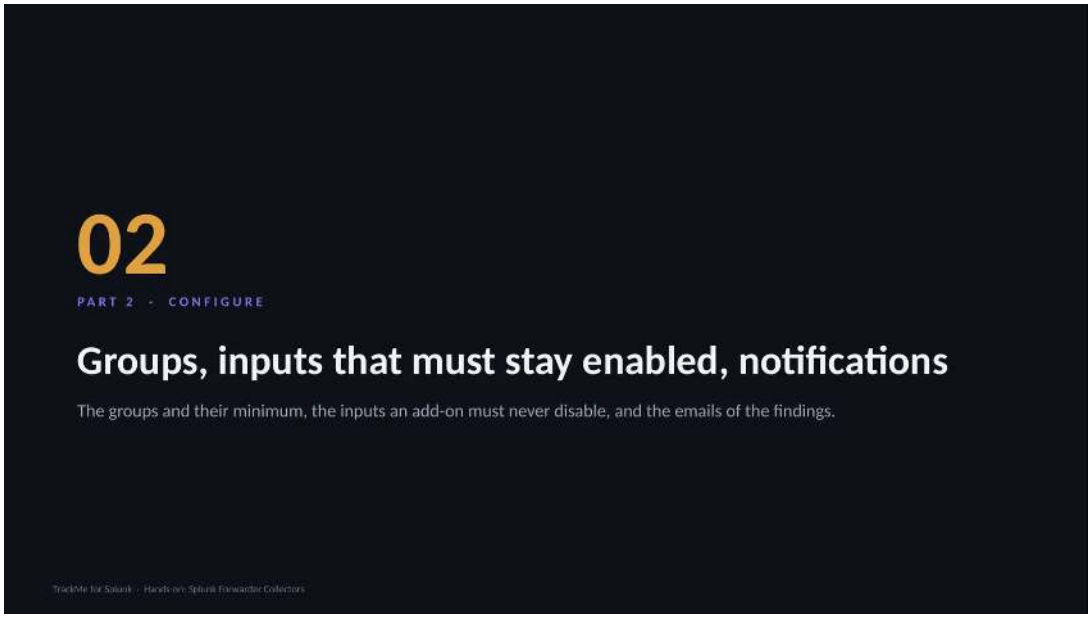
Here the two universal forwarders that monitor the rsyslog files join a new group, Syslog collectors, set to a minimum of two healthy members with inputs shared by both. Each monitor stanza of a declared collector then becomes an input, judged by what reaches the indexes.

Owner, review and creation



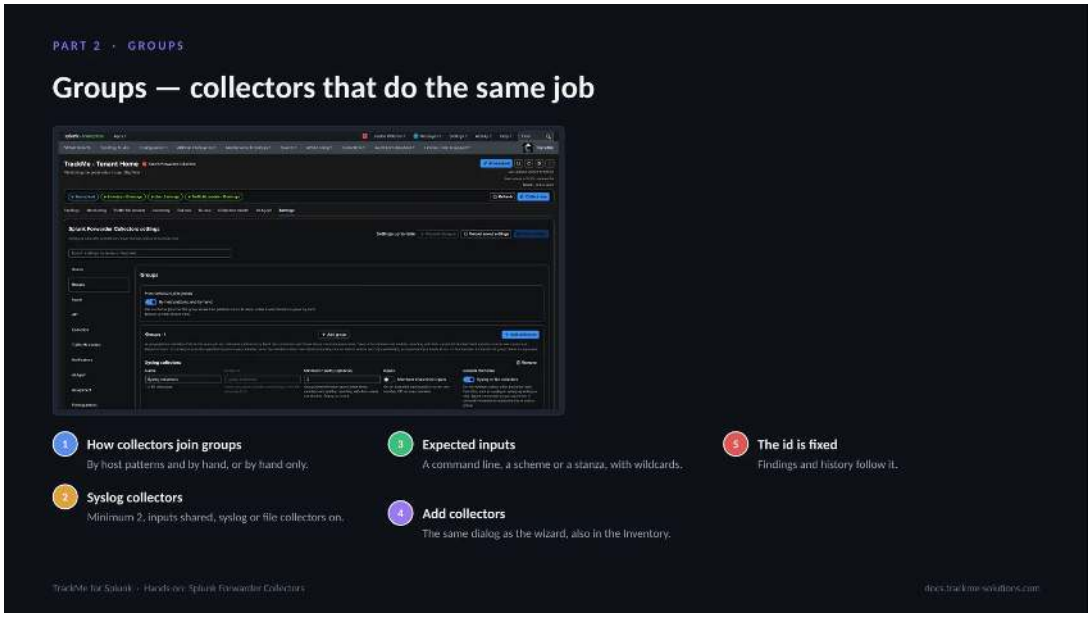
Indexes and RBAC come next, then Review and Done. The tenant owner runs the scheduled jobs and must be allowed to use every remote account the tenant reads; otherwise the wizard names the fix before the save, so a least-privilege owner works from the first run. The tenant fills progressively: the directory of collectors first, then their input registrations, then the configuration changes, replayed back up to 90 days.

Part two



Part two covers the configuration of the tenant, in its Settings and Inventory tabs: the groups and the minimum of healthy members each one requires, the inputs an add-on or an operator must never disable, and the notifications, sent by email for each transition of a Finding group or as one digest per live run. Every setting saved here applies to the next live run, without restarting a job or editing a saved search.

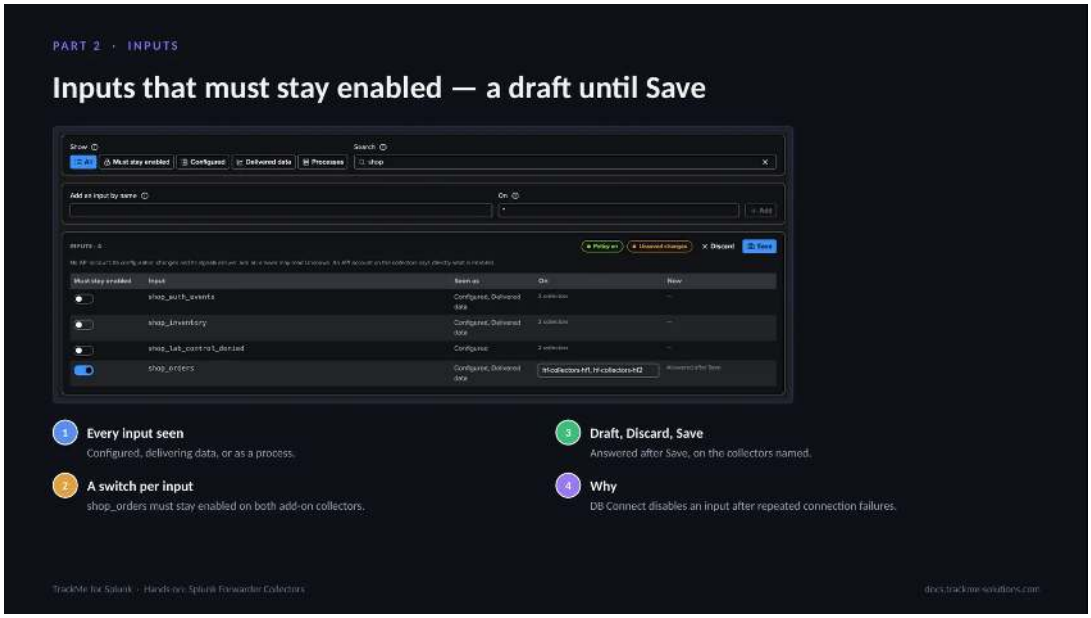
The groups



Settings, Groups. How collectors join groups comes first: by host patterns and by hand, or by hand only. Splunk has no clustering of forwarders, so a group is what the operator says it is: a name, host patterns, an optional minimum of healthy members, and the inputs every member is expected to run. Each member is judged alone and runs its own inputs, unless the members share them: a standby pair, or an add-on with its own high availability.

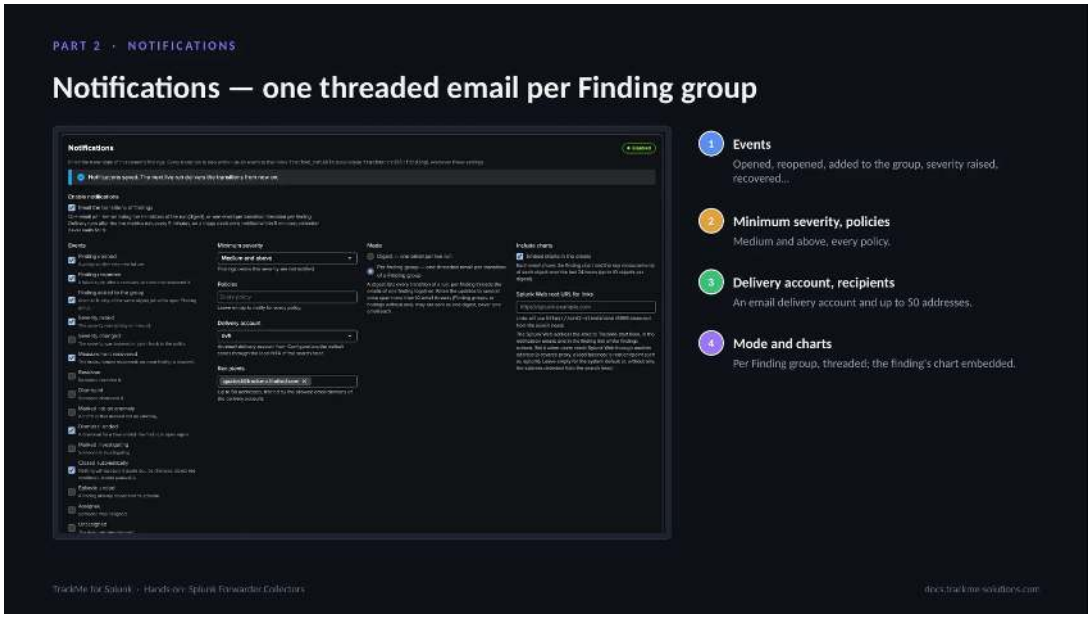
Add collectors opens the same dialog as the wizard. A group appears in Monitoring and on the Fleet map while it has a monitored member.

Inputs that must stay enabled



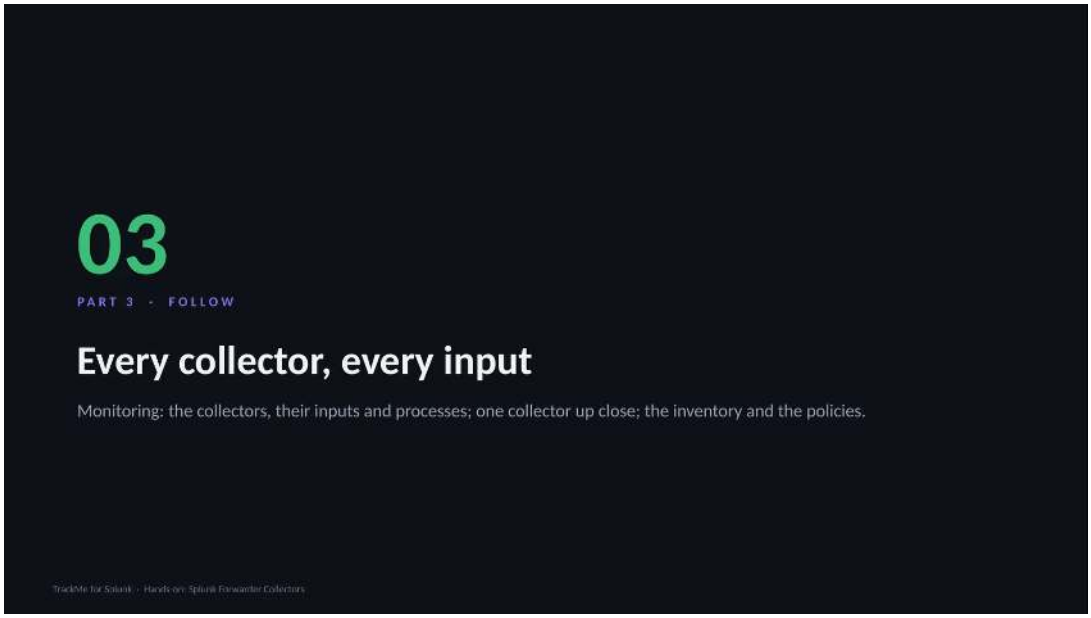
Inventory, Inputs. An add-on may disable its own input, as DB Connect does after repeated connection failures, and an operator may disable one by mistake. Here the input shop_orders is switched to must stay enabled on both add-on collectors, as a draft until Save. Whether it is enabled is read from an API account first, then from the configuration changes, then from what the input shows. Its finding is always at least high, and the AI Agent can never dismiss it while it fails.

Notifications



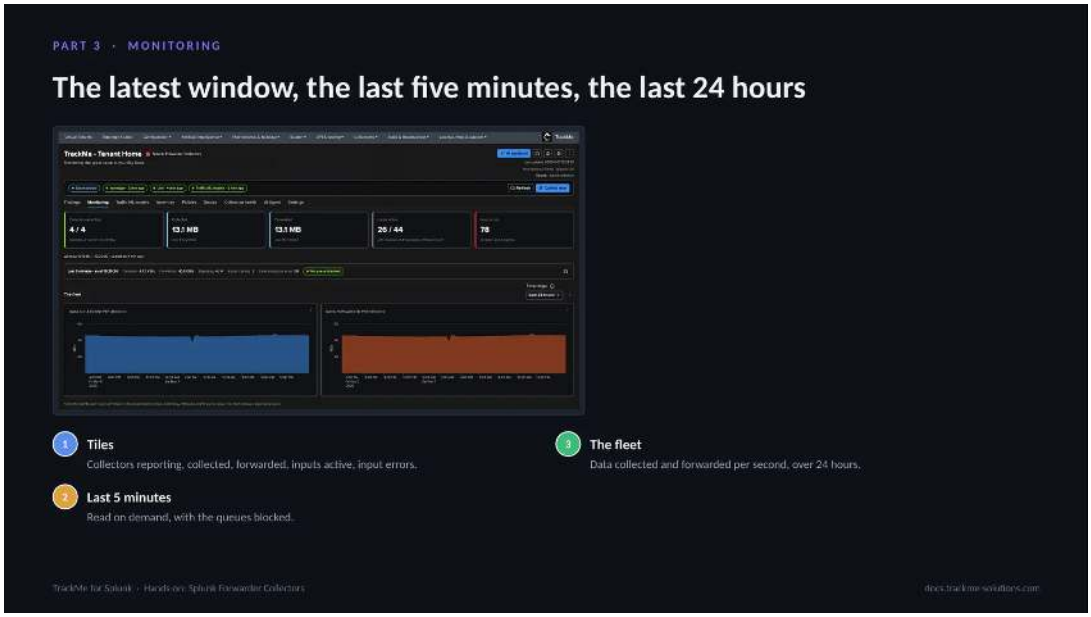
Settings, Notifications. TrackMe sends either one digest per live run, or one threaded email per transition of a Finding group, with the finding's chart and key measurements: opened, reopened, a finding added to the group, a severity raised, recovered. A minimum severity and a list of policies narrow what is sent, through a TrackMe email delivery account. Every transition is also written as an event to the tenant's notable index, whatever the notification settings.

Part three



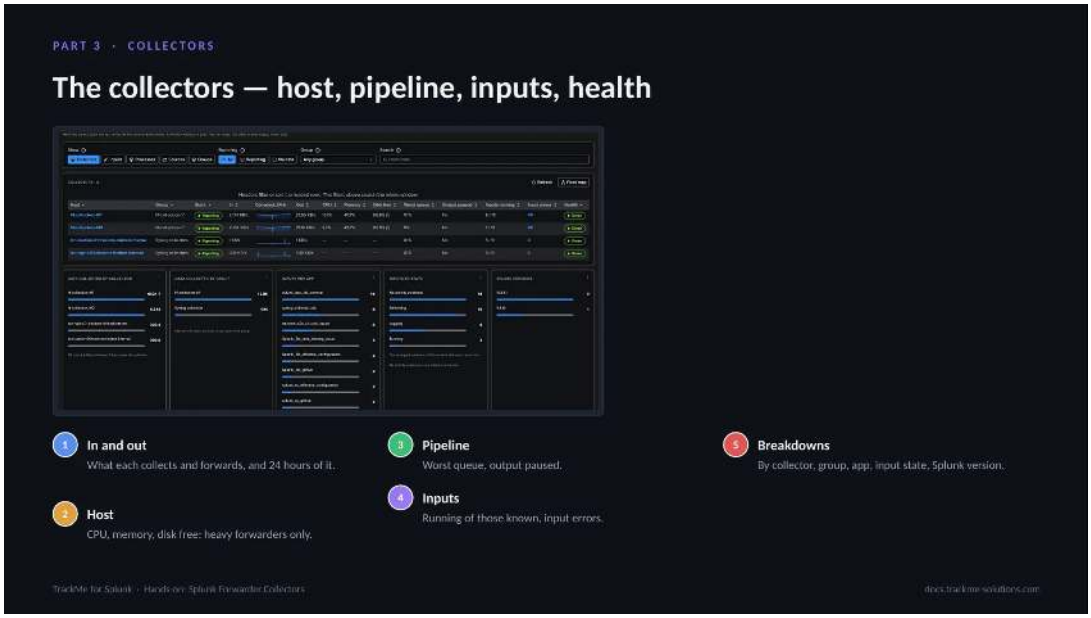
Part three follows the collectors in the tenant's workspace, its Tenant Home: Findings, Monitoring, Traffic ML models, Inventory, Policies, Source, Collection health, AI Agent and Settings. A status strip above the tabs shows the source and when the inventory, the live run and the traffic models last ran, each pill opening Collection health, with Refresh and Collect now to run the live job on demand.

Monitoring



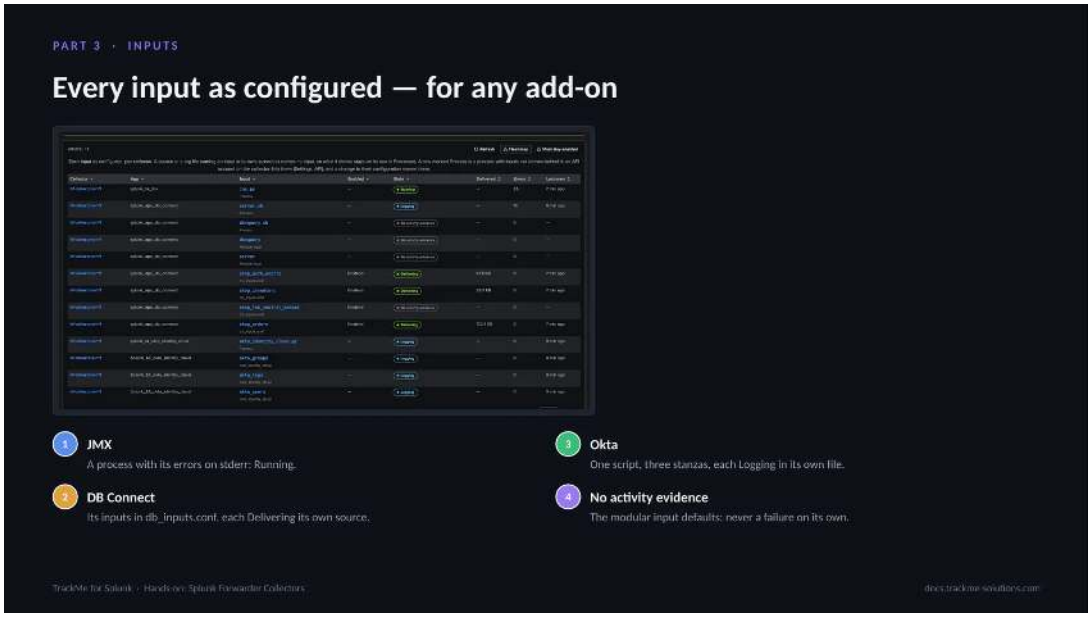
Monitoring is the main view. Its tiles give the latest collection window: the collectors reporting, what they collected and forwarded, the inputs active and the input errors. The last five minutes are read on demand, and the fleet's history comes from the points each live run writes to the tenant metric index, shown per second over 24 hours. Below, the table switches between collectors, inputs, processes, sources and groups.

The collectors



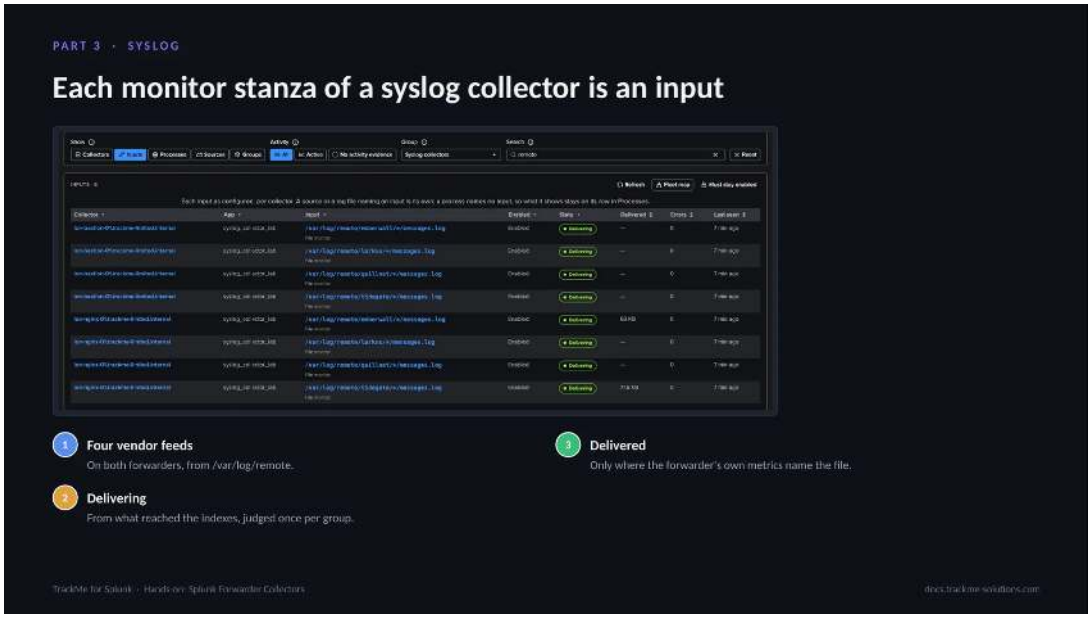
The Collectors view lists each collector with what it took in and forwarded, its volume over 24 hours, its CPU, memory and free disk, its worst queue, whether its output paused, its inputs running and its input errors, and Splunk's own health report. A universal forwarder sends no resource usage by default, so it shows no CPU, memory or disk. Breakdown cards under the table count the whole window whatever the filters.

The inputs of a heavy forwarder



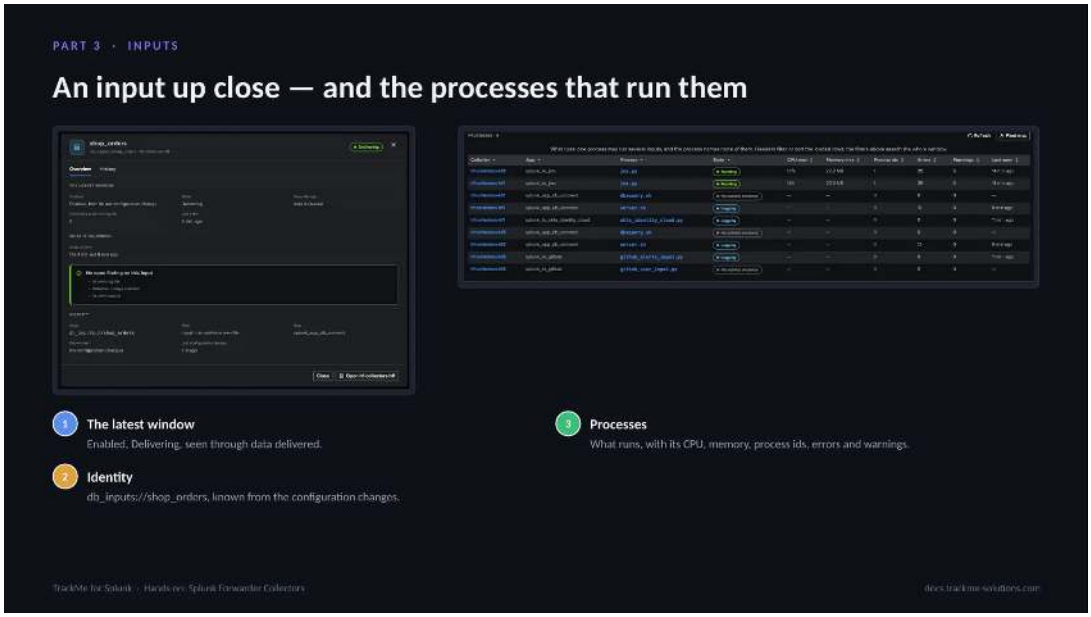
The Inputs view of one heavy forwarder shows the three ways an input is known: an API account, a configuration change, or what it delivers. The JMX add-on runs as a process with errors on stderr, DB Connect's inputs deliver from db_inputs.conf, and the Okta script logs for its three stanzas. A modular input default with No activity evidence is not a failure. A row marked Process is a process with no input known behind it.

The inputs of a syslog collector



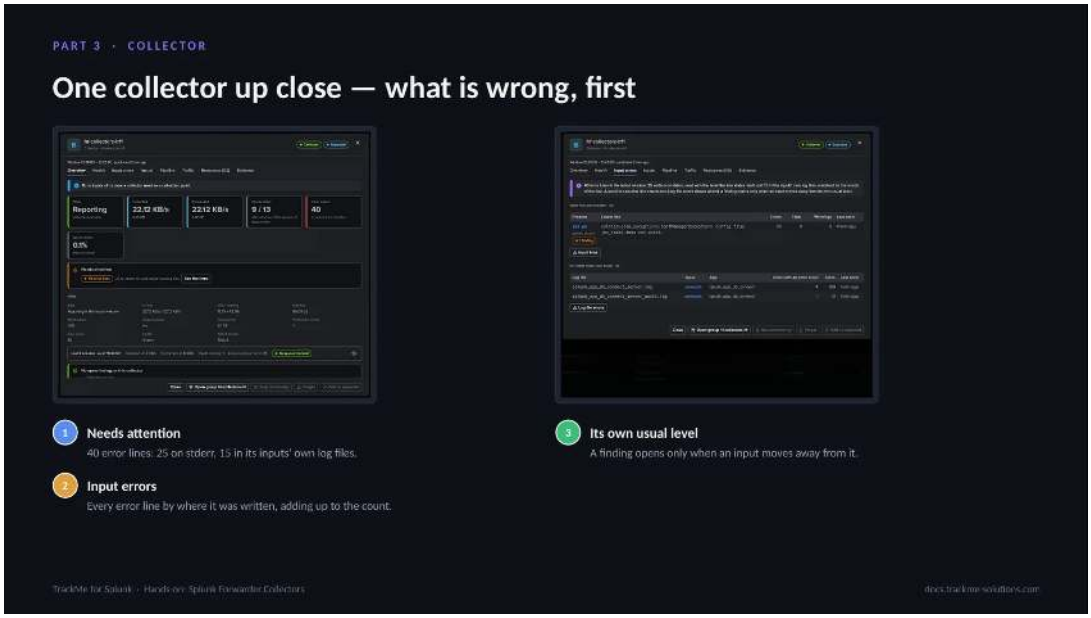
On a declared syslog collector, each monitor stanza is an input. Here the four vendor feeds of the rsyslog daemon are inputs of both universal forwarders. A stanza that names the host of its events is a feed: what reached the indexes says whether it delivers, judged once for the group, since a source does not say which member delivered it. Input went quiet opens when a feed with a learned rhythm stops.

Inputs and processes up close



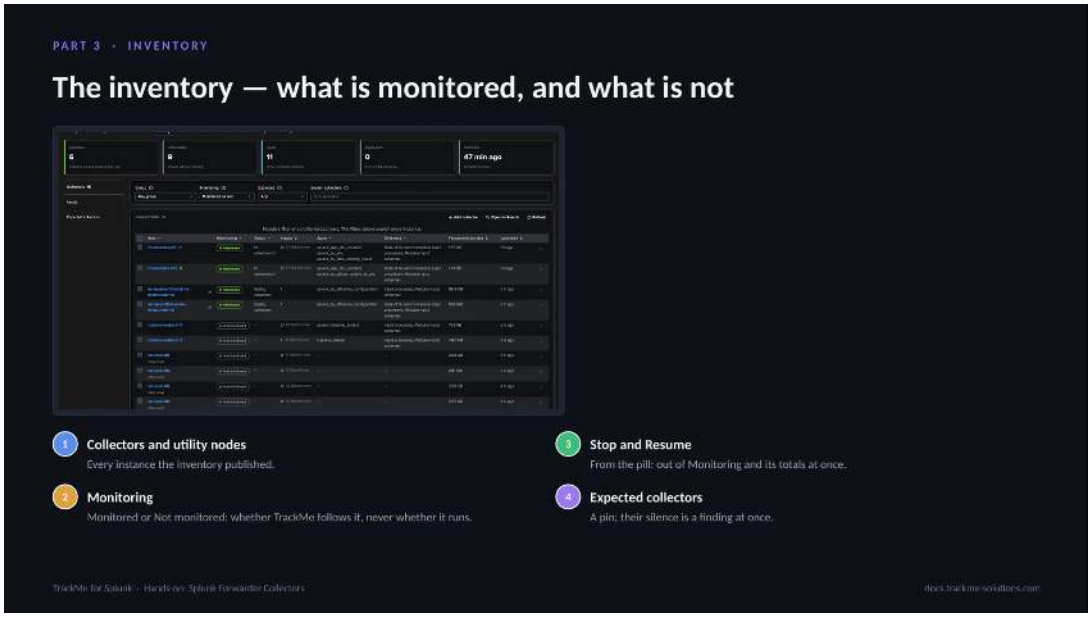
An input opens its detail: its state and how it was seen, what it delivered, its error lines, whether it must stay enabled, and its identity, here the DB Connect input shop_orders known from the configuration changes. Processes lists what runs, with its CPU, memory, process ids and the error and warning lines of its stderr. One process may run several inputs, and the process names none of them.

A collector up close



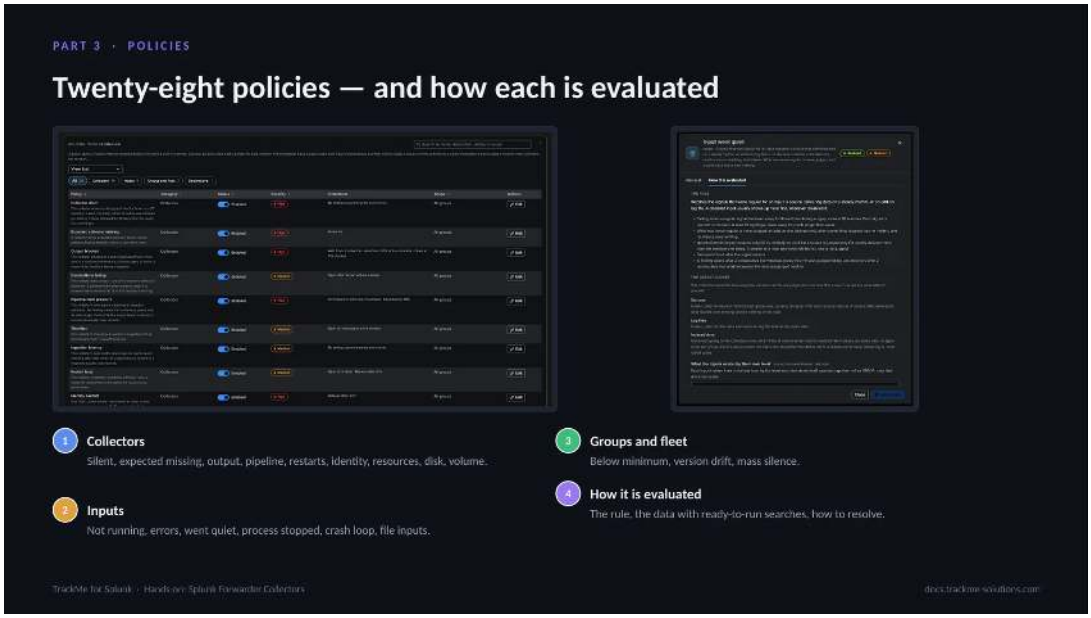
A collector opens in full and says what is wrong before anything else, with a button to the tab that explains it. Input errors counts both places an input writes errors: stderr, by the level each line states itself, since splunkd relays all of stderr as errors, and the input's own log file, matched on the words of the line. The Input errors policy opens only when an input moves away from its own usual level.

The inventory



The Inventory lists the collectors and the utility nodes with their group, inputs, apps and evidence, and whether each is monitored. Stop monitoring takes an instance out of the live reads, the policies, the Monitoring tab and the Fleet map at once, and resolves its open findings; Resume brings it back. An expected collector of a group cannot be stopped until it leaves the group's selection, so a stop never hides a missing collector.

The policies



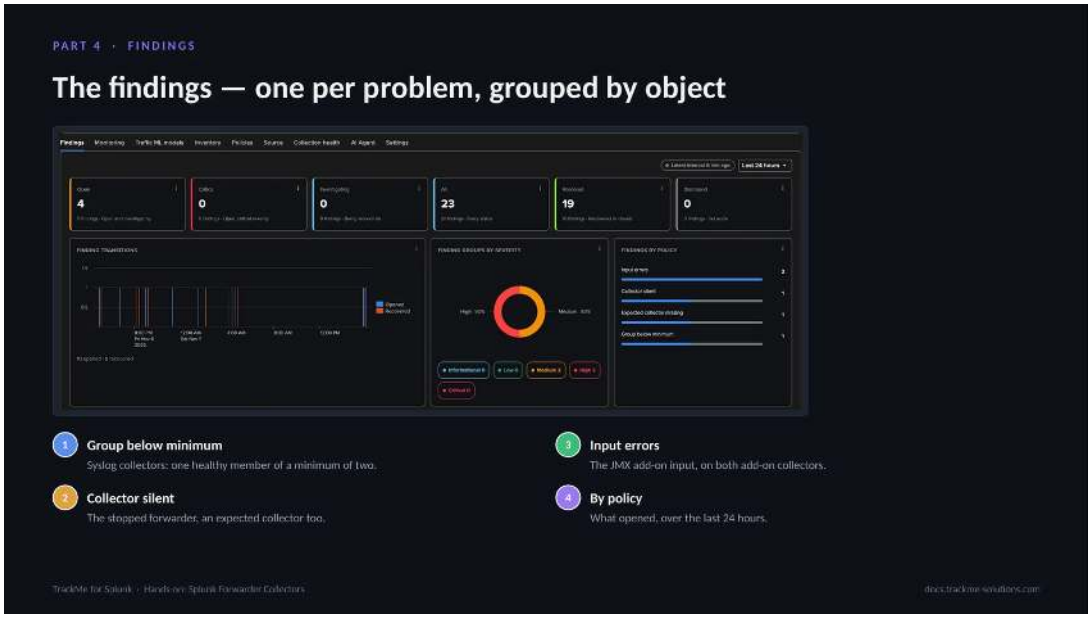
Twenty-eight built-in policies cover the collectors, their inputs, the groups and the fleet, and the destinations, each with its severity, its windows to open and recover, its scope and a bank holiday hold. How it is evaluated shows the rule as configured, the data it judges with ready-to-run searches, and how to investigate and resolve. Input went quiet, for example, judges a regular signal away for three times its largest gap and at least thirty minutes.

Part four



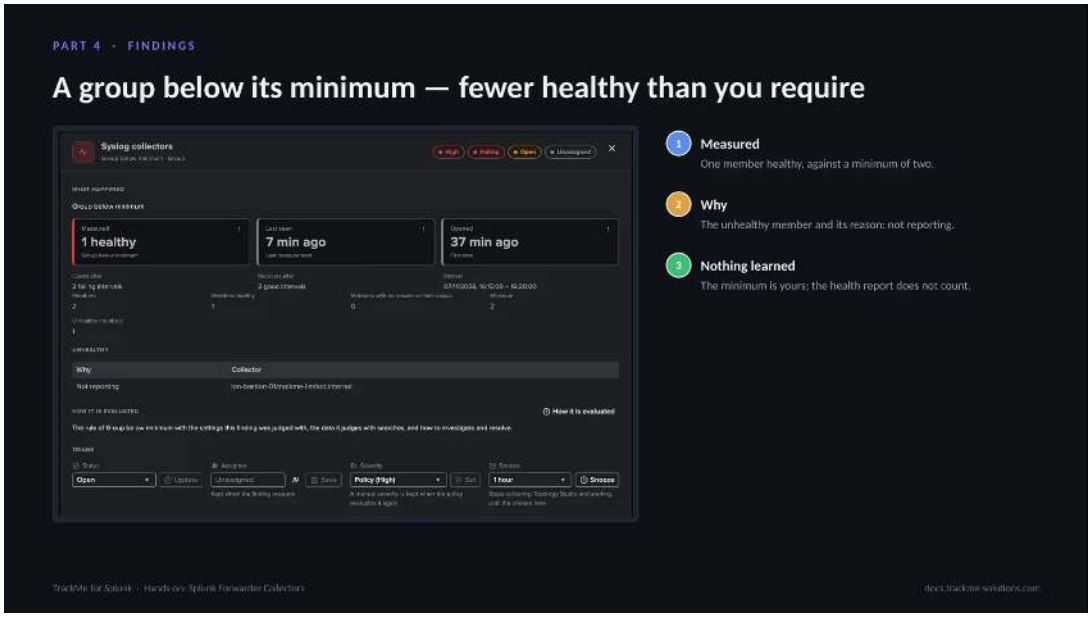
Part four reads the findings. The forwarder of one syslog collector was stopped on purpose, in a group that requires both of its members to be healthy. The findings that open follow the problem, not the number of signals: the group falls below its minimum, the stopped collector is silent and an expected collector is missing, and these two findings about one collector form one Finding group, triaged once.

The Findings tab



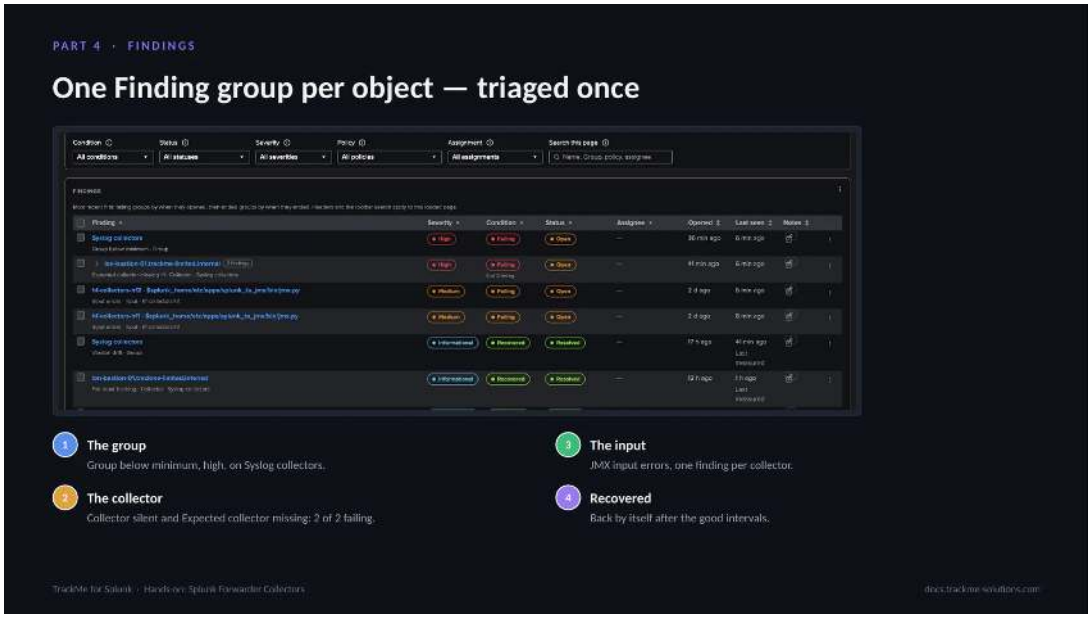
The findings centre is shared by every dedicated tenant. Its tiles count the open, critical, investigating, resolved and dismissed findings of the period, with the transitions over time, the Finding groups by severity and the findings by policy. Here, over 24 hours: four open Finding groups, none critical, nineteen resolved, and the policies Input errors, Collector silent, Expected collector missing and Group below minimum.

Group below minimum



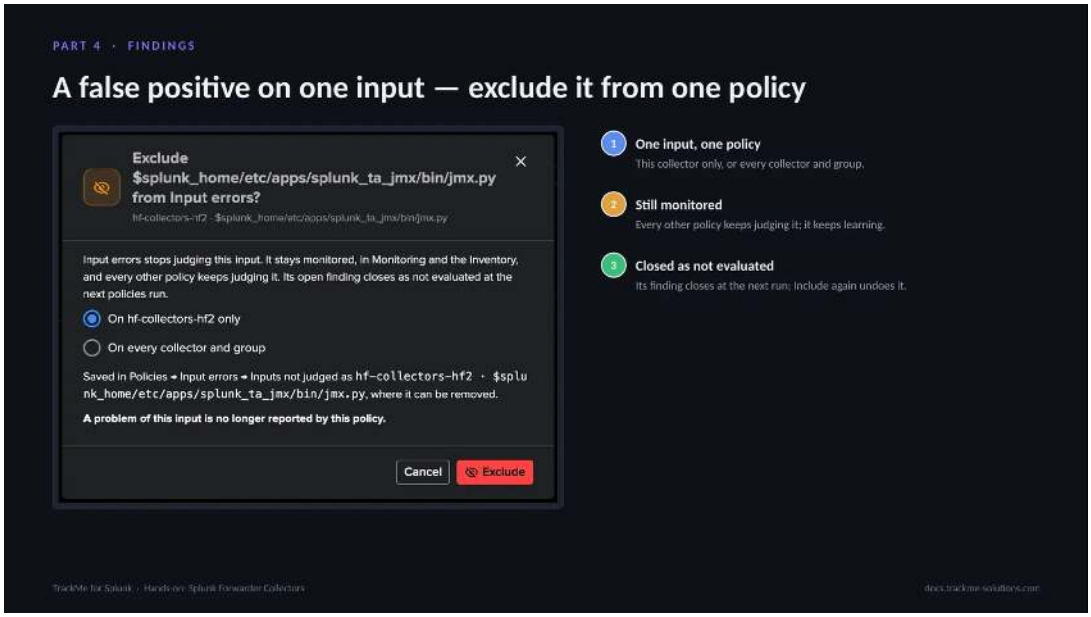
The group Syslog collectors requires two healthy members. With one forwarder stopped, Group below minimum opened, high, critical only when no member is healthy. The finding shows what it judged: two members, one healthy, the minimum, and each unhealthy member with its reason, here Not reporting. A member whose output was blocked just before it went quiet counts as blocked, not silent. The finding recovers by itself after two good intervals.

One Finding group per object



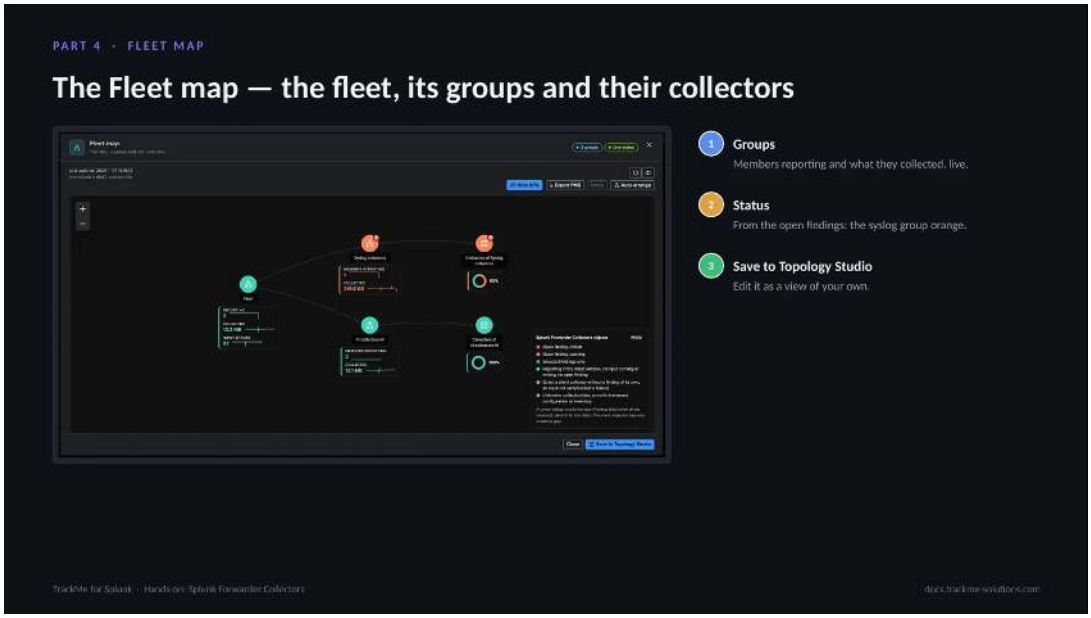
The list shows one row per Finding group. The two findings about the stopped collector, Collector silent and Expected collector missing, form one Finding group with two of two failing, triaged once. The JMX input errors open one finding per collector. A syslog feed that stops while its collectors keep running opens Input went quiet, once for the group, after a day of steady deliveries has taught its rhythm.

Excluding one input from one policy



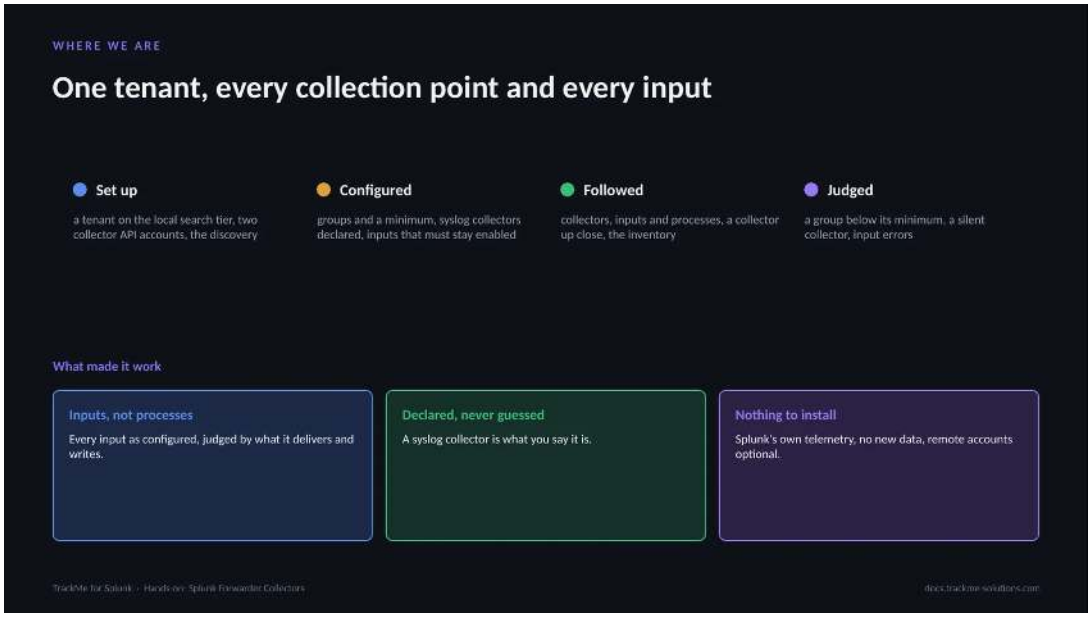
When one input keeps raising false positives in one input policy, the operator takes it out of that policy only, from its finding's menu: Exclude this input from this policy, on this collector only or on every collector and group. The input stays monitored and every other policy keeps judging it; it also keeps learning its rhythm and its usual errors, so it is judged at once if included again. Its open finding closes as not evaluated at the next policies run, never as a recovery. The entry is saved in the policy's Inputs not judged, How it is evaluated lists it, and Include again undoes it.

The Fleet map



The Fleet map, a button of the Monitoring tab, is a generated topology of the fleet, its groups and the collectors of each group, with live status from the open findings and their KPIs over time: here the syslog group in orange, one member of two reporting. Save to Topology Studio makes it a view of one's own. A group with no monitored member left is not drawn.

Where we are



A Splunk Forwarder Collectors tenant set up on four collection points, with two syslog collectors declared, groups with a minimum, and an input that must stay enabled. Every collector, input and process is followed from what Splunk already writes, with nothing installed and no new data. A stopped collector reads as a group below its minimum and one Finding group for the collector, beside the input errors of an add-on.

2.1.15 Reference white papers

The former quickstart tutorials remain available as reference material.

QUICK START - Starting with TrackMe: (feed tracking quickstart)

Starting with TrackMe!

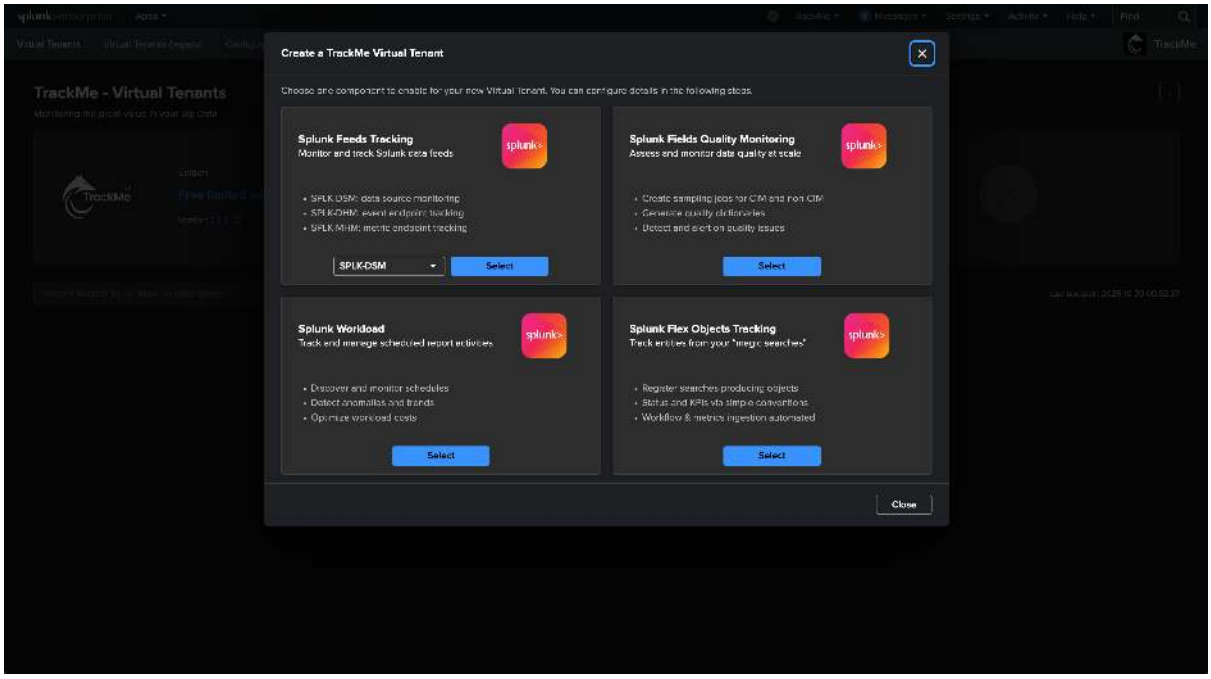
- This tutorial is a starting point for TrackMe new joiners!
- The objective is to help you get started with TrackMe, understand the basic concepts and focus on feeds Tracking for Splunk.
- In this tutorial, we will assume that you are starting entirely from scratch, and we will cover the essential steps to set up feed tracking in TrackMe, with main best practices.
- **This tutorial works in 3 main steps and can be used by all users, including Foundation Edition trial users:**
  - **Step 1:** Create a Virtual Tenant for data sources tracking
  - **Step 2:** Create a Virtual Tenant for data hosts tracking
  - **Step 3:** Alerting and notifications
- This quickstart tutorial was last updated on the 25th of May 2025, to reflect the latest TrackMe releases and especially the new Stateful Alerting features.

TrackMe is installed, what now?

Once TrackMe is installed, the following wizard will guide you through the initial setup of a new Virtual Tenant:

About Virtual Tenants:

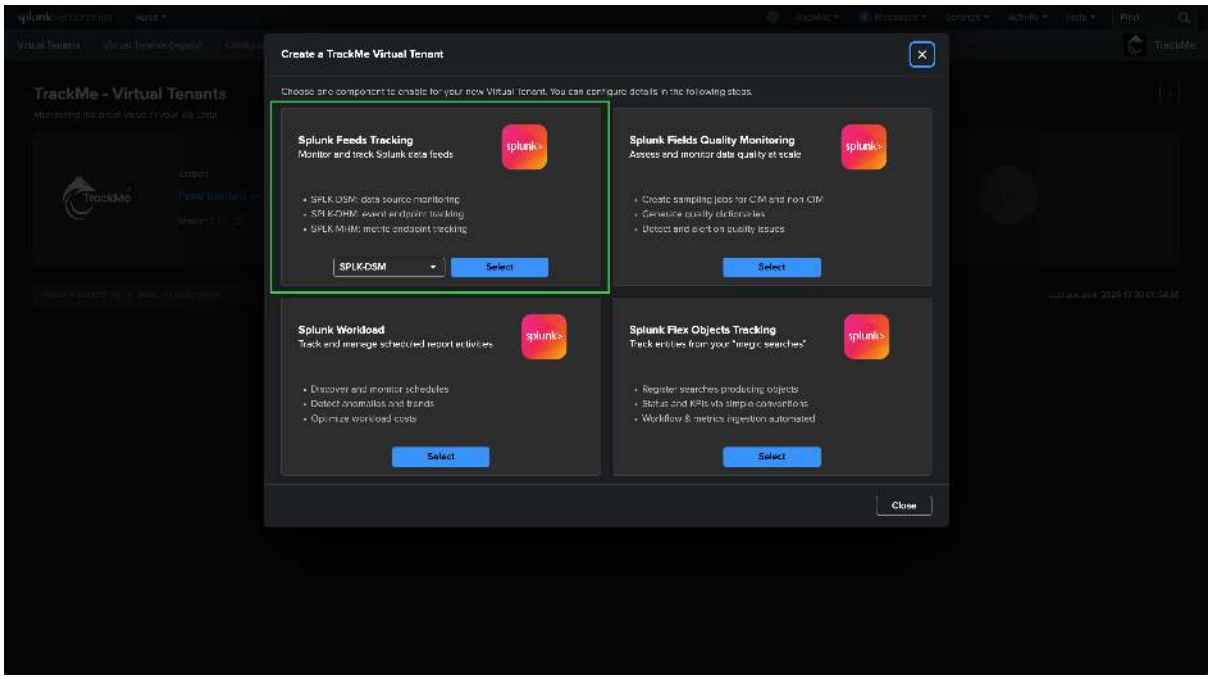
- Virtual Tenants are an essential core feature in TrackMe, this allows TrackMe to orchestrate and manage TrackMe-related knowledge objects and configuration, and address many powerful concepts such as multi-tenancy, data isolation, and more.
- You can also understand this concept as virtual instances of TrackMe within TrackMe, which you can create, destroy, and restart as needed.
- Virtual Tenants enable TrackMe components, which are designed to address use cases, such as **splk-dsm** which is the essential component for feed tracking.



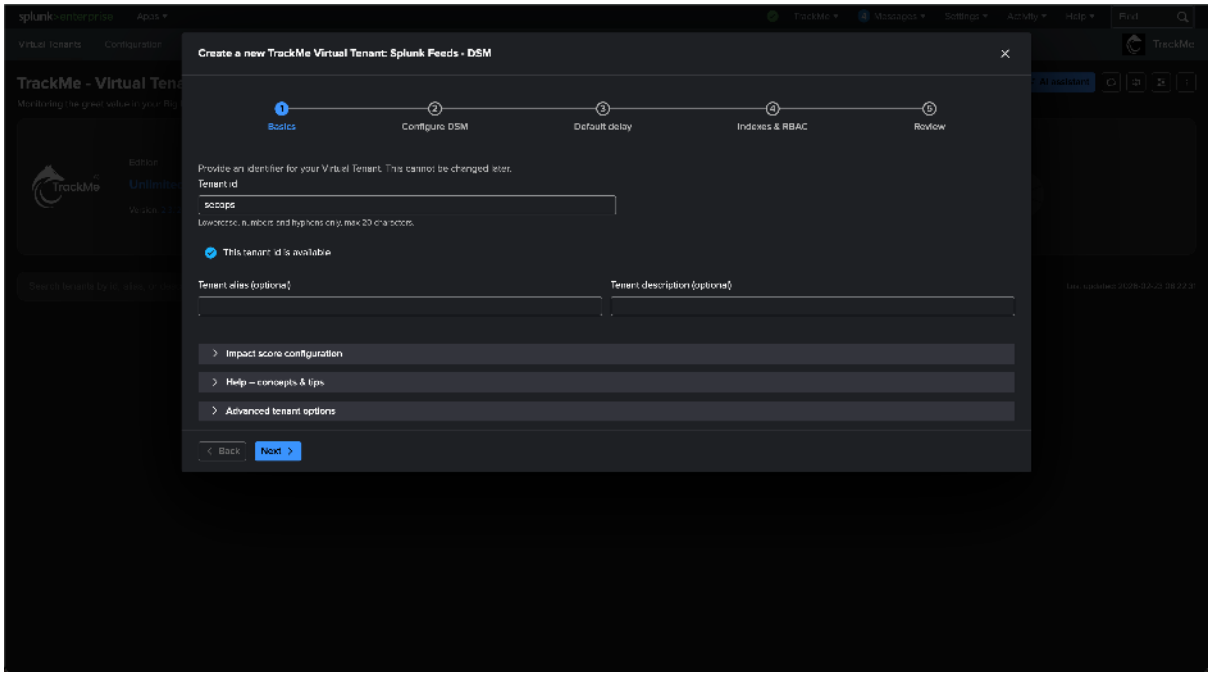
Step 1: Let's create a Virtual Tenant for data sources tracking

We will now create a Virtual Tenant to start tracking data sources:

- component: `Splunk Feeds Tracking`

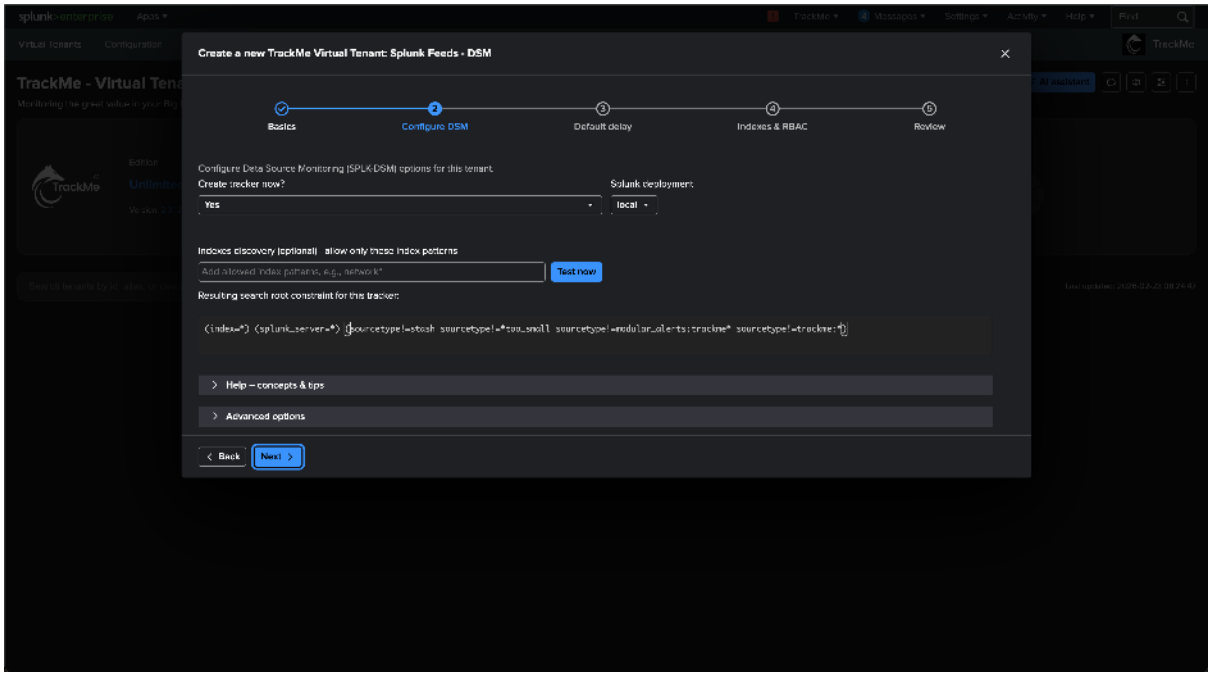


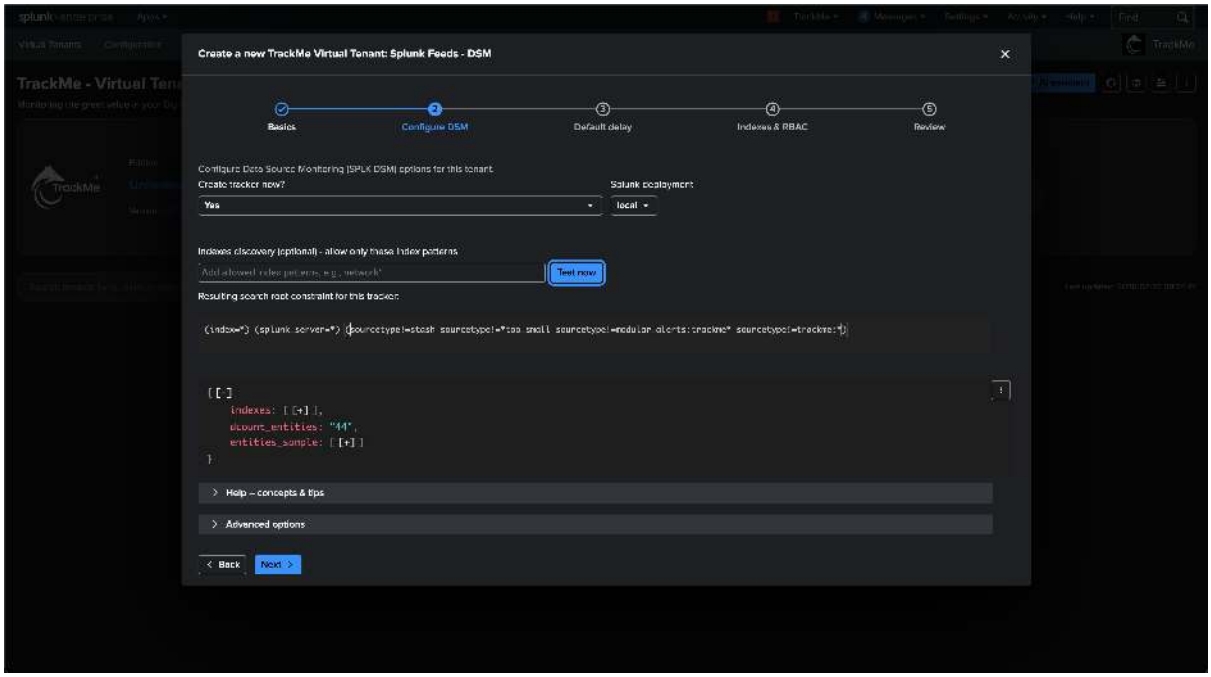
- tenant name: `secops`
- tenant alias: This is optional and can be updated at any time. If set, the alias replaces the tenant identifier in the Virtual Tenants UI.
- tenant description: This is optional and can be updated at any time. If set, the description is displayed in the Virtual Tenants UI.



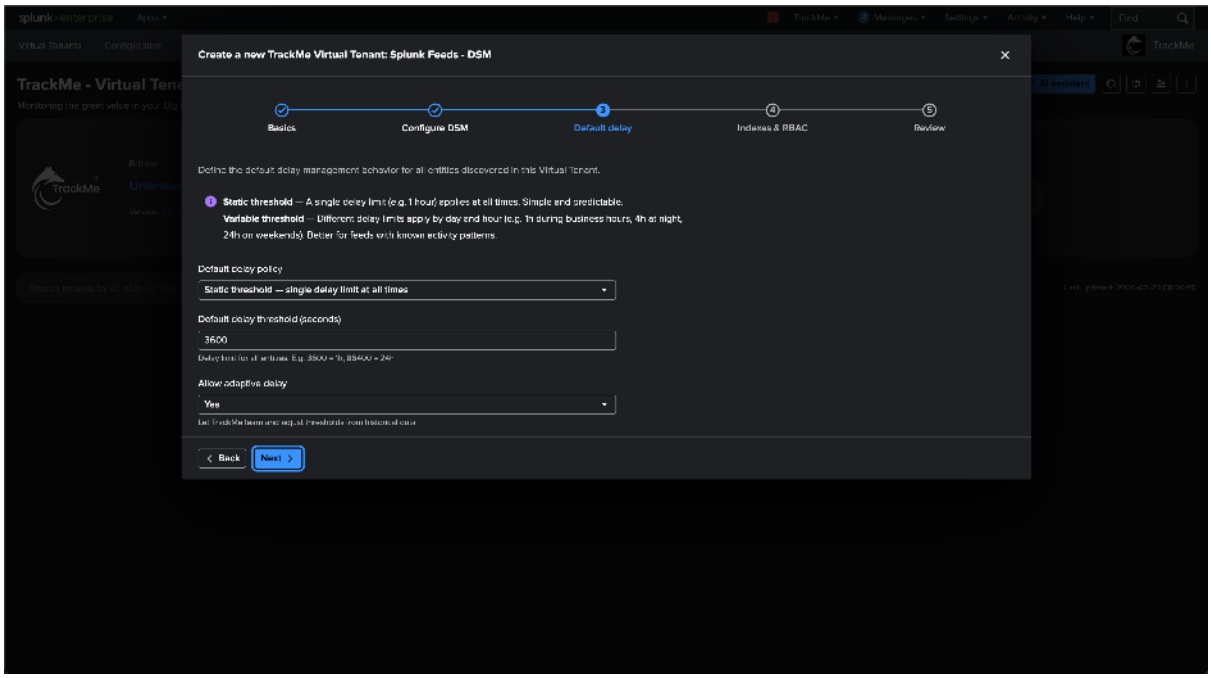
Creating tracker during the Virtual Tenant creation, or later on via the Hybrid Tracker management UI

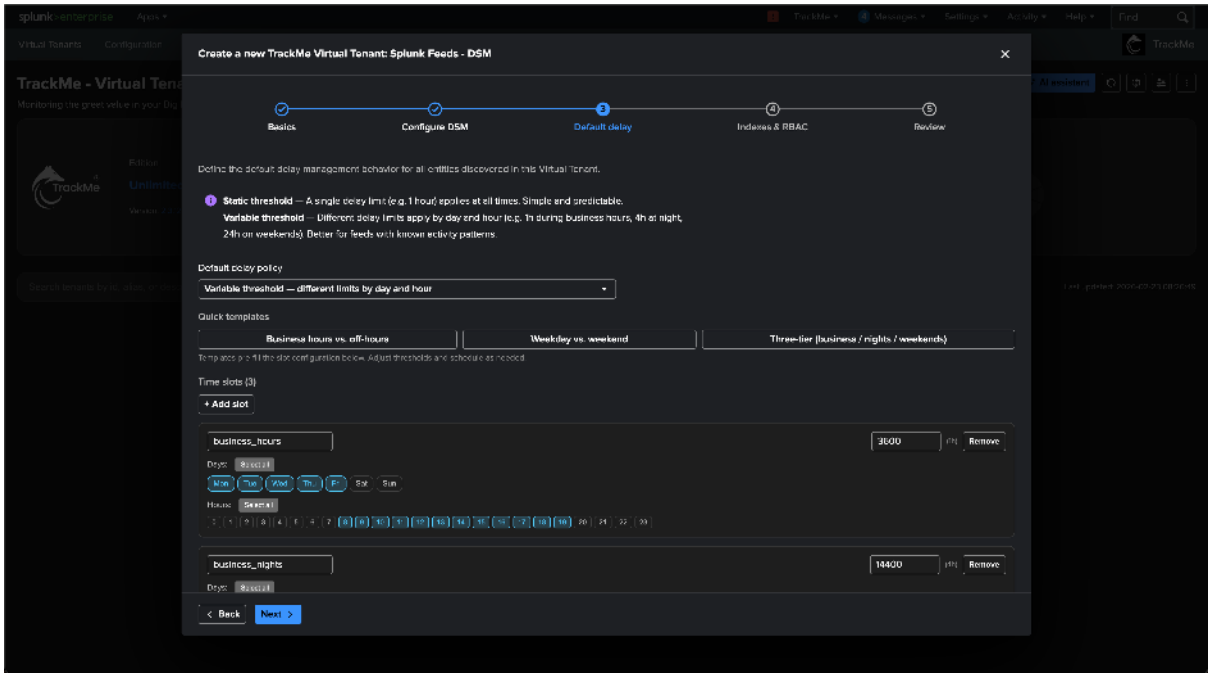
- If your environment is small enough (less than 2/3 TB per day of ingestion), you can leave the default configuration and create the tracker during this stage (so leave "Yes" for Create Hybrid tracker).
- For larger environments, and up to any super large environments, we recommend disabling the creation of the Virtual Tenant and creating the tracker later on manually via the Hybrid Tracker management UI.



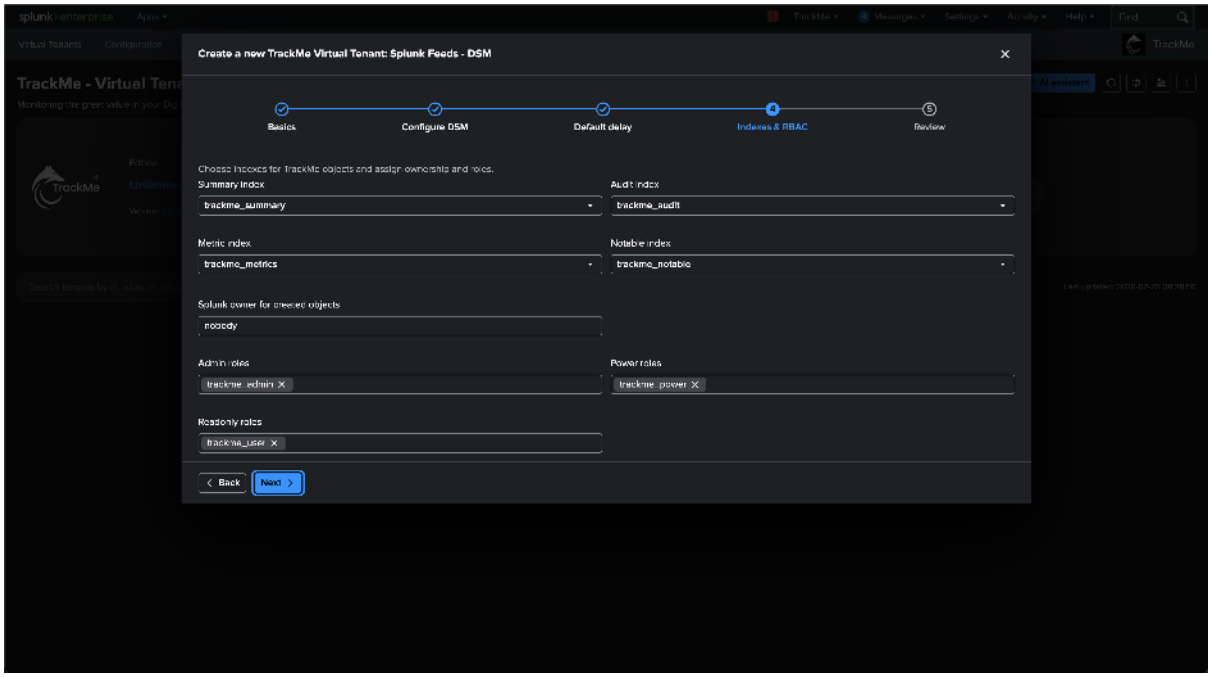


- **The next screen allows defining the default delay threshold behavior, TrackMe allows two different delay threshold policies:**
  - Static delay threshold: A single fixed threshold value that applies at all times. This is the default and traditional behavior in TrackMe.
  - Variable delay threshold: Time-aware thresholds that change dynamically based on the day of the week and the hour of the day. This allows defining tighter thresholds during business hours and more relaxed thresholds during nights, weekends, or known quiet periods.





- The final configuration screen and its top section allows defining TrackMe indexes, in most cases, the default indexes are fine, and we can leave the default configuration.



- The second part of that same screen allows defining ownership and permissions, all of these can be easily modified via TrackMe at any point in time, in short:

Owner:

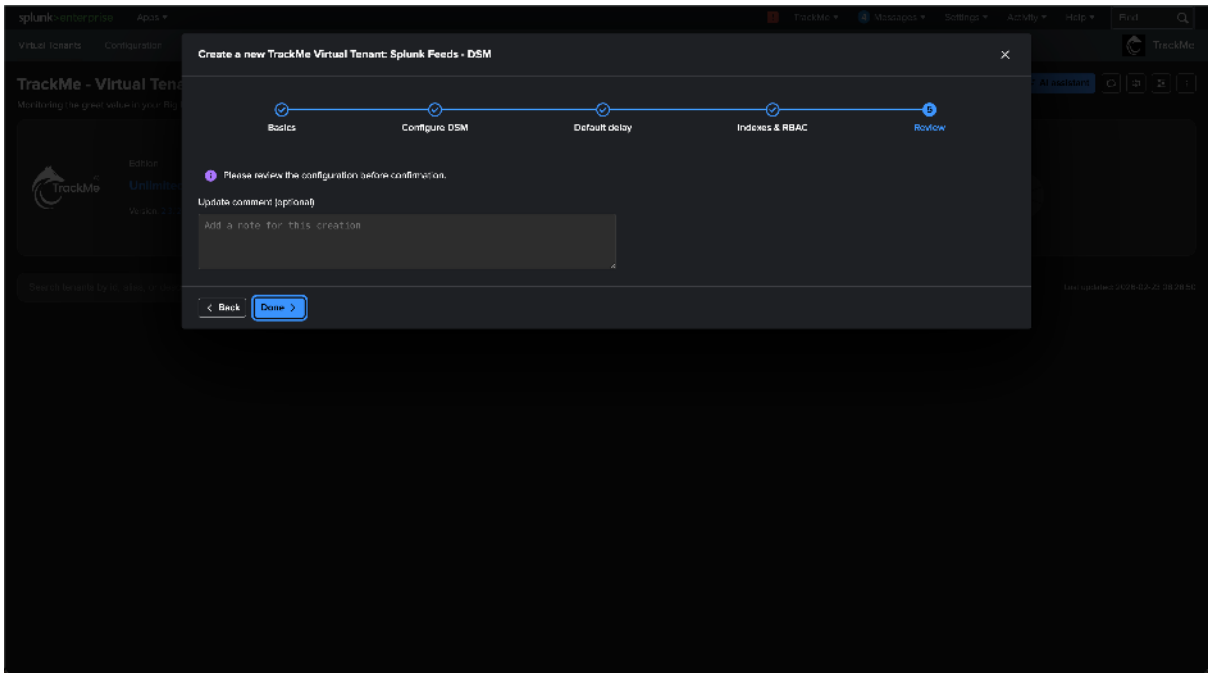
- Default owner is nobody, which refers to the splunk-system-user, this means that TrackMe will automatically reassign the ownership of the knowledge objects to the splunk-system-user.
- This is fine for most cases and is a common practice in Splunk.
- On large or strict environments, creating a service account can be considered as good practice and notably allows tracking TrackMe's related activity easily, leveraging Splunk Workload Management, see: :ref: *trackme_admin_config* for more information.

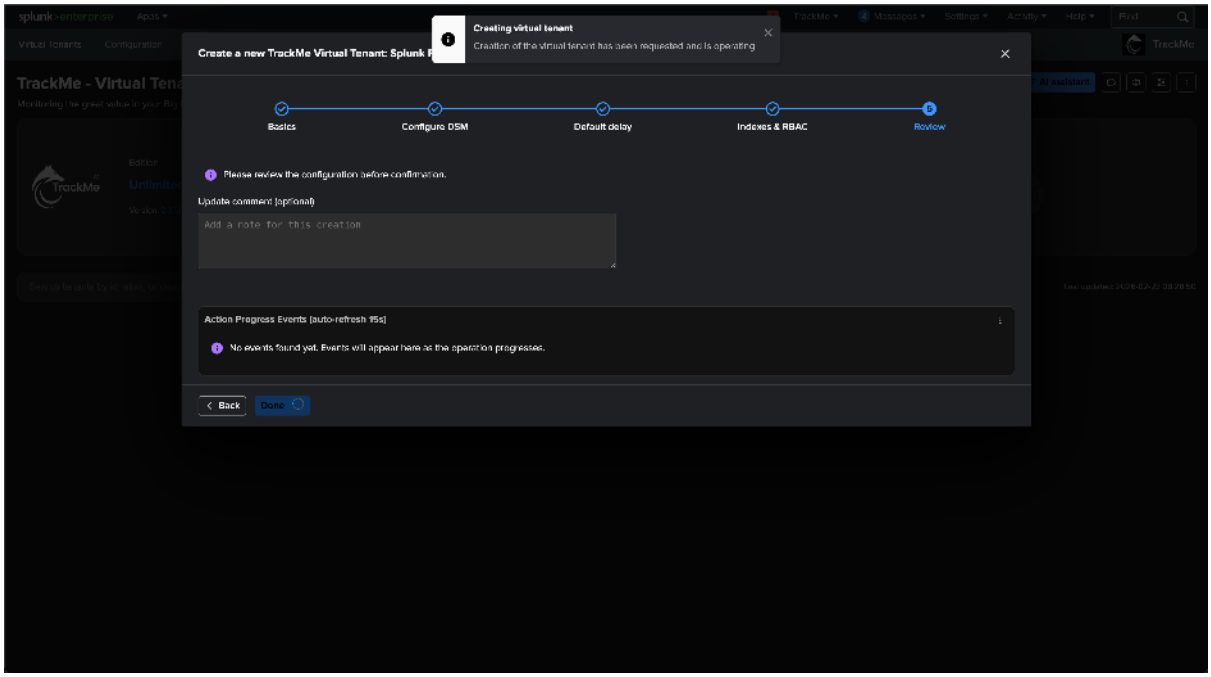
Permissions:

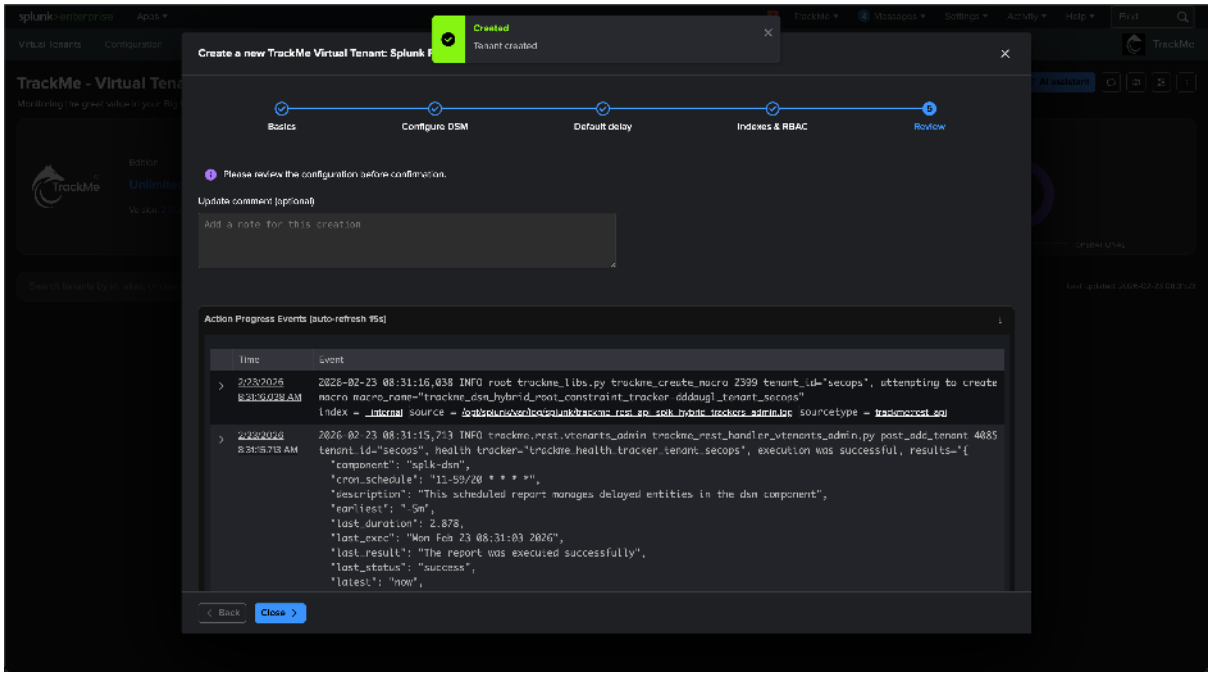
- The default permissions leverage a 3-dimensional model, from admin to power and finally read-only users.
- As a start, you generally can leave everything as default

Validate the creation of the Virtual Tenant:

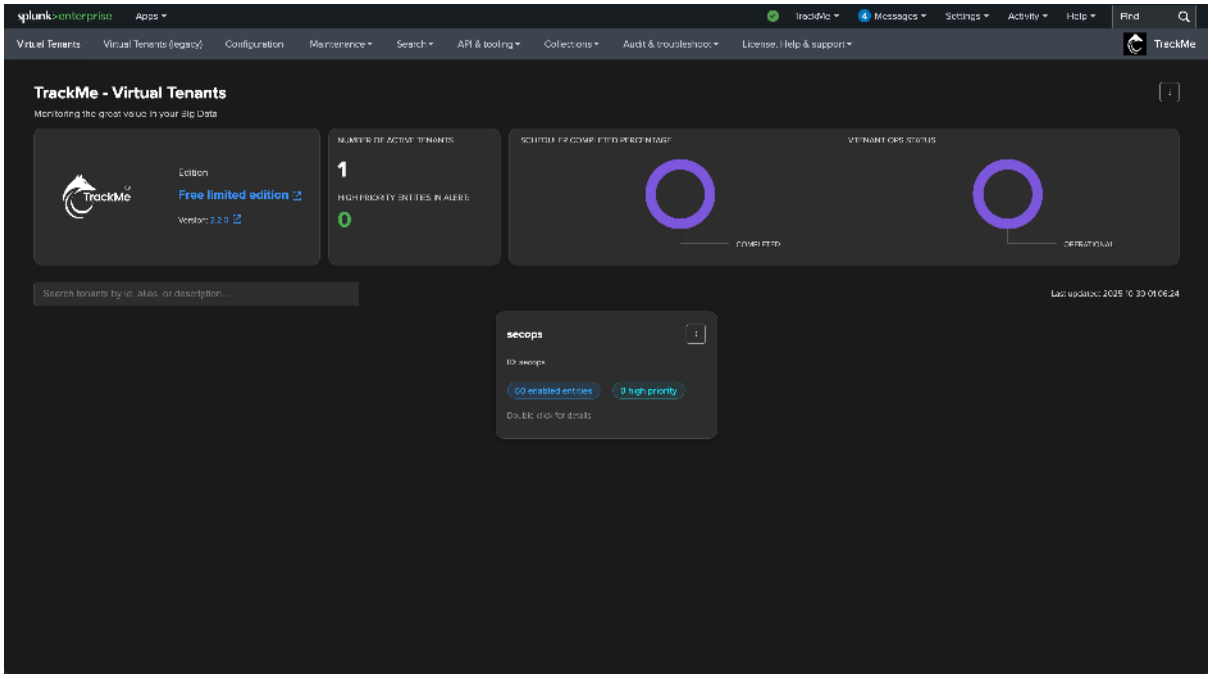
We can now validate the creation of the Virtual Tenant, after a short moment, the Virtual Tenant UI will be refreshed and a Virtual Tenant will now be available:



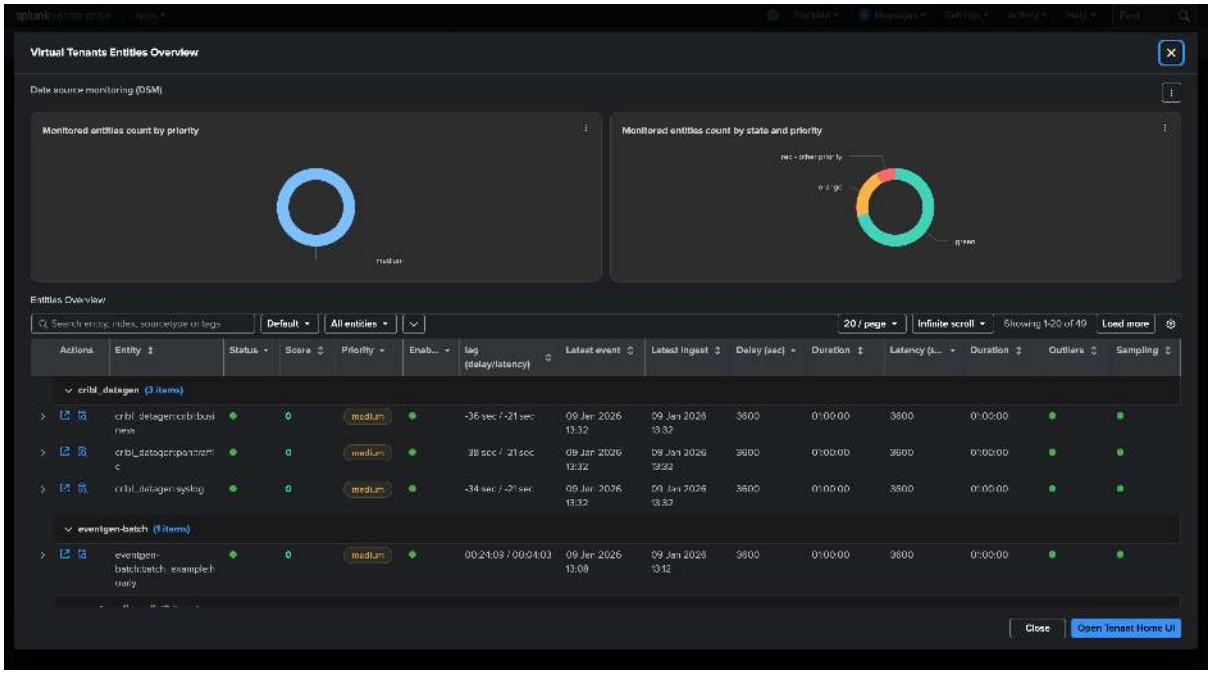




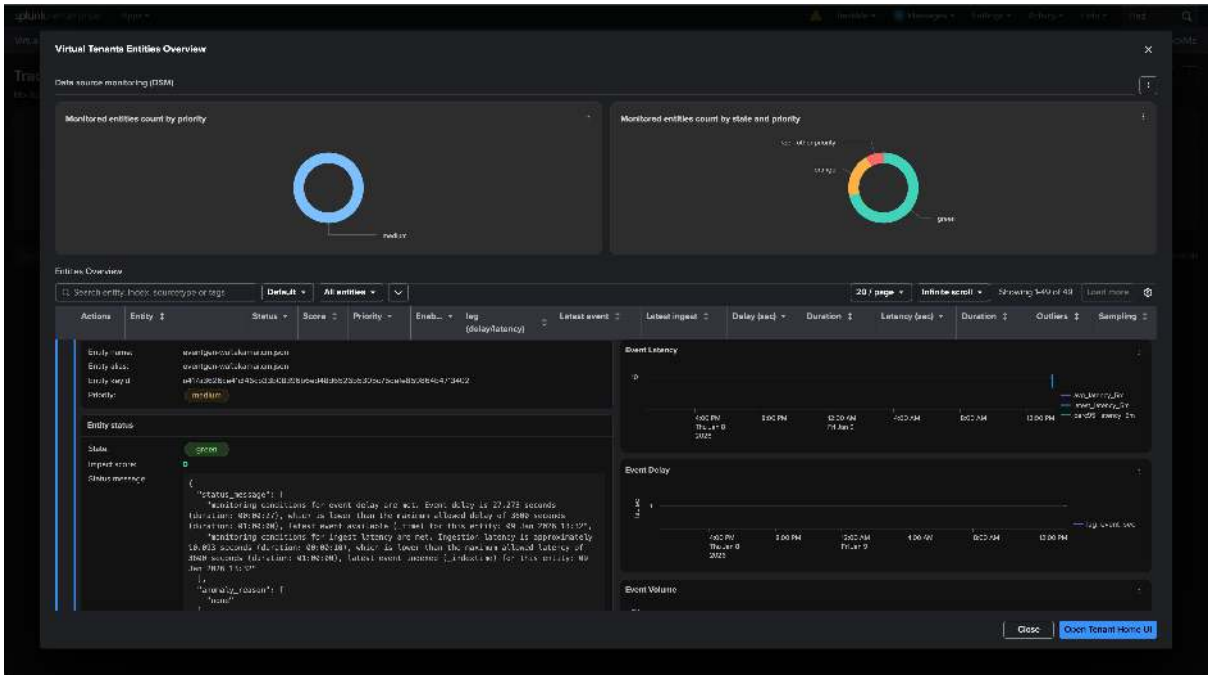
After a few minutes, the Virtual Tenant user interface will show activity of the tenant and eventually entities that were discovered:



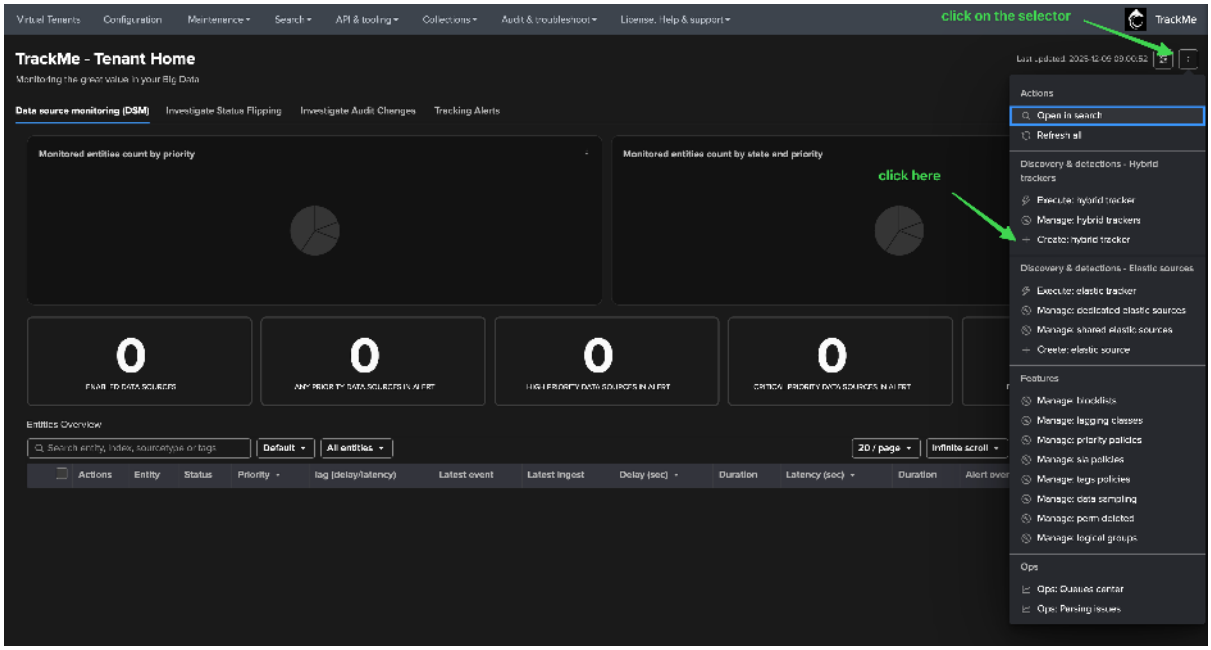
Double click on the Virtual Tenant box to open the following screen:

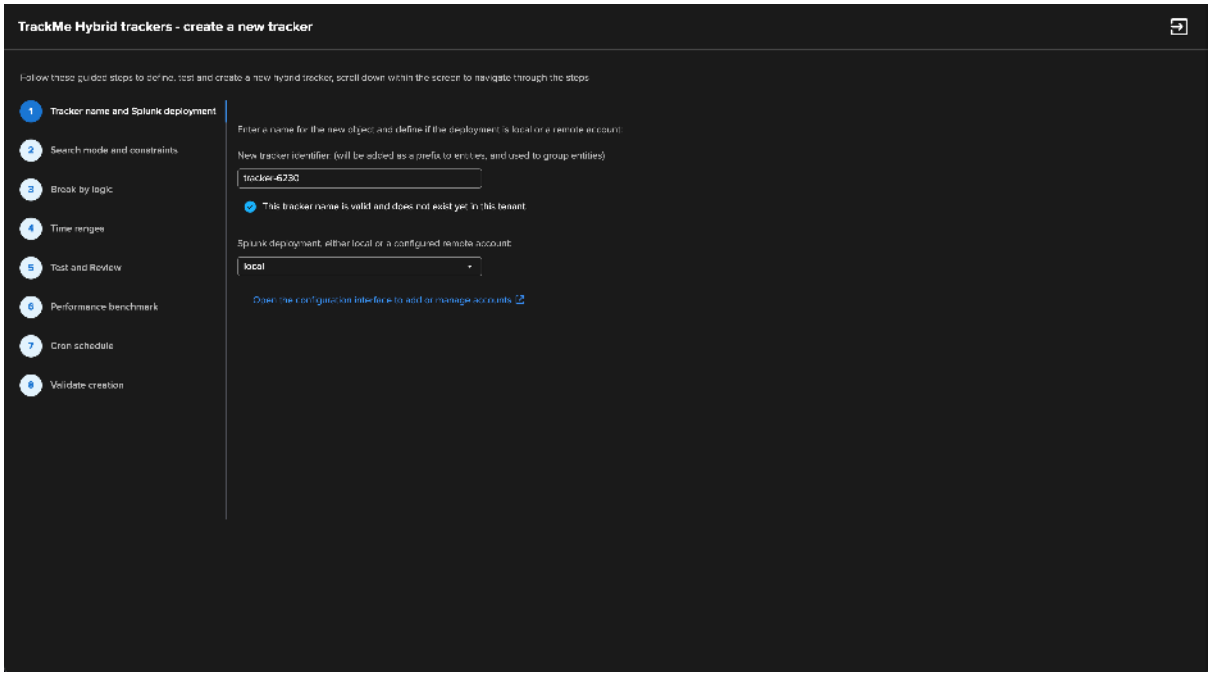


You can preview various information about the entities, and access the Home user interface for the tenant:

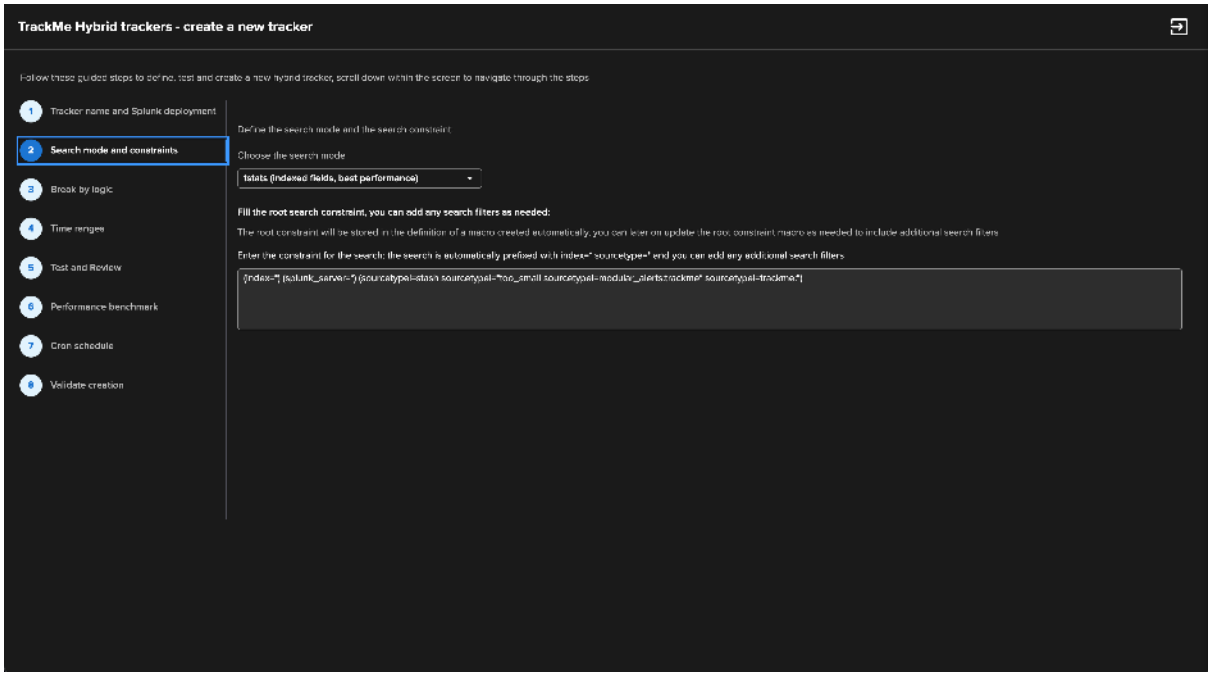


Optional: Create a hybrid tracker for splk-dsm after the creation of the Virtual Tenant





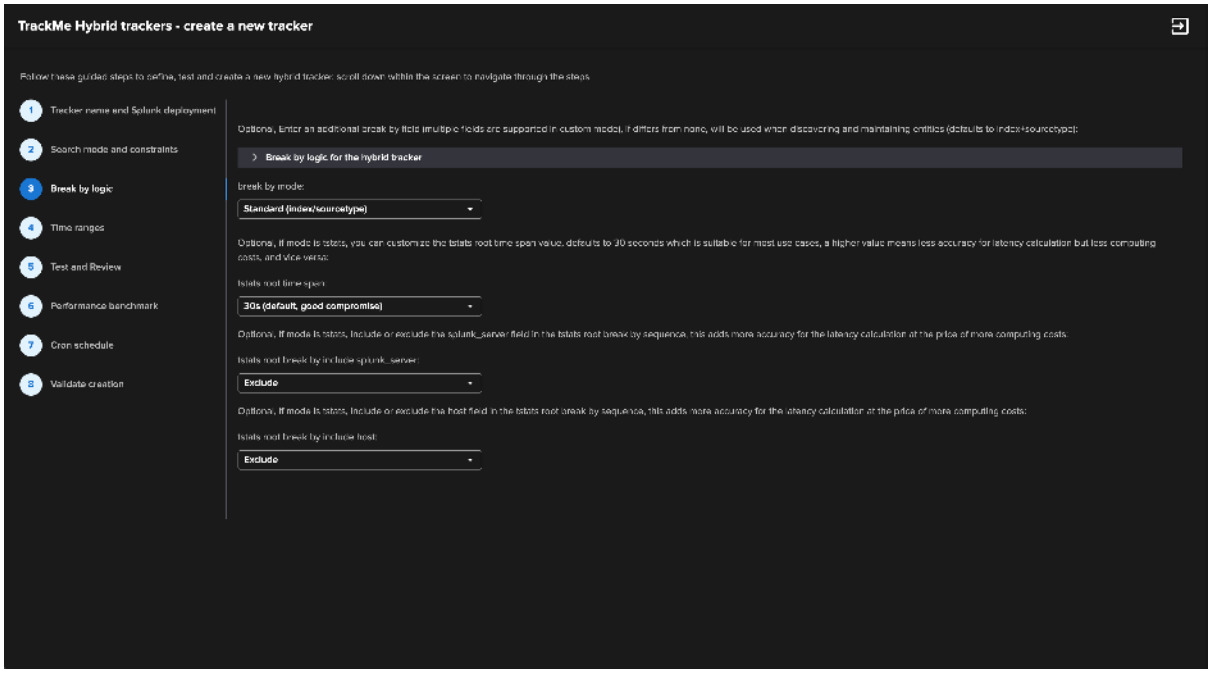
- Go to Step 2: (Search mode and constraints)
  - The default root constraint tracks all indexers, and excludes summary data (stash) and TrackMe-related data.
  - On large environments, the scaling concept for TrackMe is to dedicate Hybrid Trackers to specific indexes scope, so trackers can share the load and be executed concurrently, allowing TrackMe to scale at any level.
  - On relatively small environments, the default configuration for a single tracker is in most cases able to cope with the load with no issues.
  - Finally, if you are a Foundation Edition trial user, you can create up to eight Hybrid Trackers in total.
  - For more information about scaling TrackMe for high-scale environments, please refer to the TrackMe documentation, notably: *Large-scale environments*



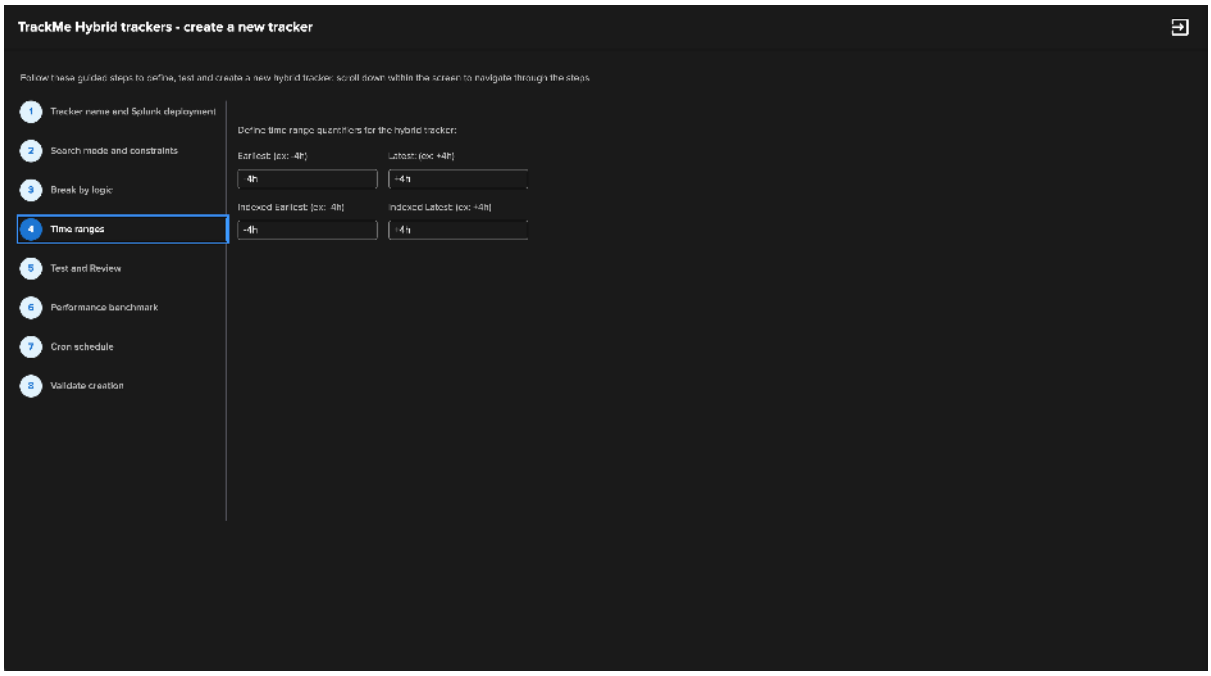
About Trackers scope:

- When creating a Hybrid Tracker, TrackMe creates a dedicated Splunk macro which contains the scope of the tracker.
- You can therefore easily start with a restricted scope such as a list of index patterns, then modify this macro at any point in time to include additional contexts.
- To retrieve the macro name after the tracker is created, click on Manage Hybrid tracker from the home UI.

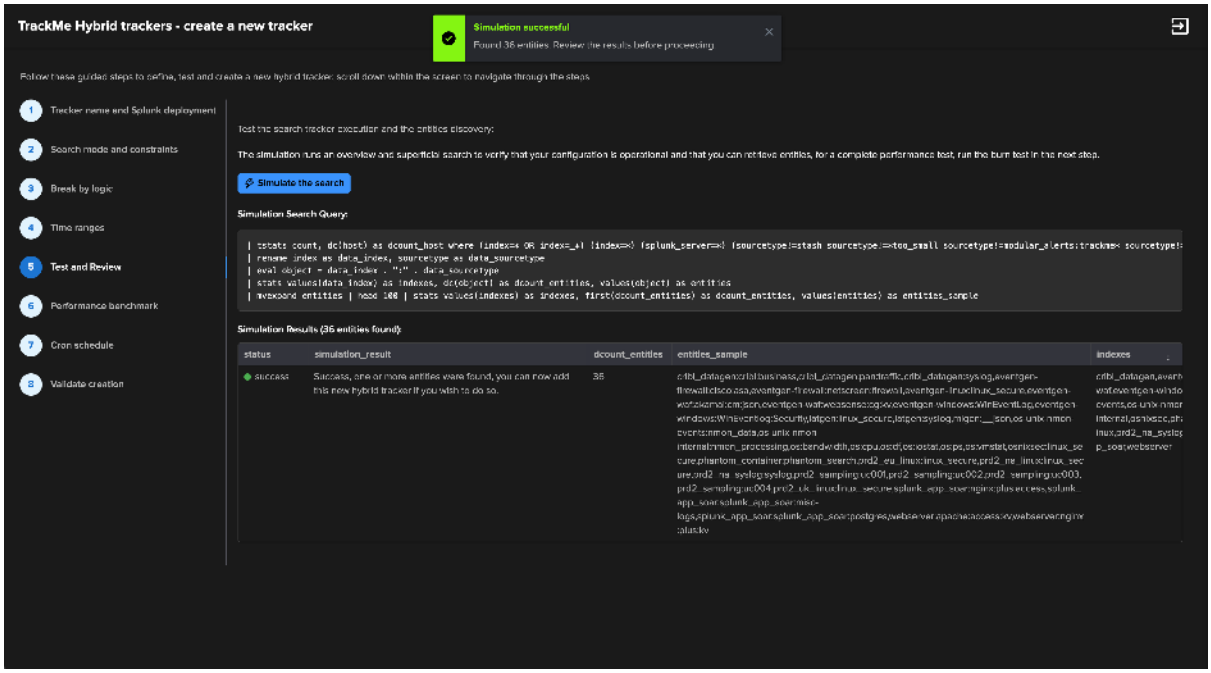
- Go to Step 3: (Break by logic)
  - By default, the behavior is to create TrackMe entities as the combination of indexes and sourcetypes, therefore: `<index>:<sourcetype>`, in most cases, this is what you want.
  - An alternative, and depending on your requirements, is to use the `merge` mode, in this case TrackMe will create one entity per index: `<index>`.
  - Finally, you can also specify a custom list (comma-separated format) of fields as part of a custom break by, this is more sophisticated and is generally used for specific use cases such as leveraging custom indexed fields in large-scale environments.



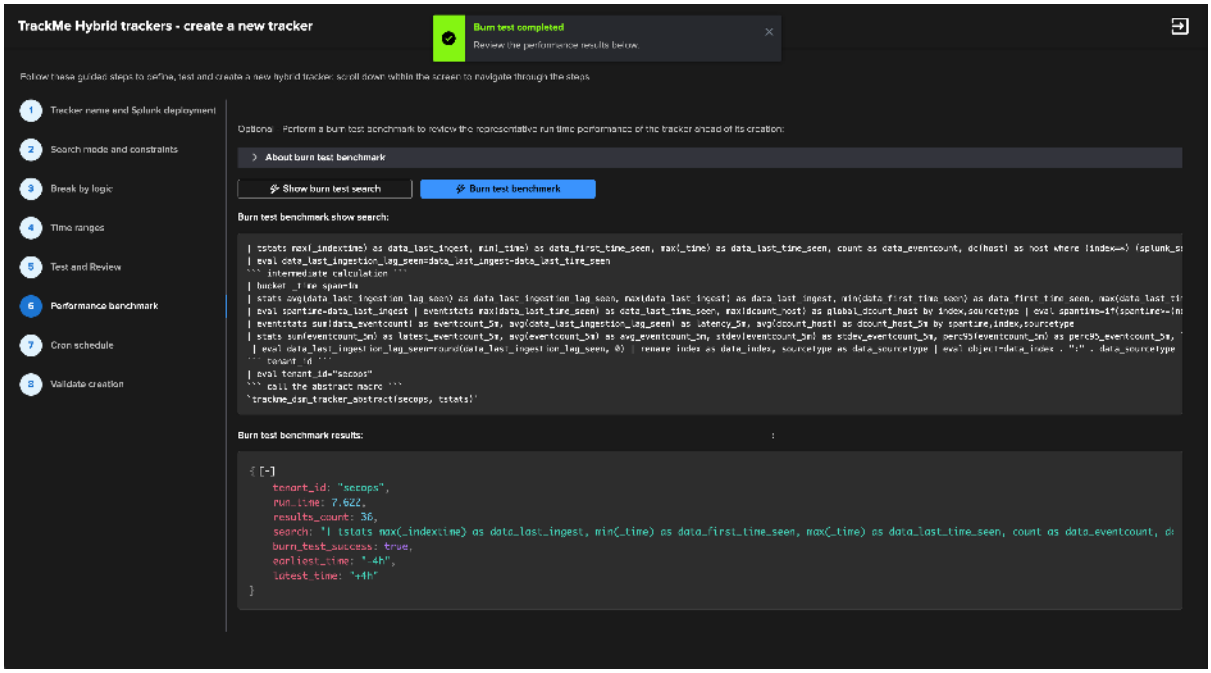
- Go to Step 4: (Time ranges)
  - By default, TrackMe uses a -4h/+4h time range for the search logic, this is generally fine for most cases.
  - In some contexts such as very high-scale contexts, or limited capacity contexts, you can eventually reduce the earliest to reduce the associated costs and run time (most likely for splk-dhm than splk-dsm).



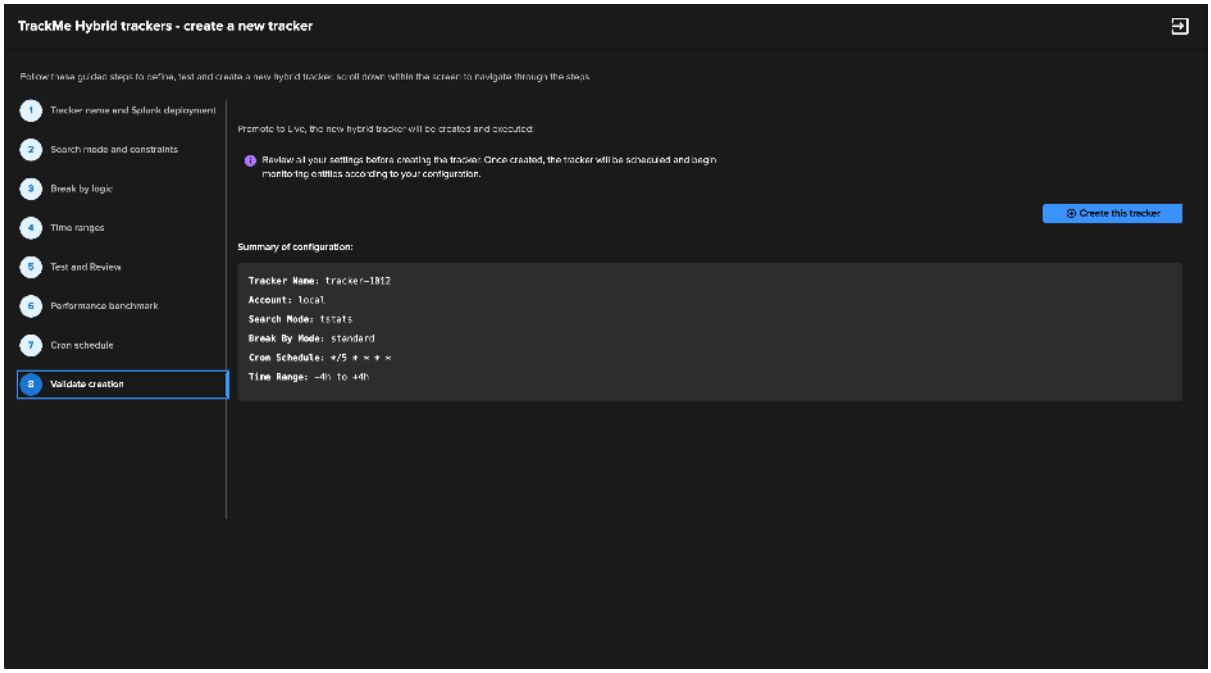
- Go to Step 5: (Test & Review)
  - This steps allows simulating the search logic and its parameters, to preview the results and notably the number of entities that would be created.

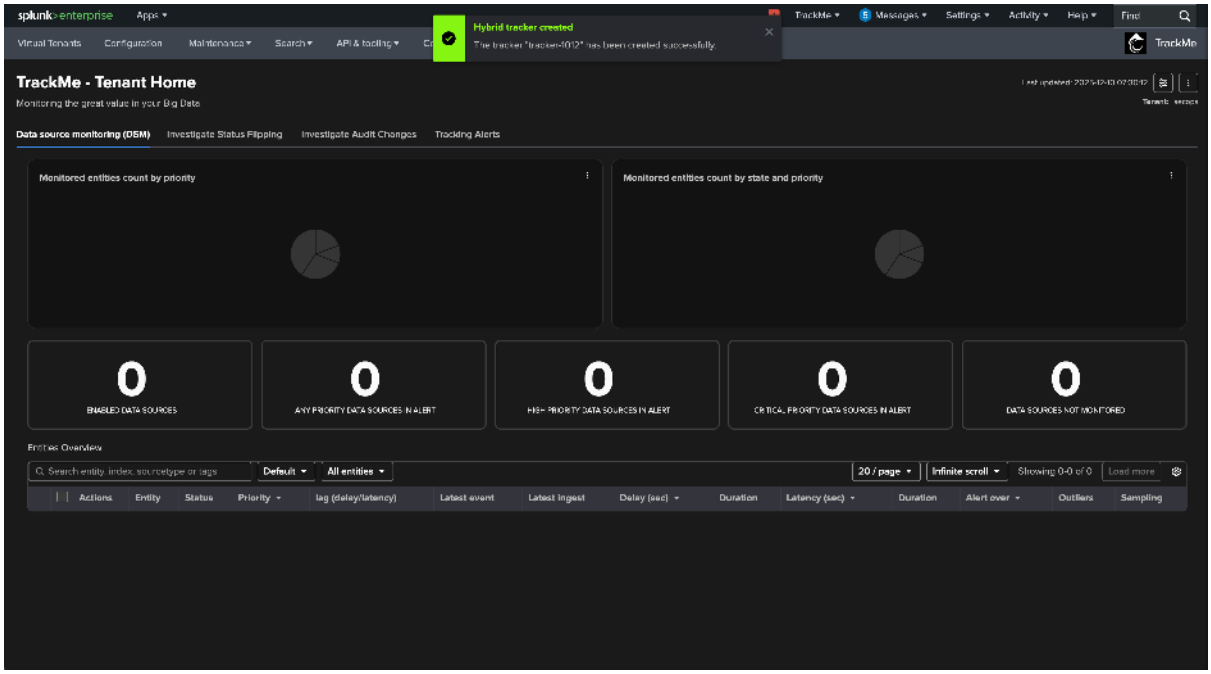


- Go to Step 6: (Performance benchmark)
  - TrackMe allows benchmarking the most expensive part of the search logic, this provides an accurate enough idea of how long the tracker will take to execute.
  - The target is to execute the search every 5 minutes (while this is under your control, in most cases we want to get as close as possible to this frequency, especially for feeds tracking).
  - This means that the search needs to be executed in less than 300 seconds to avoid generating skipping searches and missing executions.
  - The scaling idea underneath is to create multiple trackers with a reduced scope, so they can be executed concurrently, and therefore reduce the execution time.

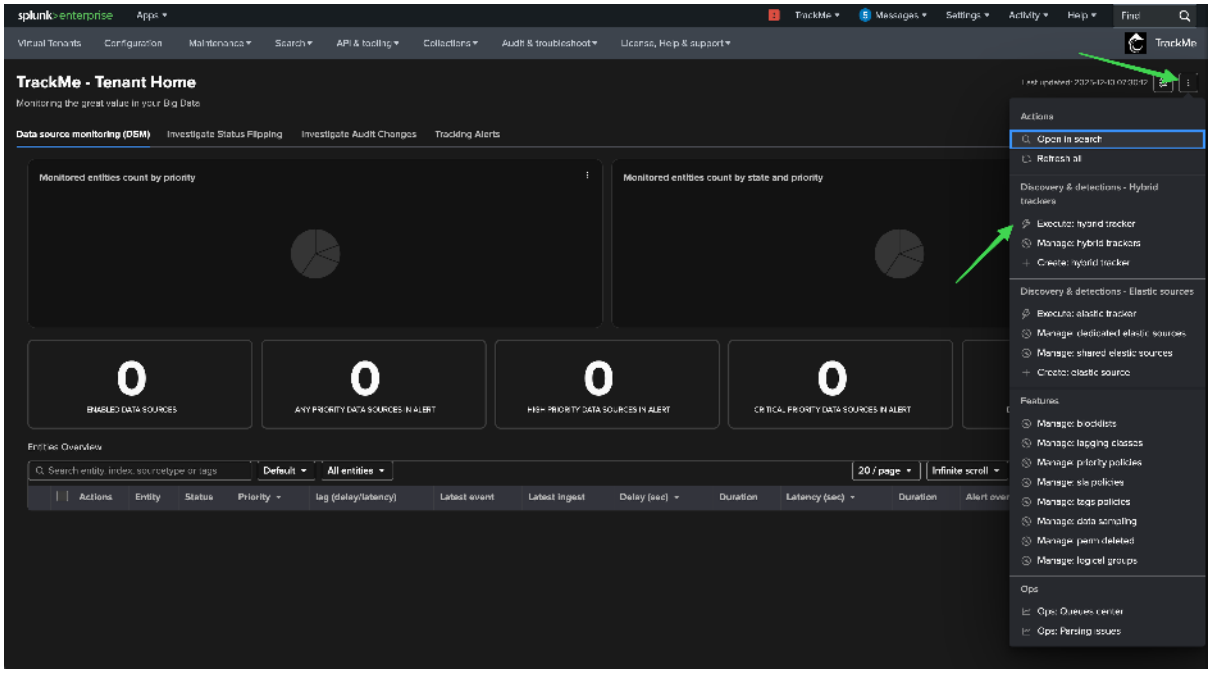


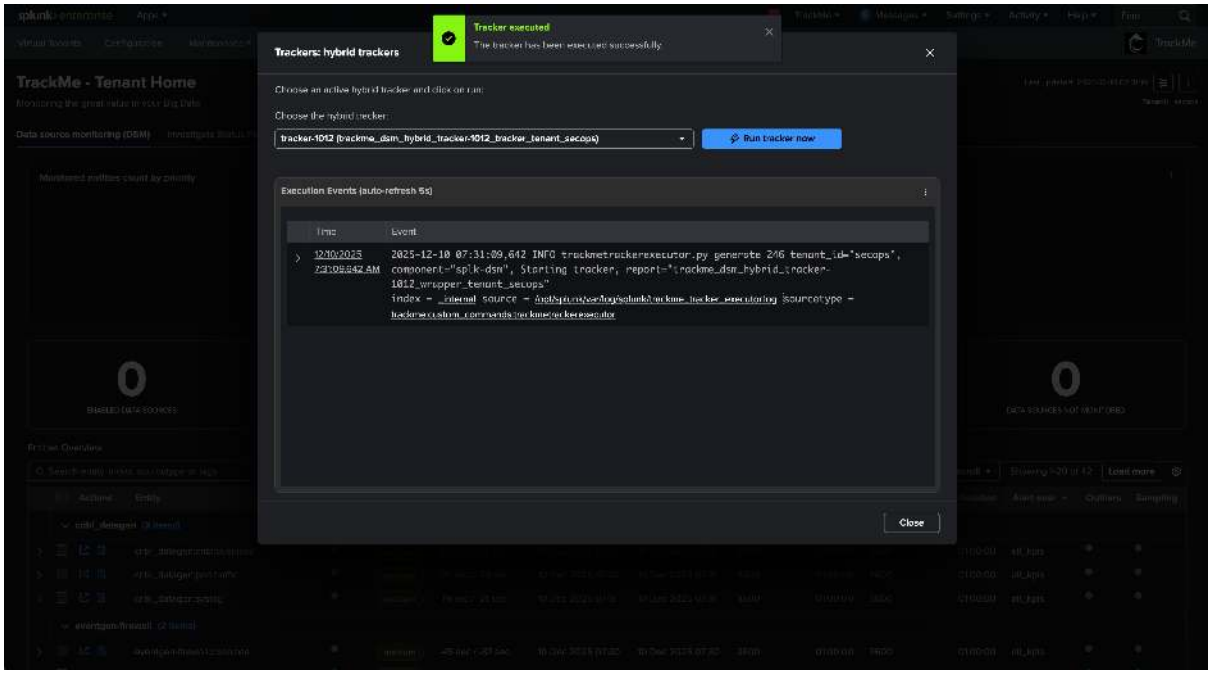
- Finally, go step “Validate creation” to create the tracker

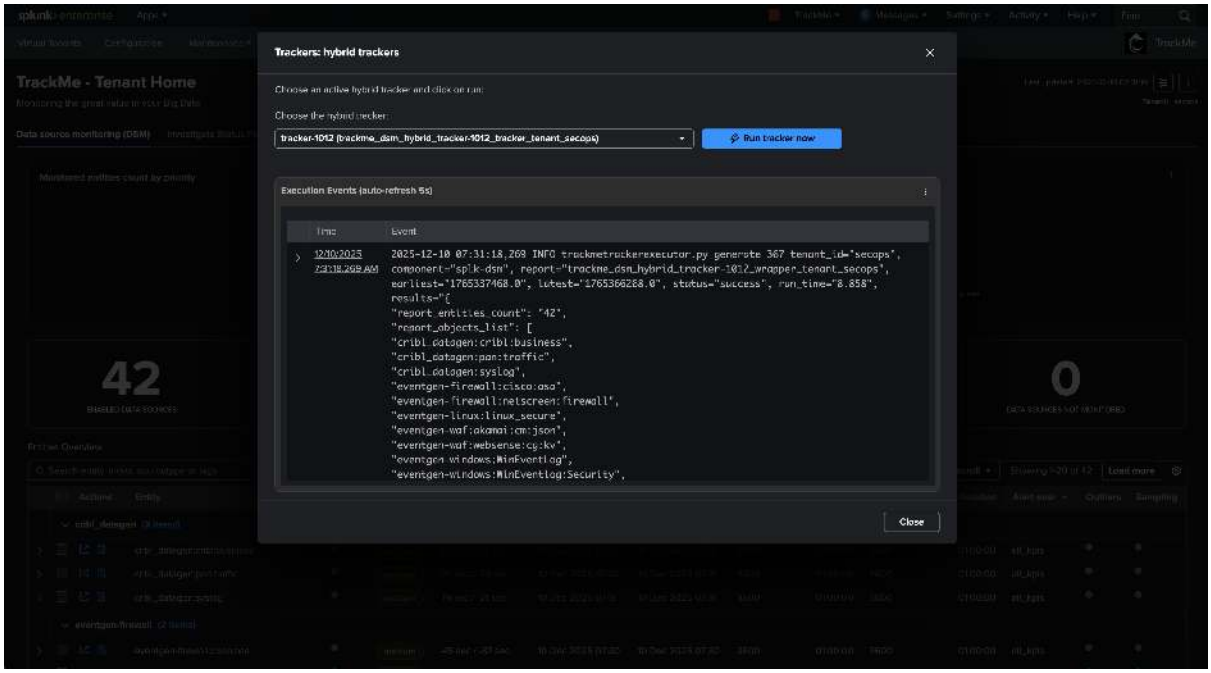




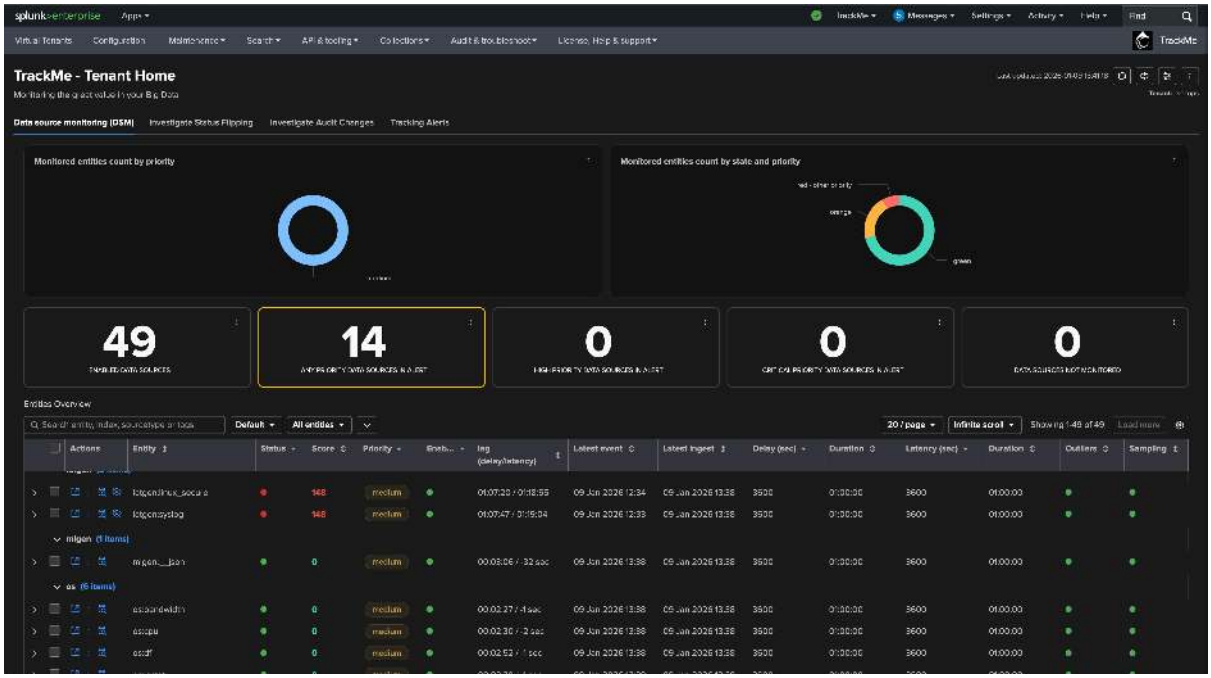
- Once created, you can from the main menu selector request the immediate execution of the tracker:


TrackMe - Tenant Home
Monitoring the great value in your Big Data
Data source monitoring (DSM)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Monitored entities count by priority
Monitored entities count by state and priority
Actions
Open in search
Refresh all
Discovery & detections - Hybrid trackers
Execute: hybrid tracker
Manage: hybrid trackers
Create: hybrid tracker
Discovery & detections - Elastic sources
Execute: elastic tracker
Manage: dedicated elastic sources
Manage: shared elastic sources
Create: elastic source
Features
Manage: blocklists
Manage: tagging classes
Manage: priority policies
Manage: sla policies
Manage: tags policies
Manage: data sampling
Manage: perm deleted
Manage: logical groups
Ops
Ops: Queues center
Ops: Parsing issues
0
ENABLED DATA SOURCES
0
ANY PRIORITY DATA SOURCES IN ALERT
0
HIGH PRIORITY DATA SOURCES IN ALERT
0
CRITICAL PRIORITY DATA SOURCES IN ALERT
Entities Overview


Tracker executed
Trackers: hybrid trackers
Choose the hybrid tracker:
tracker-1012 (trackme_dsm_hybrid_tracker-1012_tracker_tenant_secops)
Run tracker now
Execution Events (auto-refresh 5s)
Time
Event
Close



- After the tracker for first executed, entities will be created and visible in the main user interface:



Step 2: Let's create a Virtual Tenant for data host sources

We will now create a Virtual Tenant to start tracking hosts, we recommend to setup hosts tracking in a dedicated tenant, so we can easily manage various features such as disabling ML Outliers:

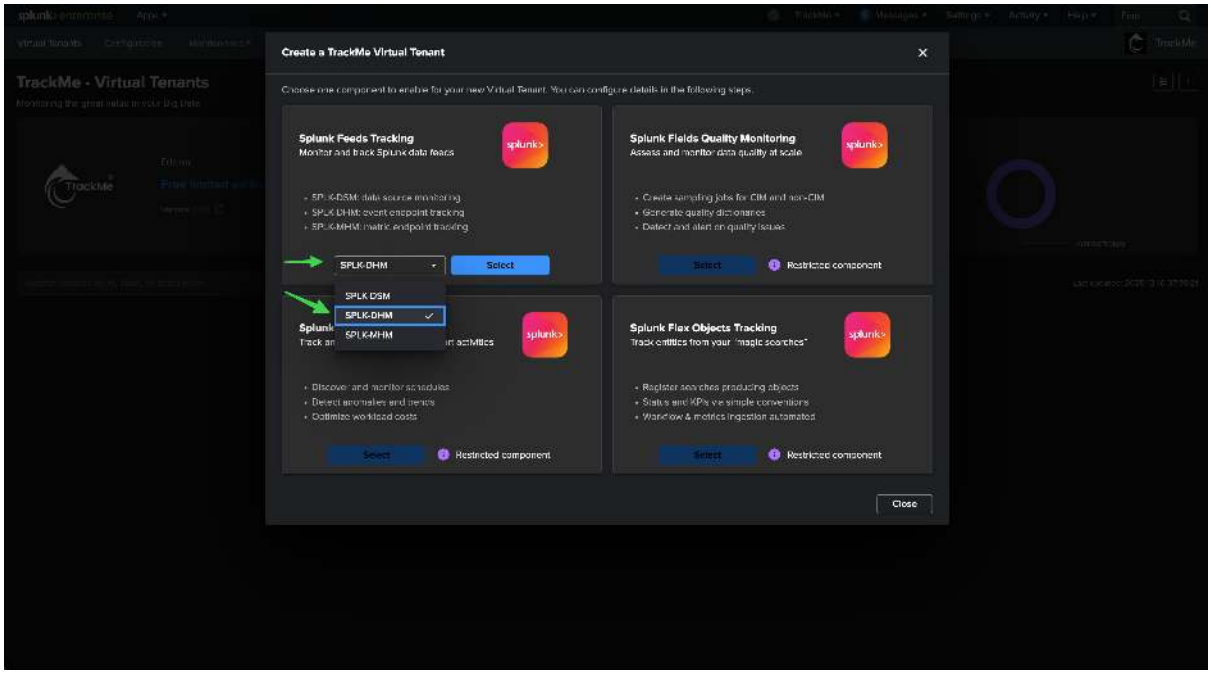
- component: `splk-dhm`
- tenant name: `endpoints`

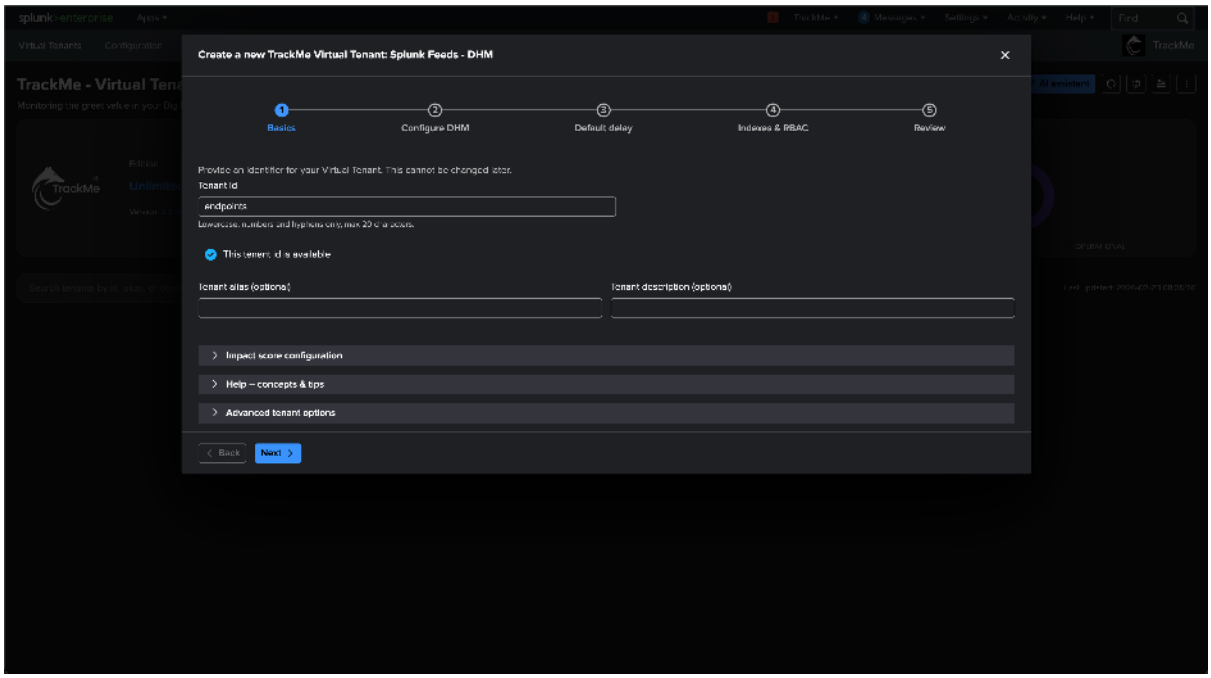
Repeat the same process for data hosts tracking:

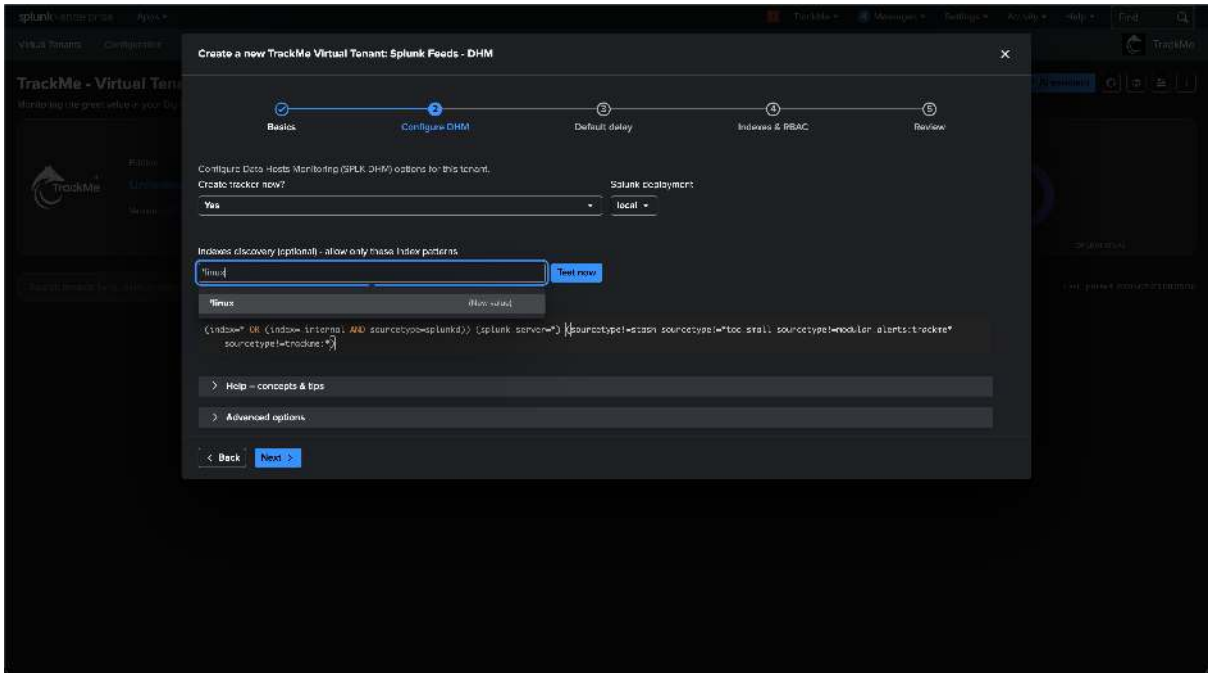
- We recommend being very strict with splk-dhm, notably we recommend restricting the scope as much as possible, and ensuring to allow only indexes that related to data generated by endpoints themselves.
- You totally can start with a certain scope, so you can first monitor the costs of the associated backends in

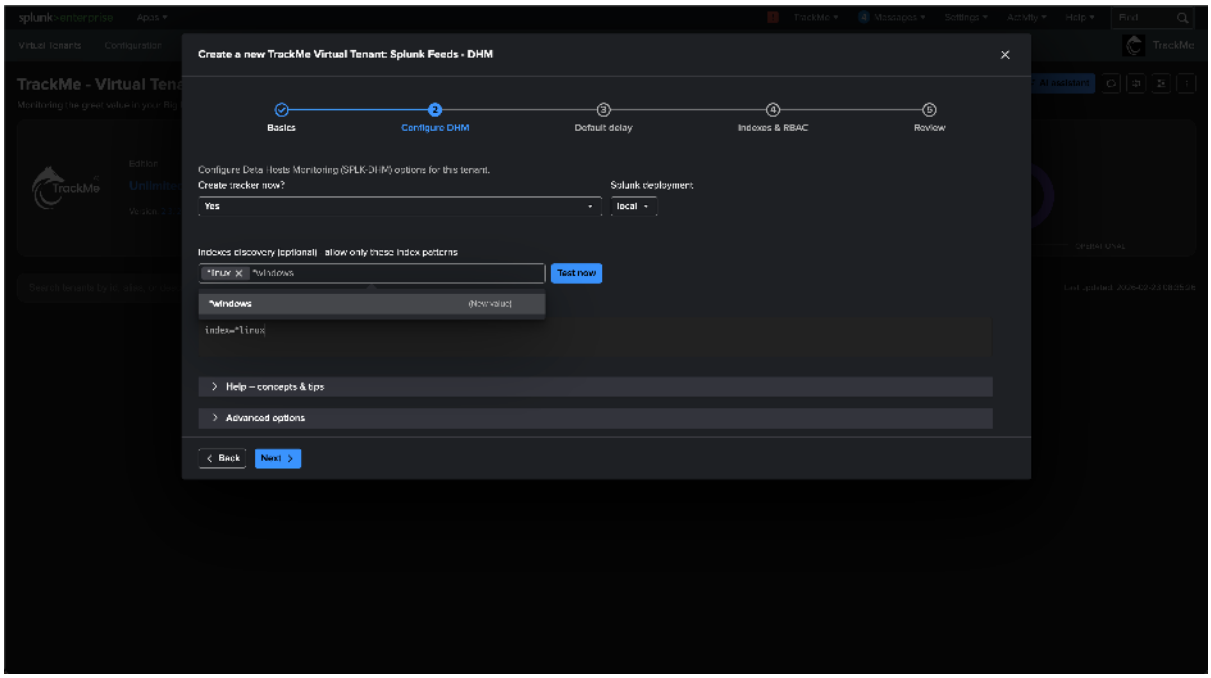
TrackMe, then gradually increase the scope as needed.

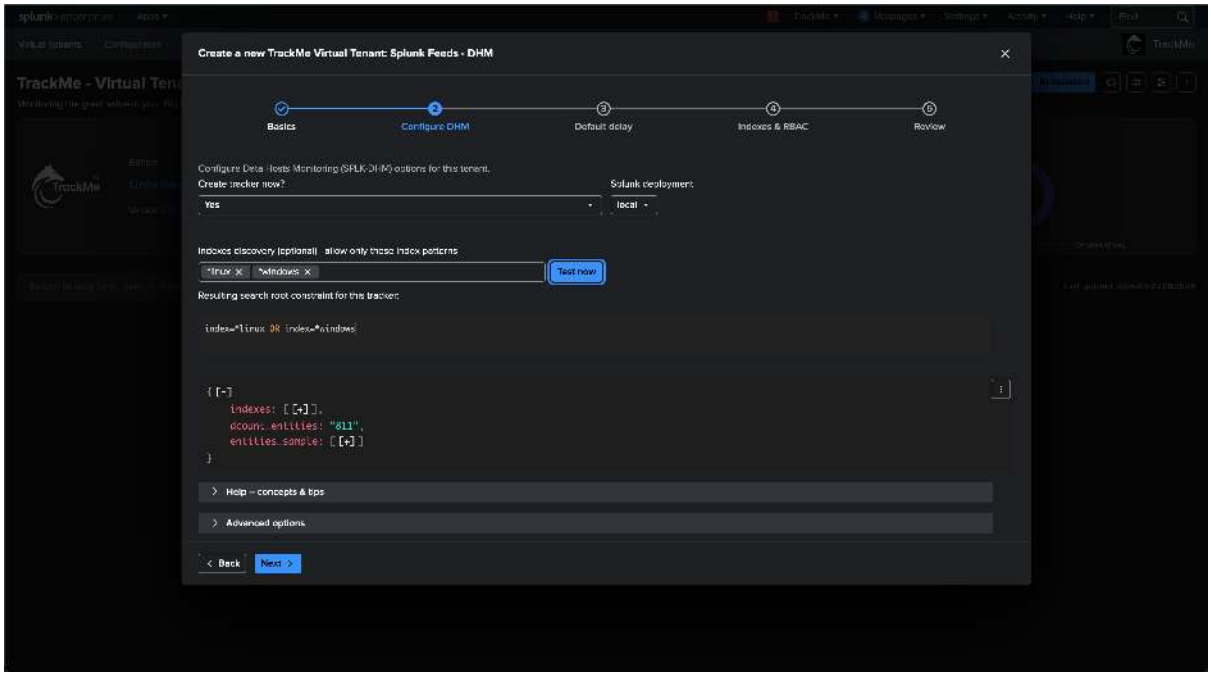
- Finally you can also choose to include or not splunkd itself, if you wish to monitor the availability of Splunk UFs/UFs internals, which reflect a healthy connectivity and status of Splunk agents.



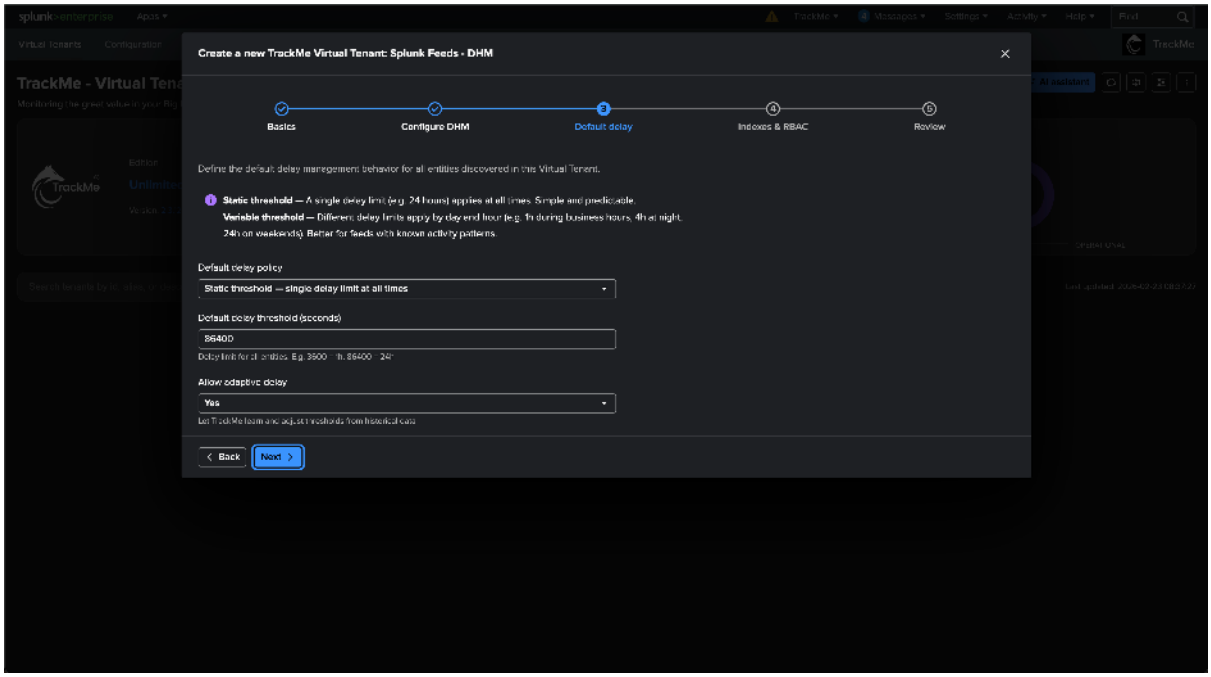


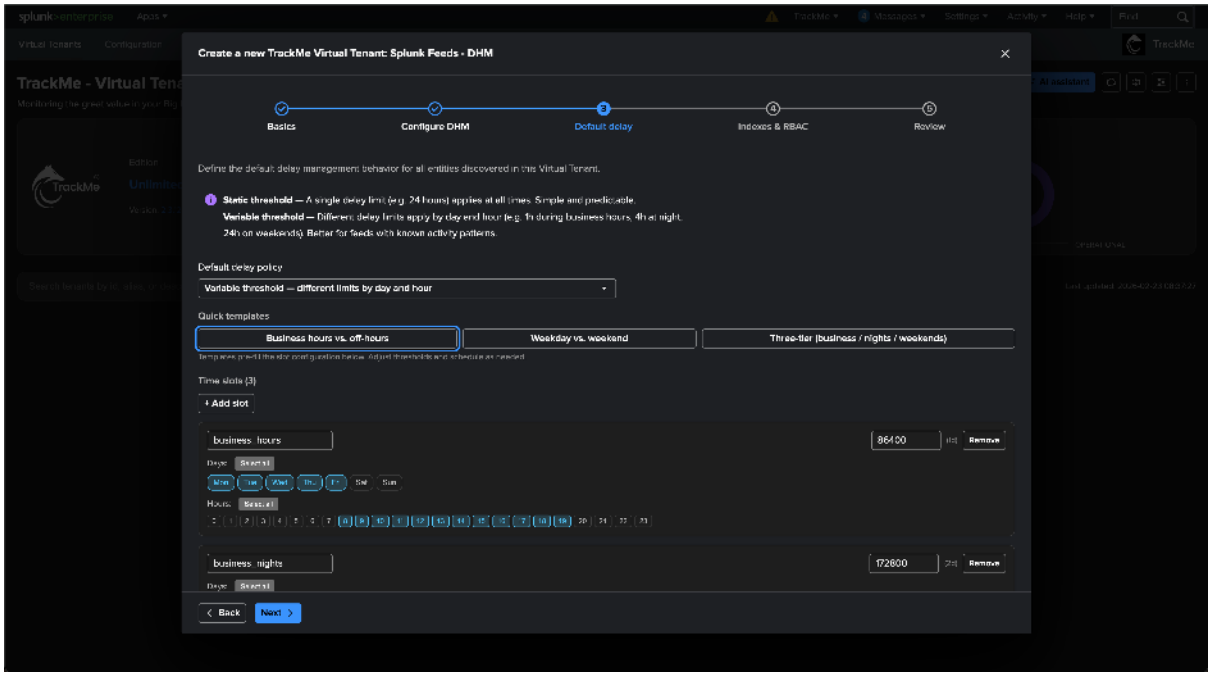

Create a new TrackMe Virtual Tenant: Splunk Feeds - DHM
Basics
Configure DHM
Default delay
Indexes & RBAC
Review
Configure Data Hosts Monitoring (SPLK-DHM) options for this tenant.
Create tracker now?
Yes
Splunk deployment:
local
Indexes discovery (optional) - allow only these index patterns
*linux
Test now
*linux
(New value)
(index=* OR (index=_internal AND sourcetype=splunkd)) (splunk_server=*) (sourcetype!=stash sourcetype!=*too small sourcetype!=module_alerts:trackme* sourcetype!=trackme:*)
Help – concepts & tips
Advanced options
Back
Next


Create a new TrackMe Virtual Tenant: Splunk Feeds - DHM
Basics
Configure DHM
Default delay
Indexes & RBAC
Review
Configure Data Hosts Monitoring (SPLK-DHM) options for this tenant.
Create tracker now?
Yes
Splunk deployment:
local
Indexes discovery (optional) - allow only these index patterns
*linux
*windows
Test now
*windows
(New value)
index=*linux
Help – concepts & tips
Advanced options
Back
Next

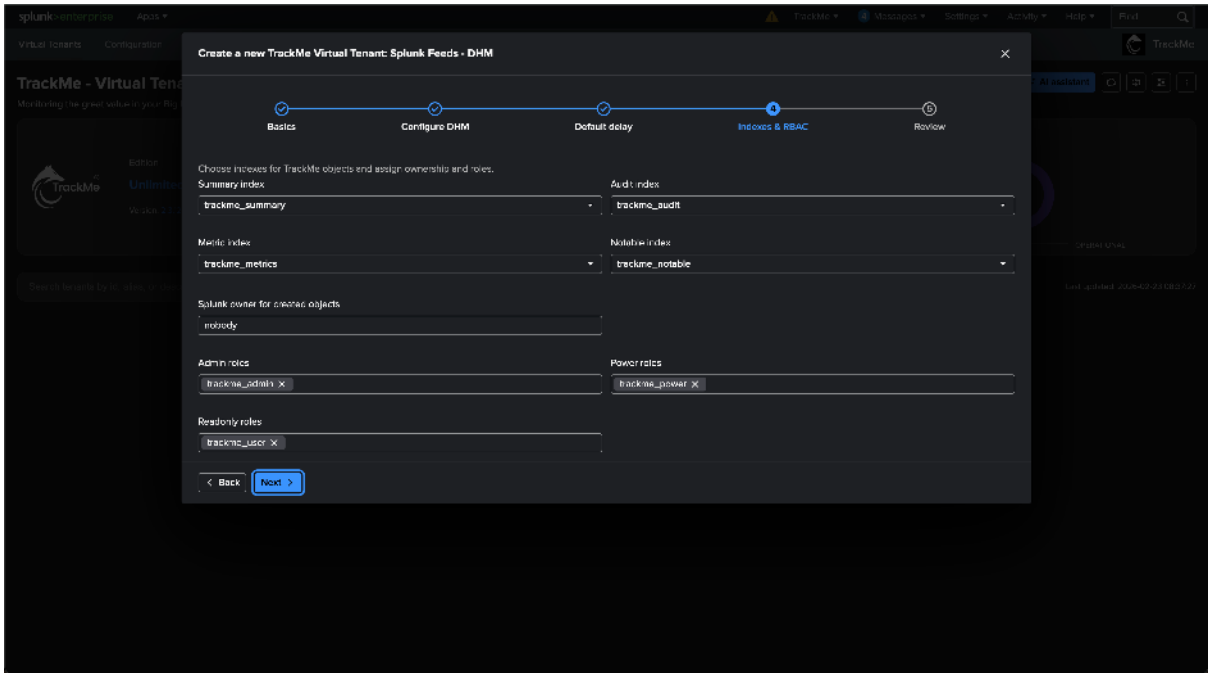


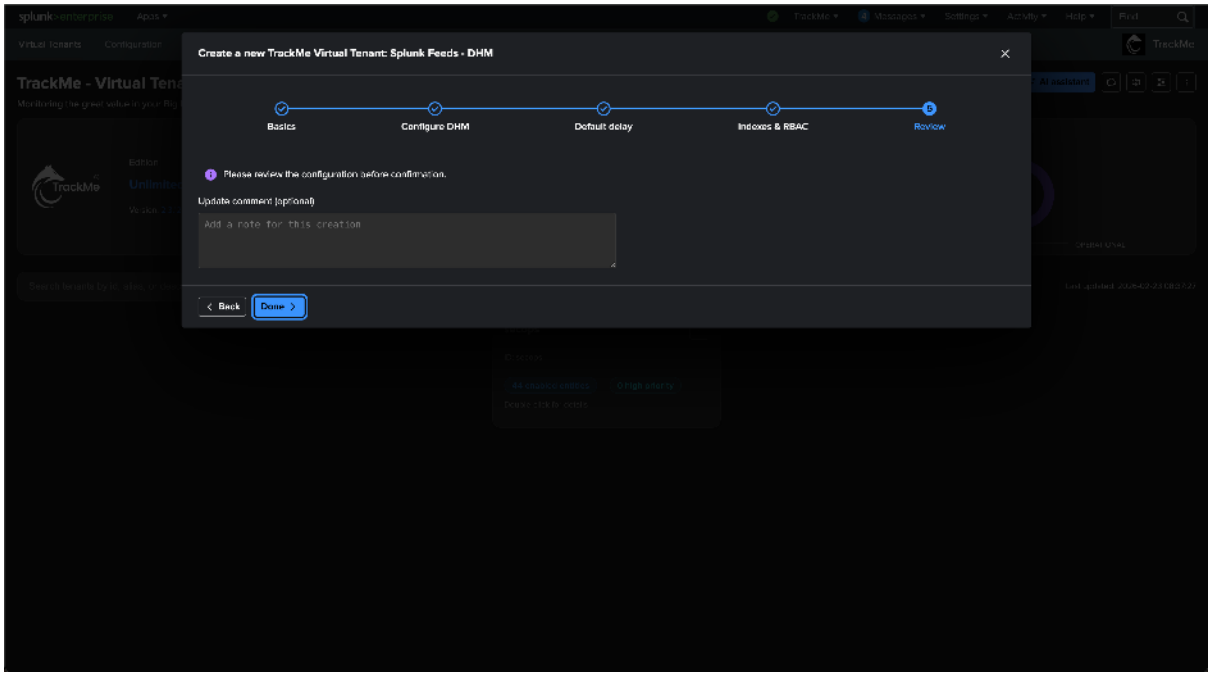
Default delay threshold behavior, choose between a static or variable delay threshold:

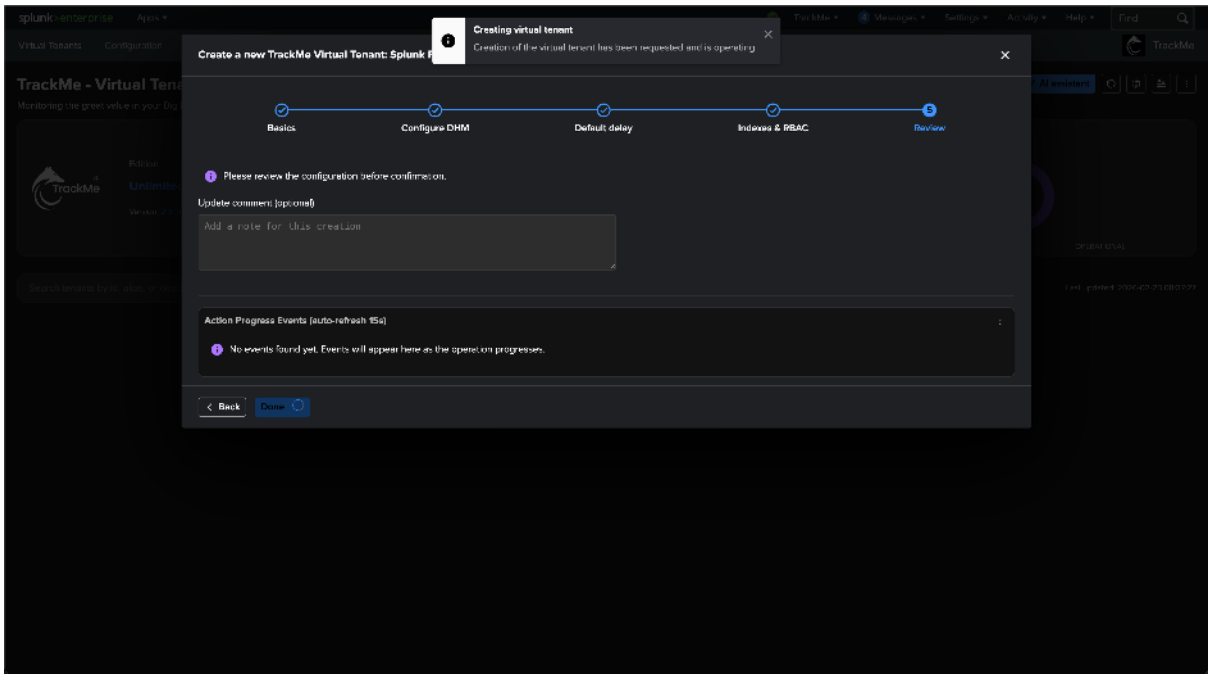


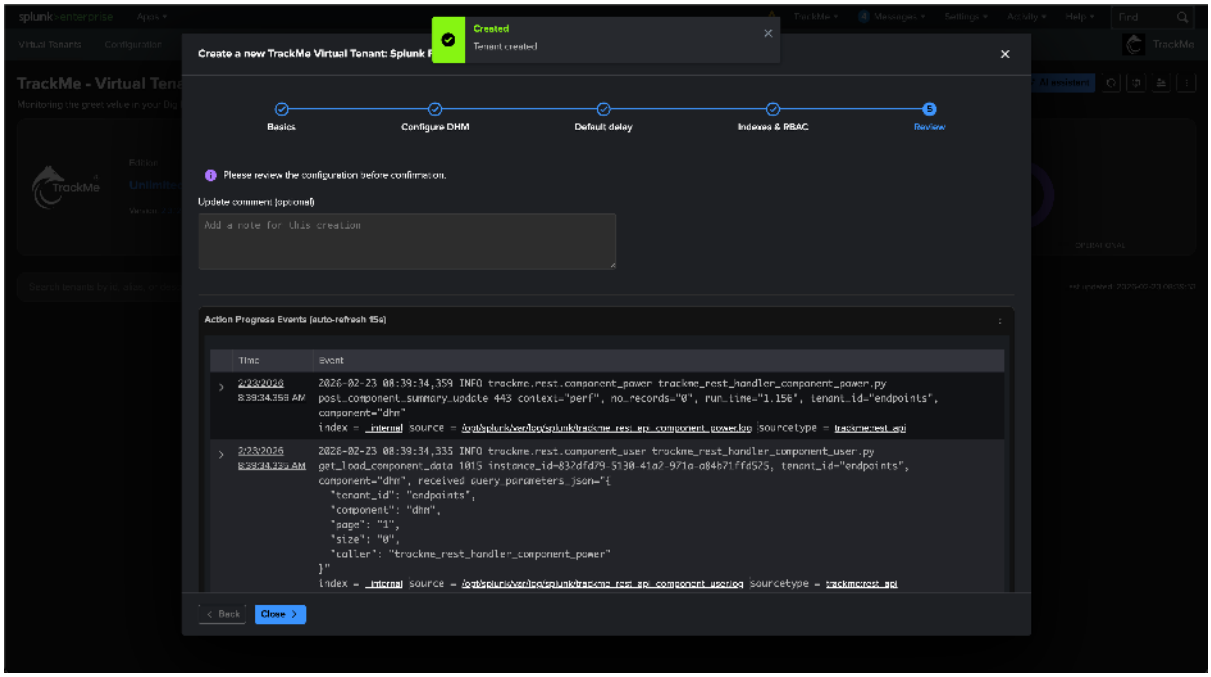


finalize the creation of the Virtual Tenant for data hosts tracking:

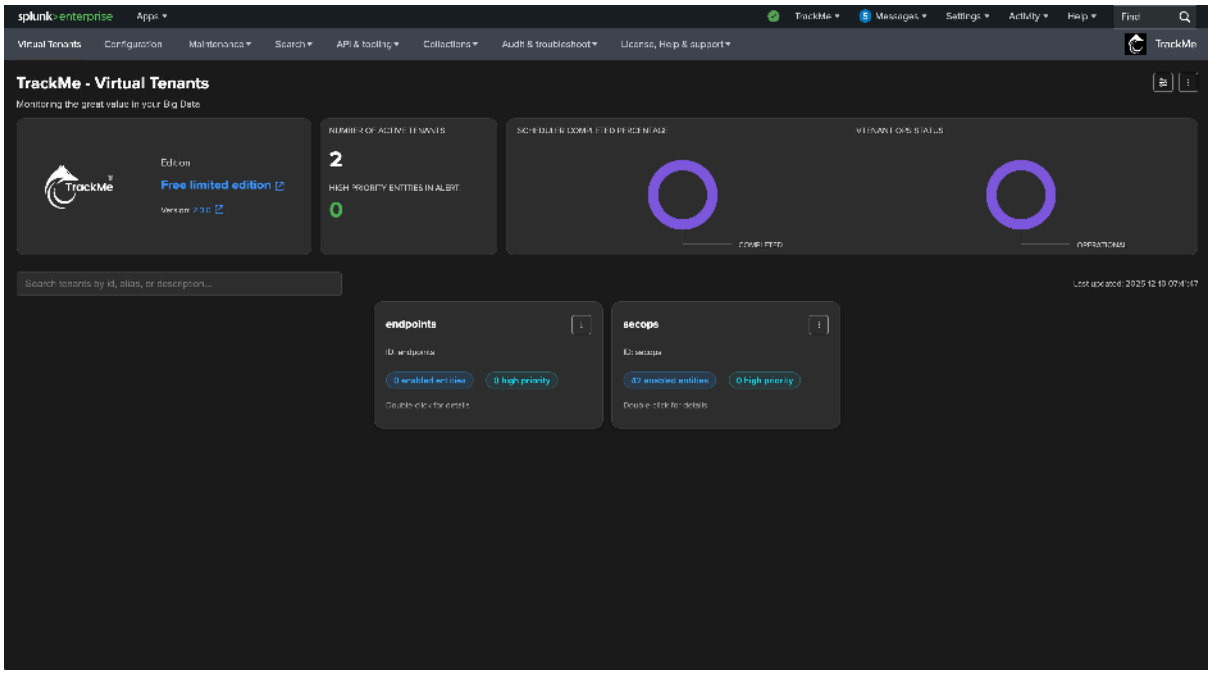


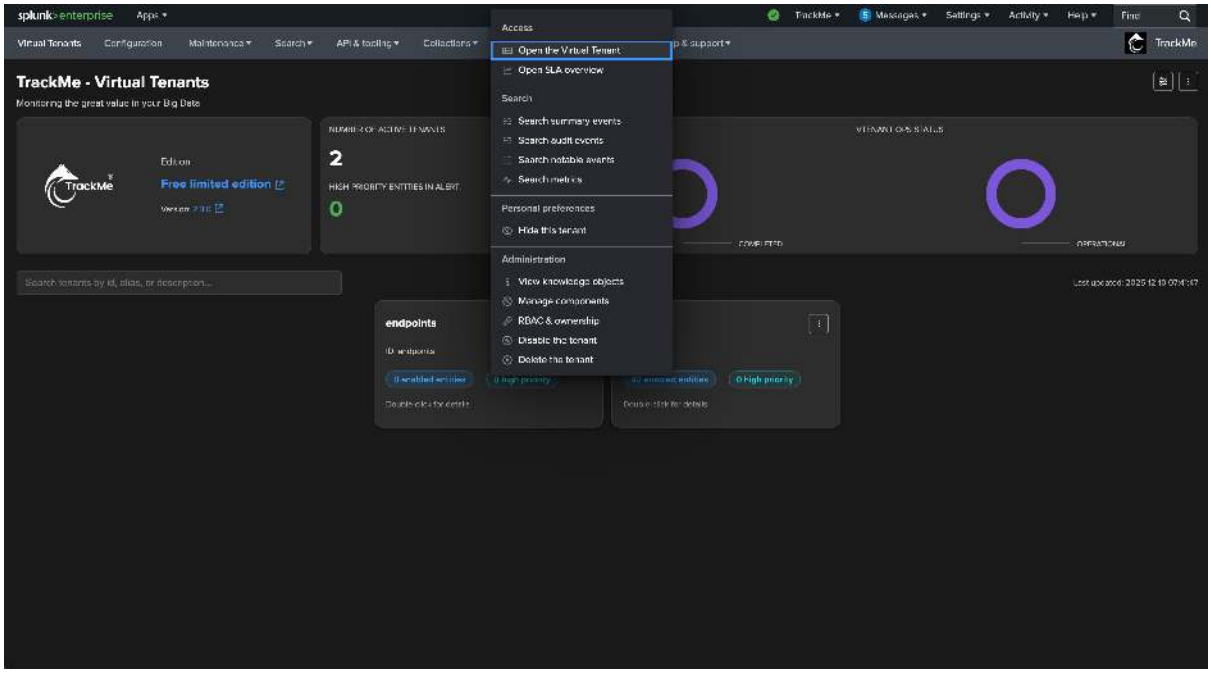

Create a new TrackMe Virtual Tenant: Splunk Feeds - DHM
Basics
Configure DHM
Default delay
Indexes & RBAC
Review
Please review the configuration before confirmation.
Update comment (optional)
Add a note for this creation
Back
Done

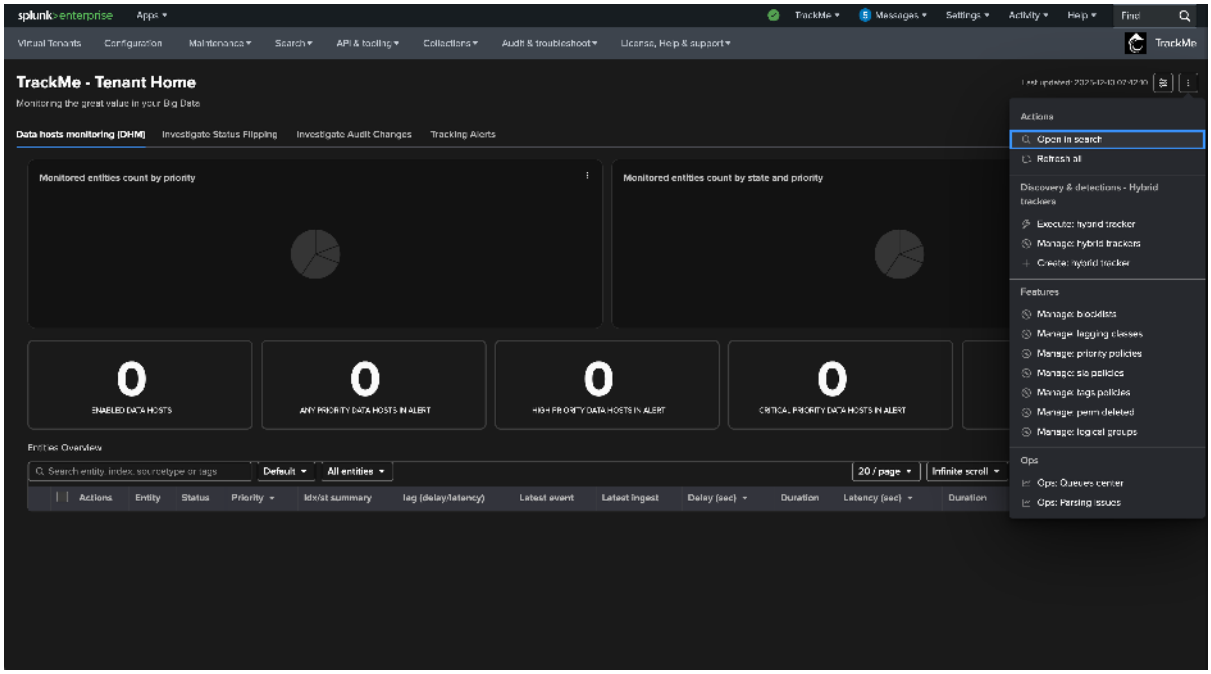

Creating virtual tenant
Create a new TrackMe Virtual Tenant: Splunk
Basics
Configure DHM
Default delay
Indexes & RBAC
Review
Please review the configuration before confirmation.
Update comment (optional)
Add a note for this creation
Action Progress Events (auto-refresh 15s)
No events found yet. Events will appear here as the operation progresses.
Back
Done

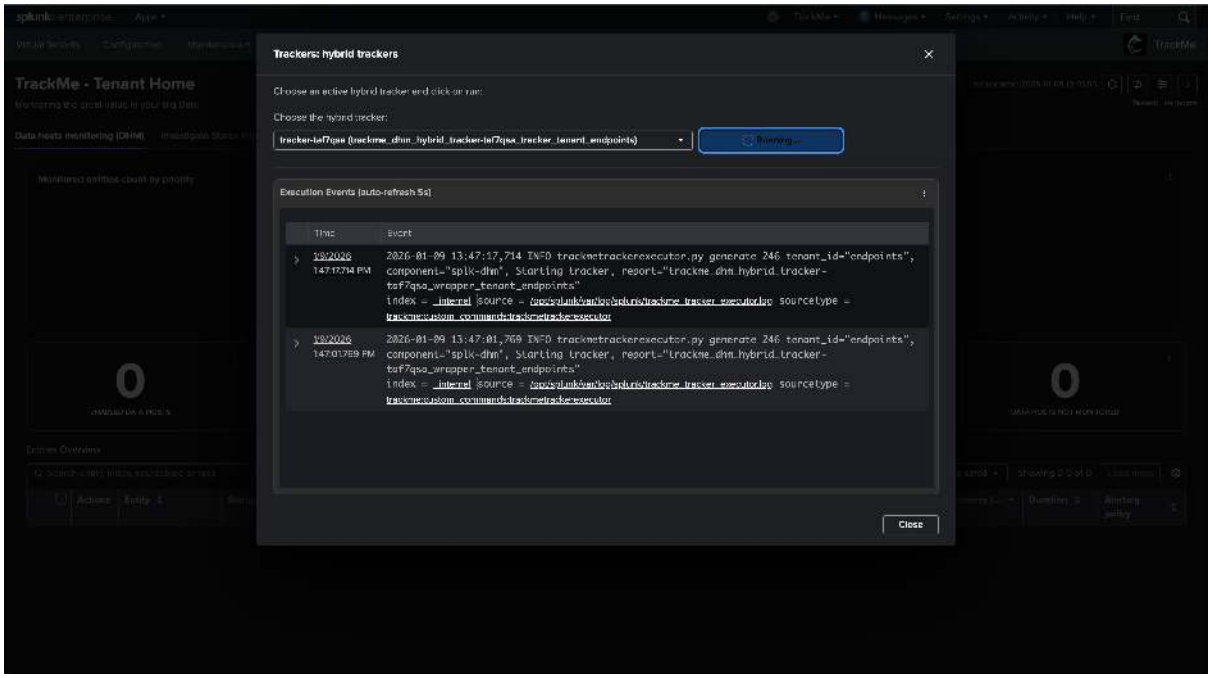


Execute tracker, access the tenant:




splunk>enterprise
Apps
TrackMe
Messages
Settings
Activity
Help
Find
Virtual Tenants
Configuration
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe - Virtual Tenants
Monitoring the great value in your Big Data
Edition
Free limited edition
2
0
Access
Open the Virtual Tenant
Open SLA overview
Search
Search summary events
Search audit events
Search notable events
Search metrics
Personal preferences
Hide this tenant
Administration
View knowledge objects
Manage components
RBAC & ownership
Disable the tenant
Delete the tenant
endpoints


splunk>enterprise
Apps
TrackMe
Messages
Settings
Activity
Help
Find
Virtual Tenants
Configuration
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe - Tenant Home
Monitoring the great value in your Big Data
Data hosts monitoring (DHM)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Monitored entities count by priority
Monitored entities count by state and priority
0
ENABLED DATA HOSTS
0
ANY PRIORITY DATA HOSTS IN ALERT
0
HIGH PRIORITY DATA HOSTS IN ALERT
0
CRITICAL PRIORITY DATA HOSTS IN ALERT
Entities Overview
Actions
Open in search
Refresh all
Discovery & detections - Hybrid trackers
Execute: hybrid tracker
Manage: hybrid trackers
Create: hybrid tracker
Features
Manage: blocklists
Manage: logging classes
Manage: priority policies
Manage: sla policies
Manage: tags policies
Manage: perm deleted
Manage: logical groups
Ops
Ops: Queues center
Ops: Parsing issues
Default
All entities
20 / page
Infinite scroll
Actions
Entity
Status
Priority
Idx/st summary
lag (delay/latency)
Latest event
Latest ingest
Delay (sec)
Duration
Latency (sec)
Duration


Trackers: hybrid trackers
Choose an active hybrid tracker and click on run:
Choose the hybrid tracker:
Execution Events (auto-refresh 5s)
Time
Event
Close

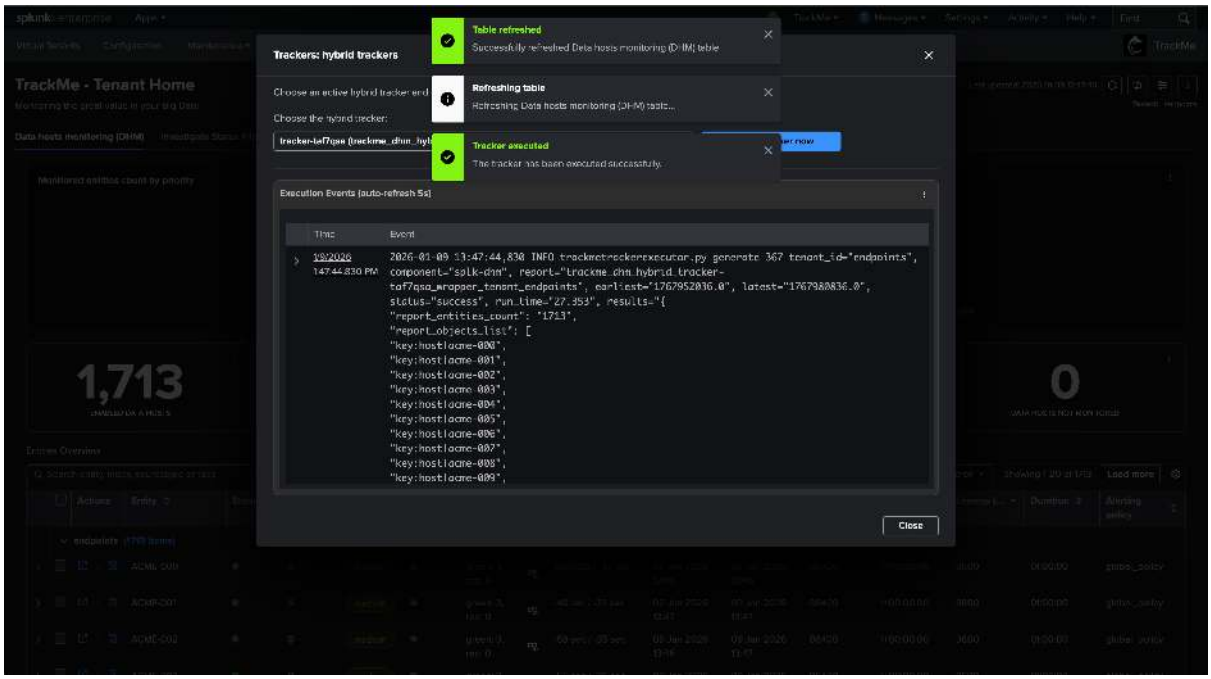
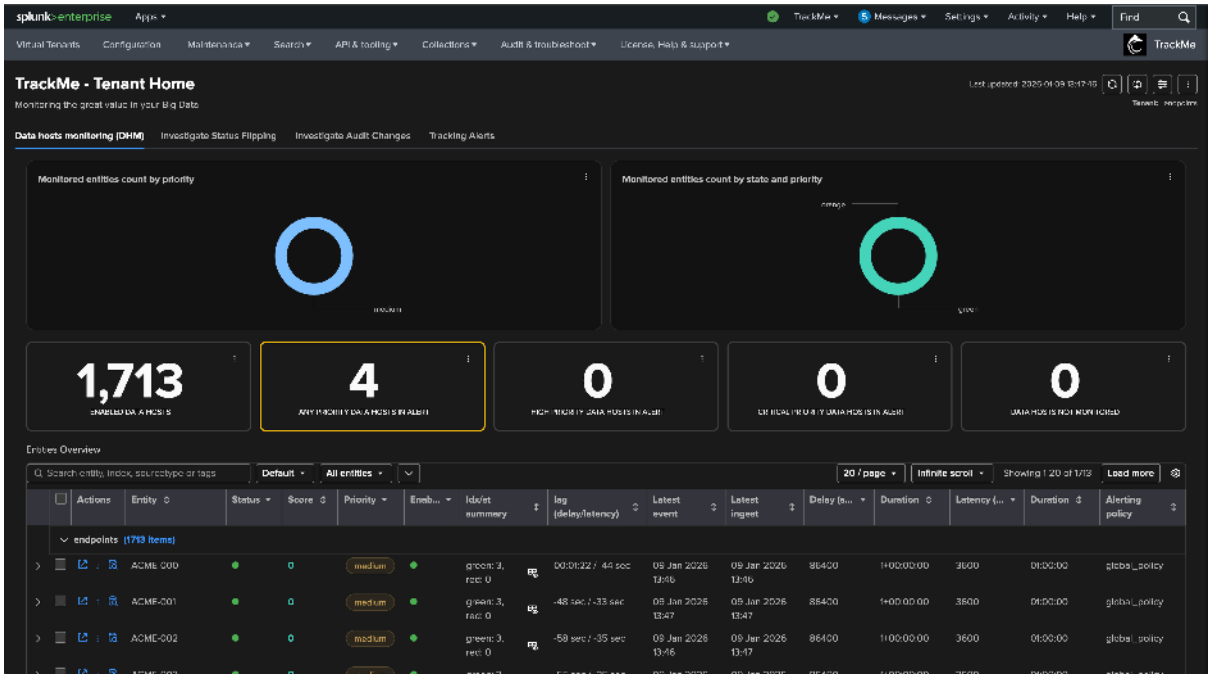


Table refreshed
Successfully refreshed Data hosts monitoring (DHM) table
Refreshing table
Refreshing Data hosts monitoring (DHM) table...
Tracker executed
The tracker has been executed successfully
Trackers: hybrid trackers
Execution Events (auto-refresh 5s)
Time
Event
Close



Step 3: Alerting and notifications

Hint

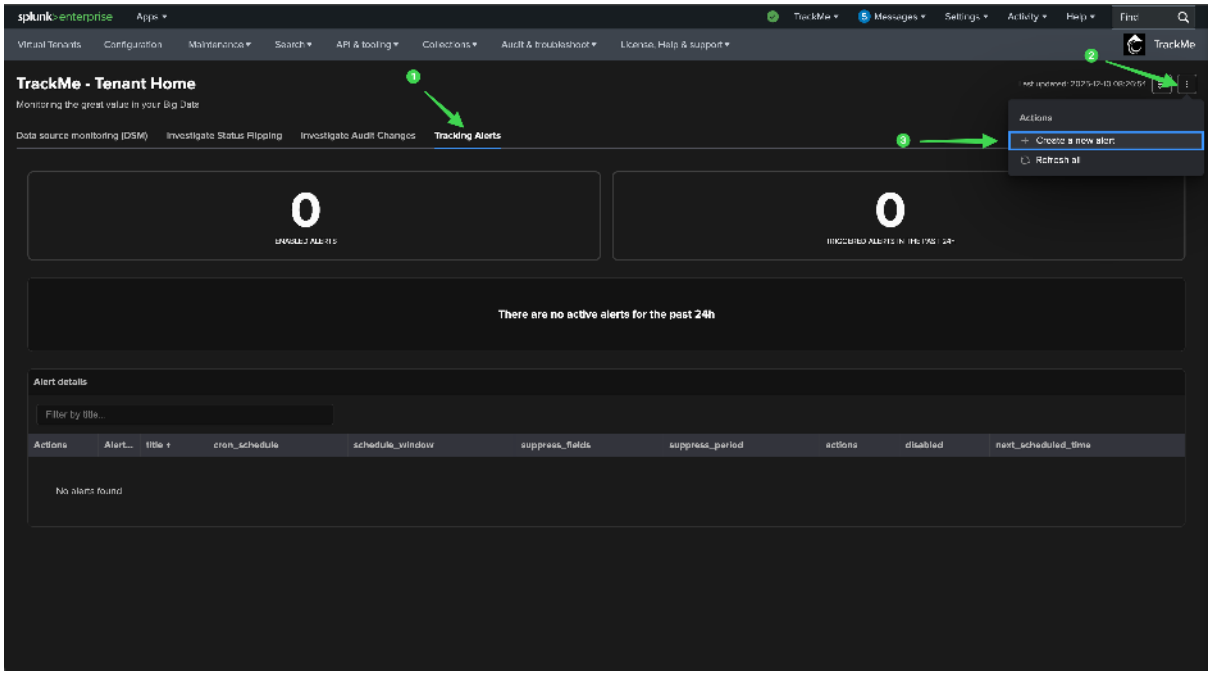
This quickstart was updated to use TrackMe's new StateFul Alerting features:

- Since TrackMe 2.1.11, a new concept of StateFul Alerting is available as the main alerting mechanism in TrackMe.
- These features provides a powerful, flexible and easy alerting capabilities as well as enhanced Email delivery notifications.
- For more information about this new concept, please refer to the *Stateful alerting* documentation.

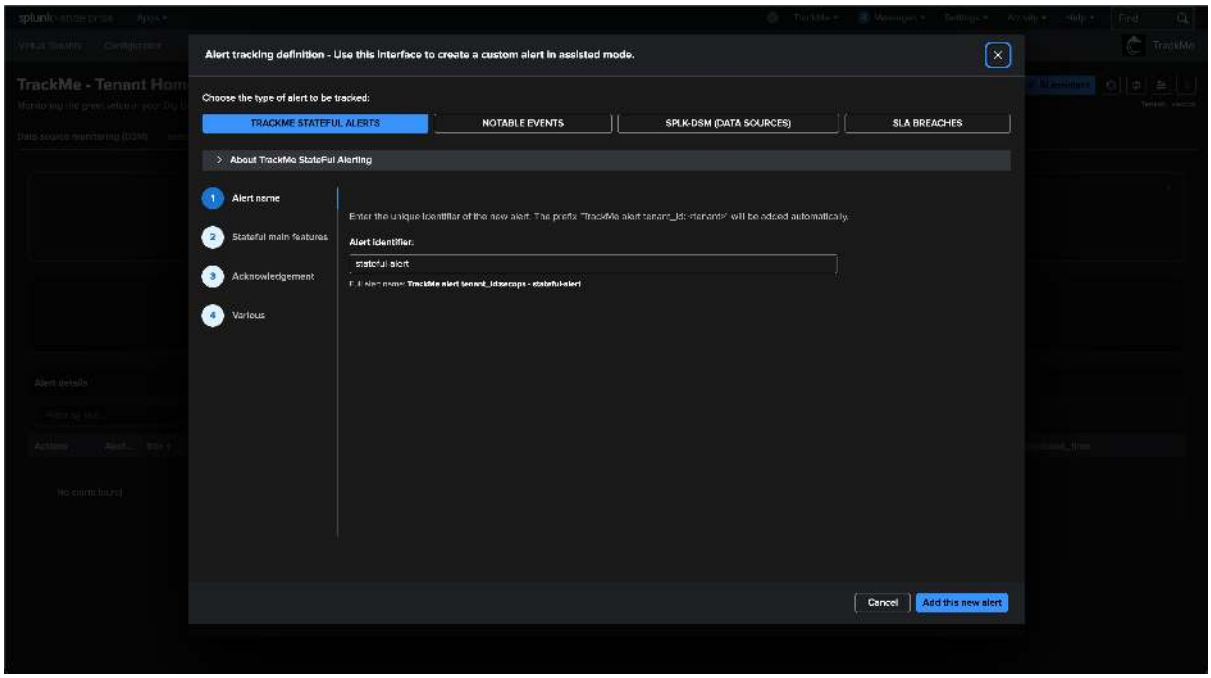
About Alerting recommendations:

- In a Production context, a true challenge is the massive number of entities, from hundreds if not not many thousdands of sources, hosts and so forth, it can be overwhelming to manage.
- We recommend a selective approach which used in conjonction with TrackMe and Splunk capabilities allow you to tackle these challenges.
- What we recommend is notably to leverage the `priority` concepts, and alert only on properly qualified entities leveraging `high` and `critical` priorities.
- This approach allows you to focus on the most important entities, and to avoid alerting fatigue by progressively qualifying your alerting logic and maturity in TrackMe over time.

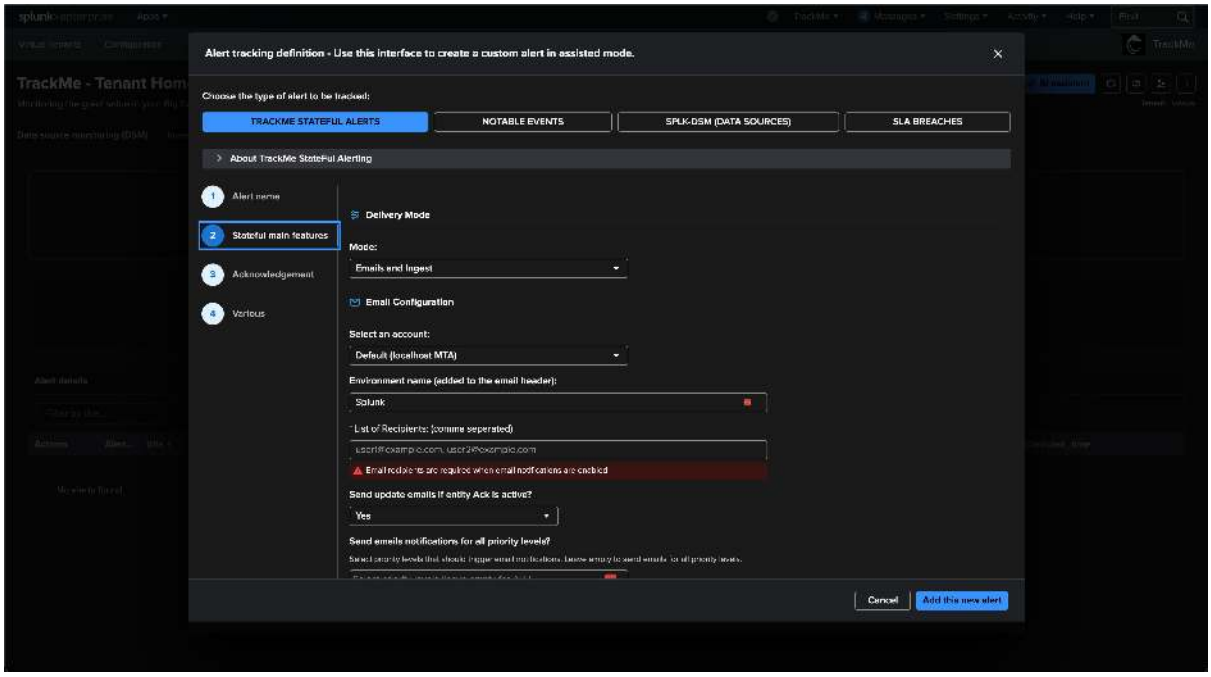
Access the Alerts creation wizard:



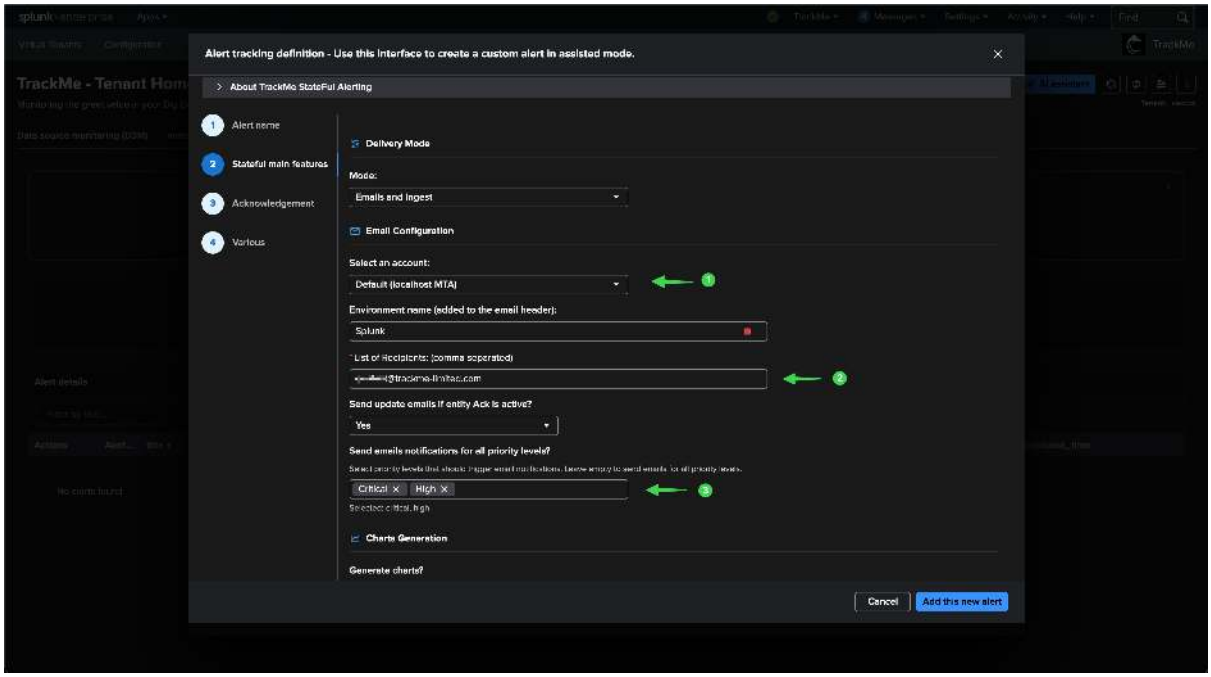
Welcoming screen for the alerts creation wizard:



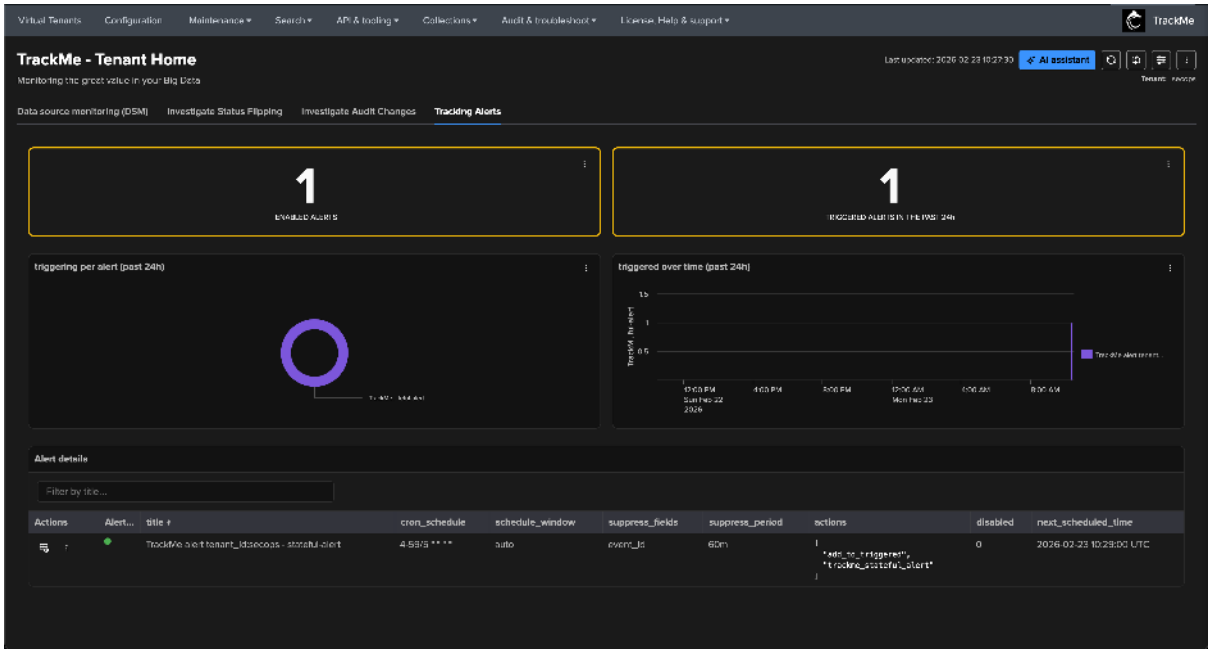
Go in step2, the default name of the alert is fine, we can leave it as is.



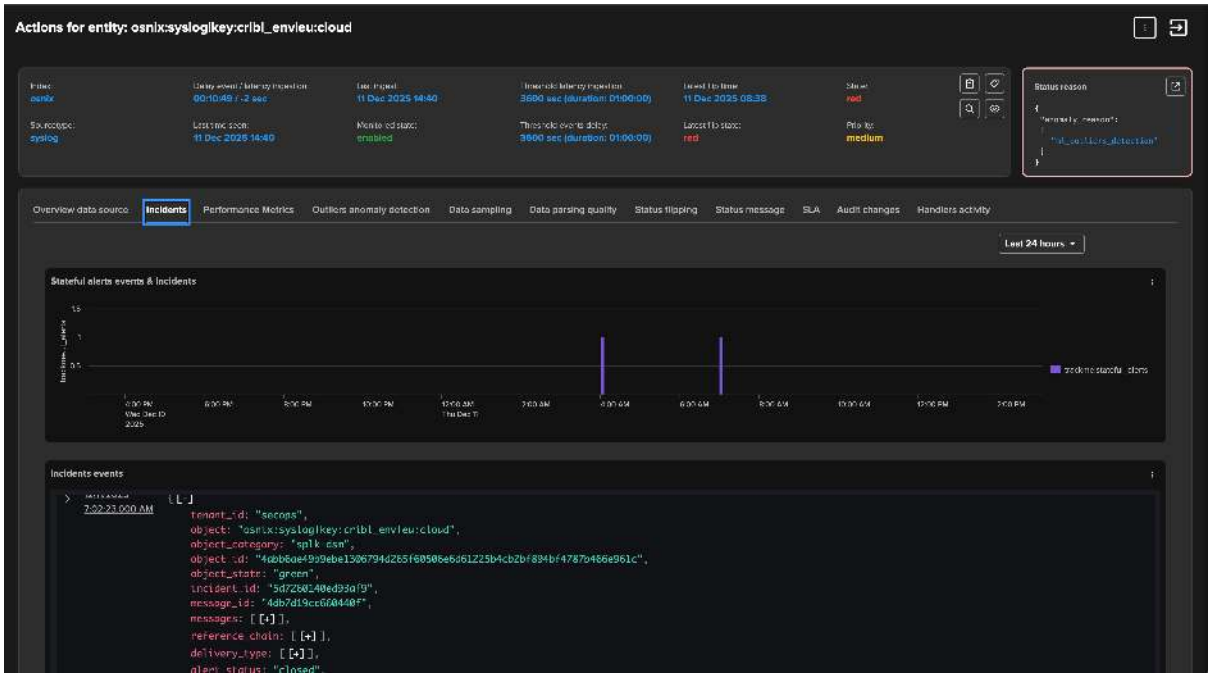
Configure the email delivery account, recipients and priority levels for email notifications:



Then validate the alerting creation:



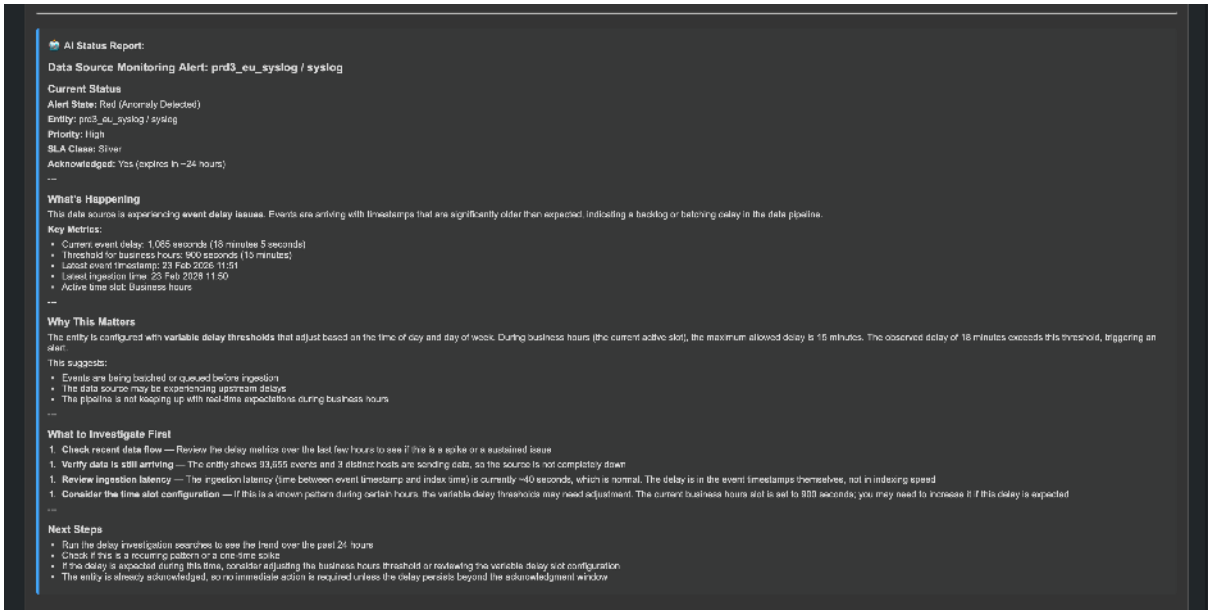
We are good to go now, if things are working as expected, any entity matching priority high or critical and currently in anomaly state, will trigger an alert!

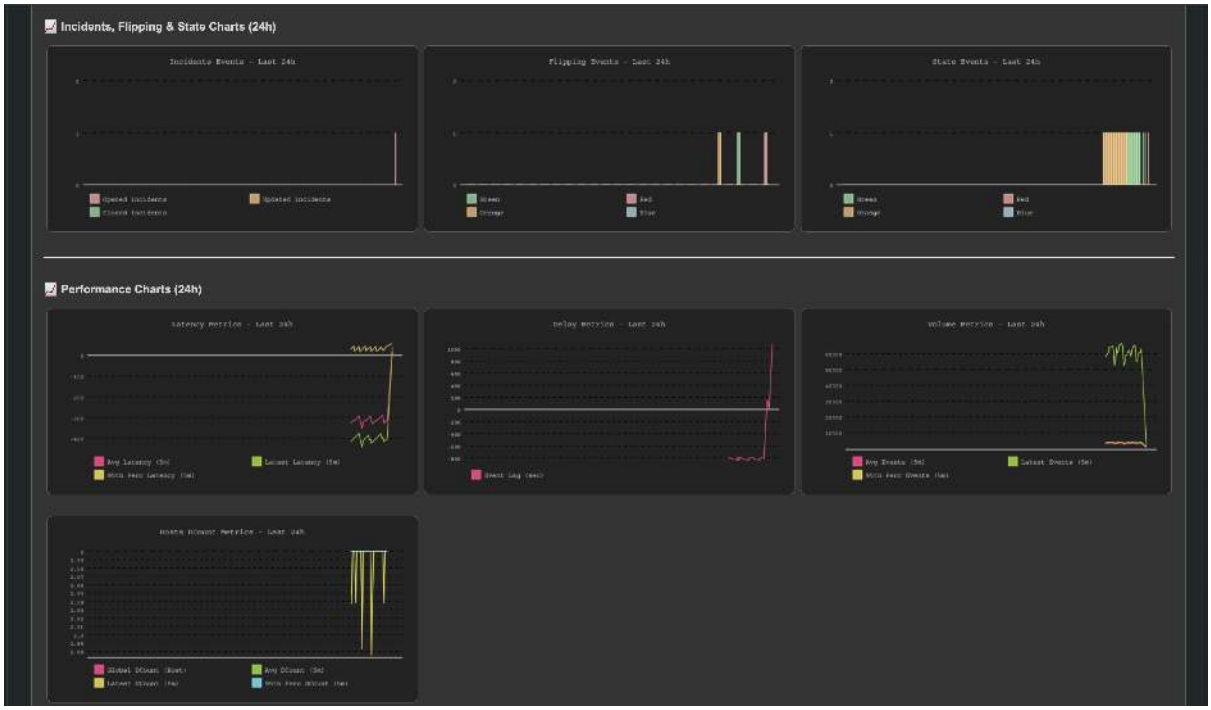


And if we did right, we also have a very nice Email notification!

Note: The AI status report requires an AI provider to be configured, if you haven't done so yet, please refer to the :ref:`ai_providers` documentation.


@trackme-limited.com
[TrackMe Incident-ID: 5839a93aef88608e] Entity prd3_eu_syslog:syslog is now in alert
To: @trackme-limited.com
TrackMe Notification
Environment: Splunk
Detection Time: Mon Feb 23 12:14:06 2026 UTC
Alert Status: opened
Tenant: secops
Alias: prd3_eu_syslog:syslog
Object: prd3_eu_syslog:syslog
Priority: high
Unknown Link
Status: red
Impact score: 100
Incident ID: 5839a93aef88608e
Detailed Information:
The entity has transitioned to an alerting state:
Alias: prd3_eu_syslog:syslog
Object: prd3_eu_syslog:syslog
Object ID: 2b5c4151e541a2be341a2a9c73e55462c549b7b4ef622b46ee8b1618beeda995
Category: splk-dsm
Anomaly Reason(s):
variable_delay_threshold_breached
Message(s):
Monitoring conditions are not met due to delay issues. Event delay is 1085.0 seconds (duration: 00:18:05), which is higher than the maximum allowed variable delay threshold (active slot: 'business_hours') of 900 seconds (duration: 00:15:00), latest event available (_time) for this entity: 23 Feb 2026 11:51, latest event ingested (_indextime) for this entity: 23 Feb 2026 11:50. This indicates that the source is receiving events with timestamps older than the threshold defined for this entity.
Message source: trackme:state
Message source ID: 8b6be13Bb2c9c7c656b970b5b00d0188a20bbcfcef1d68bf5278ad834b218982


AI Status Report:
Data Source Monitoring Alert: prd3_eu_syslog / syslog
Current Status
Alert State: Red (Anomaly Detected)
Entity: prd3_eu_syslog / syslog
Priority: High
SLA Class: Silver
Acknowledged: Yes (expires in ~24 hours)
What's Happening
This data source is experiencing event delay issues. Events are arriving with timestamps that are significantly older than expected, indicating a backlog or batching delay in the data pipeline.
Key Metrics:
Current event delay: 1,085 seconds (18 minutes 5 seconds)
Threshold for business hours: 900 seconds (15 minutes)
Latest event timestamp: 23 Feb 2026 11:51
Latest ingestion time: 23 Feb 2026 11:50
Active time slot: Business hours
Why This Matters
The entity is configured with variable delay thresholds that adjust based on the time of day and day of week. During business hours (the current active slot), the maximum allowed delay is 15 minutes. The observed delay of 18 minutes exceeds this threshold, triggering an alert.
This suggests:
Events are being batched or queued before ingestion
The data source may be experiencing upstream delays
The pipeline is not keeping up with real-time expectations during business hours
What to Investigate First
1. Check recent data flow — Review the delay metrics over the last few hours to see if this is a spike or a sustained issue
1. Verify data is still arriving — The entity shows 93,655 events and 3 distinct hosts are sending data, so the source is not completely down
1. Review ingestion latency — The ingestion latency (time between event timestamp and index time) is currently ~40 seconds, which is normal. The delay is in the event timestamps themselves, not in indexing speed
1. Consider the time slot configuration — If this is a known pattern during certain hours, the variable delay thresholds may need adjustment. The current business hours slot is set to 900 seconds; you may need to increase it if this delay is expected
Next Steps
Run the delay investigation searches to see the trend over the past 24 hours
Check if this is a recurring pattern or a one-time spike
If the delay is expected during this time, consider adjusting the business hours threshold or reviewing the variable delay slot configuration
The entity is already acknowledged, so no immediate action is required unless the delay persists beyond the acknowledgment window



Adapt to your preferences, repeat as needed, TrackMe is now Production ready!

QUICK START - TrackMe Impact-Based Alerting (IBA)

Impact-Based Alerting (IBA)

- Impact-Based Alerting (IBA) is a new core alerting model introduced in TrackMe 2.3.5.
- IBA transforms how TrackMe determines entity status by using a flexible, configurable scoring system instead of hardcoded rules.
- Inspired by Risk-Based Analysis (RBA) in security contexts, IBA allows administrators to fine-tune how different anomalies contribute to entity health status.
- IBA provides complete transparency through detailed score breakdowns, making it easy to understand why an entity is in a particular state.
- For users migrating from previous versions, the transition is seamless and requires no manual intervention.
- This guide focuses on feeds tracking (DSM and DHM components) as the primary use case, with extensions to other TrackMe components covered in later sections.

What is Impact-Based Alerting?

Impact-Based Alerting (IBA) is TrackMe's intelligent scoring system that determines entity health status based on a cumulative impact score. Instead of using fixed rules, IBA aggregates scores from various anomaly types over a 24-hour rolling window, providing a flexible and tunable approach to alerting.

Domain Positioning

- Inspired by security-grade risk scoring, IBA brings impact-driven alerting to data monitoring. IBA applies risk-based scoring principles to data reliability and observability in TrackMe.

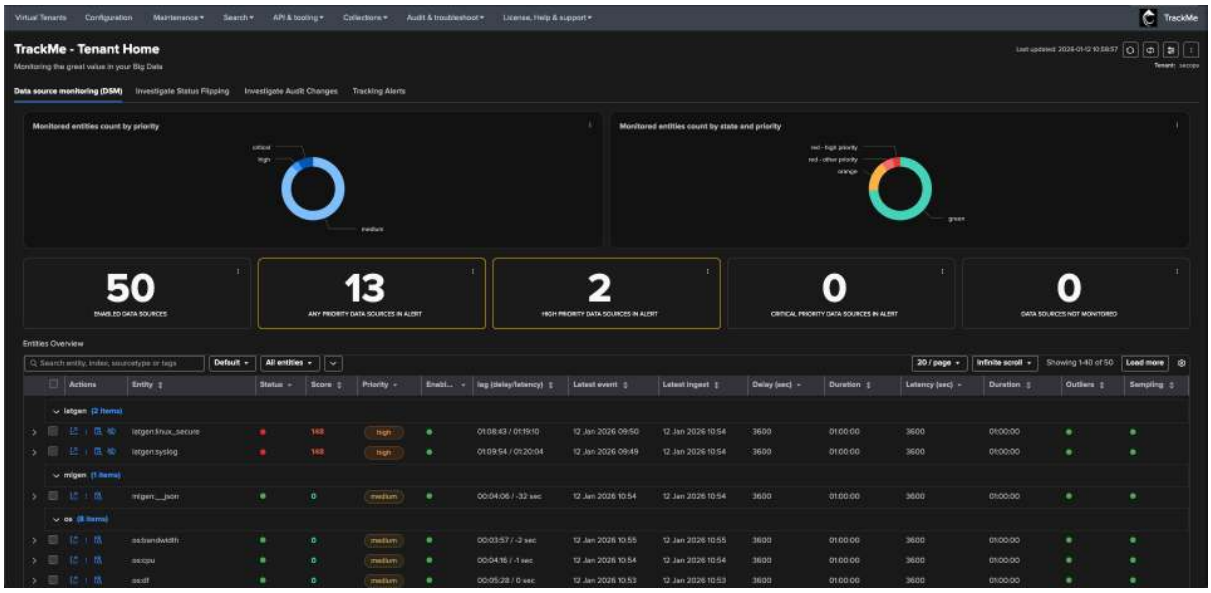
Key Concepts:

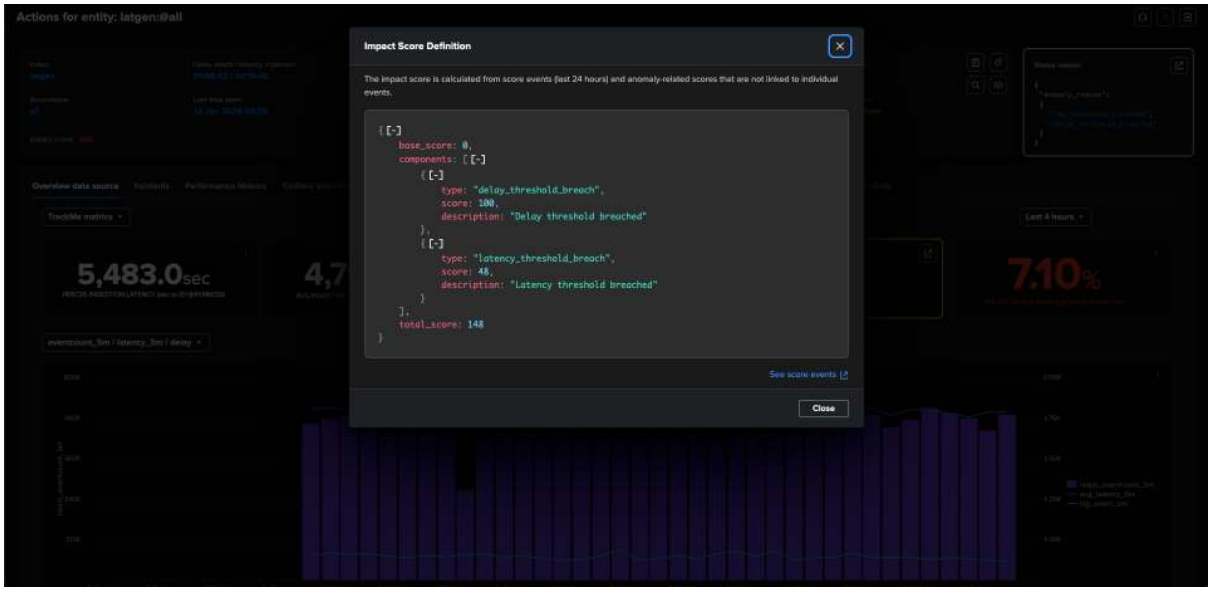
- **Impact Score**: A numeric value (0-100) assigned to each anomaly type when detected

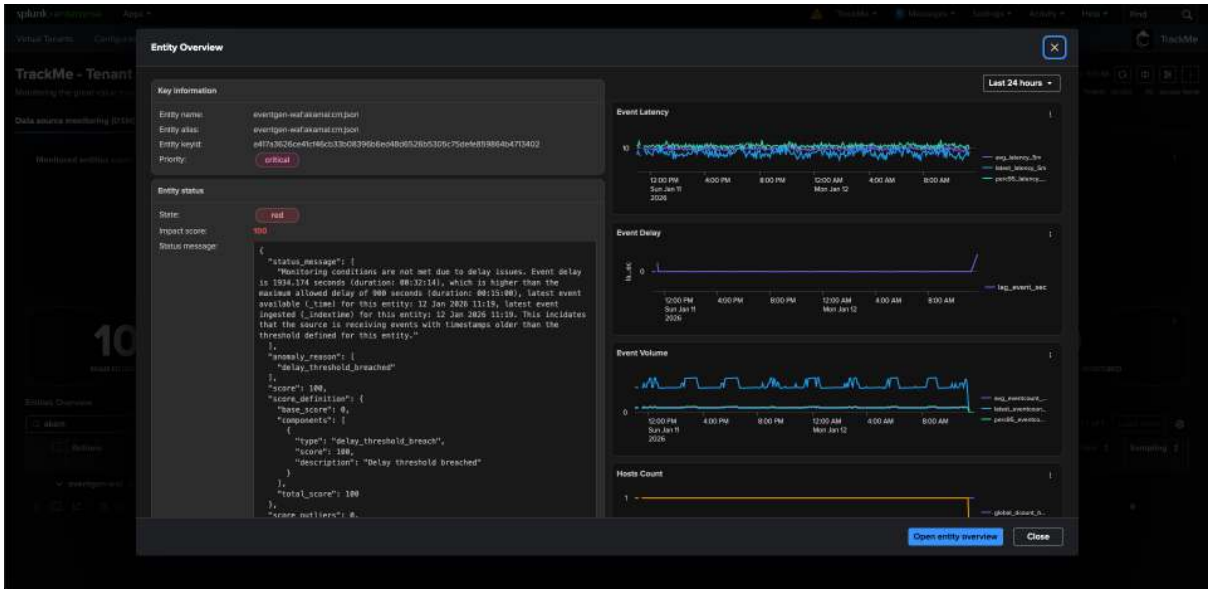
- **Total Score**: The cumulative sum of all impact scores for an entity
- **Score Aggregation**: Scores are aggregated over a 24-hour rolling window
- **Status Determination**: Entity status (green, orange, red, blue) is determined by the total score
- **Score Transparency**: Every score includes a detailed breakdown showing exactly which anomalies contributed

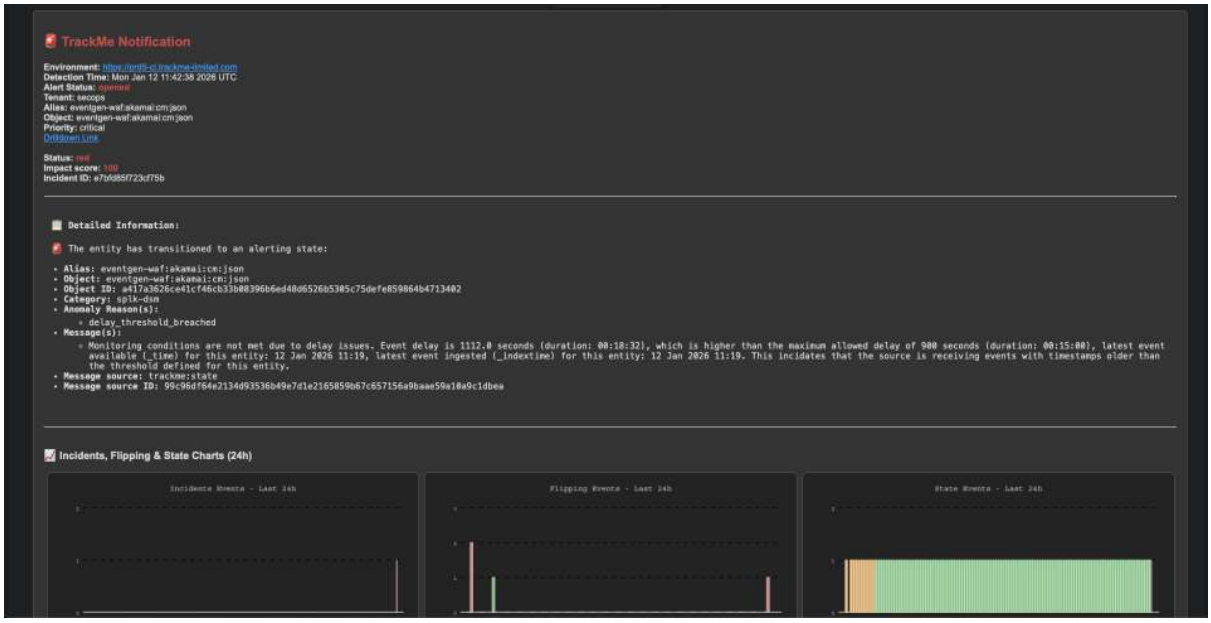
Why IBA Matters:

- **Smarter Alerting**: Fine-tune impact scores per tenant, component, and anomaly type to match your environment's priorities
- **Reduced False Positives**: One-click suppression capabilities allow you to mark outliers or entire entities as false positives
- **Complete Transparency**: Detailed score breakdowns show exactly why an entity is in a particular state
- **Flexible Configuration**: Adjust impact scores through an intuitive UI without code changes
- **Audit Trail**: Full traceability of score events and anomaly reasons, even when suppressed









Understanding Entity Status in IBA

TrackMe entities can have four possible statuses, each determined by the total impact score:

Green Status

- **Condition**: Total score = 0 (no anomalies detected)
- **Meaning**: The entity is healthy and operating normally
- **Visual Indicator**: Green status dot

Orange Status

- **Condition**: Total score > 0 and < 100 (anomalies present but below critical threshold)
- **Meaning**: Potential issues detected that require attention but don't warrant a critical alert
- **Visual Indicator**: Orange status dot
- **Use Case**: Early warning for issues that may escalate if not addressed

Red Status

- **Condition**: Total score >= 100 (critical threshold breached)
- **Meaning**: Critical issues detected that require immediate attention
- **Visual Indicator**: Red status dot
- **Use Case**: Triggers alerts and notifications for immediate action

Blue Status

- **Condition**: Special state (e.g., logical group protection, minimal disruption time)
- **Meaning**: Entity is part of a logical group that provides protection
- **Visual Indicator**: Blue status dot

How Impact Scoring Works for Feeds Tracking (DSM)

The Data Sources Monitoring (DSM) component tracks data sources (index/sourcetype combinations) and uses IBA to determine entity status based on various anomaly types.

DSM Anomaly Types and Default Impact Scores:

Anomaly Type	Default Score	Description
Delay Threshold Breach	100	Data flow is interrupted - no new events received (static delay policy)
Variable Delay Threshold Breach	100	Data flow is interrupted - no new events received within the active variable delay slot threshold (variable delay policy, since v2.3.12)
Latency Threshold Breach	48	Events are being indexed but with high latency
Data Sampling Anomaly	36	Quality issues detected in event format recognition
Min Hosts Dcount Breach	100	Minimum number of distinct hosts threshold breached
Future Tolerance Breach	36	Events detected with timestamps too far in the future
ML Outliers Detection	36 (default)	Machine Learning models detected abnormal volume patterns

Score Calculation Example:

Let's consider a DSM entity with multiple anomalies:

1. **Base Score**: 0 (no upstream threshold breaches from hybrid trackers)
2. **Delay Threshold Breach** (or **Variable Delay Threshold Breach**): Score = 100 (default)
3. **ML Outliers Detection**: Score = 36 (default)
4. **Total Score**: 136

Result: Red status (score >= 100)

Score Definition Breakdown:

The system maintains a detailed breakdown of the score:

```
{
  "base_score": 0,
  "components": [
    {
      "type": "delay_threshold_breach",
      "score": 100,
      "description": "Delay threshold breached"
    },
    {
      "type": "ml_outliers_detection",
```

(continues on next page)

(continued from previous page)

```
      "score": 36,
      "description": "ML outliers detected"
    }
  ],
  "score_outliers": 36.0,
  "score_source": ["lowerbound_outlier"],
  "total_score": 136.0
}
```

Understanding the Score Components:

- **base_score**: Score from upstream tracker/search results (e.g., threshold breaches detected by hybrid trackers)
- **components**: Array of anomaly components contributing to the score, each with: - **type**: The anomaly type (e.g., "delay_threshold_breach") - **score**: The impact score for this anomaly - **description**: Human-readable description
- **score_outliers**: Cumulative outlier score (if ML outliers are detected)
- **score_source**: Array of score sources (e.g., "lowerbound_outlier", "false_positive", "manual_score")
- **total_score**: Final calculated score used for status determination

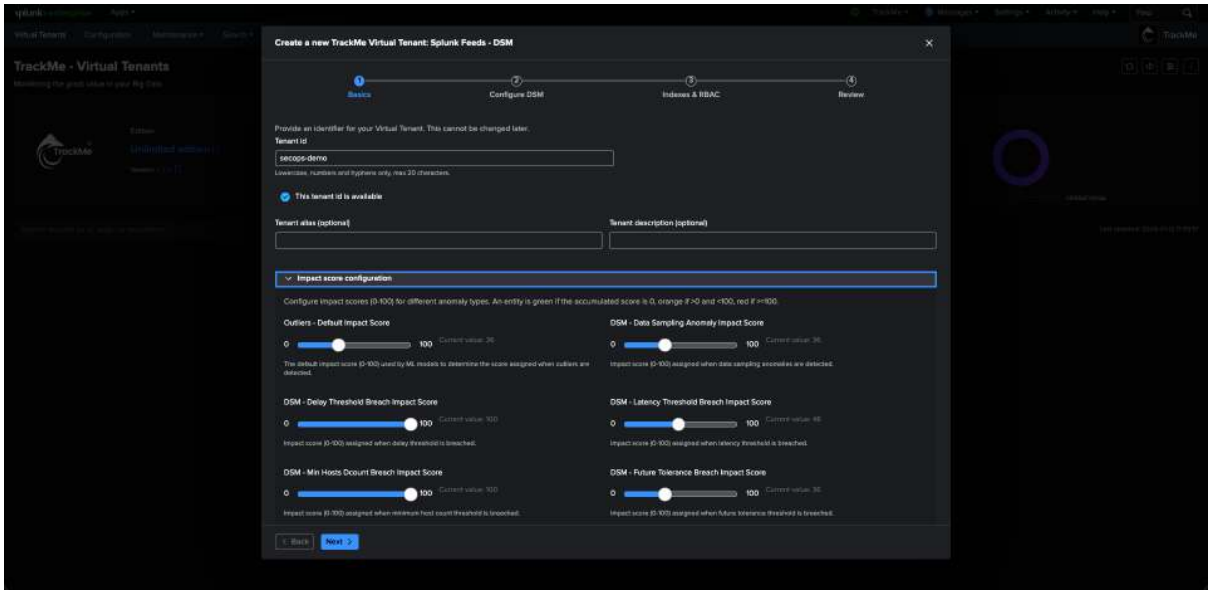
Configuring Impact Scores for DSM

Impact scores are configured per Virtual Tenant account, allowing you to customize how different anomaly types contribute to entity status for each tenant.

During Virtual Tenant Creation:

When creating a new Virtual Tenant for DSM (Data Sources Monitoring), you can configure impact scores in the wizard:

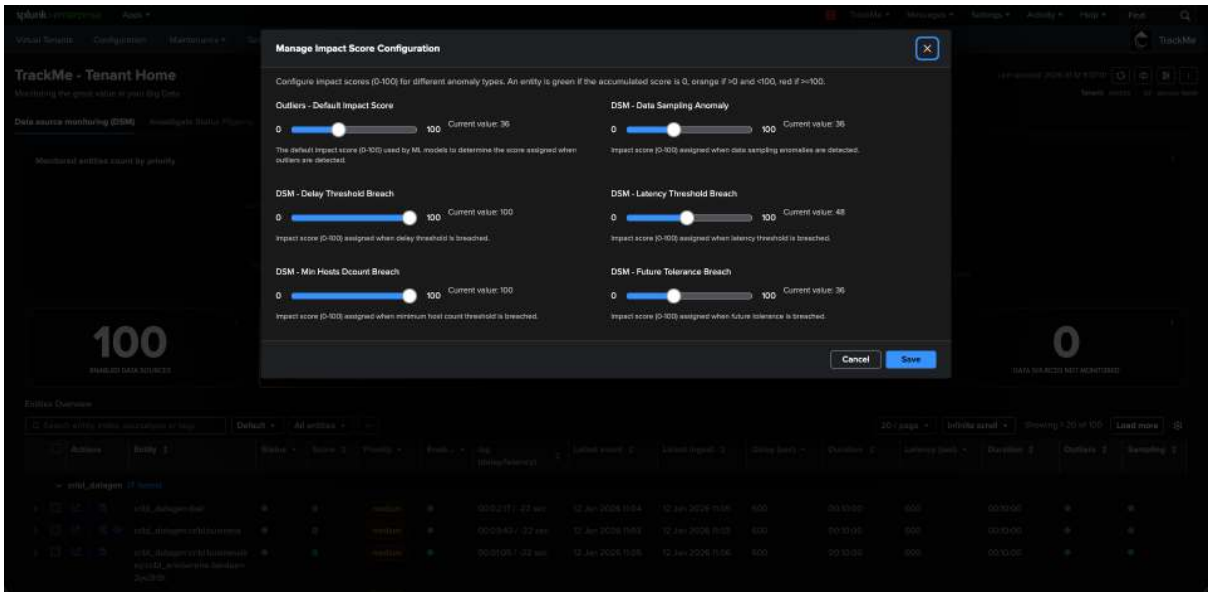
1. Navigate to the "Impact score configuration" section (collapsible panel)
2. Adjust sliders (0-100) for each anomaly type:
   - **Outliers - Default Impact Score**: Applies to ML outlier detections (default: 36)
   - **DSM - Data Sampling Anomaly Impact Score**: Quality issues (default: 36)
   - **DSM - Delay Threshold Breach Impact Score**: Data flow interruption (default: 100) - applies to both static and variable delay policies
   - **DSM - Latency Threshold Breach Impact Score**: High indexing latency (default: 48)
   - **DSM - Min Hosts Dcount Breach Impact Score**: Minimum hosts threshold (default: 100)
   - **DSM - Future Tolerance Breach Impact Score**: Future timestamp issues (default: 36)



After Virtual Tenant Creation:

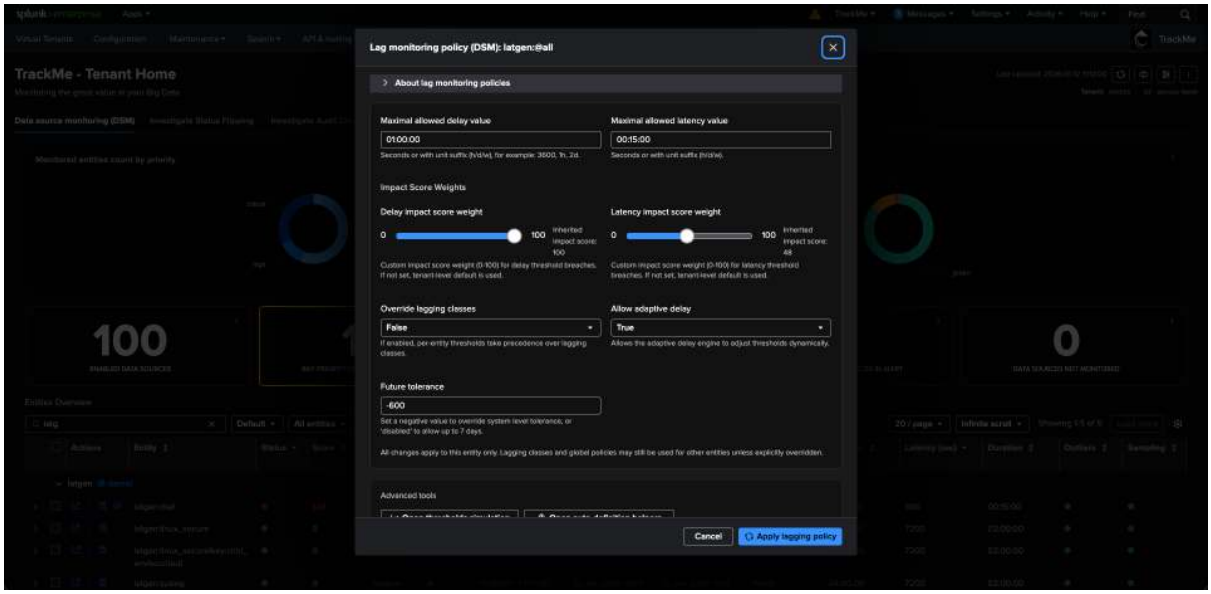
You can update impact scores for existing tenants:

1. Navigate to the Virtual Tenant home interface
2. Click the kebab menu (three dots) for the tenant
3. Select “Manage: Impact score”
4. Adjust the sliders for each anomaly type
5. Save the configuration



Per-Entity Score Weight Overrides (DSM and DHM):

For DSM and DHM components, you can override impact score weights on a per-entity basis through the lagging policy configuration. This allows fine-grained control for specific entities that may have different priorities or requirements than the tenant-level defaults.



How to Configure Per-Entity Score Weights:

1. Navigate to the entity detail modal for the DSM or DHM entity you want to configure
2. Click on "Manage lagging policy" (or access via the entity kebab menu)
3. In the lagging policy modal, locate the "Impact Score Weights" section
4. Adjust the sliders for: - **Delay impact score weight**: Custom impact score (0-100) for delay threshold breaches - **Latency impact score weight**: Custom impact score (0-100) for latency threshold breaches
5. The modal displays the "Inherited impact score" value, showing the tenant-level default
6. Click "Apply lagging policy" to save the entity-specific overrides

Important Notes:

- Per-entity score weights override tenant-level defaults for that specific entity only
- Other entities continue to use tenant-level defaults unless they also have per-entity overrides
- Per-entity overrides apply only to delay and latency threshold breaches (not other anomaly types)
- These overrides are stored in the entity's lagging policy configuration
- Other TrackMe components (MHM, FLX, FQM, WLK) use different mechanisms for per-entity customization

Use Cases for Per-Entity Overrides:

- **Critical Data Sources**: Increase delay/latency scores for mission-critical feeds
- **Low-Priority Sources**: Decrease scores for less critical feeds to reduce alert noise
- **Special Handling**: Adjust scores for entities with known patterns or maintenance windows
- **Gradual Rollout**: Test new score configurations on specific entities before applying tenant-wide

Best Practices for DSM Impact Score Configuration:

- **Delay Threshold Breach** (and **Variable Delay Threshold Breach**): Keep at 100 (default) - this is a critical indicator of data flow interruption
- **Latency Threshold Breach**: Consider 48-60 range - high latency is important but less critical than delay
- **Data Sampling Anomaly**: Adjust based on your quality requirements (default: 36)
- **ML Outliers**: Start with default (36) and adjust based on false positive rates
- **Future Tolerance Breach**: Keep at default (36) unless you have specific requirements

Example Configuration Scenarios:

Scenario 1: High-Volume, Low-Latency Tolerance Environment

- Delay Threshold Breach: 100 (critical)
- Latency Threshold Breach: 60 (increased importance)
- Data Sampling Anomaly: 50 (quality is important)
- ML Outliers: 30 (reduce false positives)

Scenario 2: Quality-Focused Environment

- Delay Threshold Breach: 100 (critical)
- Latency Threshold Breach: 40 (less critical)
- Data Sampling Anomaly: 60 (quality is paramount)
- ML Outliers: 40 (balance detection and false positives)

Understanding Score Aggregation and Time Windows

Impact scores are aggregated over a **24-hour rolling window**, meaning:

- Scores from anomalies detected in the last 24 hours contribute to the total score
- Older scores automatically expire after 24 hours
- The total score reflects the current state of the entity over this window

How Score Events Work:

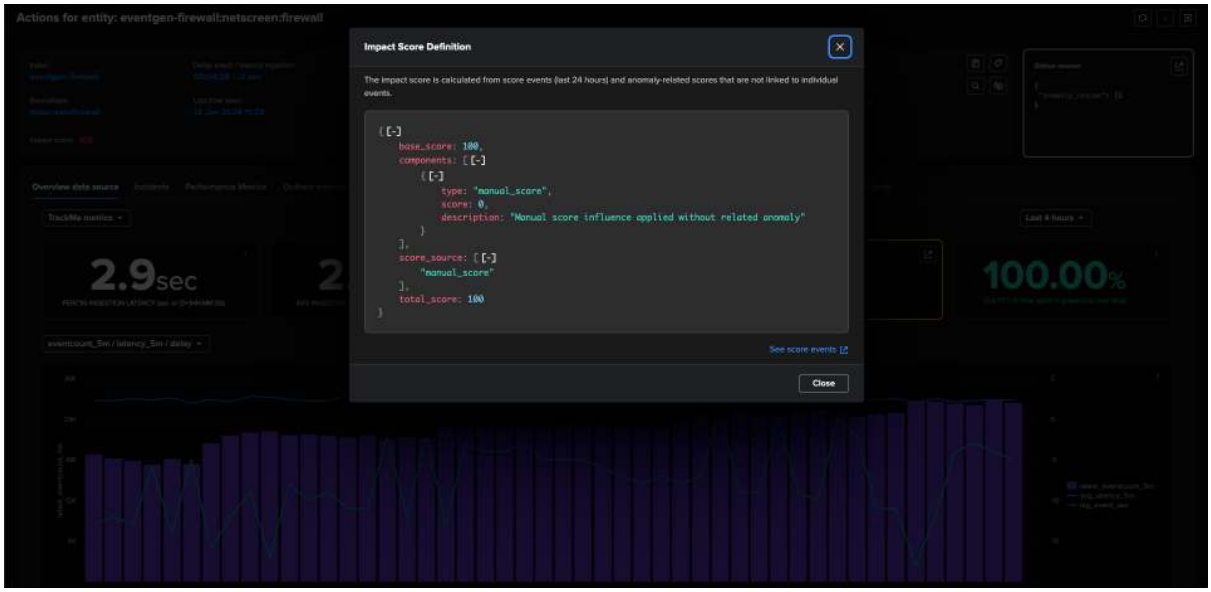
Score events are generated when anomalies are detected and stored in TrackMe's metrics store. These events include:

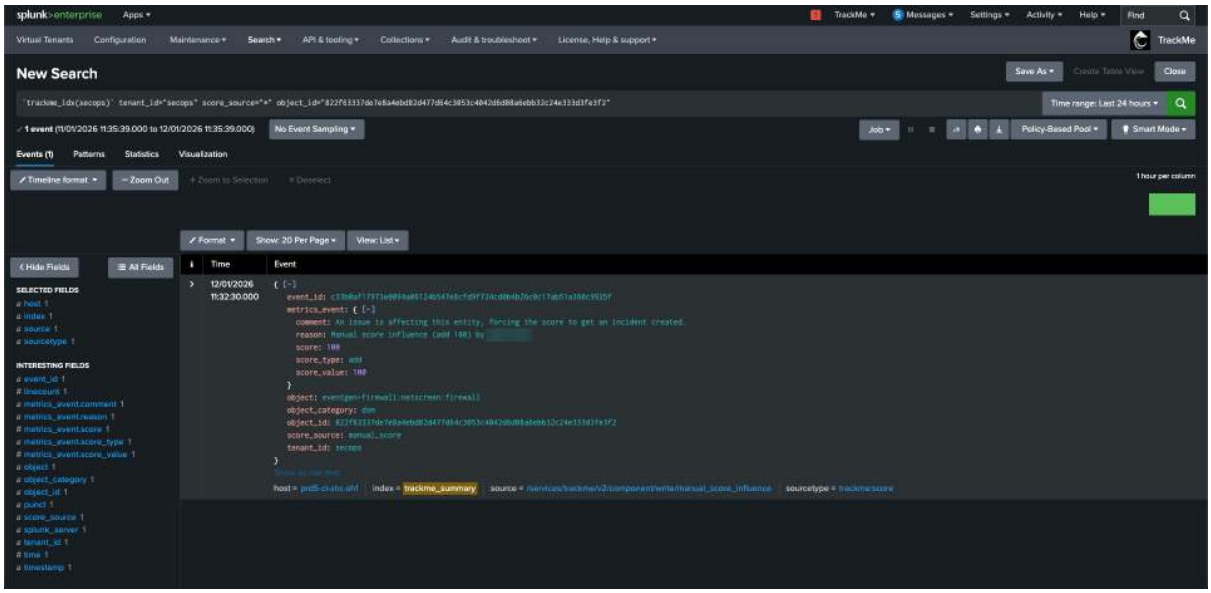
- **Timestamp**: When the anomaly was detected
- **Score Value**: The impact score for this anomaly
- **Score Source**: The source of the score (e.g., "delay_threshold_breach", "ml_outliers_detection")
- **Anomaly Details**: Additional context about the anomaly

Viewing Score Events:

You can view the underlying score events for any entity:

1. Click on the impact score value in the entity detail modal or overview table
2. This opens the Score Definition Modal showing the breakdown
3. Click "See score events" link at the bottom
4. This opens a Splunk search showing all score events for the entity in the last 24 hours





Score Event Query Example:

The score events search query looks like:

```
`trackme_idx(tenant_id)` tenant_id="your_tenant_id" score_source="*"␣
↪object_id="your_object_id"
```

The query filters score events for the specific entity and shows: - When each score event was generated - The score value (positive or negative) - The score source - Additional metadata

False Positive Management

IBA provides powerful false positive management capabilities, allowing you to suppress alerts while maintaining full audit trail transparency.

Global False Positive (Entity-Level):

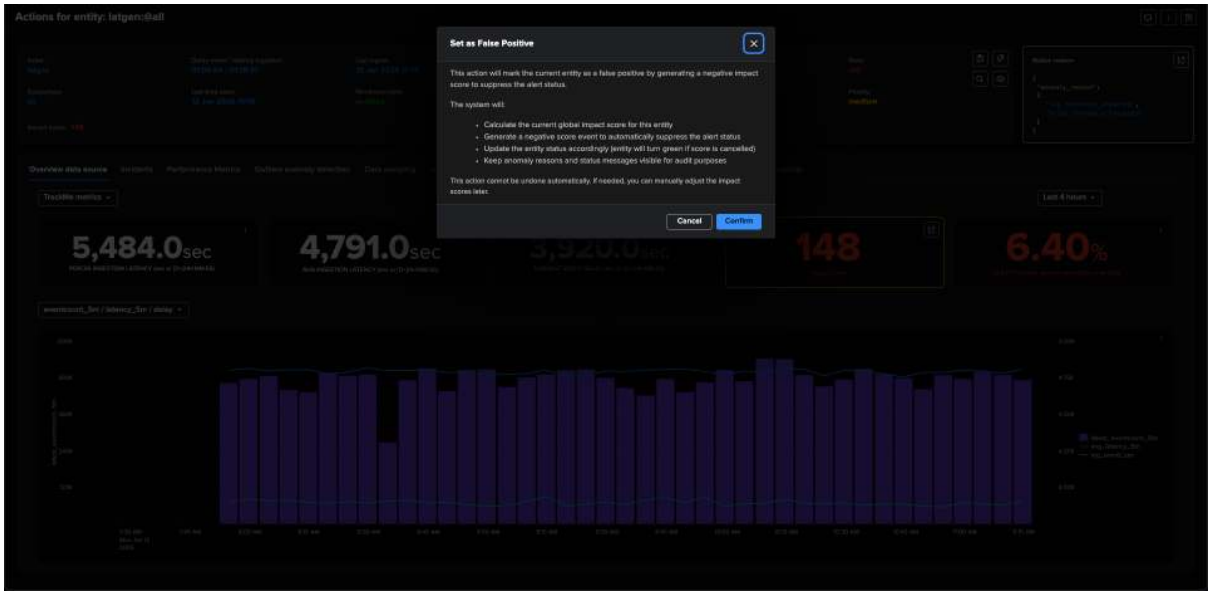
To suppress all alerts for an entity:

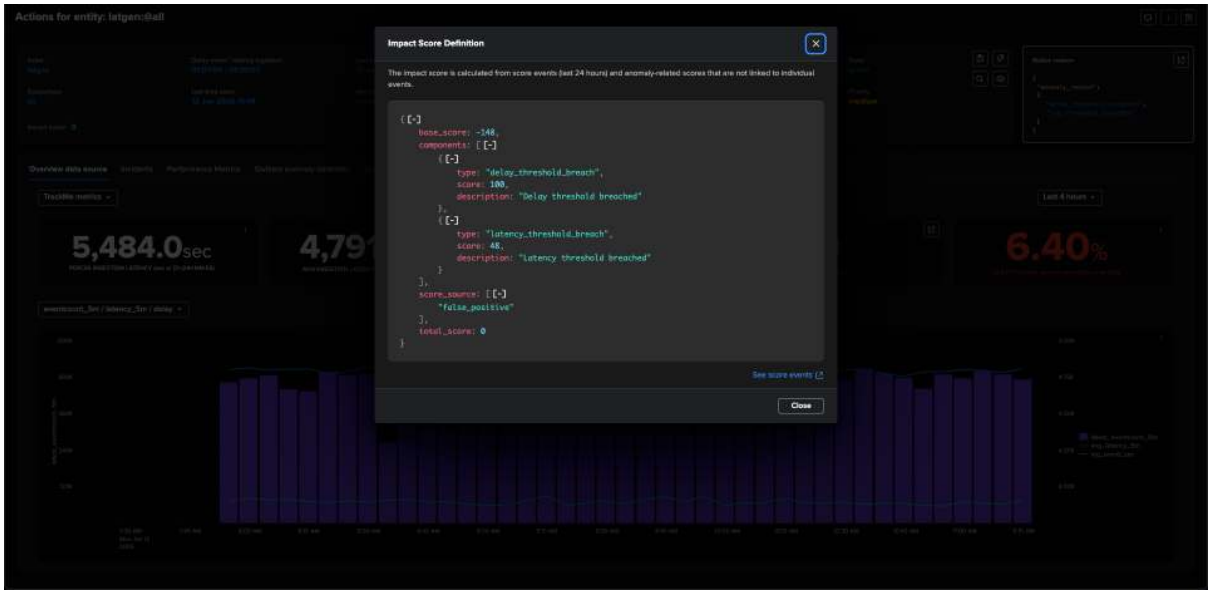
1. Open the entity detail modal
2. Click the kebab menu (three dots)

3. Select “Set as false positive” (first action in “Actions” category)
4. The system generates a negative score event that cancels out all positive scores
5. The entity transitions to green status when *total_score <= 0*
6. Anomaly reasons remain visible for audit purposes

Important Notes:

- False positive actions require power/admin user privileges
- Anomaly reasons are preserved even when suppressed (for audit trail)
- Score events are traceable - you can see what was suppressed and when
- False positives can be reversed by adjusting scores manually if needed





Outliers False Positive:

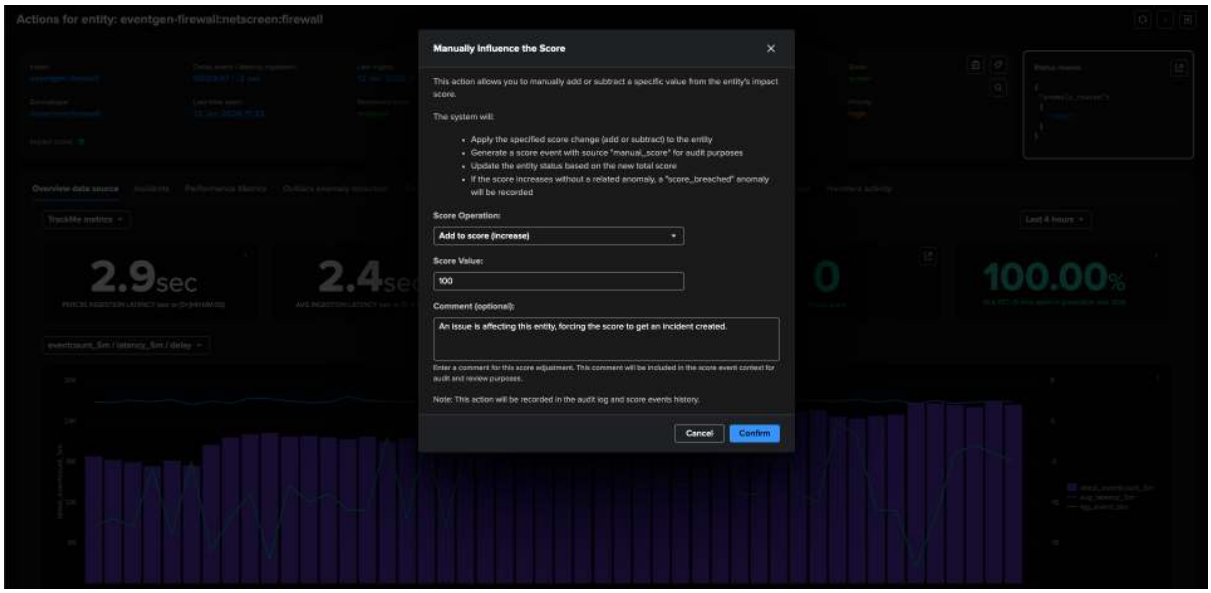
When ML outliers are detected but you determine they are false positives:

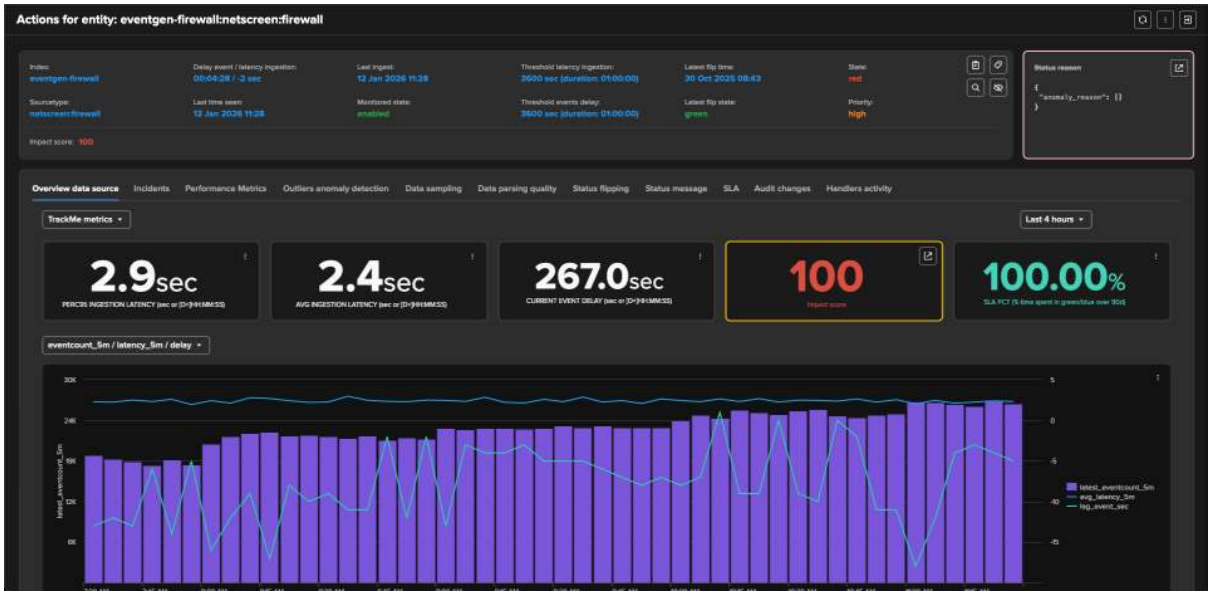
1. Navigate to the Outliers overview tab
2. Click “Set as false positive” button (or use the kebab menu)

3. The system generates a negative score event that cancels out the positive outlier score
4. The entity state transitions based on the resulting score: - If *score_outliers* >= *100*: Red status - If *score_outliers* > *0* and < *100*: Orange status (with *ml_outliers_detection* in anomaly reasons) - If *score_outliers* <= *0*: Green status (outliers suppressed)

Manual Score Influence

For fine-grained control, power/admin users can manually add or subtract specific values from an entity's impact score.





Use Cases:

- Manually adjusting scores for entities that need special attention
- Fine-tuning entity status without triggering false positive suppression
- Adding temporary score adjustments for testing or investigation
- Subtracting scores to reduce alert severity

How to Use Manual Score Influence:

1. Open the entity detail modal

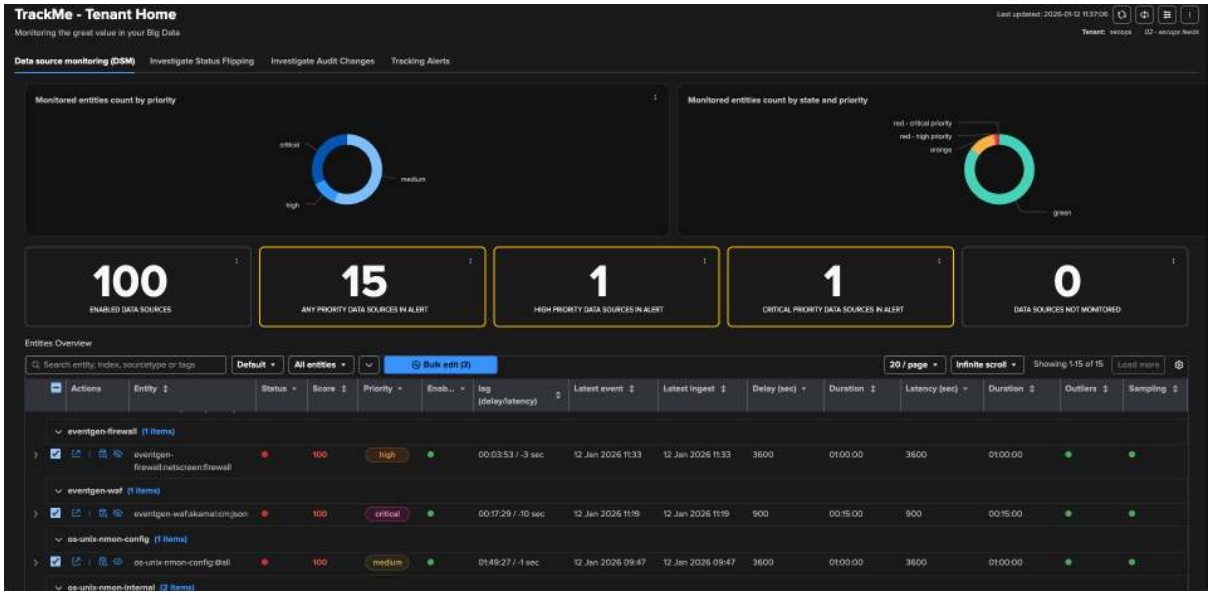
2. Click the kebab menu (three dots)
3. Select “Manually influence the score” (second action in “Actions” category)
4. Choose operation type: - **Add to score**: Increase the score (positive value) - **Subtract from score**: Decrease the score (negative value)
5. Enter a value (1-100)
6. The system generates a score event with *score_source=”manual_score”*
7. Entity status updates based on the new total score

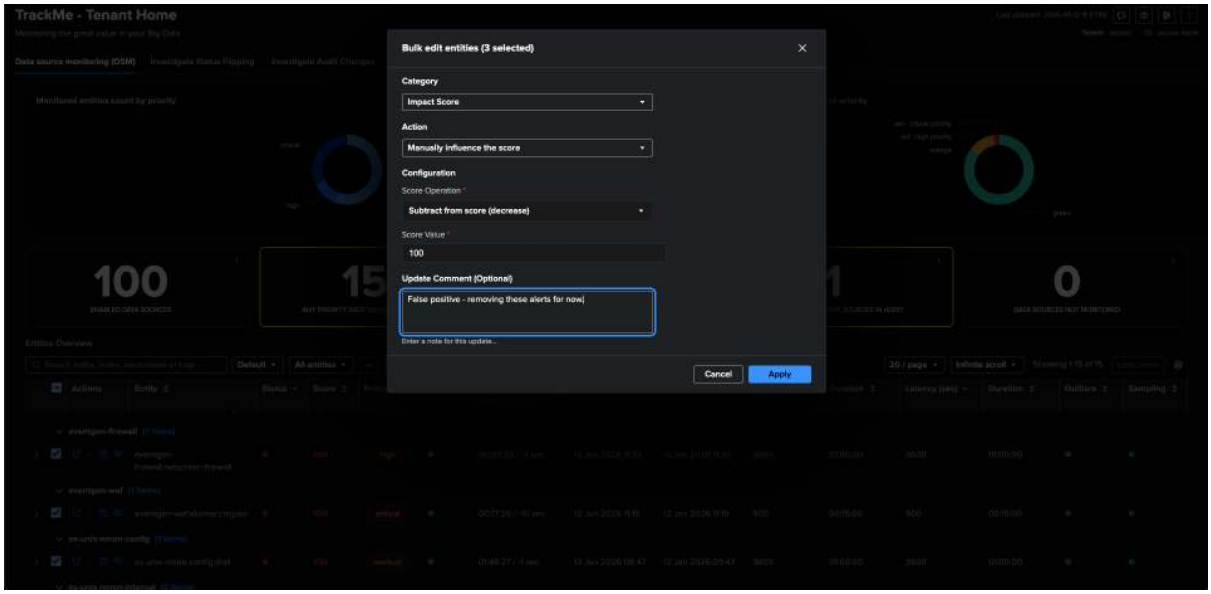
Important Notes:

- Manual score events are included in the score definition breakdown
- If manual score increases cause positive score without related anomalies, a *score_breached* anomaly is recorded
- Manual score events are traceable in score events search
- Manual scores persist until they expire (24-hour window) or are adjusted

Bulk Operations

IBA supports bulk operations for managing impact scores across multiple entities efficiently.





Bulk False Positive:

1. Select multiple entities in the overview table
2. Click “Bulk edit” button
3. Select “Set as false positive” action (first in “Impact Score” category)
4. The system sequentially processes each entity
5. Summary toast notification shows success/skip/error counts

Bulk Manual Score Influence:

1. Select multiple entities in the overview table
2. Click “Bulk edit” button
3. Select “Manually influence the score” action (second in “Impact Score” category)
4. Choose operation type (add/subtract) and enter score value
5. The system sequentially processes each entity
6. Summary toast notification shows success/skip/error counts

Important Notes:

- Bulk operations include a 500ms delay between successful calls to allow Splunk to ingest metrics
- Individual failures don’t stop the entire operation
- API responses indicating “no action needed” are ignored (counted as skips)
- Summary notifications provide clear feedback on the operation results

Impact Scoring for Data Hosts Monitoring (DHM)

The Data Hosts Monitoring (DHM) component tracks data from a host perspective (sourcetypes associated with hosts) and uses similar IBA principles as DSM.

DHM Anomaly Types and Default Impact Scores:

Anomaly Type	Default Score	Description
Delay Threshold Breach	100	Data flow is interrupted - no new events from host (static delay policy)
Variable Delay Threshold Breach	100	Data flow is interrupted - no new events from host within the active variable delay slot threshold (variable delay policy, since v2.3.12)
Latency Threshold Breach	48	Events from host are being indexed but with high latency
Future Tolerance Breach	36	Events detected with timestamps too far in the future
ML Outliers Detection	36 (default)	Machine Learning models detected abnormal volume patterns

Key Differences from DSM:

- DHM focuses on host-level tracking rather than index/sourcetype combinations
- DHM does not include "Data Sampling Anomaly" or "Min Hosts Dcount Breach" anomaly types
- Otherwise, the scoring mechanism works identically to DSM

Configuring Impact Scores for DHM:

The configuration process is identical to DSM:

1. During Virtual Tenant creation, select "splk-dhm" component
2. Configure impact scores in the "Impact score configuration" section: - **Outliers - Default Impact Score**: Default: 36 - **DHM - Delay Threshold Breach Impact Score**: Default: 100 - **DHM - Latency Threshold Breach Impact Score**: Default: 48 - **DHM - Future Tolerance Breach Impact Score**: Default: 36

Per-Entity Score Weight Overrides:

Like DSM, DHM also supports per-entity score weight overrides through the lagging policy configuration. You can override delay and latency impact score weights for individual host entities using the same process described in the DSM section above.

Best Practices for DHM:

- Similar to DSM, keep Delay Threshold Breach at 100 (critical)
- Adjust Latency Threshold Breach based on your requirements (default: 48)
- Consider host-specific requirements when configuring scores
- Use false positive management for known host maintenance windows

Extending IBA to Other TrackMe Components

While this guide focuses on feeds tracking (DSM and DHM), IBA extends to all TrackMe components. Here's a brief overview:

Metrics Host Monitoring (MHM):

- **Metric Alert**: Default score 100 (critical metric threshold breached)
- **Future Tolerance Breach**: Default score 36

Flex Objects (FLX):

- **Inactive**: Default score 100 (entity is inactive)
- **Status Not Met**: Default score 100 (custom status condition not met)
- **Threshold Breach**: Configurable per threshold (default: 100)

Fields Quality Monitoring (FQM):

- **Status Not Met**: Default score 100 (quality condition not met)

- **Threshold Breach**: Configurable per threshold (default: 100)

Workload (WLK):

- **Skipping Searches**: Default score 100
- **Execution Errors**: Default score 100
- **Orphan Search**: Default score 100
- **Execution Delayed**: Default score 100
- **Out of Monitoring Times**: Default score 100
- **Status Not Met**: Default score 100

Configuration for Other Components:

The impact score configuration process is similar across all components:

1. Create or manage a Virtual Tenant for the component
2. Configure impact scores in the wizard or management UI
3. Adjust scores based on your priorities and requirements
4. Use false positive management and manual score influence as needed

Score Definition Modal and Transparency

The Score Definition Modal provides complete transparency into how impact scores are calculated.

Accessing the Score Definition Modal:

- Click on the impact score value in the entity detail modal (cartouche)
- Click on the impact score value in the overview table
- Click the link icon on the Impact Score single view card in overview tabs

What the Modal Shows:

1. **Explanation Text**: "The impact score is calculated from score events and anomaly-related scores that are not linked to individual events."
2. **Score Definition Breakdown**: JSONTree component showing: - *base_score*: Base score from score events - *components*: Array of anomaly components with scores - *score_outliers*: Outlier score if present - *score_source*: Array of score sources - *total_score*: Final calculated score
3. **Score Events Link**: "See score events" link that opens a Splunk search showing all score events for the entity in the last 24 hours

Understanding the Breakdown:

Each component in the score definition includes: - **type**: The anomaly type (e.g., "delay_threshold_breach") - **score**: The impact score for this anomaly - **description**: Human-readable description

This breakdown makes it easy to understand: - Why an entity is in a particular state - Which anomalies are contributing to the score - How much each anomaly contributes - Whether false positives or manual scores are affecting the total

Best Practices and Recommendations

Getting Started:

1. **Start with Defaults**: Begin with default impact scores and observe behavior
2. **Monitor False Positives**: Track false positive rates and adjust scores accordingly
3. **Use Score Definition Modal**: Regularly review score breakdowns to understand entity states
4. **Leverage False Positive Management**: Use false positive suppression for known issues

5. **Gradual Tuning**: Adjust scores incrementally based on your environment's needs

Scoring Strategy:

- **Critical Anomalies**: Keep at 100 (e.g., delay threshold breach, inactive entities)
- **Warning Anomalies**: Use 36-60 range (e.g., latency, outliers, future tolerance)
- **Quality Anomalies**: Adjust based on quality requirements (e.g., data sampling)
- **Balance Detection and False Positives**: Adjust ML outlier scores based on detection accuracy

Operational Practices:

- **Regular Review**: Periodically review score definitions to ensure they match your priorities
- **Documentation**: Document any custom score configurations for your team
- **Audit Trail**: Use score events search to investigate entity state changes
- **Testing**: Use manual score influence to test alerting thresholds before making permanent changes

Troubleshooting:

- **Unexpected Red Status**: Check score definition modal to see which anomalies contributed
- **False Positives**: Use false positive management to suppress known issues
- **Score Not Updating**: Verify score events are being generated (check score events search)
- **Configuration Not Applied**: Ensure Virtual Tenant account has impact score fields configured

Conclusion

Impact-Based Alerting (IBA) represents a fundamental shift in how TrackMe evaluates and alerts on data quality issues. By providing:

- **Flexible Configuration**: Fine-tune impact scores per tenant, component, and anomaly type
- **Complete Transparency**: Detailed score breakdowns and traceable score events
- **False Positive Management**: One-click suppression with full audit trail
- **Manual Control**: Fine-grained score adjustments for special cases
- **Bulk Operations**: Efficient management across multiple entities

IBA puts you in complete control of how TrackMe evaluates entity health, making alerting smarter, more tunable, and far more actionable.

Next Steps:

- Create or update Virtual Tenants with impact score configuration
- Monitor entity states and review score definitions regularly
- Adjust impact scores based on your environment's priorities
- Leverage false positive management and manual score influence as needed
- Explore IBA features for other TrackMe components (MHM, FLX, FQM, WLK)

For more information about specific TrackMe components and advanced IBA features, refer to the TrackMe administration and user guides.

QUICK START - Design and automate through lookup knowledge

Automate priority, tags and SLA management through a single lookup

- This quickstart guides you through the design and implementation of **lookup-based policies** in TrackMe, a powerful approach to automate the management of priority, tags, and SLA classes across your monitored entities.
- The key idea: **a single lookup file** (CSV or KVstore) containing your knowledge about your Splunk environment can drive **all three policy types** at once — priority assignment, tag automation, and SLA classification.
- **This quickstart provides two end-to-end examples:**
  - **Data source monitoring (DSM):** Using an index-based lookup to assign priorities, tags, and SLA classes to data source entities
  - **Data host monitoring (DHM):** Using a host-based lookup to assign priorities, tags, and SLA classes to data host entities
- **Prerequisites:** An existing Virtual Tenant with discovered entities, and a lookup file uploaded to Splunk (via the Splunk App for Lookup File Editing or any other method).

Hint

For detailed documentation on each individual policy type, refer to:

- *Managing priority via policy* — Priority policies
- *Accessing, defining and managing tags policies* — Tags policies
- *Defining SLA policies to assign SLA classes automatically* — SLA policies

The concept: one lookup, three policies

The idea behind lookup-based policies is simple: you maintain a single lookup file that contains your knowledge about your Splunk environment, and TrackMe uses it to automate the management of priority, tags, and SLA classes across your monitored entities.

Consider the following lookup file `example_cmdb_indexes.csv` with three columns:

Example:

index	priority	tags
os	low	os
prd*_*_linux	critical	production
webserver	critical	web

From this single lookup, TrackMe can create three policies:

- **Priority policy:** Map the `index` column to entity fields, and use the `priority` column to assign priority levels (low, medium, high, critical) to matching entities.
- **Tags policy:** Using the same lookup and field mapping, use the `tags` column with a comma separator to assign one or more tags to matching entities.
- **SLA policy:** Using the same lookup and field mapping, use the `priority` column with **value mappings** to translate priority levels into SLA classes (e.g., `critical` → `gold`, `high` → `silver`, `low` → `platinum`).

Hint

This approach is particularly powerful because:

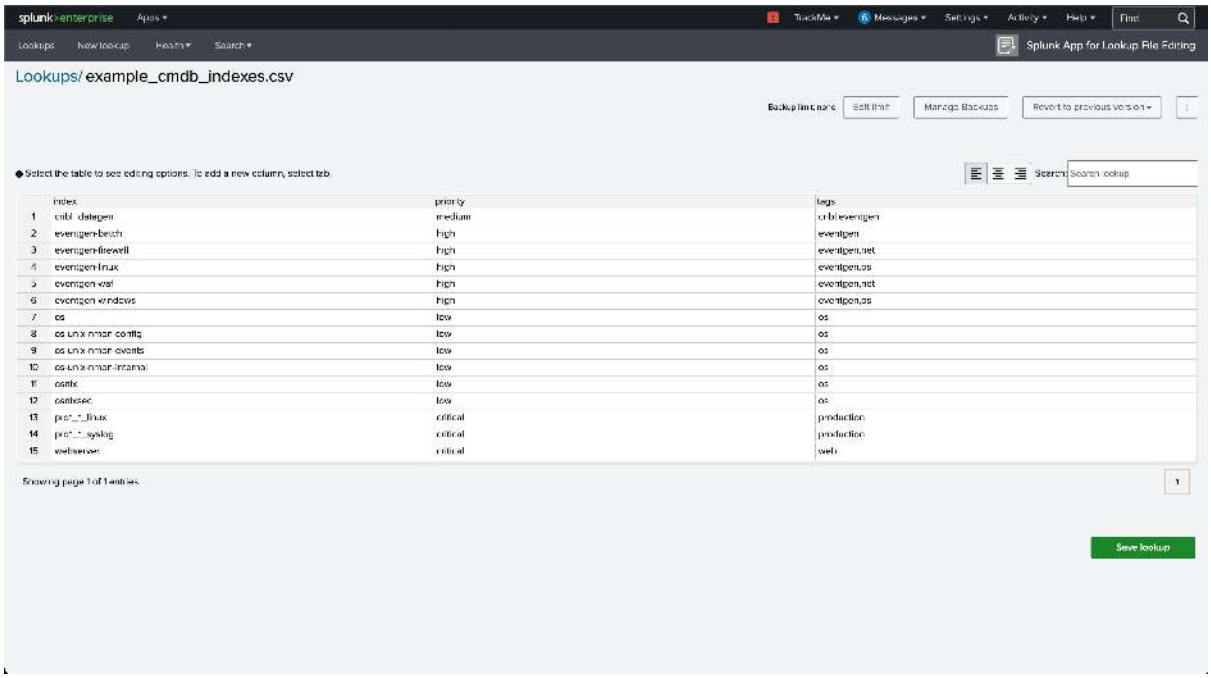
- All three policies share the same lookup file — a single source of truth for your Splunk knowledge.
- Updates to the lookup are automatically reflected when the policy trackers run.
- Field mappings support **wildcard** matching (using `*` and `?` patterns), so you can match complex index or host naming conventions with simple patterns.
- Each policy includes a **simulation mode** that lets you preview the results before applying changes.

Data source monitoring (DSM) example

In this example, we will demonstrate the complete workflow for data source monitoring, using the `example_cmdb_indexes.csv` lookup shown above. We will map the `index` field in the lookup to the `data_index` entity field, which allows matching data source entities by the index they belong to.

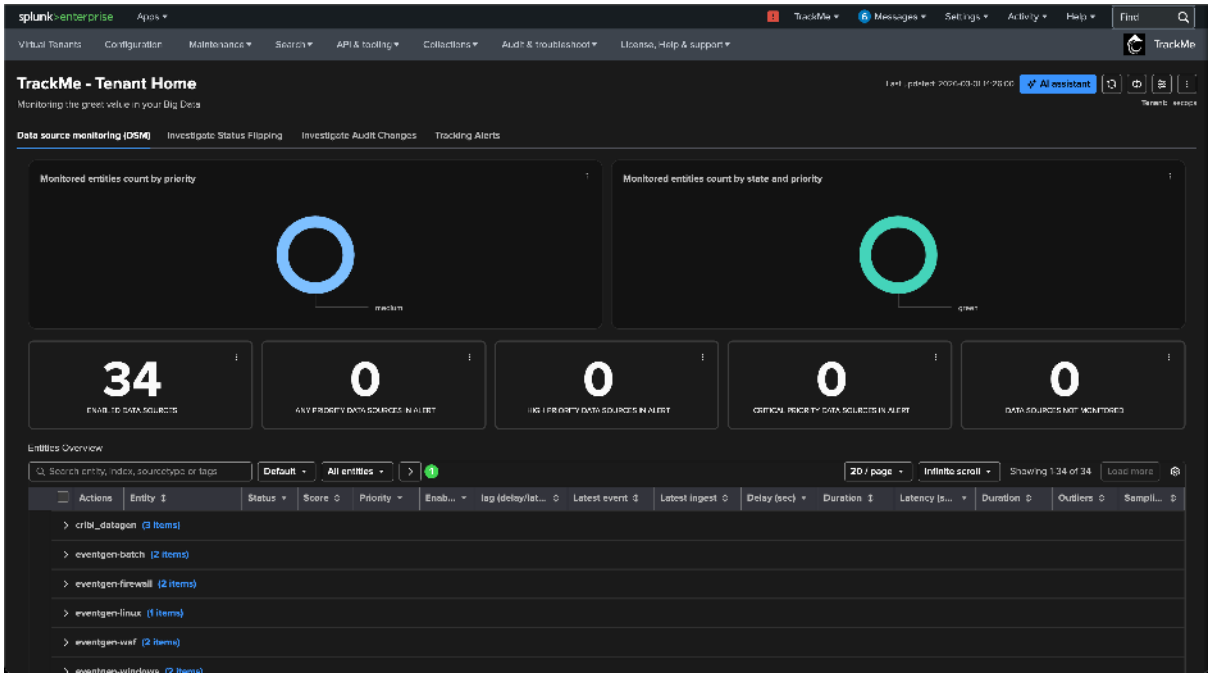
The lookup file

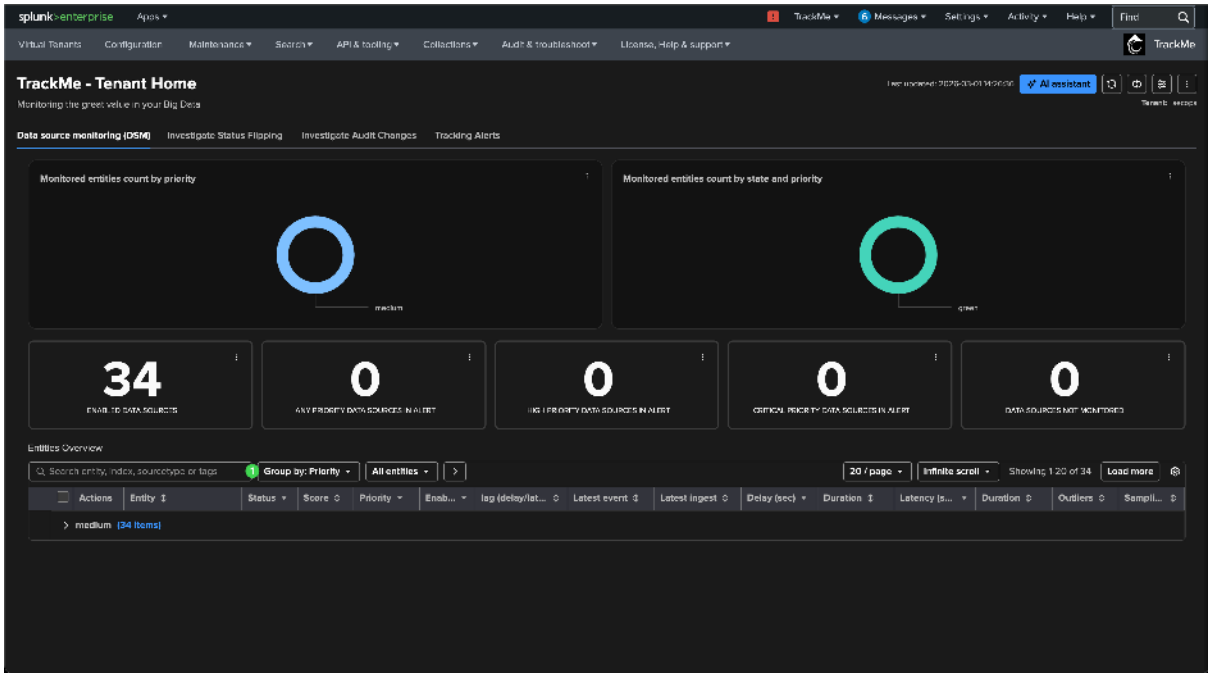
The lookup file `example_cmdb_indexes.csv` is uploaded to Splunk and contains 15 rows mapping indexes to priorities and tags:



Initial state

Before creating any policies, all 34 data source entities are at the default priority level (medium) and have no tags or SLA classes assigned:

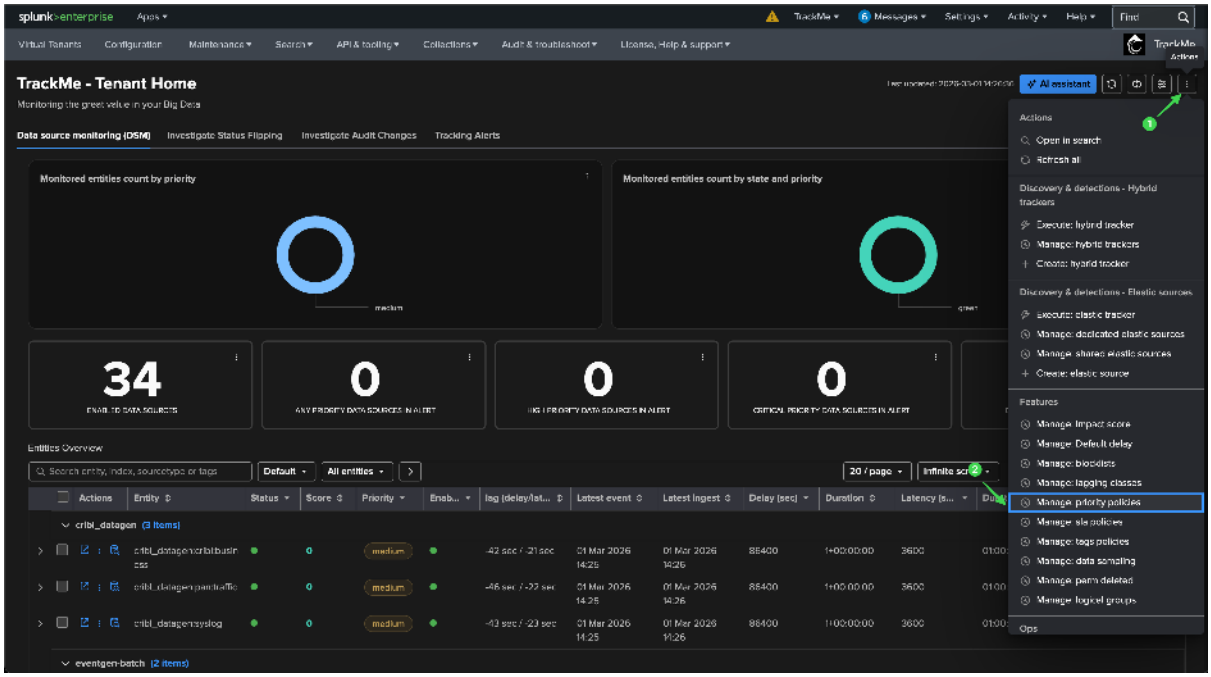


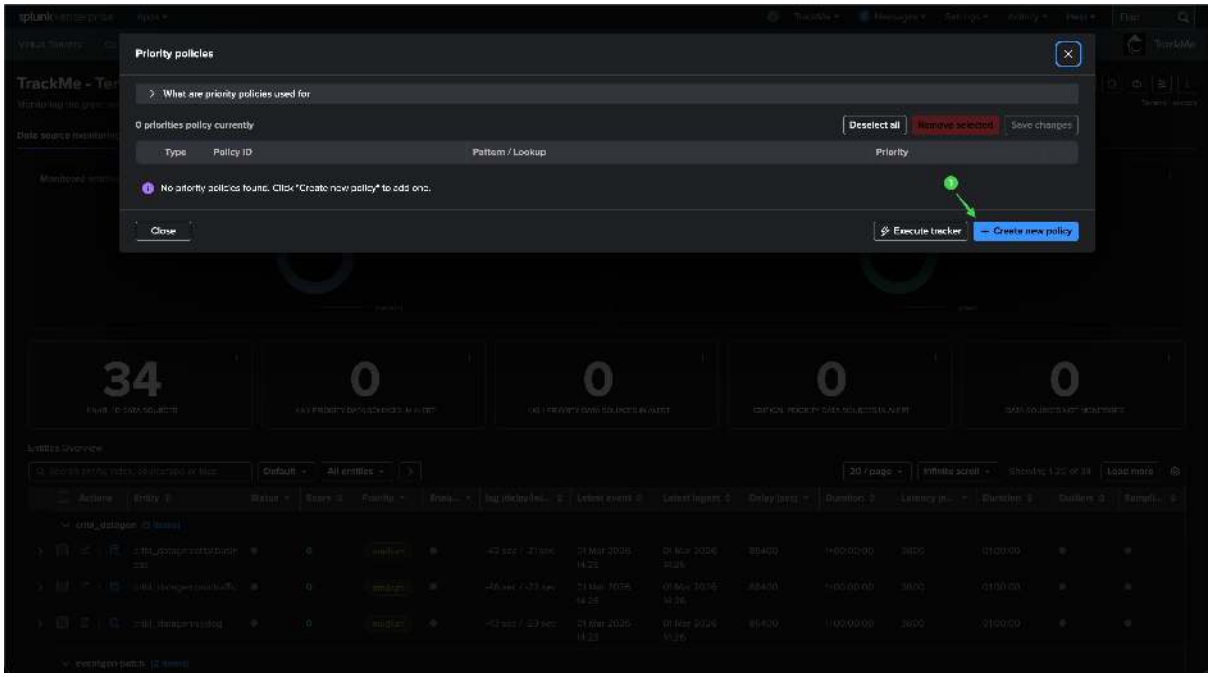


Step 1: Create a priority policy

Access the priority policies screen:

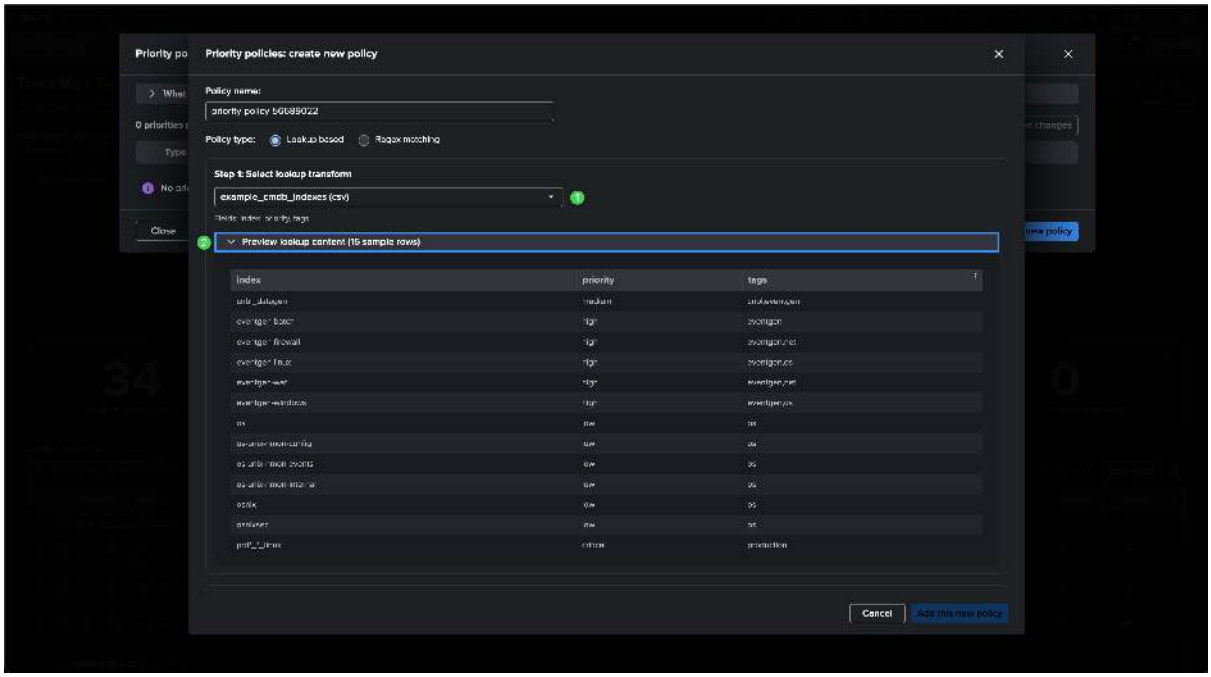
From the Tenant Home page, click the **Actions** button in the top-right corner, then select **Manage priority policies**.

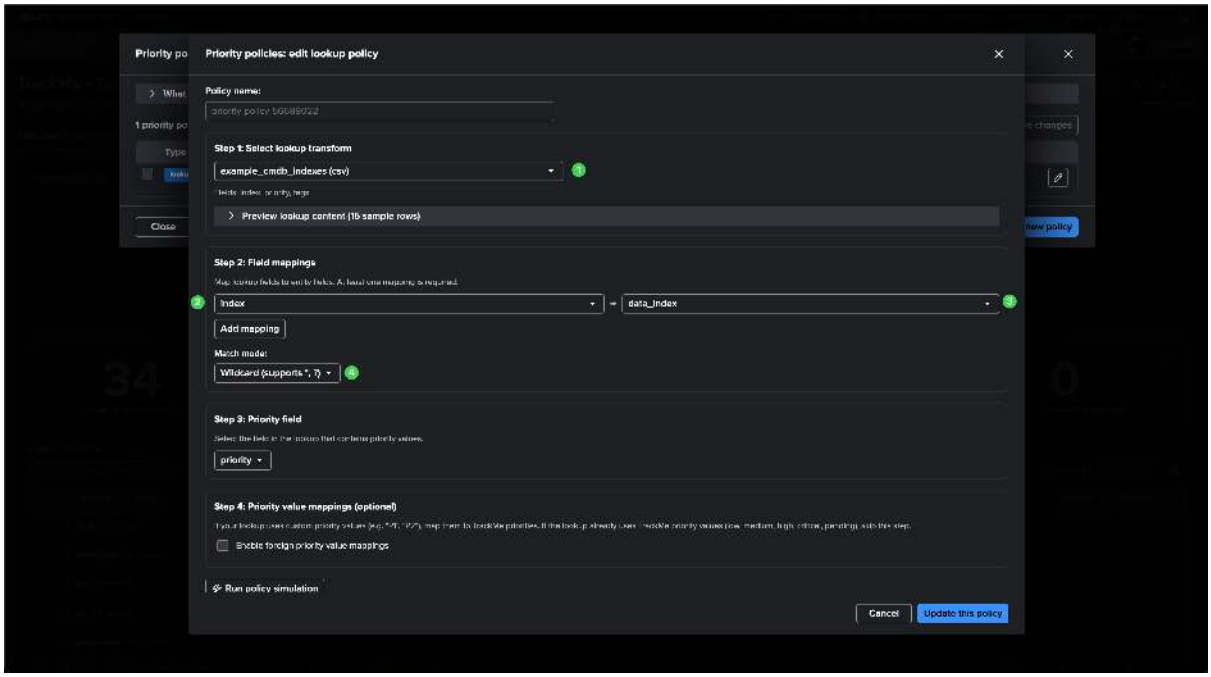


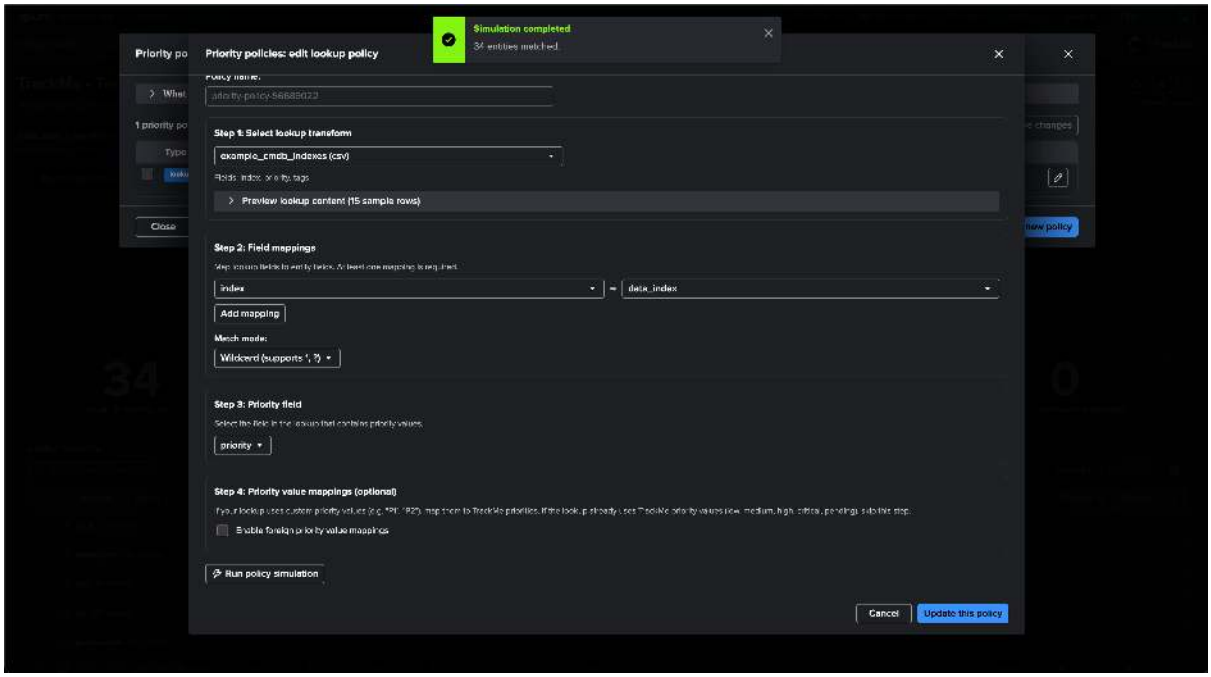


Create a lookup-based priority policy:

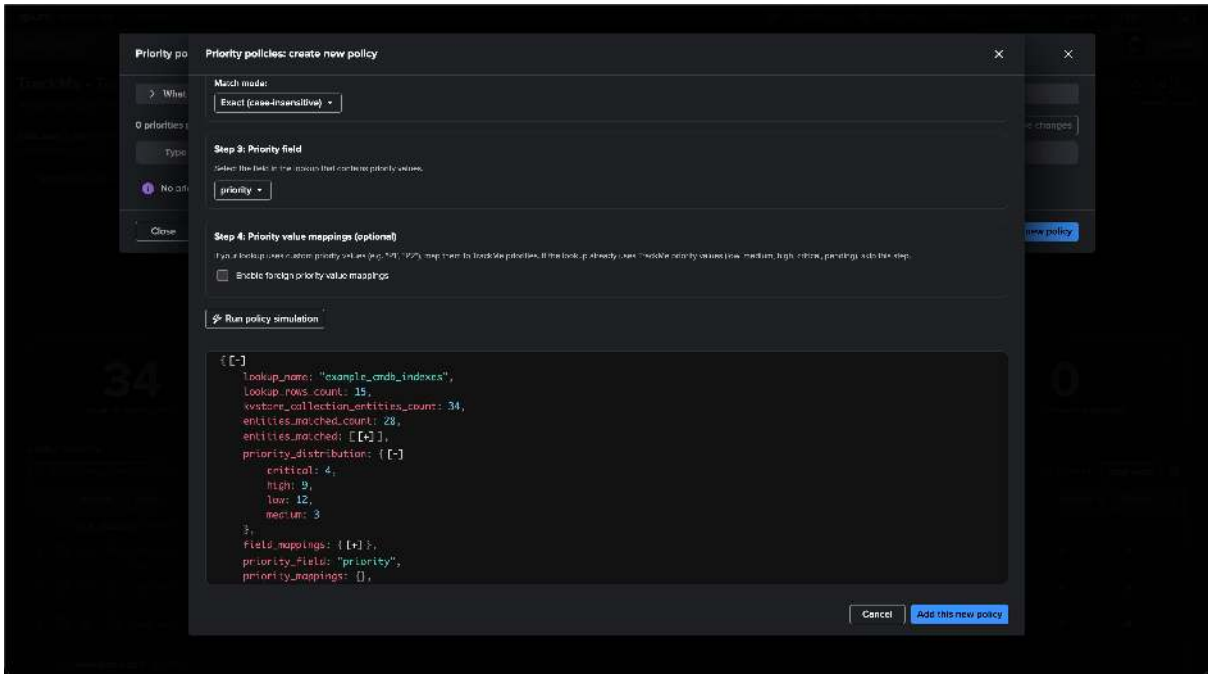
- Click **Create new policy** and select the **Lookup based** mode
- In **Step 1**, select the lookup transform `example_cmdb_indexes (csv)` — TrackMe lists all available lookup transforms and shows a preview of sample rows
- In **Step 2**, configure the **field mapping**: map the lookup field `index` to the entity field `data_index`, and select the **match mode** (Wildcard or Exact)
- In **Step 3**, select `priority` as the **priority field** — the lookup already uses TrackMe priority values (low, medium, high, critical), so no value mappings are needed
- In **Step 4**, priority value mappings are optional and not required in this example since the lookup already uses TrackMe-compatible priority values


Priority policies: create new policy
Policy name:
Policy type:
Lookup based
Regex matching
Step 1: Select lookup transform
example_cmdb_indexes (csv)
Preview lookup content (15 sample rows)
index
priority
tags
Cancel


Priority policies: edit lookup policy
Policy name:
Step 1: Select lookup transform
example_cmdb_indexes (csv)
Preview lookup content (15 sample rows)
Step 2: Field mappings
index
data_index
Add mapping
Match mode:
Wildcard (supports *, ?)
Step 3: Priority field
priority
Step 4: Priority value mappings (optional)
Enable foreign priority value mappings
Run policy simulation
Cancel
Update this policy

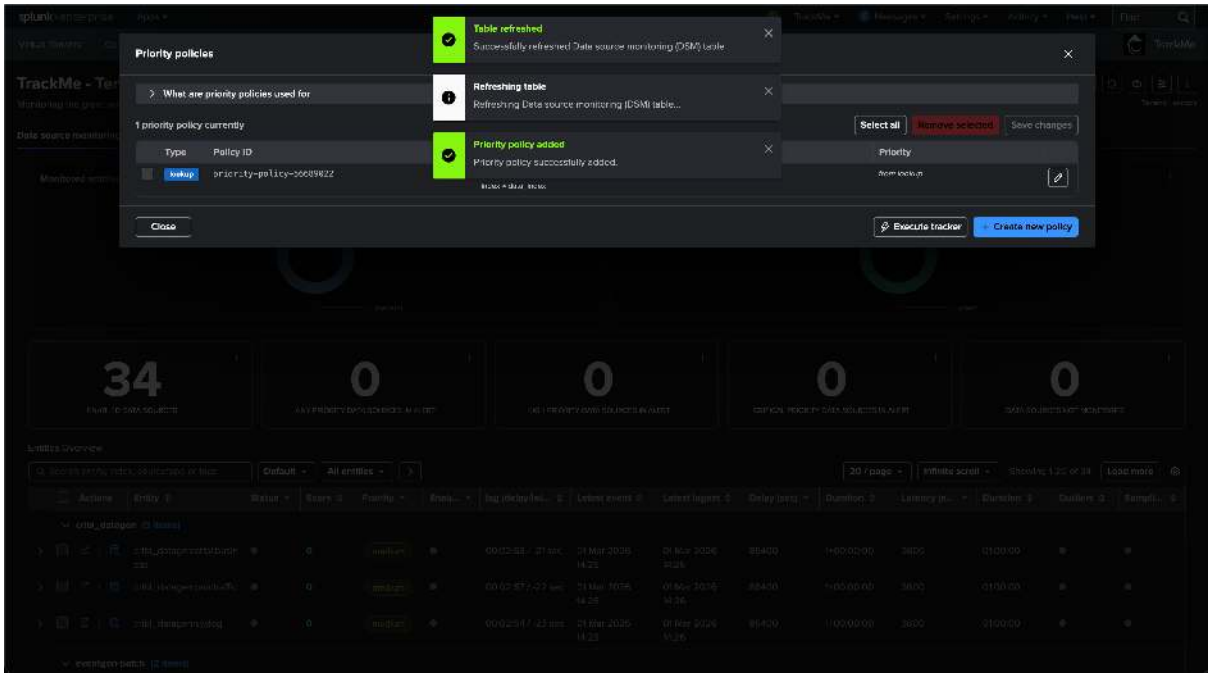


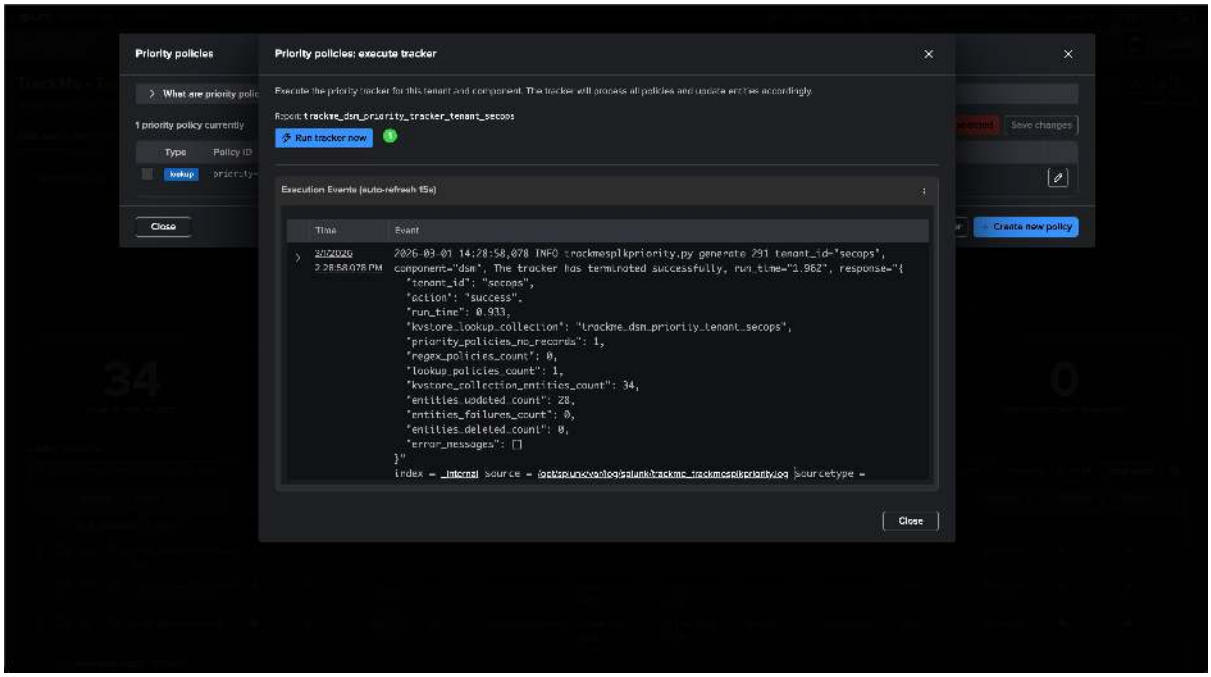
Run the simulation to preview which entities would be matched and what priorities would be assigned. The simulation results show the number of matched entities and the priority distribution:



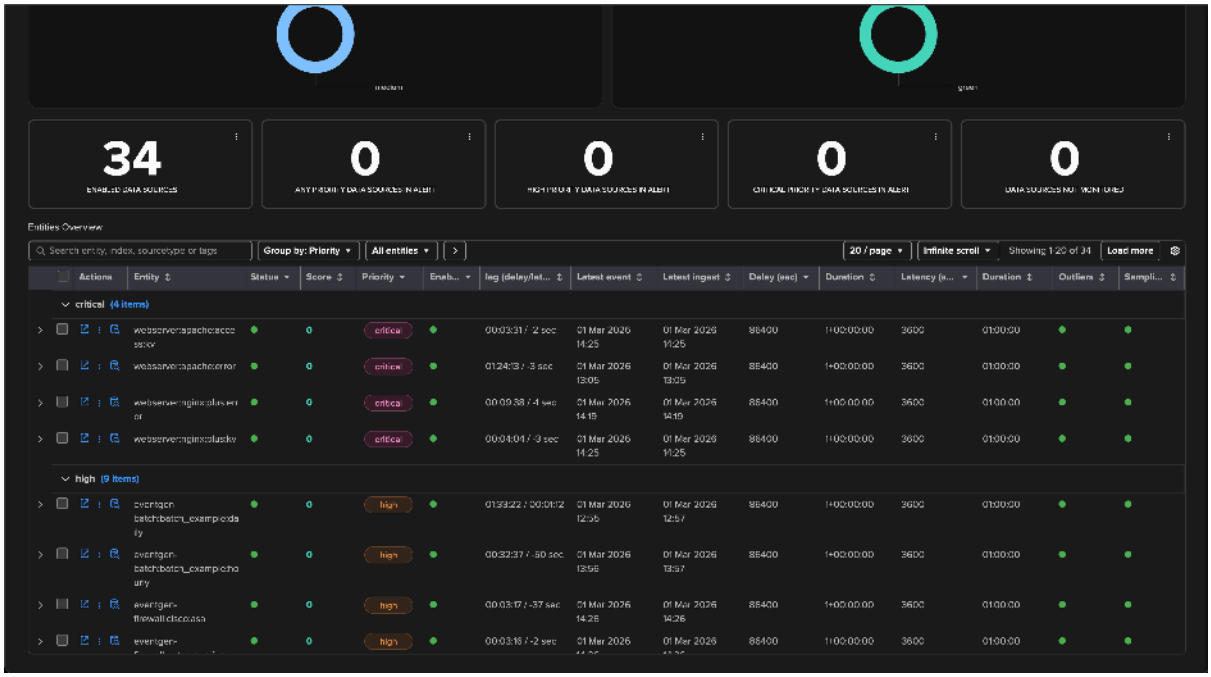
Add the policy and execute the tracker:

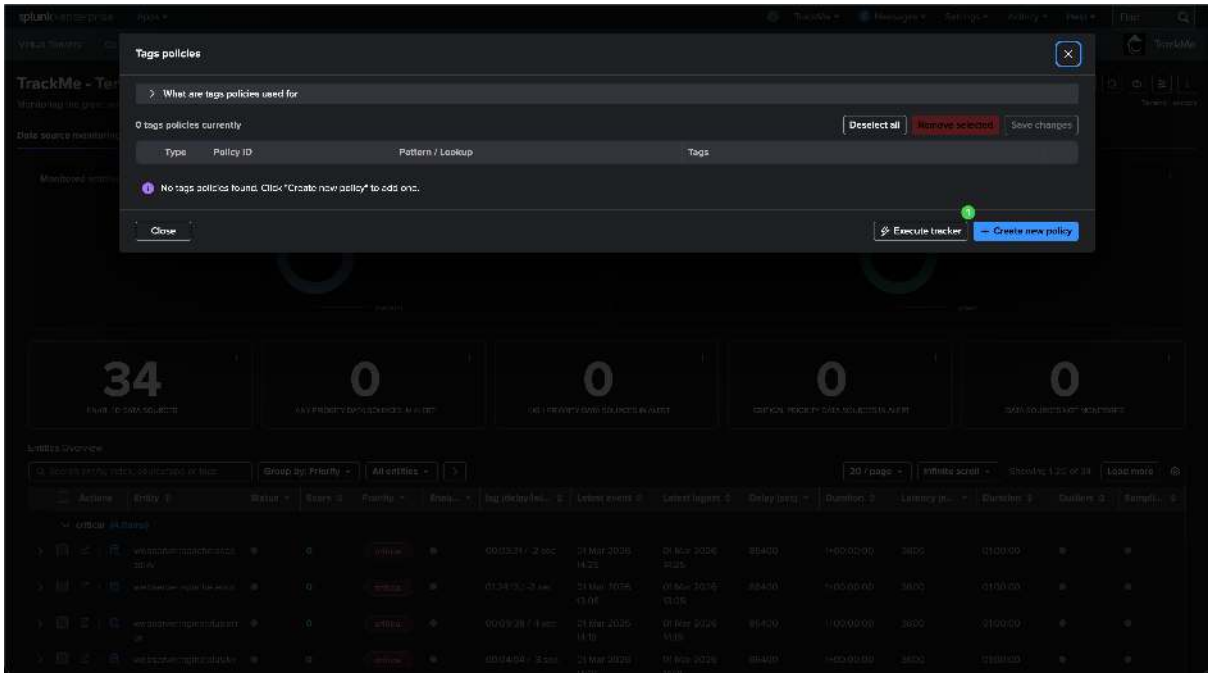
Once the policy is created, click **Execute tracker** to apply the priority assignments to all matching entities:





Result: Entities are now assigned the correct priority levels based on the lookup — the Tenant Home reflects the new priority distribution (critical, high, medium, low):

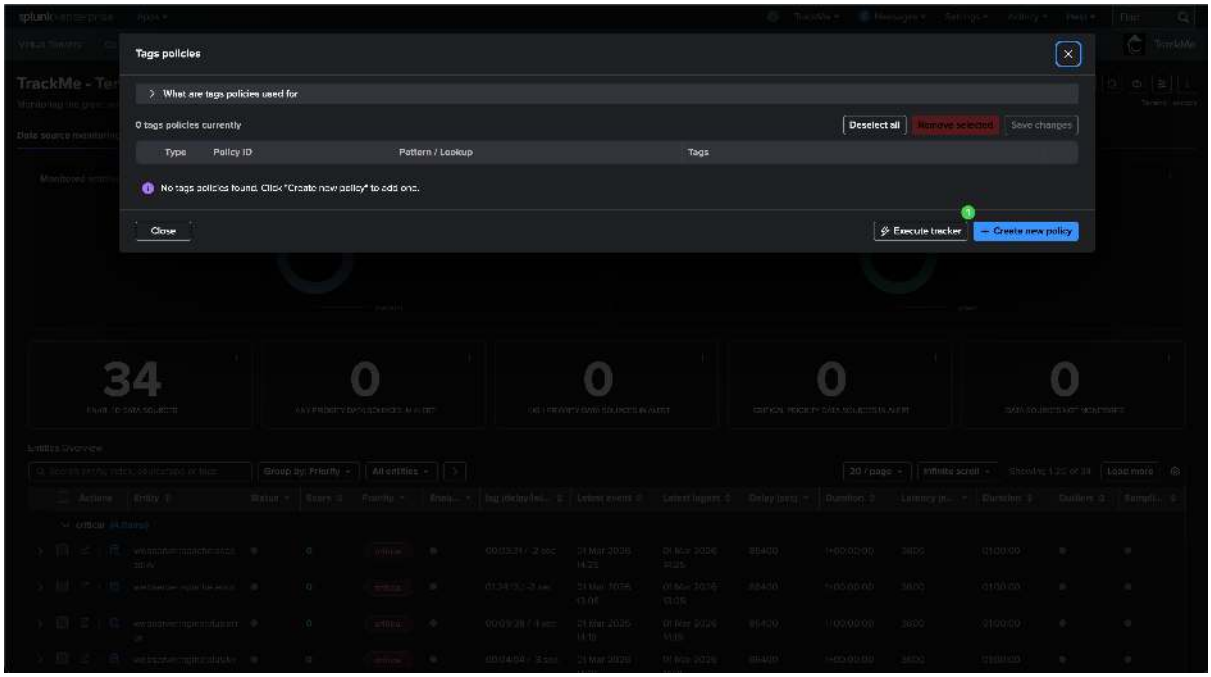




Step 2: Create a tags policy

Access the tags policies screen:

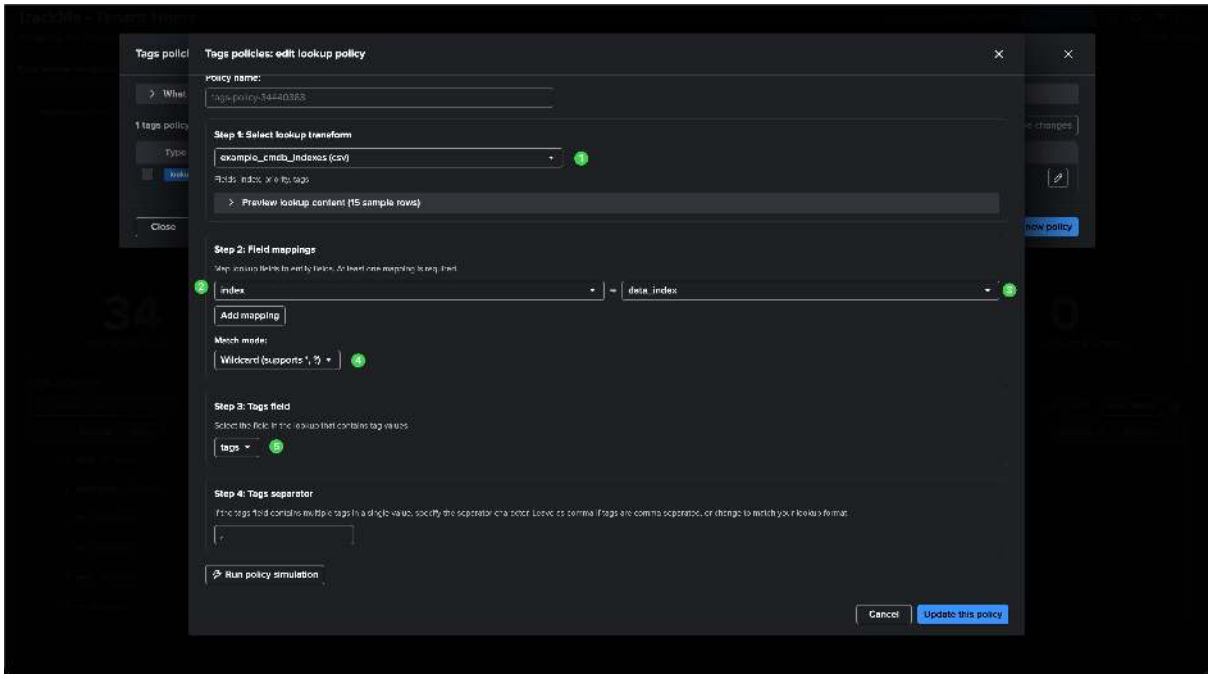
From the Tenant Home page, click the **Actions** button, then select **Manage tags policies**.



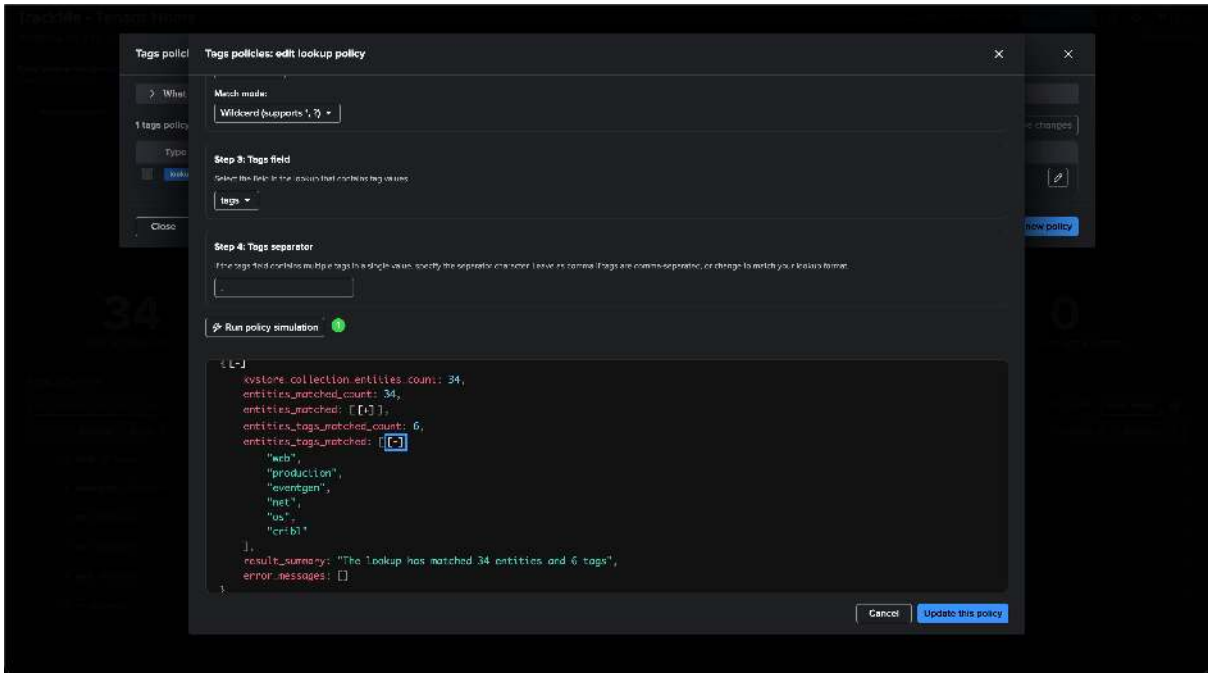
Create a lookup-based tags policy:

Using the same lookup transform, configure the tags policy:

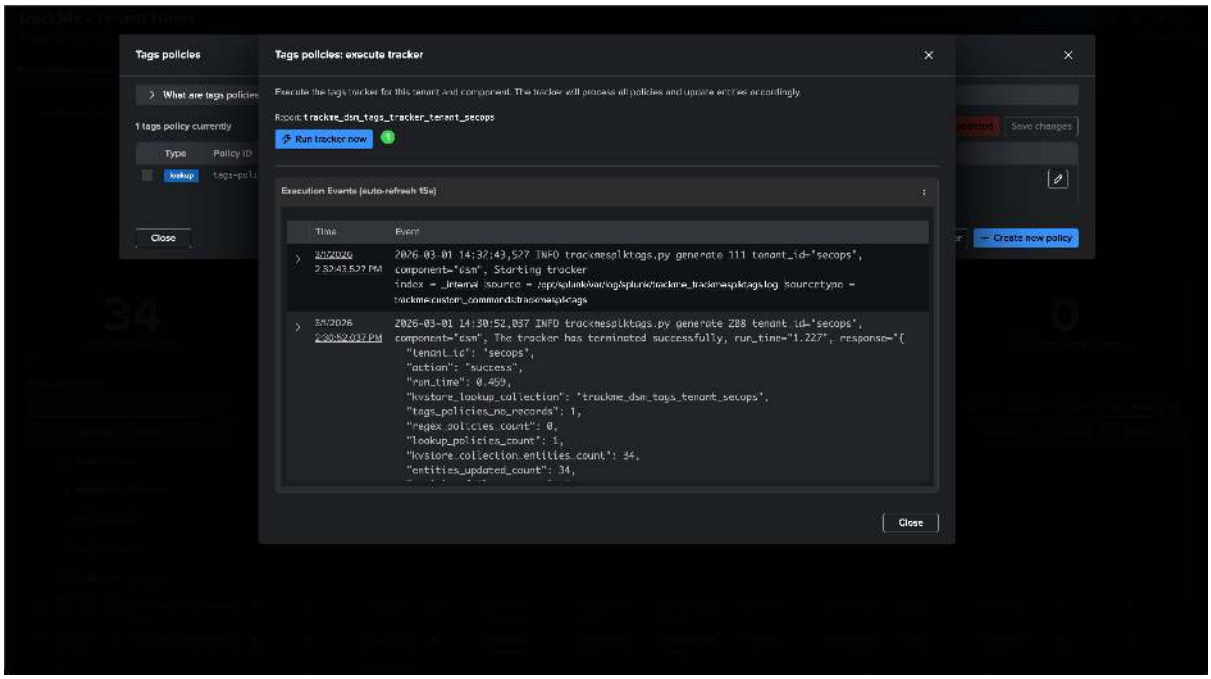
- In **Step 1**, select the same lookup transform `example_cmdb_indexes (csv)`
- In **Step 2**, configure the same **field mapping**: `index` → `data_index`, with **Wildcard** match mode
- In **Step 3**, select `tags` as the **tags field**
- In **Step 4**, set the **tags separator** to `,` (comma) — this tells TrackMe to split the tags column value into individual tags (e.g., `eventgen,os` becomes two separate tags: `eventgen` and `os`)



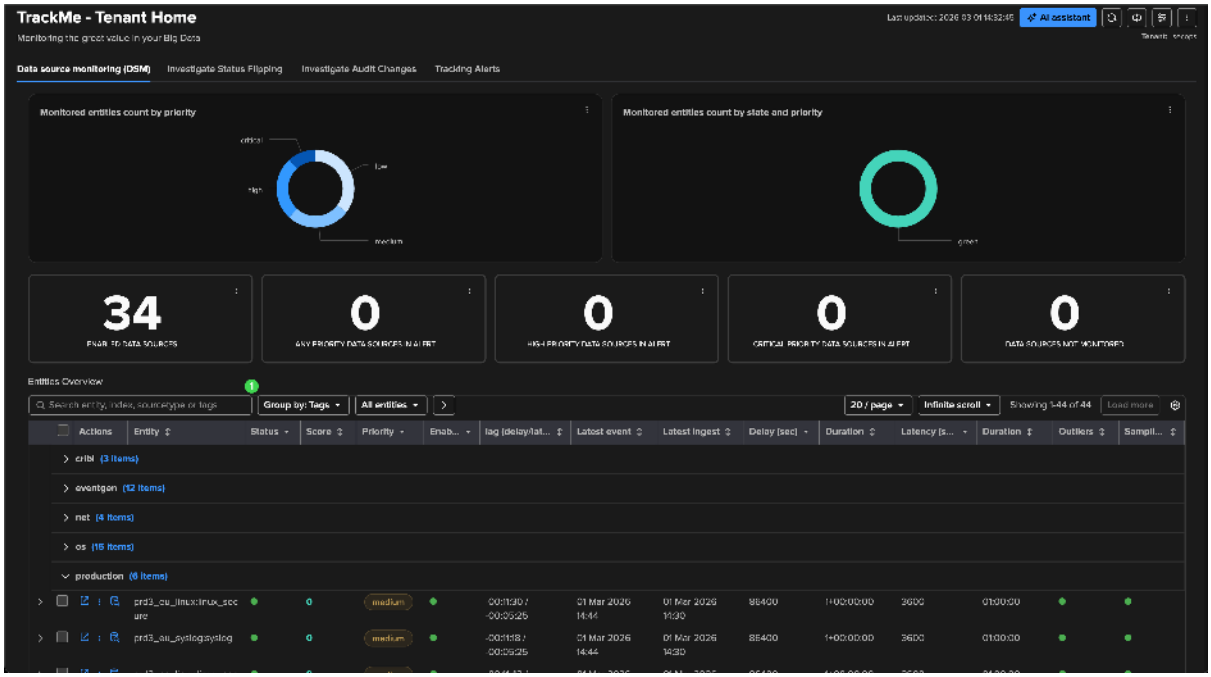
Run the simulation to preview tag assignments. The simulation shows 34 matched entities and 6 distinct tags discovered:

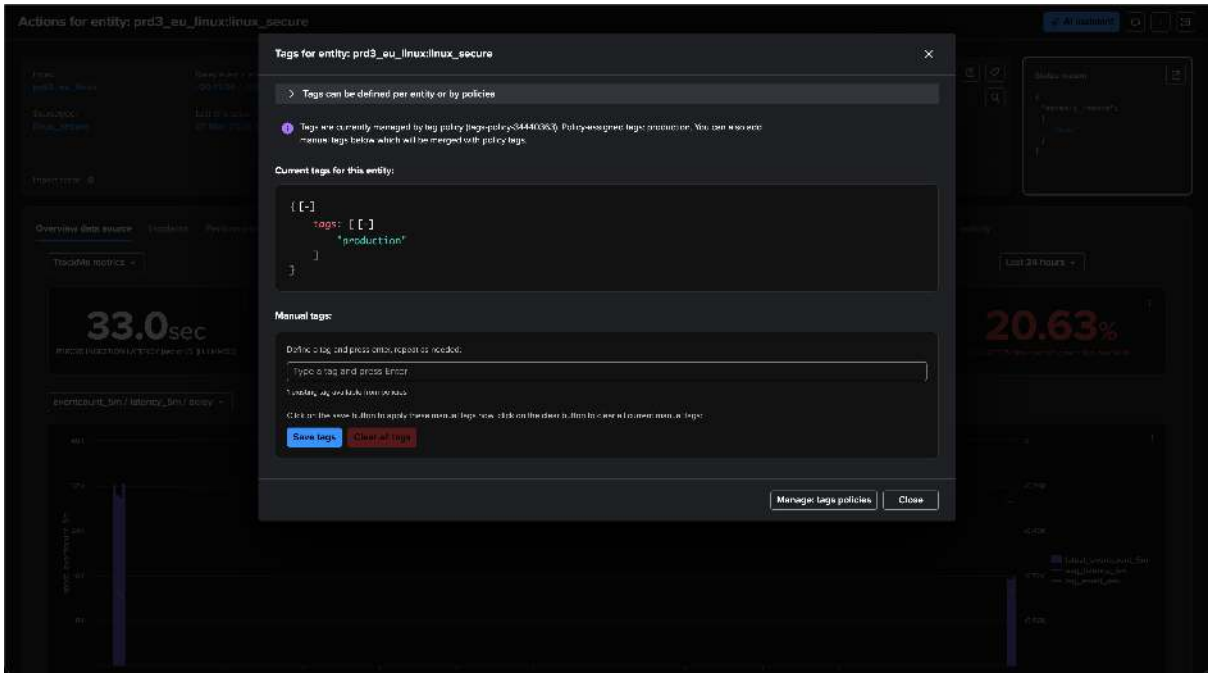


Execute the tracker to apply the tag assignments:



Result: Entities are now tagged, and you can group the Tenant Home by tags to see the tag distribution. Clicking on an entity's tags dialog shows the tags assigned by the policy:

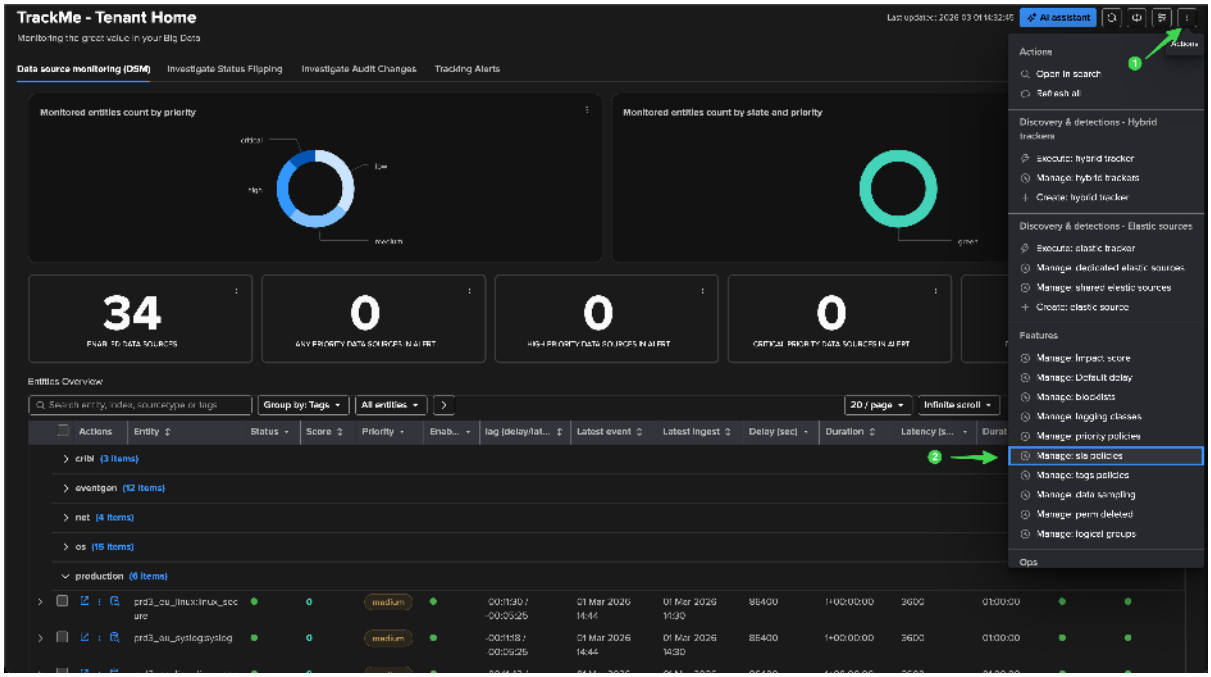




Step 3: Create an SLA policy

Access the SLA policies screen:

From the Tenant Home page, click the **Actions** button, then select **Manage SLA policies**.

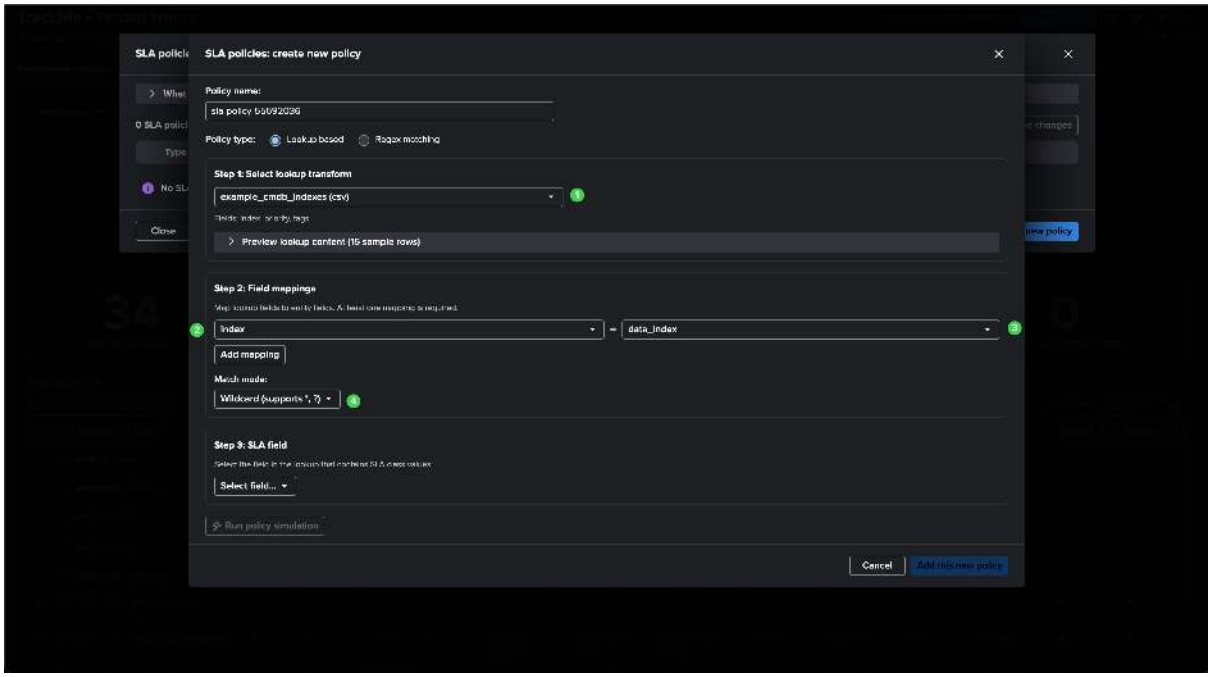


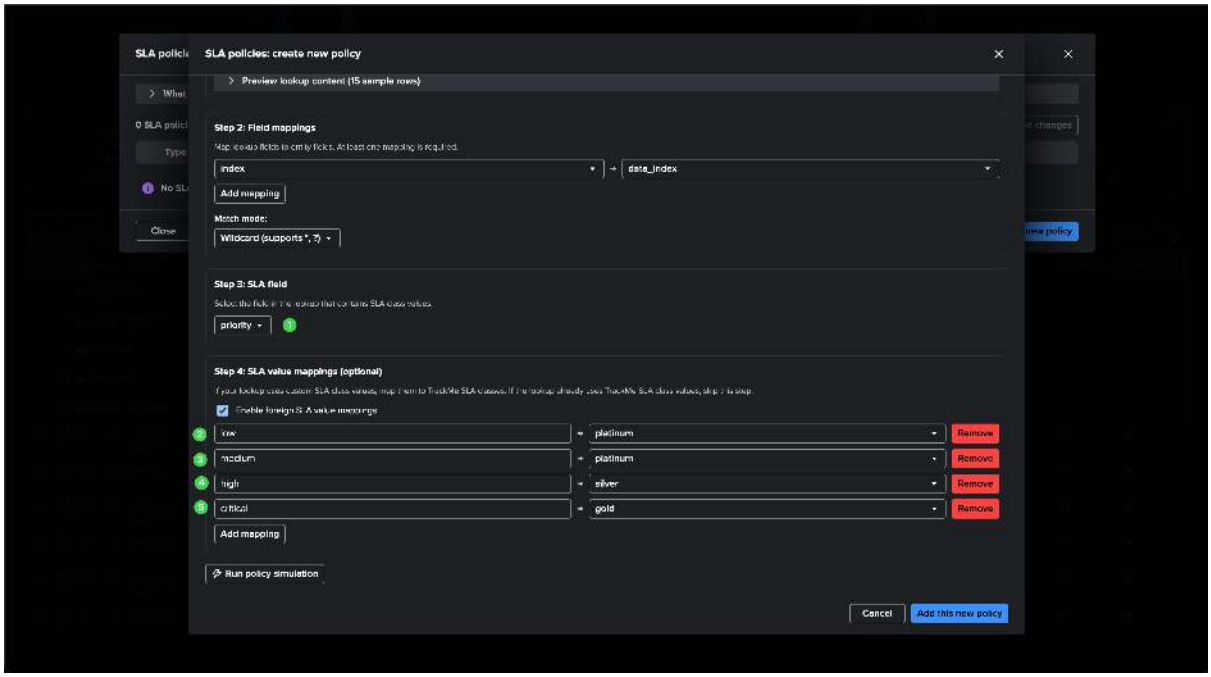
Create a lookup-based SLA policy:

Using the same lookup transform, configure the SLA policy — but this time, we leverage **value mappings** to translate the priority column into SLA classes:

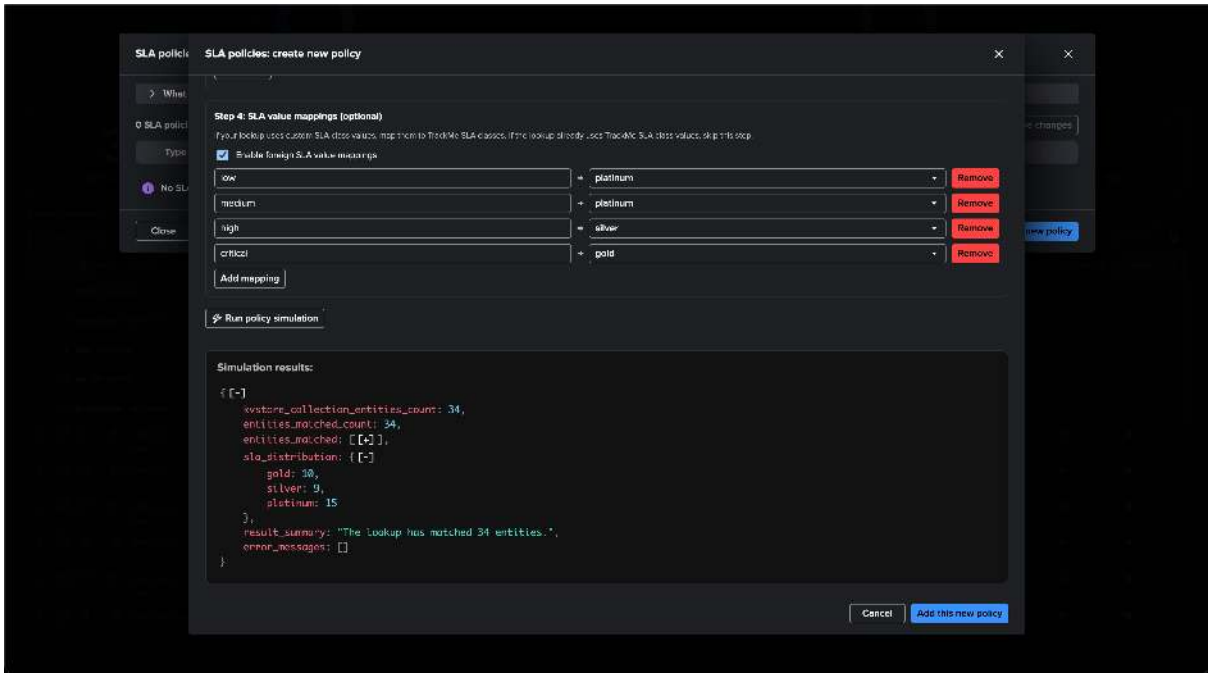
- In **Step 1**, select the same lookup transform `example_cmdb_indexes (csv)`
- In **Step 2**, configure the same **field mapping**: `index` → `data_index`, with **Wildcard** match mode
- In **Step 3**, select `priority` as the **SLA field** — since the lookup does not have a dedicated SLA column, we reuse the priority column and translate values via mappings
- In **Step 4**, enable **SLA value mappings** and define the translation:

Lookup value (priority)	→	SLA class
low	→	platinum
medium	→	platinum
high	→	silver
critical	→	gold

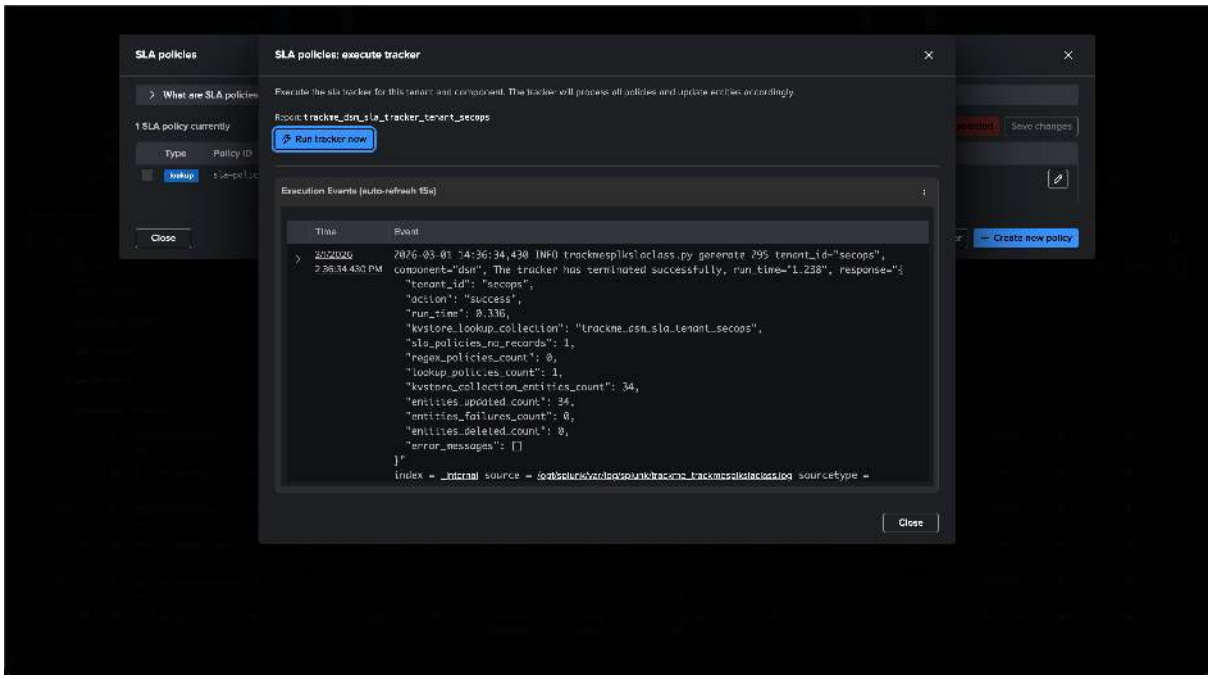




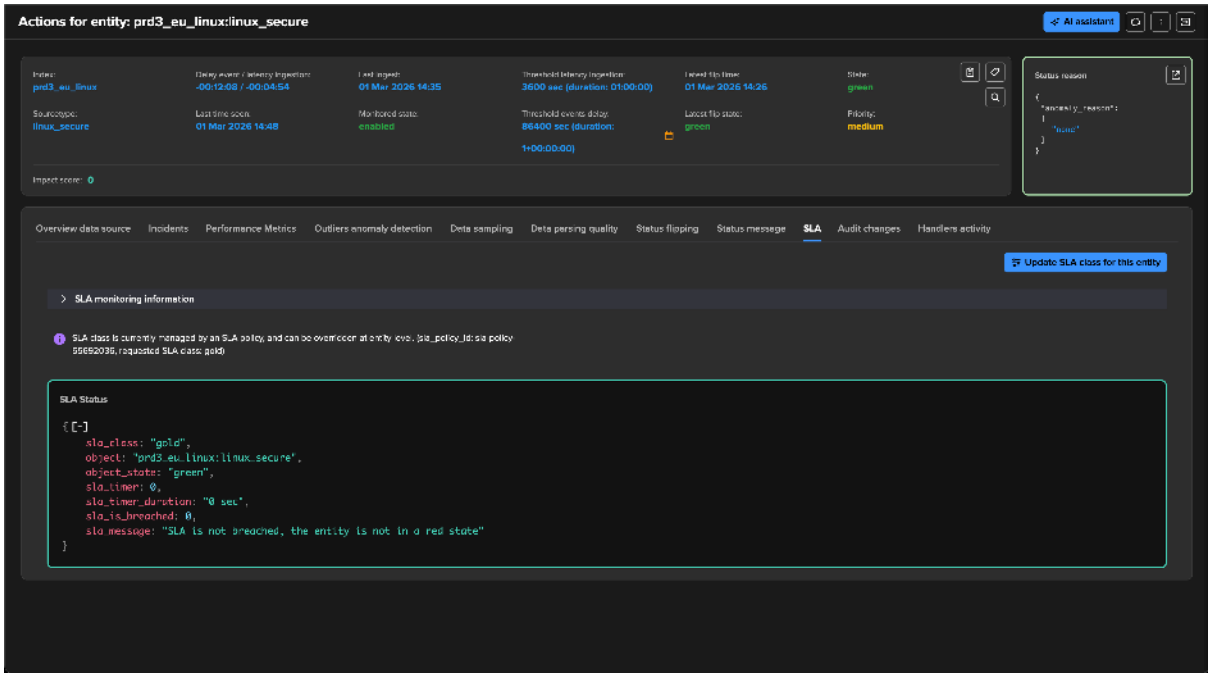
Run the simulation to preview SLA class assignments. The simulation shows the SLA distribution across all 34 entities:



Execute the tracker to apply the SLA class assignments:



Result: Opening an entity's SLA tab confirms the SLA class has been assigned by the policy. In this example, `prd3_eu:linux:linux_secure` is assigned the `gold` SLA class, as its index matches the `prd*_*_linux` wildcard pattern with a `critical` priority mapped to `gold`:



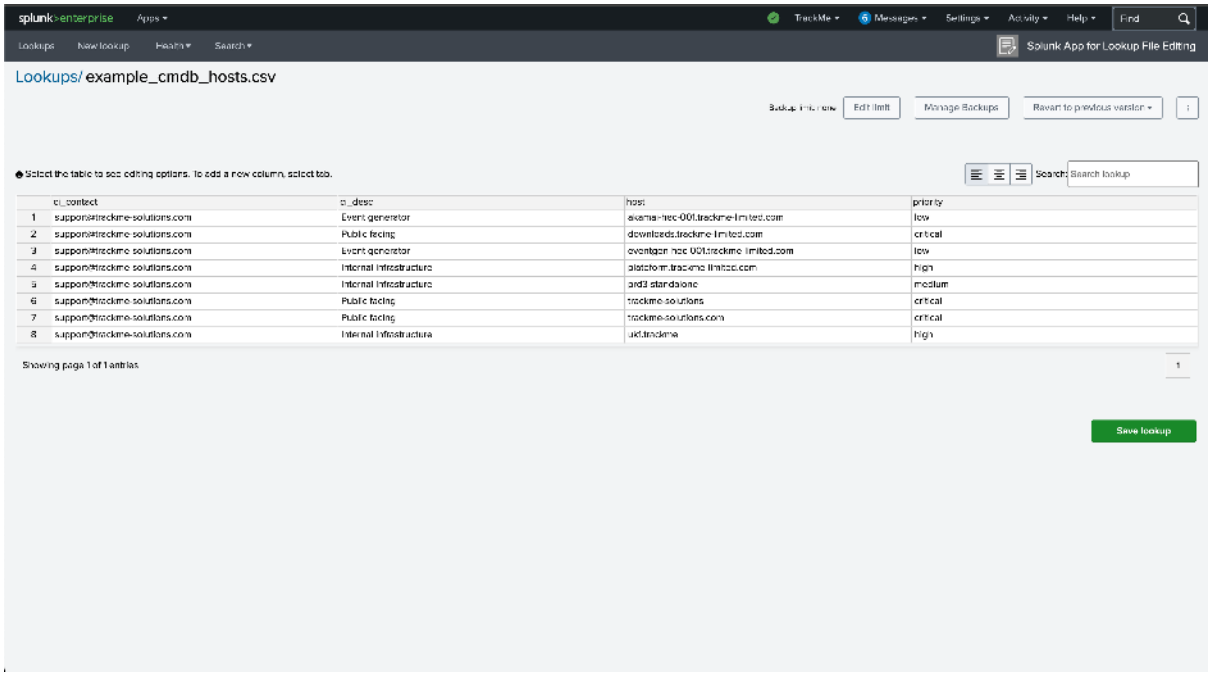
Data host monitoring (DHM) example

In this example, we demonstrate the same workflow for data host monitoring, using a host-based lookup `example_cmdb_hosts.csv`. The key differences from the DSM example are:

- The lookup maps **hosts** (by FQDN) instead of indexes
- The field mapping uses `host` → `alias` (the entity alias field in DHM)
- The match mode is **Exact** (hosts have specific FQDNs, no wildcards needed)
- The **tags field** uses the `ci_desc` column (leveraging an existing CMDB description field as tags)

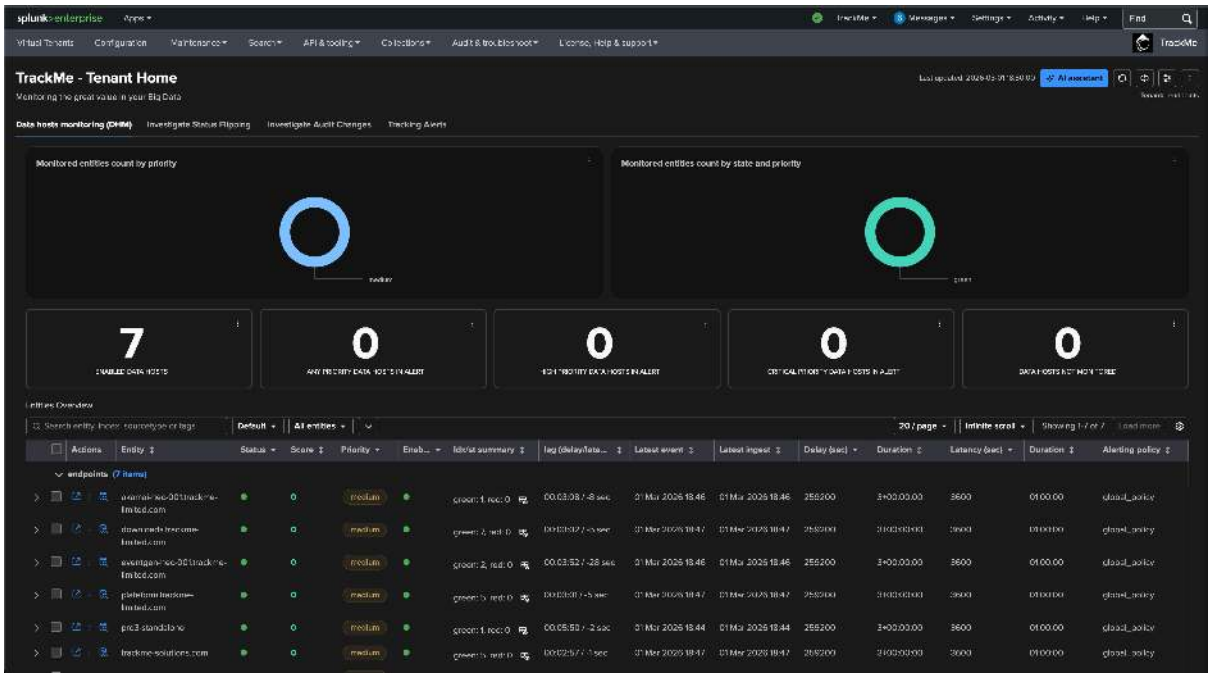
DHM - The lookup file

The lookup file `example_cmdb_hosts.csv` is uploaded to Splunk and contains 8 rows with columns `ci_contact`, `ci_desc`, `host`, and `priority`:



DHM - Initial state

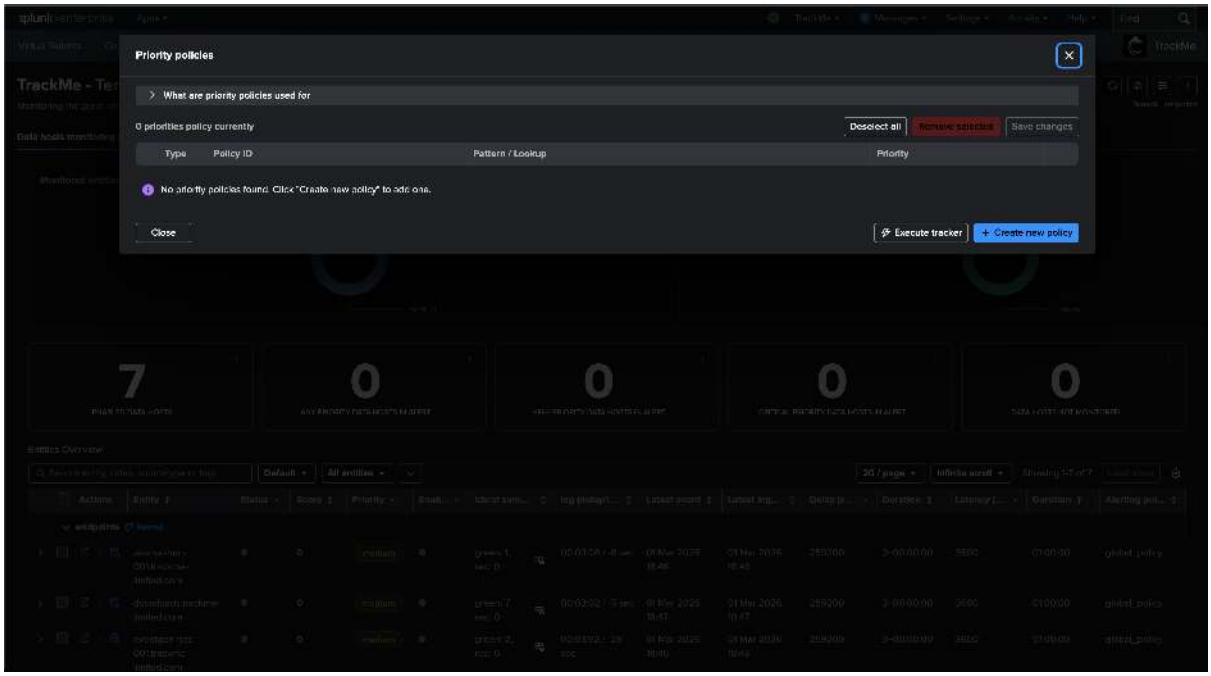
Before creating any policies, all 7 data host entities are at the default priority level (medium):

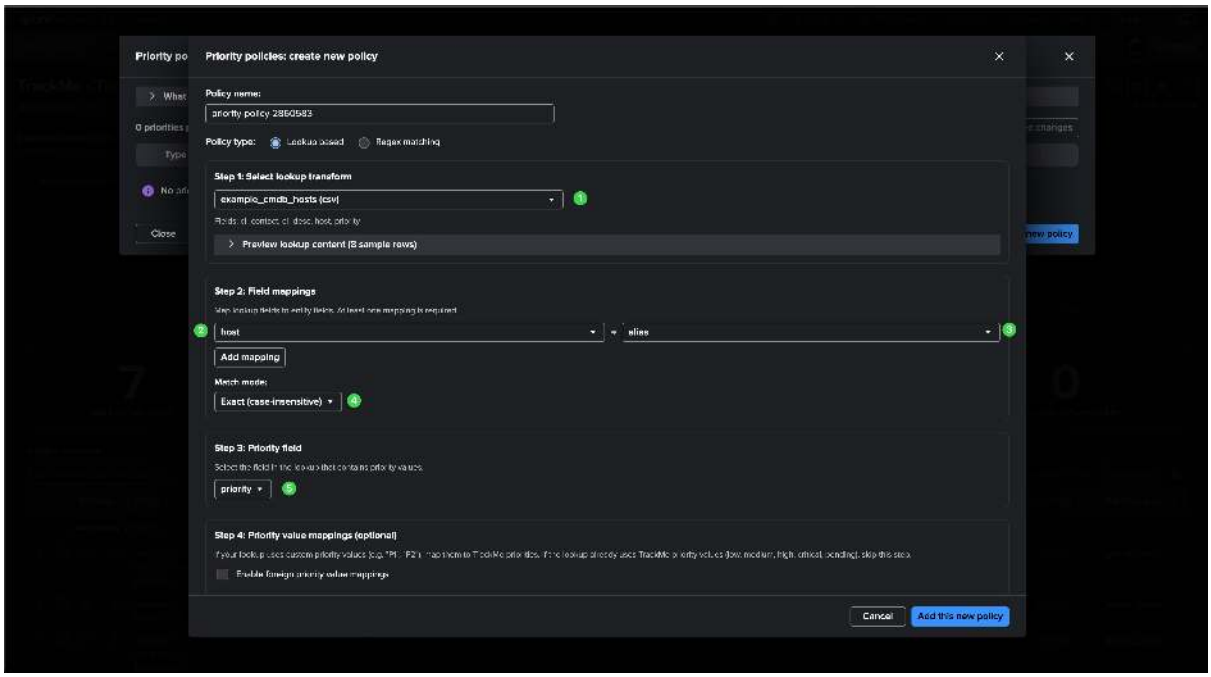


DHM - Step 1: Create a priority policy

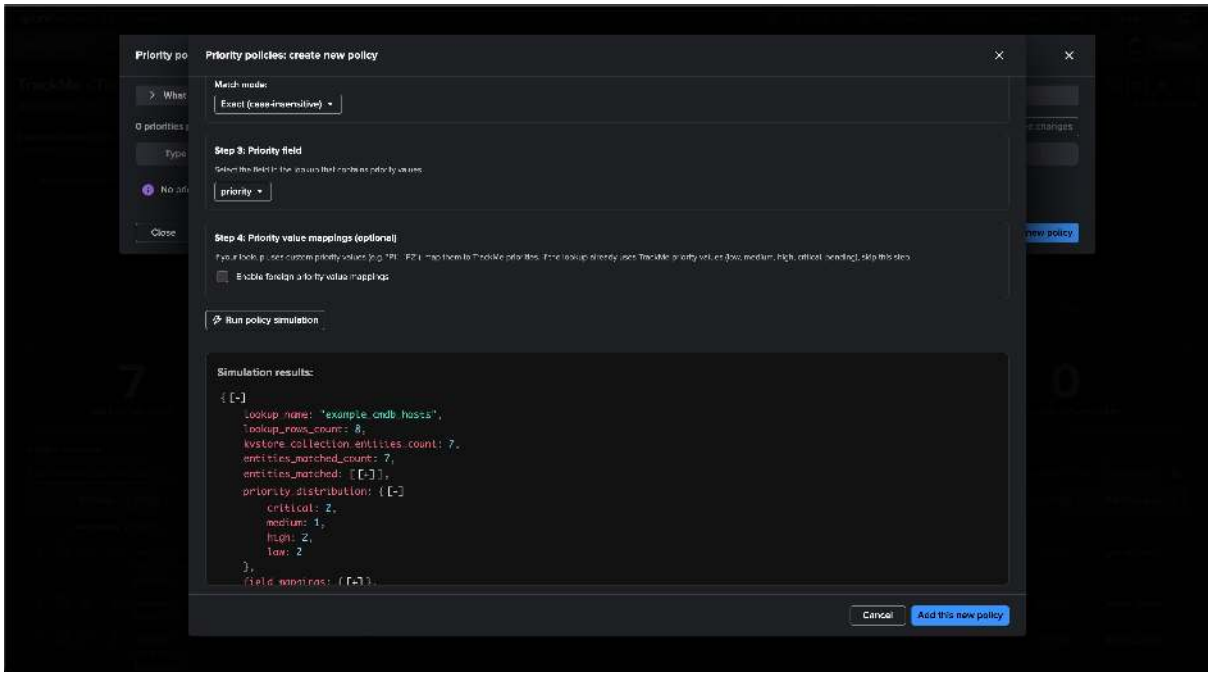
Access the priority policies screen and create a lookup-based policy:

- Select the lookup transform `example_cmdb_hosts (csv)`
- Configure the **field mapping**: `host` → `alias`, with **Exact (case-insensitive)** match mode
- Select `priority` as the **priority field**

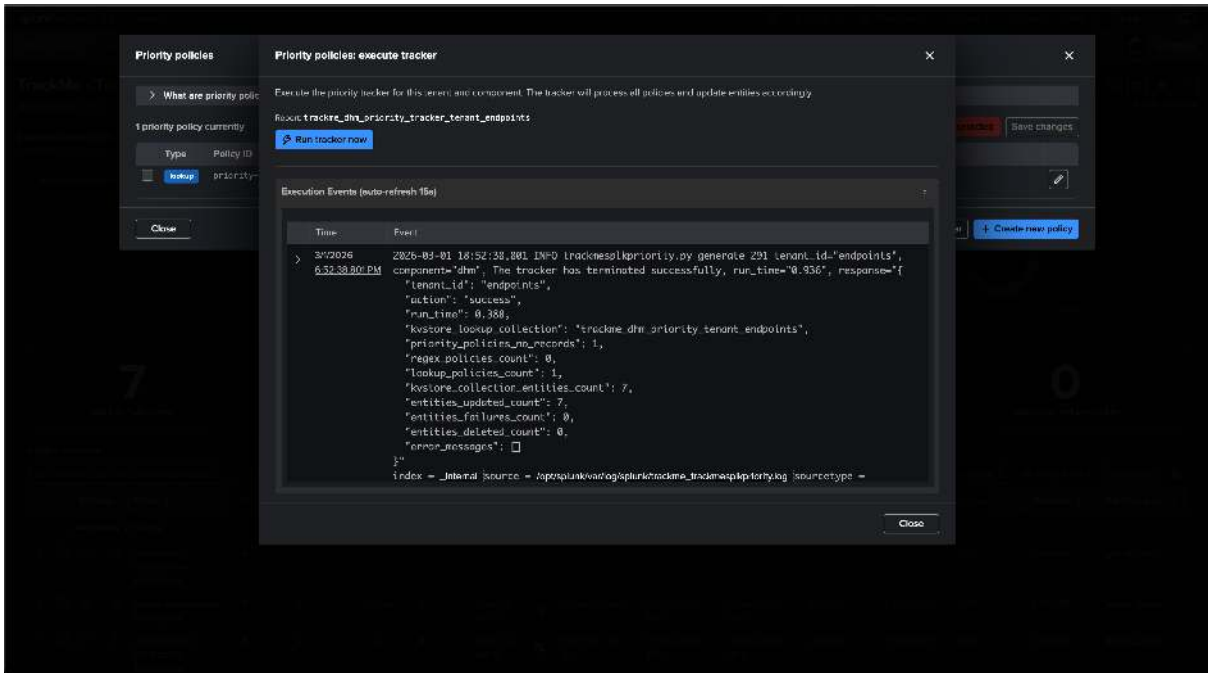




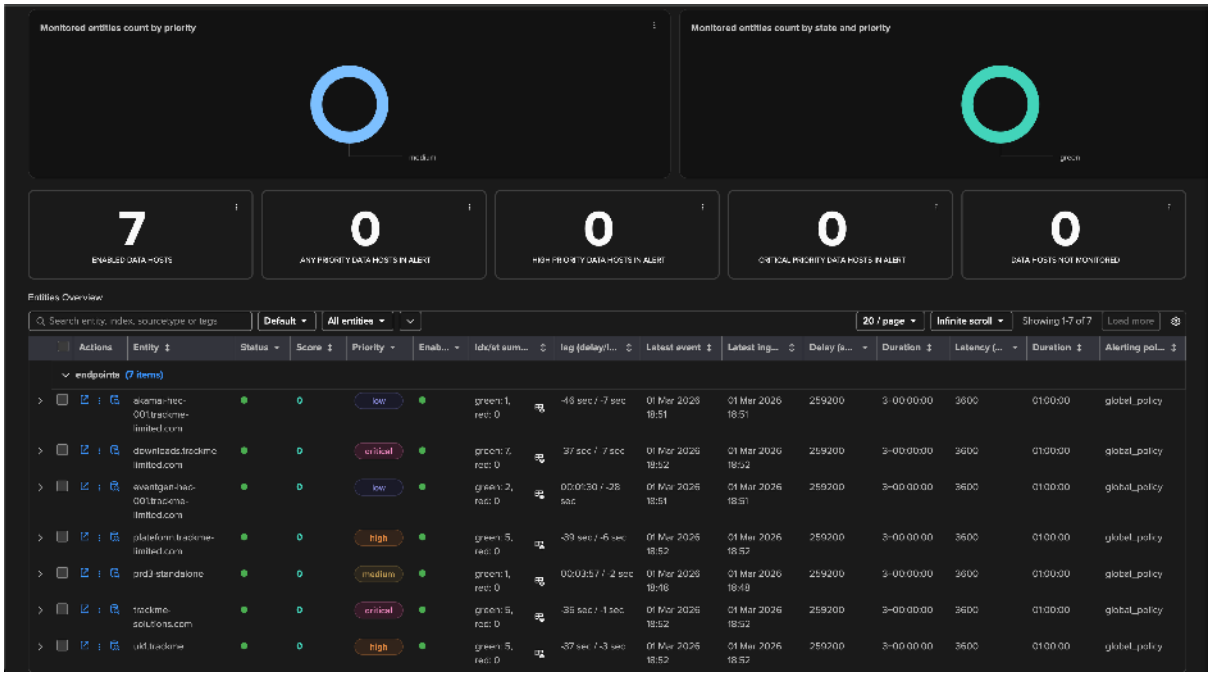
Run the simulation — all 7 entities are matched with the correct priority distribution (critical: 2, high: 2, medium: 1, low: 2):

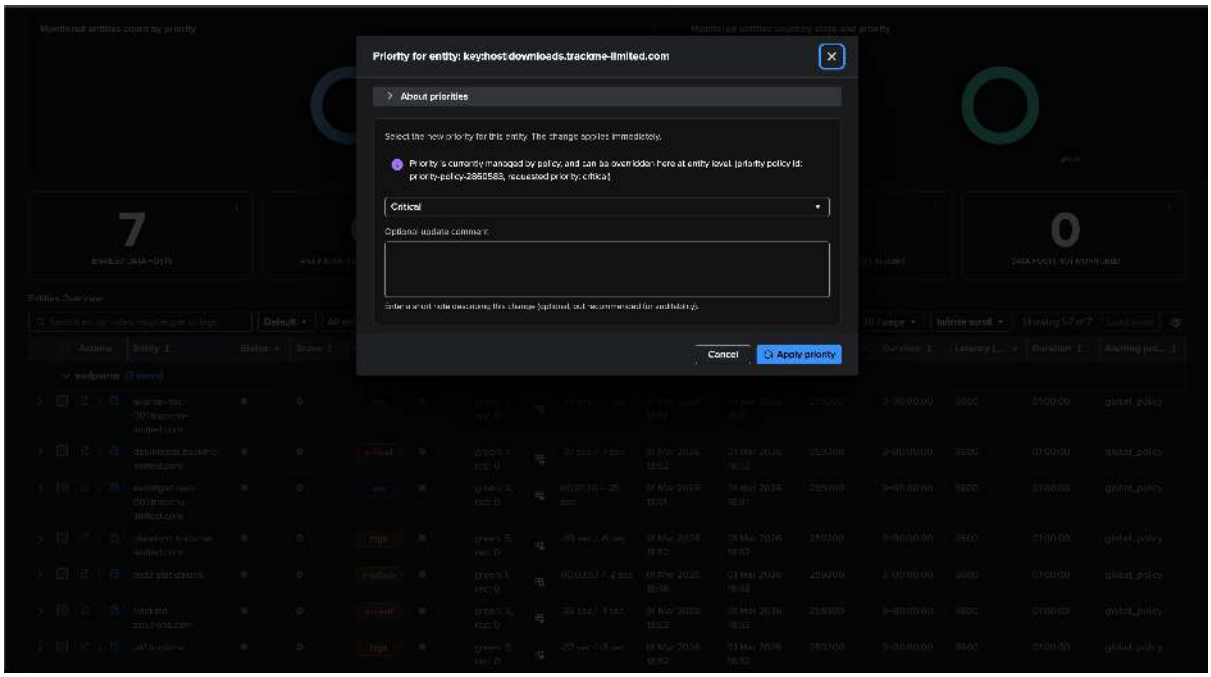


Execute the tracker to apply the priority assignments:



Result: Entities are now assigned the correct priority levels. The entity-level priority dialog shows the priority is managed by the policy and can be overridden at entity level:

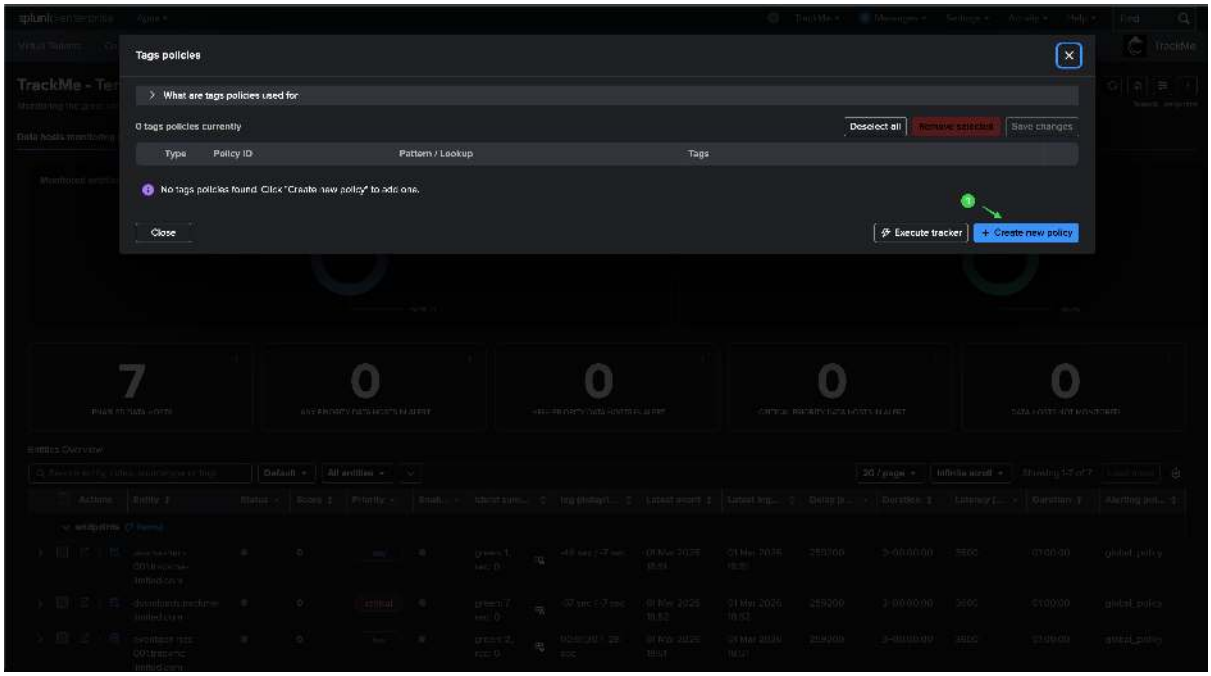


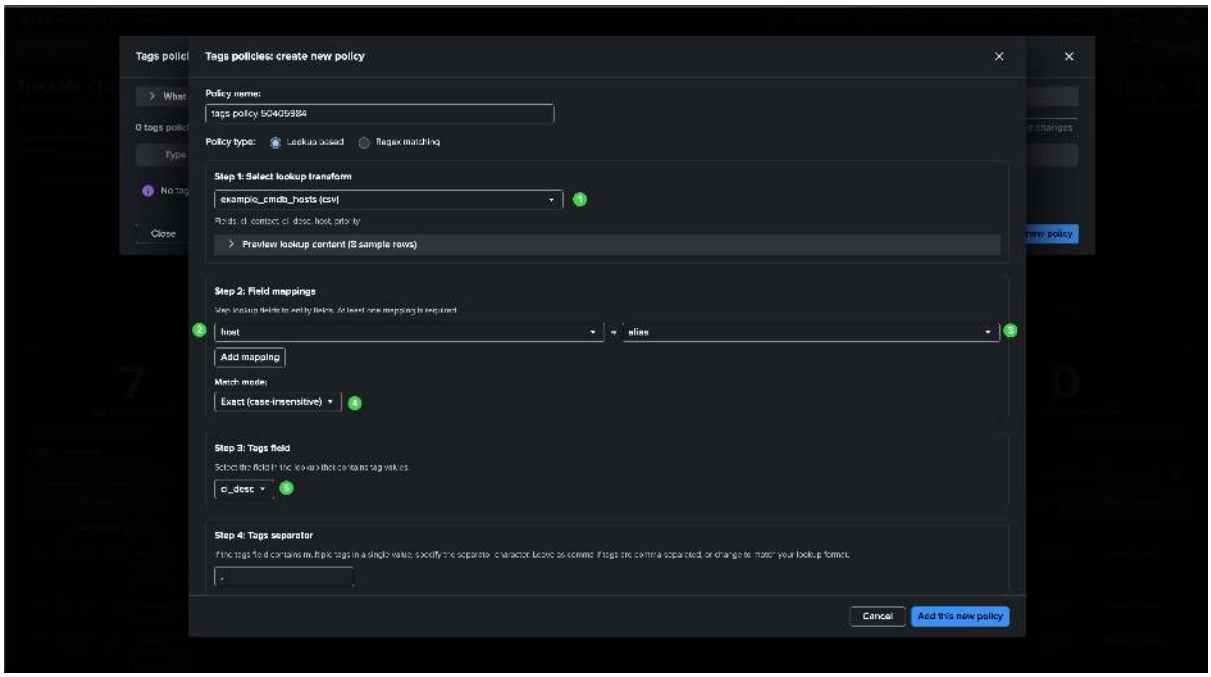


DHM - Step 2: Create a tags policy

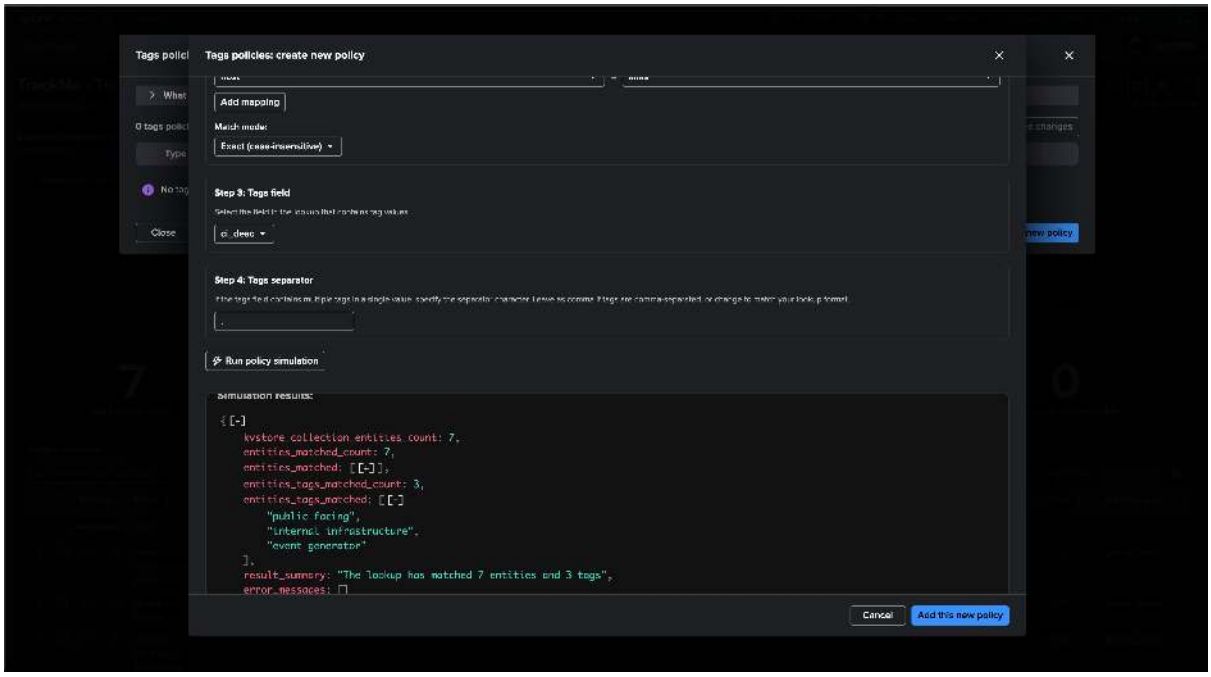
Create a lookup-based tags policy using the same lookup transform:

- Field mapping: `host` → `alias`, with **Exact** match mode
- Tags field: `ci_desc` — in this example, we leverage the CMDB description column as the source of tags (e.g., "Public facing", "Internal infrastructure", "Event generator")
- Tags separator: `,` (comma)

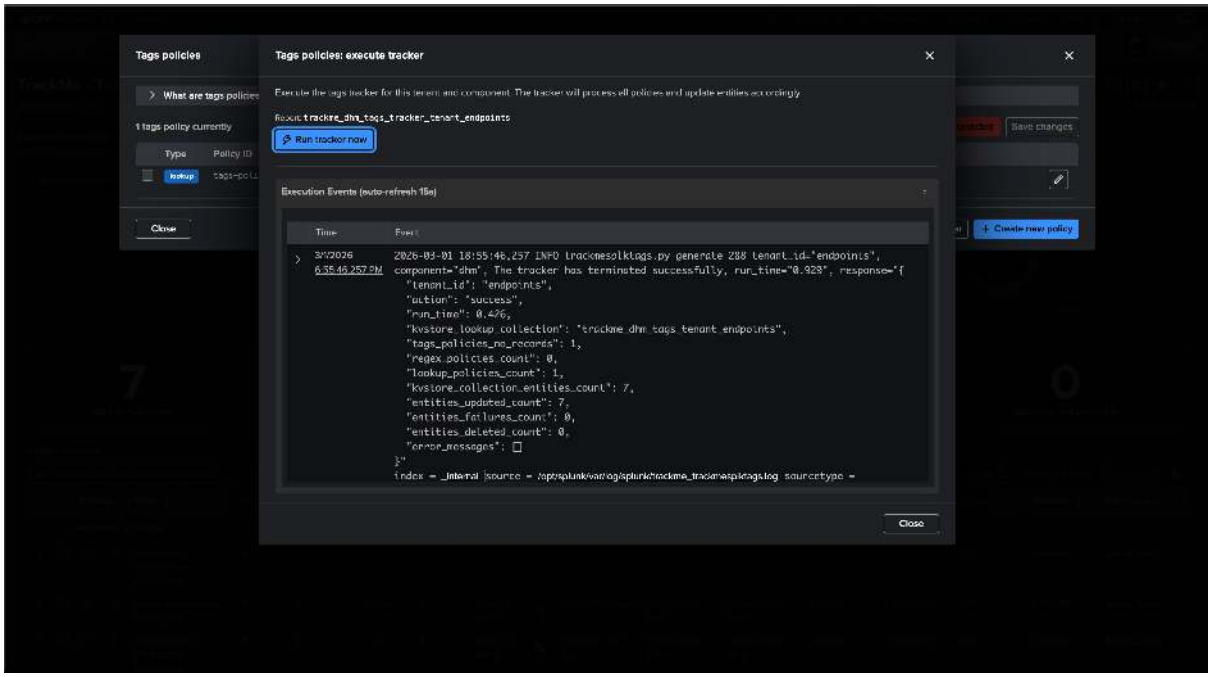




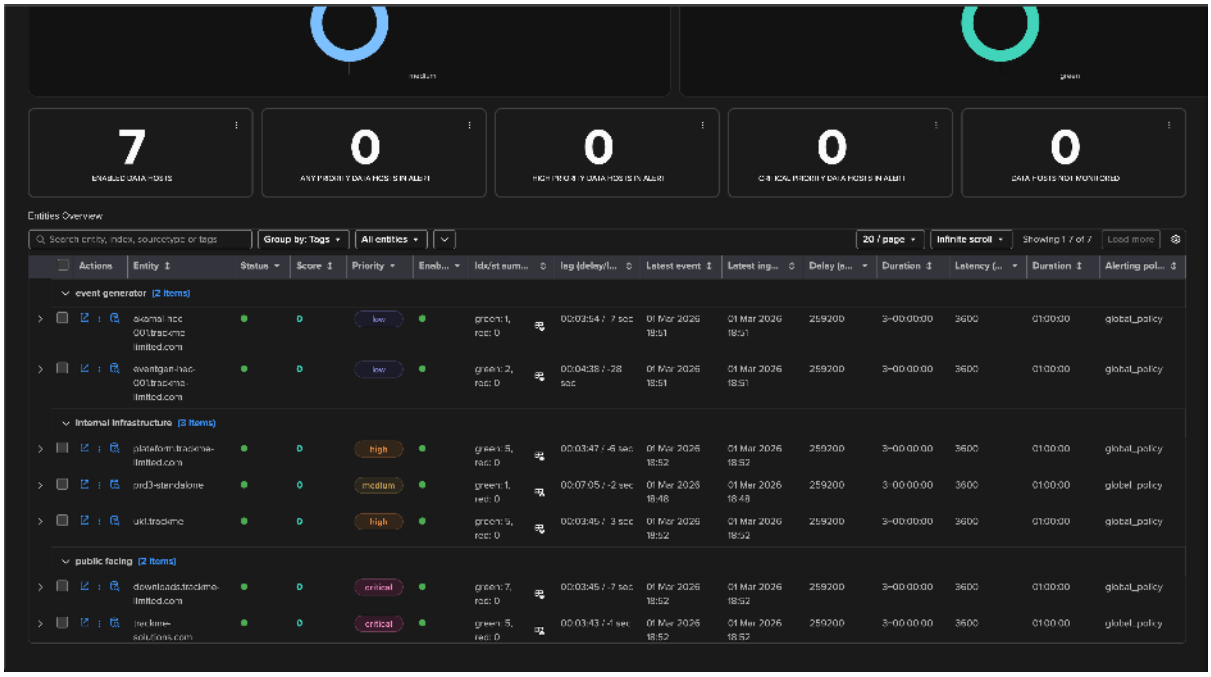
Run the simulation — 7 entities matched, 3 distinct tags discovered (Public facing, Internal infrastructure, Event generator):

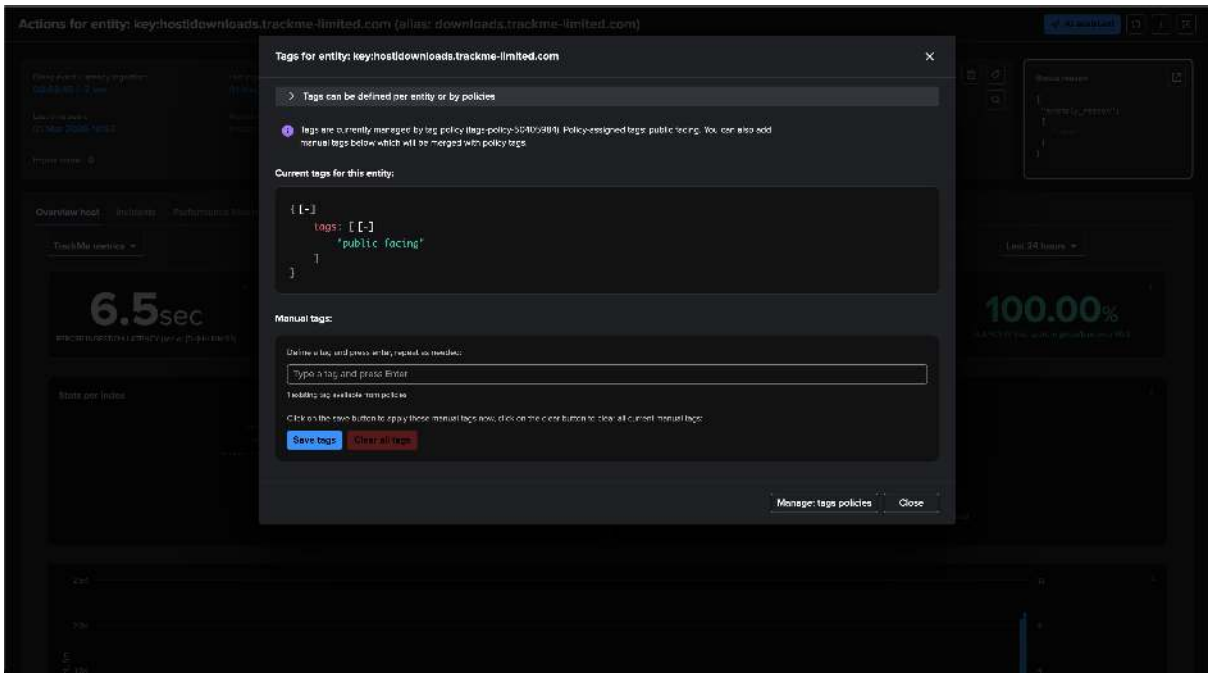


Execute the tracker to apply the tag assignments:



Result: Entities are now tagged and can be grouped by tags in the Tenant Home. The entity tags dialog confirms the tag assignment by the policy:

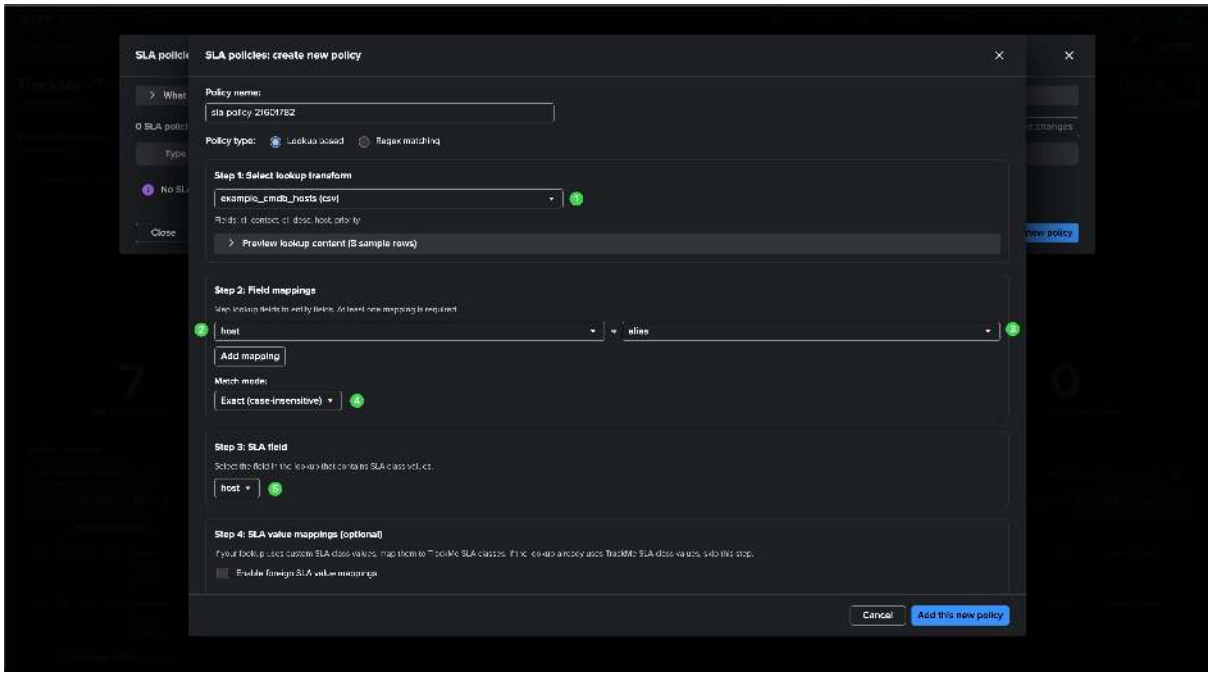


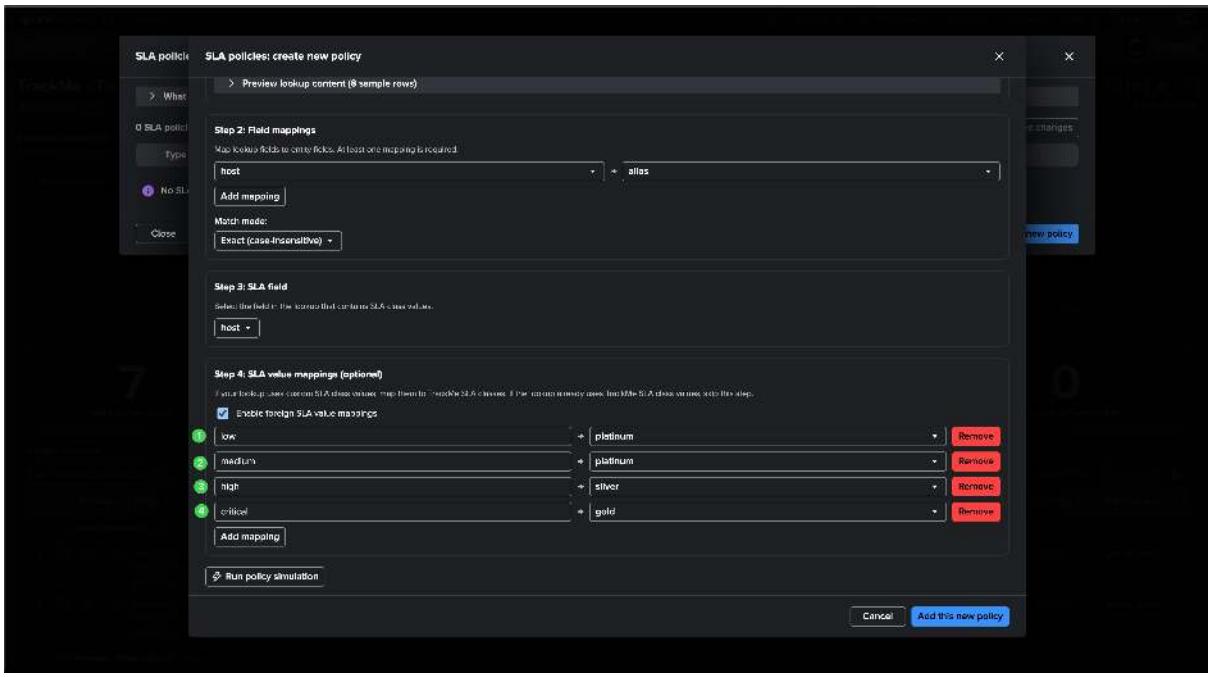


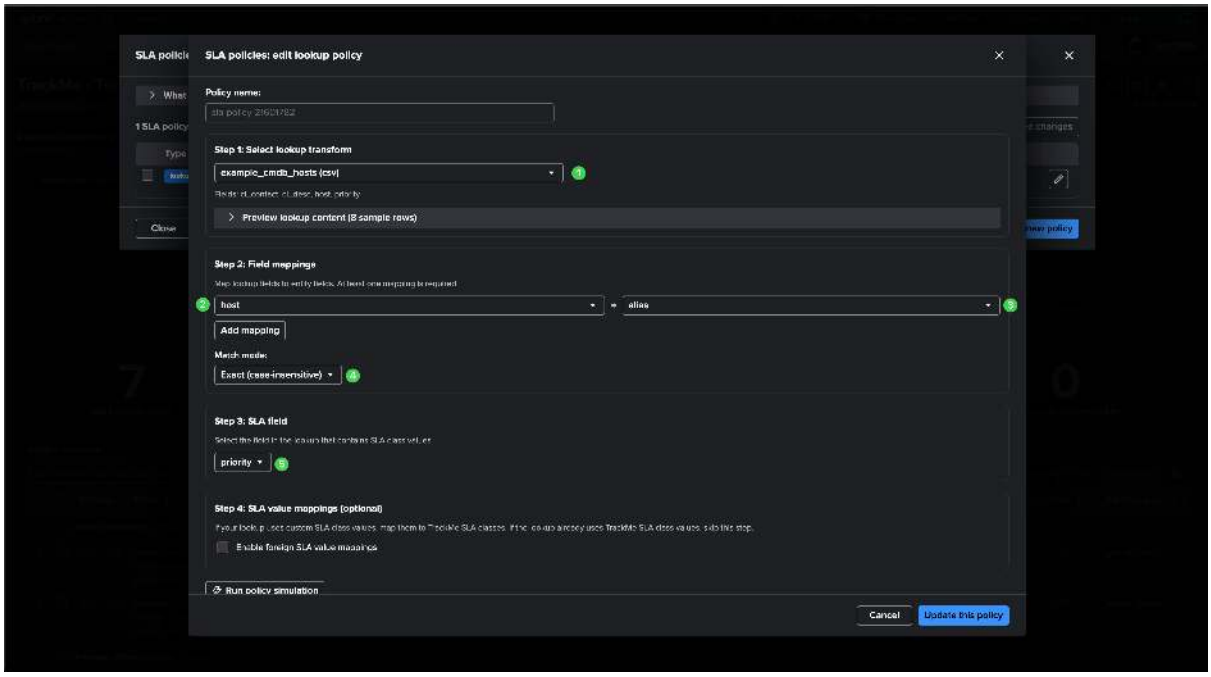
DHM - Step 3: Create an SLA policy

Create a lookup-based SLA policy using the same lookup transform:

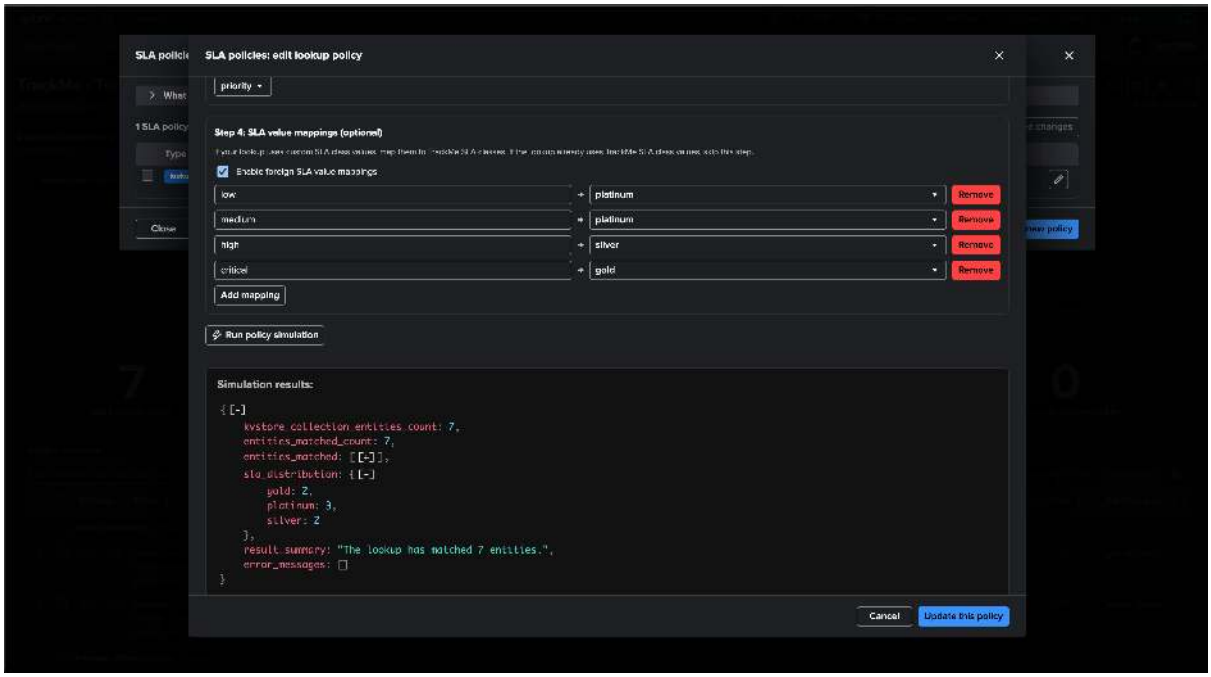
- Field mapping: `host` → `alias`, with **Exact** match mode
- SLA field: `priority` — as in the DSM example, we reuse the priority column and translate values via mappings
- Enable **SLA value mappings**: `low` → `platinum`, `medium` → `platinum`, `high` → `silver`, `critical` → `gold`


SLA policies: create new policy
Policy name:
sla policy 21601782
Policy type:
Lookup based
Regex matching
Step 1: Select lookup transform
example_cmdb_hosts (csv)
Preview lookup content (8 sample rows)
Step 2: Field mappings
host
alias
Add mapping
Match mode:
Exact (case-insensitive)
Step 3: SLA field
host
Step 4: SLA value mappings (optional)
Cancel
Add this new policy

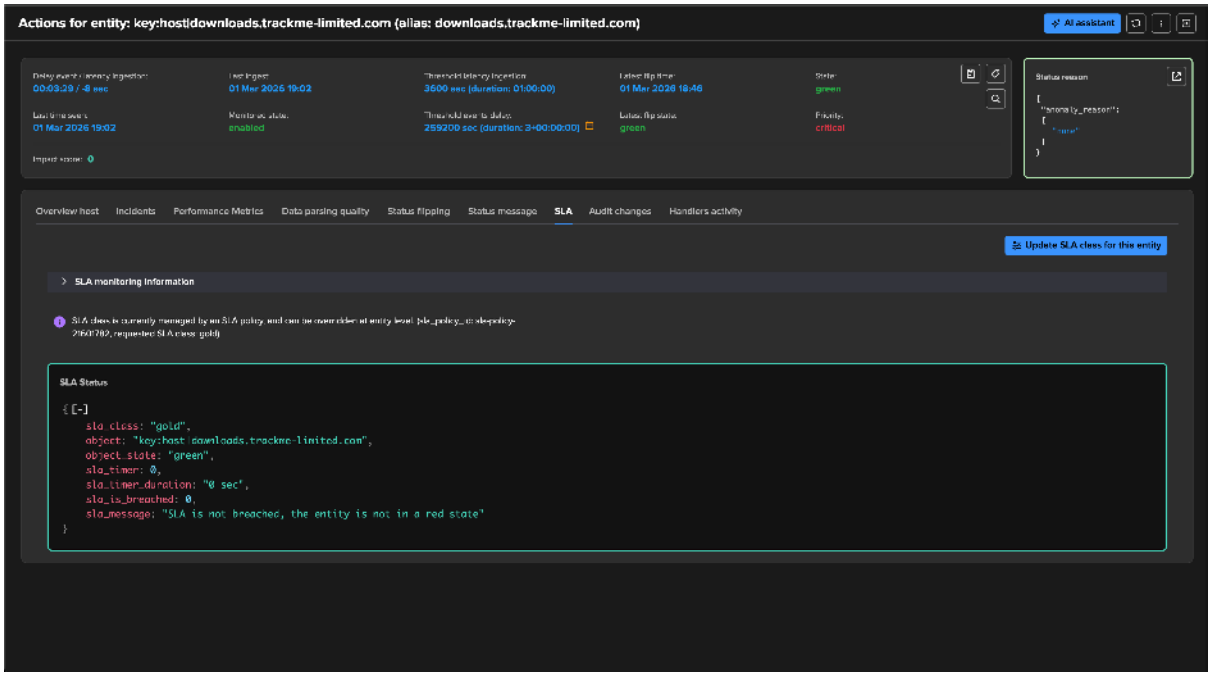

SLA policies: create new policy
Preview lookup content (8 sample rows)
Step 2: Field mappings
host
alias
Add mapping
Match mode:
Exact (case-insensitive)
Step 3: SLA field
host
Step 4: SLA value mappings (optional)
Enable foreign SLA value mappings
low
platinum
Remove
medium
platinum
Remove
high
silver
Remove
critical
gold
Remove
Add mapping
Run policy simulation
Cancel
Add this new policy



Run the simulation — 7 entities matched with SLA distribution (gold: 2, platinum: 3, silver: 2):



Result: Opening an entity's SLA tab confirms the SLA class assignment. In this example, `downloads.trackme-limited.com` is assigned the `gold` SLA class (critical priority mapped to gold):



Summary and best practices

Key takeaways

- **Single source of truth:** Maintain one lookup file per scope (e.g., one for indexes, one for hosts) and let TrackMe derive priority, tags, and SLA from it.
- **Wildcard matching:** Use wildcard match mode with patterns like `prd*_*_linux` to match complex naming conventions without enumerating every possible entity.
- **Value mappings:** When your lookup doesn't use TrackMe's native values (e.g., custom priority levels or no dedicated SLA column), use value mappings to translate on the fly.
- **Simulate before applying:** Always run a simulation before adding a policy to verify the match count and value distribution.
- **Execute trackers:** After creating or modifying policies, execute the relevant tracker to apply changes immediately. Trackers also run on schedule, so any future lookup updates will be picked up automatically.
- **Tags separator:** When a single lookup cell contains multiple tags (e.g., `eventgen,os`), configure the tags separator so TrackMe splits them into individual tags.
- **Conflict resolution: If multiple policies match the same entity, TrackMe applies the following resolution rules:**
  - **Priority:** Highest priority level wins across all policies
  - **Tags:** Tags are merged (union) across all matching policies
  - **SLA:** Highest-rank SLA class wins across all policies

QUICK START - Cross-tenant domain views with Virtual Groups

Aggregate entities from multiple tenants into unified domain views — without any new monitoring infrastructure

- This quickstart guides you through the design and implementation of **Virtual Groups** in TrackMe — a powerful approach to creating cross-tenant, read-only aggregation views that appear alongside your regular

tenant cards in the Virtual Tenants grid.

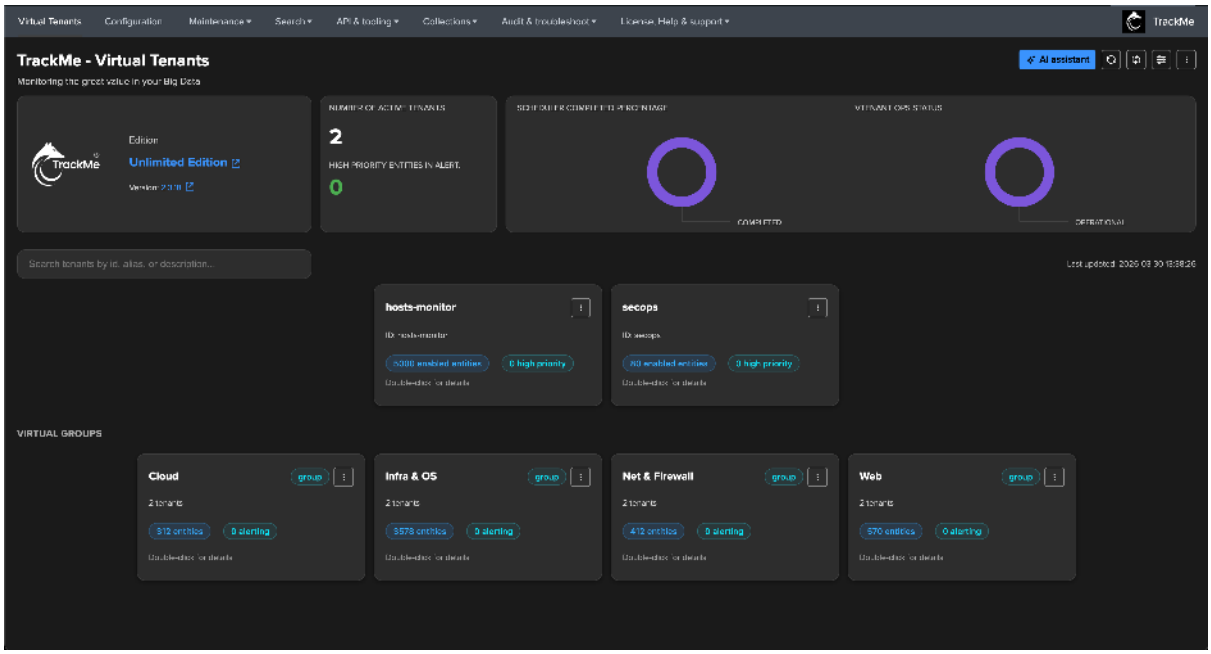
- The key idea: **tags policies** distribute your entities across logical domains (cloud, infra, network…), and **Virtual Groups** surface exactly those domains as first-class cards — spanning multiple tenants at once.
- This quickstart is built around a concrete example with **two tenants** and **~10,000 entities**, culminating in **four Virtual Group cards** each scoped to a distinct business domain.
- **Prerequisites:** An existing TrackMe deployment with at least one Virtual Tenant, discovered entities, and a lookup file (CSV or KVstore) containing knowledge about your environment.

Hint

For detailed reference documentation on tags policies, refer to *Accessing, defining and managing tags policies*.

The goal: what we are building

Before diving into the steps, here is what the end result looks like. Two tenants — `hosts-monitor` (Data Host Monitoring) and `secops` (Data Source Monitoring) — hold roughly 10,000 entities between them. After this quickstart, the Virtual Tenants grid will show **four Virtual Group cards**, each aggregating entities from both tenants, scoped to a specific business domain:



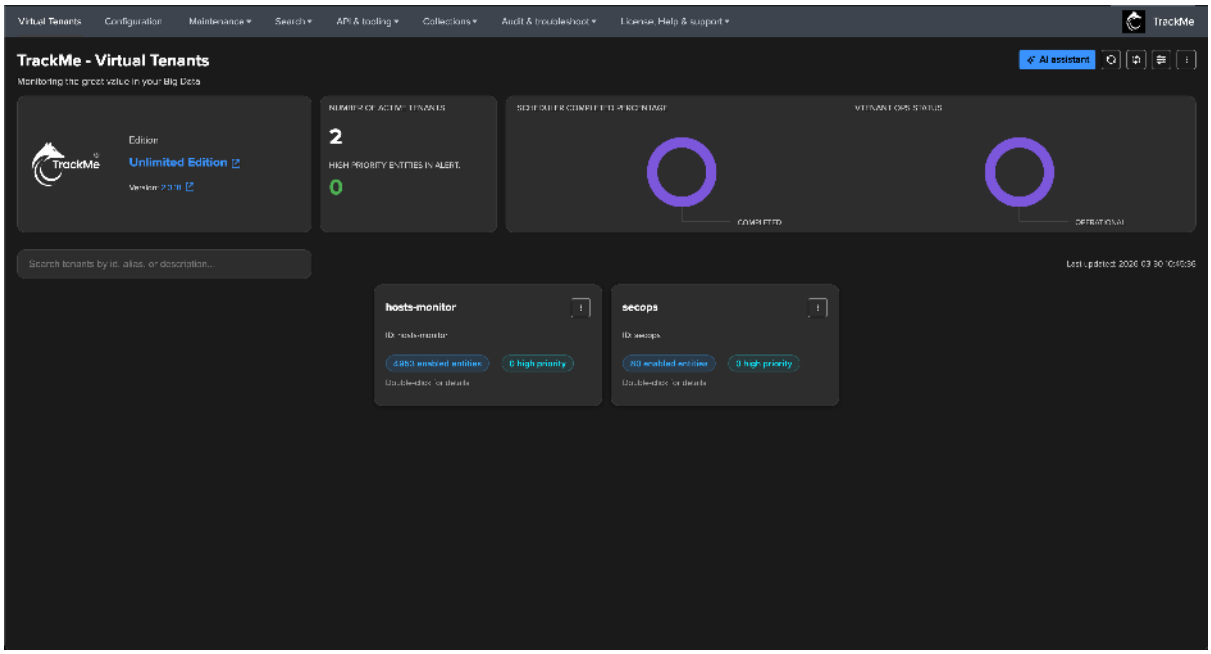
Each group card shows:

- The **number of tenants** contributing entities to the group
- The **total entity count** (aggregated across all tenants and components in scope)
- The **number of alerting entities** (red/orange state)
- A context menu to open the group overview, refresh, edit, or delete

The design relies on a single, elegant building block: **tags**. Every entity is tagged with its business domain by a lookup-based tags policy. Virtual Groups then use a free-form filter expression (`tags="cloud"`, `tags="infra" OR tags="os"`, etc.) to pick up exactly the right entities — automatically, from any tenant.

Our scenario

We start with two tenants in the Virtual Tenants grid, no Virtual Groups yet:



- **hosts-monitor** — Data Host Monitoring (DHM) tenant. Monitors ~5,000 data hosts.
- **secops** — Data Source Monitoring (DSM) tenant. Monitors ~5,000 data source entities.

Both tenants share the same domain taxonomy: each entity belongs to one or more of these domains — `cloud`, `infra`, `os`, `net`, `firewall`, `syslog`, `web`, `identity`. This knowledge lives in a single lookup file: `example_cmdb_indexes.csv`, which maps index patterns (for DSM) and host patterns (for DHM) to tags.

The plan:

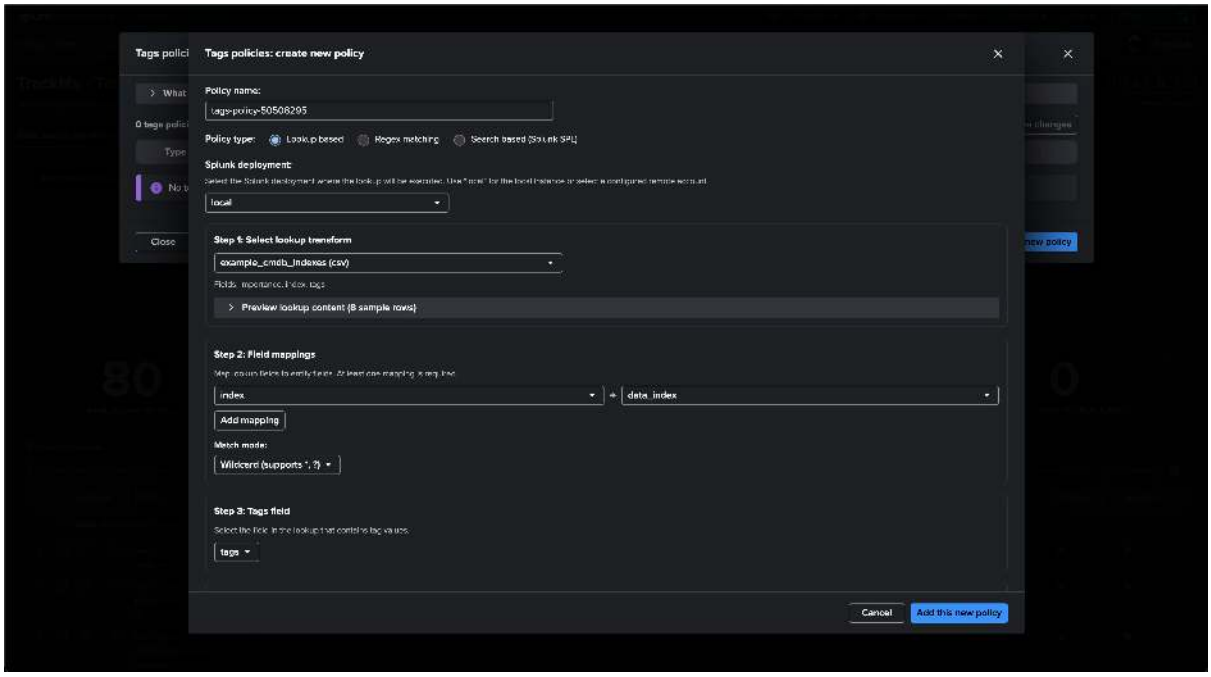
1. Create a **lookup-based tags policy** on `hosts-monitor` (DHM) to tag all data hosts
2. Create a **lookup-based tags policy** on `secops` (DSM) to tag all data sources
3. Create one **Virtual Group** per domain, referencing both tenants

Step 1 — Tag entities in hosts-monitor (DHM)

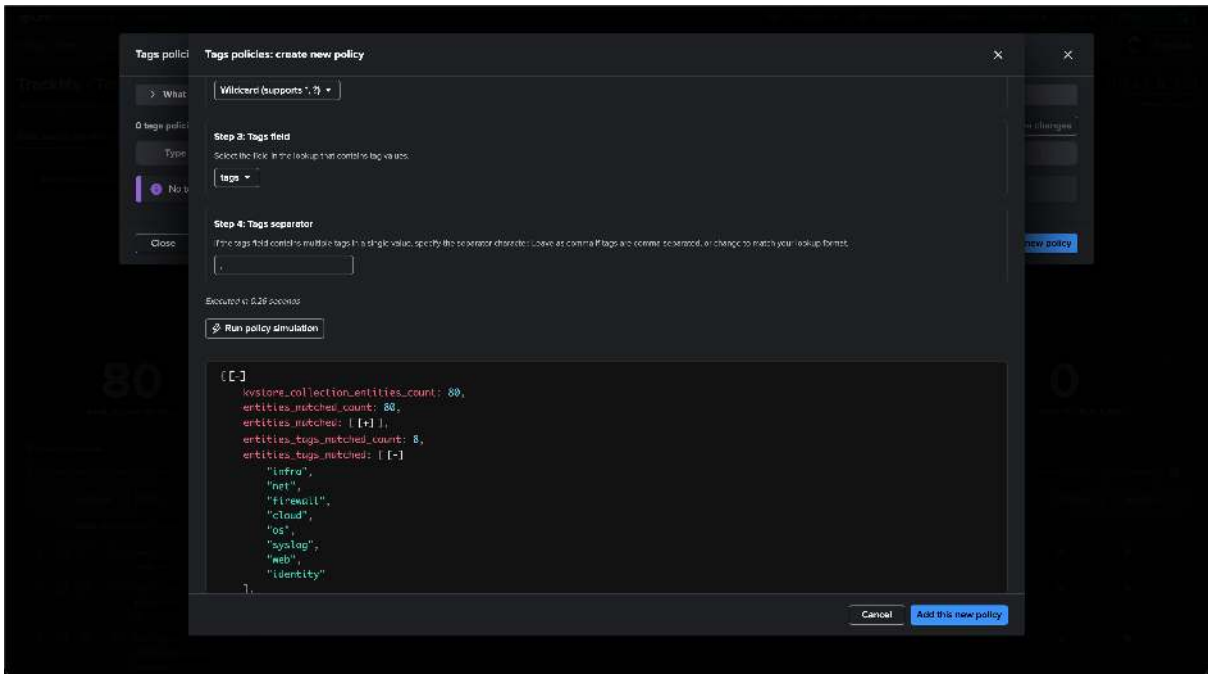
From the `hosts-monitor` Tenant Home, open **Actions → Manage tags policies** and create a new **Lookup based** policy.

Configure the policy wizard:

- **Step 1:** Select the lookup transform `example_cmdb_indexes (csv)`
- **Step 2:** Field mapping — map lookup field `index` to entity field `data_index`, match mode **Wildcard**
- **Step 3:** Tags field — select `tags`



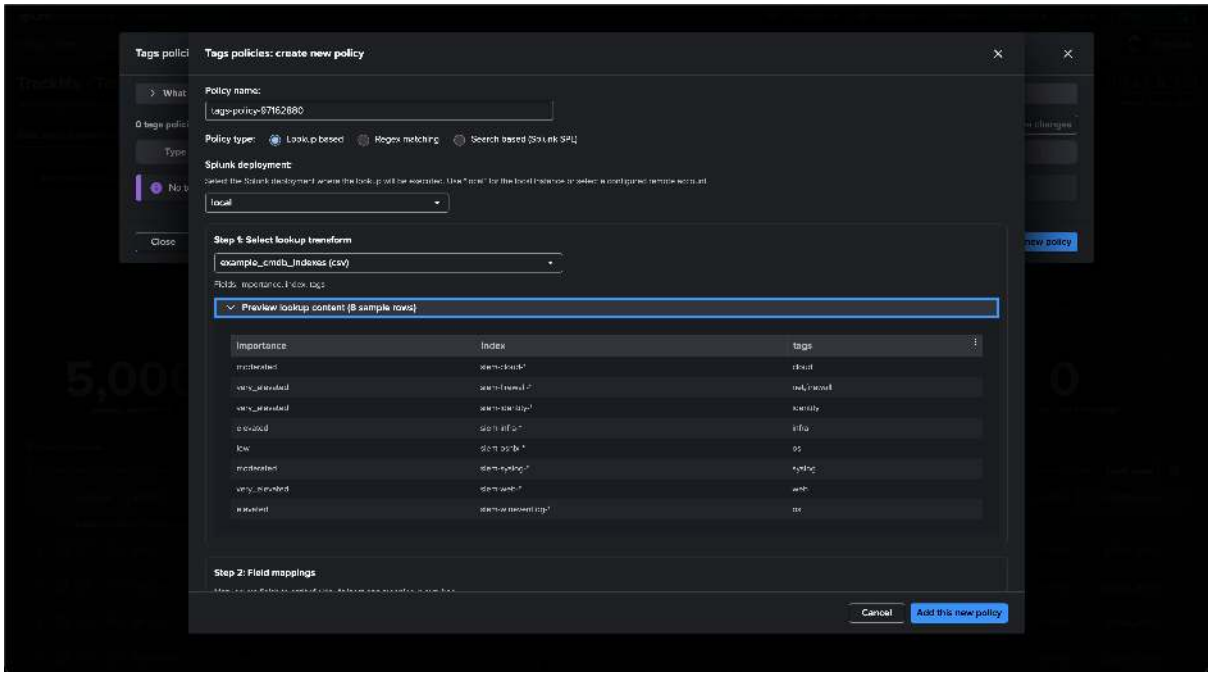
Run the **policy simulation** before saving. The simulation scans the entire entity collection and previews which entities would be matched and what tags would be assigned. For `hosts-monitor`, 80 entities are matched and 8 distinct tags are discovered (`infra`, `net`, `firewall`, `cloud`, `os`, `syslog`, `web`, `identity`):

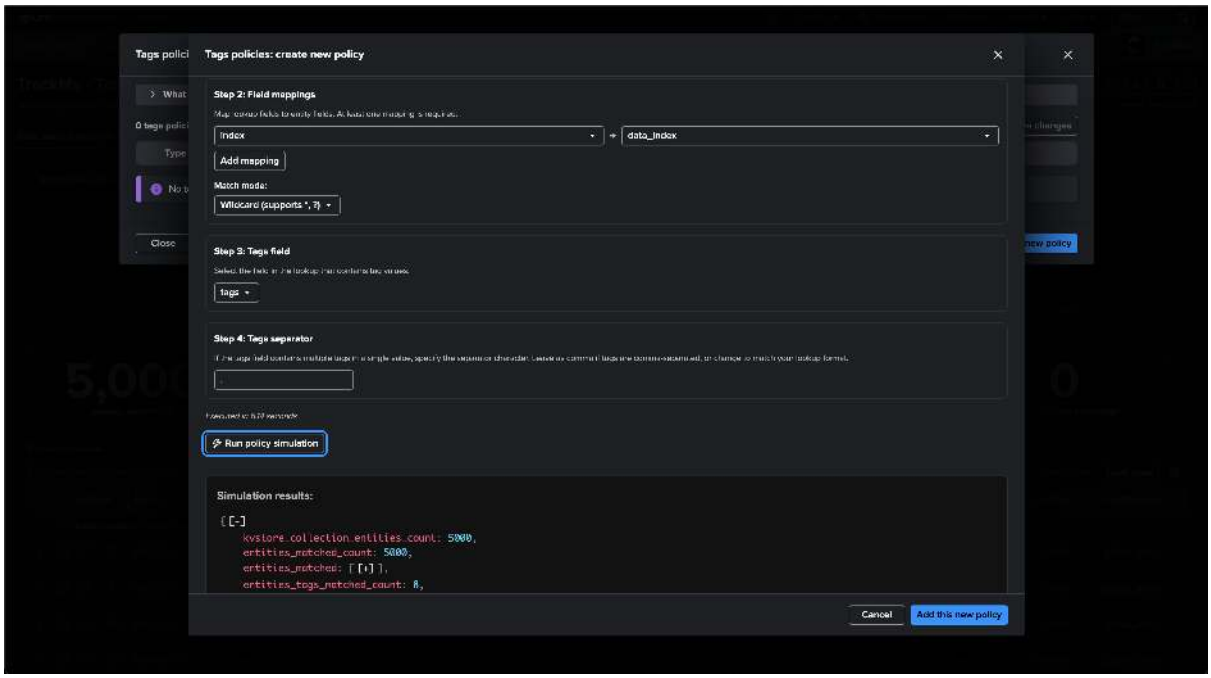


Click **Add this new policy**, then **Execute tracker** to apply the tag assignments immediately.

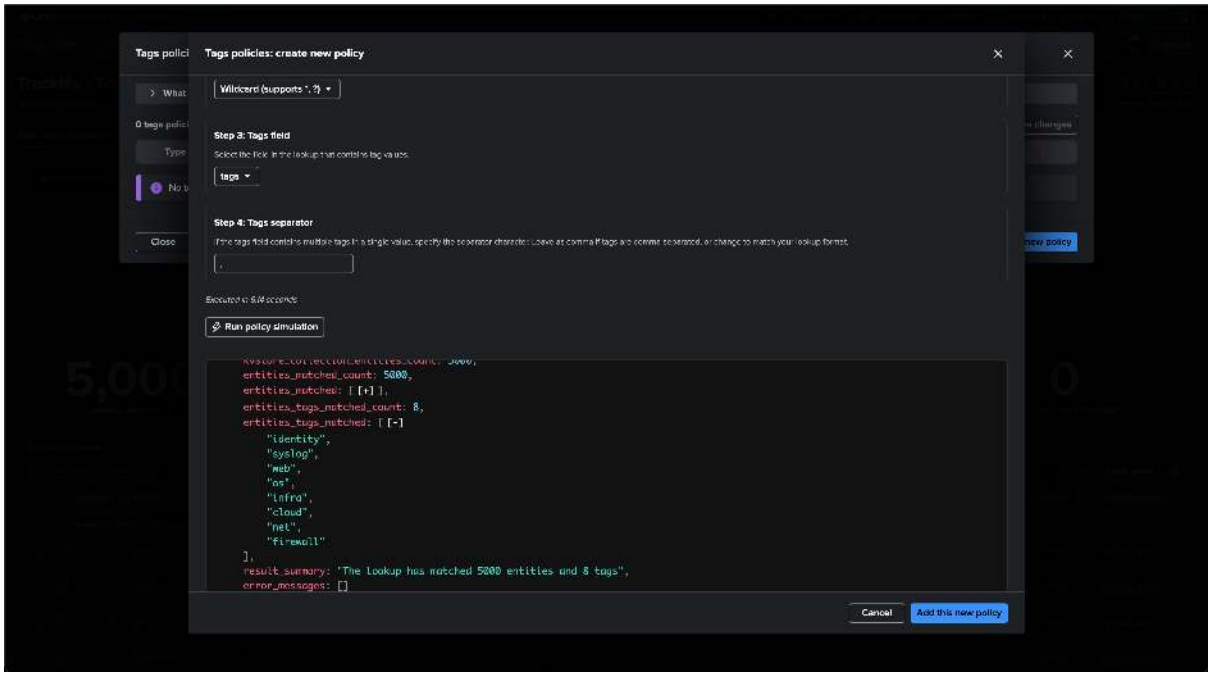
Step 2 — Tag entities in secops (DSM)

Repeat the same process on the `secops` tenant. The same lookup `example_cmdb_indexes (csv)` is used — the lookup preview confirms the mapping of index patterns to tags:





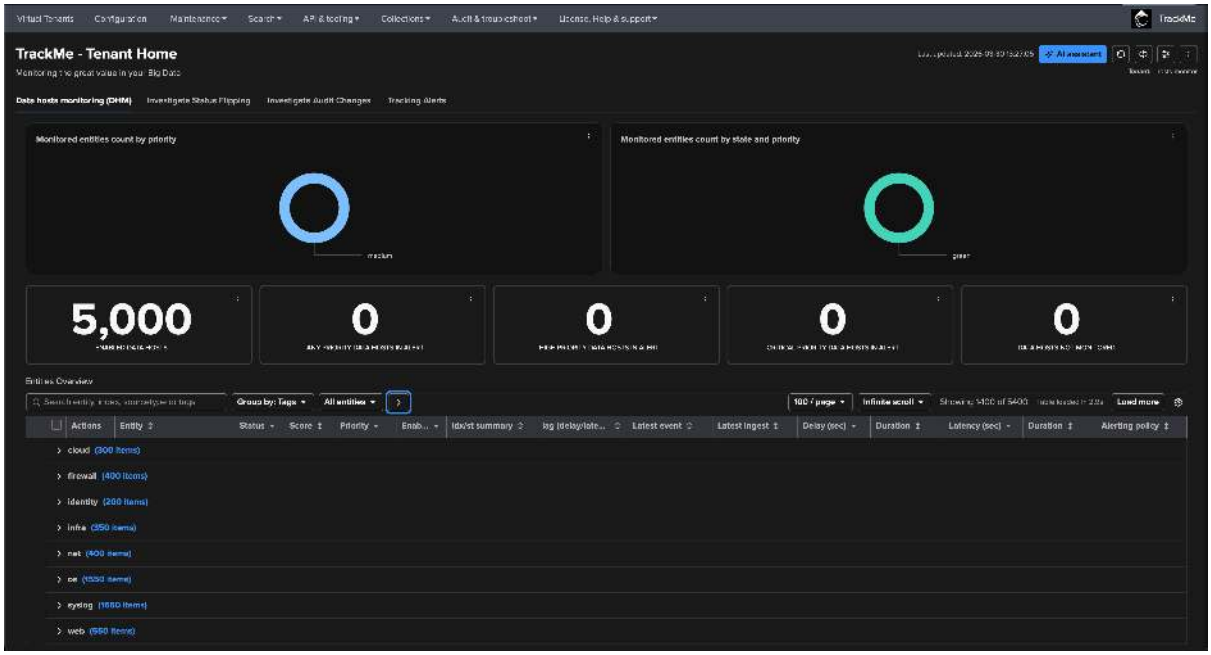
With 5,000 data sources in scope, the simulation confirms all entities are matched and 8 distinct tags are discovered. The result summary at the bottom of the simulation output confirms the coverage:



Execute the tracker. All data sources in `secops` are now tagged with their business domain.

Result: entities distributed across domains

With both tags policies applied, every entity across both tenants is now tagged. The Tenant Home **Group by Tags** view makes this immediately visible — entities are cleanly distributed across 8 domain buckets:



This is the foundation Virtual Groups build upon. The tags are the single source of truth — Virtual Groups simply reference them.

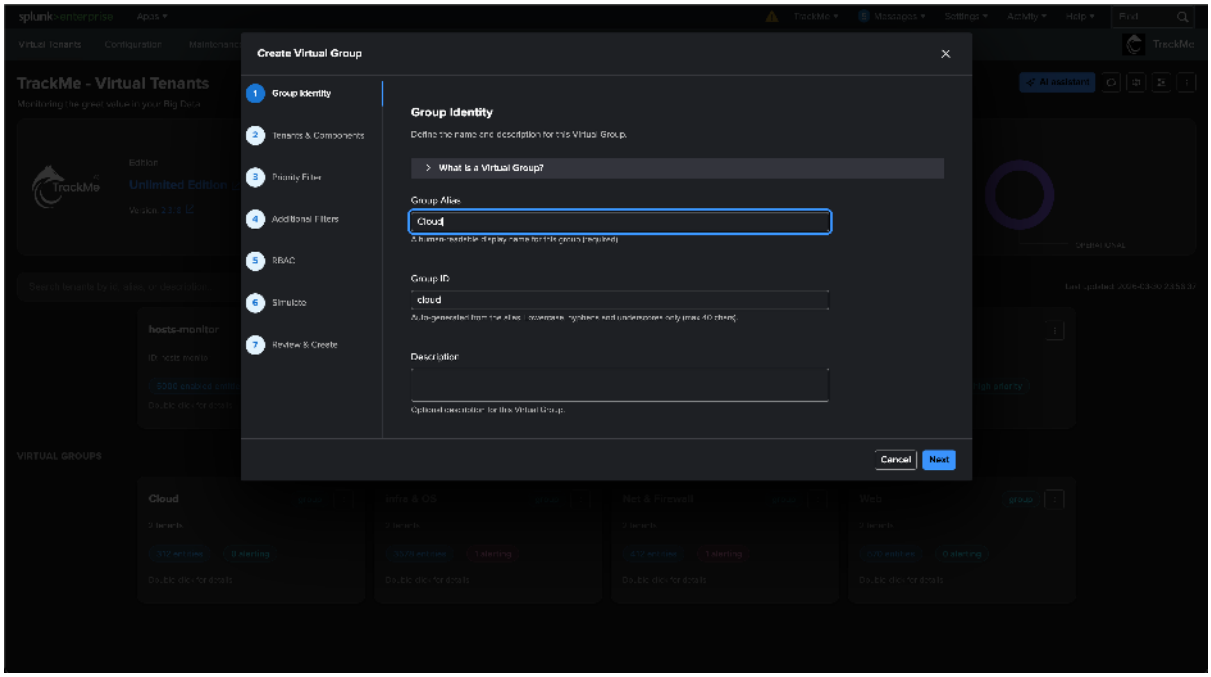
Step 3 — Create Virtual Groups

Now we create one Virtual Group per domain. Each group spans both tenants and both components (DHM + DSM), filtered by the relevant tag(s).

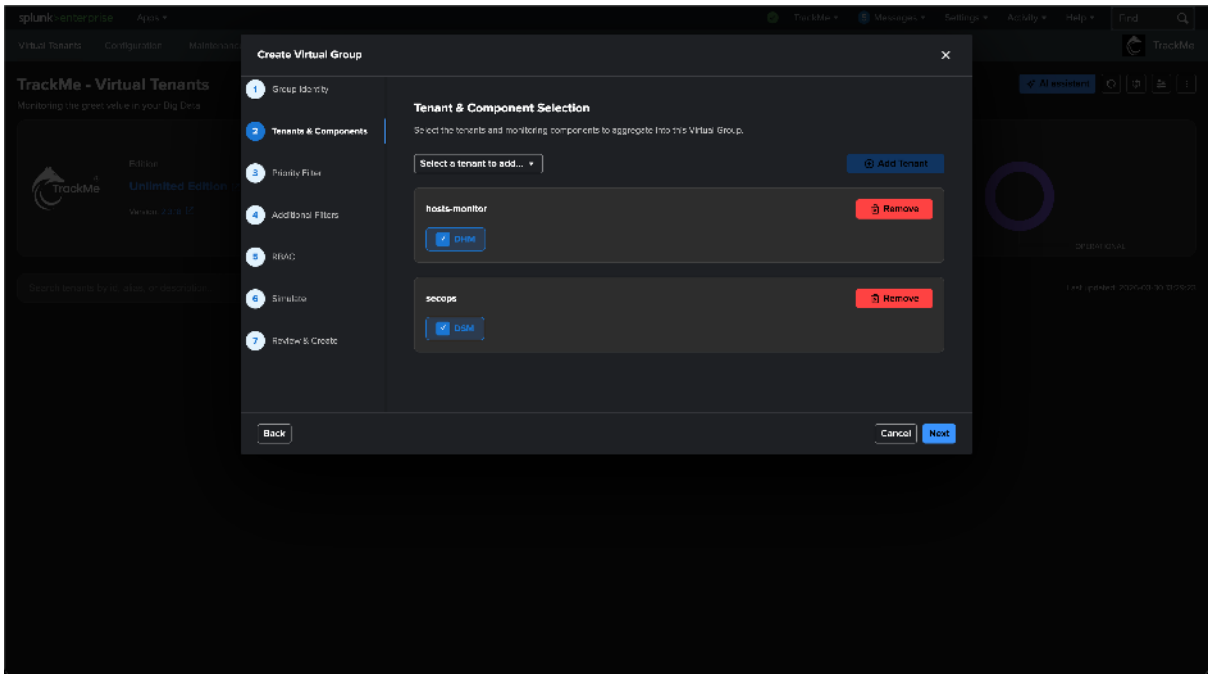
Creating the "Cloud" group

Open the **Administration** menu (top-right) and select **Create a virtual group**. The wizard opens.

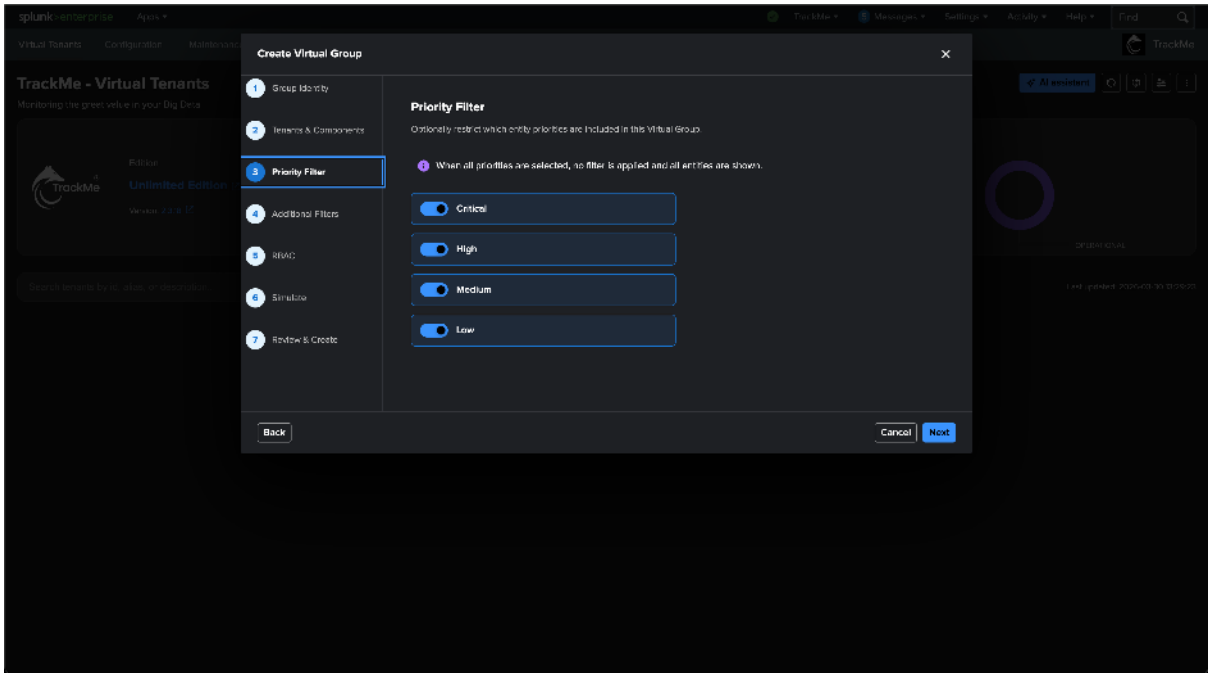
Step 1 — Group Identity: Give the group a human-readable alias and an auto-generated ID:



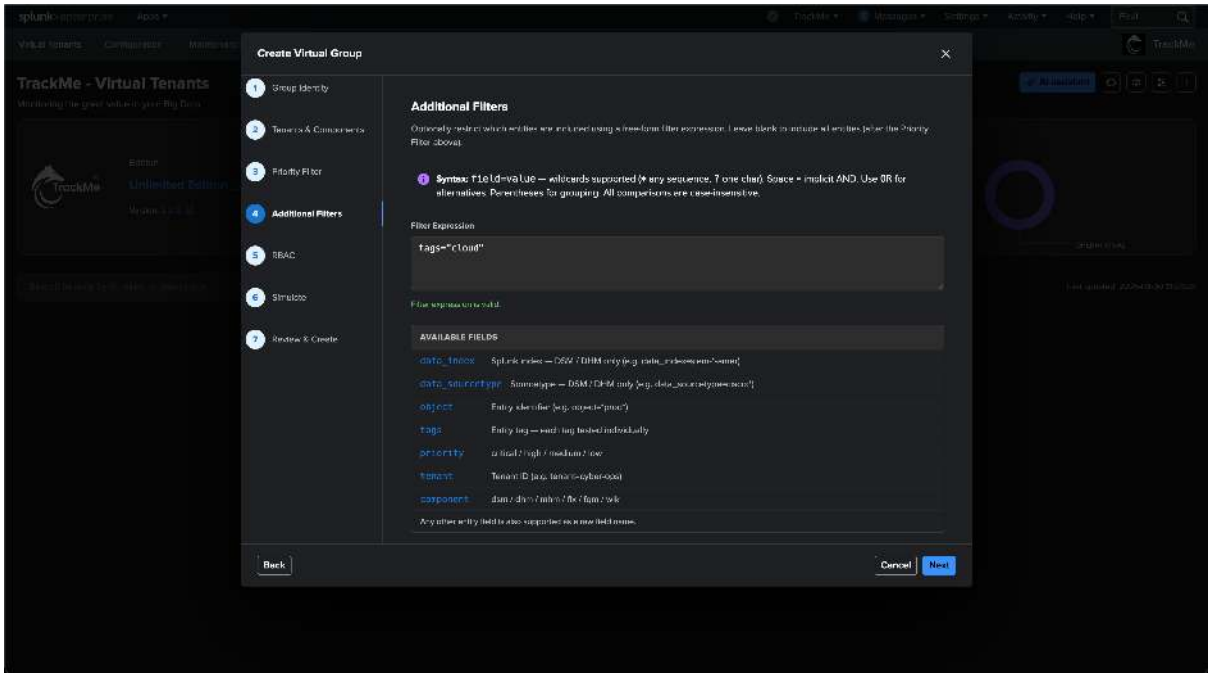
Step 2 — Tenants & Components: Add both tenants to the scope. For each tenant, select the relevant monitoring components. Here we include DHM from `hosts-monitor` and DSM from `secops`:



Step 3 — Priority Filter: Optionally restrict the group to entities of specific priority levels. Leaving all priorities enabled means all entities matching the domain filter are included — the most common choice for a full domain view:



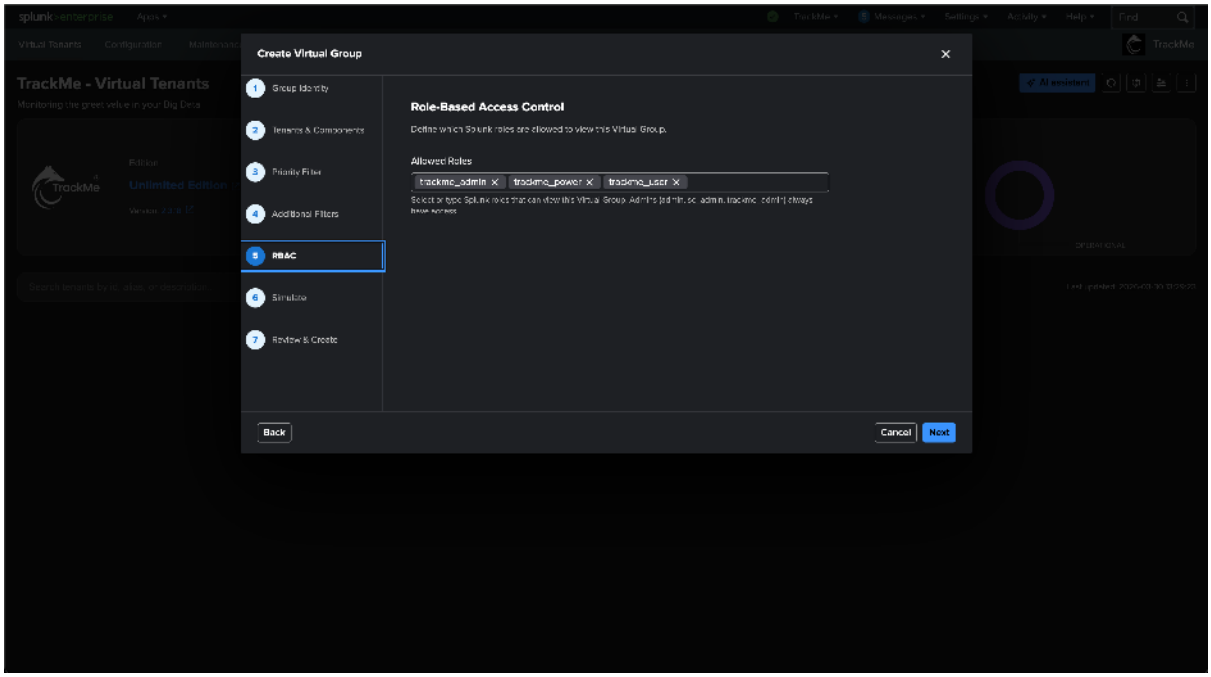
Step 4 — Additional Filters: This is where the domain scoping happens. Enter a free-form filter expression using entity fields. For the `Cloud` group, the expression is simply `tags="cloud"`:



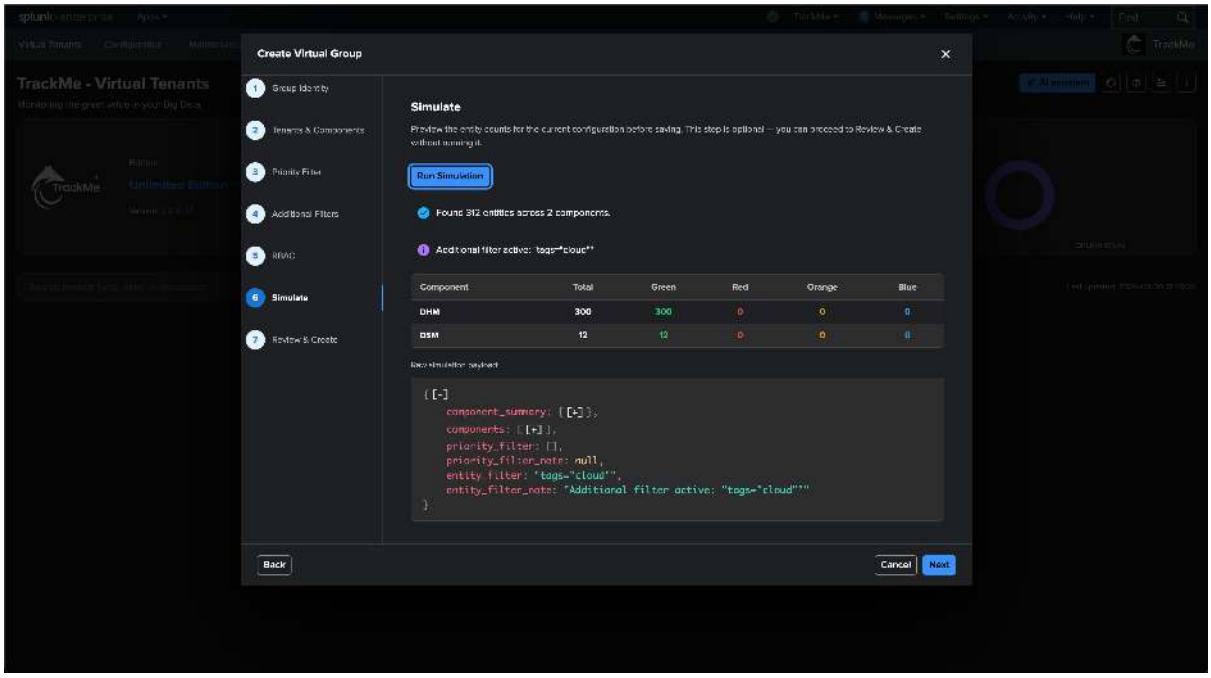
Hint

The filter syntax supports wildcards (`*`, `?`), logical operators (`AND`, `OR`), and parentheses for grouping. Comparisons are case-insensitive. Any entity field is supported — `tags`, `data_index`, `alias`, `priority`, `component`, and more. Leave the field blank to include all entities (after applying the Priority Filter above).

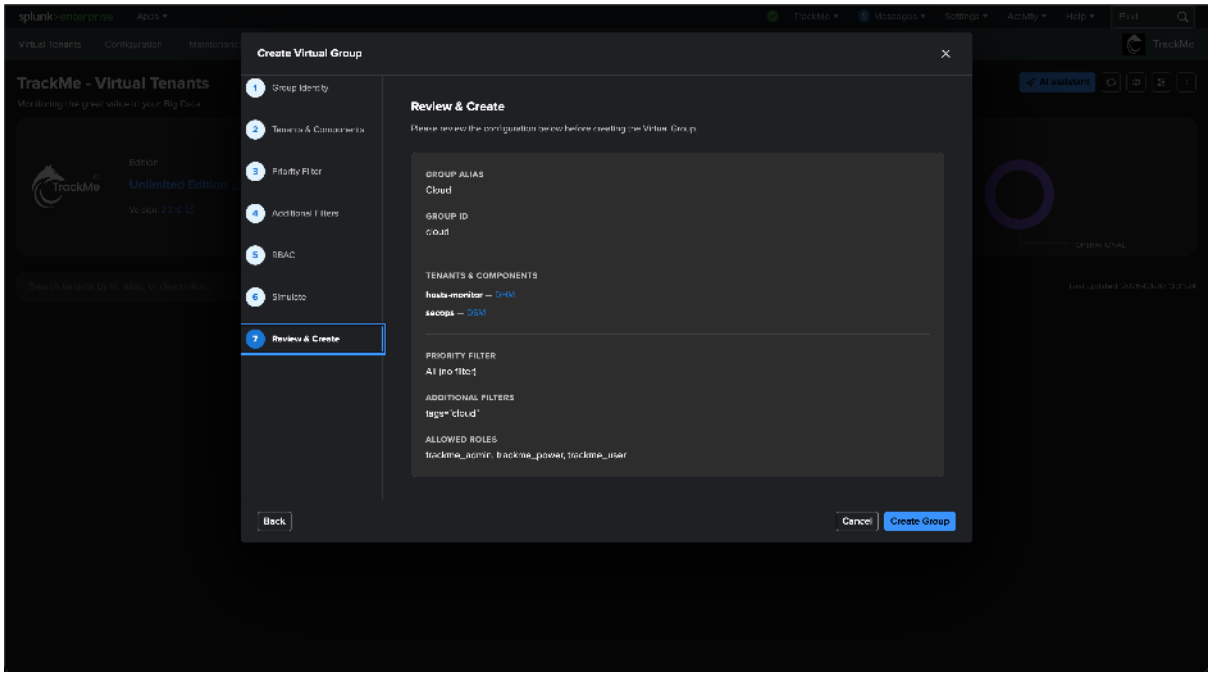
Step 5 — RBAC: Define which Splunk roles can see this Virtual Group. By default, all three TrackMe roles are pre-populated. You can remove roles to restrict visibility to specific teams:



Step 6 — Simulate: Before creating the group, run a simulation to preview how many entities will be included across each component. For the `Cloud` group, 312 entities are found across 2 components (DHM and DSM), confirming the filter is correct:



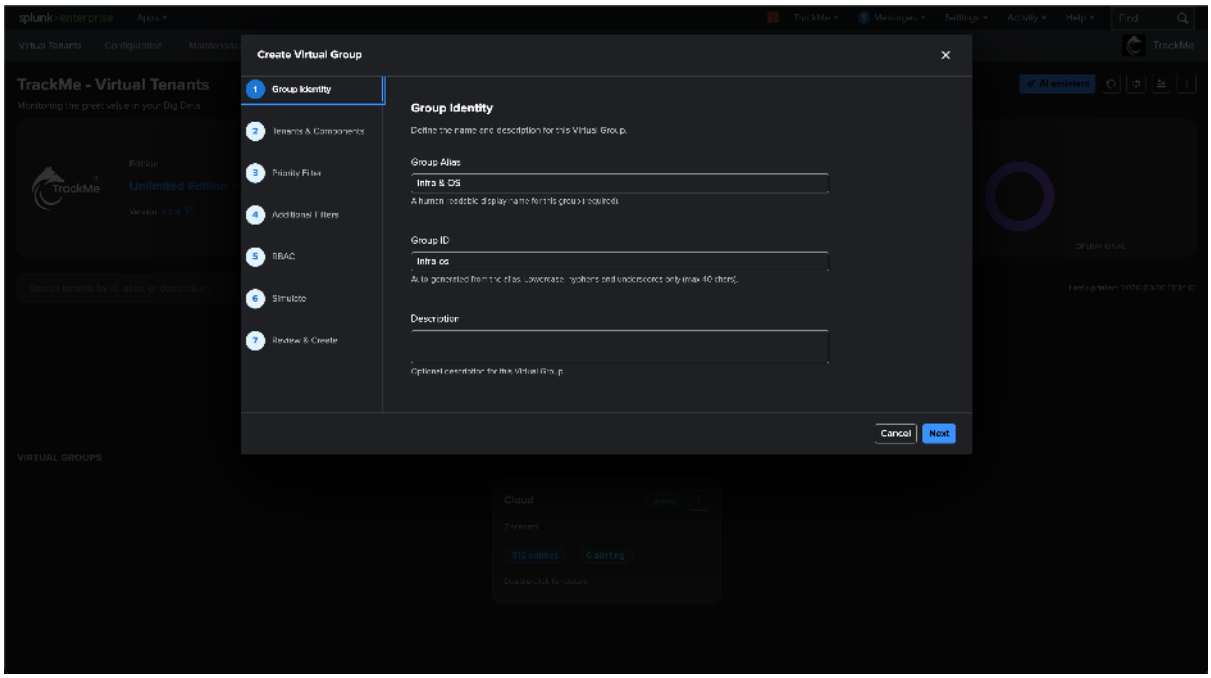
Step 7 — Review & Create: A final summary confirms all settings before creation. Click **Create Group**:

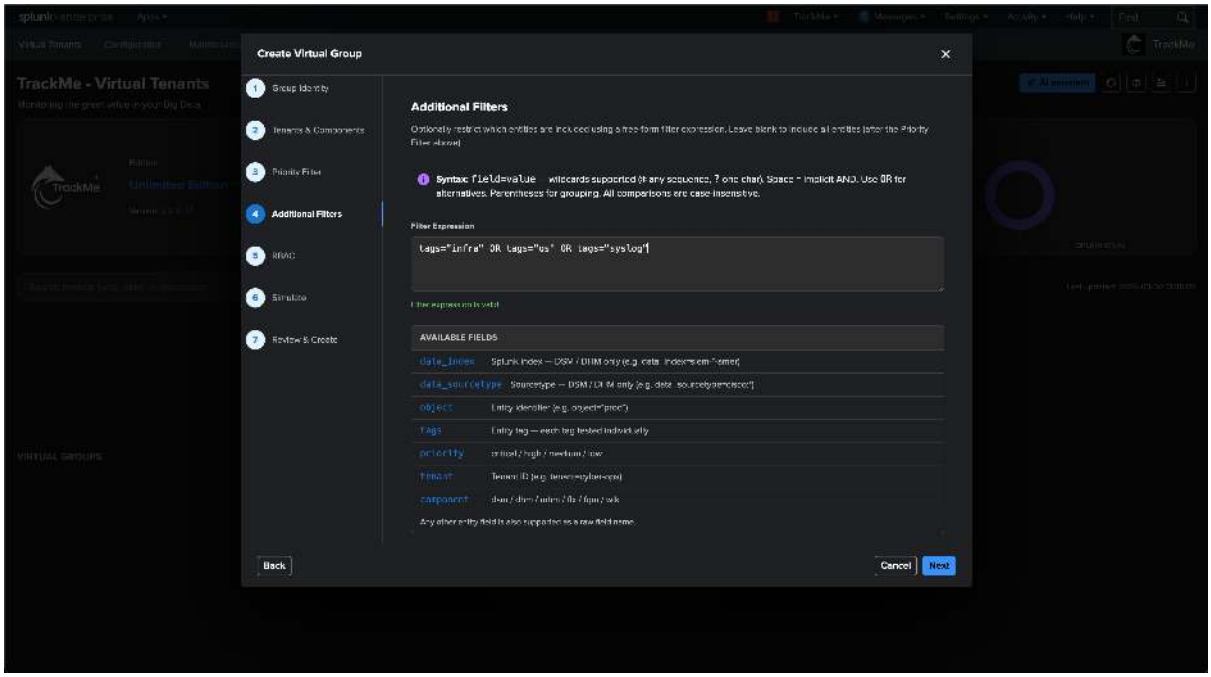


Creating the remaining groups

Repeat the wizard for each remaining domain. The only things that change between groups are the **alias** and the **filter expression**.

For example, the `Infra & OS` group aggregates three related domains into one view — useful when `infra`, `os`, and `syslog` are managed by the same team and benefit from a consolidated perspective:

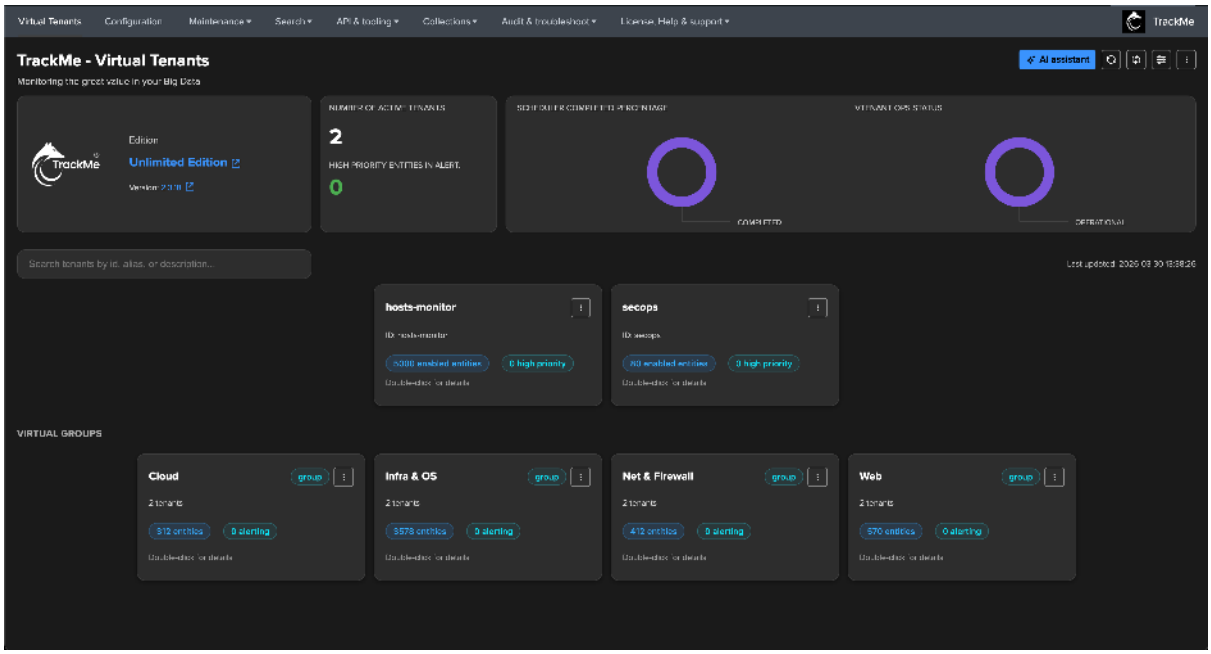




The `Net & Firewall` and `Web` groups follow the same pattern with their respective tag expressions (`tags="net" OR tags="firewall"` and `tags="web"`).

Step 4 — The final result

With all four groups created, the Virtual Tenants grid now shows a dedicated **Virtual Groups** section beneath the tenant cards. Each group card displays the aggregated entity count and alert status at a glance:

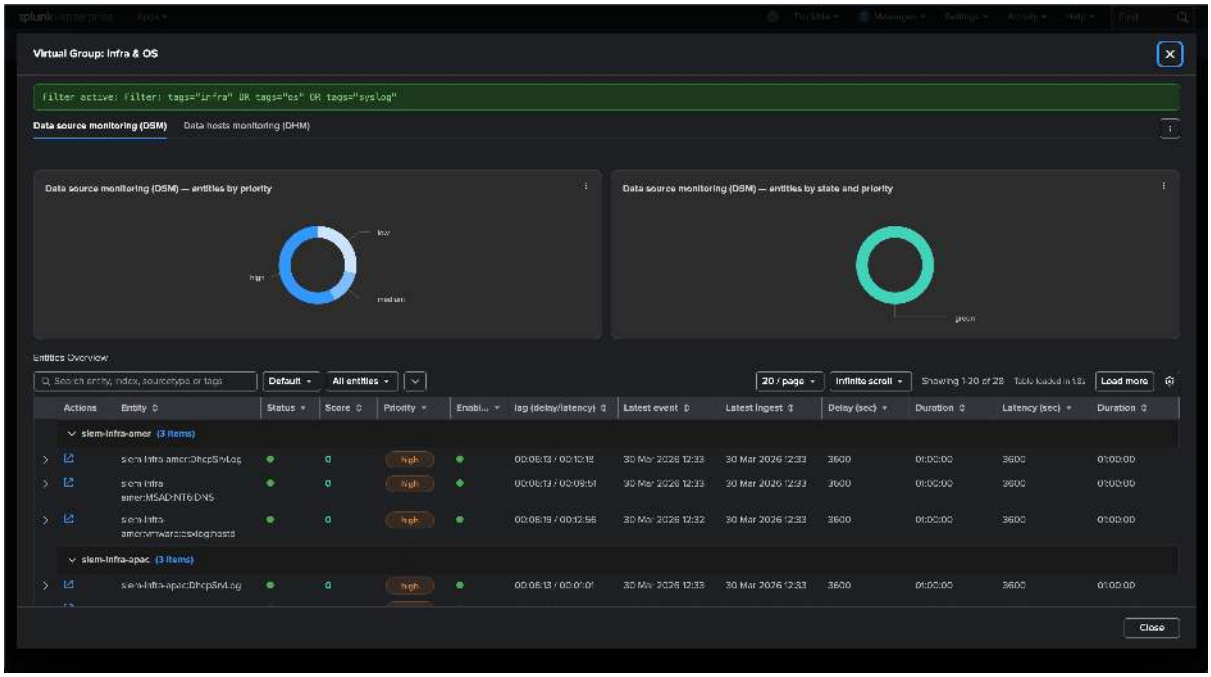


All four groups are live, sourcing their data from both tenants simultaneously. The cards update automatically alongside the rest of the Virtual Tenants grid — no manual refresh required.

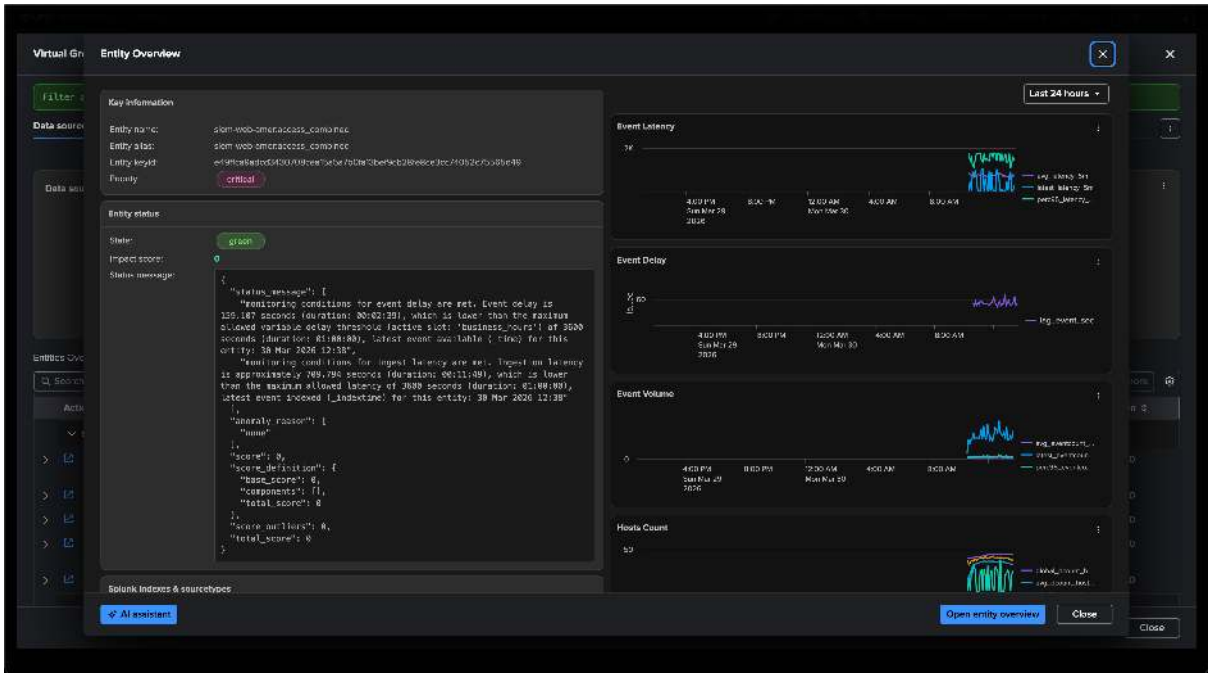
Exploring a Virtual Group

Double-click any Virtual Group card (or click **Open** in its context menu) to open the **Virtual Group Overview**. The overview provides a tabbed, per-component breakdown of all aggregated entities.

The active filter expression is shown at the top so it is always clear which entities are in scope. Charts give an immediate sense of the health distribution (by priority and by state) for each component, followed by the full entity table:

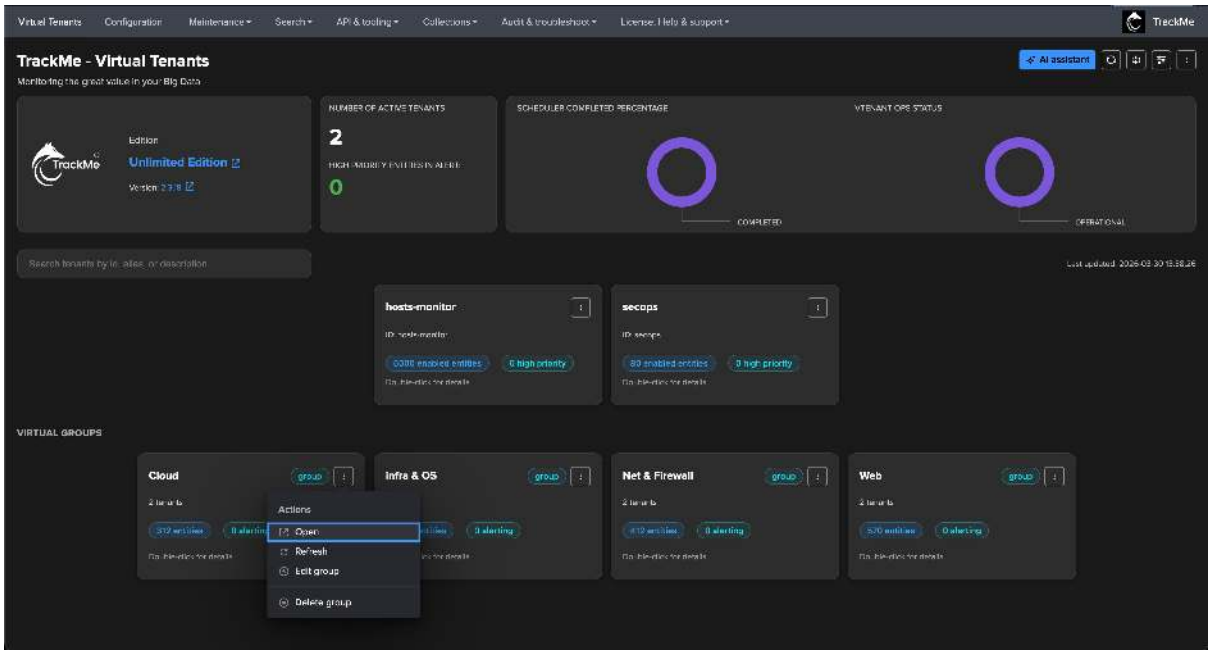


From the entity table, clicking on any entity opens its full **Entity Overview** — the same deep-dive view available from a regular Tenant Home, with event latency, delay, volume and hourly count charts, key information panel, and AI assistant access:



Managing Virtual Groups

Each Virtual Group card has a context menu (right-click or the ⋮ icon) with four actions:



- **Open** — opens the Virtual Group Overview modal
- **Refresh** — triggers an immediate re-fetch of entity counts for this group
- **Edit group** — reopens the creation wizard pre-populated with the current configuration (admin only)
- **Delete group** — removes the Virtual Group definition (admin only); no entity data is deleted

Summary and best practices

Key takeaways

- **Tags are the foundation.** Virtual Groups do not move or copy entities — they reference them by tag. Invest in a clean, consistent tagging strategy and your groups will always stay up to date automatically.
- **One lookup, many groups.** A single lookup-based tags policy per tenant can feed any number of Virtual Groups. Add a new domain? Add a row to the lookup, run the tracker, create a new group — done.
- **Compose domains freely.** The filter expression supports full boolean logic (`AND`, `OR`, parentheses, wildcards). You can combine multiple tags to represent composite domains (e.g., `Infra & OS`), add priority constraints, or filter by any entity field.
- **Cross-tenant by design.** A single Virtual Group can span any number of tenants and any combination of monitoring components (DSM, DHM, MHM, VOL, FLX, FQM, WLK). This is its core value: one view, many sources.
- **Read-only aggregation.** Virtual Groups do not participate in alerting, scoring, or the decision maker. They are pure visibility constructs — no monitoring side-effects.
- **Simulate before creating.** The Simulate step in the wizard lets you verify entity counts before committing. Use it to catch typos in filter expressions and confirm that both components contribute the expected entities.
- **RBAC per group.** Different groups can have different visibility rules. Use this to surface domain-specific views only to the teams that need them, without exposing the full tenant scope.
- **Priority filter for focused views.** When a domain contains thousands of entities, consider creating a

companion group with `Priority Filter = Critical + High` only — giving on-call teams a laser-focused alert view alongside the full domain view.

2.2 Use-case demos

Worked demonstrations of TrackMe in action — feeds monitoring, 360° services monitoring, fields quality (CIM and non-CIM), and Flex Objects.

2.2.1 Use Case Demo: This is All About Feeds

Hint

About impact score since TrackMe v2.3.5

- Since this release, TrackMe implements an impact score system to determine the status of an entity based on the KPIs, thresholds and anomalies.
- Impact score based analysis provides deep flexibility and control.
- The impact score is influenced by anomalies weights, such as a delay breach, as well as score events that can be generated by analysts (declaring false positives, manual score influence, etc.) or the system itself. (outliers detection for instance)

Use Case Demo: This is All About Feeds

- This white paper provides an overview of Splunk feeds tracking with TrackMe.
- It covers the key concepts and features around tracking data sources and hosts availability and indexing performance.
- Feeds tracking is a key feature of TrackMe, representing the root use case of TrackMe, and in many cases, the first reason why customers use TrackMe.
- This use case demo documentation explains and demonstrates the key features of TrackMe feeds tracking.

High Level Overview

At the source of TrackMe concepts stands the idea of tracking data sources and hosts availability and indexing performance.

The key features can be summarized as follows:

- `Discovery`: Discover and organize data sources automatically based on meaningful key metadata.
- `Persistence`: Persist entities and their key information over iterations, using various features from Splunk KVstore collections to events and metrics indexing.
- `Availability`: Track entities availability over time based on `delay` thresholds, to detect when a given entity has stopped forwarding events to Splunk.
- `Performance`: Track entities indexing performance over time based on `latency` thresholds, to detect when a given entity is suffering from indexing performance issues.
- `Quality`: Provide out-of-the-box and automated capabilities to detect meaningful lack of quality in the entities data.
- `Metrics`: Generate and store metrics in a high-performance way, using Splunk metric store capabilities.
- `Knowledge`: Accumulate knowledge over time to improve investigations and decision-making.

- `Machine Learning`: Implement meaningful and automated Machine Learning models to detect volume variations and potential anomalies on entities.
- `Flexibility`: Provide flexible and powerful capabilities to cover from basic tracking use cases to the most complex ones.
- `Alerting`: Provide out-of-the-box powerful and advanced alerting capabilities, from state-aware alerting to rich email deliveries, state-aware actions, and more.
- `Scalability`: Scale at any level, from few terabytes environments to hundreds of terabytes environments.
- `User Experience`: Provide a rich, easy, and fast user experience, with user interfaces and APIs that provide fast answers and browsing capabilities with TrackMe entities.

This use demo documentation covers four common and most valuable feeds related use cases:

- `Feed Interruption Detection`
- `Feed Indexing Performance Issues Detection`
- `Feed Volume Variation Detection`
- `Feed Quality Issues Detection`
- `Feed Hosts Distinct Count Anomaly Detection`

Many more use cases can be implemented with TrackMe, and through our Flex Trackers components, you can create custom monitoring solutions tailored to your specific needs.

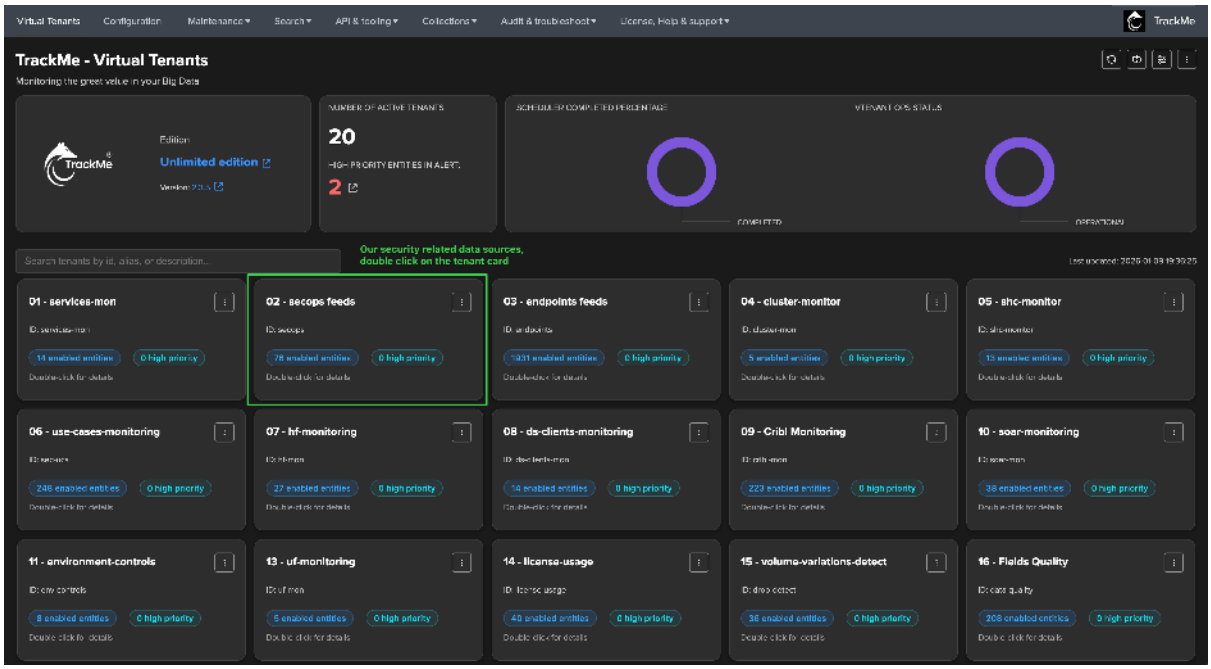
Feeds Tracking with TrackMe Data Sources Monitoring (splk-dsm)

The Data Sources Monitoring TrackMe component (splk-dsm) is the key component to track data sources availability and indexing performance.

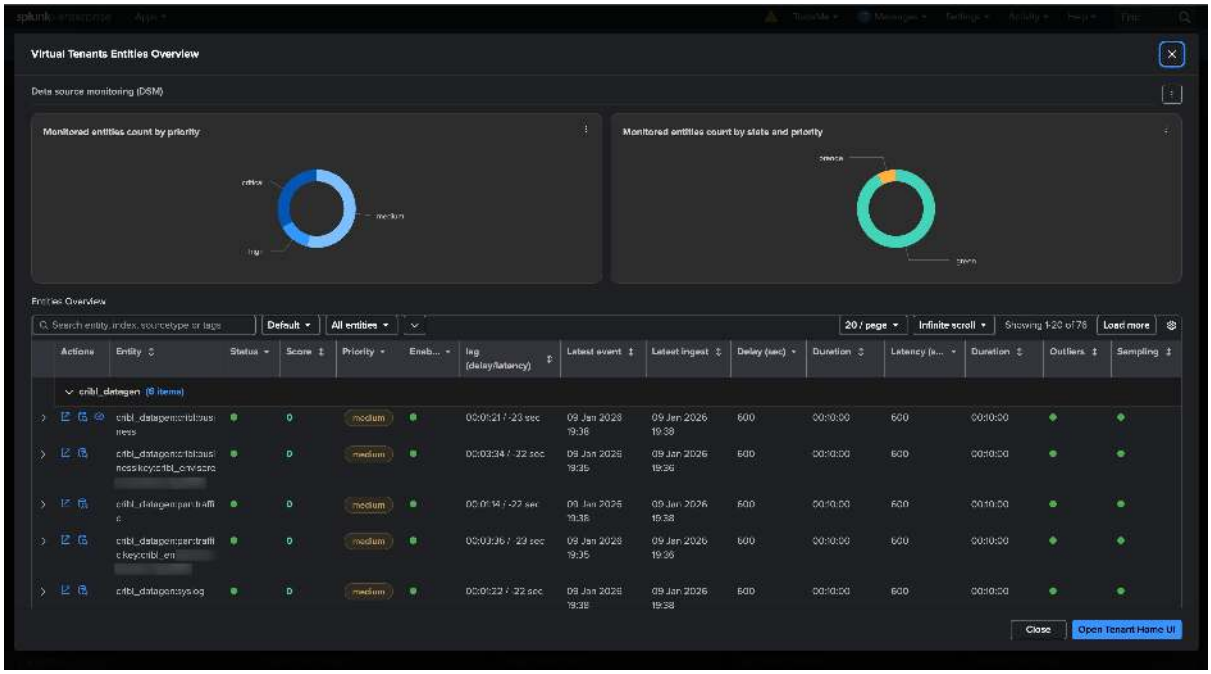
In its simplest form, it is tracking feeds, creating and organising entities based on `indexes` and `sourcetypes` metadata.

The Virtual Tenants UI

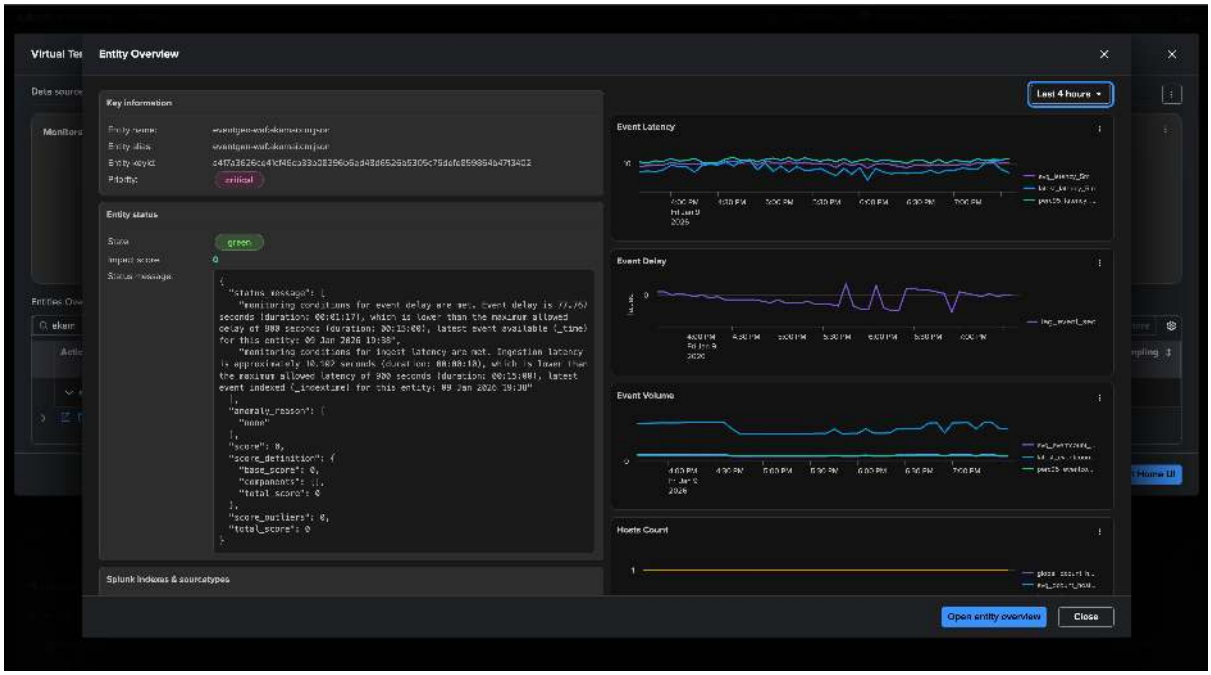
Virtual Tenants allow customers to segment use cases based on their needs, from geographically oriented use cases to use cases based on data sensitivity or perimeters:



Virtual Tenant Entities Overview:

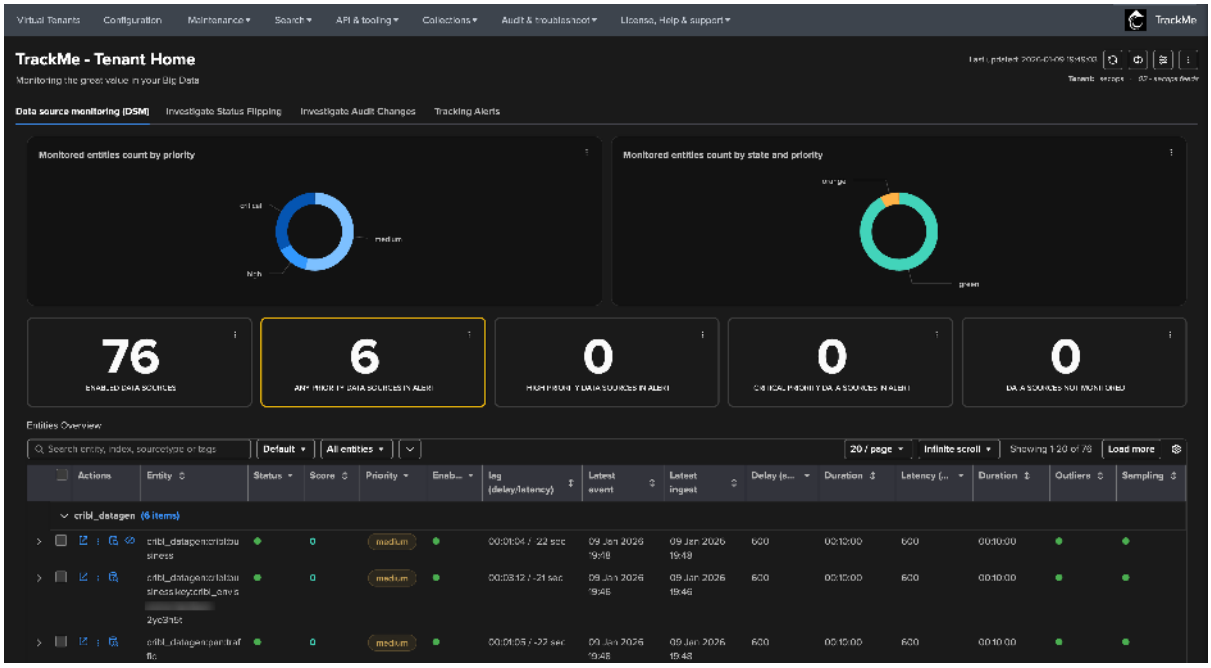


Expand or click on the name of an entity access the entity overview screen:

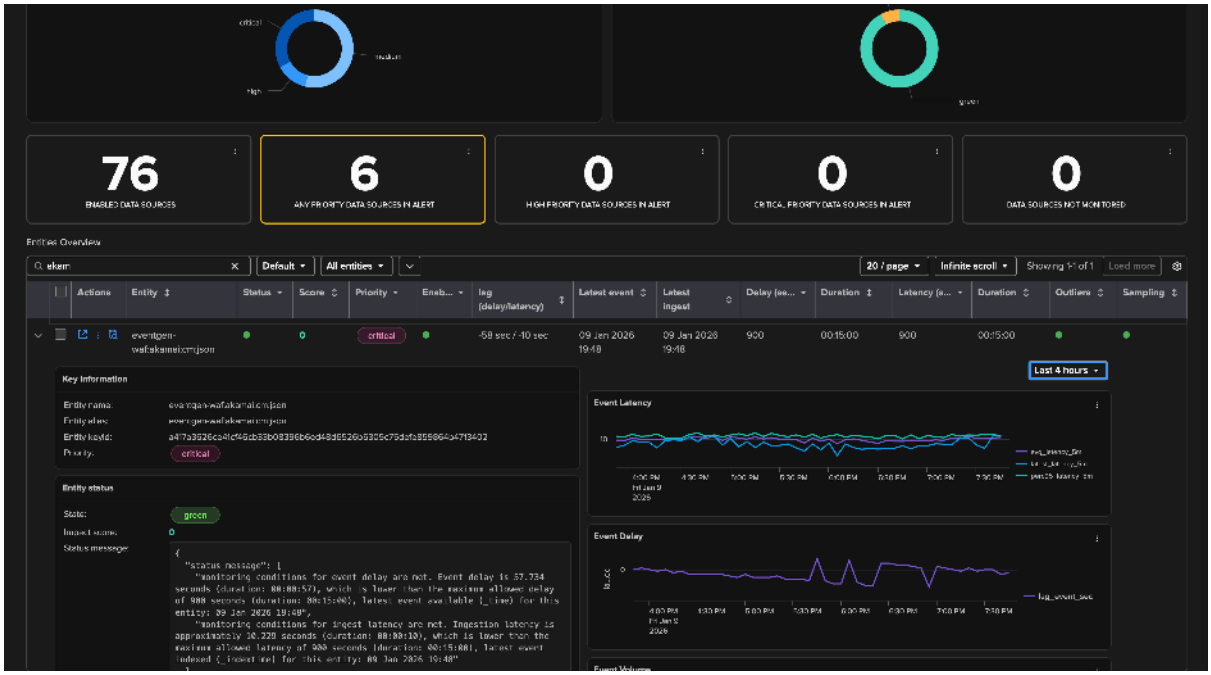


The Tenants Home UI

The Home UI is the main user interface to manage your TrackMe entities:

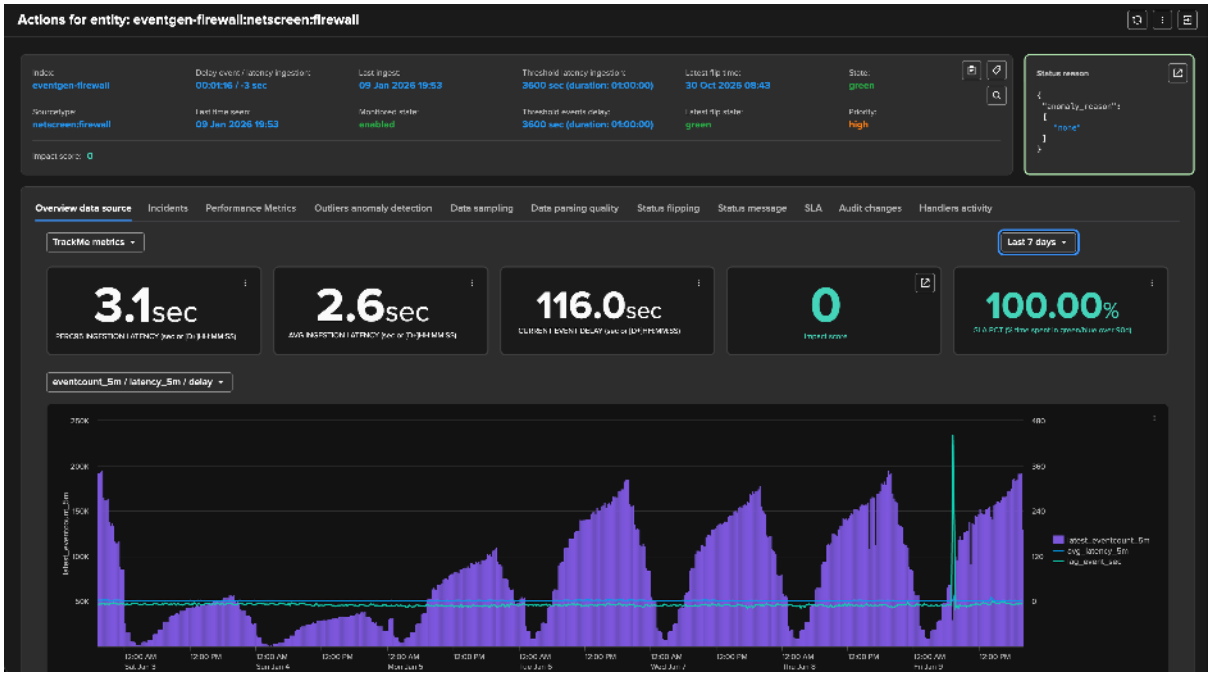


You can similarly access the entity overview by expanding or clicking on the name of an entity in the Tenants Home UI:



To Access to the status of a given entity, representing a Splunk feed (by default through its index and source-type metadata), click on the left hande side icon:

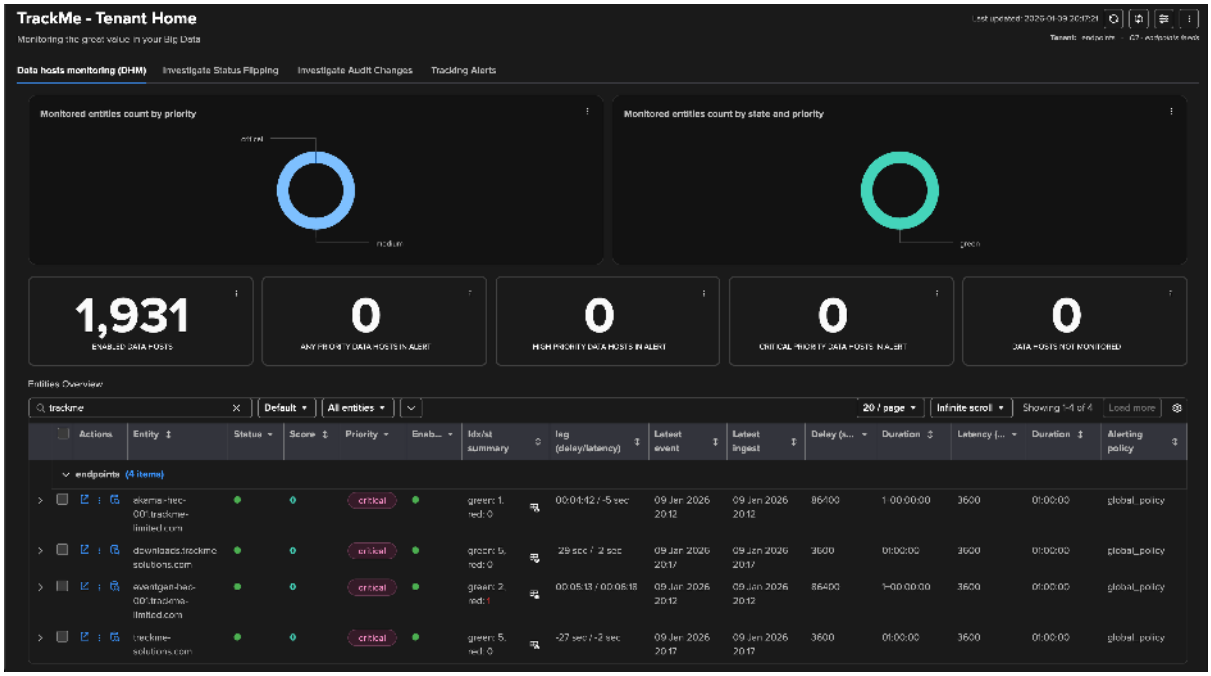
The entity details page provides a rich set of information about the entity, including its status, availability, indexing performance, metrics, knowledge, machine learning, flexibility, alerting, scalability, user experience, and more:

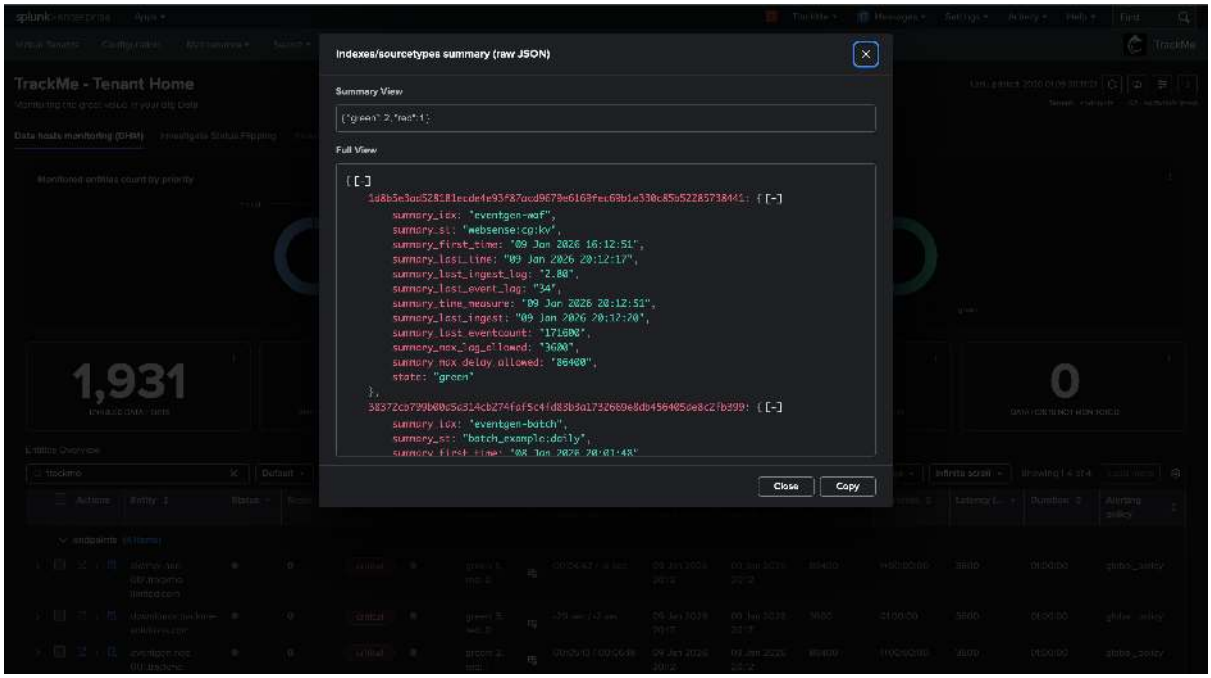


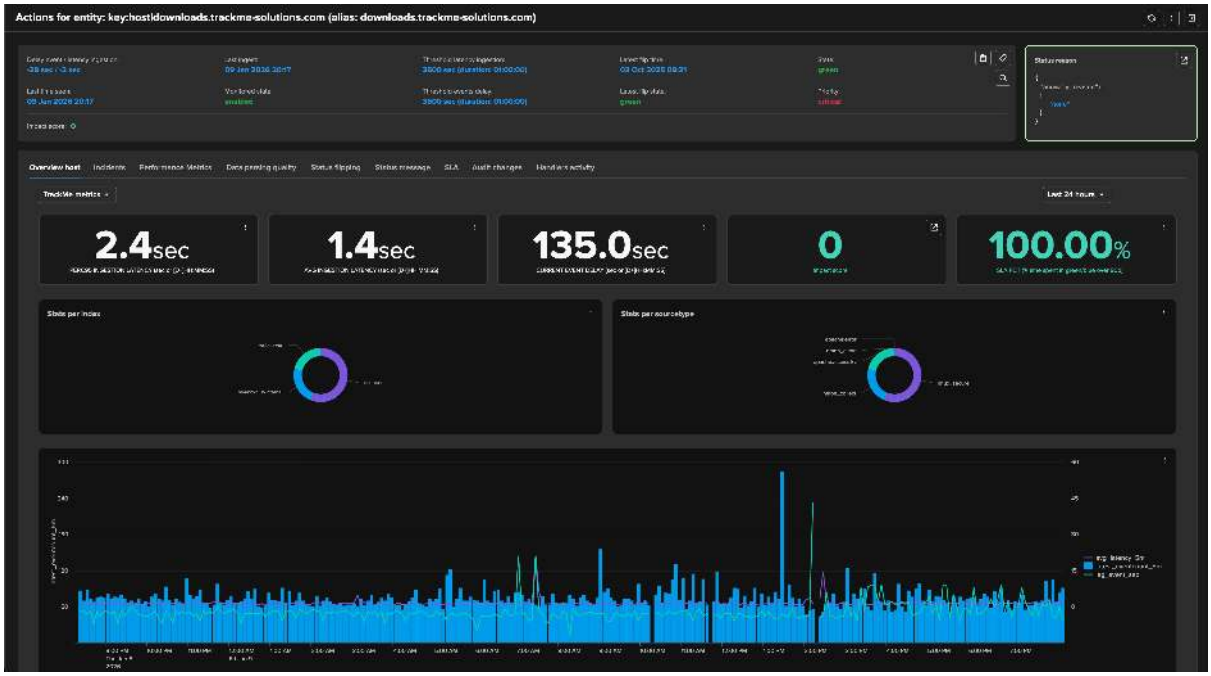
Hosts Tracking with TrackMe Hosts Sources Monitoring (splk-dhm)

The second main components for feeds tracking is the Hosts Sources Monitoring TrackMe component (splk-dhm).

It acts similarly to the Data Sources Monitoring TrackMe component (splk-dsm), at the difference that it is tracking data from host events producer perspective, which means sourcetypes associates with a given host depending on the tracker scope of data:







Primary Feeds Key Performance Indicators (KPIs)

Hint

About impact score since TrackMe v2.3.5

- Since this release, TrackMe implements an impact score system to determine the status of an entity based on the KPIs, thresholds and anomalies.
- Impact score based analysis provides deep flexibility and control.
- For feeds tracking and primary delay and latency KPIs, the default behaviour sets a delay breach anomaly as a major factor weight, and latency as a secondary factor weight.
- This means that by default, a feed affected by latency issues only appears in a warning state, while a feed

affected by delay issues appears in a alerting state.

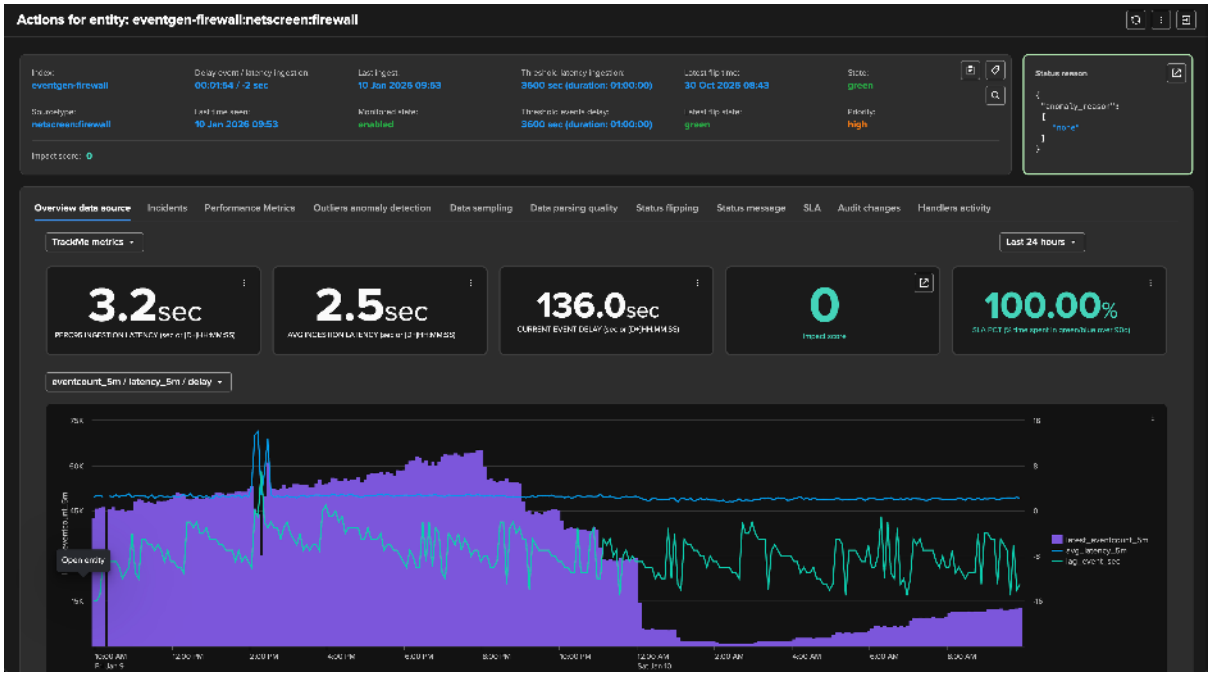
TrackMe is all about Key Performance Indicators (KPIs) and high performing metrics, this means that whatever the use case, we turn data into metrics then act against these.

In all components, you will find a tab called `Performance Metrics` which lists available metrics for the given entity.

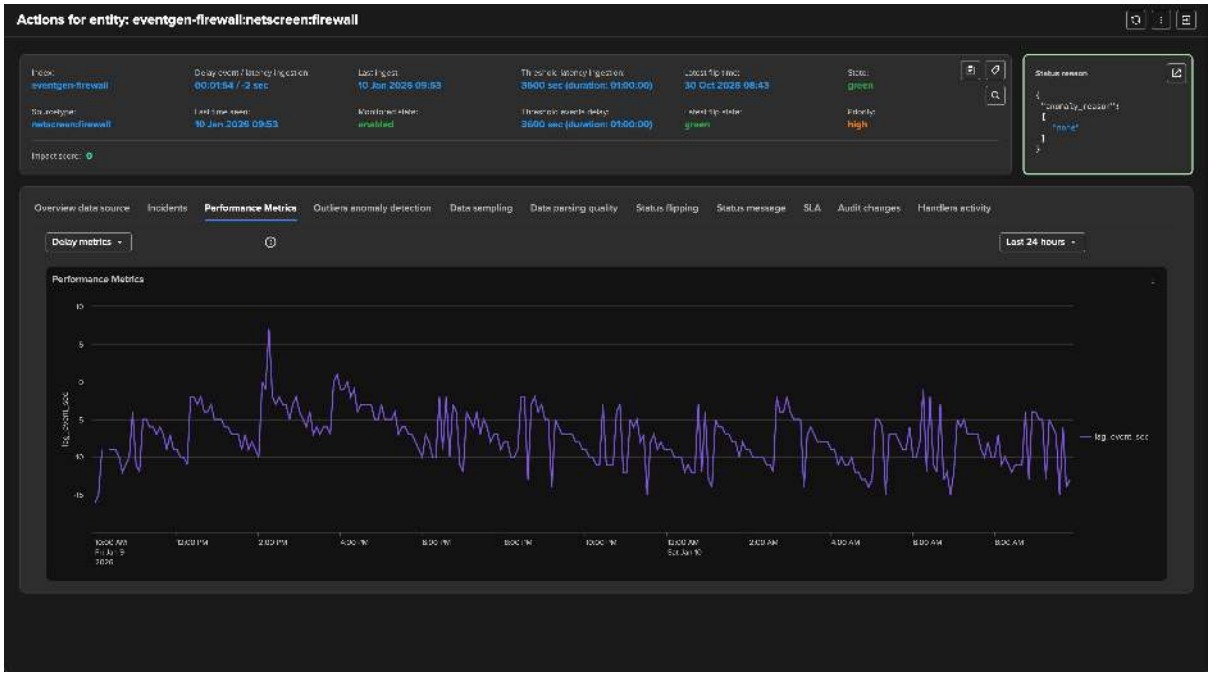
For Data Sources and Hosts, the primary KPIs are:

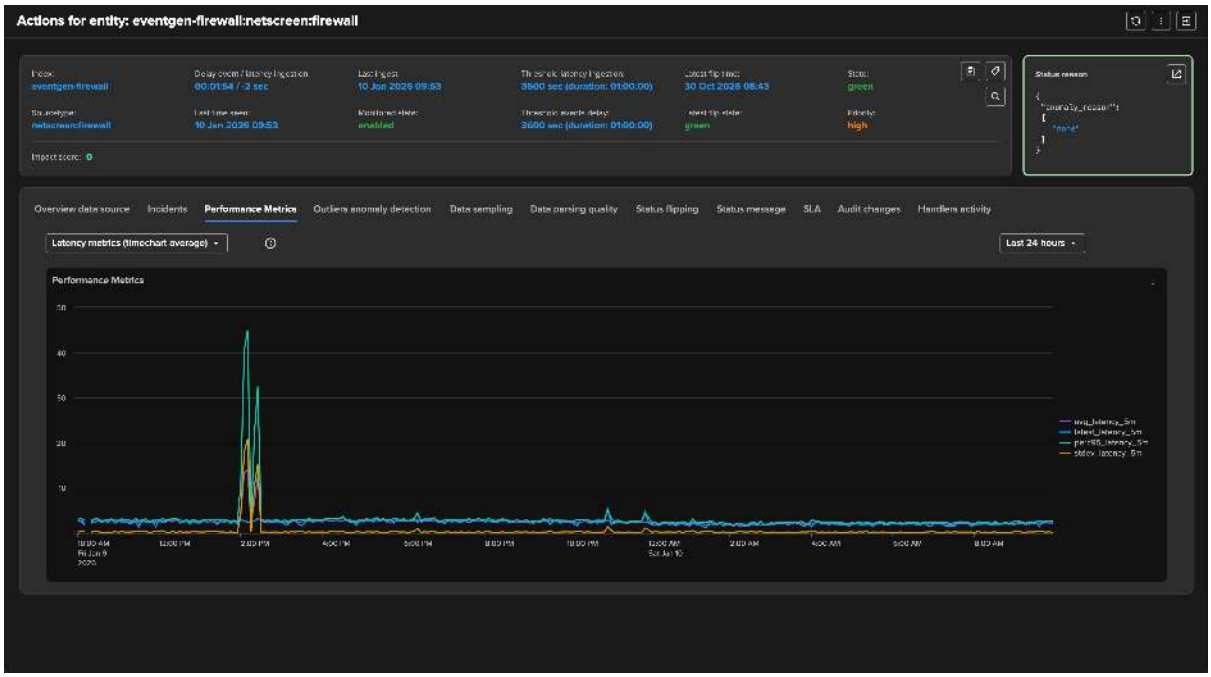
- `Latency`: The performance at the indexing level from the Splunk perspective, as the difference between the indexed time and the event time.
- `Delay`: How late the data is, calculated as the difference between now when the tracker is executed, and the very last event time.

The Entity overview shows both KPIs in the main view:

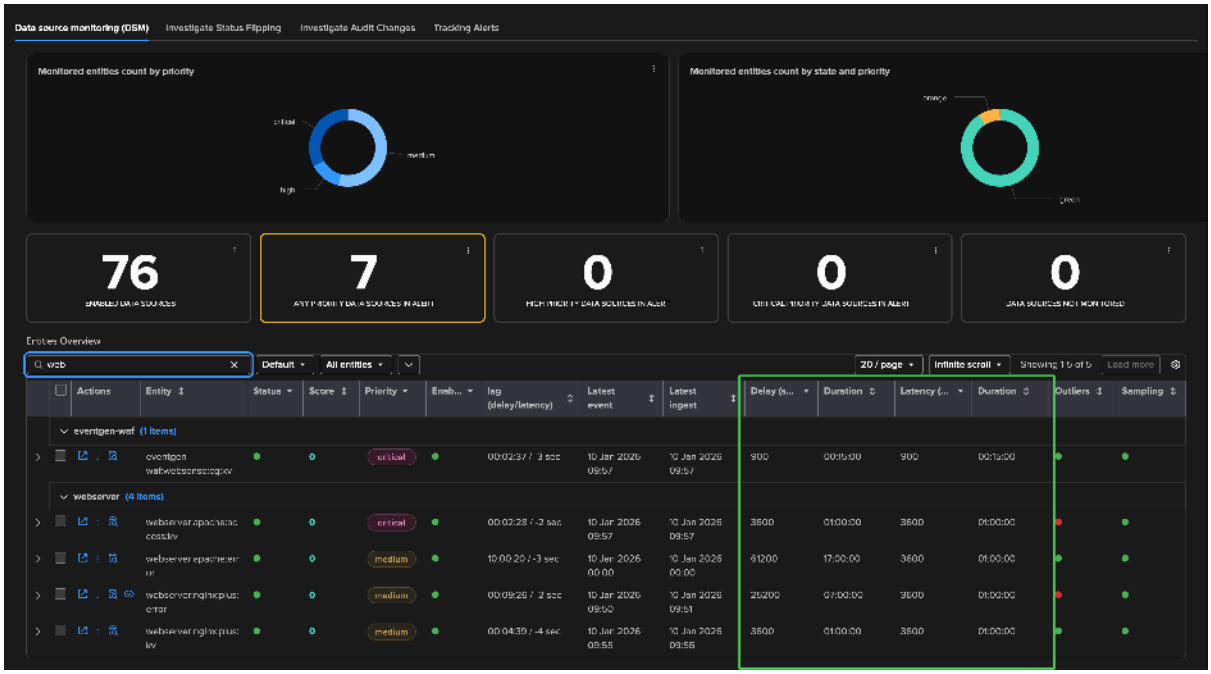


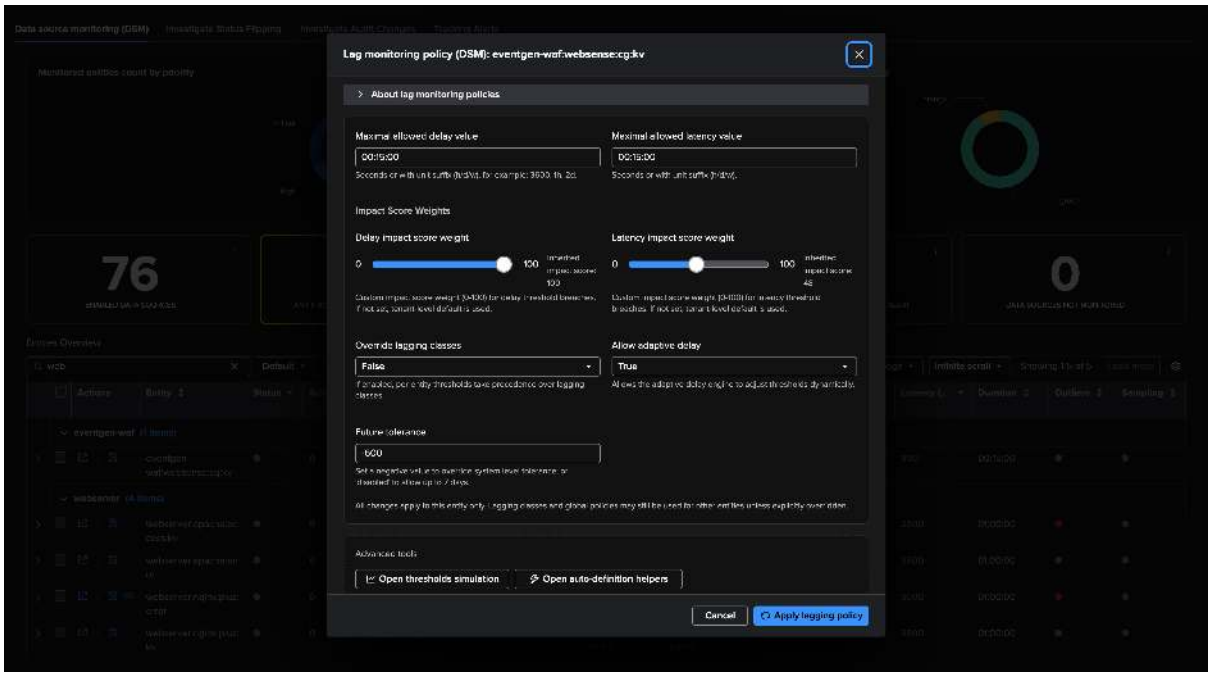
You can review in details metrics in the "Performance metrics" tab:





TrackMe applies configurable thresholds to influence the impact score calculation:





Use Case: Feed Interruption Detection

A first essential use case you want to cover when it comes to feeds availability monitoring is the detection of feed interruptions. In many use cases, you might expect the data to be flowing into Splunk in real-time or semi-real-time, though this isn't necessarily the case, and you need to have the capabilities to handle this on a case-by-case basis.

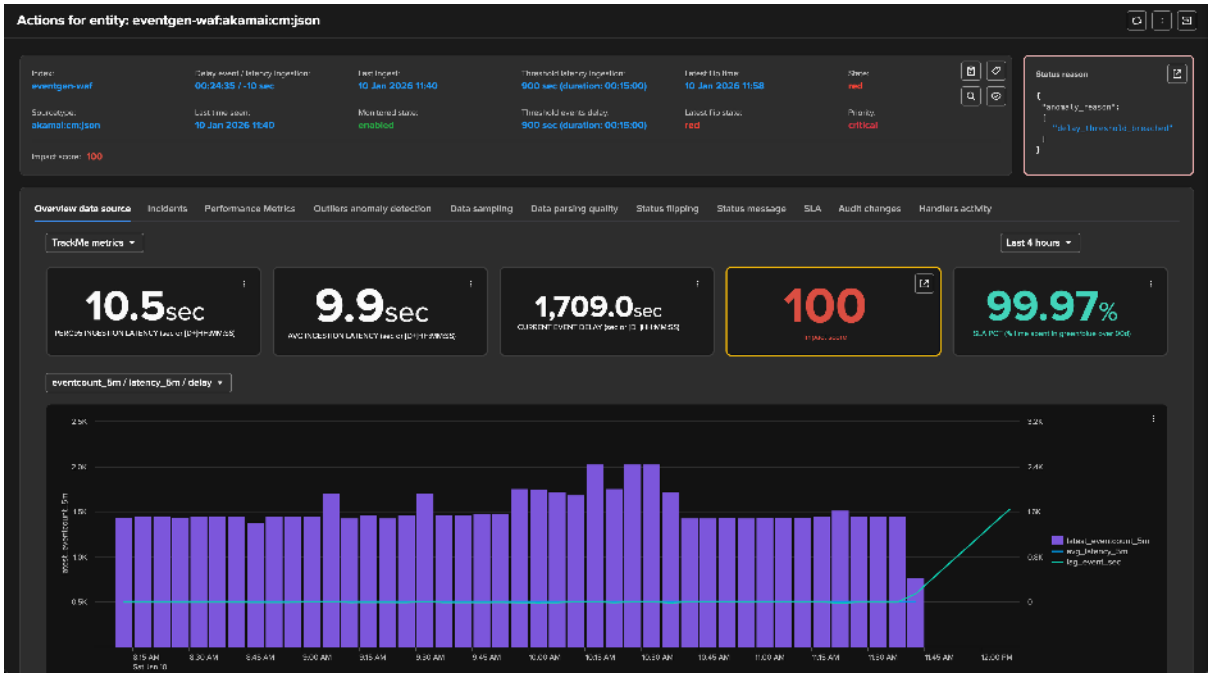
To cover this use case, TrackMe tracks the very latest event available on a per-entity basis, that is the event from the timestamp point of view (field _time). We calculate the difference between now, when the tracker is executed, and the last event, which gives you an indicator in seconds.

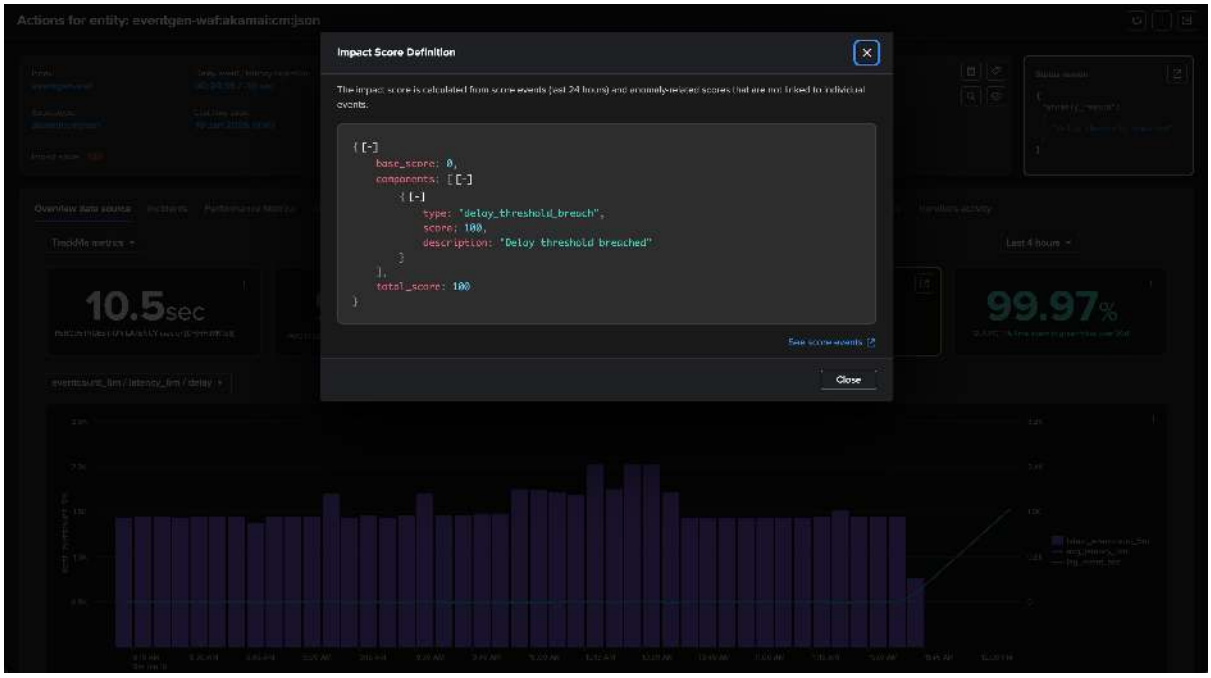
If this indicator breaches the threshold:

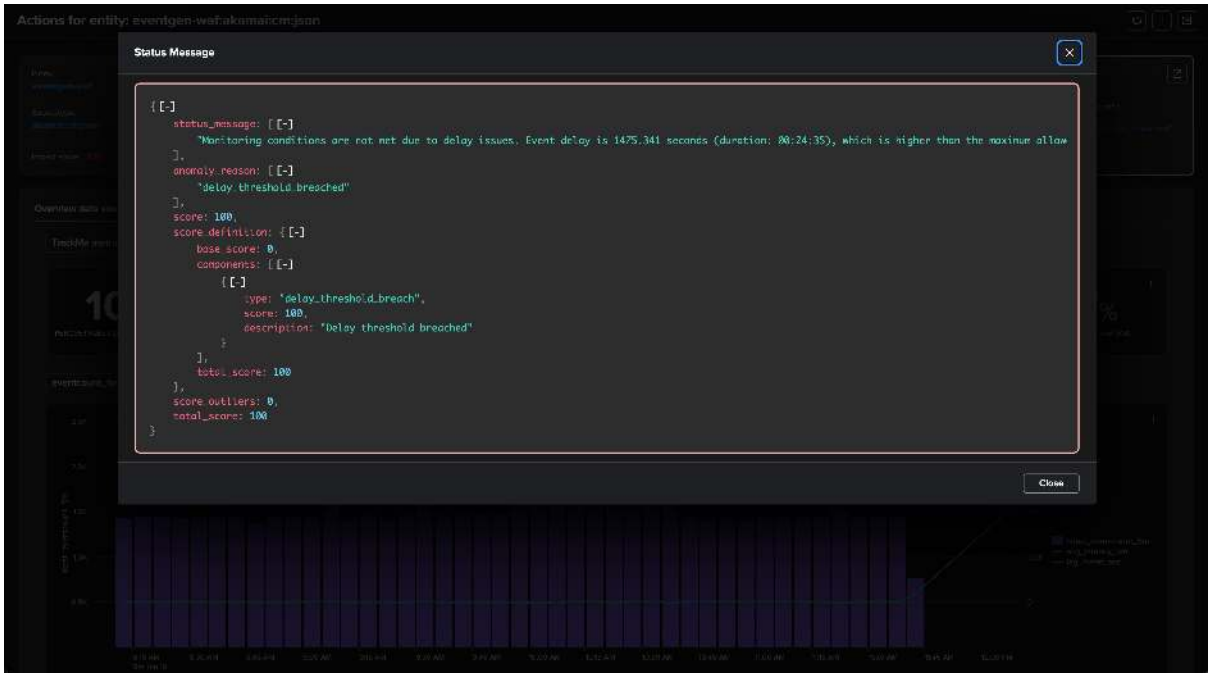
- The impact score is increased by 100 points by default, which raises the entity status to alert.
- The anomaly reason is set to `delay_threshold_breached` for entities using the static delay policy, or `variable_delay_threshold_breached` for entities using the variable delay policy (since TrackMe v2.3.12).

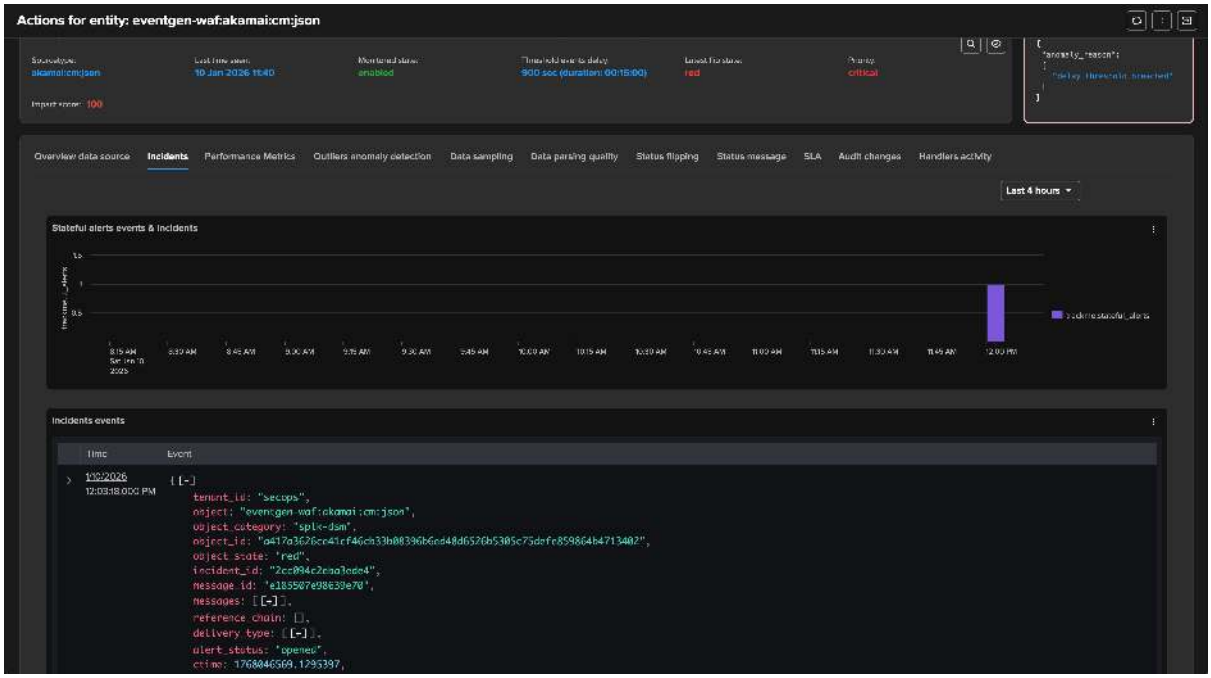
The workflow can be summarized as follows:

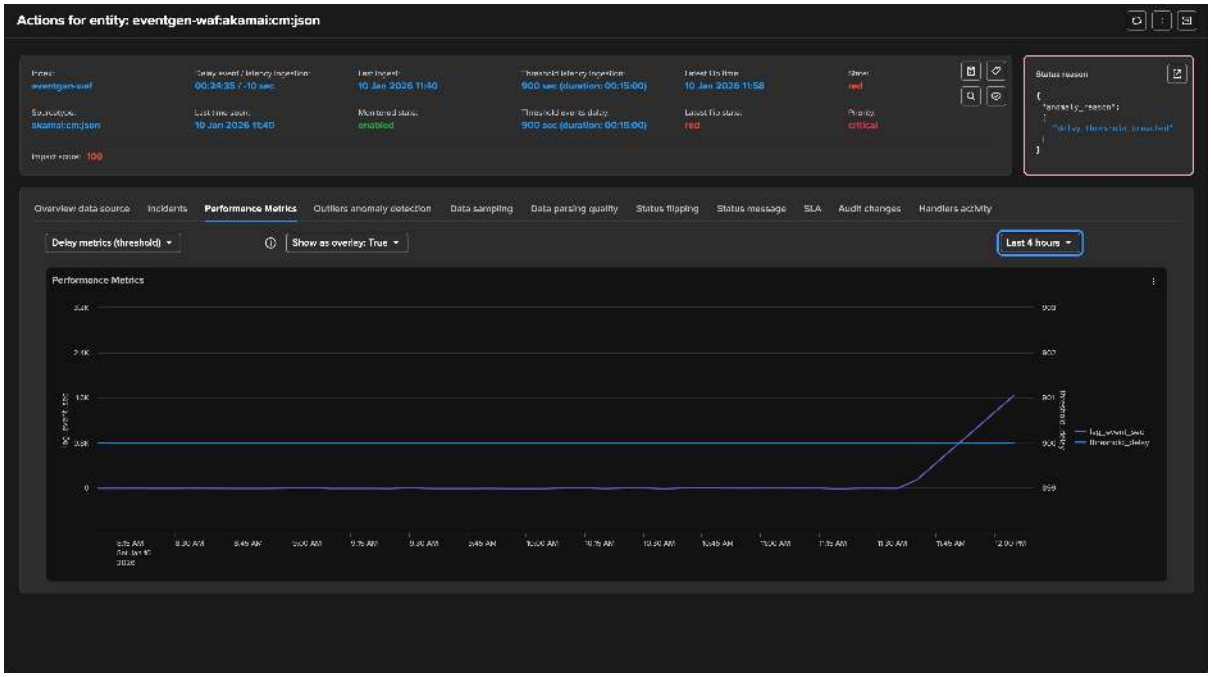
- The delay key performance indicator breaches the configured threshold value for this entity.
- TrackMe identifies the issue and raises the impact score by the associated weight, 100 by default. (anomalies score weight can be configured at the level of the tenant and overriden per entity)
- If the impact score is greater or equal to 100, the entity status is set red. (alerting state)
- TrackMe generates a flipping event which marks the entity state transition.
- If a TrackMe alert is configured, the alert fires and TrackMe runs associated actions. (email notification, incident creation, automated acknowledgement, etc.)

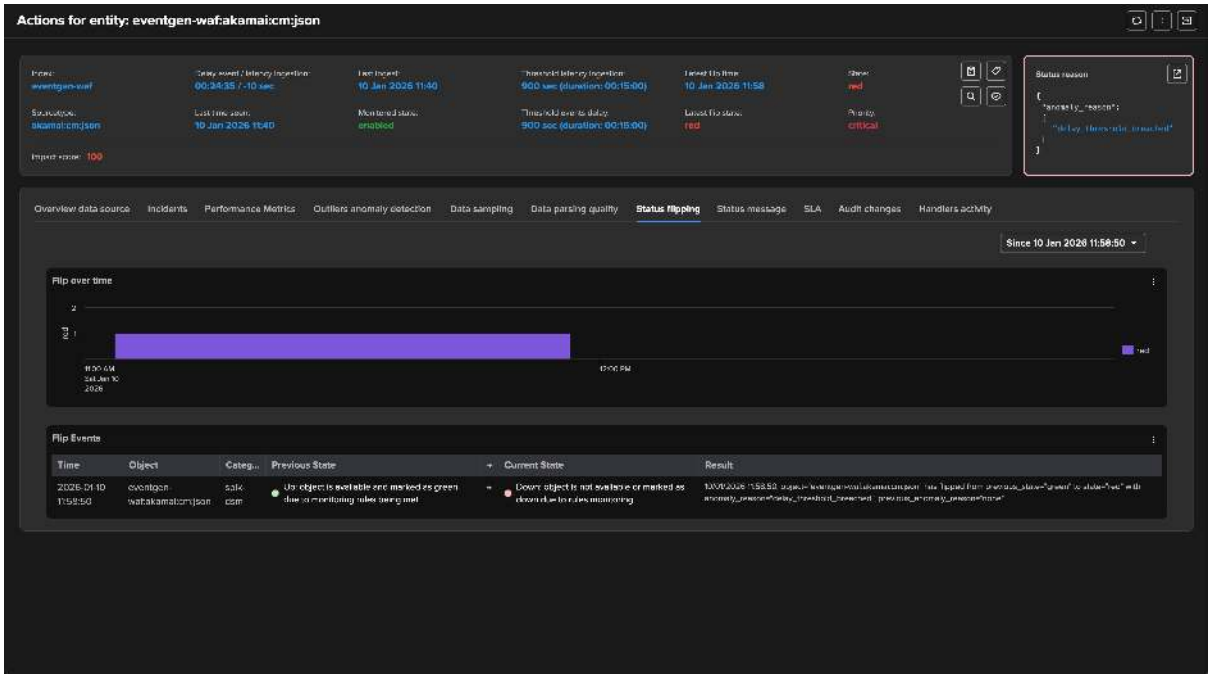


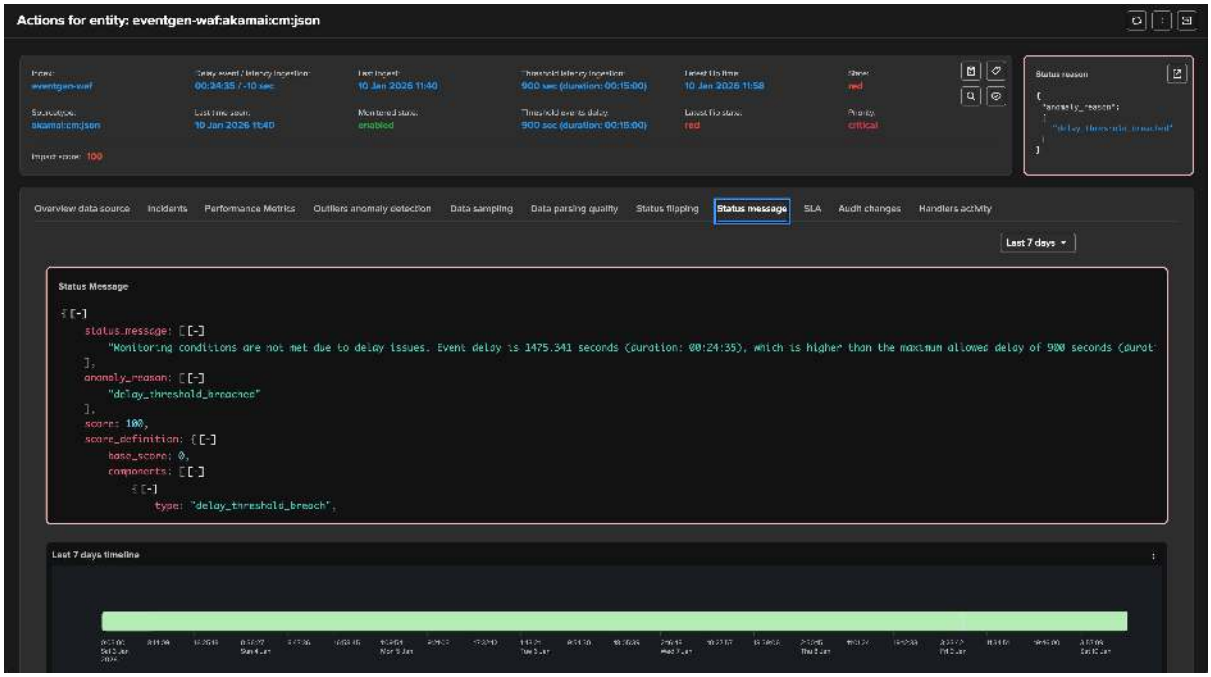


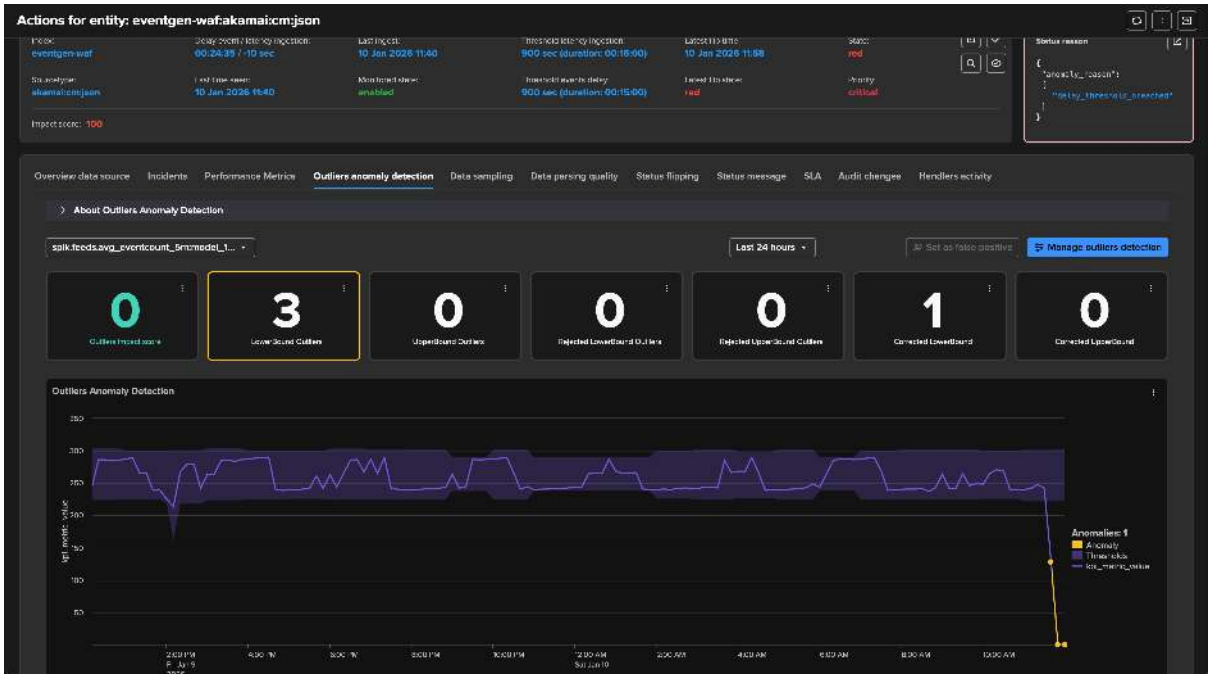

Actions for entity: eventgen-waf:akamai:cm:json
Status Message
{ [-]
status_message: [[-]
"Monitoring conditions are not met due to delay issues. Event delay is 1475.341 seconds (duration: 00:24:35), which is higher than the maximum allow
],
anomaly_reason: [[-]
"delay_threshold_breached"
],
score: 100,
score_definition: { [-]
base_score: 0,
components: [[-]
{ [-]
type: "delay_threshold_breach",
score: 100,
description: "Delay threshold breached"
}
],
total_score: 100
},
score_outliers: 0,
total_score: 100
}
Close

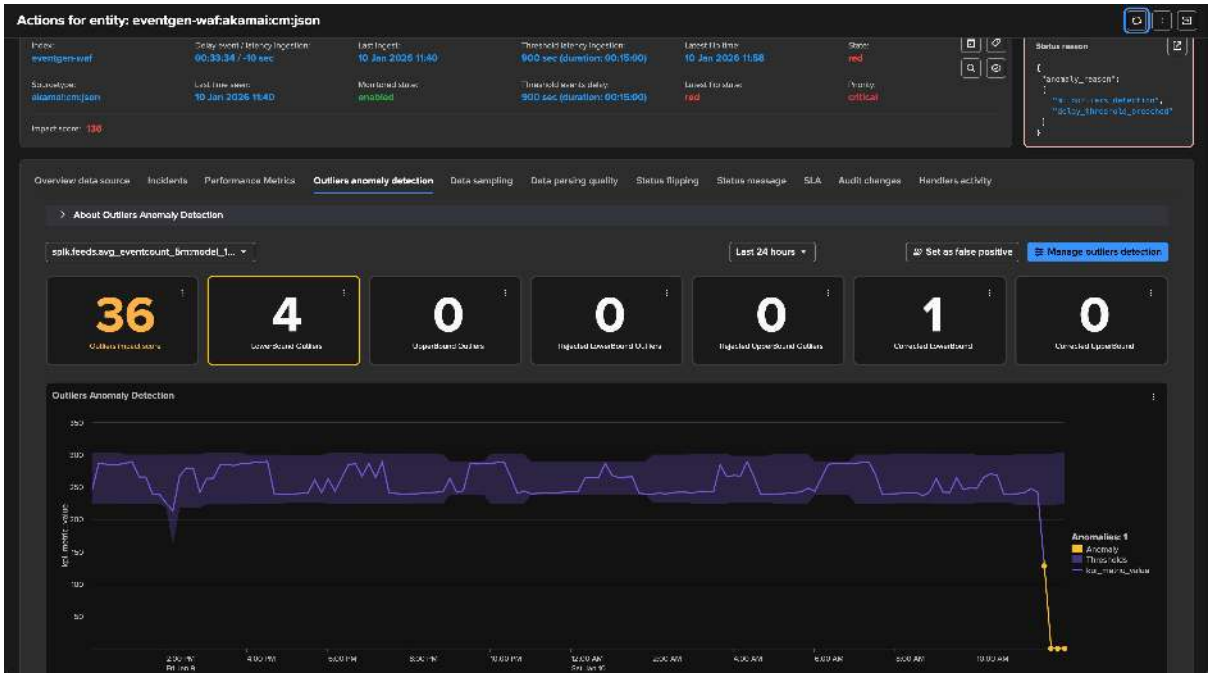

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Last 4 hours
Stateful alerts events & Incidents
Incidents events
Time
Event
tenant_id: "secops",
object: "eventgen-waf:akamai:cm:json",
object_category: "splk-dsm",
alert_status: "opened",

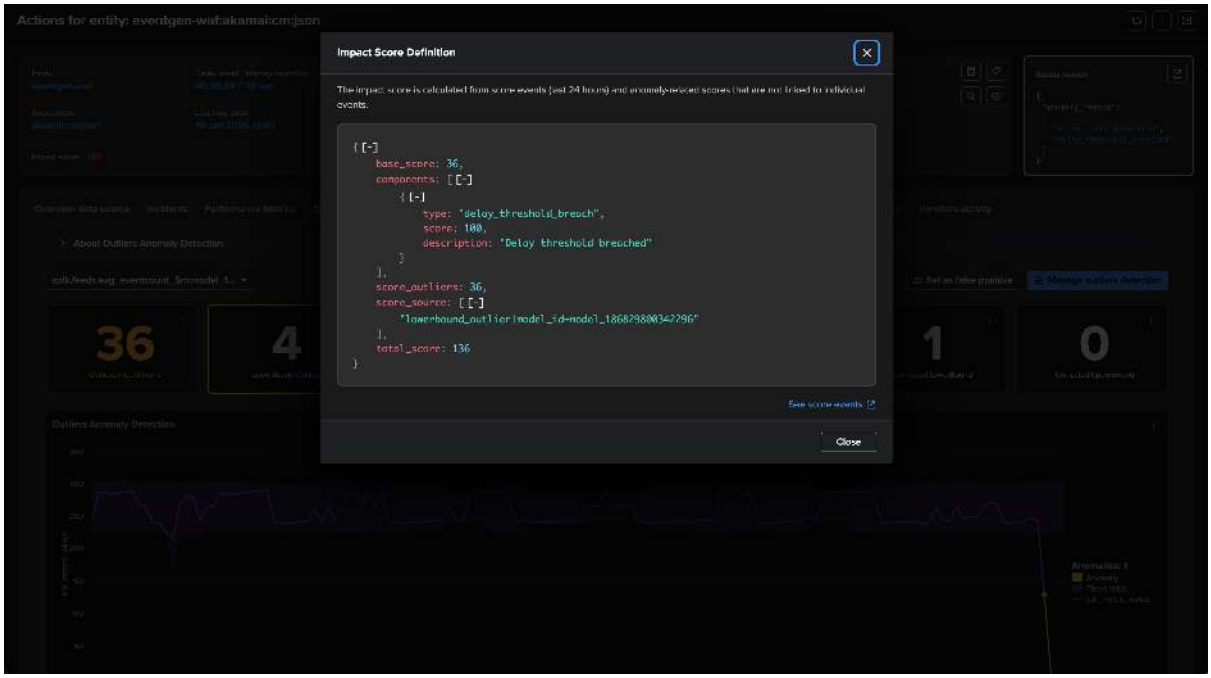

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Delay metrics (threshold)
Show as overlay: True
Last 4 hours
Performance Metrics


Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Since 10 Jan 2026 11:58:50
Flip over time
Flip Events
Time
Object
Previous State
Current State
Result

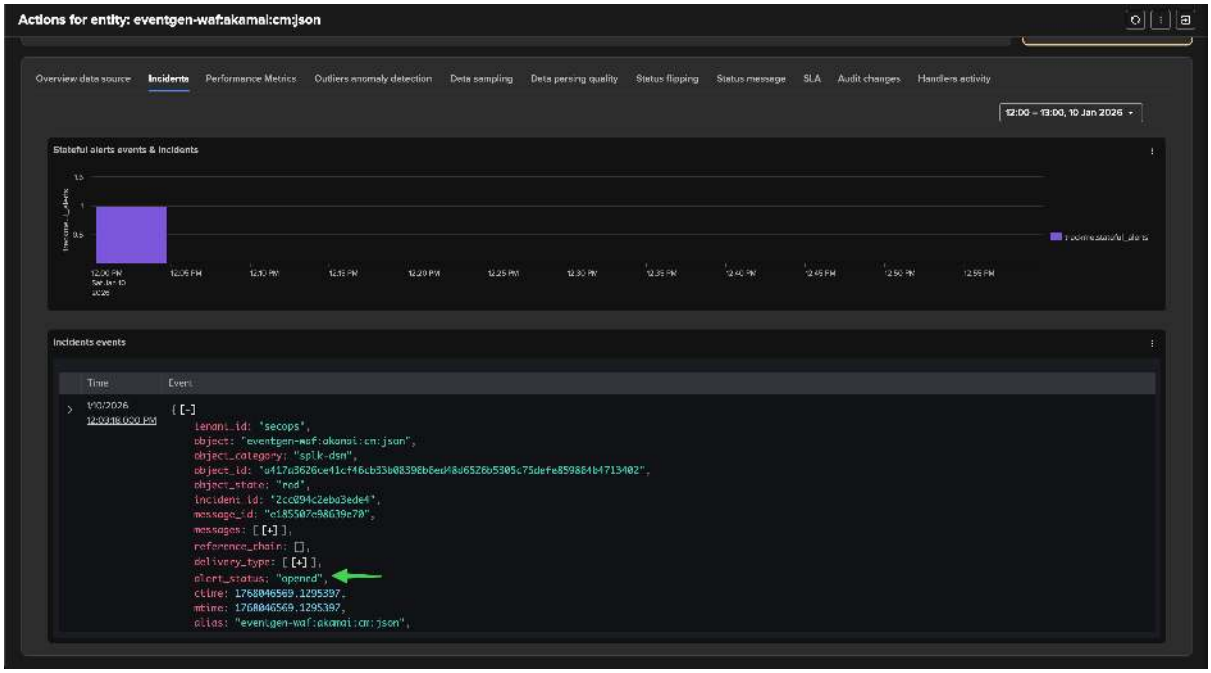

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Last 7 days
Status Message
Last 7 days timeline

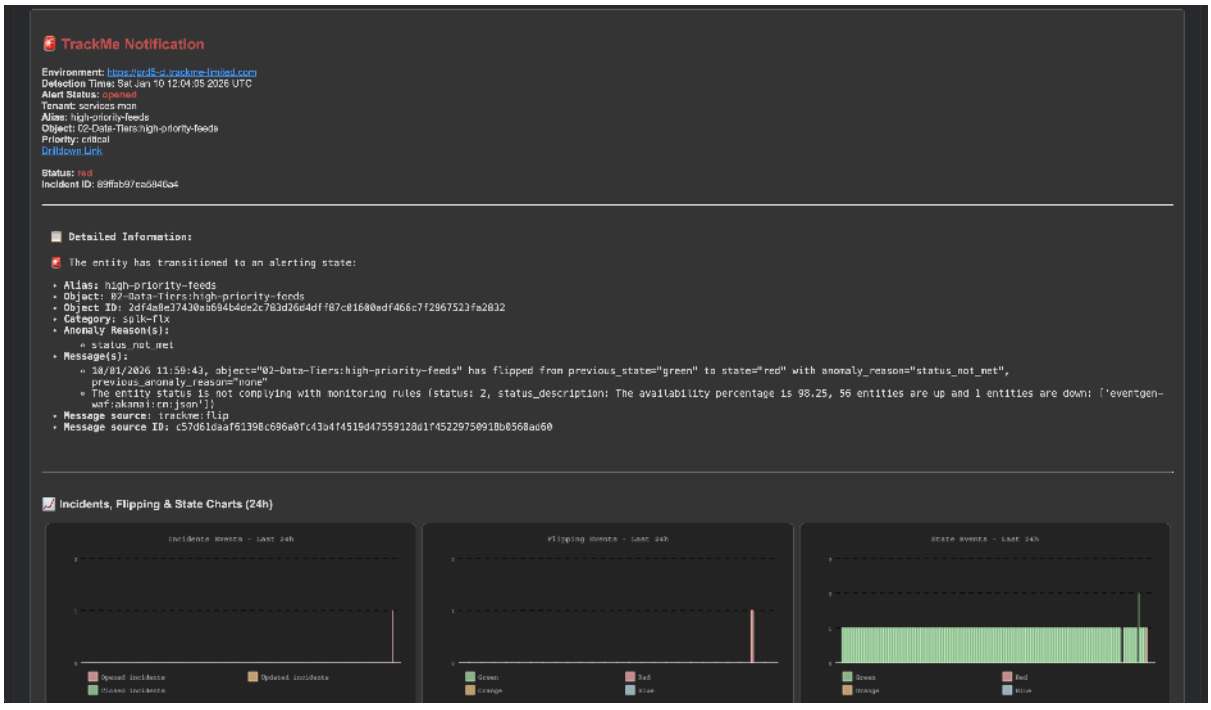

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
About Outliers Anomaly Detection
Last 24 hours
Manage outliers detection
0
3
0
0
0
1
0
Outliers Anomaly Detection



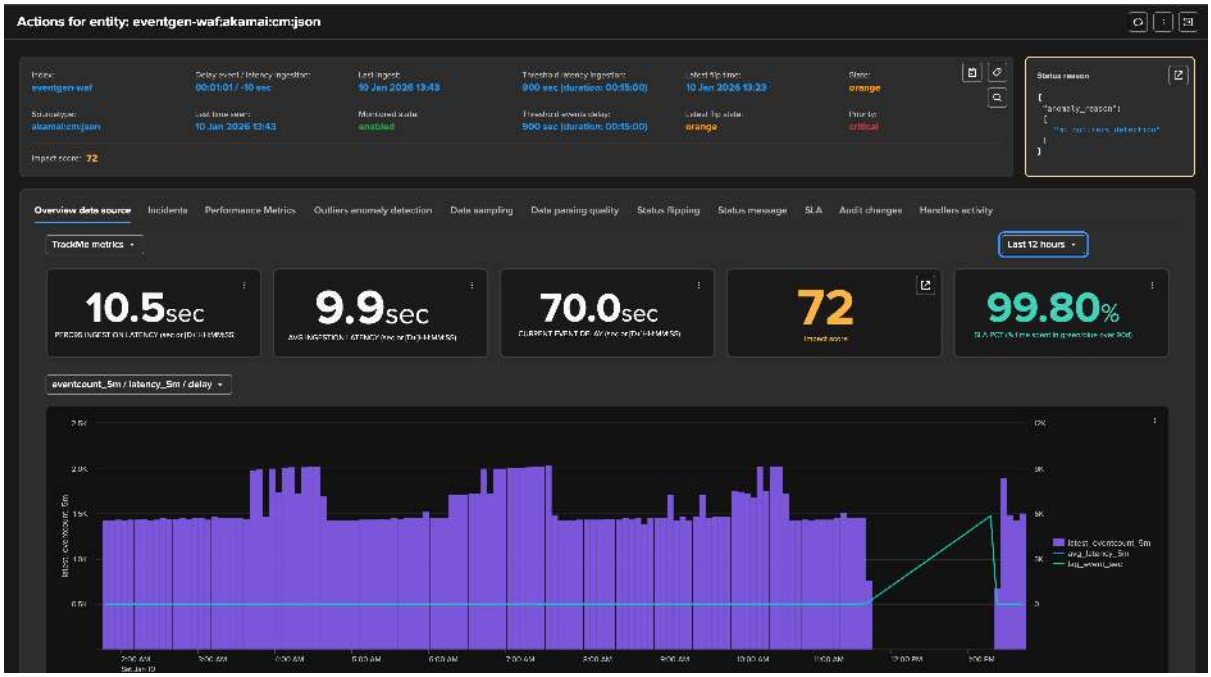


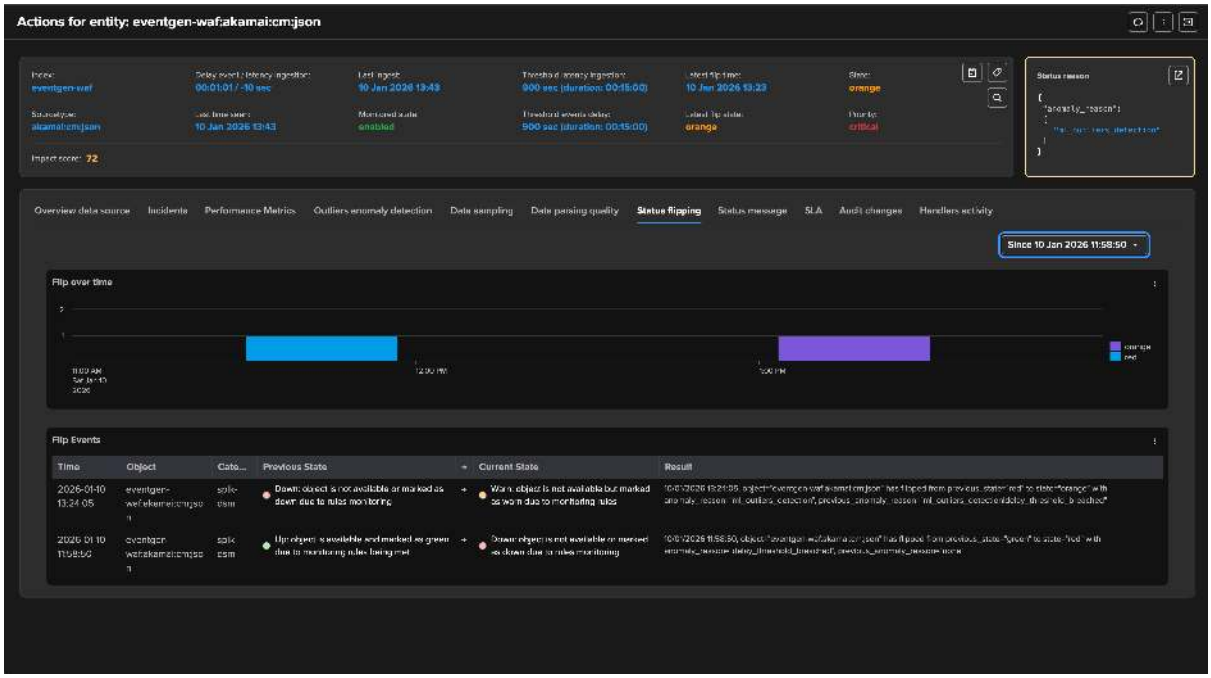
If alerting has been setup, TrackMe would generate an alert with this content:

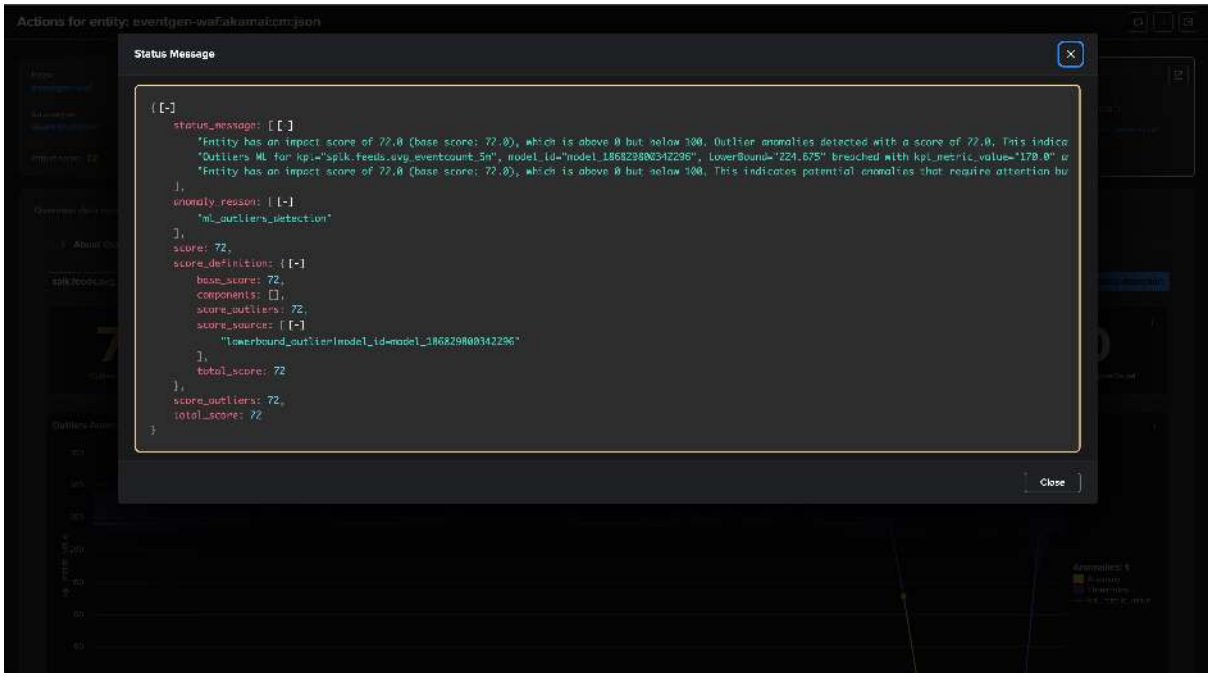


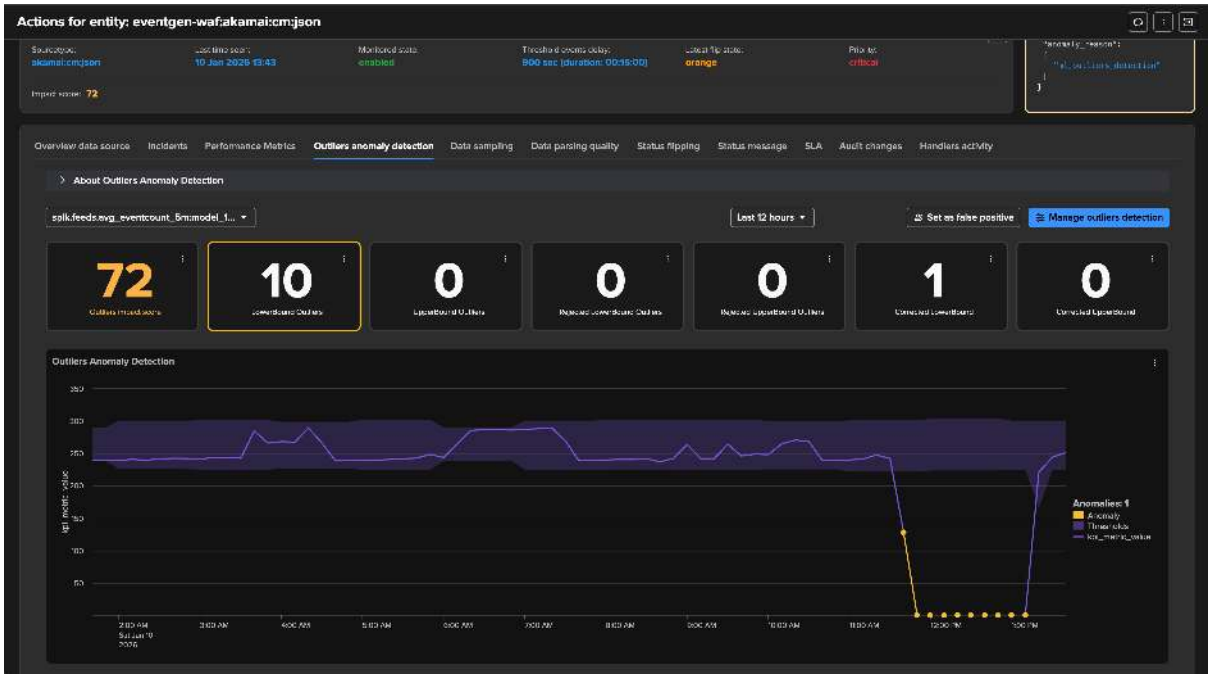


At some point, the issue was resolved and the data is back in Splunk:

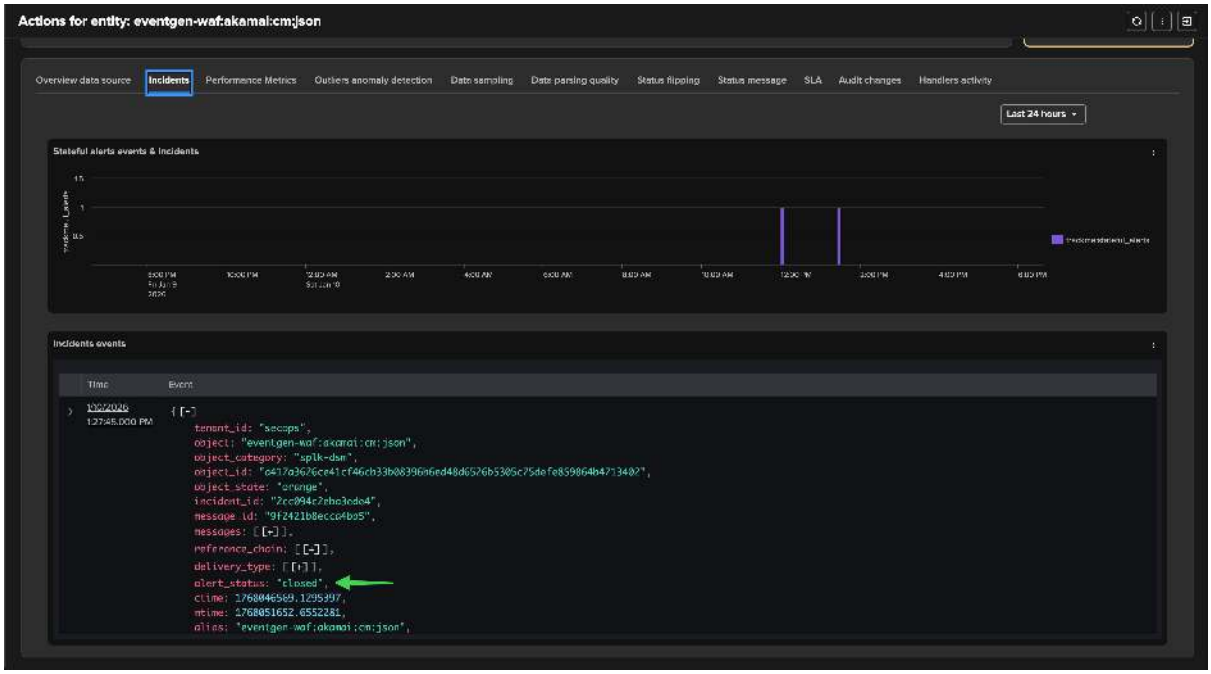

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
TrackMe metrics
Last 12 hours
10.5sec
9.9sec
70.0sec
72
99.80%
eventcount_5m / latency_5m / delay

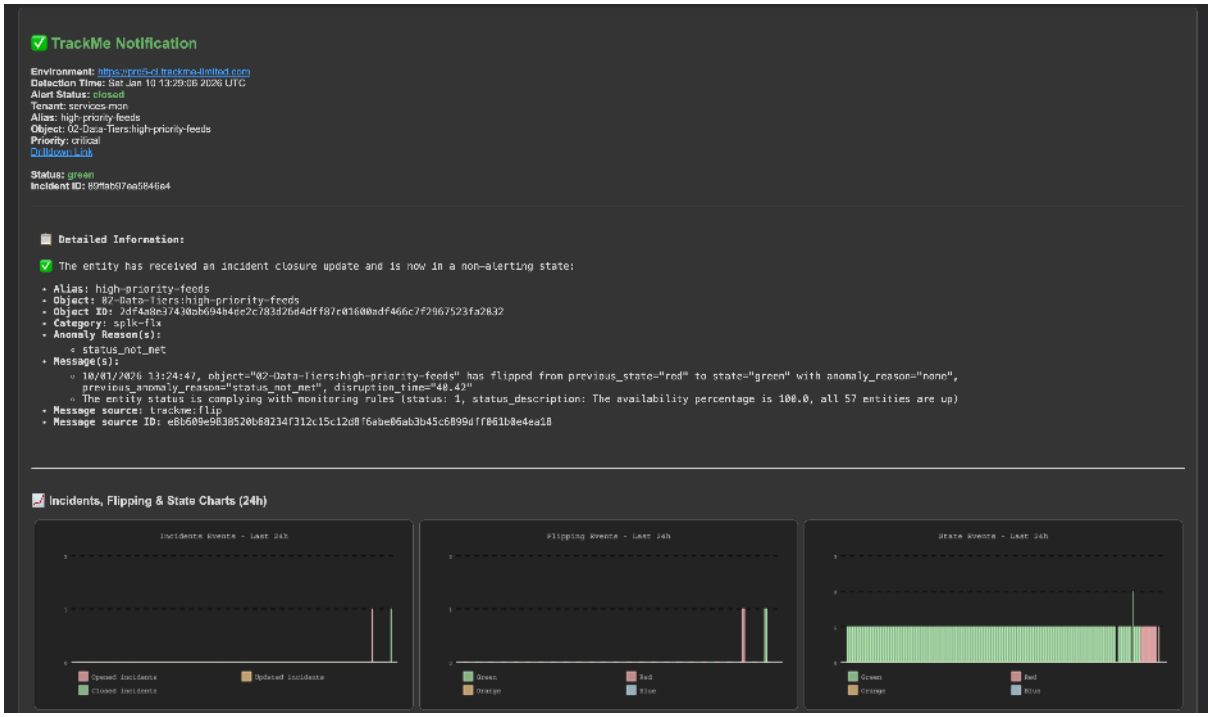

Actions for entity: eventgen-waf:akamai:cm:json
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Since 10 Jan 2026 11:58:50
Flip over time
Flip Events
Time
Object
Cate...
Previous State
Current State
Result





The alert is resolved and a closure notification is generated and sent:





About Adaptive and Variable Delay Thresholding

In addition with the threshold concept above, TrackMe provides two advanced delay threshold management approaches:

Adaptive Delay Thresholding (Machine Learning)

TrackMe leverages Machine Learning to adapt the delay threshold based on the knowledge accumulated over time on a per entity basis, and depending on the conditions.

This is called "Adaptive Delay Thresholding" and is widely documented here: *DSM — Data Source Monitoring*.

This feature is very valuable to reduce the administration costs by automatically defining the best suitable values according to the entity behaviour, it can increase and decrease thresholds in function of the events and statuses.

Variable Delay Thresholding (since TrackMe v2.3.12)

Variable delay thresholds allow the delay threshold for an entity to change dynamically based on the day of the week and the hour of the day. This allows defining tighter thresholds during business hours and more relaxed thresholds during quiet periods such as nights and weekends.

This feature is widely documented here: *Variable delay*.

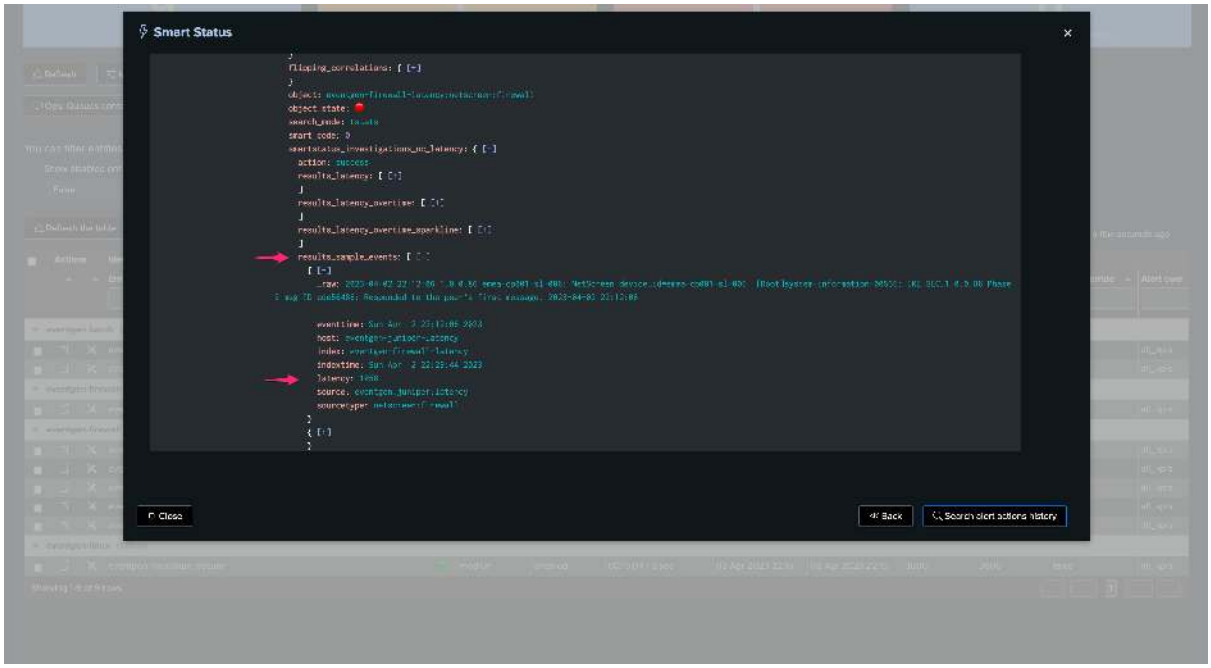
Adaptive delay and Variable delay are mutually exclusive

An entity can use either adaptive delay (static policy with ML-driven threshold adjustments) or variable delay (time-aware threshold slots), but not both simultaneously.

Use Case: Feed Indexing Performance Issues Detection

Understanding Latency in Splunk

Latency is a critical metric that measures the time gap between when an event is generated and when it becomes available in Splunk. Think of it as the "travel time" for your data:



Why Latency Matters

When latency issues occur, they can cause significant problems:

- **Search Inconsistency**: Searches run at different times may return different results
- **Delayed Insights**: Critical information arrives later than expected
- **Resource Impact**: Can indicate underlying system problems

Common Causes of Latency

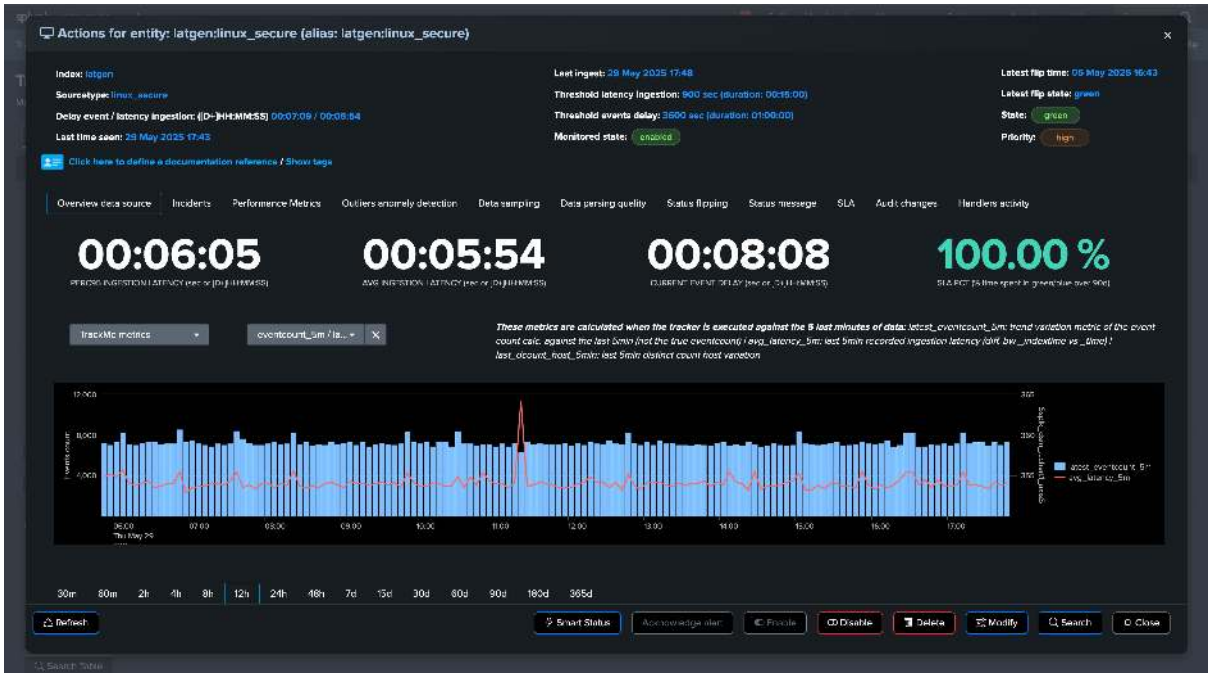
Latency issues can stem from various sources:

- Queue congestion
- Resource constraints
- Network bottlenecks
- System overload
- Configuration issues

How TrackMe Detects Latency Issues

TrackMe monitors latency by comparing the indexed time against the event time. When latency exceeds your configured threshold, TrackMe triggers an anomaly with `anomaly_reason="latency_threshold_breached"`.

Let's consider the following entity, currently the latency is very low and the entity is considered as healthy:





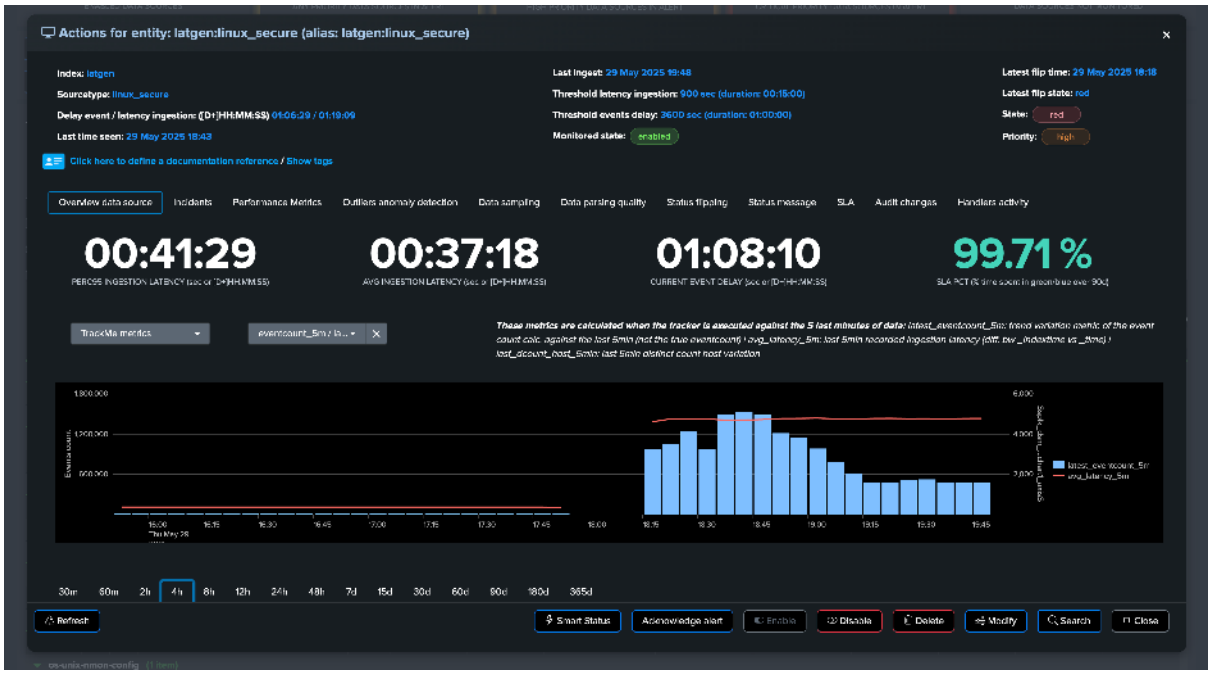
Suddenly, the entity starts experiencing latency issues. We begin receiving events with a significant time gap between their generation and indexing:

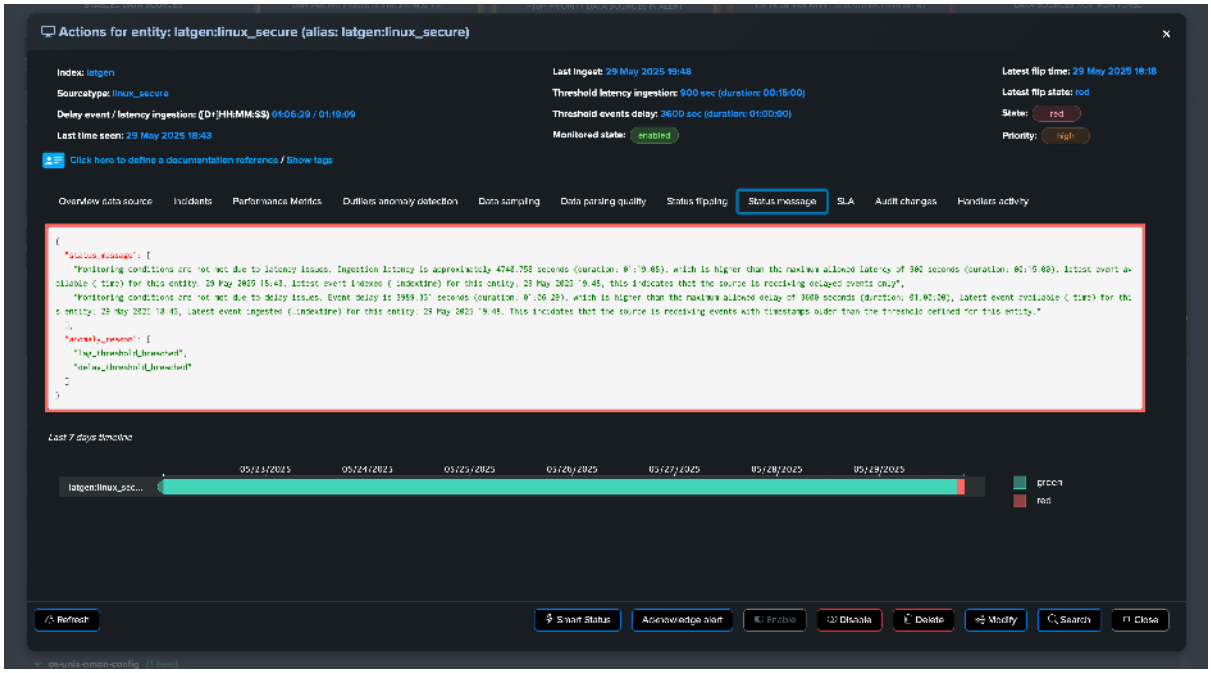
Some additional comments about latency:

- It is likely that latency will also affect the delay, but this is not necessarily the case.
- You can well be receiving a mix of real-time or quasi-real-time events, while at the same time receiving events made available with a large latency.
- This can be the case for a variety of reasons, from data feeds catching up situations to simply having collectors

suffering from performance issues while others sending to the same context do not.

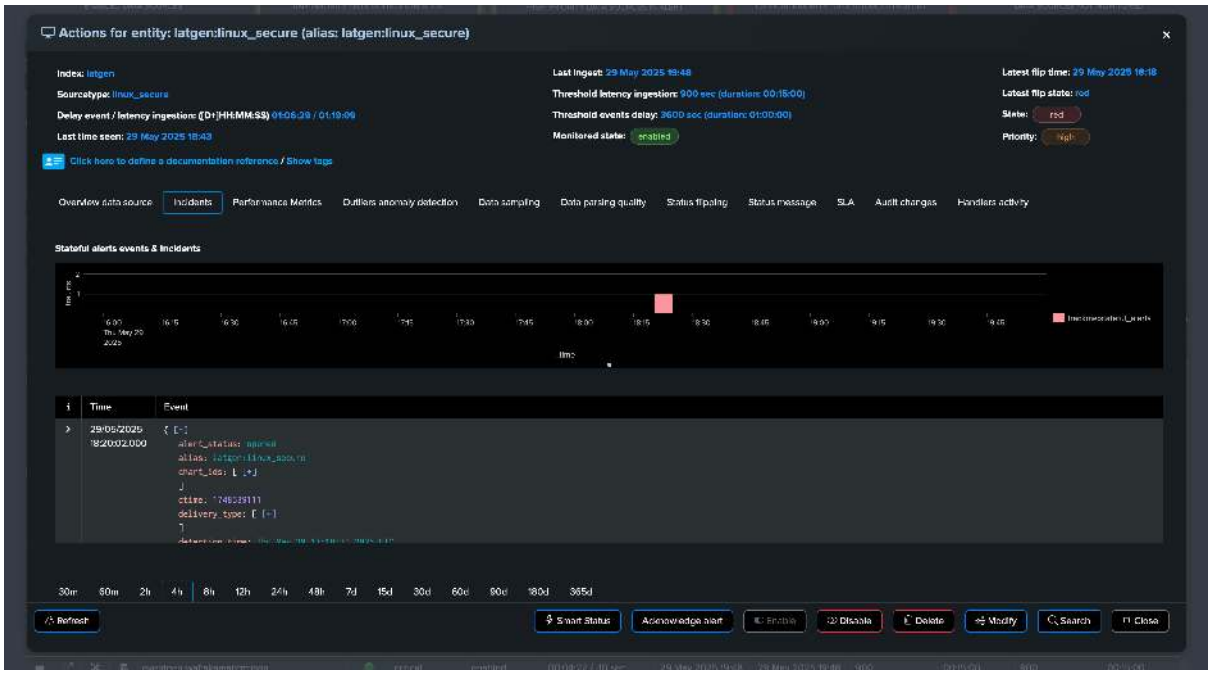
The situation escalates as the entity experiences high latency combined with a dramatic increase in event volume:

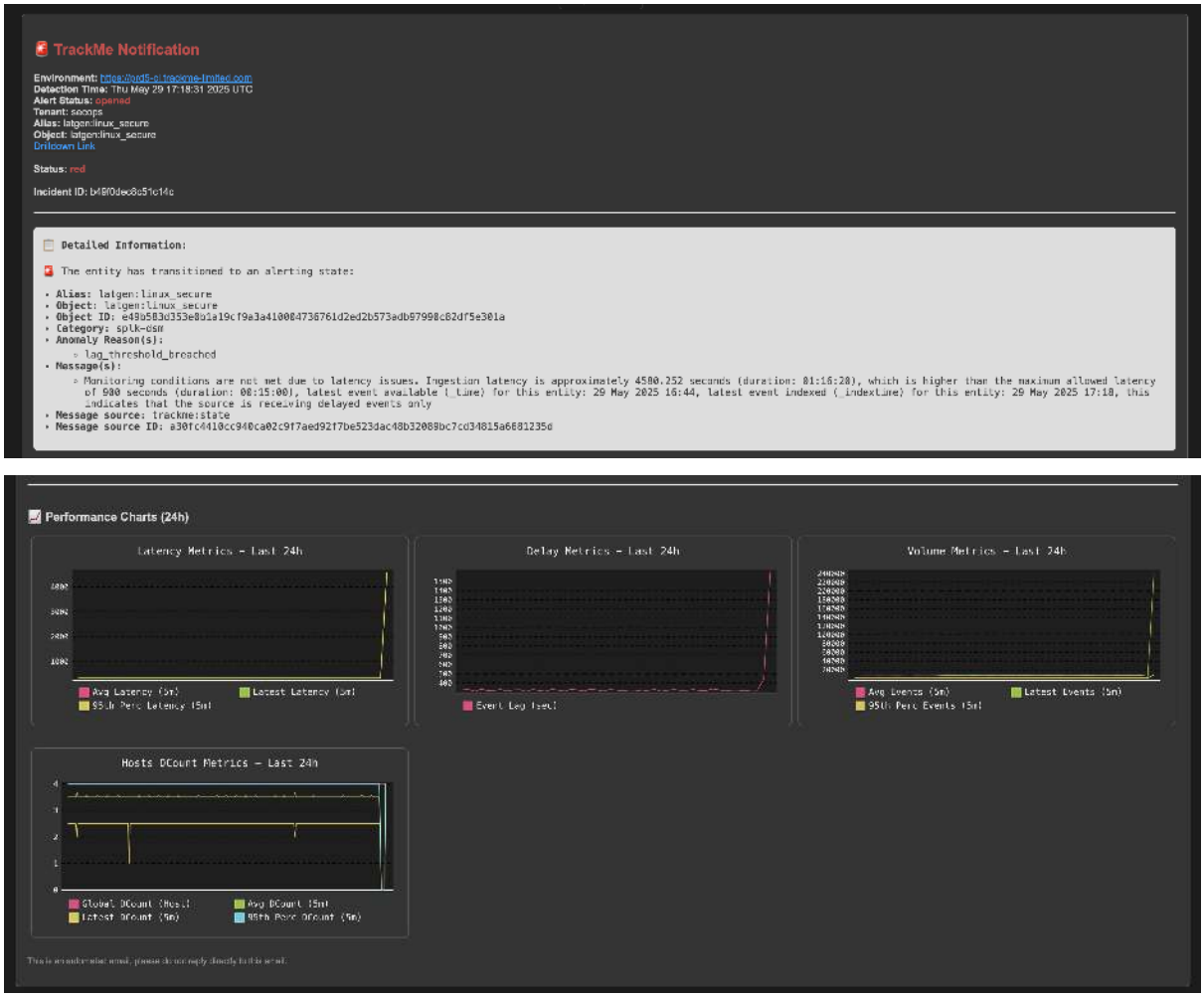




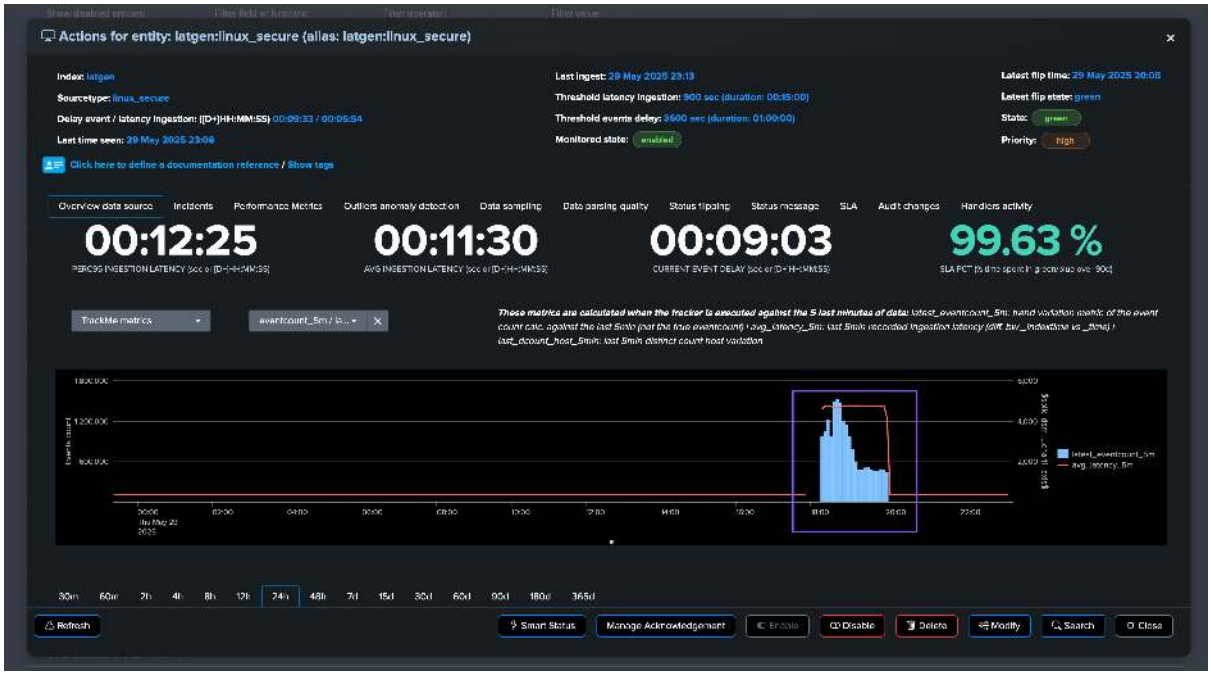


TrackMe immediately responds by opening an incident and sending a notification:

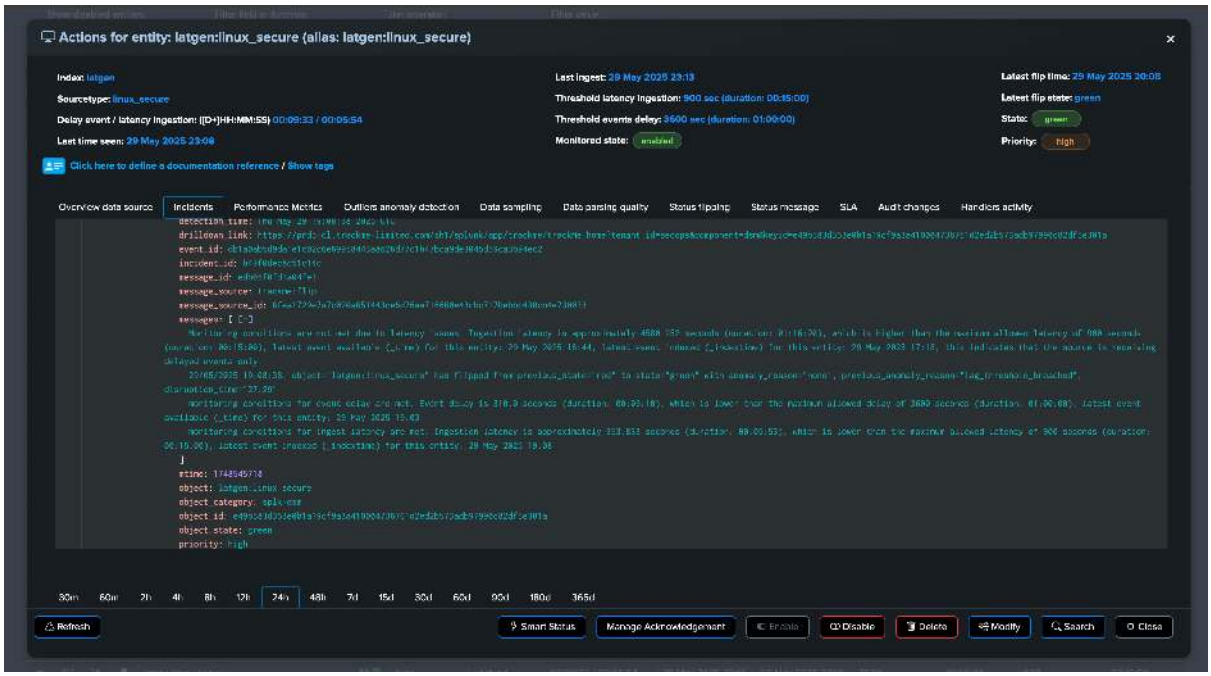




After the issue is resolved, the entity returns to a healthy state and the incident is automatically closed:




Actions for entity: latgen:linux_secure (alias: latgen:linux_secure)
Index: latgen
Sourcetype: linux_secure
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:09:33 / 00:05:54
Last time seen: 29 May 2025 23:08
Last ingest: 29 May 2025 23:13
Threshold latency ingestion: 900 sec (duration: 00:15:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 29 May 2025 20:08
Latest flip state: green
State: green
Priority: high
Click here to define a documentation reference / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Latency metrics (timechart average)
Metrics notes:
Metrics f_latency_5m are driven by the ingest and calculated using the 5 last minutes of indexed data when the tracker is executed
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Manage Acknowledgement
Disable
Delete
Modify
Search
Close

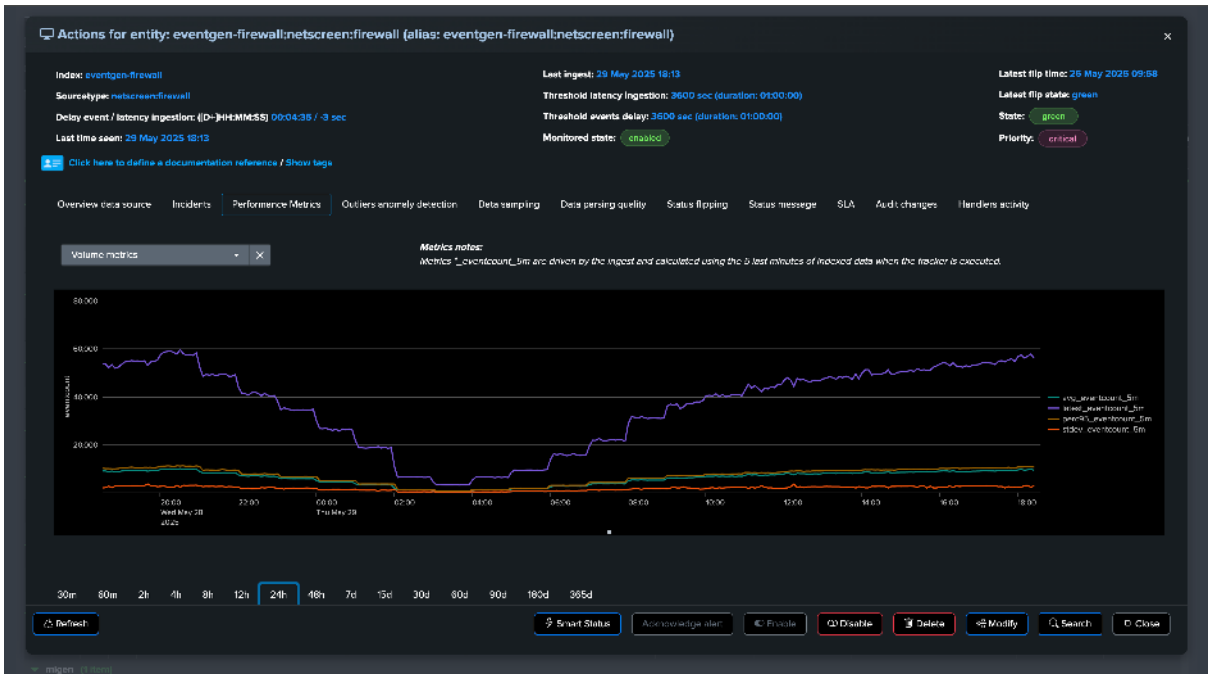

Actions for entity: latgen:linux_secure (alias: latgen:linux_secure)
Index: latgen
Sourcetype: linux_secure
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:09:33 / 00:05:54
Last time seen: 29 May 2025 23:08
Last ingest: 29 May 2025 23:13
Threshold latency ingestion: 900 sec (duration: 00:15:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 29 May 2025 20:08
Latest flip state: green
State: green
Priority: high
Click here to define a documentation reference / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Manage Acknowledgement
Disable
Delete
Modify
Search
Close

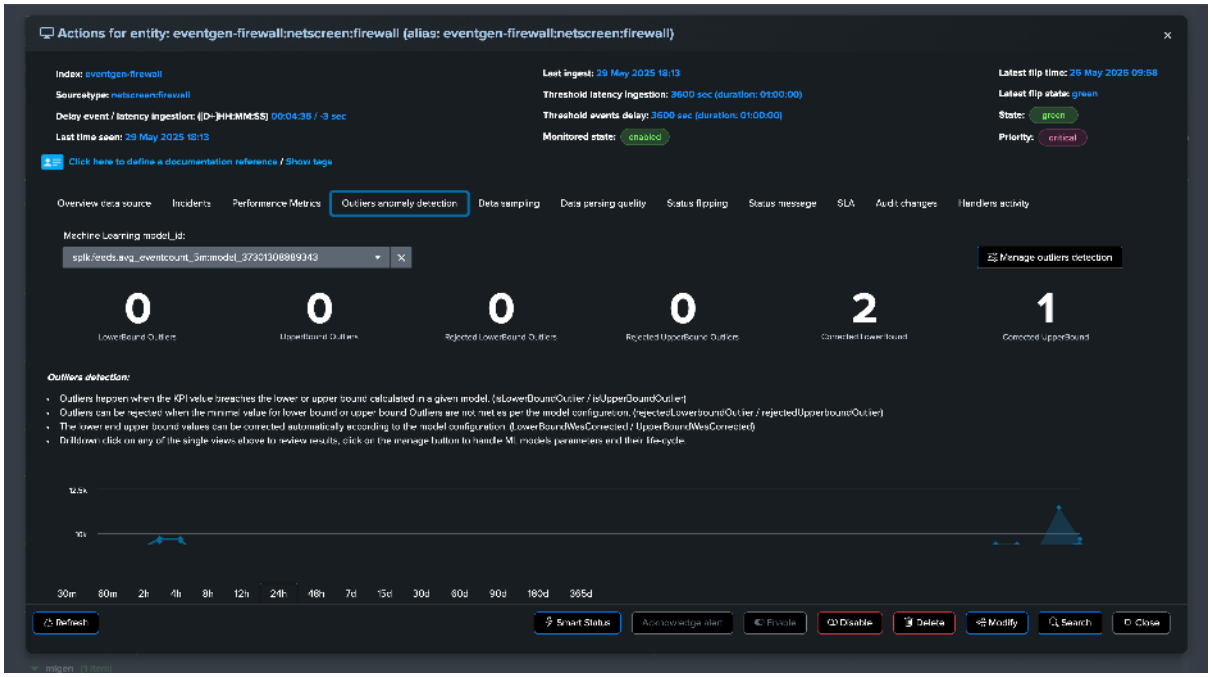

TrackMe Notification
Environment: https://prd5-el.trackme-limited.com
Detection Time: Thu May 29 19:08:38 2025 UTC
Alert Status: closed
Tenant: scoops
Alias: latgen:linux_secure
Object: latgen:linux_secure
Drilldown Link
Status: green
Incident ID: b4f0dec8c51c14c
Detailed Information:
The entity has received an incident closure update and is now in a non-alerting state:
Alias: latgen:linux_secure
Object: latgen:linux_secure
Object ID: e49a583d353e80a1a19cf9a3a4180047361d2ed2b573adb97990c82df5e301a
Category: splk-dsm
Anomaly Reason(s):
none
Message(s):
29/05/2025 19:08:38, object="latgen:linux_secure" has flipped from previous_state="red" to state="green" with anomaly_reason="none", previous_anomaly_reason="lag_threshold_breached", disruption_time="27.29"
monitoring conditions for event delay are met. Event delay is 318.0 seconds (duration: 00:05:18), which is lower than the maximum allowed delay of 3600 seconds (duration: 01:00:00), latest event available (_time) for this entity: 29 May 2025 19:03
monitoring conditions for ingest latency are met. Ingestion latency is approximately 353.833 seconds (duration: 00:05:53), which is lower than the maximum allowed latency of 900 seconds (duration: 00:15:00), latest event indexed (_indextime) for this entity: 29 May 2025 19:08
Message source: trackme:flip
Message source ID: 6fcb1729e7b7c020a651443ce6d76aa716660e43cbc717bc0bc430c04e730033

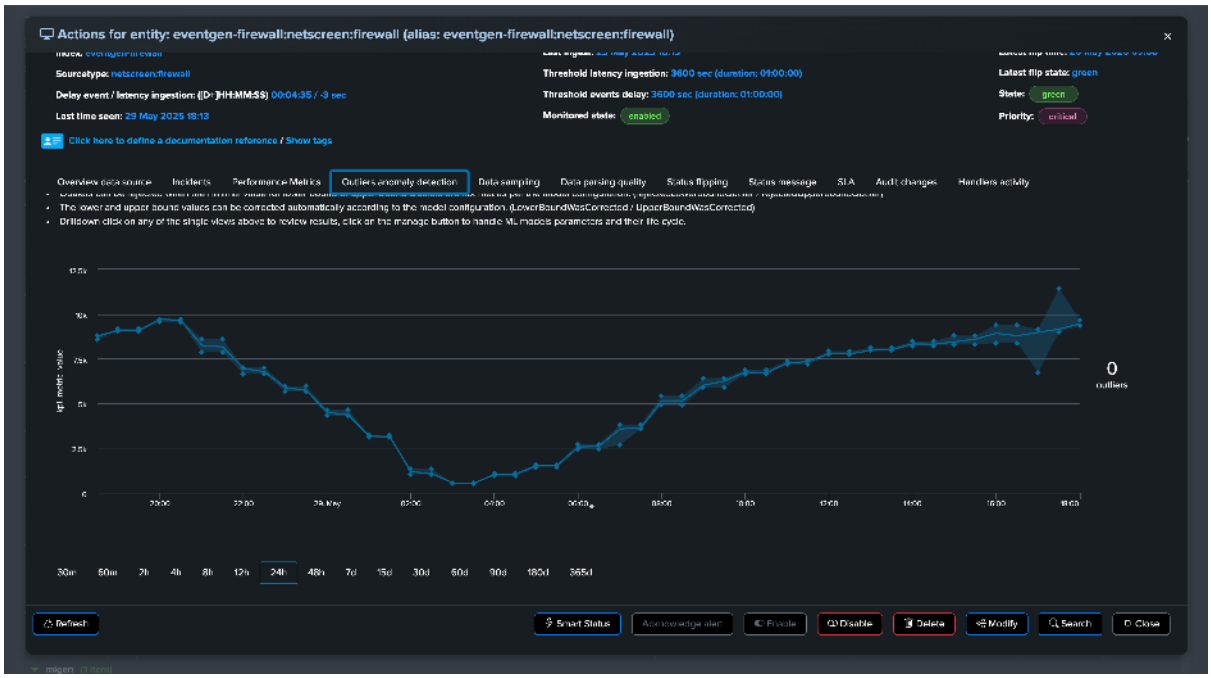
Use Case: Abnormally Low Volume Trend Detection (Outliers Detection)

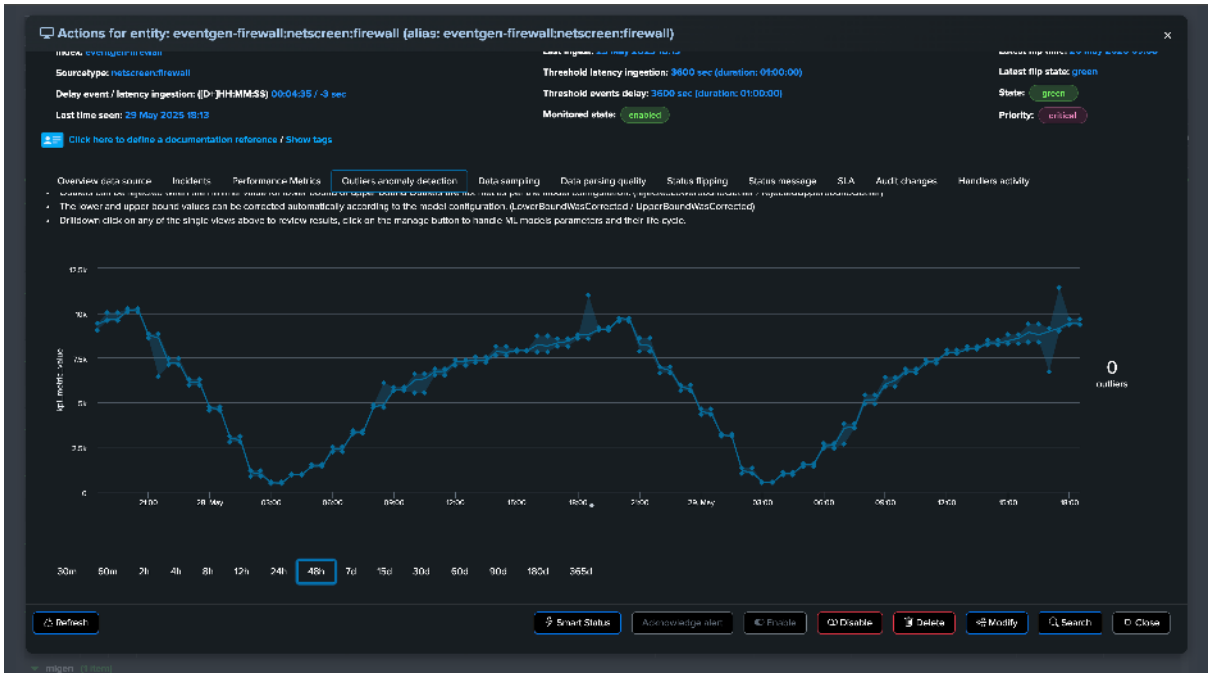
In the following example, our entity is healthy, we have data making it, not suffering from latency, and the volume is stable and expected:



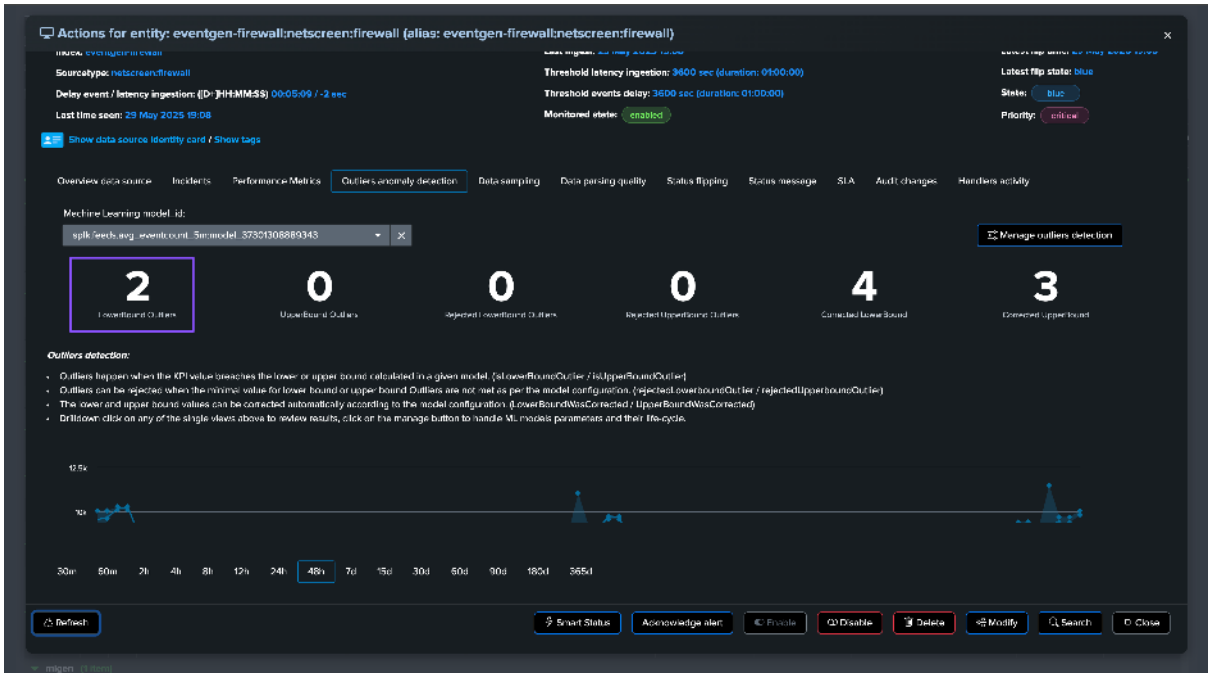


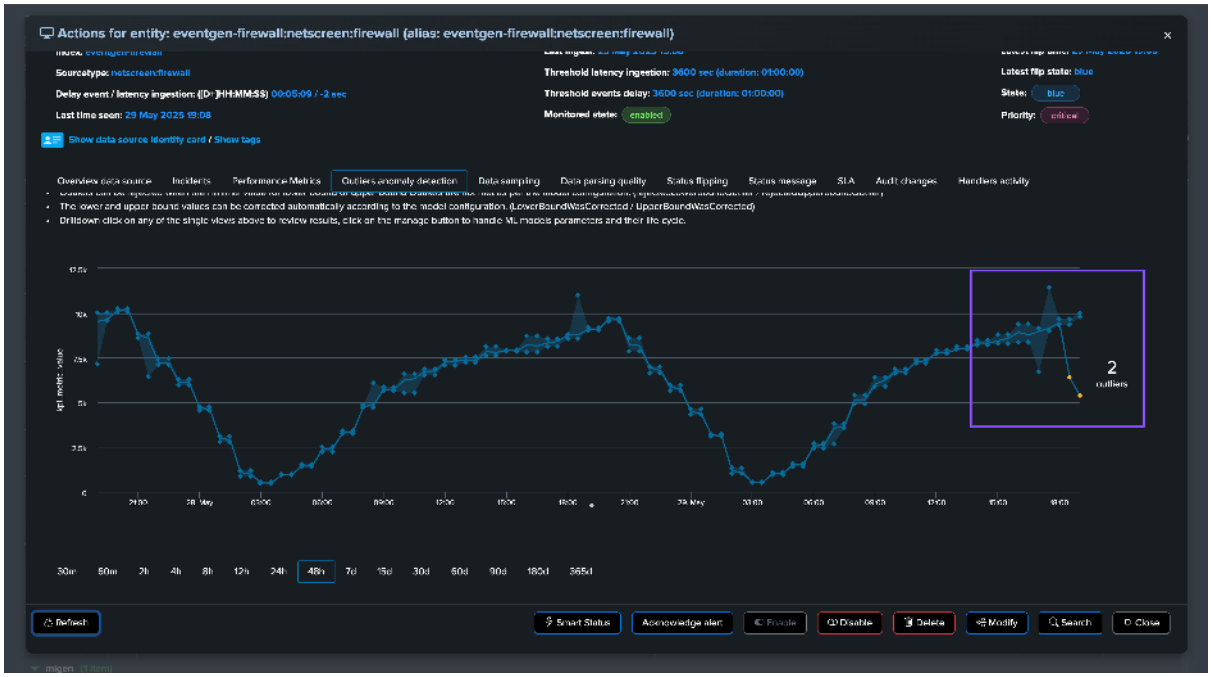

Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Outliers anomaly detection
Machine Learning model_id:
Manage outliers detection
0
0
0
0
2
1
Outliers detection:
Refresh
Smart Status
Disable
Delete
Modify
Search
Close


Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Outliers anomaly detection
0
outliers
Refresh
Smart Status
Disable
Delete
Modify
Search
Close

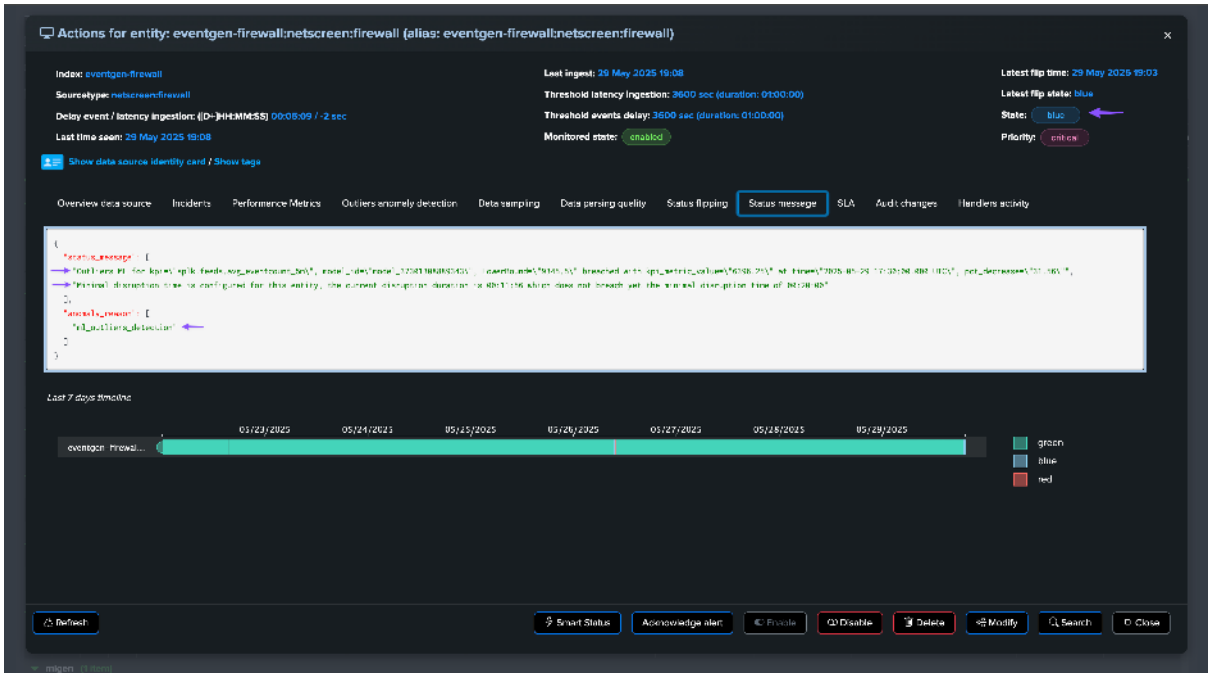


At some point, we start to see issues, we are still receiving events, but in a proportion that is not expected and indicates a potential issue:



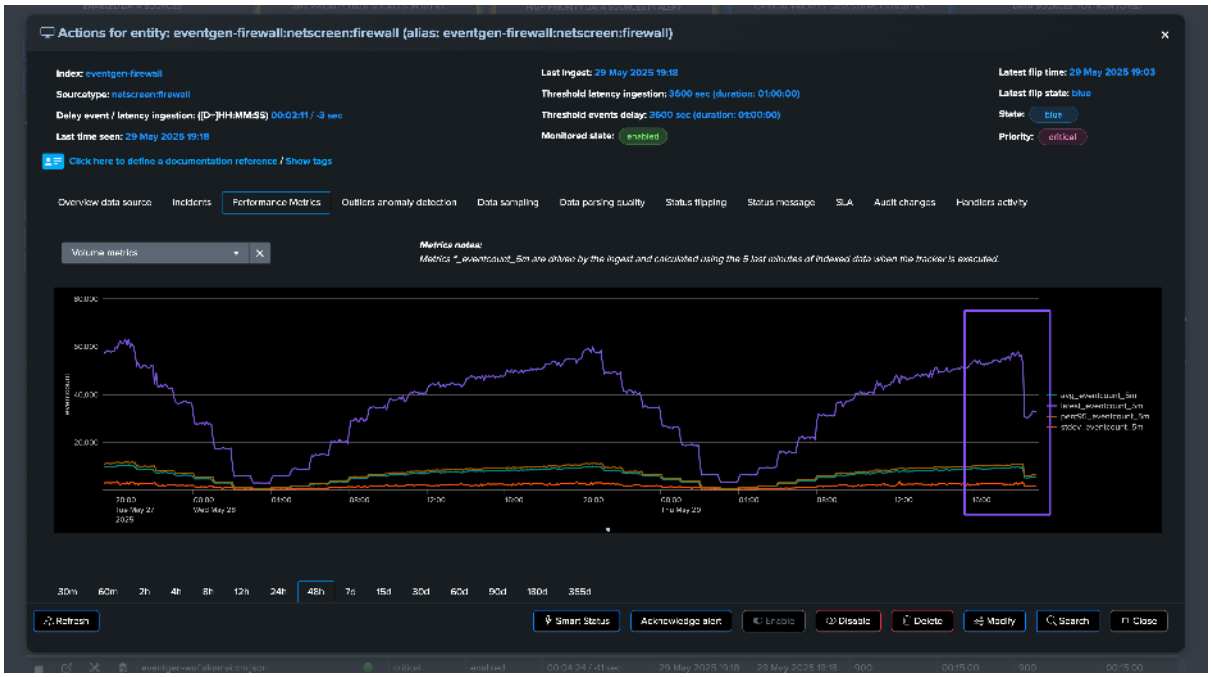


As this entity had a disruption queue configured, once the duration is breached, the entity transitioned to red and is now officially in alert:

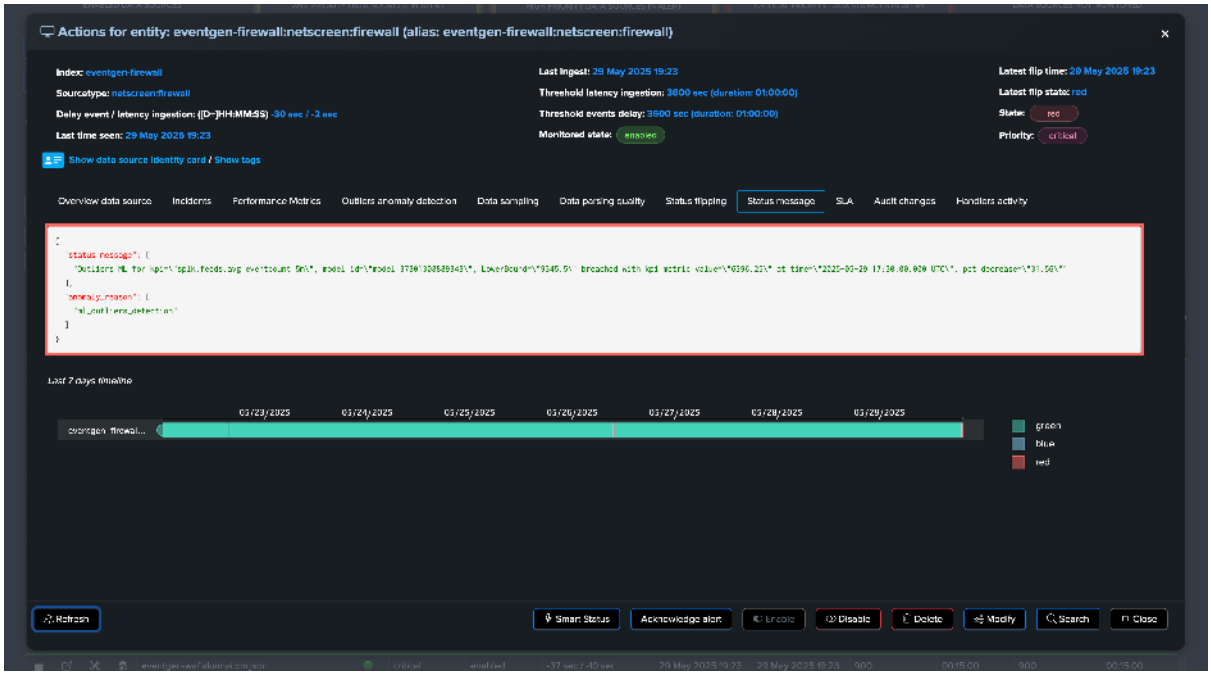


A significant amount of events is still making it to Splunk, without looking at this from the lens of Machine Learning and accumulated knowledge, we would have a hard time understanding that an abnormally low volume is happening.

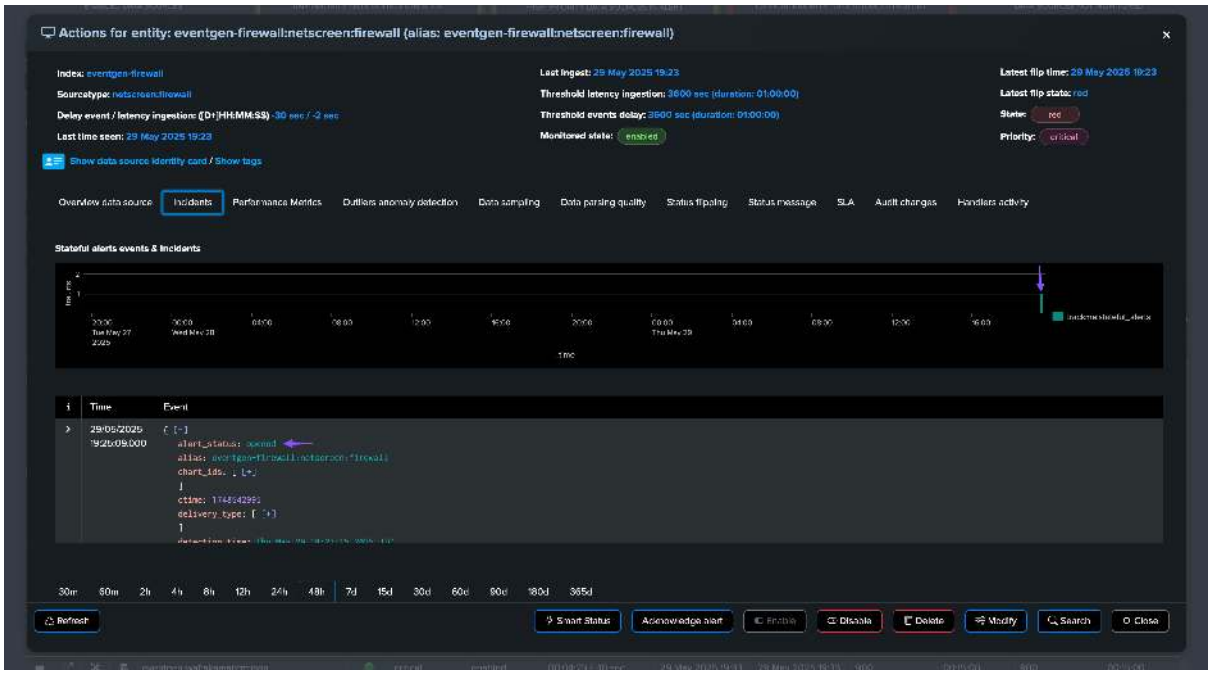


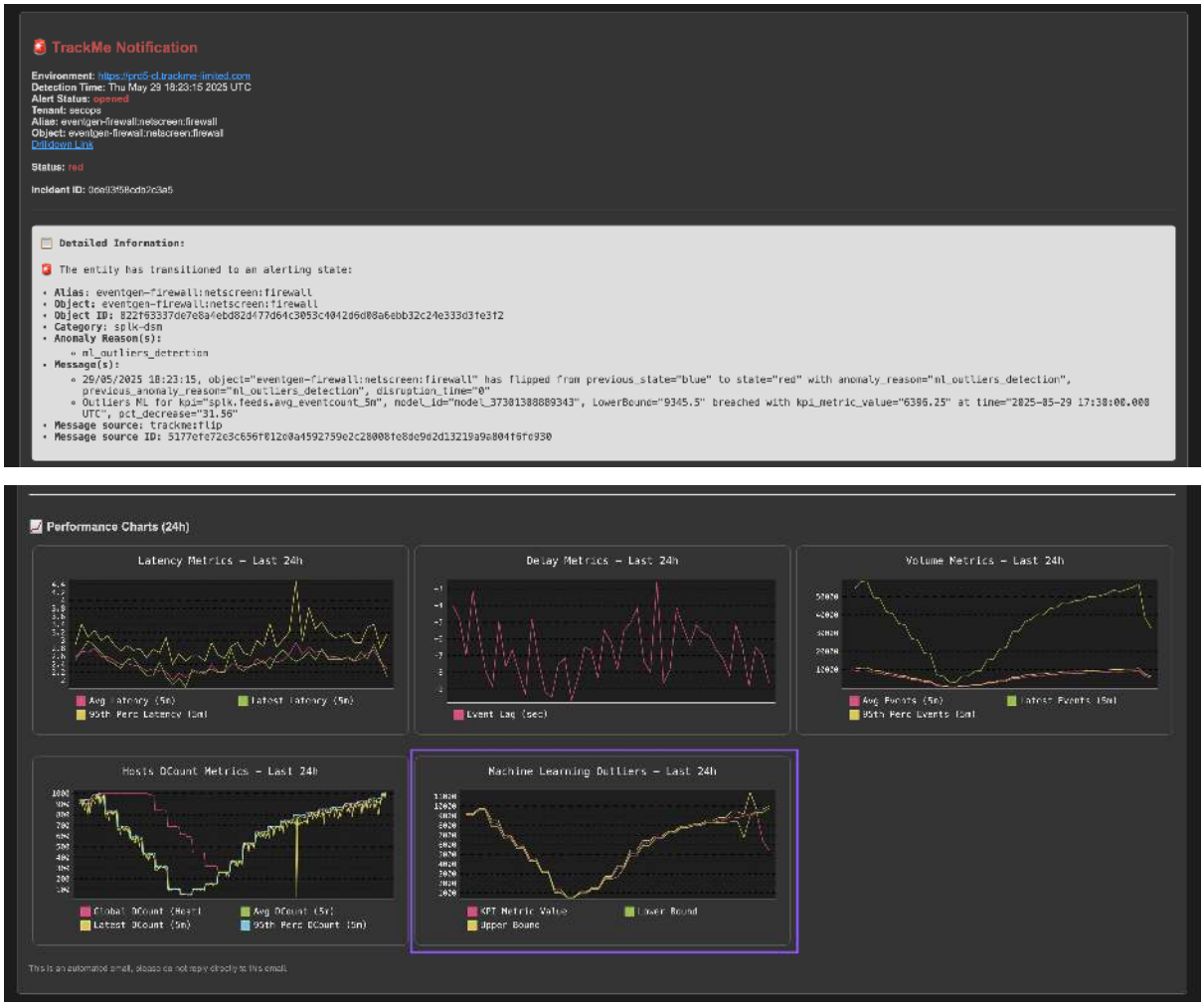


Once the disruption duration is over, the entity is now in alert:

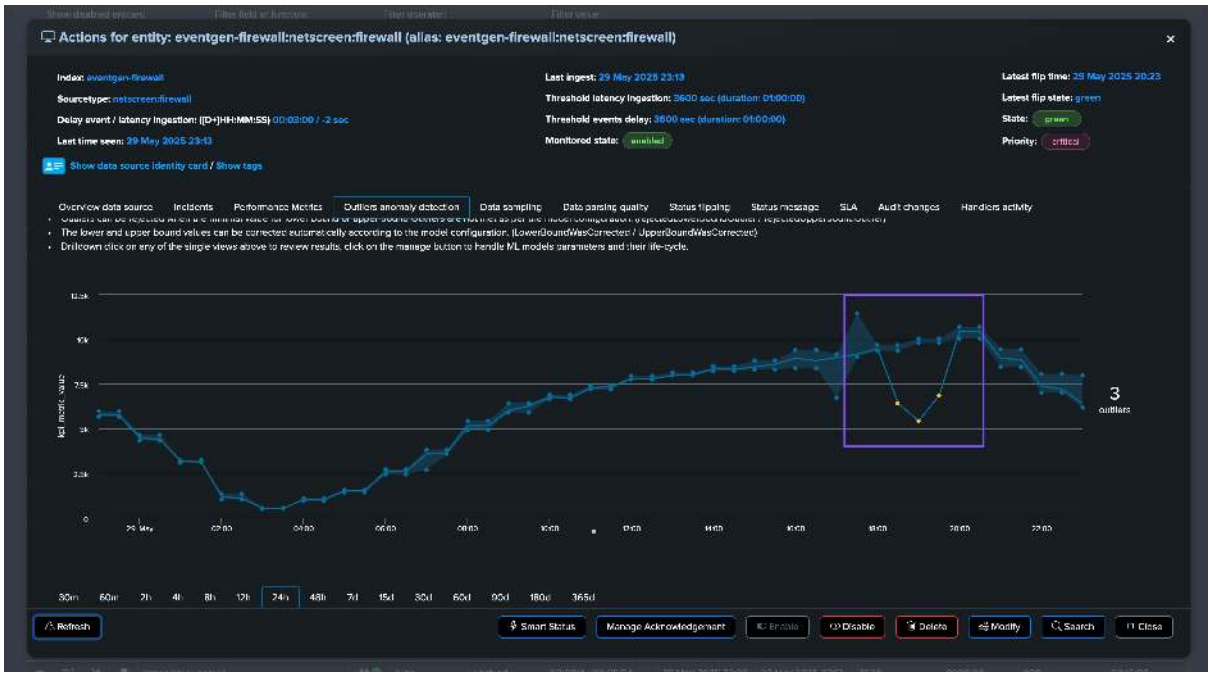


An incident is opened and a notification is sent:

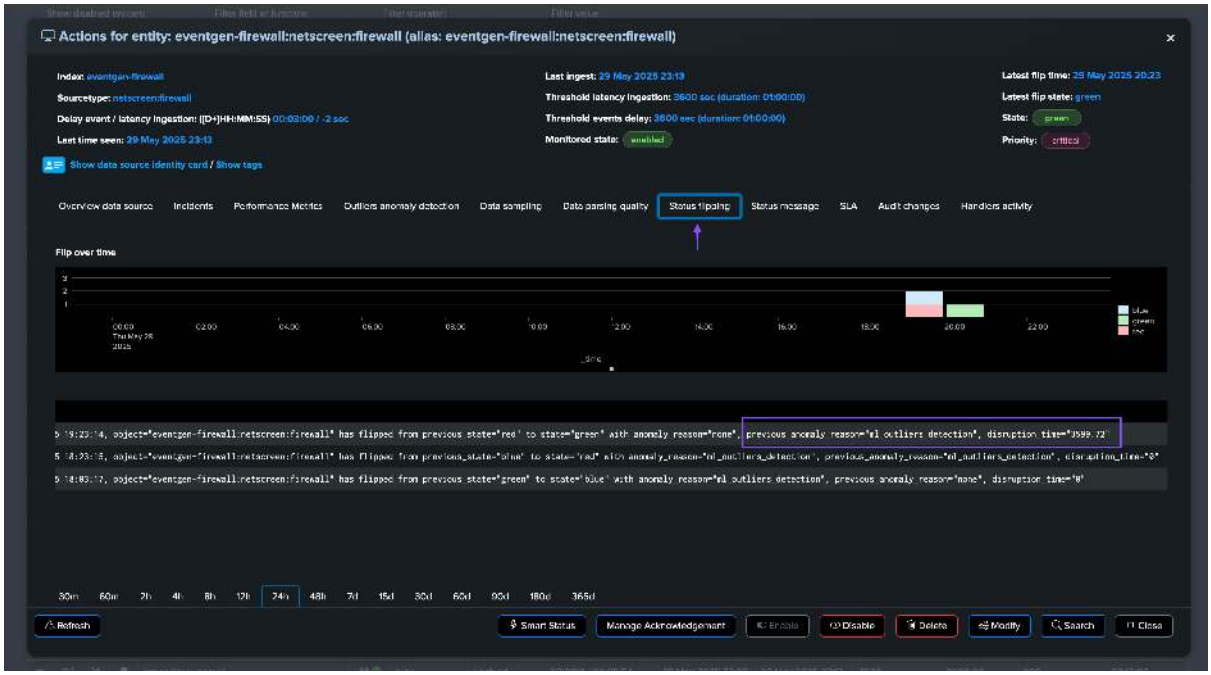


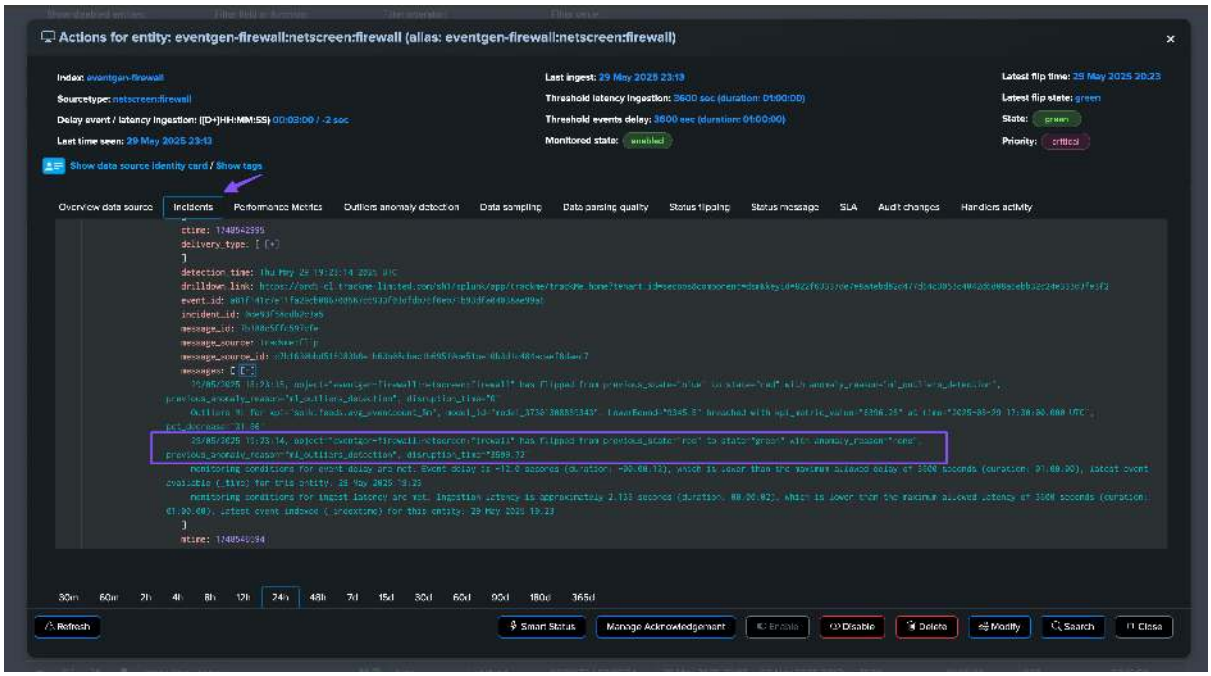


Once the issue is resolved, the entity will return to green after the Outliers anomaly condition is resolved and TrackMe has processed the ML rendering for this entity: (this can take some time depending on your settings and scale)



The incident is closed and a closure notification is sent:





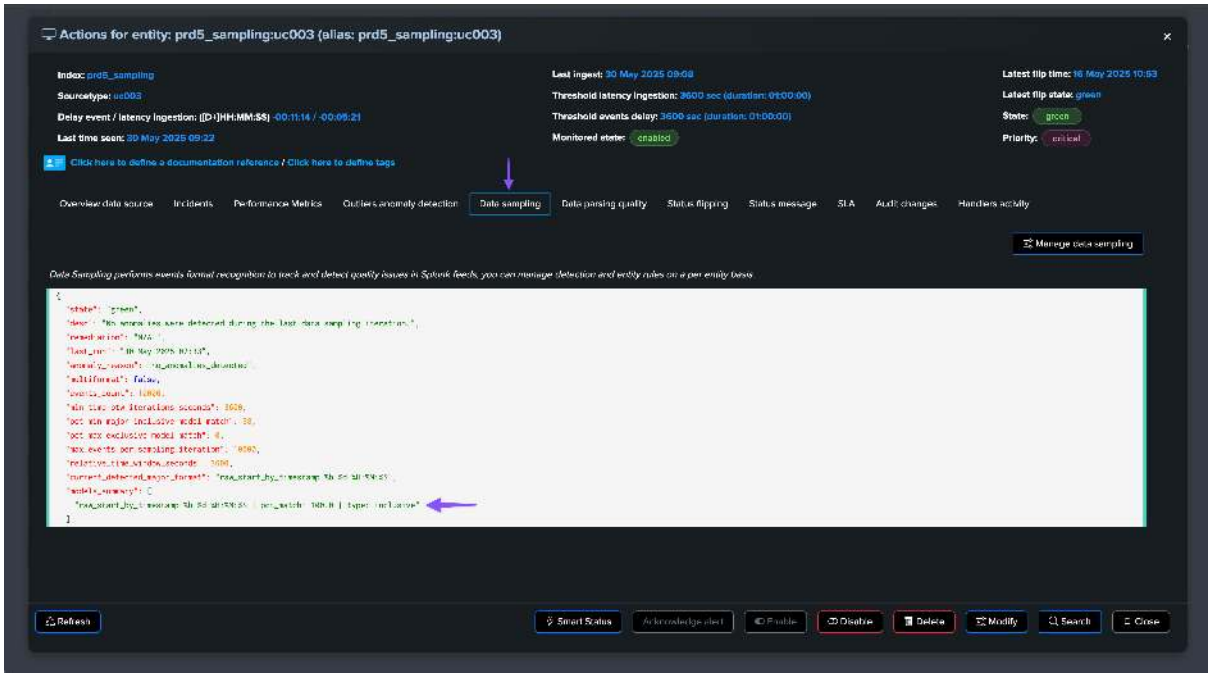
Use Case: Quality Issues Detection (Event Format Recognition)

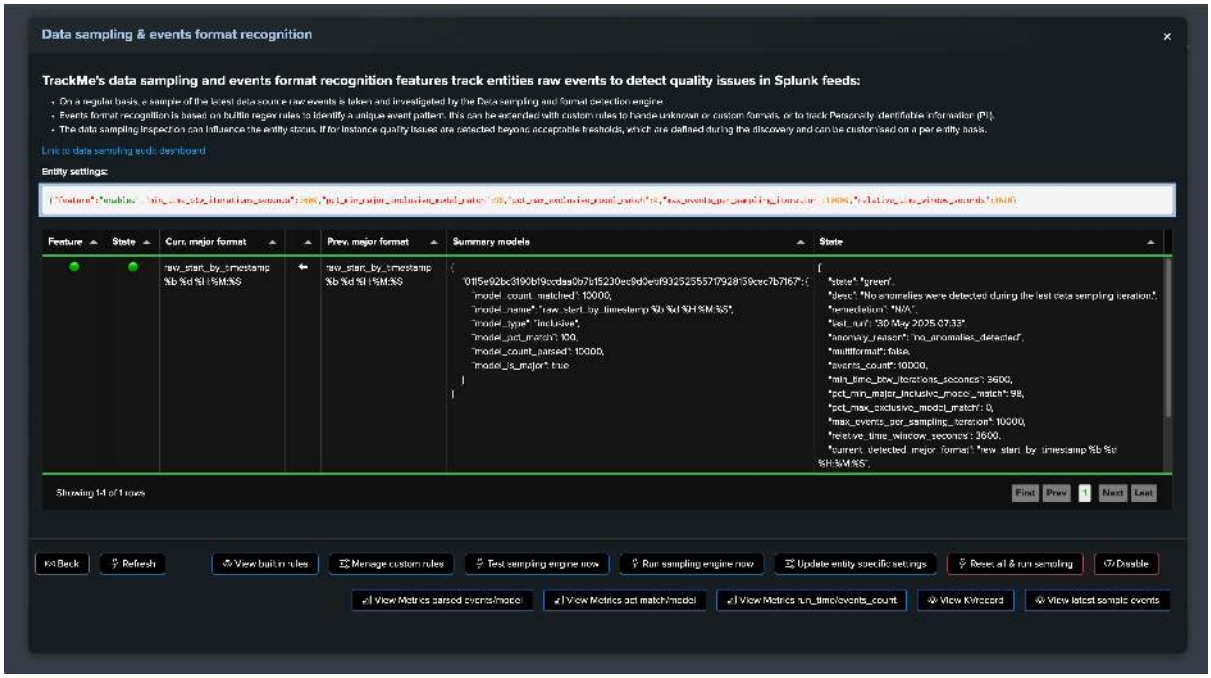
In Data Source monitoring (splk-dsm), TrackMe performs automated event format recognition using the events format recognition engine.

During this process, we will pick samples of events regularly, 10k events by default, which will be processed through the engine to detect the format of events, classified by types.

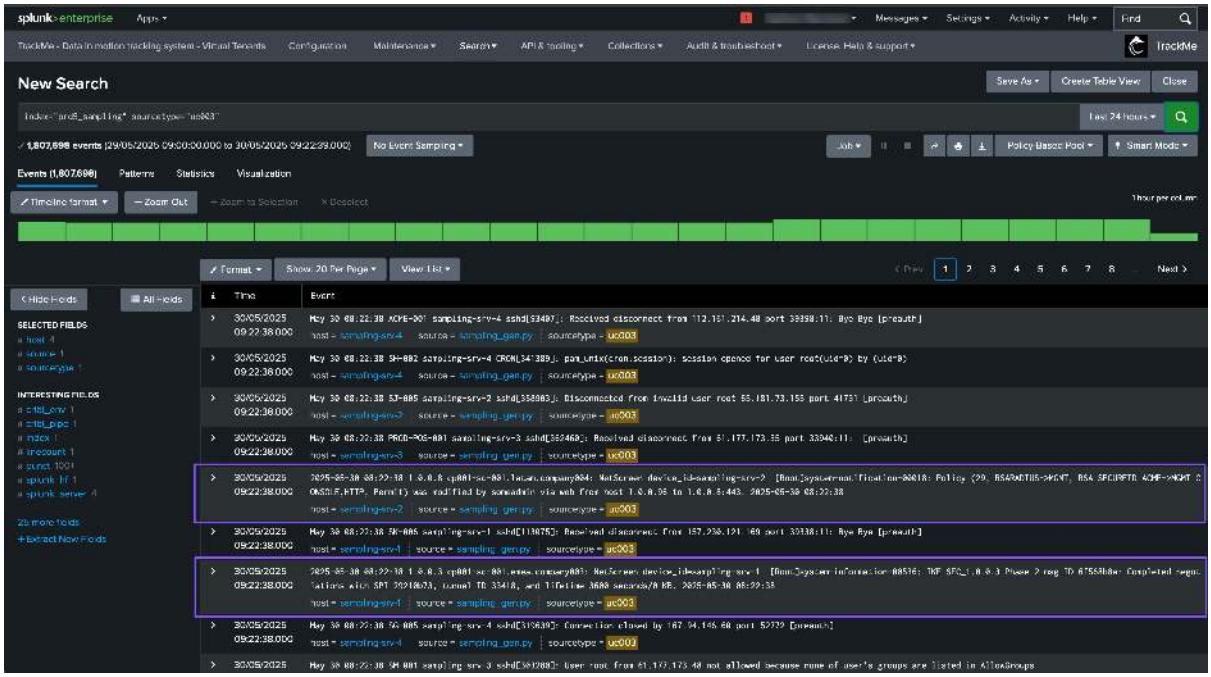
By default, TrackMe would alert against a lack of quality if it detects that the main format goes below 98% of the events being correctly classified.

Let's consider the following entity, the data sampling engine detected a single format and the entity is considered as healthy:



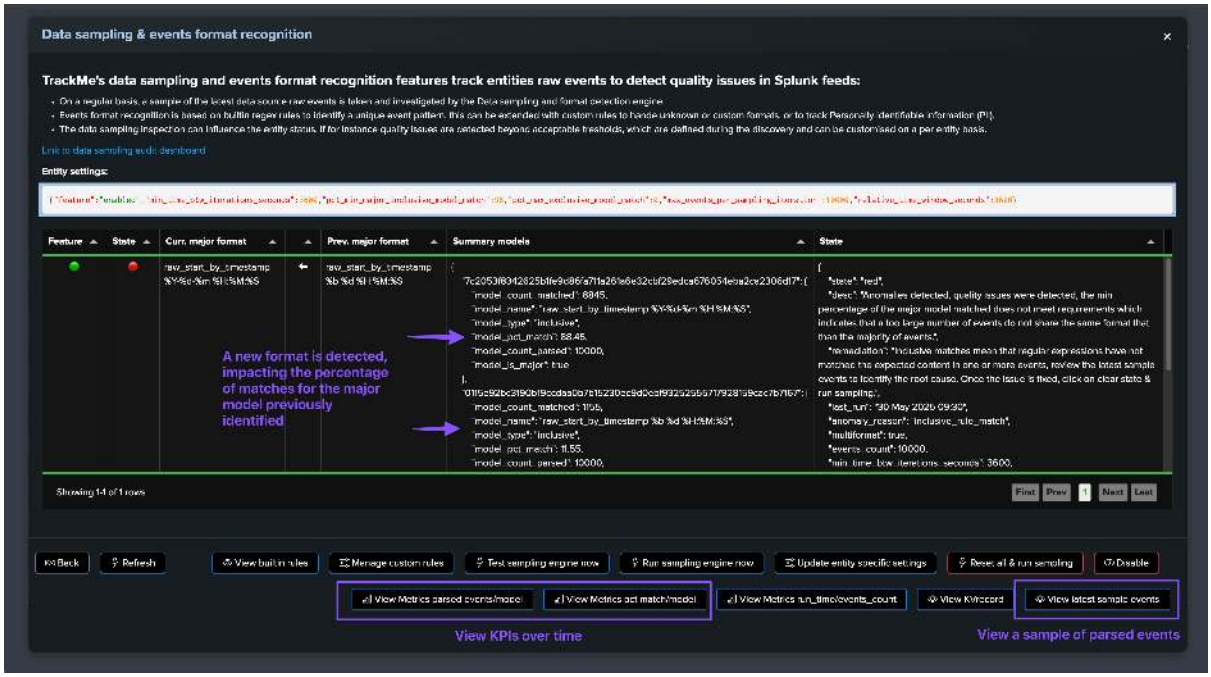


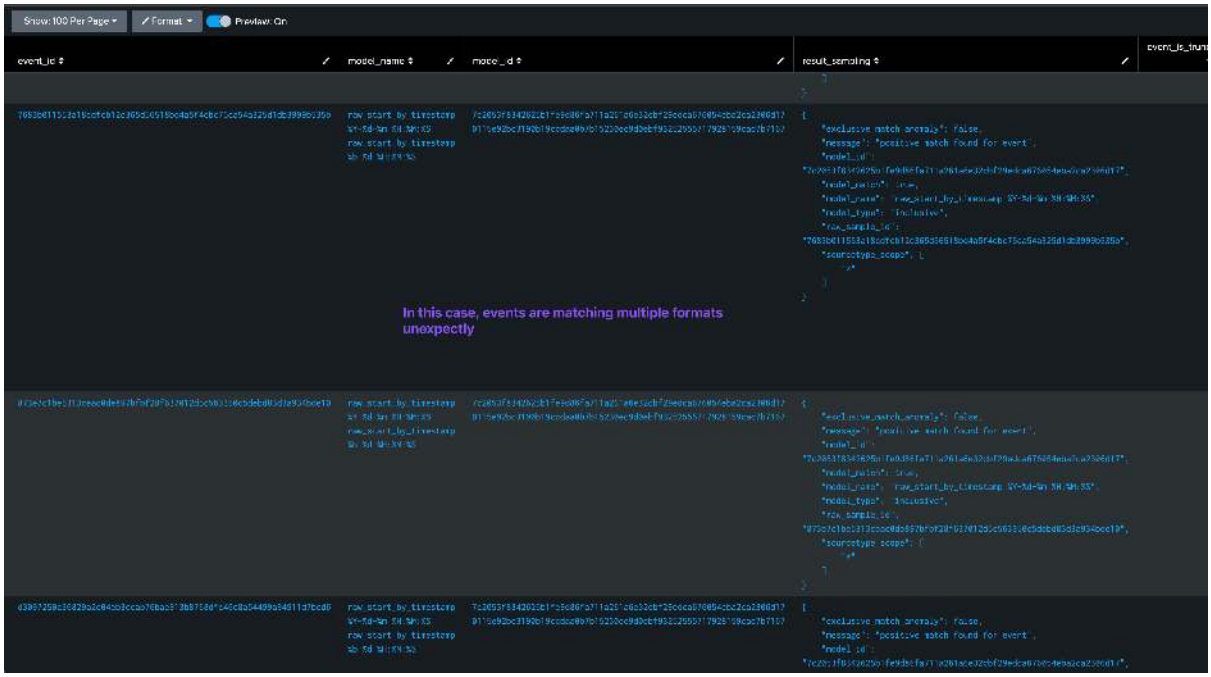
At some point, new sources are being on-boarded within the same sourcetype, however, mistakenly the expected format is not configured properly and events are making it with a different and non-expected structure:

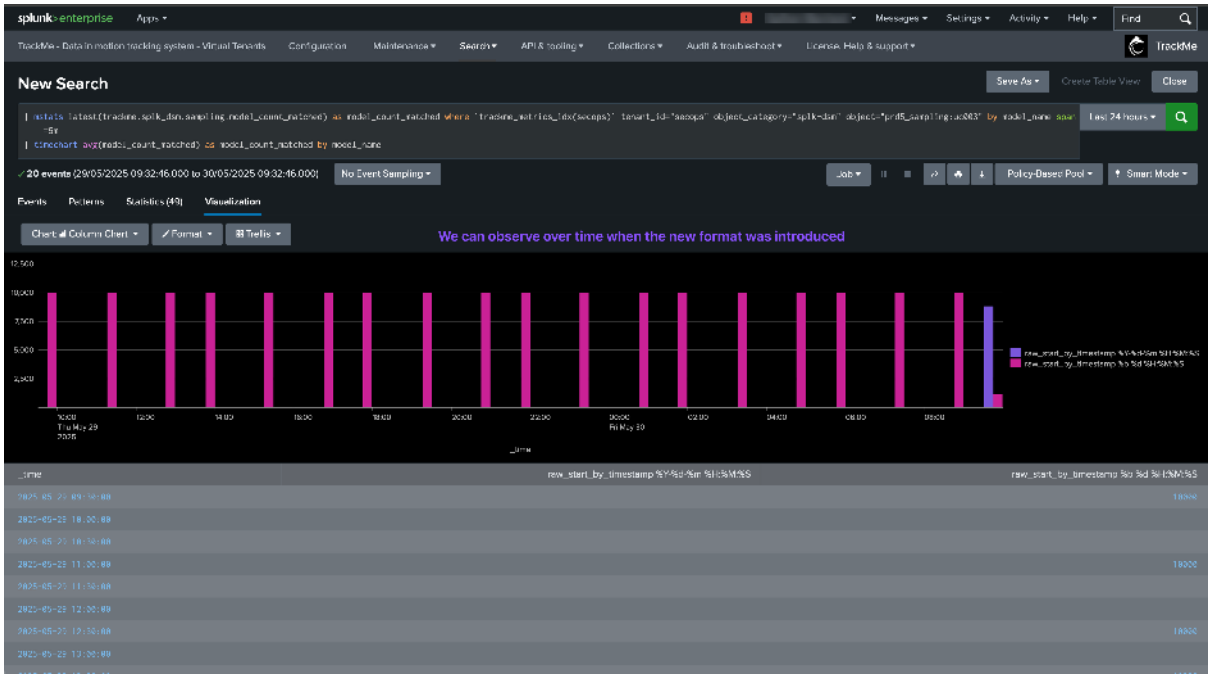


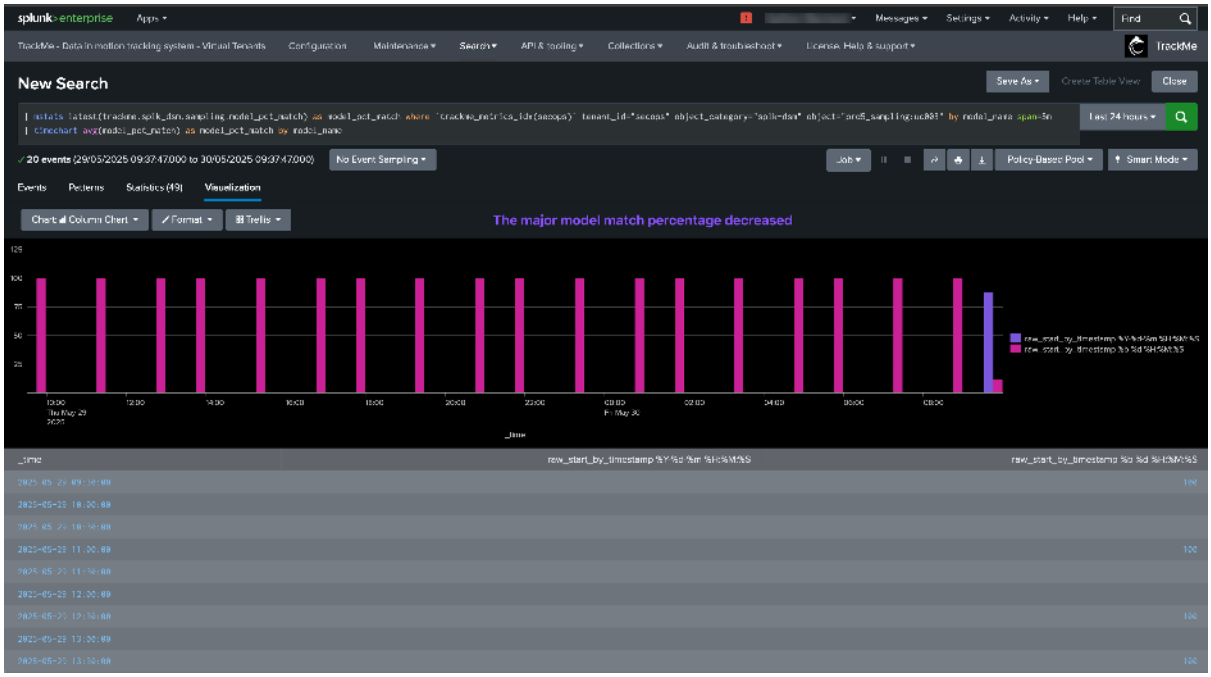
This may or may not be a strong issue, but in most use cases, receiving different types of events within the same sourcetype is not expected and leads to failures at the index time level, for instance, the timestamp extraction, or at search time with failures to extract fields for unstructured events.

The data sampling engine will detect that the format has changed and that the main format previously detected is not matching the required percentage of events, and that new event formats were introduced, leading to an alert for quality issues:

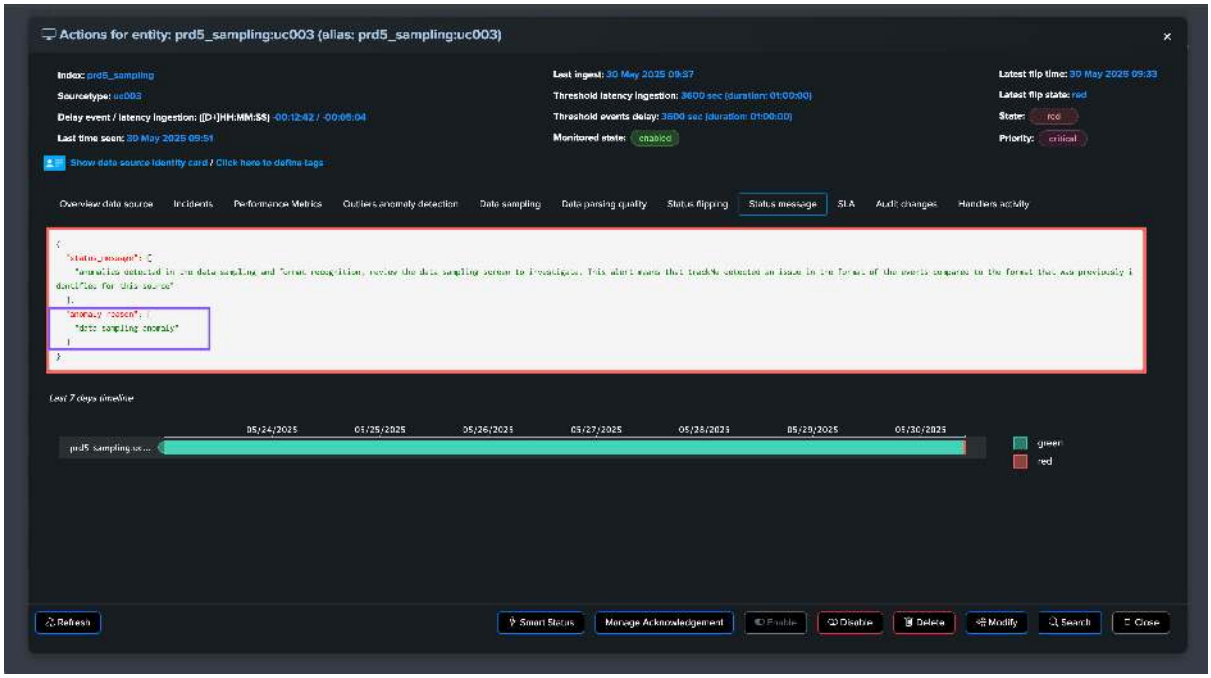


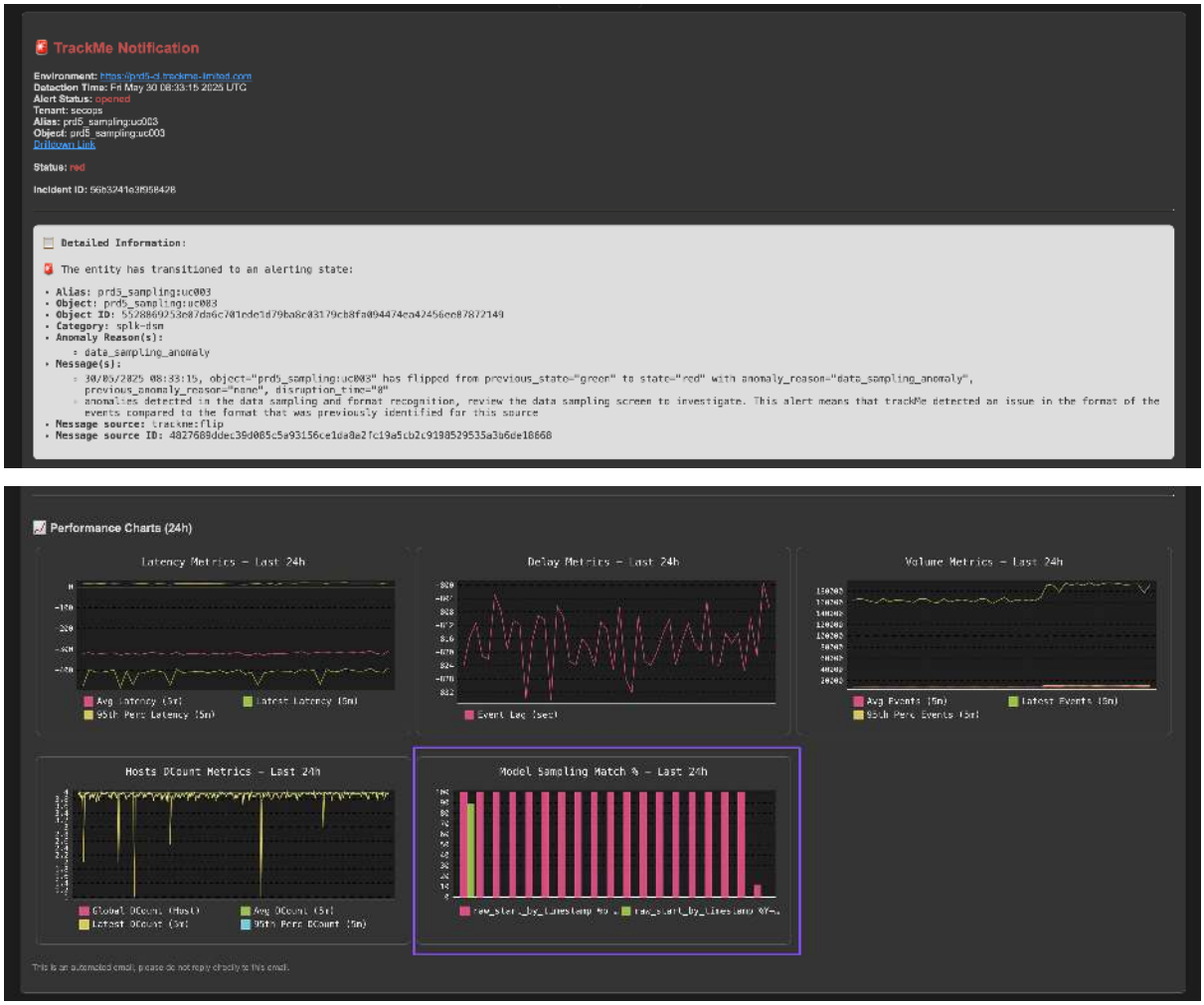

In this case, events are matching multiple formats unexpectly


New Search
We can observe over time when the new format was introduced



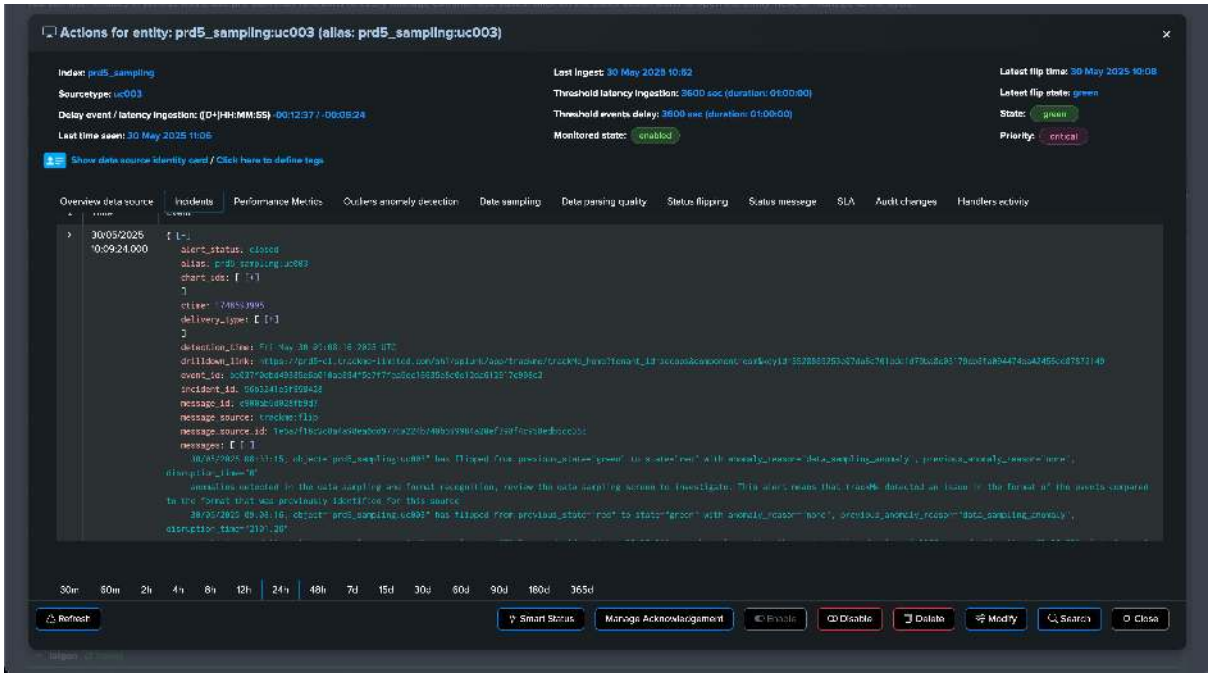
The quality issue detected influences the entity status, and an anomaly_reason="data_sampling_anomaly" is generated:





Once the root cause is identified and the issue is fixed, after some time the entity will return to green when the expected format is correctly detected again:







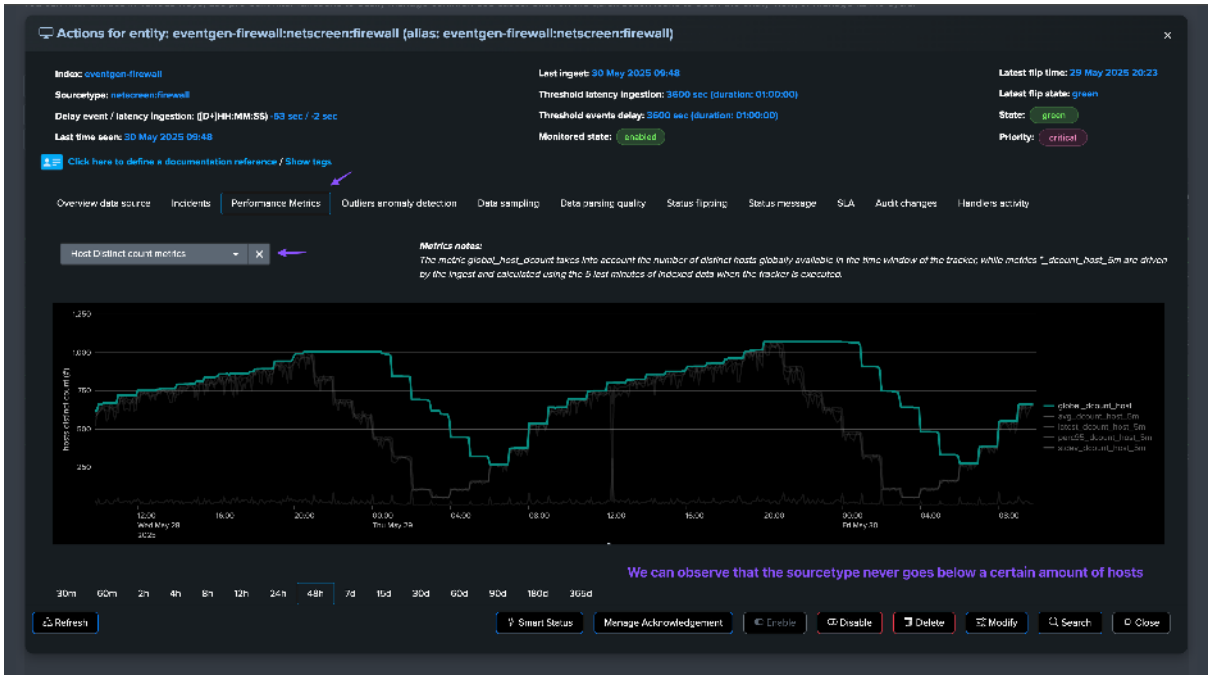
Use Case: Hosts Distinct Count Anomaly

Another use case is to detect anomalies in your feeds based on the amount of distinct hosts making it to Splunk.

This is a valuable use case especially at large scale that allows easily detecting issues, such as the potential loss of intermediate collectors or the potential loss of a major source.

This activity can be performed using a static threshold against one of the distinct hosts metrics, as well as for more advanced use cases using Machine Learning Outliers detection.

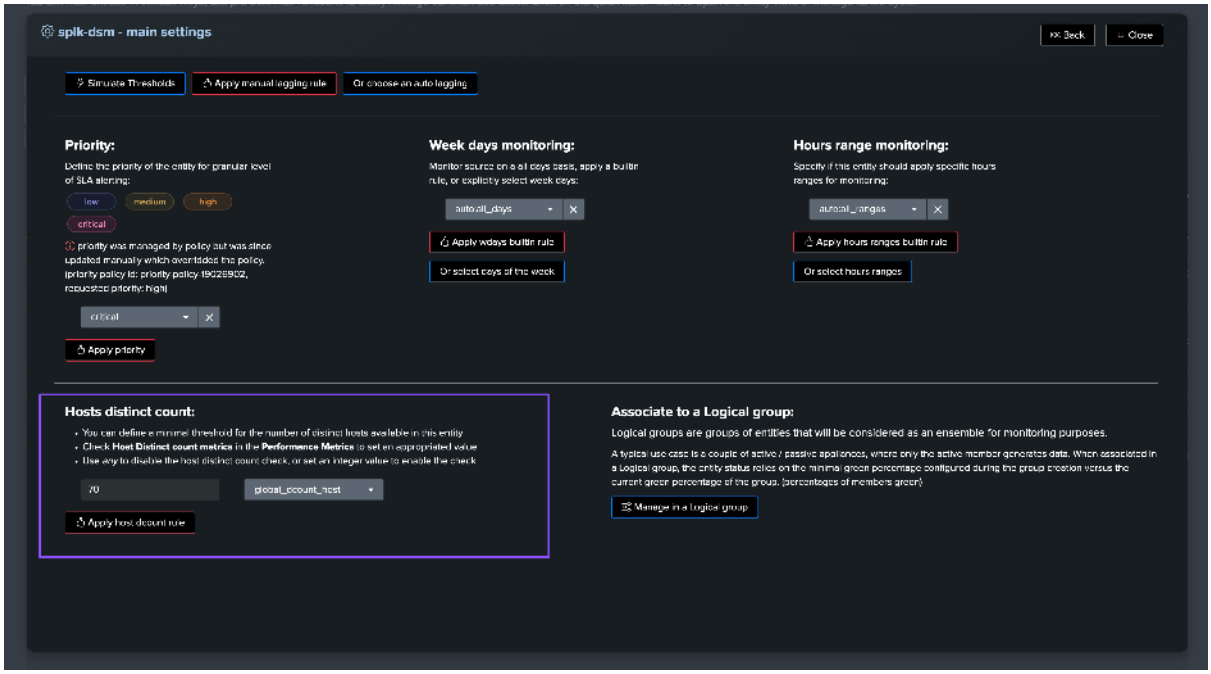
The following example demonstrates the use of a static threshold to detect anomalies in the distinct hosts count, we can observe with the following entity that a certain amount of hosts are systematically available:



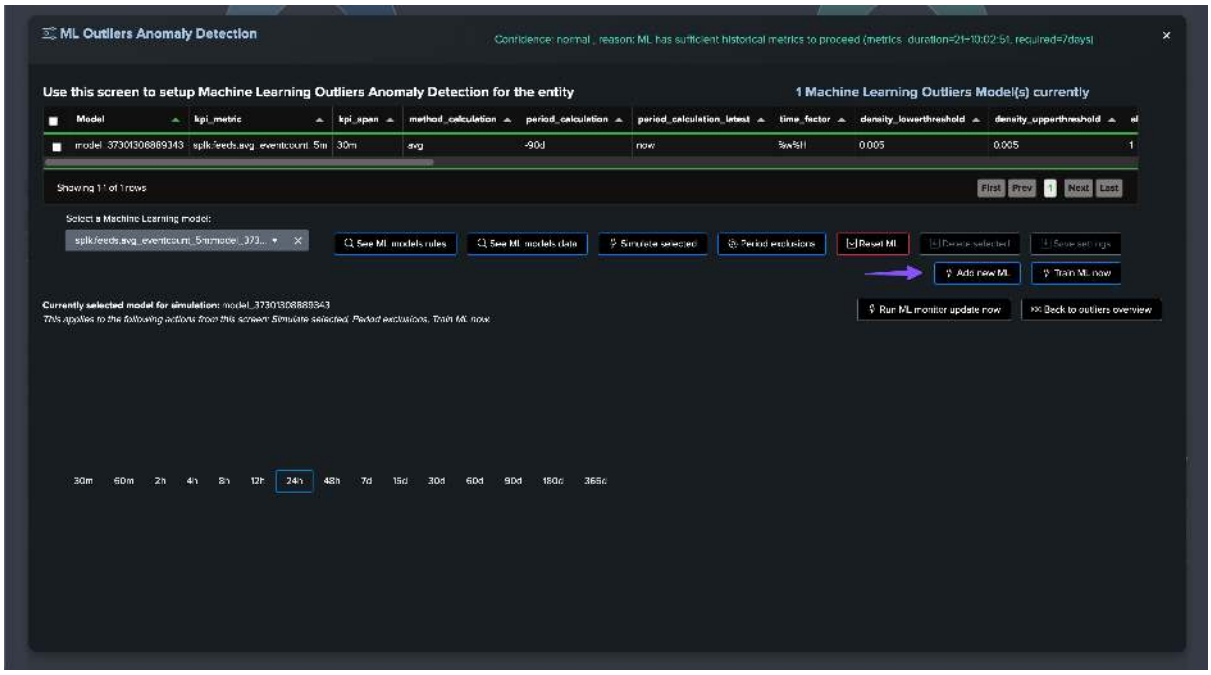
The big picture must be considered:

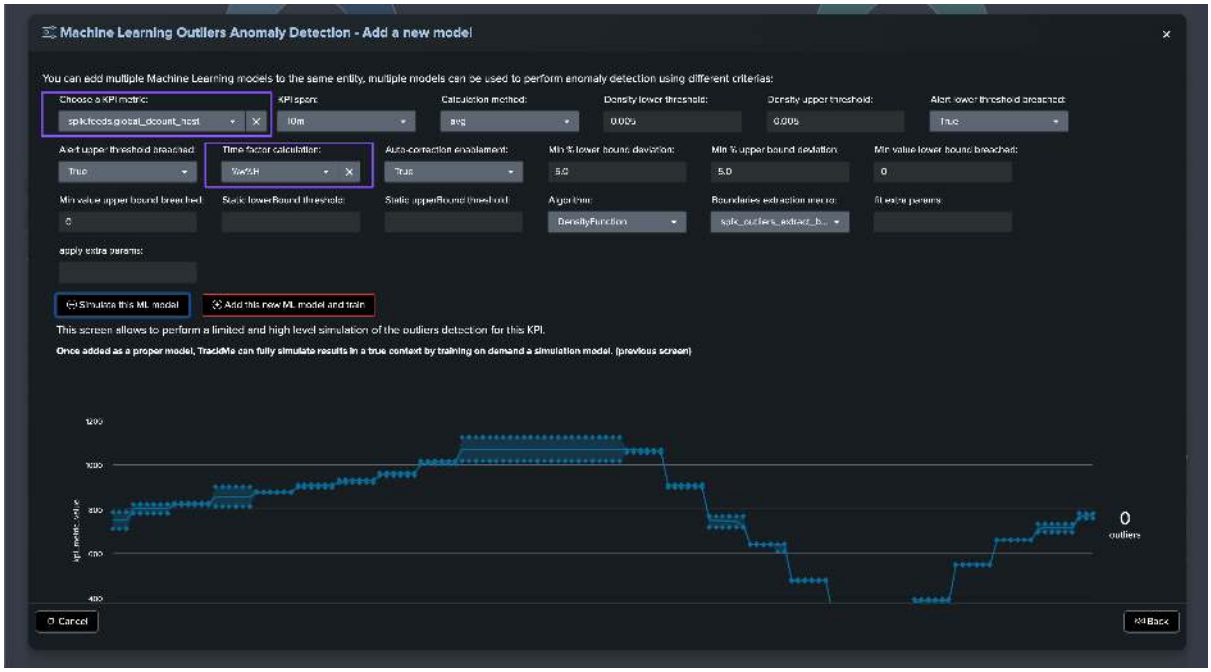


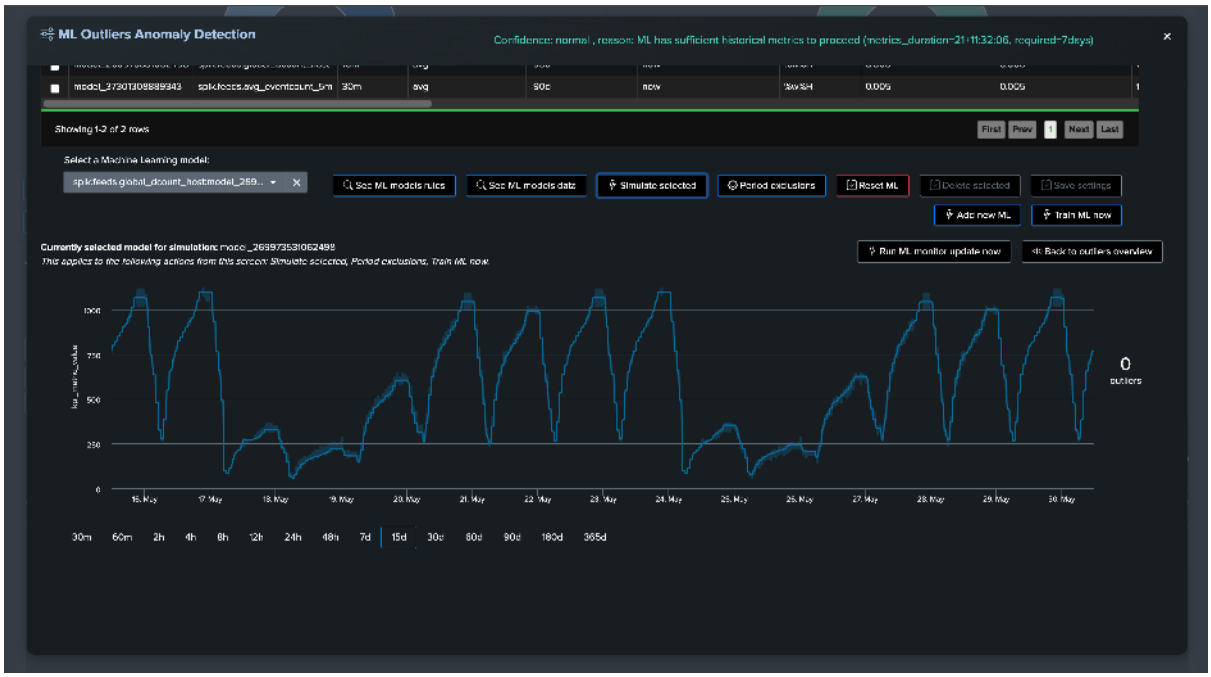
Based on this observation, we can configure a static minimal threshold for this entity, if the amount of distinct hosts goes below this threshold, the entity will be considered as in alert:

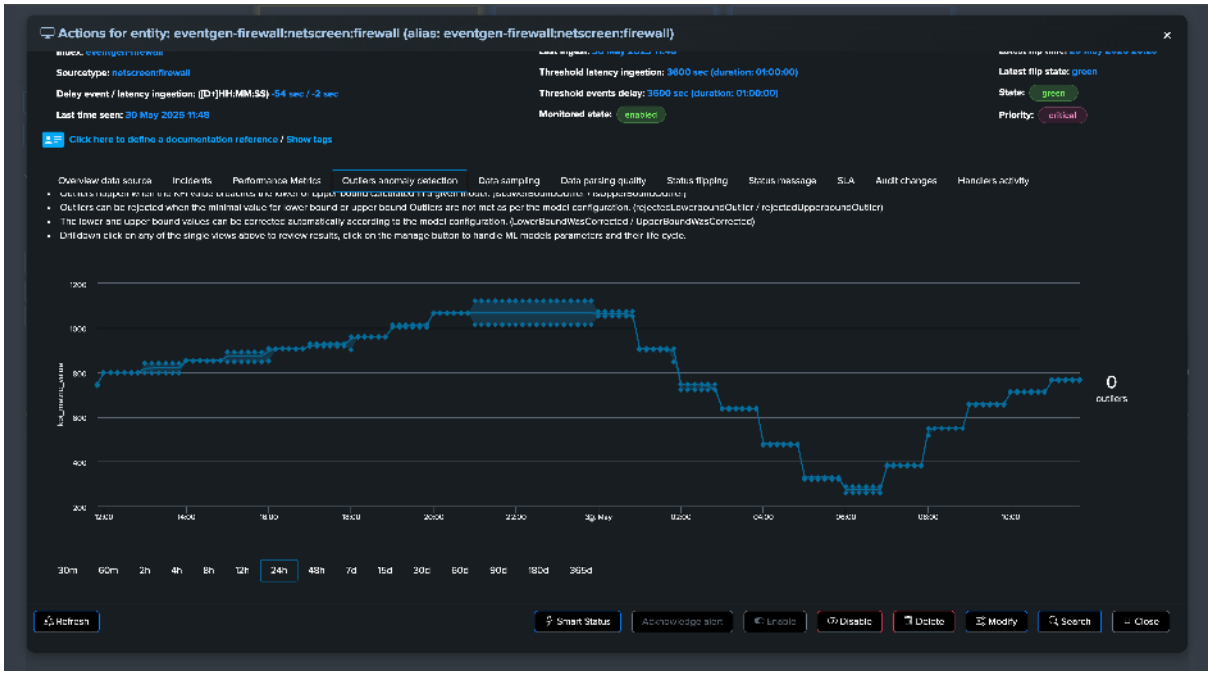


An alternative method or used in conjunction with the static threshold, is to use Machine Learning Outliers detection to detect anomalies in the distinct hosts count while taking into account time patterns:




Machine Learning Outliers Anomaly Detection - Add a new model
You can add multiple Machine Learning models to the same entity, multiple models can be used to perform anomaly detection using different criterias:
Choose a KPI metric:
KPI span:
Calculation method:
Density lower threshold:
Density upper threshold:
Alert lower threshold breached:
Alert upper threshold breached:
Time factor calculation:
Auto-correction enablement:
Min % lower bound deviation:
Min % upper bound deviation:
Min value lower bound breached:
Min value upper bound breached:
Static lowerBound threshold:
Static upperBound threshold:
Algorithm:
Boundaries extraction macro:
fit extra params:
apply extra params:
Simulate this ML model
Add this new ML model and train
This screen allows to perform a limited and high level simulation of the outliers detection for this KPI.
Once added as a proper model, TrackMe can fully simulate results in a true context by training on demand a simulation model. (previous screen)
0
outliers
Cancel
Back


ML Outliers Anomaly Detection
Confidence: normal , reason: ML has sufficient historical metrics to proceed (metrics_duration=21:11:32:06, required=7days)
Showing 1-2 of 2 rows
First
Prev
Next
Last
Select a Machine Learning model:
See ML models rules
See ML models data
Simulate selected
Period exclusions
Reset ML
Delete selected
Save settings
Add new ML
Train ML now
Currently selected model for simulation:
This applies to the following actions from this screen: Simulate selected, Period exclusions, Train ML now
Run ML monitor update now
Back to outliers overview
0
outliers
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d

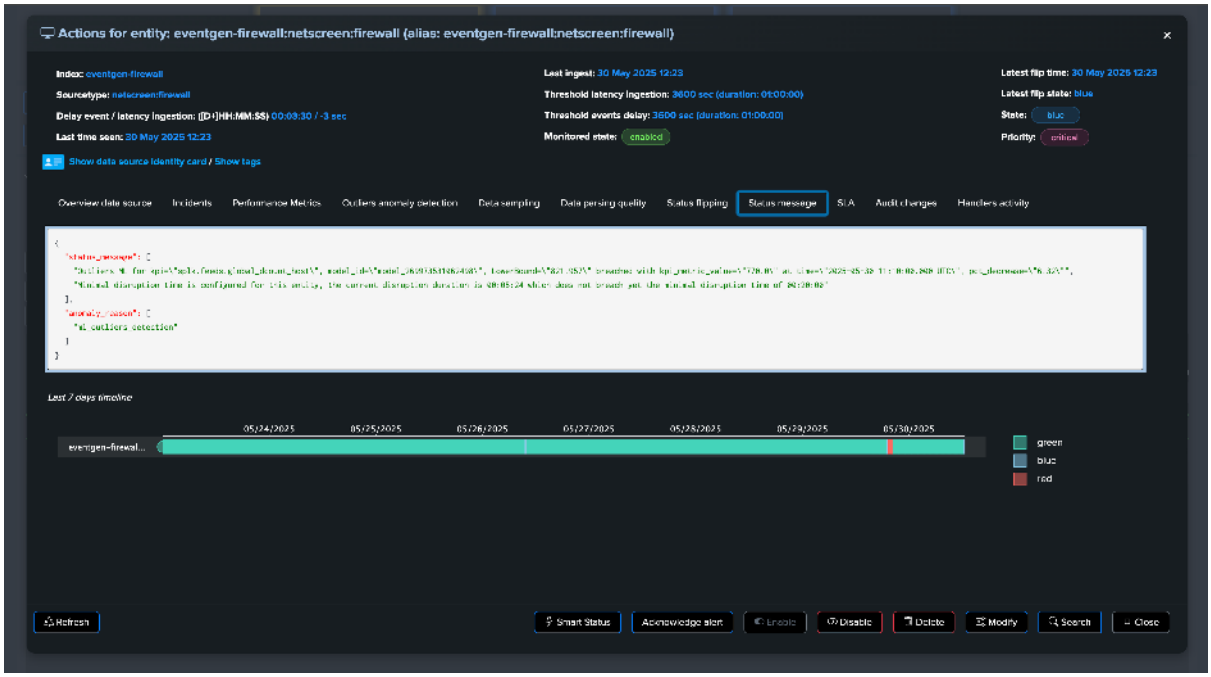


If at some point, the amount of distinct hosts goes below the threshold, the entity will be considered as in anomaly, and an incident will be opened:

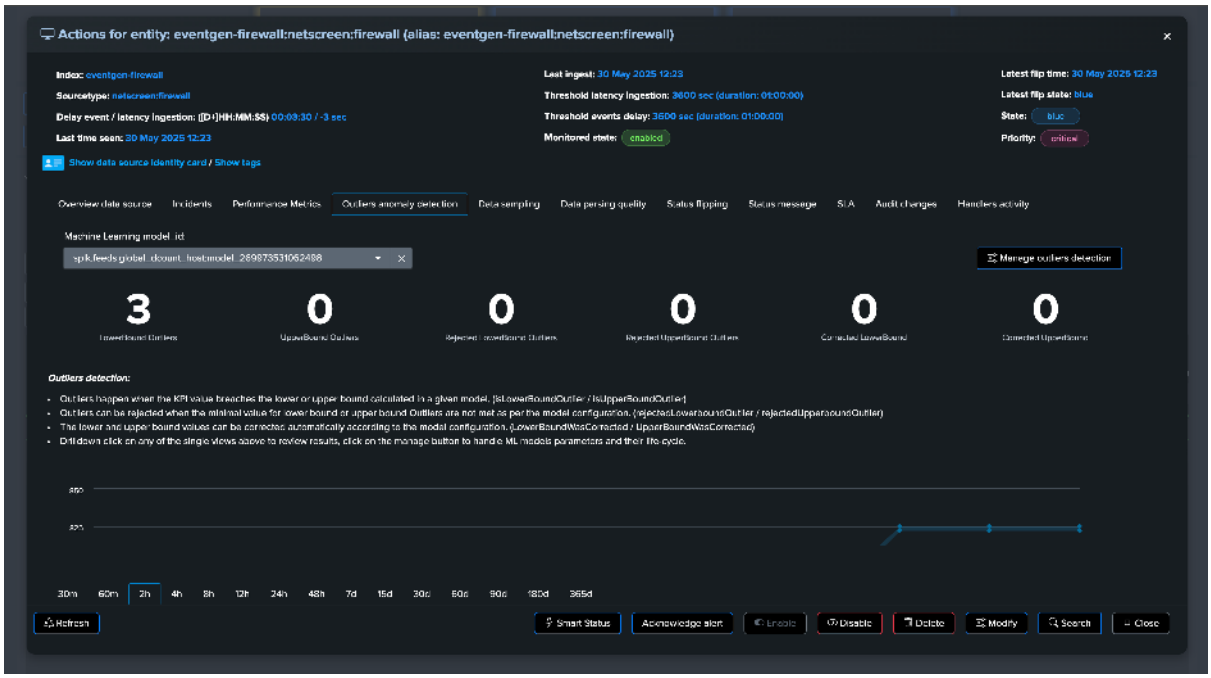
Notes:

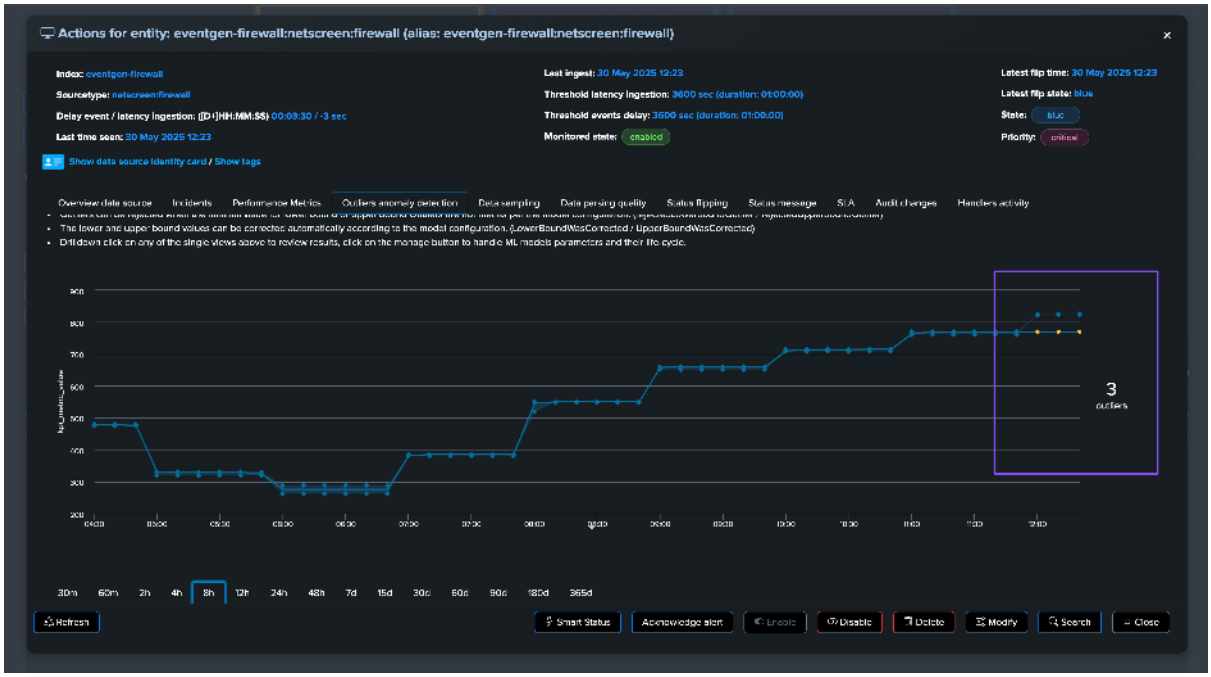
- The "global" distinct count KPI measures the number of unique hosts across the entire tracker period. Due to this global aggregation, there may be a delay in detecting anomalies as the system needs to accumulate enough data to establish a clear pattern.
- A decrease in the number of distinct hosts often correlates with a decrease in event volume. In such cases, TrackMe's Machine Learning Outliers detection may trigger alerts for both volume-based and distinct hosts models simultaneously, providing multiple indicators of the same underlying issue.

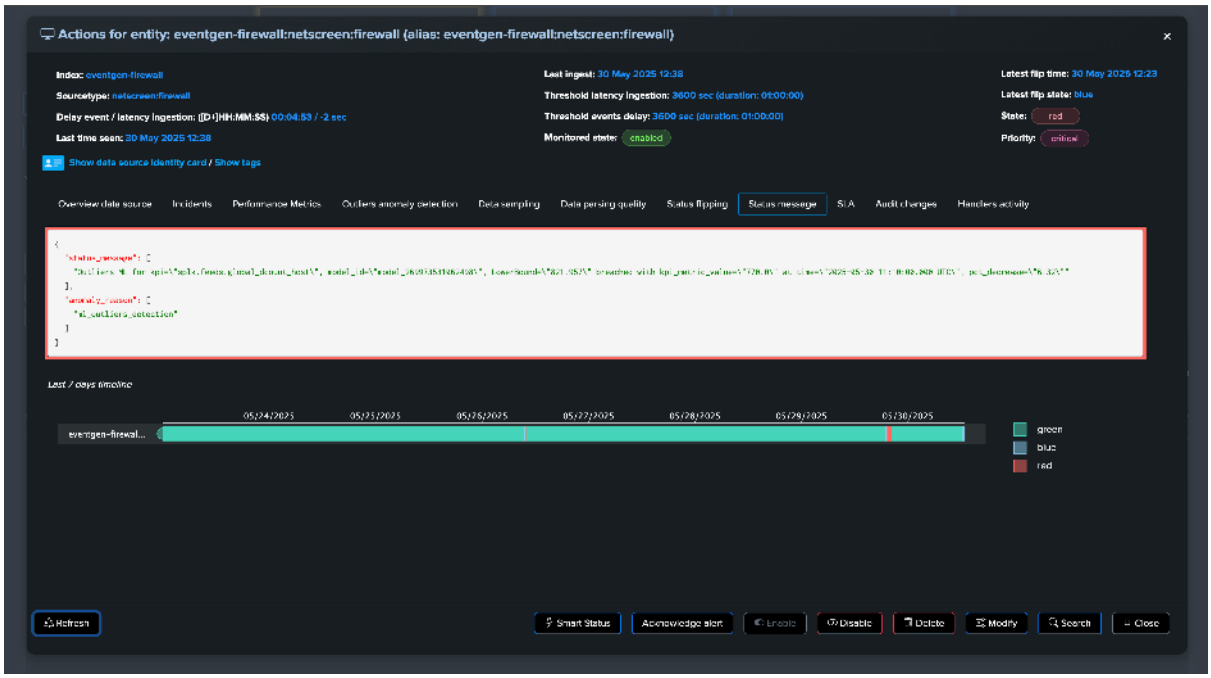
For the purposes of this demo, we have set both the static threshold and Machine Learning Outliers detection. After some time, both detection methods trigger:

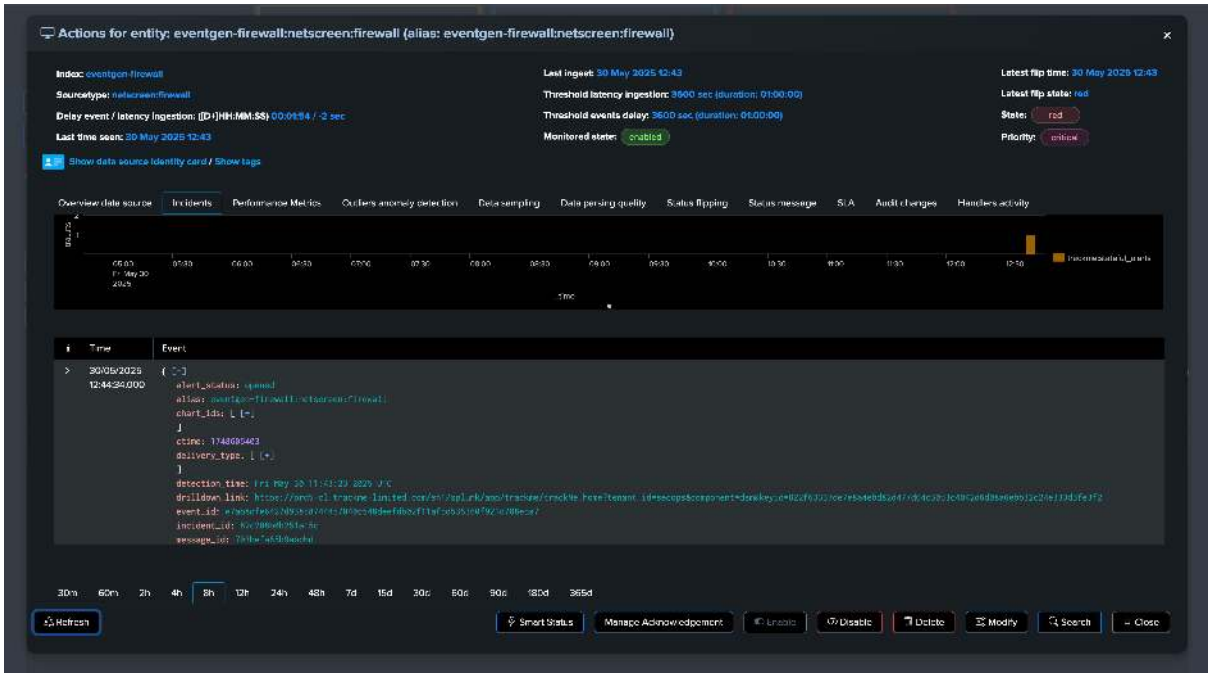



Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Index: eventgen-firewall
Sourcetype: netscreen:firewall
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:00:30 / -3 sec
Last time seen: 30 May 2025 12:23
Last ingest: 30 May 2025 12:23
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 30 May 2025 12:23
Latest flip state: blue
State: blue
Priority: critical
Show data source identity card / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Host Distinct count metrics
Metrics notes:
The metric global_host_dcount takes into account the number of distinct hosts globally available in the time window of the tracker, while metrics *_dcount_host_5m are driven by the ingest and calculated using the 5 last minutes of indexed data when the tracker is executed.
global_dcount_host
avg_dcount_host_5m
max_dcount_host_5m
perc95_dcount_host_5m
stdev_dcount_host_5m
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close


Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Index: eventgen-firewall
Sourcetype: netscreen:firewall
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:00:30 / -3 sec
Last time seen: 30 May 2025 12:23
Last ingest: 30 May 2025 12:23
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 30 May 2025 12:23
Latest flip state: blue
State: blue
Priority: critical
Show data source identity card / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Machine Learning model id
spk:feeds:global_dcount_hostmodel_2699735310624B8
Manage outliers detection
3
0
0
0
0
0
Outliers detection:
Outliers happen when the KPI value breaches the lower or upper bound calculated in a given model, (isLowerBoundOutlier / isUpperBoundOutlier)
Outliers can be rejected when the minimal value for lower bound or upper bound Outliers are not met as per the model configuration, (rejectedLowerBoundOutlier / rejectedUpperBoundOutlier)
The lower and upper bound values can be corrected automatically according to the model configuration, (LowerBoundWasCorrected / UpperBoundWasCorrected)
Drilldown click on any of the single views above to review results, click on the manage button to handle ML models parameters and their life cycle.
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close

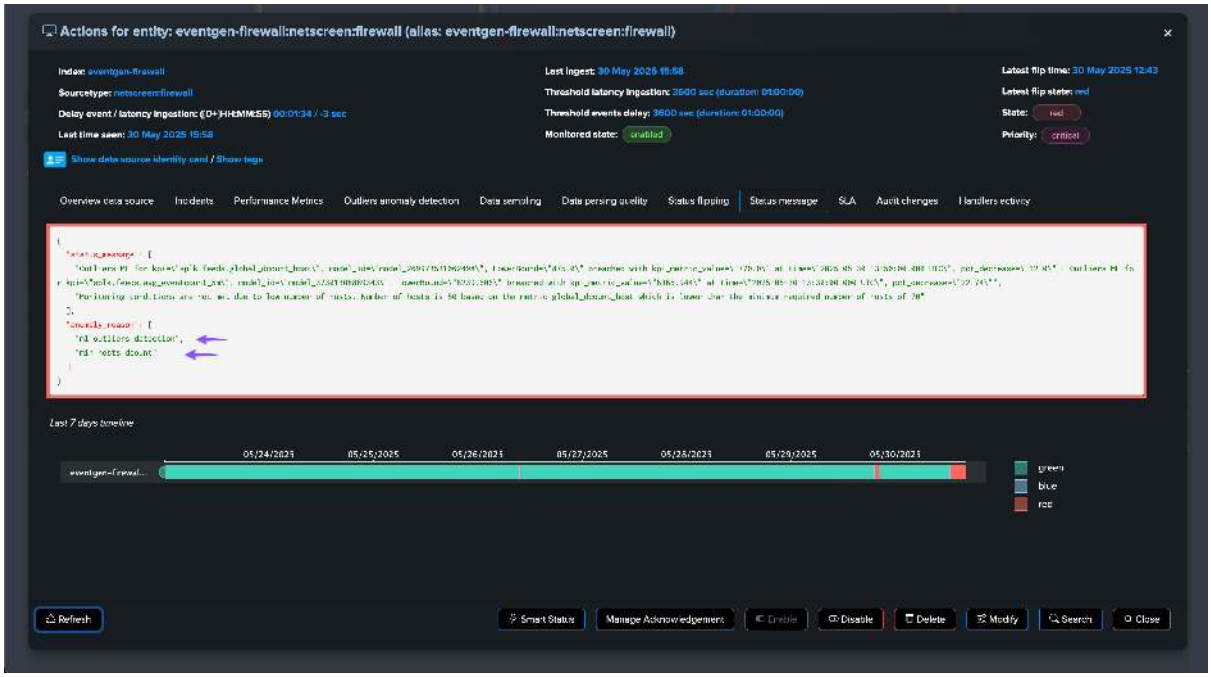

Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Index: eventgen-firewall
Sourcetype: netscreen:firewall
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:03:30 / -3 sec
Last time seen: 30 May 2025 12:23
Last ingest: 30 May 2025 12:23
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 30 May 2025 12:23
Latest flip state: blue
State: blue
Priority: critical
Show data source identity card / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
The lower and upper bound values can be corrected automatically according to the model configuration. (LowerBoundWasCorrected / UpperBoundWasCorrected)
Drilldown click on any of the single views above to review results, click on the manage button to handle ML models parameters and their life cycle.
3
outliers
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close


Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Index: eventgen-firewall
Sourcetype: netscreen:firewall
Delay event / latency ingestion: [[D+]HH:MM:SS] 00:04:53 / -2 sec
Last time seen: 30 May 2025 12:38
Last ingest: 30 May 2025 12:38
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 30 May 2025 12:23
Latest flip state: blue
State: red
Priority: critical
Show data source identity card / Show tags
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
"status_message": [
"ML_outliers_detection"
]
Last 7 days timeline
05/24/2025
05/25/2025
05/26/2025
05/27/2025
05/28/2025
05/29/2025
05/30/2025
eventgen-firewall...
green
blue
red
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close

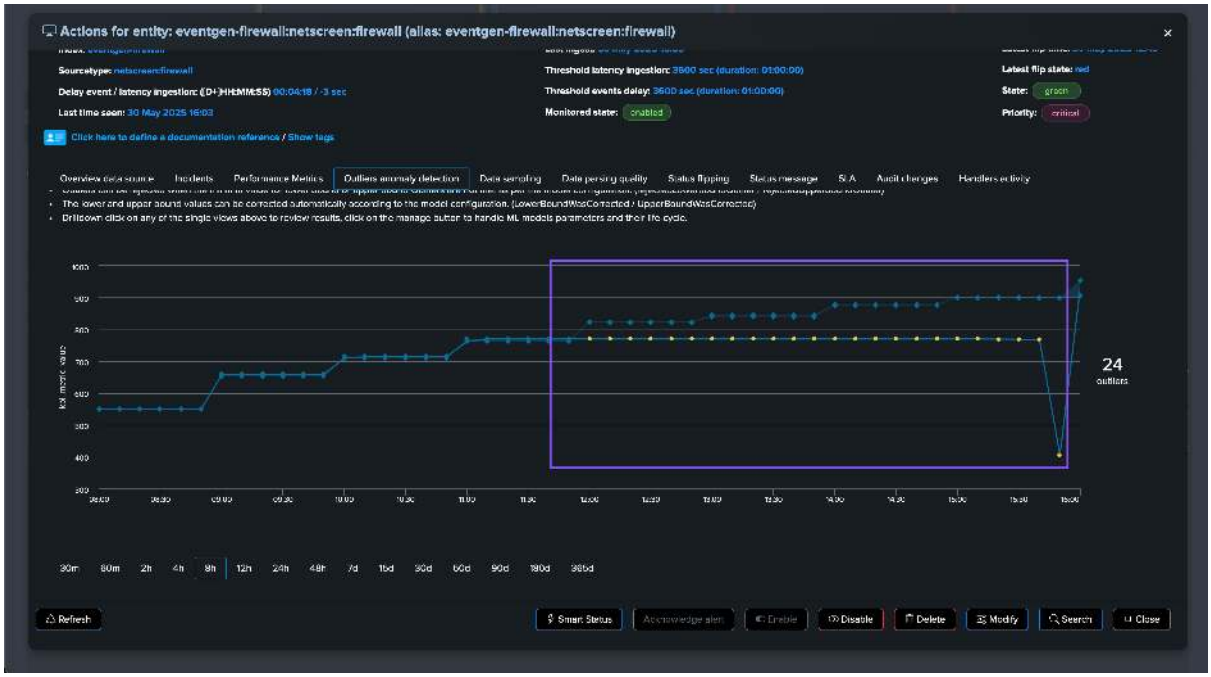

Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Time
Event
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Manage Acknowledgement
Enable
Disable
Delete
Modify
Search
Close

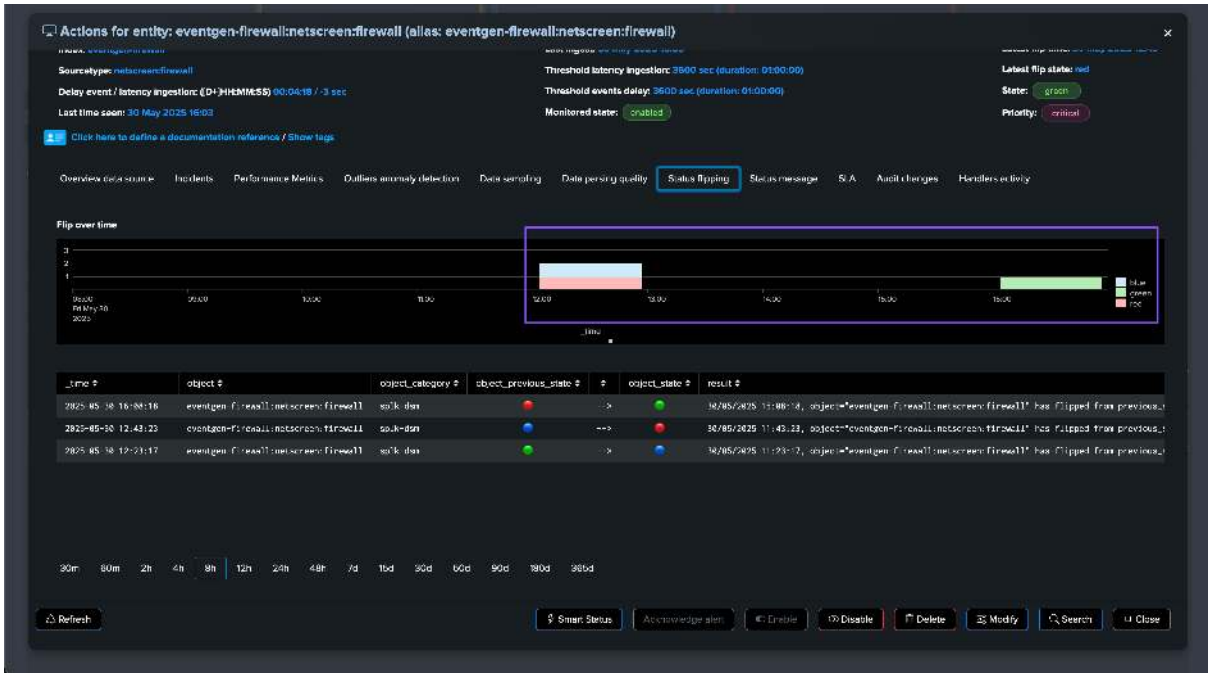

TrackMe Notification
Environment: https://prd5-cl.trackme-limited.com
Detection Time: Fri May 30 11:43:23 2025 UTC
Alert Status: opened
Tenant: secops
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Drilldown Link
Status: red
Incident ID: 82c208b0b251af5c
Detailed Information:
The entity has transitioned to an alerting state:
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Object ID: 822f63337de7e8a4ebd82d477d64c3053c4042d6e88a6ebb32c24e333d3fe3f2
Category: splk-dsm
Anomaly Reason(s):
ml_outliers_detection
Message(s):
30/05/2025 11:43:23, object="eventgen-firewall:netscreen:firewall" has flipped from previous_state="blue" to state="red" with anomaly_reason="ml_outliers_detection", previous_anomaly_reason="ml_outliers_detection", disruption_time="0"
Outliers ML for kpi="splk.feeds.global_dcount_host", model_id="model_289973531062498", LowerBound="821.957" breached with kpi_metric_value="770.8" at time="2025-05-30 11:10:00.000 UTC", pct_decrease="6.32"
Message source: trackme:flip
Message source ID: bccef7332b01c646b0d5ef4ca9433f2a2abf8b3e3eaf86a1d269975a8e402237
This is an automated email, please do not reply directly to this email.

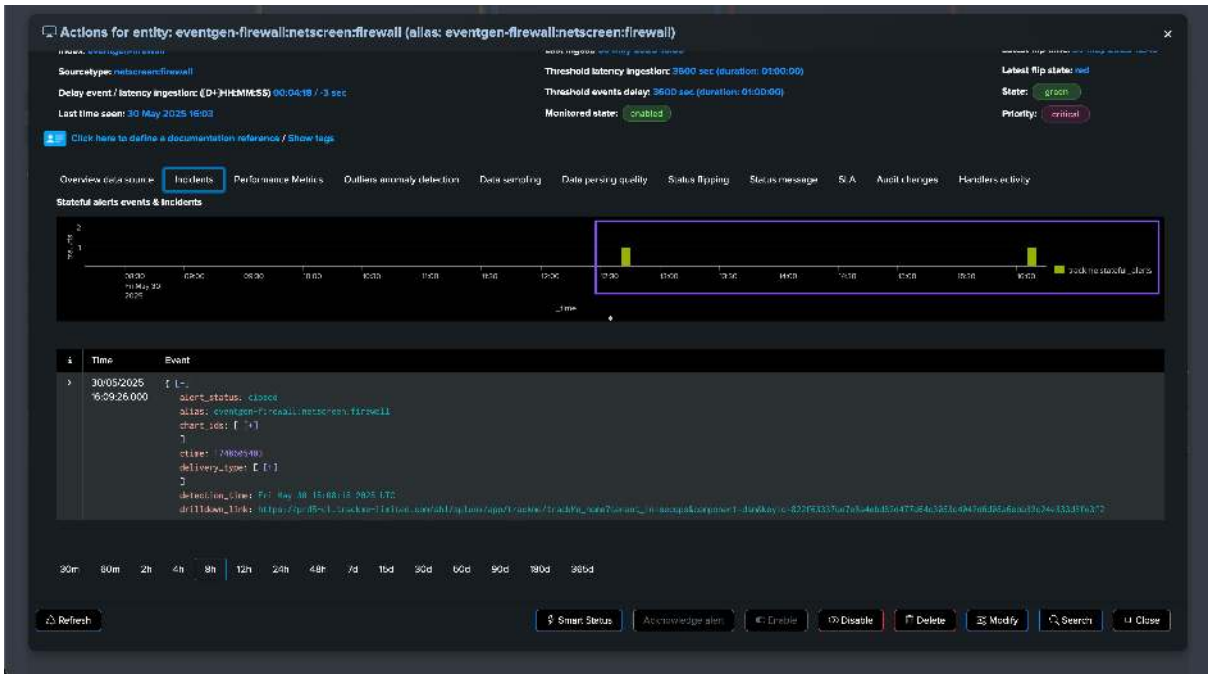

Performance Charts (24h)
Latency Metrics - Last 24h
Delay Metrics - Last 24h
Volume Metrics - Last 24h
Hosts DCount Metrics - Last 24h
Machine Learning Outliers - Last 24h

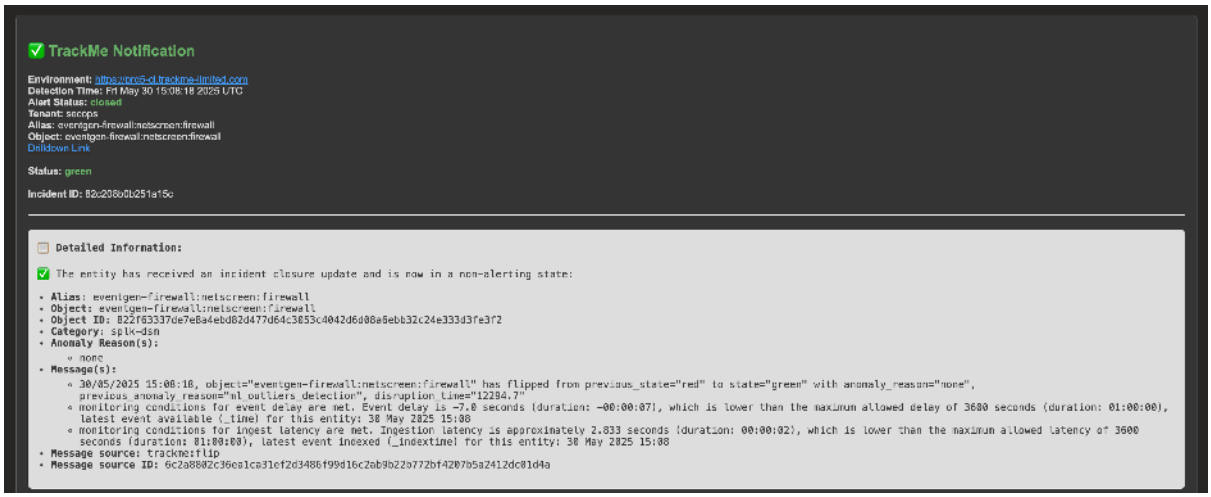


As usual, once the issue is resolved, the entity returns to green after the Outliers anomaly condition is resolved and TrackMe has processed the ML rendering for this entity:




Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Sourcetype: netscreen:firewall
Delay event / latency ingestion: (D+)HH:MM:SS) 00:04:18 / -3 sec
Last time seen: 30 May 2025 16:03
Click here to define a documentation reference / Show tags
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip state: red
State: green
Priority: critical
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Flip over time
_time
object
object_category
object_previous_state
object_state
result
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close


Actions for entity: eventgen-firewall:netscreen:firewall (alias: eventgen-firewall:netscreen:firewall)
Sourcetype: netscreen:firewall
Delay event / latency ingestion: (D+)HH:MM:SS) 00:04:18 / -3 sec
Last time seen: 30 May 2025 16:03
Click here to define a documentation reference / Show tags
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip state: red
State: green
Priority: critical
Overview data source
Incidents
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Handlers activity
Stateful alerts events & incidents
_time
Time
Event
30/05/2025 16:09:26.000
alert_status: closed
alias: eventgen-firewall:netscreen:firewall
chart_idx: [0]
ctime: 1748620140
delivery_type: [0]
detection_time: Fri May 30 15:08:18 2025 UTC
drilldown_link: https://prd5-cl.trackme-limited.com/en-US/app/trackme/flip_alerts?earliest=1748620140&latest=1748620140&form.object=eventgen-firewall:netscreen:firewall&form.object_category=splk-dsn
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close


TrackMe Notification
Environment: https://prd5-cl.trackme-limited.com
Detection Time: Fri May 30 15:08:18 2025 UTC
Alert Status: closed
Tenant: secops
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Drilldown Link
Status: green
Incident ID: 82c208b0b251a15c
Detailed Information:
The entity has received an incident closure update and is now in a non-alerting state:
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Object ID: 822f63337de7e8a4ebd82d477d64c3853c4042d6d08a8ebb32c24e333d3fe3f2
Category: splk-dsn
Anomaly Reason(s):
none
Message(s):
30/05/2025 15:08:18, object="eventgen-firewall:netscreen:firewall" has flipped from previous_state="red" to state="green" with anomaly_reason="none", previous_anomaly_reason="ml_outliers_detection", disruption_time="12284.7"
monitoring conditions for event delay are met. Event delay is -7.0 seconds (duration: -00:00:07), which is lower than the maximum allowed delay of 3600 seconds (duration: 01:00:00), latest event available (_time) for this entity: 30 May 2025 15:08
monitoring conditions for ingest latency are met. Ingestion latency is approximately 2.833 seconds (duration: 00:00:02), which is lower than the maximum allowed latency of 3600 seconds (duration: 01:00:00), latest event indexed (_indextime) for this entity: 30 May 2025 15:08
Message source: trackmeflip
Message source ID: 6c2a888f2c36ea1ca31ef2d3486f99d16c2ab9b22b772bf4207b5a2412dc81d4a

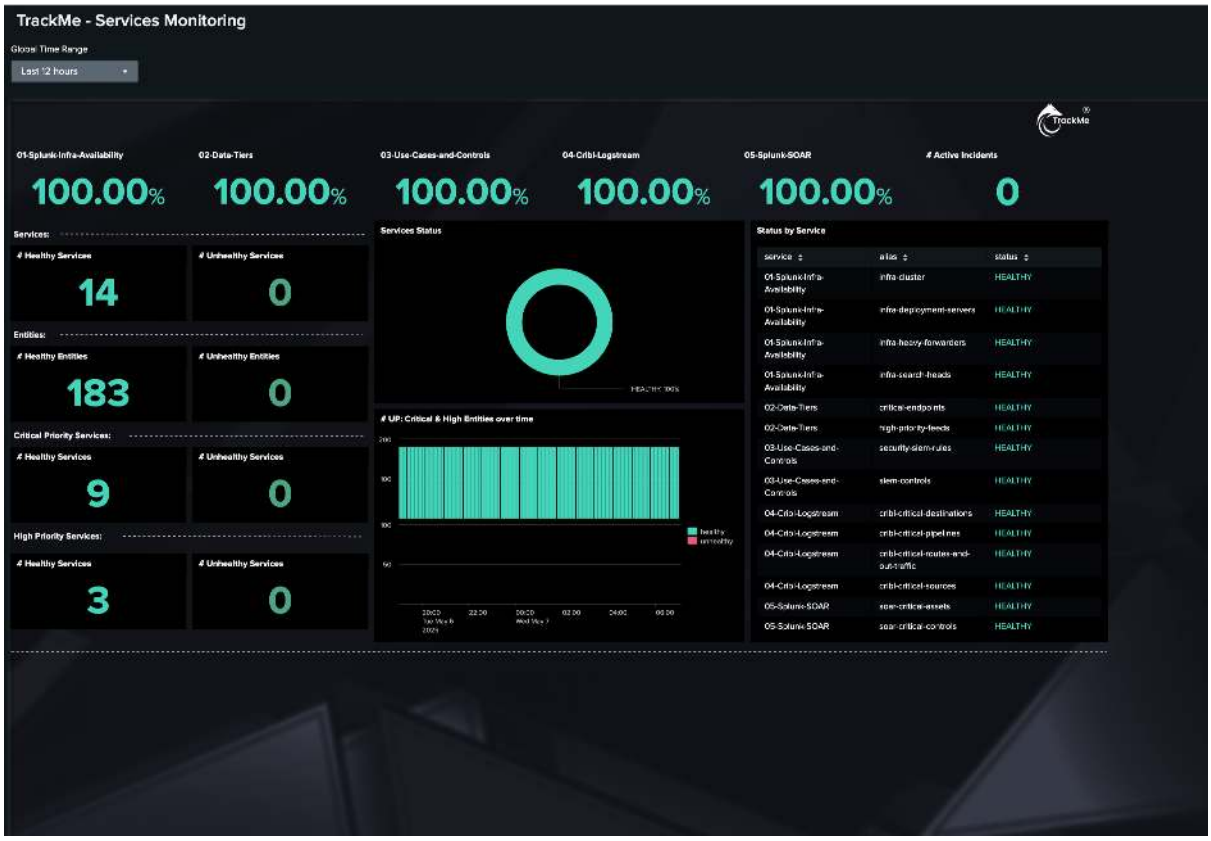
2.2.2 Use Case Demo: 360 Services Monitoring with TrackMe

Use Case Demo: 360 Services Monitoring with TrackMe

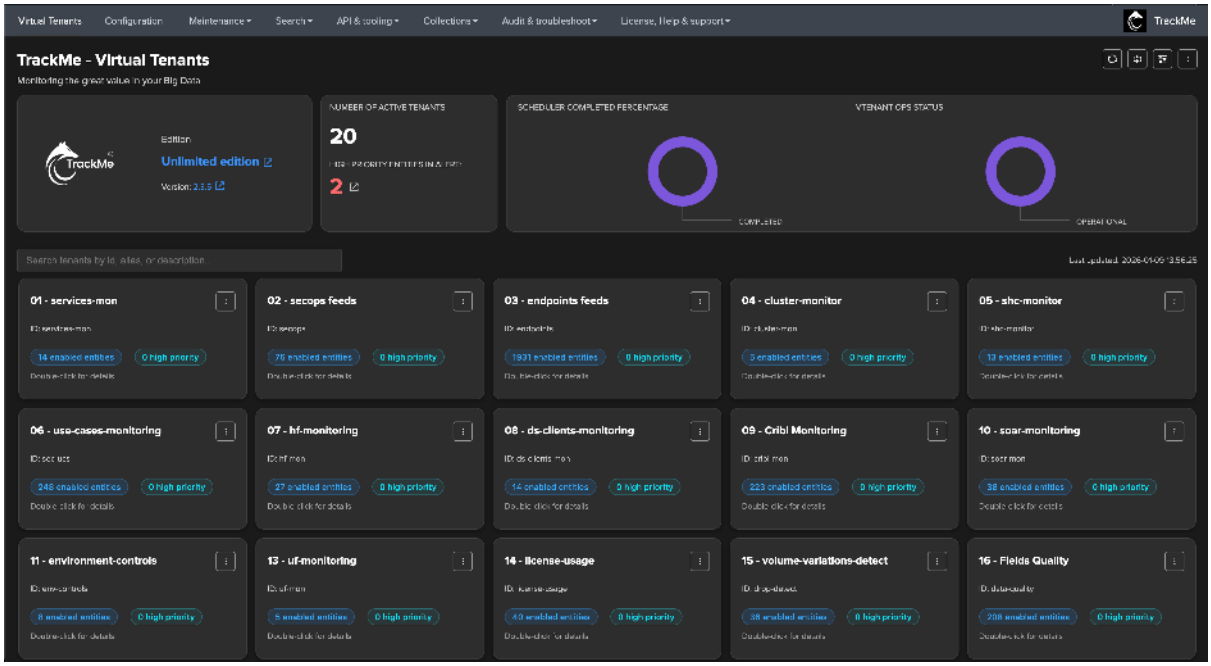
- This use case demo demonstrates how TrackMe can be used to perform a **360 degree monitoring** of the different services that are commonly composing Splunk environments, with addition of third parties notably Cribl Logstream.
- **The purpose of this demo is to show step by step how to design and implement TrackMe concepts and features, notably:**
  - `Data Tiers:` Monitoring high priority Splunk Feeds availability and performance using TrackMe component `splk-dsm`.
  - `Data Tiers:` Monitoring high priority endpoints availability (Think about Active Directory domain controllers, Checkpoint firewalls, etc.) using TrackMe component `splk-dsm`.
  - `Data Tiers:` Monitoring abnormal volume variations in Splunk indexes and Splunk license usage using TrackMe component `splk-flx`. (Since TrackMe 2.4.18 the `splk-vol` component covers per-index licensed volume out of the box — see *VOL — Volume Outliers*.)
  - `Splunk Tiers:` Monitoring key aspects and metrics of Splunk Indexers Cluster using TrackMe component `splk-flx`.
  - `Splunk Tiers:` Monitoring key aspects and metrics of Splunk Search Head Cluster or Search Heads using TrackMe component `splk-flx`.
  - `Splunk Tiers:` Monitoring key aspects of Splunk Heavy Forwarder tiers using TrackMe component `splk-flx`. (Since TrackMe 2.4.22 the Splunk S2S tenant covers the receivers and intermediate forwarders, and since 2.4.24 the Splunk Forwarder Collectors tenant covers the heavy forwarders used as collection points and their inputs — see *Splunk S2S* and *Splunk Forwarder Collectors*.)
  - `Splunk Tiers:` Monitoring Splunk deployment servers and clients using TrackMe component `splk-flx`. (Since TrackMe 2.4.22 the Splunk UF Fleet tenant covers the forwarding agents and their Deployment Servers — see *Splunk UF Fleet*.)
  - `Use Cases & Controls:` Monitoring Splunk core & Splunk Enterprise Security use cases using TrackMe Workload component `splk-wlk`.
  - `Use Cases & Controls:` Monitoring various environments control points using TrackMe component `splk-flx`.
  - `Cribl Logstream:` Monitoring Cribl Logstream availability and performance using TrackMe component `splk-flx`. (Since TrackMe 2.4.19 the Cribl Stream and Cribl Edge tenants monitor Cribl directly — see *Cribl Stream* and *Cribl Edge*.)
  - `Splunk SOAR tier:` Monitoring Splunk SOAR platforms using TrackMe component `splk-flx`.
- Some of the components leveraged in this demo are restricted features available in TrackMe Enterprise Edition & Unlimited Edition.
- **This demo documentation is currently a work in progress** and will be updated in the future to reflect the latest features and capabilities of TrackMe.

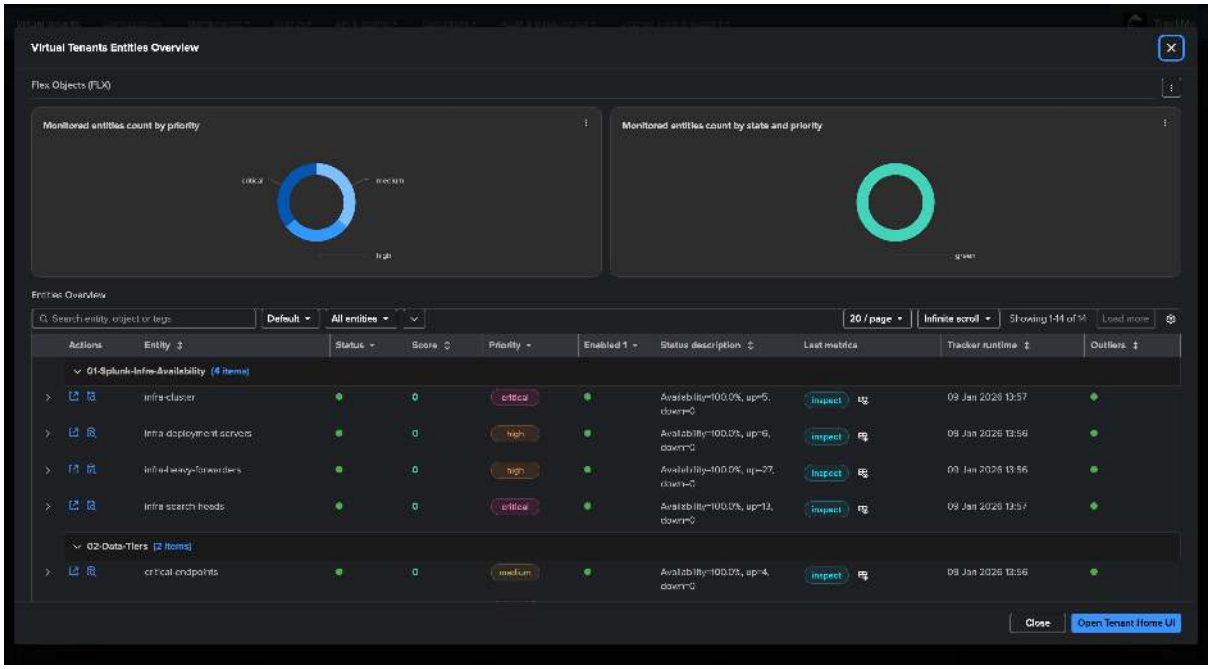
Pictures Gallery

The following image shows a template Splunk dashboard which calls TrackMe Flex converging entities, which transparently correlate the status of TrackMe entities to form the representation of the different tiers in the environment: (find this template in the API & Tooling menu, from TrackMe 2.1.18)

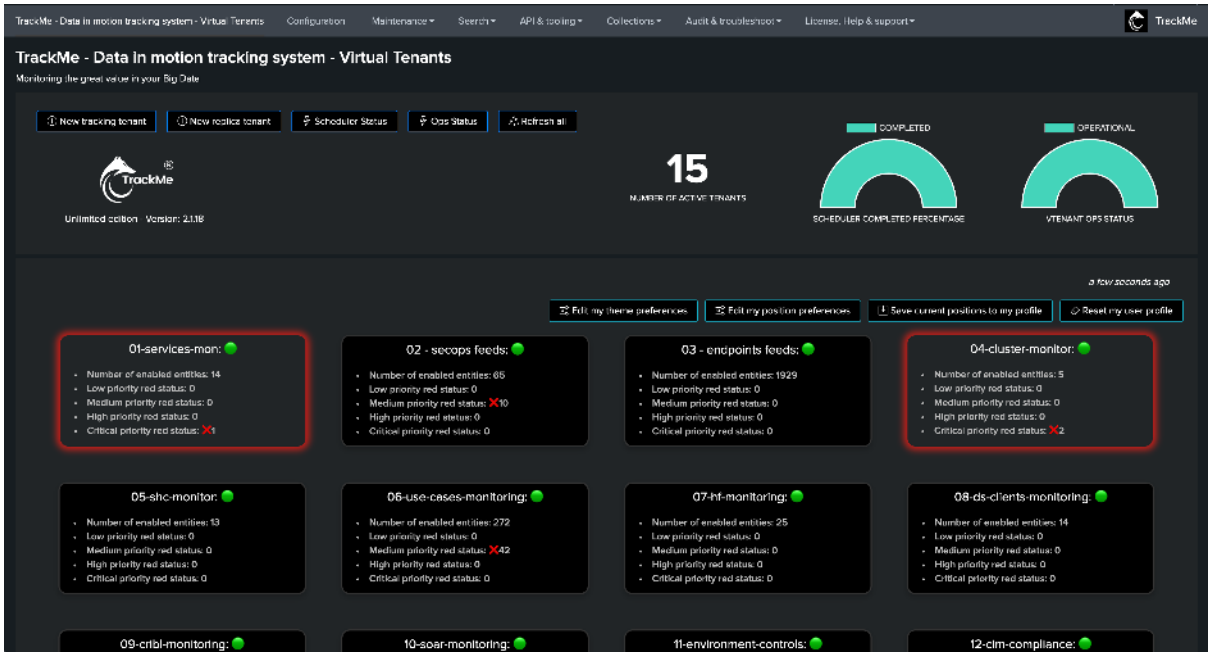


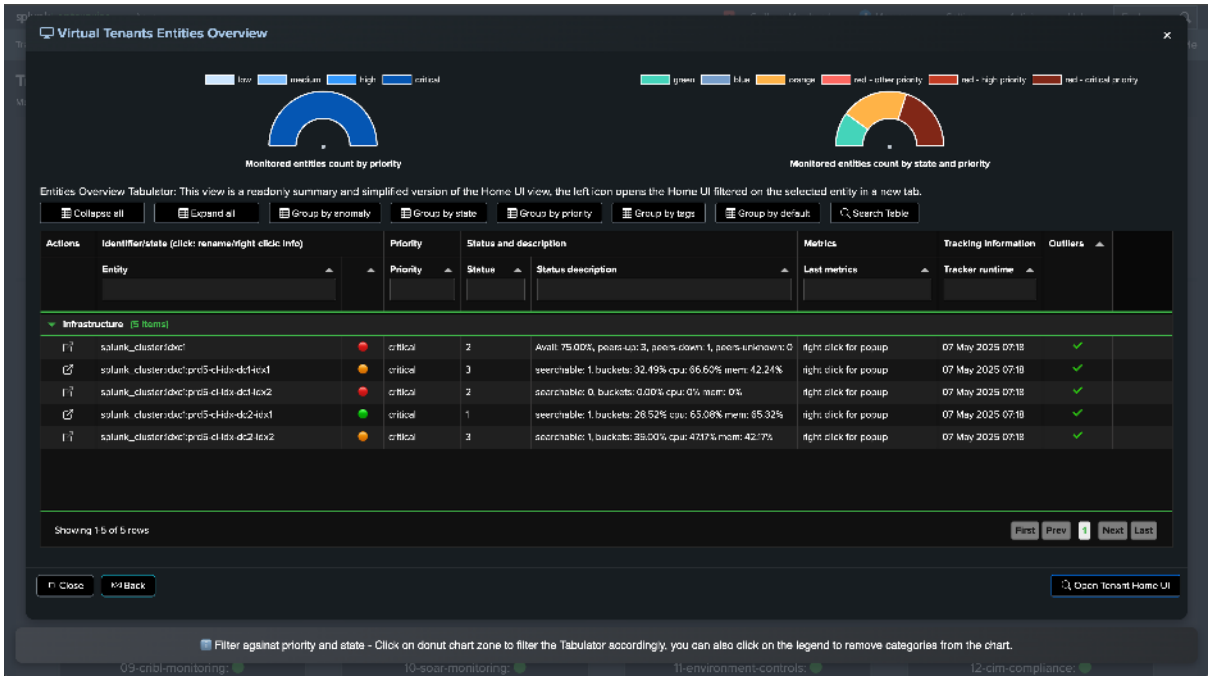
The following images show TrackMe Virtual Tenants Home page:



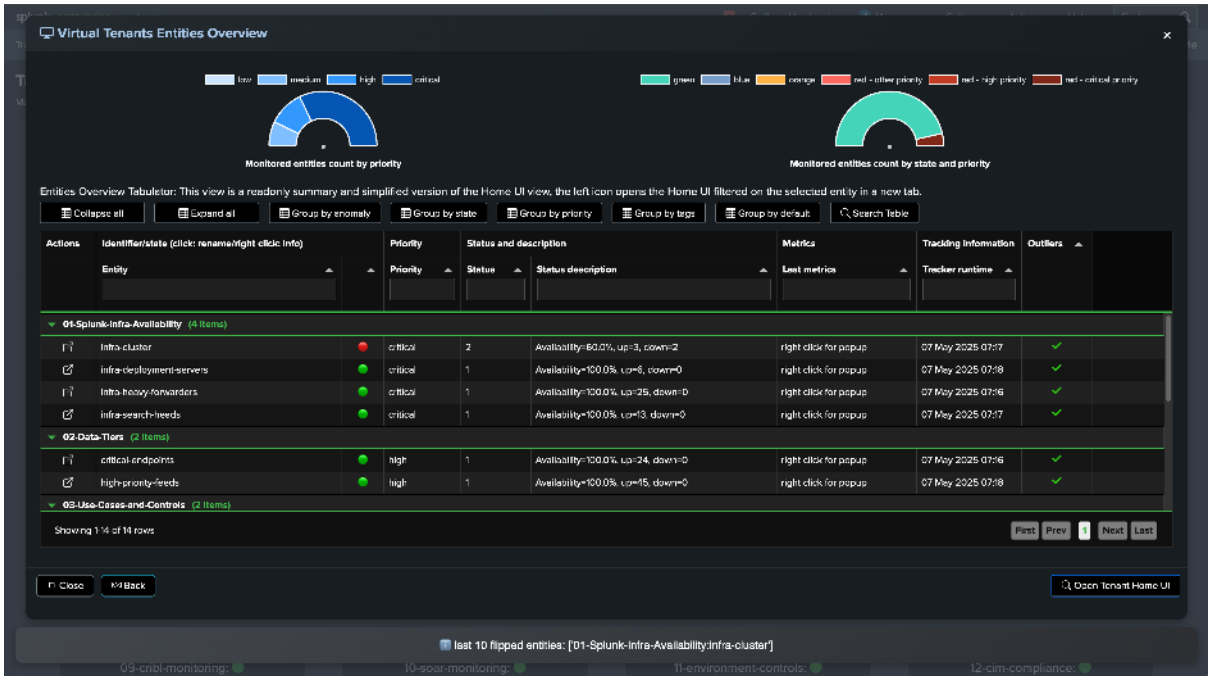


An incident is affecting the Splunk Indexers Cluster tier:

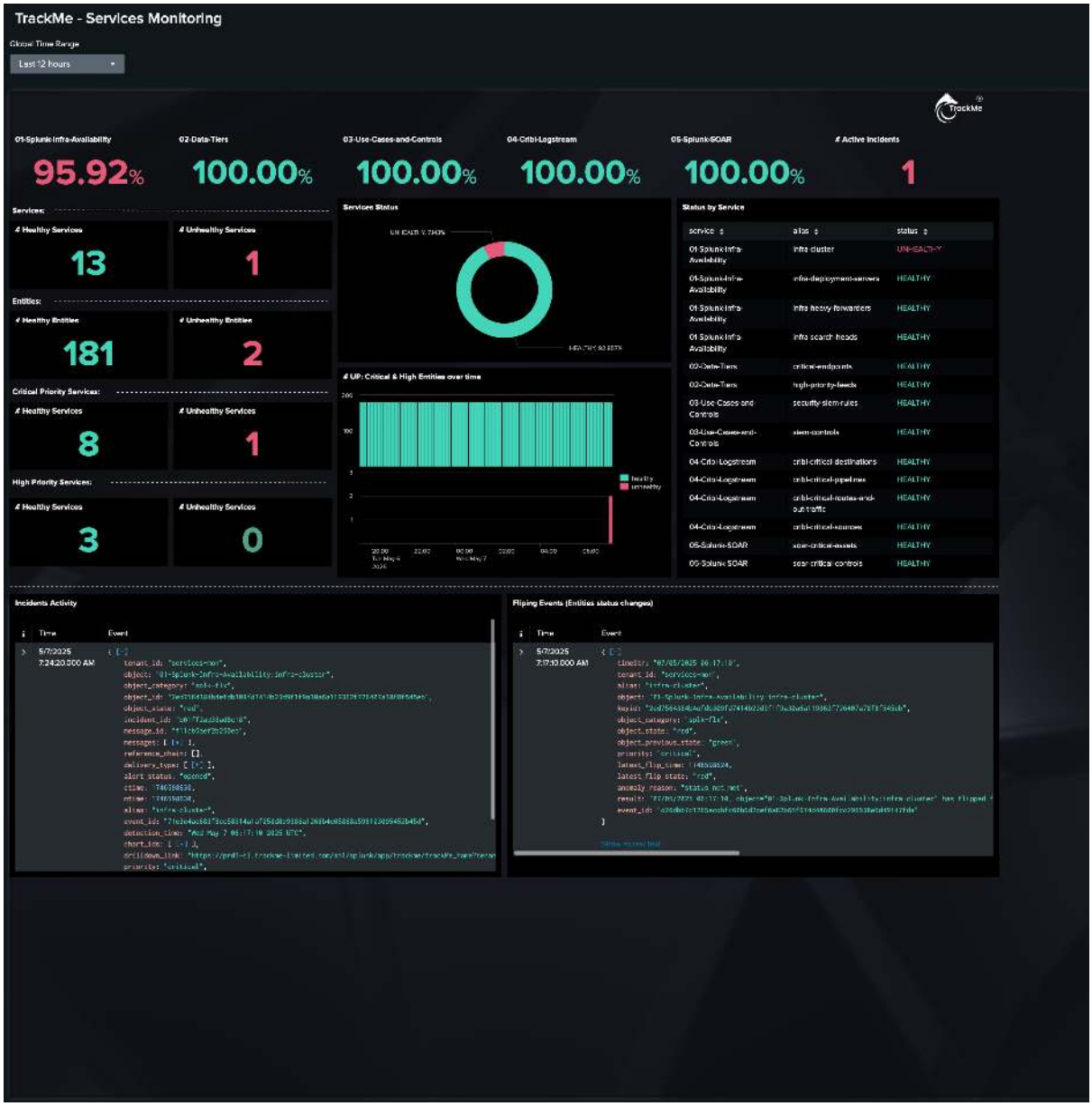




The Services Monitoring Virtual Tenant view:



The Services Monitoring Dashboard view:

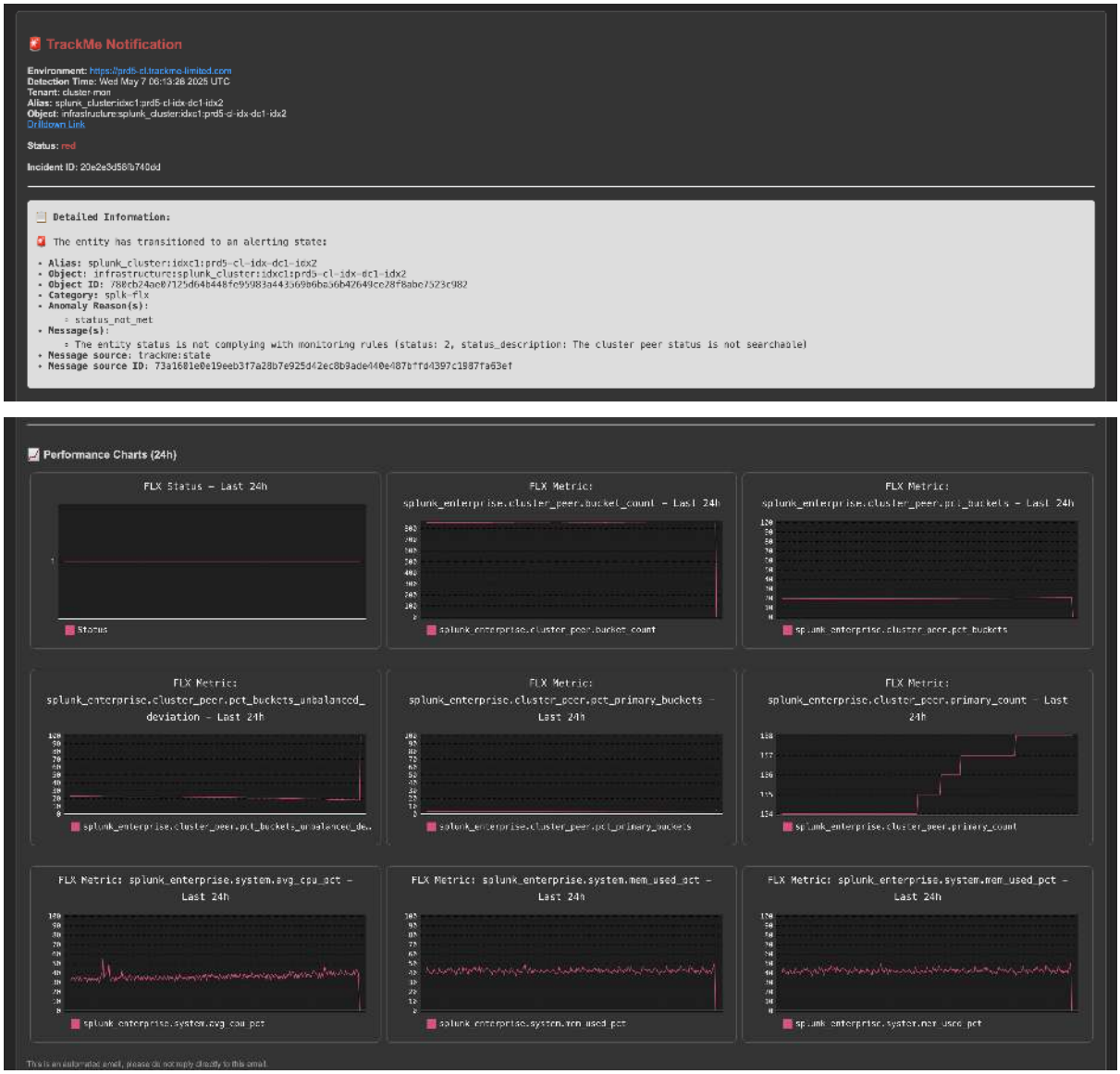


The StateFul Opening incident Email notification:



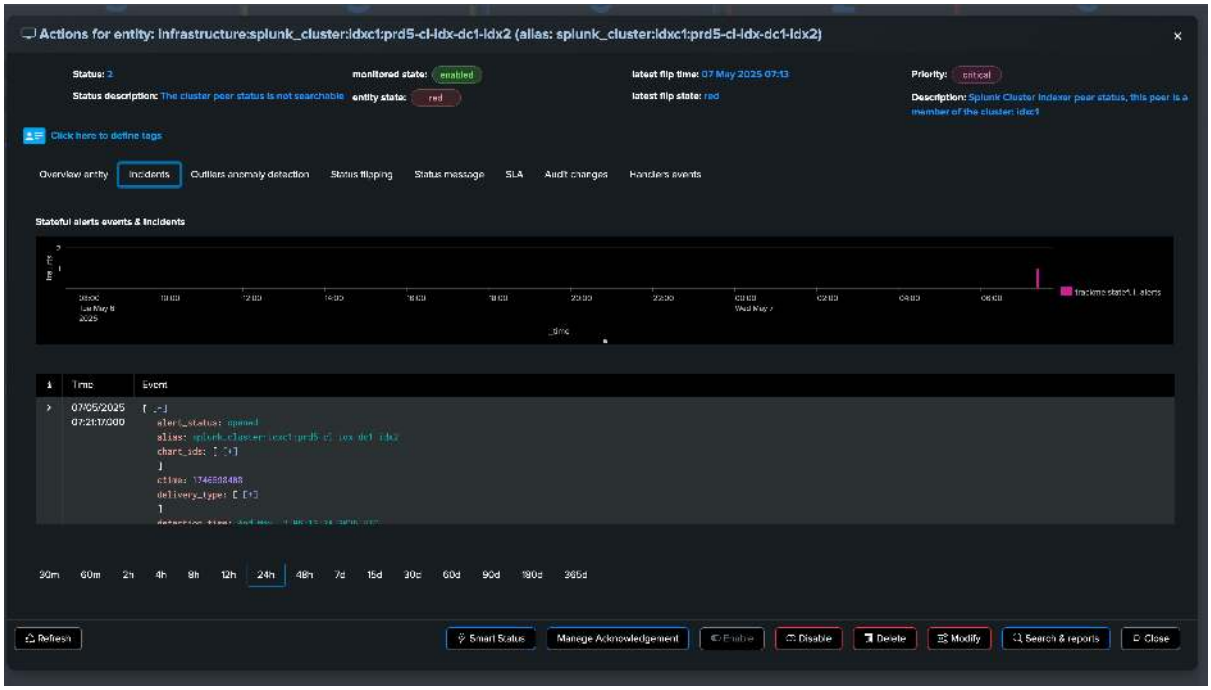


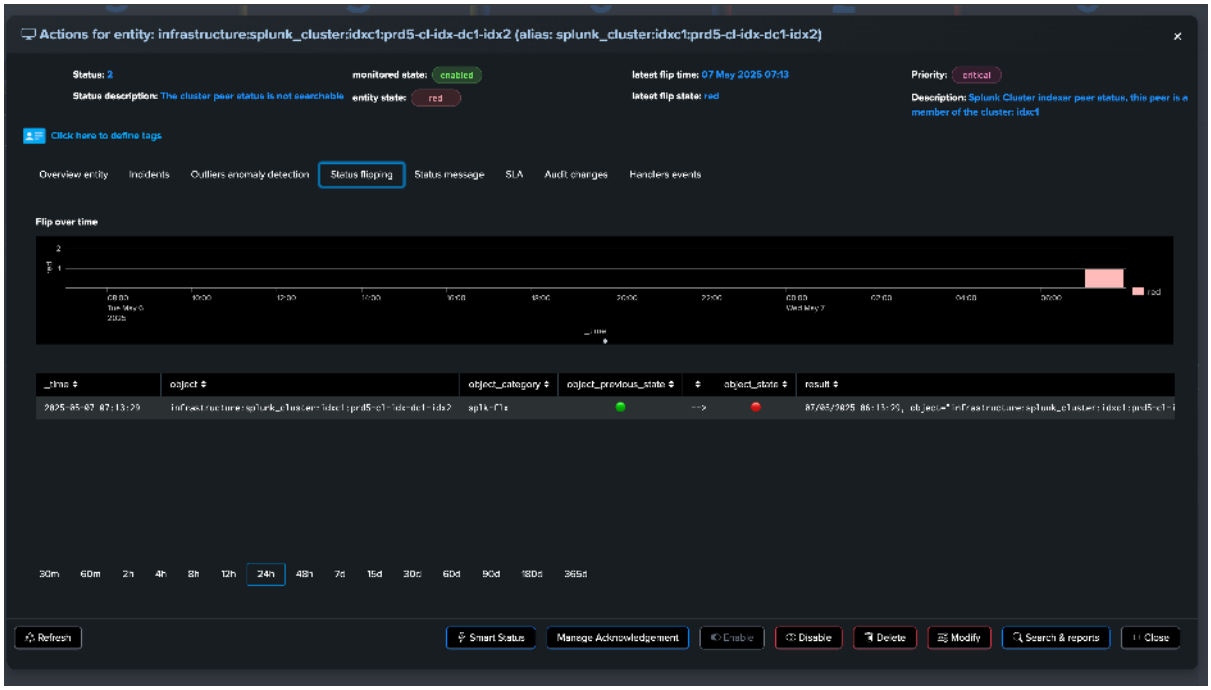
The StateFul Opening incident notification from the Splunk Indexer tier tenant:

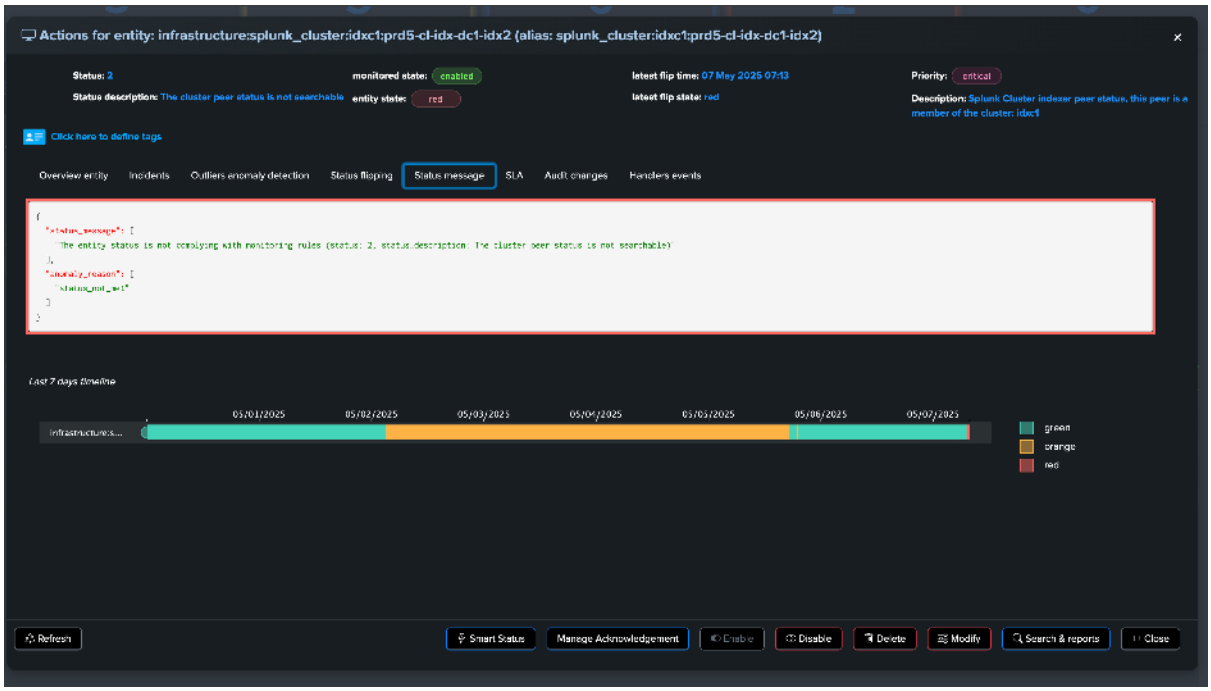


Several views of the faulty Splunk indexer:


Actions for entity: infrastructure:splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2 (alias: splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2)
Status: 2
Status description: The cluster peer status is not searchable
monitored state: enabled
entity state: red
latest flip time: 07 May 2025 07:13
latest flip state: red
Priority: critical
Description: Splunk Cluster Indexer peer status, this peer is a member of the cluster: idxc1
Click here to define tags
Overview entity
Incidents
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Handlers events
Metrics for this entity:
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Manage Acknowledgement
Enable
Disable
Delete
Modify
Search & reports
Close

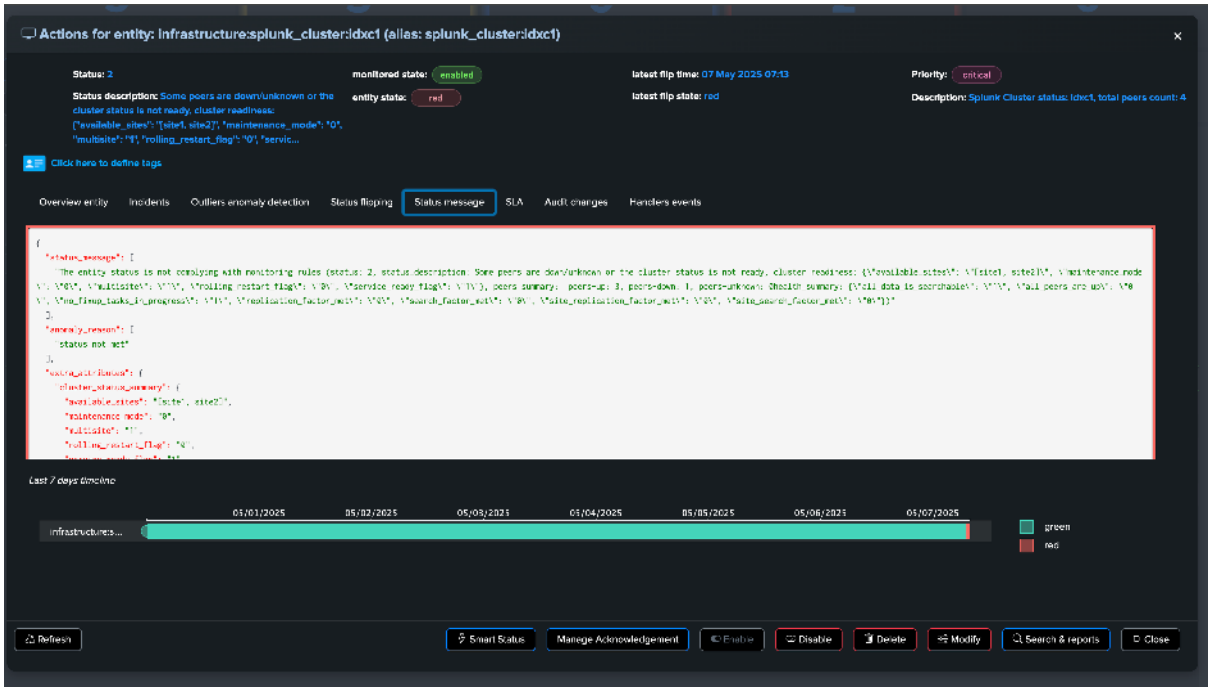

Actions for entity: infrastructure:splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2 (alias: splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2)
Status: 2
Status description: The cluster peer status is not searchable
monitored state: enabled
entity state: red
latest flip time: 07 May 2025 07:13
latest flip state: red
Priority: critical
Description: Splunk Cluster Indexer peer status, this peer is a member of the cluster: idxc1
Click here to define tags
Overview entity
Incidents
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Handlers events
Stateful alerts events & Incidents
Time
Event
07/05/2025 07:21:17.000
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Manage Acknowledgement
Enable
Disable
Delete
Modify
Search & reports
Close



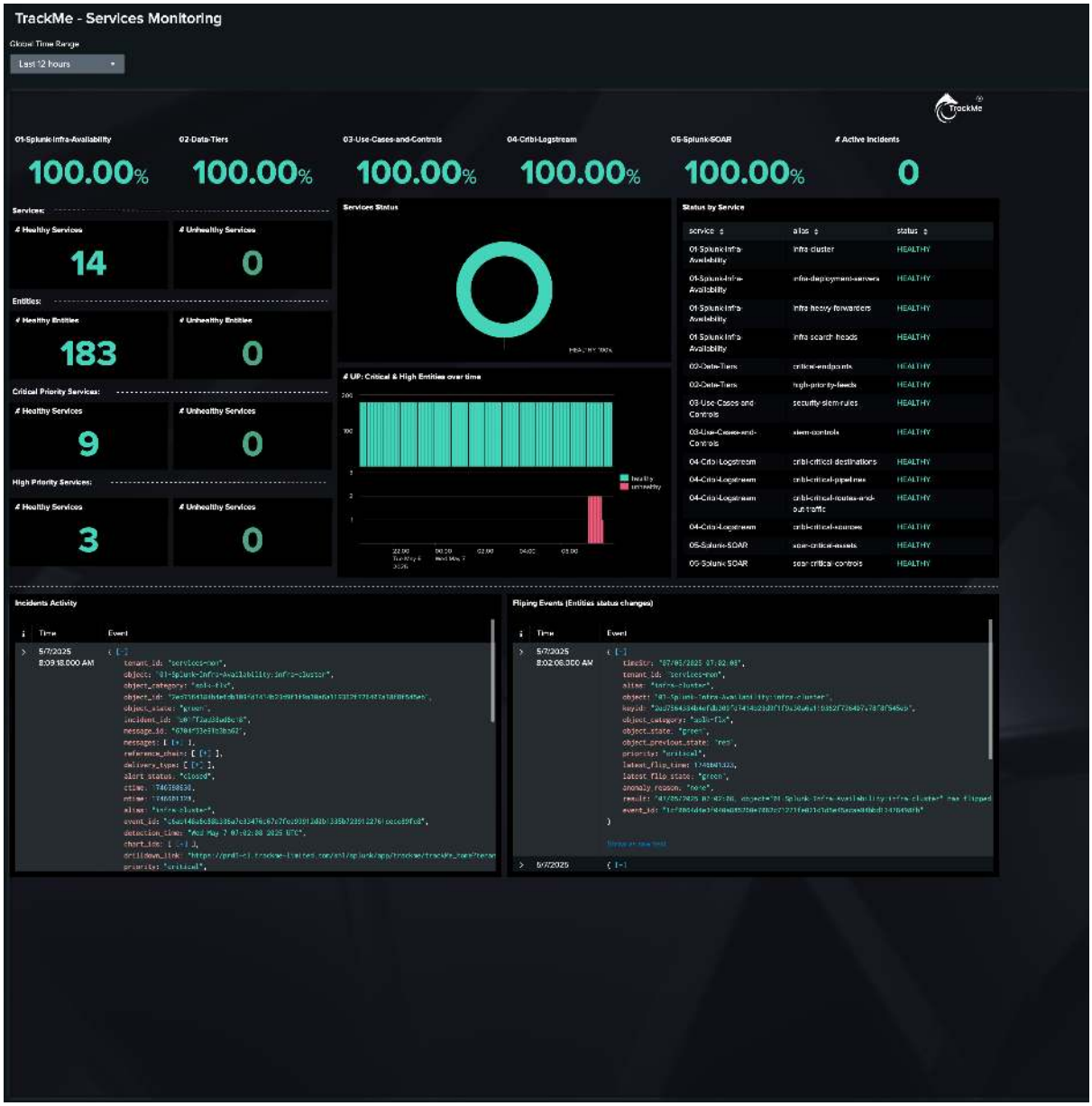


A few pictures from the global Splunk cluster entity view:





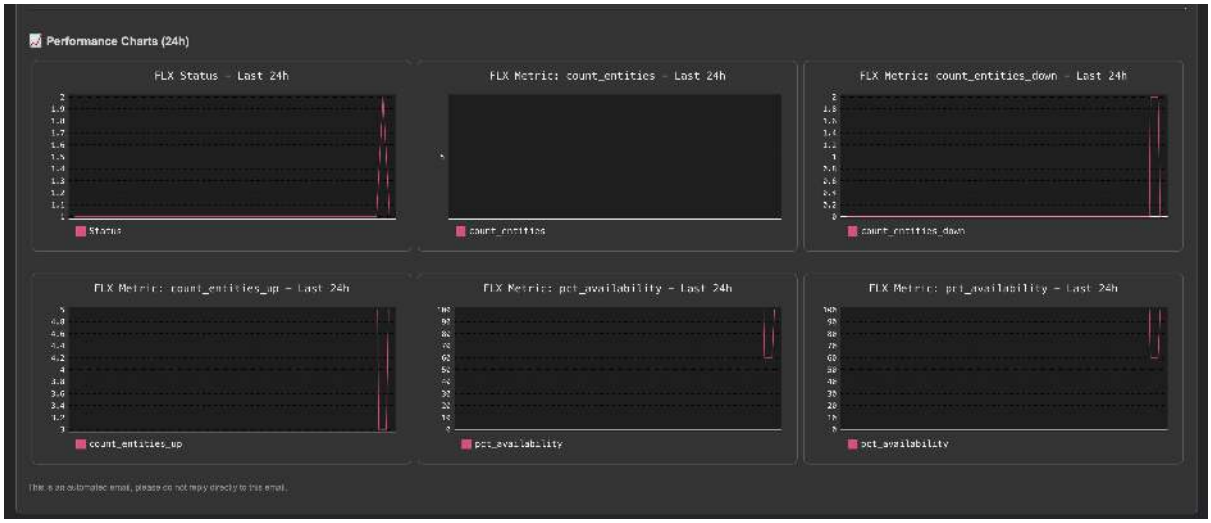
After some time, the issue is resolved, the faulty indexer is back in service, the incident is closed and the dashboard shows the updated situation for our Splunk tiers:



Closure incident Email notifications were sent to the team:

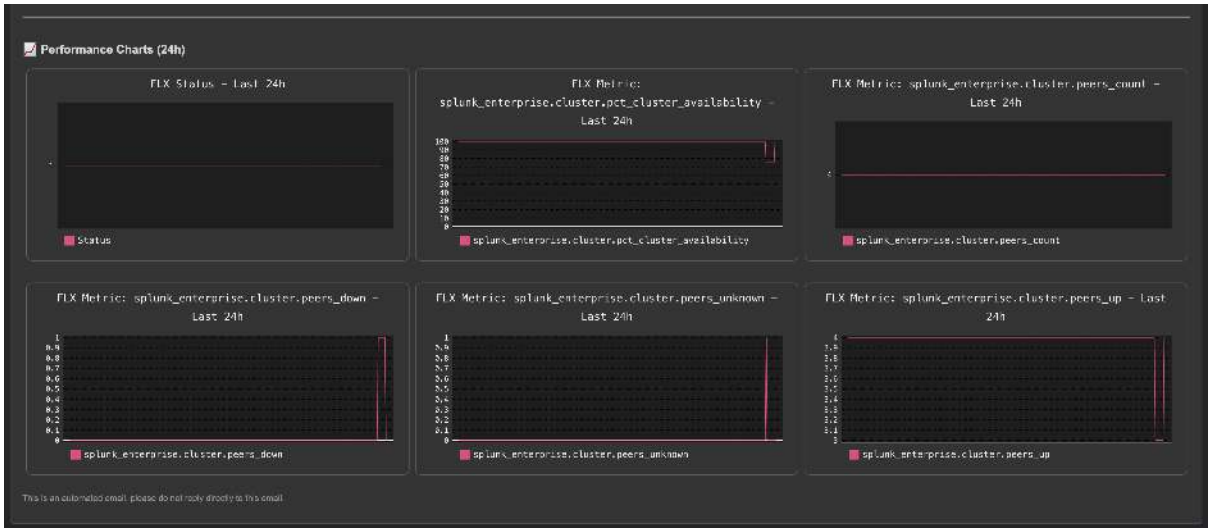
Notes: for the documentation purposes, we show notification for the service as well as the entities, but you may choose to only send notifications for the tiers services.




Performance Charts (24h)
FLX Status - Last 24h
FLX Metric: count_entities - Last 24h
FLX Metric: count_entities_down - Last 24h
FLX Metric: count_entities_up - Last 24h
FLX Metric: pct_availability - Last 24h
FLX Metric: pri_availability - Last 24h
Status
count_entities
count_entities_down
count_entities_up
pct_availability
pct_availability
This is an automated email, please do not reply directly to this email.


TrackMe Notification
Environment: https://prd5-cl.trackme-limited.com
Detection Time: Wed May 7 08:53:23 2025 UTC
Tenant: cluster-mon
Alias: splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2
Object: infrastructure:splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2
Drilldown Link
Status: green
Incident ID: 20e2e3d58fb740dd
Detailed Information:
The entity has received an incident closure update and is now in a non-alerting state:
Alias: splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2
Object: infrastructure:splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2
Object ID: 780cb24ae07125d64b448fe95803e443569b5ba56b42649ce28f8a3e7523c982
Category: splk-flx
Anomaly Reason(s):
none
Message(s):
07/05/2025 06:53:23, object="infrastructure:splunk_cluster:idxc1:prd5-cl-idx-dc1-idx2" has flipped from previous_state="red" to state="green" with anomaly_reason="none", previous_anomaly_reason="status_not_met", disruption_time="12.31"
The entity status is complying with monitoring rules (status: 1, status_description: The cluster peer status is searchable and properly balanced)
Message source: trackme:flip
Message source ID: 1e7125b5e5c7b73137c3b4758a424d38ff6b3799a3567ee0cd1f8e2238bd78cd


TrackMe Notification
Environment: https://prd5-cl.trackme-limited.com
Detection Time: Wed May 7 08:58:28 2025 UTC
Tenant: cluster-mon
Alias: splunk_cluster:idxc1
Object: infrastructure:splunk_cluster:idxc1
Drilldown Link
Status: green
Incident ID: 9bed851dd3584c9b
Detailed Information:
The entity has received an incident closure update and is now in a non-alerting state:
Alias: splunk_cluster:idxc1
Object: infrastructure:splunk_cluster:idxc1
Object ID: aa3aaa9e155c2bd5124a2aa0b0602c9ad66a0e6a7e7e266eb4507660cde2e26f
Category: splk-flx
Anomaly Reason(s):
none
Message(s):
07/05/2025 06:58:28, object="infrastructure:splunk_cluster:idxc1" has flipped from previous_state="red" to state="green" with anomaly_reason="none", previous_anomaly_reason="status_not_met", disruption_time="8.13"
The entity status is complying with monitoring rules (status: 1, status_description: All peers are up and running, peers summary: up:4, down: 0, unknown: 0 / cluster readiness: rolling_restart_flag: 0, service_ready_flag: 1, health_summary: {"all_data_is_searchable": "1", "all_peers_are_up": "1", "no_fixup_tasks_in_progress": "1", "replication_factor_met": "1", "search_factor_met": "1", "site_replication_factor_met": "1", "site_search_factor_met": "1"})
Message source: trackme:flip
Message source ID: 8167b2e84fab1d24dc5c2efa43b898af5b4ec8cc18e0d496aaae3758f7361edb



2.2.3 Use Case Demo: Fields Quality (CIM and non-CIM)

Hint

New component: Fields Quality (splk-fqm) - since TrackMe 2.1.22!

- **TrackMe 2.1.22** introduced a new component called **Fields Quality (splk-fqm)** which is designed to perform continuous fields quality assessment on Splunk data.
- This component implements the concepts described in this white paper, built into TrackMe, with flexibility, scalability and automated wizards.
- This white paper was updated to focus on the new component and its capabilities, with associated sections available in the annexes.
- **TrackMe version 2.1.23** provided several important improvements to the Fields Quality component.
- **TrackMe version 2.1.24** introduced full support for non-CIM data contexts in the Fields Quality component.
- **TrackMe version 2.1.26** slightly enhanced the performance of simulation and execution phases, and provided full support for custom break by definitions for both CIM and non CIM contexts.
- **TrackMe version 2.1.27** provided the capability to create additional monitor only collect jobs through the wizard, allowing to re-use existing quality samples events for different aggregations purposes, such as aggregating per datamodel in a different tenant than the original tenant performing the collect and pre-processing.

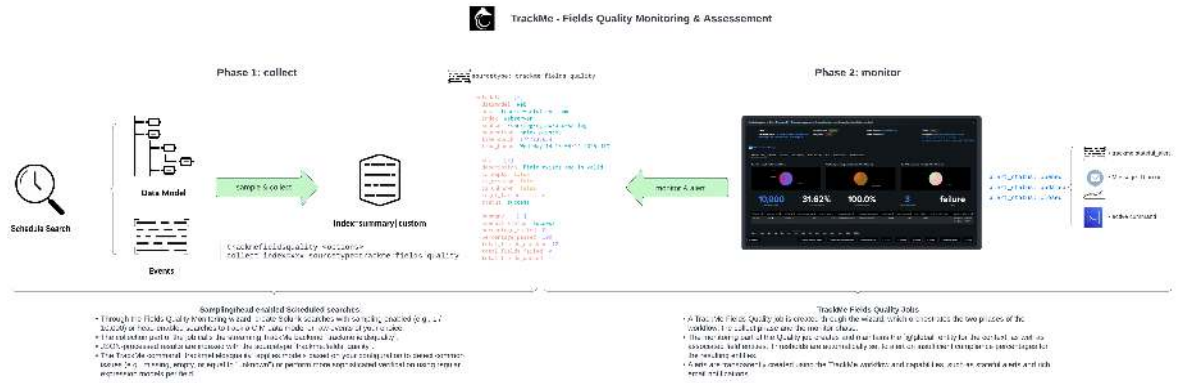
Use Case Demo: Fields Quality

- This white paper describes a new concept for performing continuous fields quality assessment on Splunk data.
- This leverages different scalable Splunk techniques and TrackMe components to perform this task.
- Fields quality assessment is a crucial aspect of Splunk monitoring, as it helps ensure that the data is accurate and consistent and ready to serve use cases.
- By implementing this, you will be able to build a solid, robust, and scalable solution to monitor the quality of fields parsing for your various sourcetypes in Splunk, as well as receive automated alerts when fields quality issues are detected.
- These concepts are applicable to both Splunk Common Information Model (CIM) and non-CIM data.

- This work was made possible through the support and deep collaboration of several major fellow TrackMe customers. Thank you!

High level workflow and diagram

The following diagram shows the high-level workflow for fields quality assessment:



From a high-level perspective, the workflow is as follows:

In the Fields Quality component, you create collect jobs which consist of two phases

Phase 1: collect

- Through the Collect job creation wizard, you define a new `collect` job which is tied to a data context (a Splunk search constraint) associated with a data dictionary.
- The data dictionary can be generated by TrackMe according to the CIM data model, or you can set the collect job to use a previously generated data dictionary. (Dictionaries can be shared between collect jobs)
- Once validated and executed, the collect job starts collecting and processing event samples, stored in a summary index and associated with a source value unique per collect job.

Phase 2: monitor

- In the collect job, a second set of `monitor` trackers are created, which are set to be executed automatically 10 minutes after the collect job is executed.
- The `monitor` job parses and processes the JSON events generated by the collect job, and discovers and maintains TrackMe entities associated with the collect job.

Hint

Common Information Model (CIM) context versus raw events context

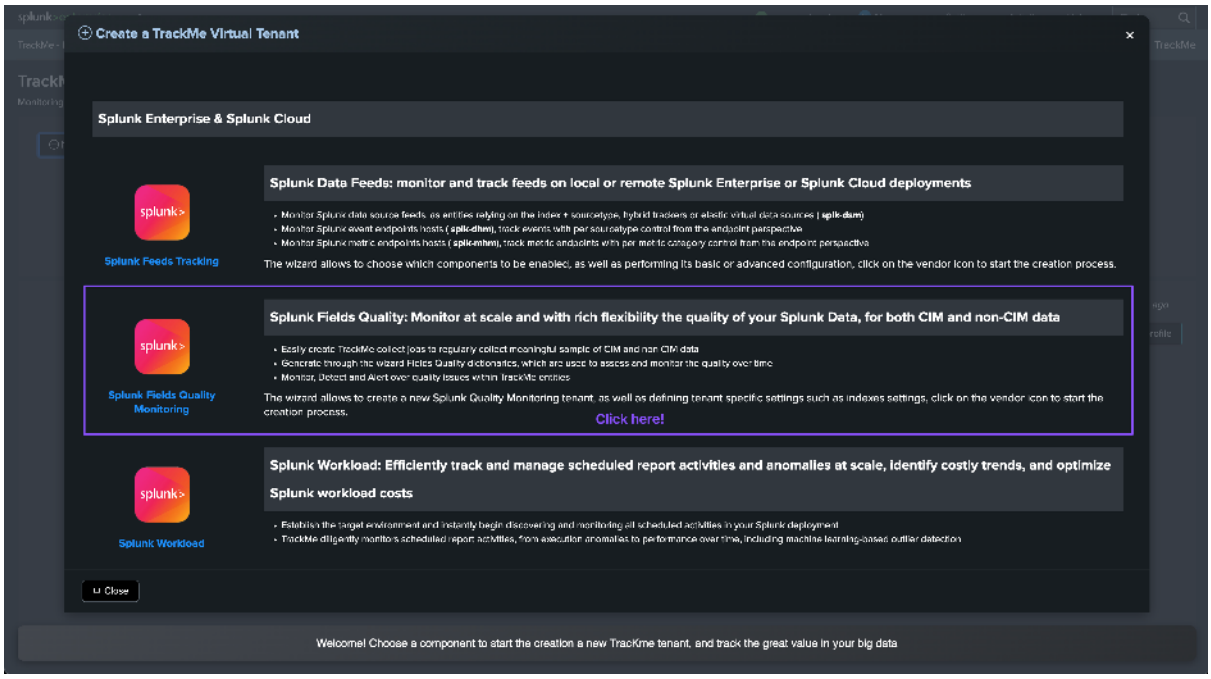
- These concepts apply equally to both the Common Information Model and any raw events in Splunk.
- CIM parsing quality is generally a critical topic when use cases heavily rely on CIM, but there can be many use cases where you need to ensure the parsing quality outside of a CIM context.
- These concepts are applicable to both CIM and non-CIM data, and you can simply use the same approach for both.

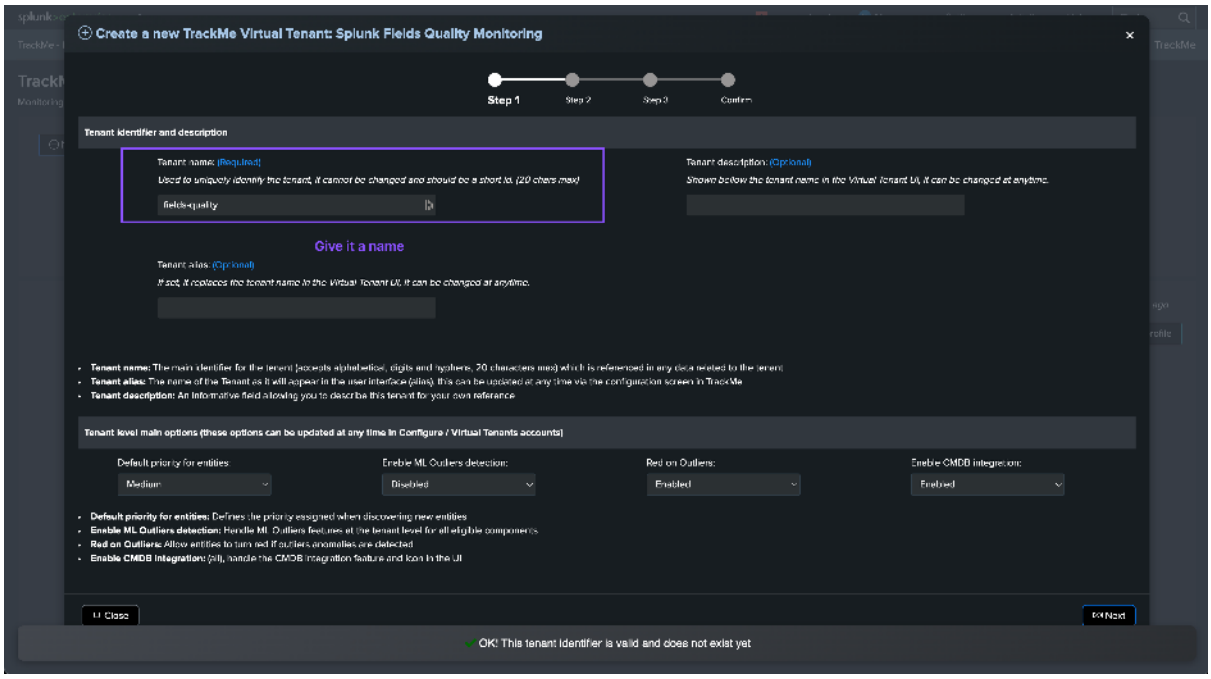
Preparing a Virtual Tenant for the Fields Quality component

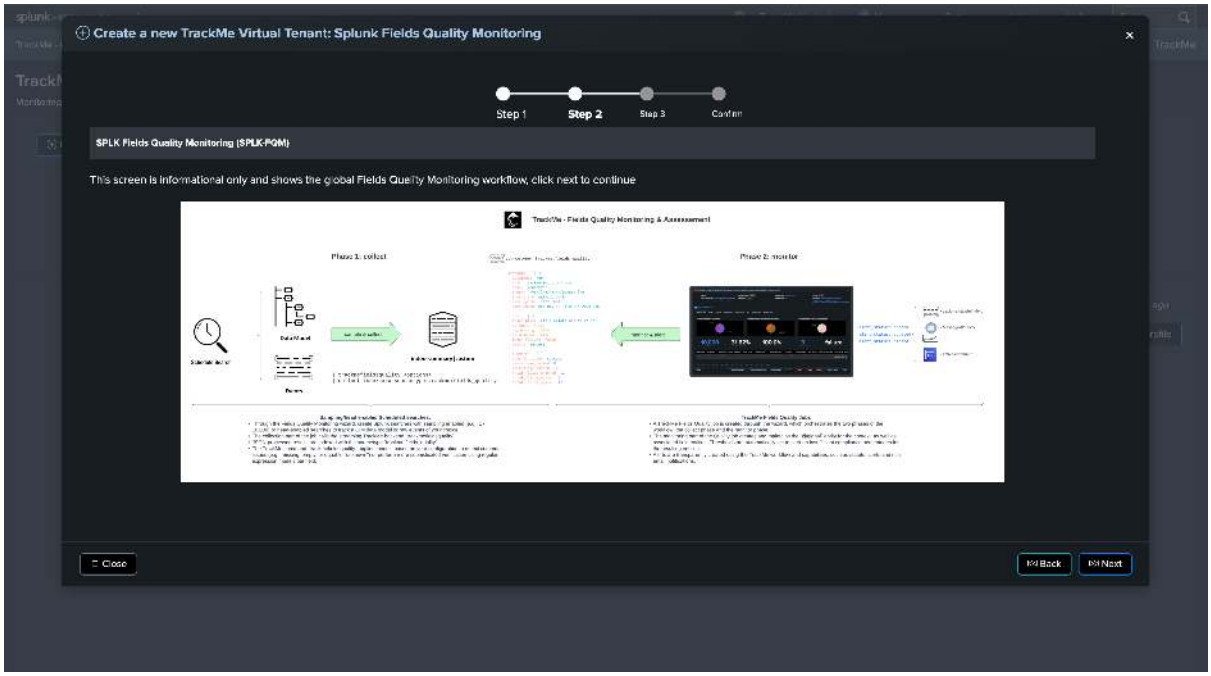
You can either create a fresh new Virtual Tenant in a couple of clicks, or add the component to an existing Virtual Tenant.

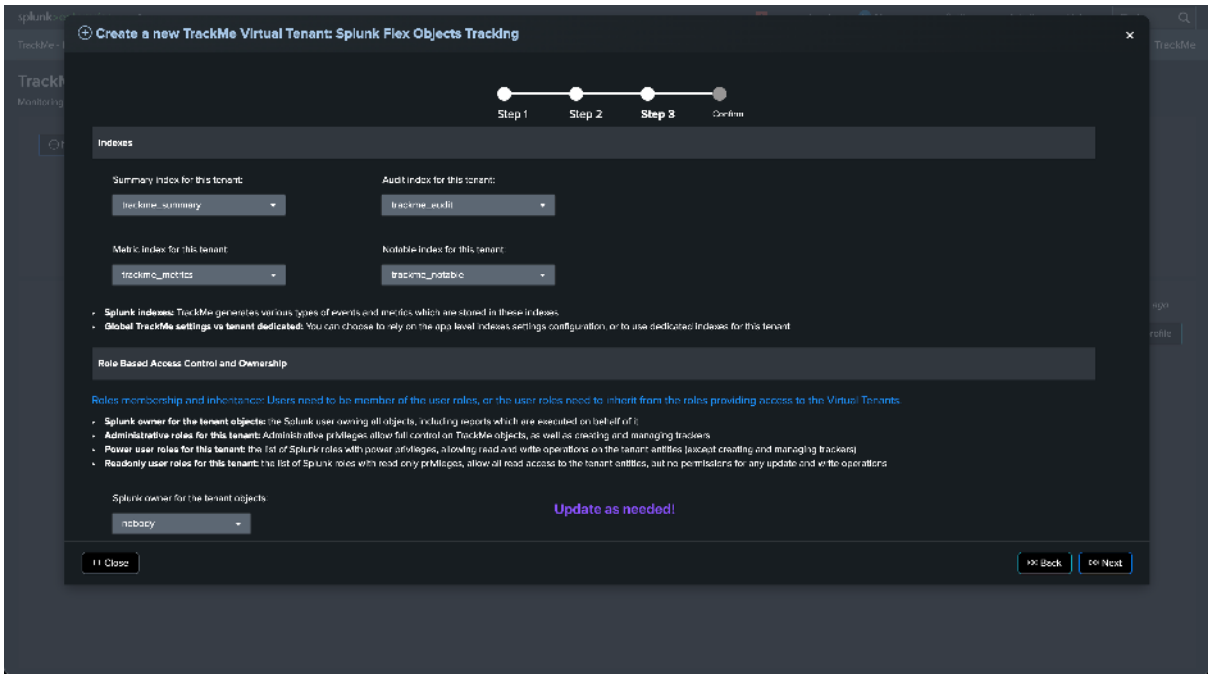
Creating a new Virtual Tenant

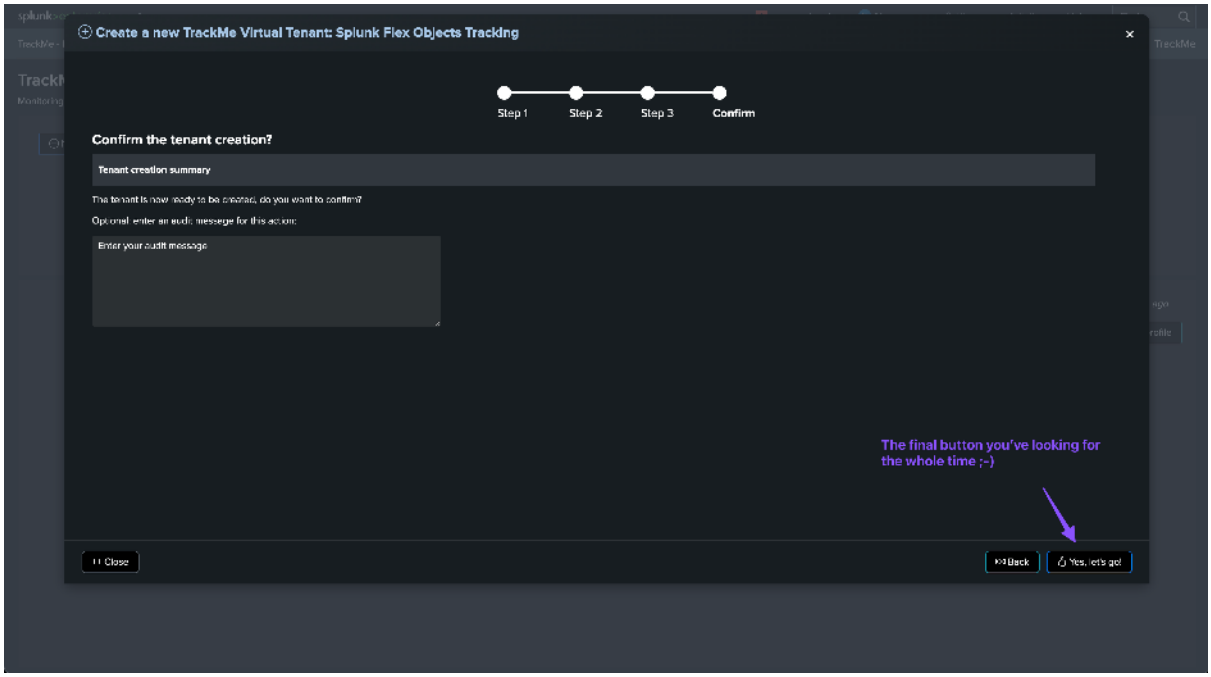
To create a new Virtual Tenant, you can follow these steps:





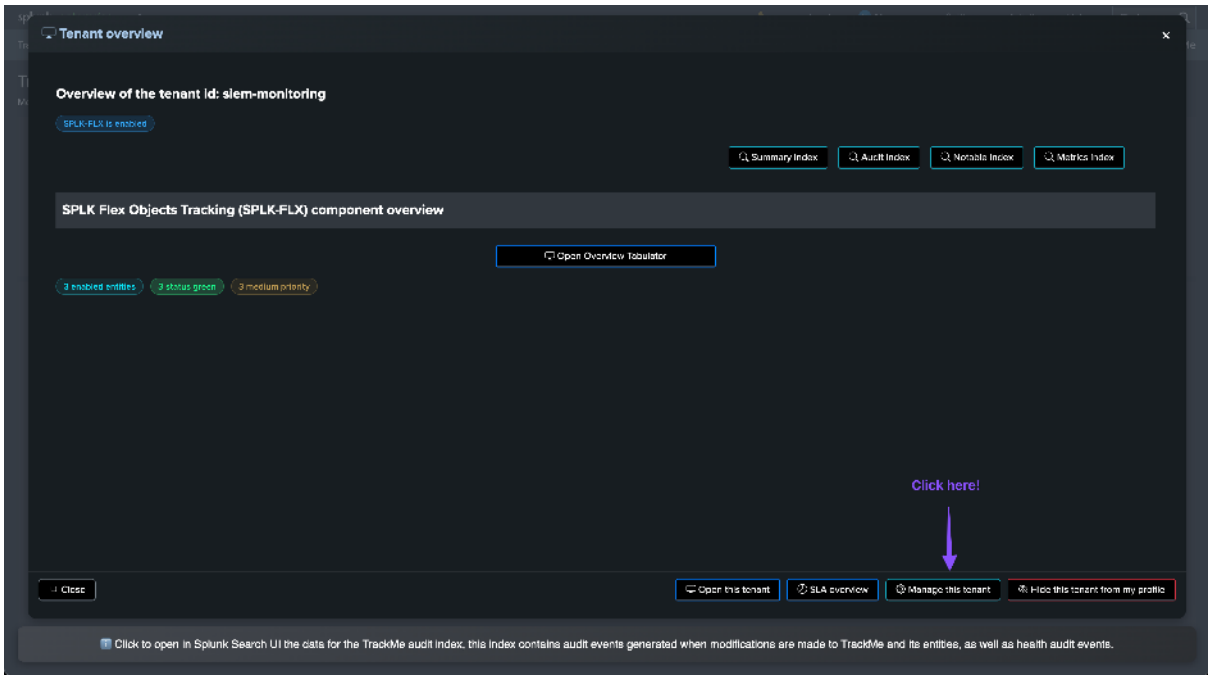

Create a new TrackMe Virtual Tenant: Splunk Fields Quality Monitoring
Step 1
Step 2
Step 3
Confirm
SPLK Fields Quality Monitoring (SPLK-FQM)
This screen is informational only and shows the global Fields Quality Monitoring workflow, click next to continue
Close
Back
Next

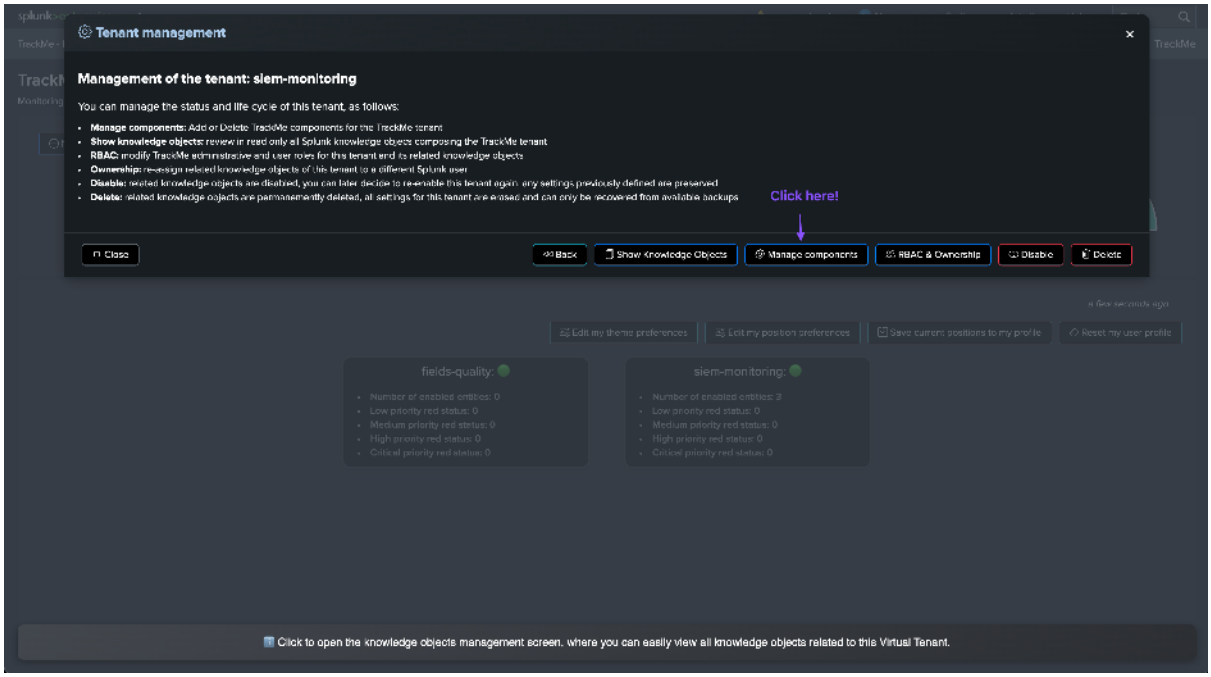

Create a new TrackMe Virtual Tenant: Splunk Flex Objects Tracking
Step 1
Step 2
Step 3
Confirm
Indexes
Summary index for this tenant:
Audit index for this tenant:
Metric index for this tenant:
Notable index for this tenant:
Role Based Access Control and Ownership
Update as needed!
Close
Back
Next

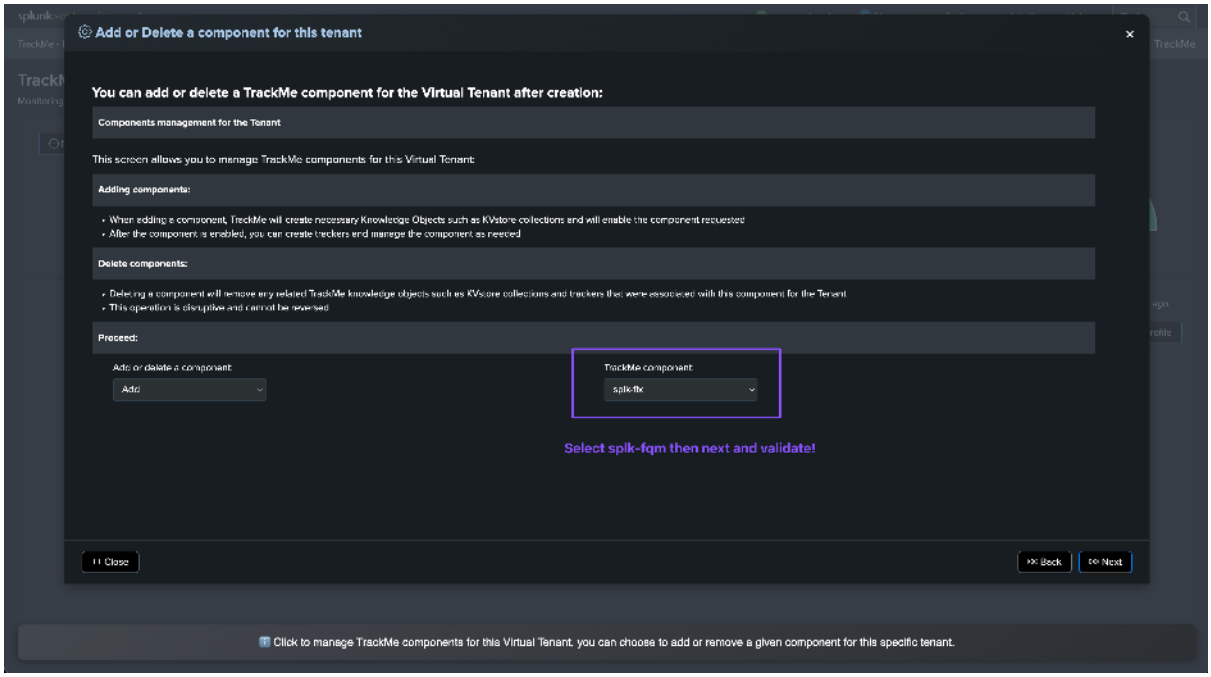


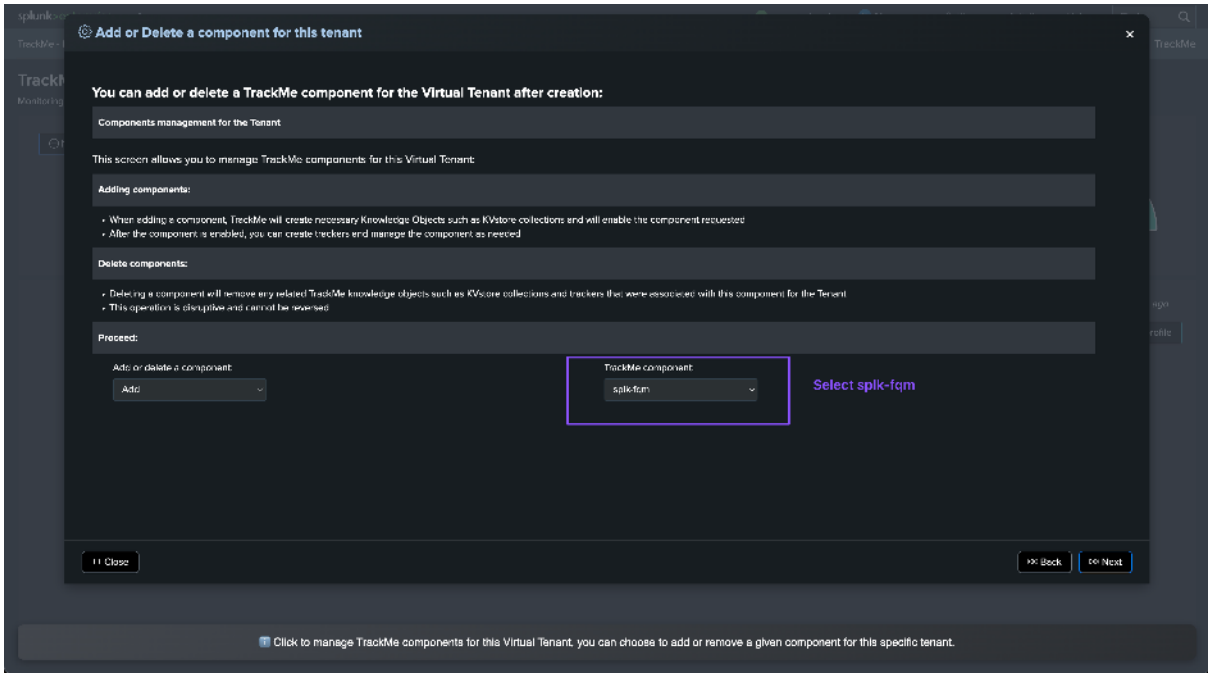
Adding the Fields Quality component to an existing Virtual Tenant

You can also add the Fields Quality component to an existing Virtual Tenant:




Tenant management
Management of the tenant: siem-monitoring
You can manage the status and life cycle of this tenant, as follows:
Manage components: Add or Delete TrackMe components for this TrackMe tenant
Show knowledge objects: review in read only all Splunk knowledge objects composing the TrackMe tenant
RBAC: modify TrackMe administrative and user roles for this tenant and its related knowledge objects
Ownership: re-assign related knowledge objects of this tenant to a different Splunk user
Disable: related knowledge objects are disabled, you can later decide to re-enable this tenant again: any settings previously defined are preserved
Delete: related knowledge objects are permanently deleted, all settings for this tenant are erased and can only be recovered from available backups
Click here!
Close
Back
Show Knowledge Objects
Manage components
RBAC & Ownership
Disable
Delete
Edit my theme preferences
Edit my position preferences
Save current positions to my profile
Reset my user profile
fields-quality:
Number of enabled entities: 0
Low priority red status: 0
Medium priority red status: 0
High priority red status: 0
Critical priority red status: 0
siem-monitoring:
Number of enabled entities: 3
Low priority red status: 0
Medium priority red status: 0
High priority red status: 0
Critical priority red status: 0
Click to open the knowledge objects management screen, where you can easily view all knowledge objects related to this Virtual Tenant.


Add or Delete a component for this tenant
You can add or delete a TrackMe component for the Virtual Tenant after creation:
Components management for the Tenant
This screen allows you to manage TrackMe components for this Virtual Tenant:
Adding components:
When adding a component, TrackMe will create necessary Knowledge Objects such as KVstore collections and will enable the component requested
After the component is enabled, you can create trackers and manage the component as needed
Delete components:
Deleting a component will remove any related TrackMe knowledge objects such as KVstore collections and trackers that were associated with this component for the Tenant
This operation is disruptive and cannot be reversed
Proceed:
Add or delete a component:
Add
TrackMe component:
splk-fix
Select splk-fqm then next and validate!
Close
Back
Next
Click to manage TrackMe components for this Virtual Tenant, you can choose to add or remove a given component for this specific tenant.



Creating your first collect job

Hint

About CIM and non CIM data contexts:

- Since TrackMe 2.1.24, the Fields Quality component supports both CIM and non-CIM data contexts.
- The main difference between creating a collect job for non CIM contexts relies on the creation of the data dictionary in the wizard and in the same time than the collect job creation.
- For non CIM contexts, you need to define the fields to be monitored by declaring the fields of interests in the data dictionary.
- On the other hand and for CIM contexts, TrackMe can generate a new data dictionary for you and based on the selected CIM data model.

Once we have a Virtual Tenant with the Fields Quality component enabled, we can create our first collect job.

To start the wizard, click on `Manage: trackers & dictionaries` then choose an action button:

`Create a new Quality job (for CIM, generate or use existing dictionary):`

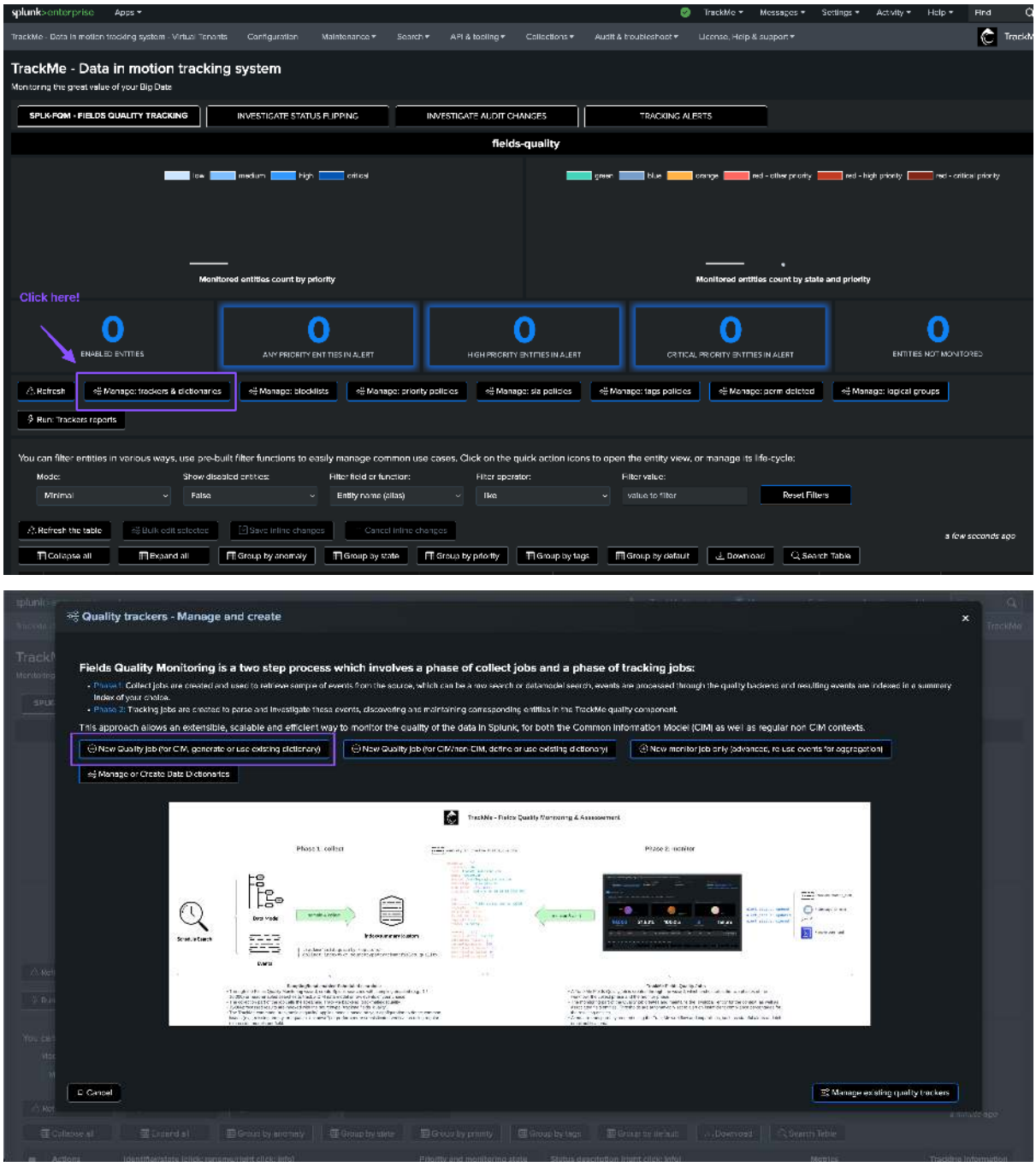
- This wizard allows creating a new collect job from scratch, and which targets a Common Information Model (CIM) only context.
- The wizard can generate the data dictionary for you, based on the selected CIM data model, and some of your preferences.
- You can also select an existing data dictionary to be used for the collect job, if any.

`Create a new Quality job (for CIM or non CIM, define or use existing dictionary):`

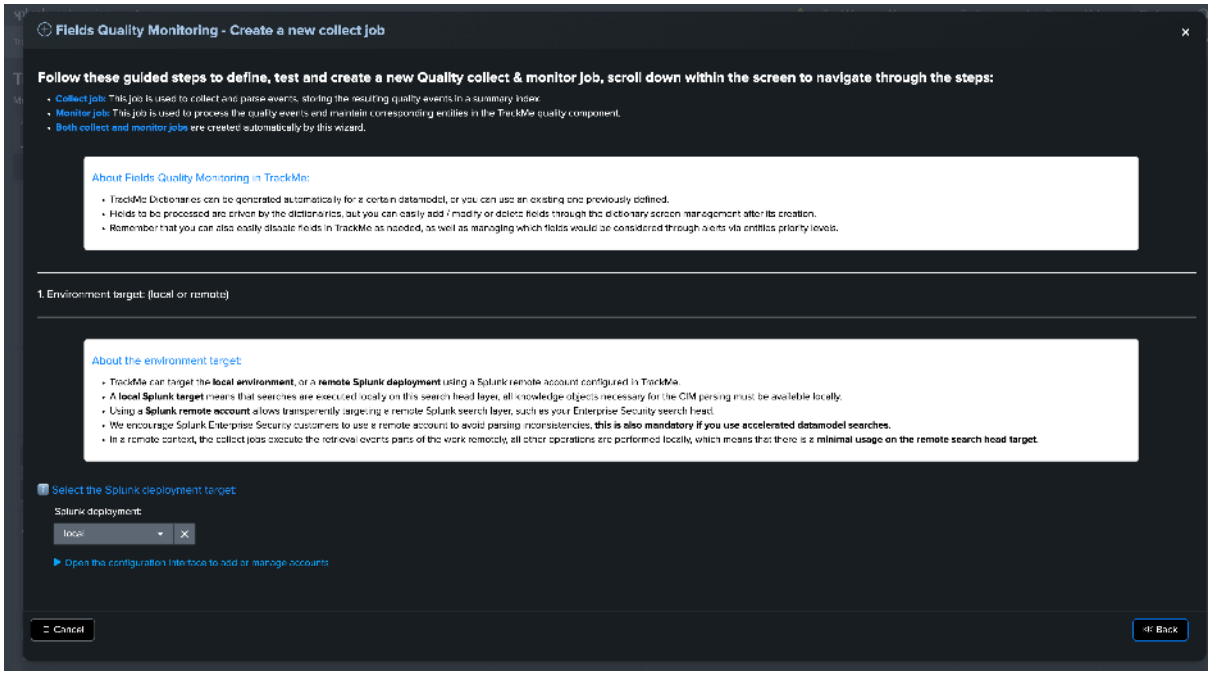
- This wizard allows creating a new collect job for both CIM and non CIM contexts, the main difference being the way the data dictionary is defined.
- You can either choose to create and define the data dictionary within the wizard, or use an existing data dictionary.
- It is intended primarly to be used for non CIM contexts, or contexts where you want to have full control over the data dictionary definition at the stage of the collect job creation.

`New monitor job only (advanced, re-use events for aggregation):`

- This is an advanced option, intented to allow creating a monitor only collect job, and re-use the events generated by a previous collect job for different aggregation purposes.
- It can be used for instance to track fields quality for existing collectors, but in a different way, such as aggregating at the level of the datamodel rather than datamode/index/sourcetype, without re-collecting and pre-processing the same sources, nor affecting the source tenant way of aggregating and processing the quality entities.
- To use this feature, you first need an existing configured collect job and you should have identified your use case, and particularly the aggregation definition. (break by definition)



Wizard welcoming screen



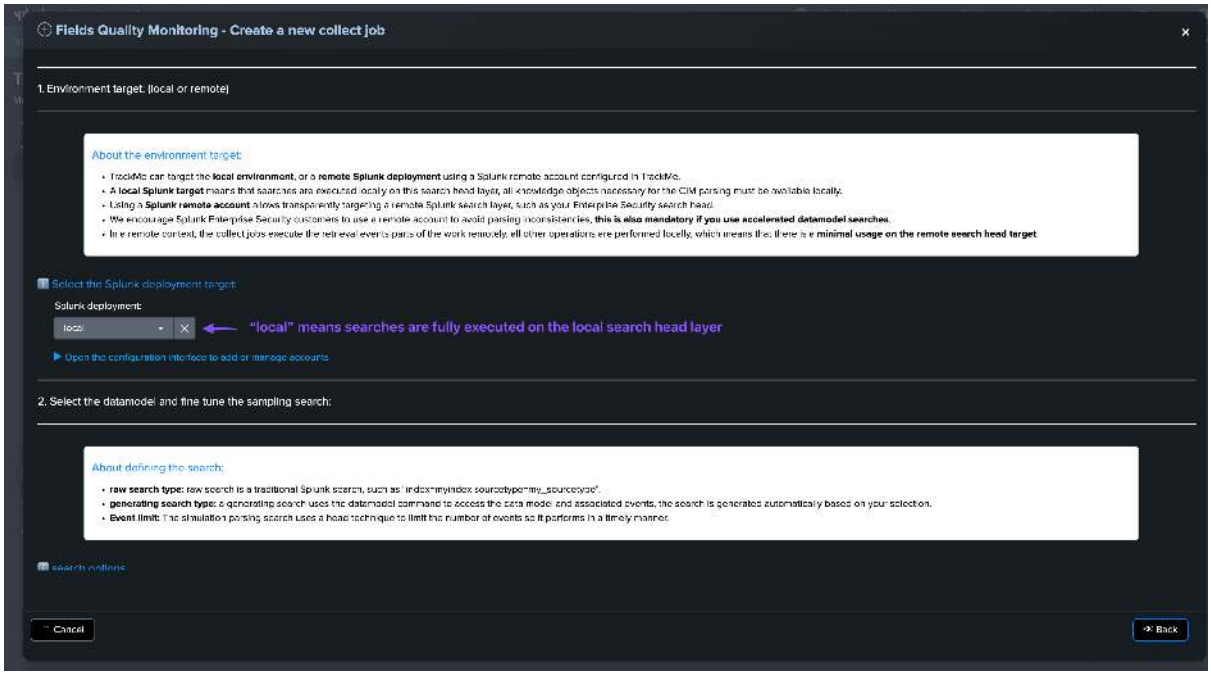
Environment target selection

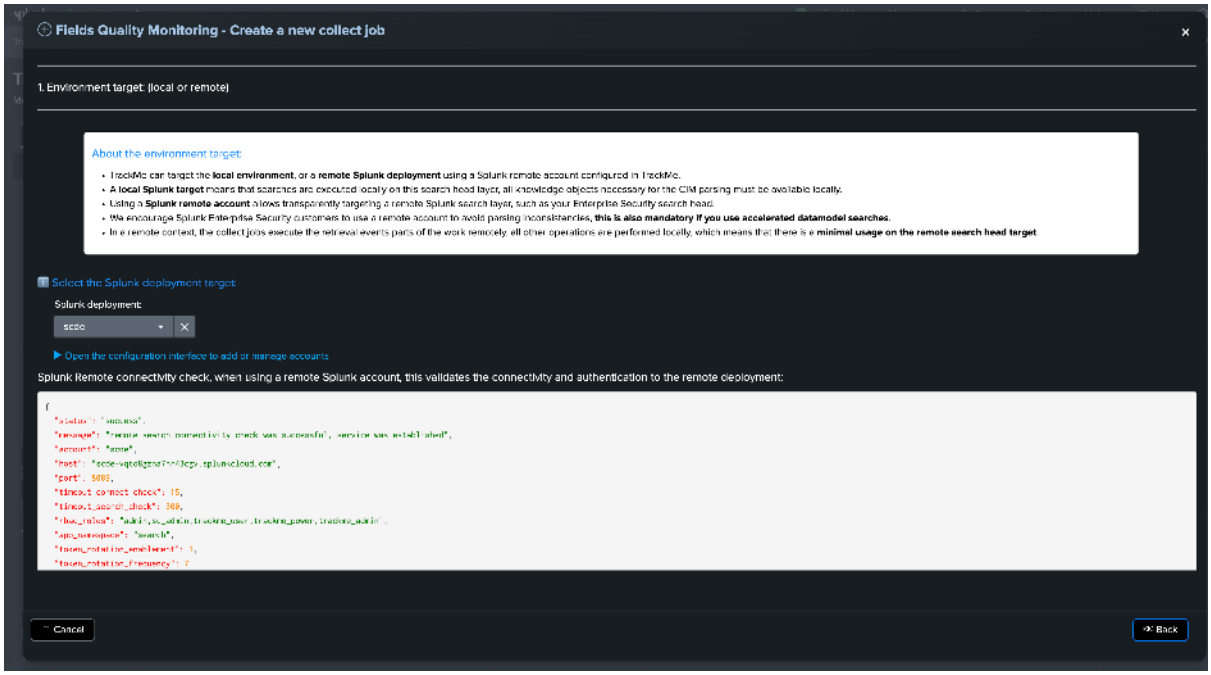
The first step is to select whether the environment target is the local search head or a remote search head:

Hint

About the environment target:

- TrackMe can target the local environment, or a remote Splunk deployment using a Splunk remote account configured in TrackMe.
- A local Splunk target means that searches are executed locally on this search head layer. All knowledge objects necessary for the CIM parsing must be available locally.
- Using a Splunk remote account allows transparently targeting a remote Splunk search layer, such as your Enterprise Security search head.
- We encourage Splunk Enterprise Security customers to use a remote account to avoid parsing inconsistencies. This is also mandatory if you use accelerated datamodel searches.
- In a remote context, the collect jobs execute the retrieval events parts of the work remotely. All other operations are performed locally, which means there is minimal usage on the remote search head target.





Datamodel and root search definition

Hint

About CIM and non CIM data contexts:

- The datamodel selection applies to CIM contexts only.
- When creating a collect job for non CIM contexts, you simply need to define the root search constraint.

The next step is to define the type of the search to be executed, the datamodel selection and finally the root search constraints:

About defining the search:

- Raw search type: A raw search is a traditional Splunk search, such as "index=myindex sourcetype=my_sourcetype".
- Generating search type: A generating search uses the datamodel command to access the data model and associated events. The search is generated automatically based on your selection.
- Event limit: The simulation parsing search uses a head technique to limit the number of events so it performs in a timely manner.

Hint

About your design strategy:

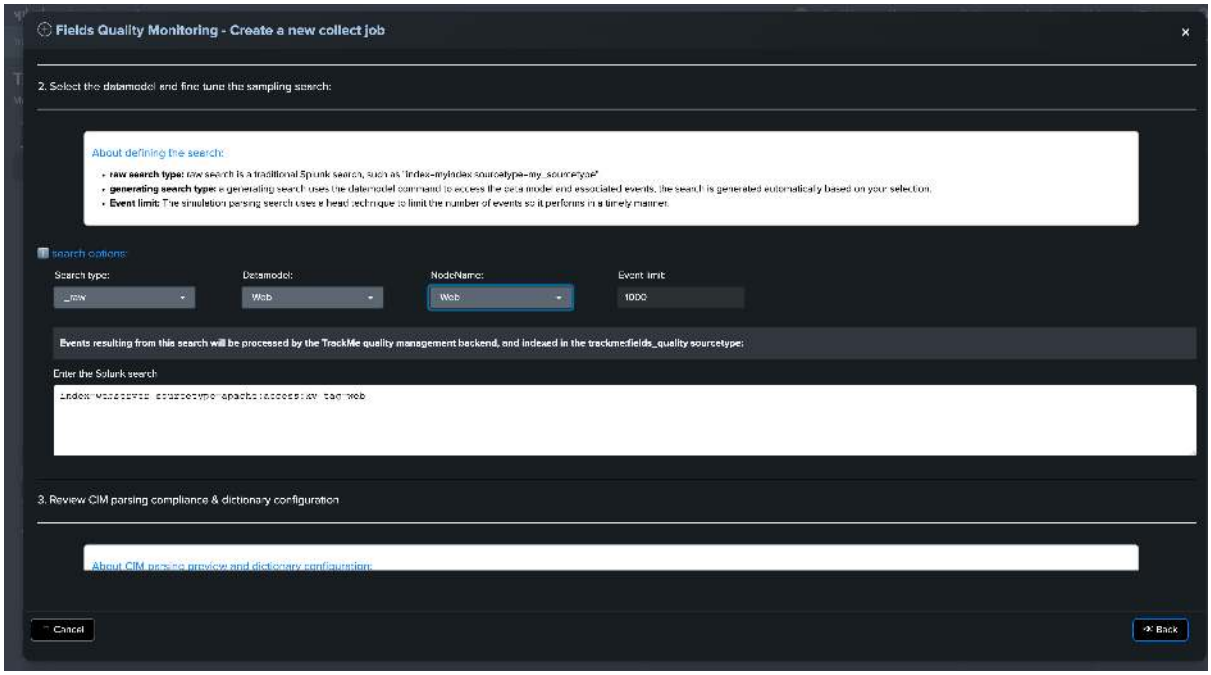
- The best strategy is the one that matches your preferences, custom context and objectives.
- A single collect job can technically handle multiple indexes, sourcetypes, and so forth, BUT it is tied to a specific datamodel and data dictionary.
- Therefore, at minimum you want to segregate your collect jobs per datamodel and sourcetype. (or context feeds, for instance one collect job for WinEvent logs, one for Linux secure logs, etc.)
- To ensure a good level of sampling, you can also be as granular as needed, for instance one collect job per datamodel/index/sourcetype. (or context feed)

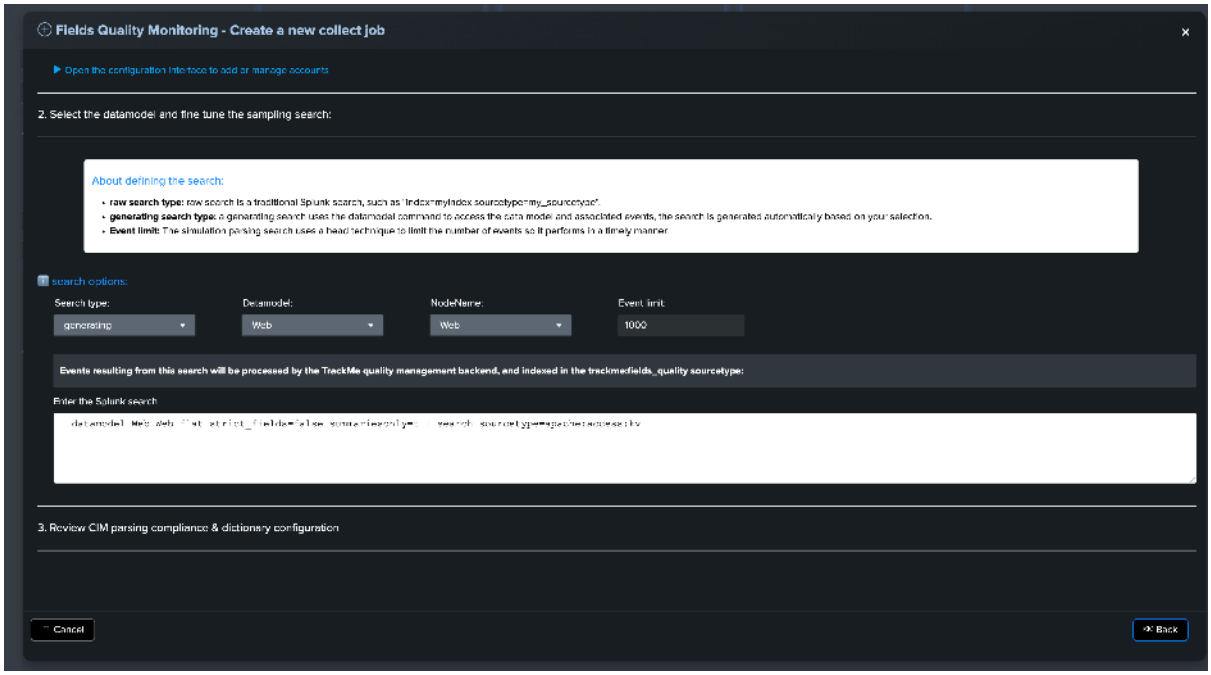
Hint

About search performance, costs, raw versus generating searches:

- Overall, the collect jobs are very efficient and inexpensive.
- TrackMe implements two techniques: the `head` technique and the `sampling` technique to produce fast and performant collect phase searches.
- The monitor phase is dependent on the associated collect job events produced and is also very efficient.
- Note that when selecting a generating search, TrackMe generates the base root search automatically, according to the selected datamodel.
- Finally, generating searches are direct consumers of the datamodel and therefore more restricted in context, while raw searches offer all levels of flexibility in terms of filters.

Examples of both approaches:





Example of a raw search definition targeting the Web datamodel context:

```
index=webserver sourcetype=apache:access:kv tag=web
```

Example of a generating search definition targeting the Web datamodel context:

```
| datamodel Web Web flat strict_fields=false summariesonly=t | search␣
↪sourcetype=apache:access:kv
```

We generally recommend having at minimum one collect job per datamodel/sourcetype.

Data dictionary generation or selection (for CIM context)

About CIM parsing preview and dictionary configuration:

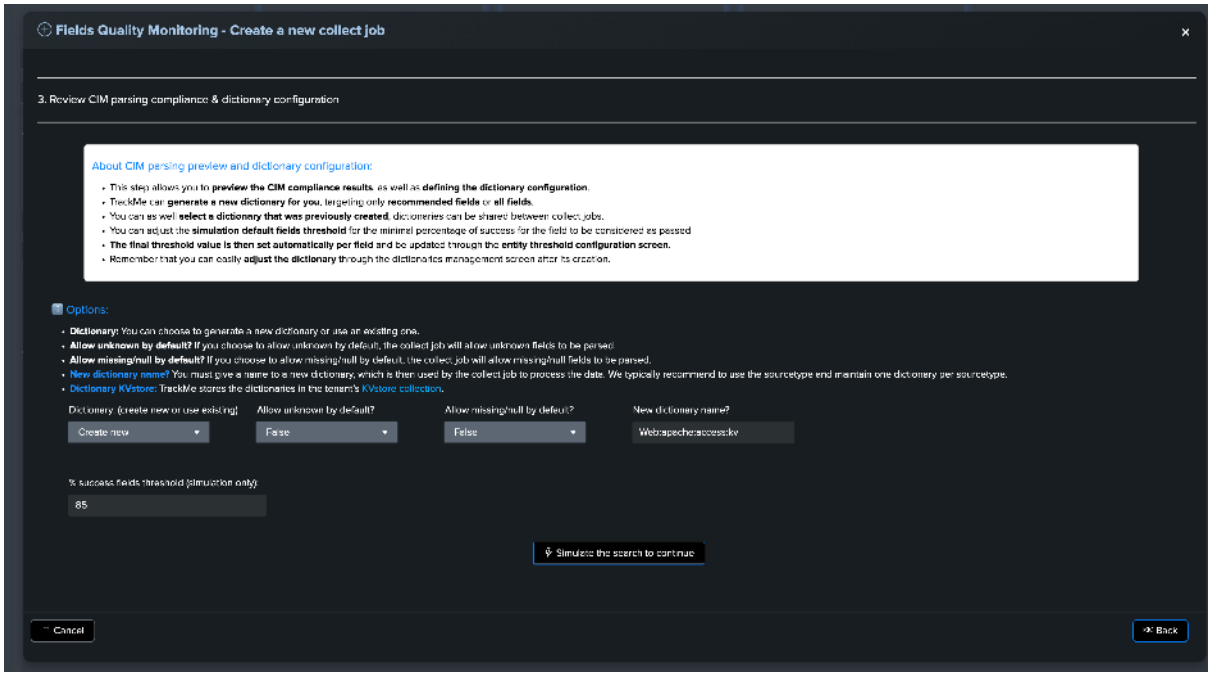
- This step allows you to preview the CIM compliance results, as well as define the dictionary configuration.
- TrackMe can generate a new dictionary for you, targeting only recommended fields or all fields.
- You can also select a dictionary that was previously created. Dictionaries can be shared between collect jobs.
- You can adjust the simulation default fields threshold for the minimal percentage of success for the field to be considered as passed.
- The final threshold value is then set automatically per field and can be updated through the entity threshold configuration screen.
- Remember that you can easily adjust the dictionary through the dictionaries management screen after its creation.

Hint

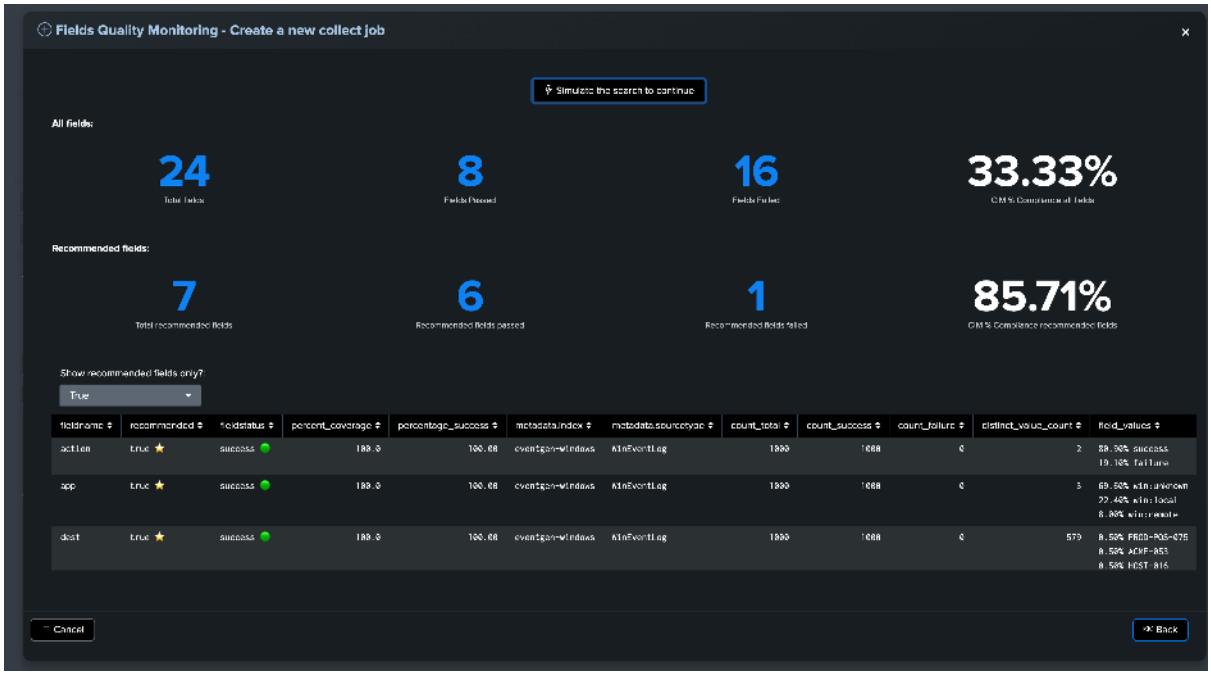
Naming the data dictionary:

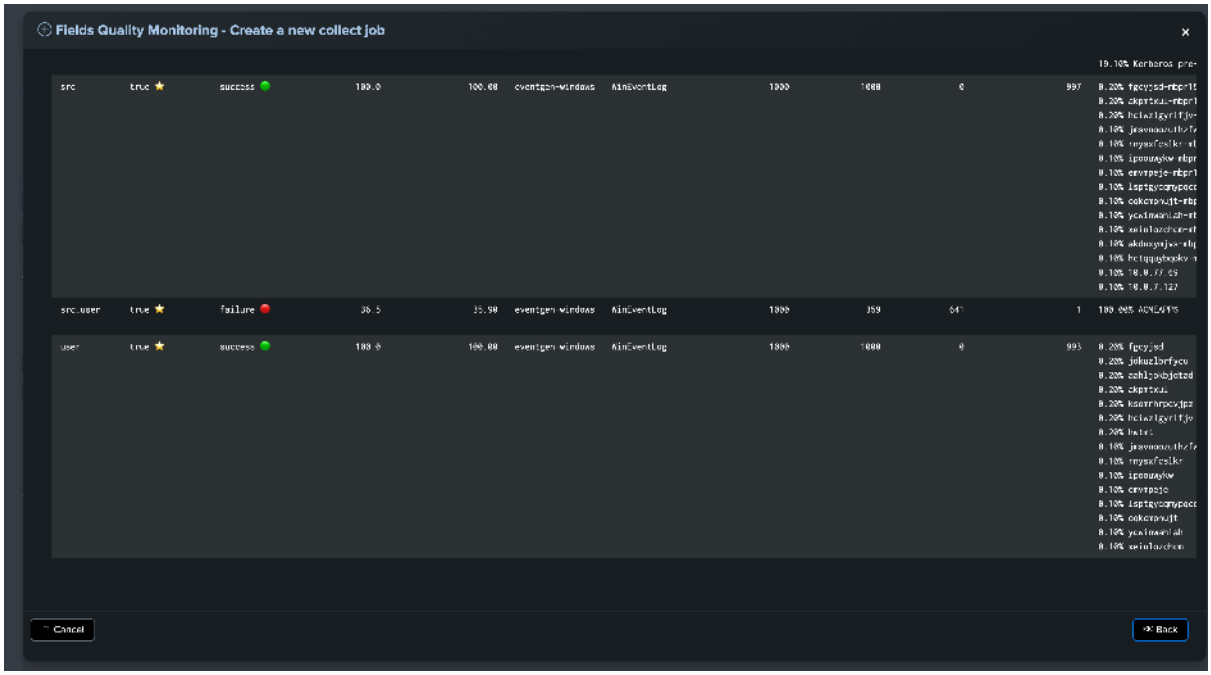
- The data dictionary name is defined in this step input field.
- We recommend having a meaningful naming convention, such as `<datammodel>:<sourcetype>`
- Remember that dictionaries can be shared between collect jobs.
- In our example, we name our new dictionary `Web:apache:access:kv`

Generate a new dictionary, or select an existing one, and adjust default behavior and thresholds:



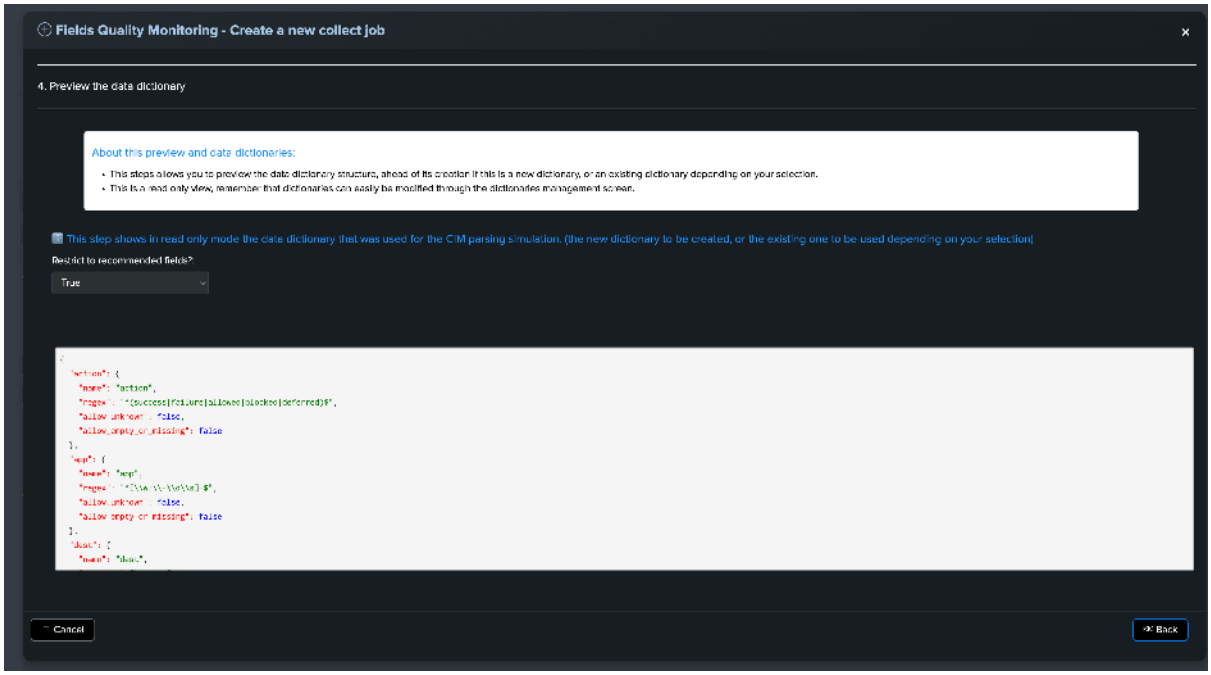
Finally, execute the sampling search to preview the CIM compliance results, which shows key information about the expected results:





Previewing the data dictionary (CIM context only)

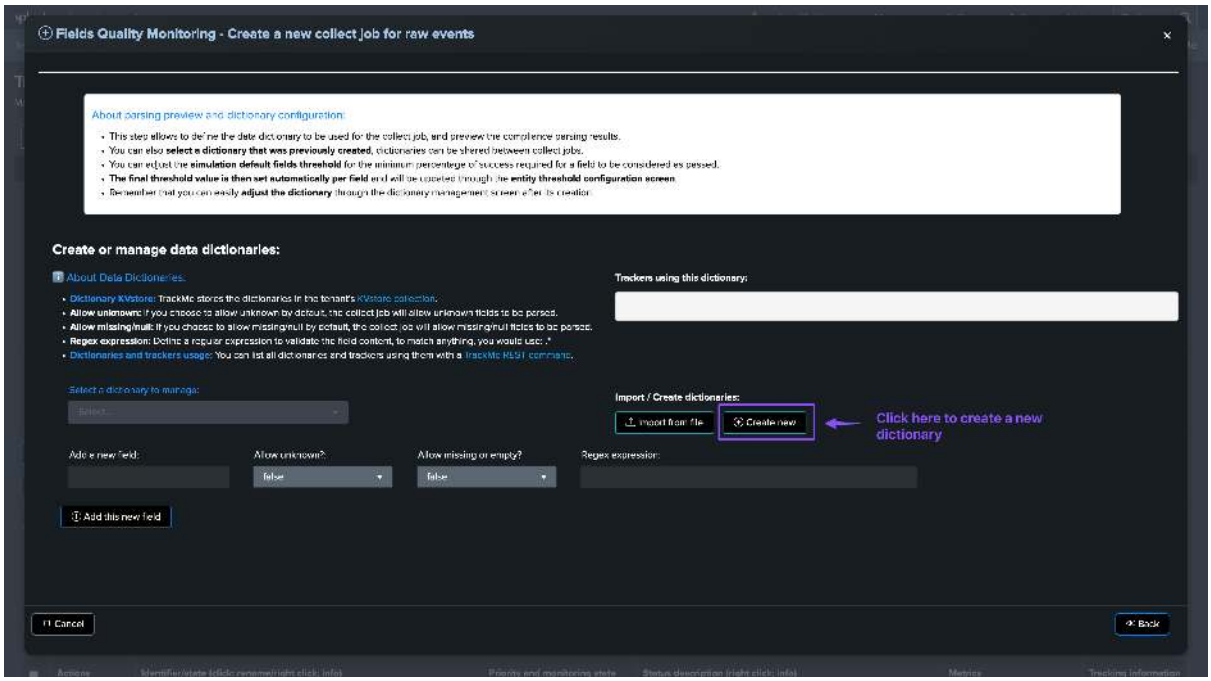
This part of the wizard is informational only and shows a preview of the structure of the data dictionary:

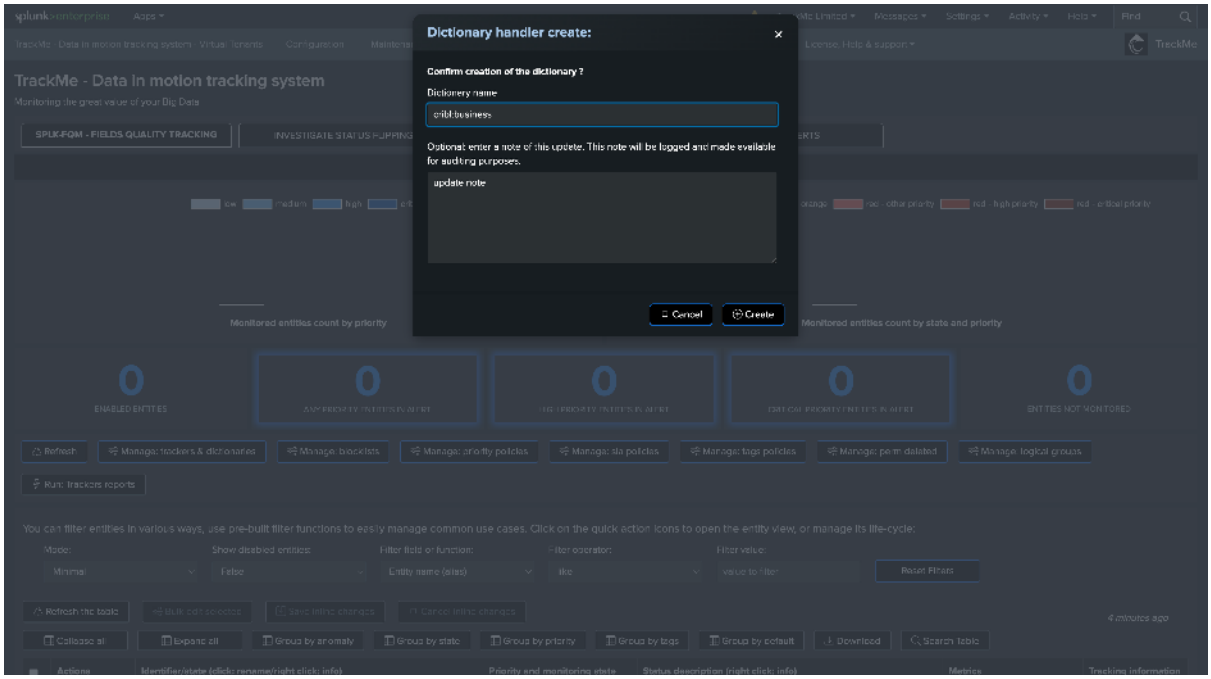


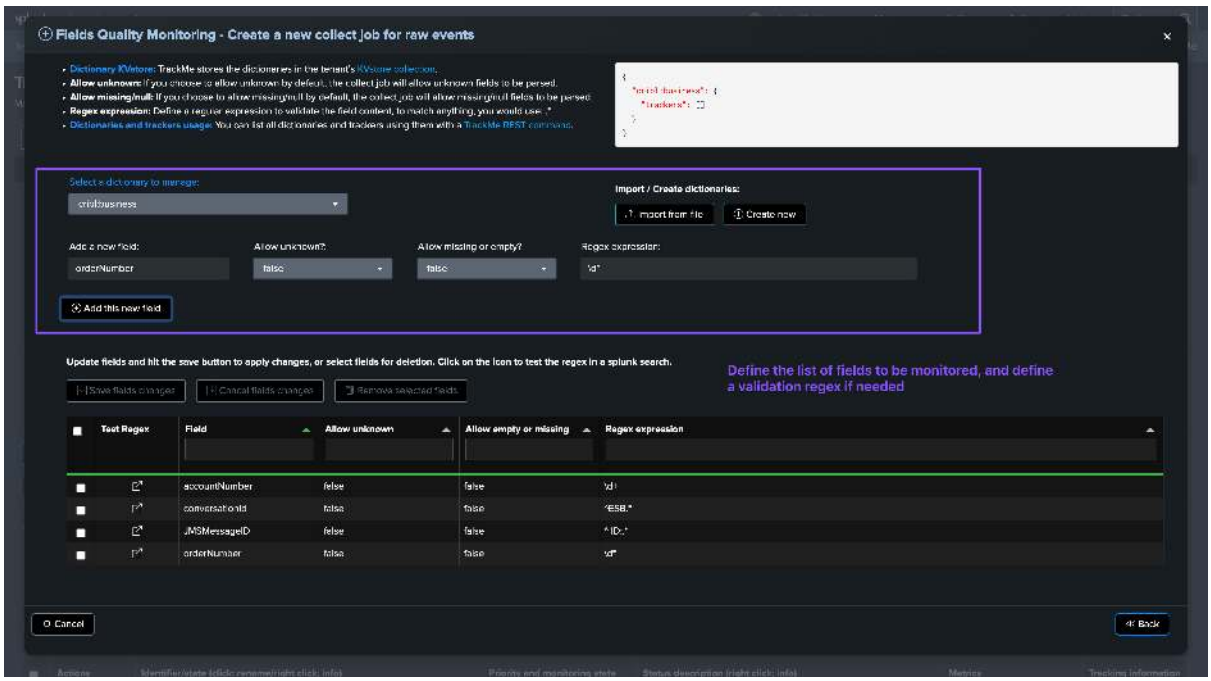
Data dictionary generation or selection (for non CIM context)

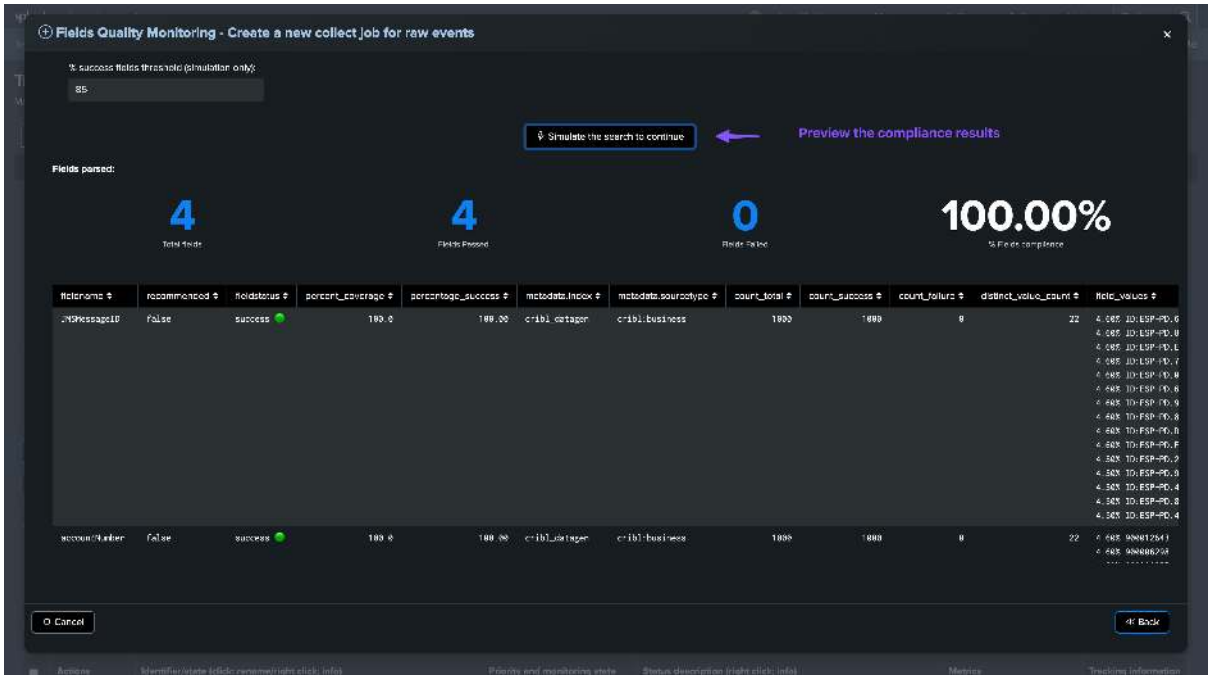
For non CIM contexts, you need to define the fields to be monitored by declaring the fields of interests in the data dictionary.

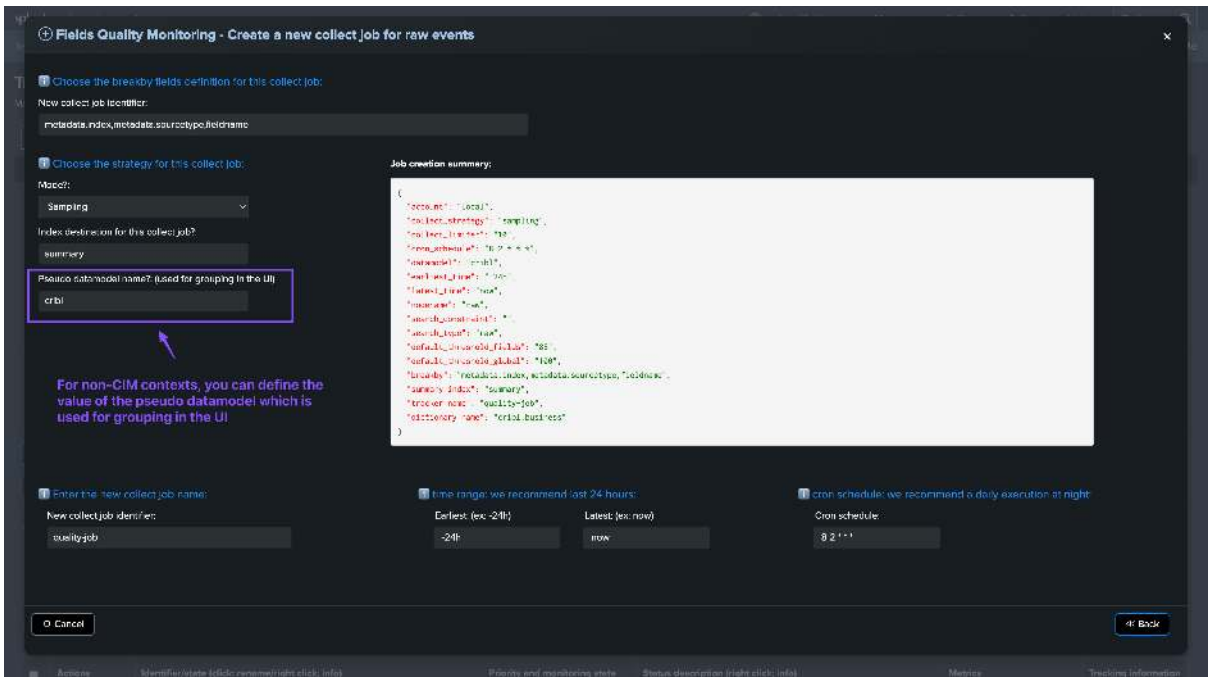
The data dictionary definition step can be performed before the collect job creation via the `Manage: dictionaries` screen, or during the collect job creation wizard.




Dictionary handler create:
Confirm creation of the dictionary ?
Dictionary name:
cribl:business
Optional: enter a note of this update. This note will be logged and made available for auditing purposes.
update note
Cancel
Create
TrackMe - Data in motion tracking system
Monitored entities count by priority
Monitored entities count by state and priority
ENABLED ENTITIES
ENTITIES NOT MONITORED


Fields Quality Monitoring - Create a new collect job for raw events
Dictionary KVstore: TrackMe stores the dictionaries in the tenant's KVstore collection.
Allow unknown: If you choose to allow unknown by default, the collect job will allow unknown fields to be parsed.
Allow missing/null: If you choose to allow missing/null by default, the collect job will allow missing/null fields to be parsed.
Regex expression: Define a regular expression to validate the field content, to match anything, you would use: .*
Dictionaries and trackers usage: You can list all dictionaries and trackers using them with a TrackMe REST command.
Select a dictionary to manage:
cribl:business
Import / Create dictionaries:
Import from file
Create new
Add a new field:
orderNumber
Allow unknown?
false
Allow missing or empty?
false
Regex expression:
.*
Add this new field
Update fields and hit the save button to apply changes, or select fields for deletion. Click on the icon to test the regex in a splunk search.
Define the list of fields to be monitored, and define a validation regex if needed
Save fields changes
Cancel fields changes
Remove selected fields
Test Regex
Field
Allow unknown
Allow empty or missing
Regex expression
accountNumber
false
false
.*
conversationId
false
false
^ESB.*
JMSMessageID
false
false
^ID:.*
orderNumber
false
false
.*
Cancel
Back

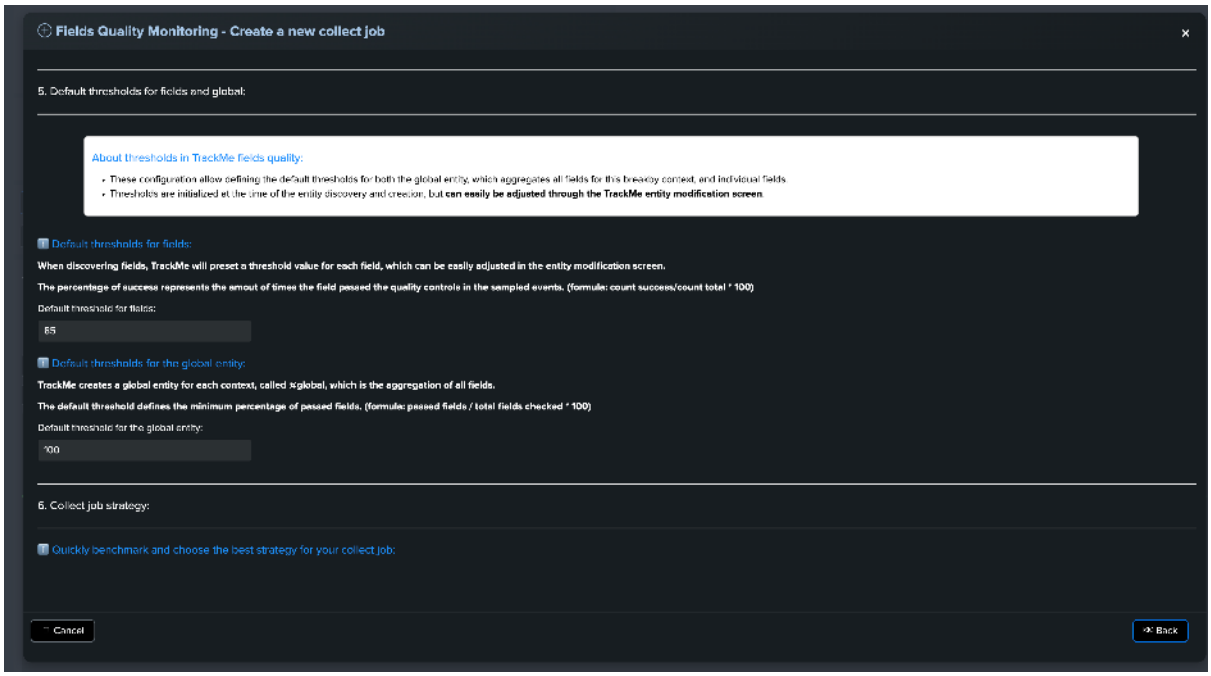




Default thresholds for fields and global

About thresholds in TrackMe fields quality:

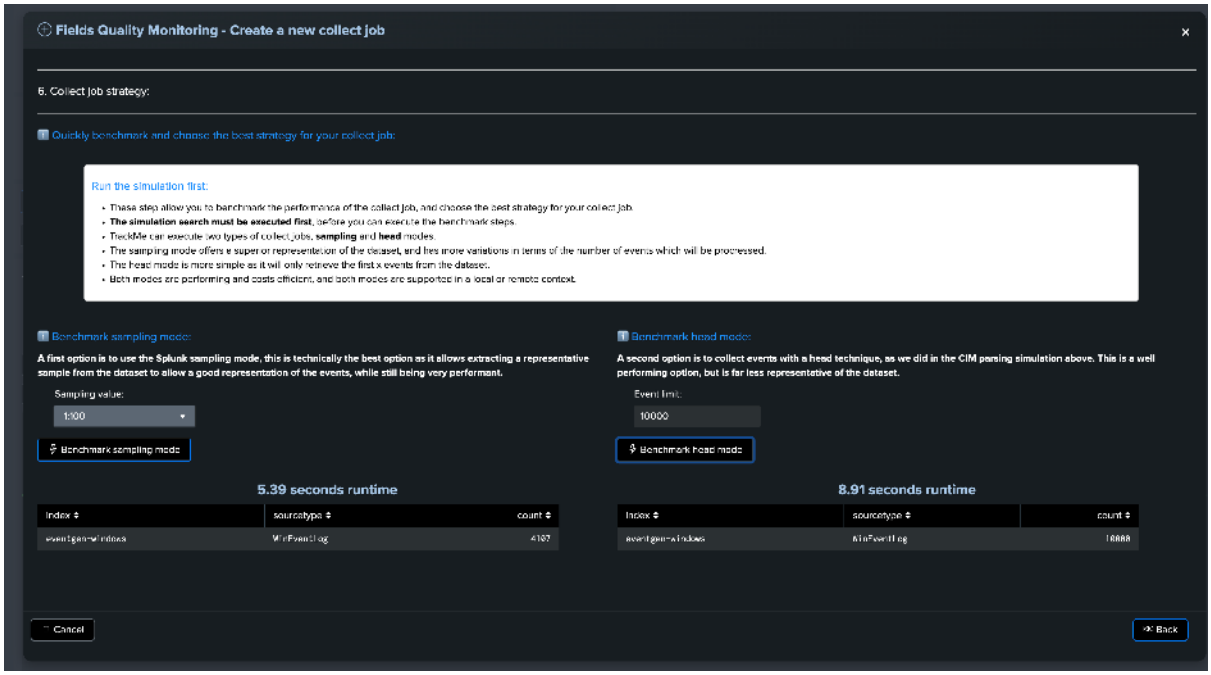
- These configurations allow you to define the default thresholds for both the global entity, which aggregates all fields for this breakby context, and individual fields.
- Thresholds are initialized at the time of entity discovery and creation, but can easily be adjusted through the TrackMe entity modification screen.



Benchmark the collect job strategy

About the collect job strategy benchmark:

- These steps allow you to benchmark the performance of the collect job and choose the best strategy for your collect job.
- The simulation search must be executed first, before you can execute the benchmark steps.
- TrackMe can execute two types of collect jobs: sampling and head modes.
- The sampling mode offers a superior representation of the dataset and has more variations in terms of the number of events that will be processed.
- The head mode is simpler as it will only retrieve the first x events from the dataset.
- Both modes are performant and cost-efficient, and both modes are supported in a local or remote context.



Collect job main settings

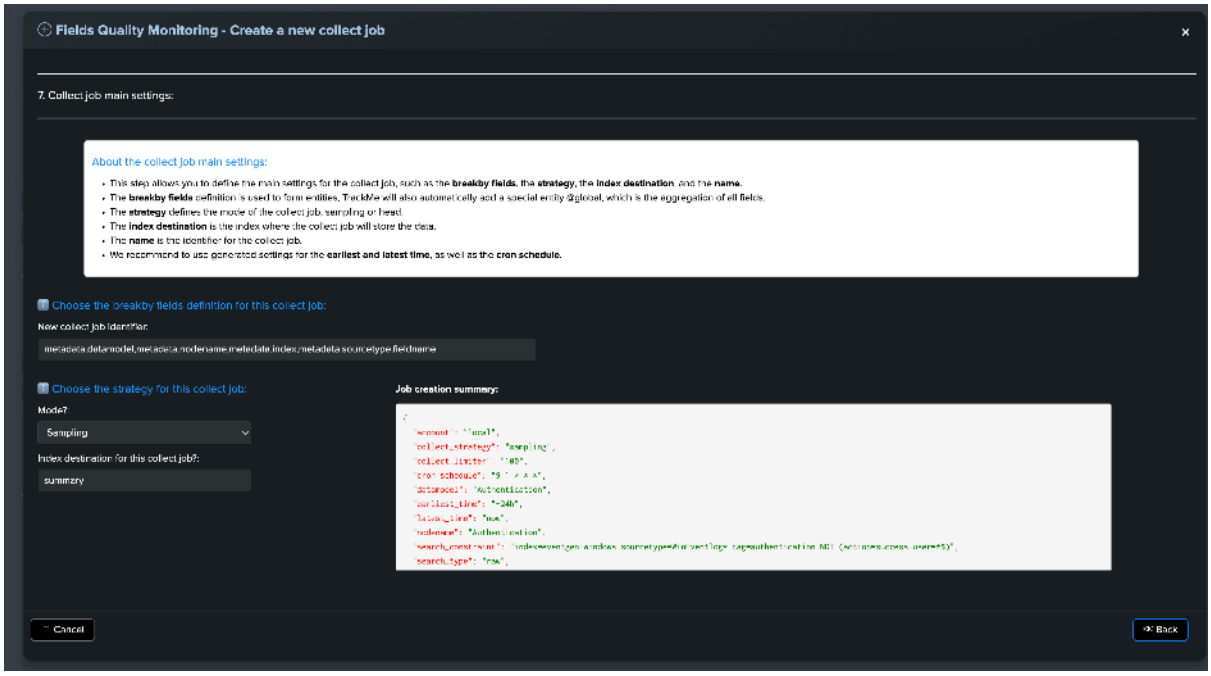
About the collect job main settings:

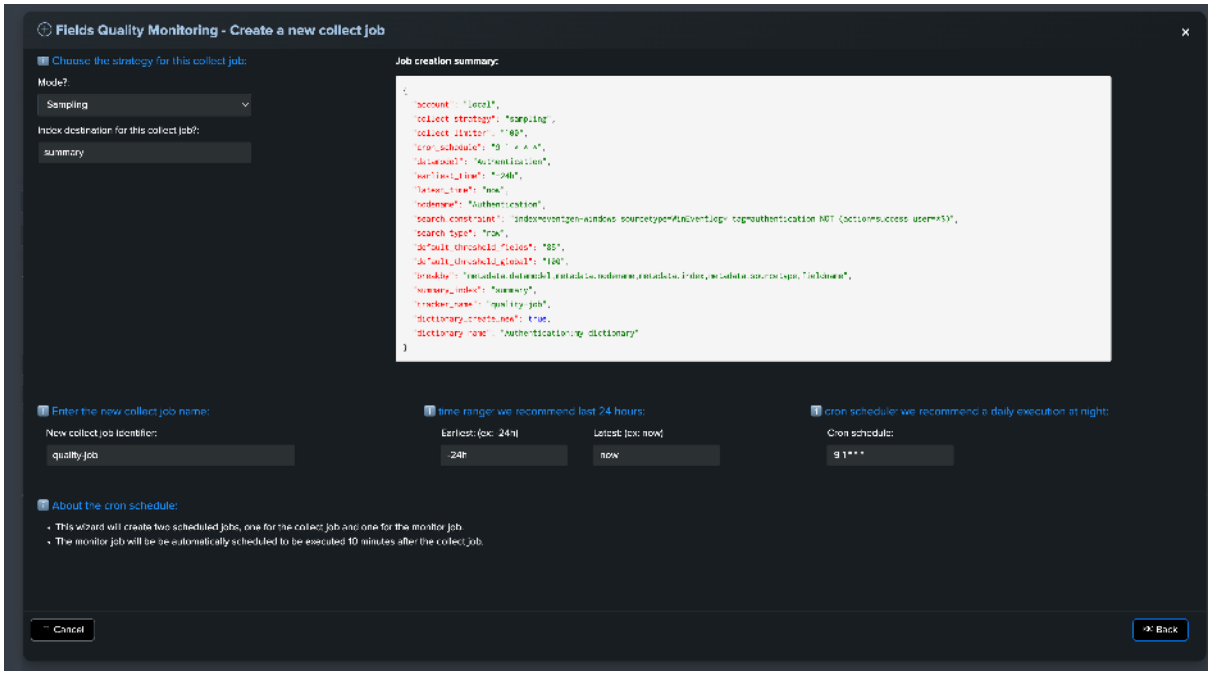
- This step allows you to define the main settings for the collect job, such as the breakby fields, strategy, index destination, and name.
- The breakby fields definition is used to form entities. TrackMe will also automatically add a special entity @global, which is the aggregation of all fields.
- The strategy defines the mode of the collect job: sampling or head.
- The index destination is the index where the collect job will store the data.
- The name is the identifier for the collect job.
- We recommend using generated settings for the earliest and latest time, as well as the cron schedule.

Hint

About the cron schedule:

- This wizard will create two scheduled jobs, one for the collect job and one for the monitor job.
- The monitor job will be automatically scheduled to be executed 10 minutes after the collect job.

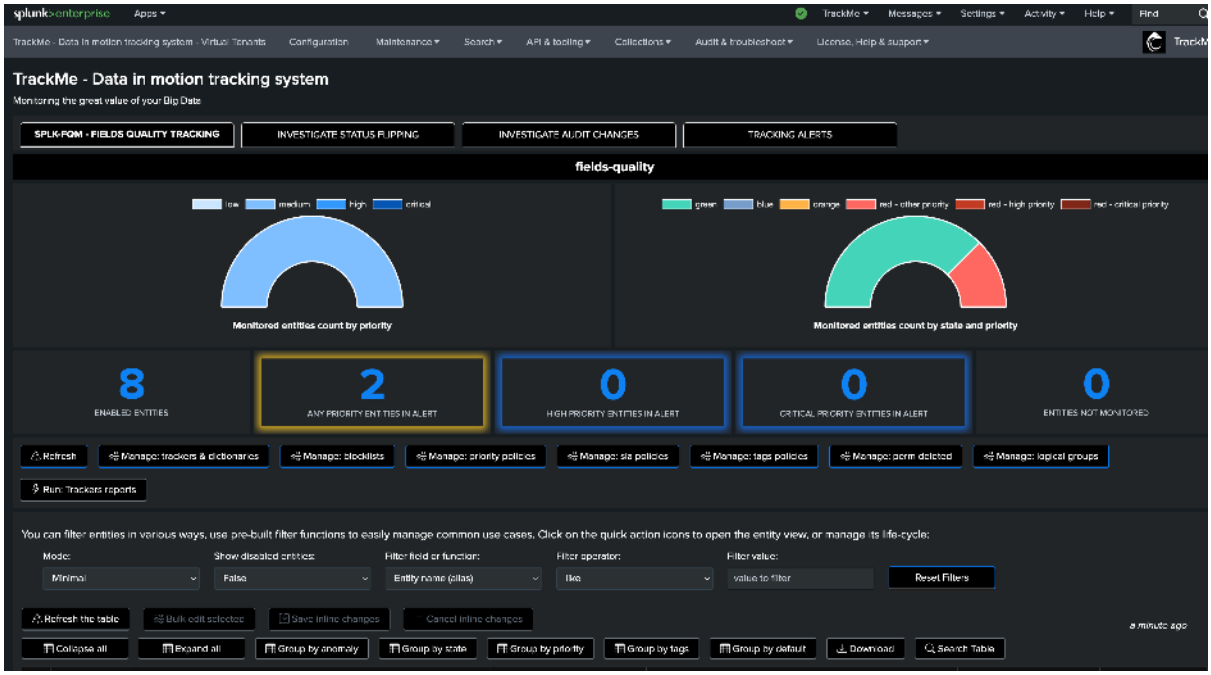


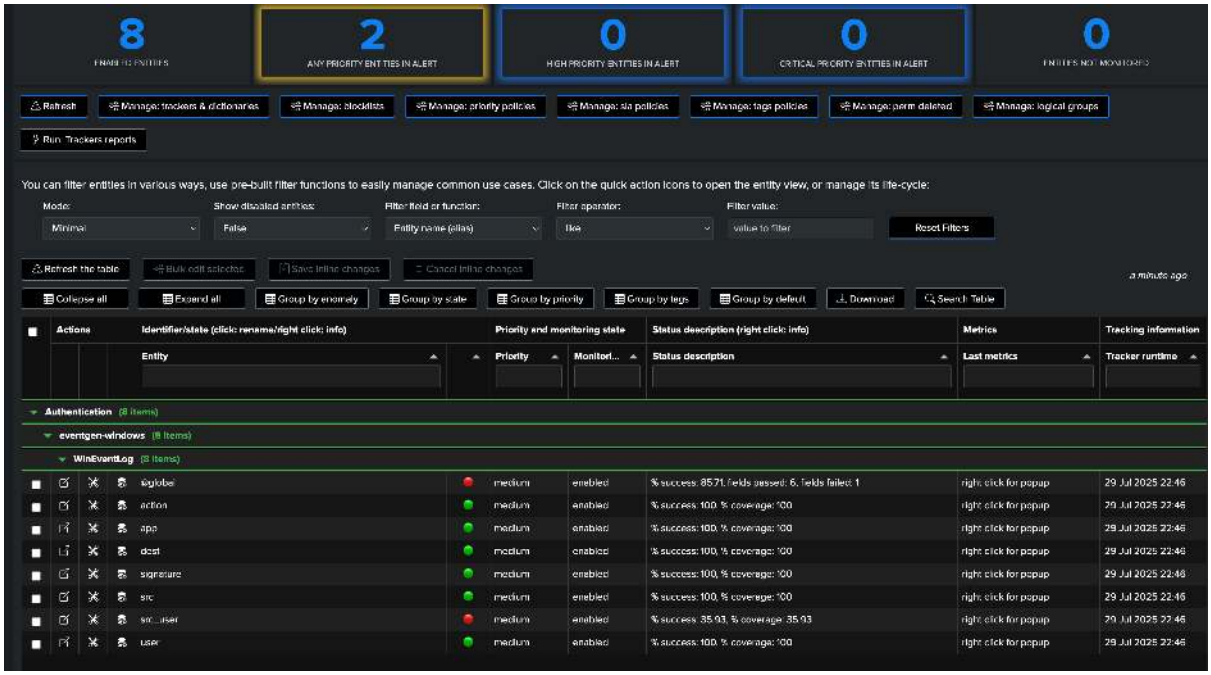


Validate the collect job creation

Finally! Hit the magic button and let TrackMe do the job.

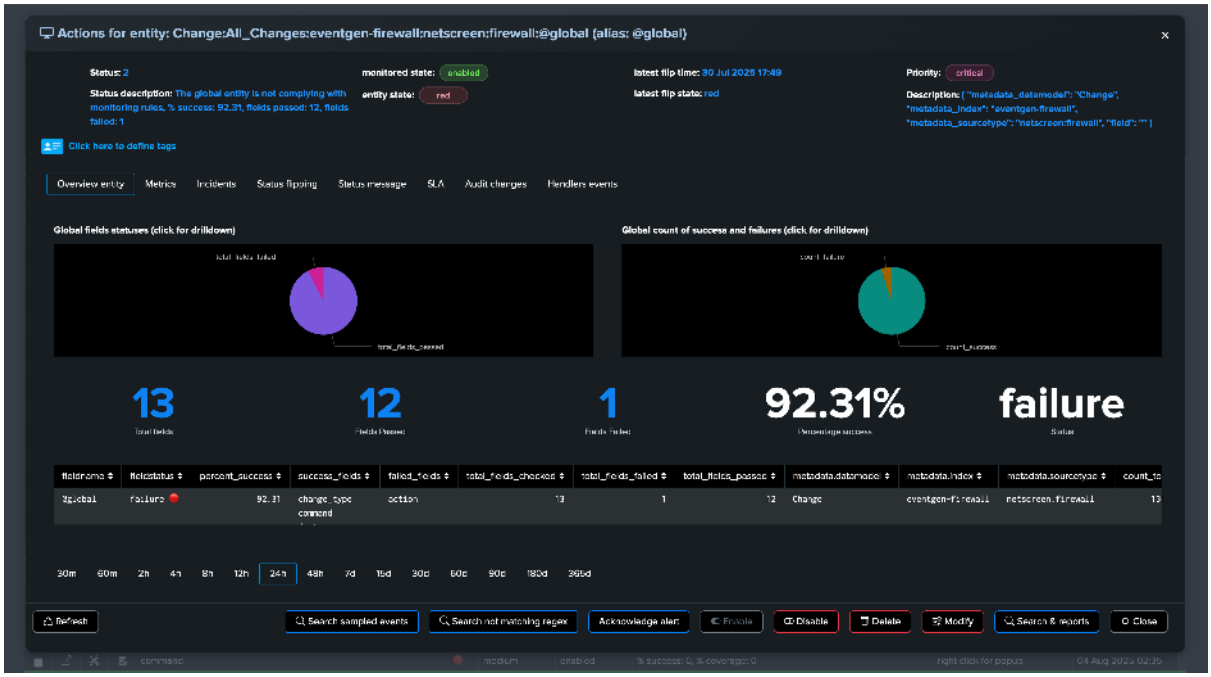
Once created, you can immediately execute the `collect` phase of the job, **then** the `monitor` phase to start seeing entities in TrackMe:

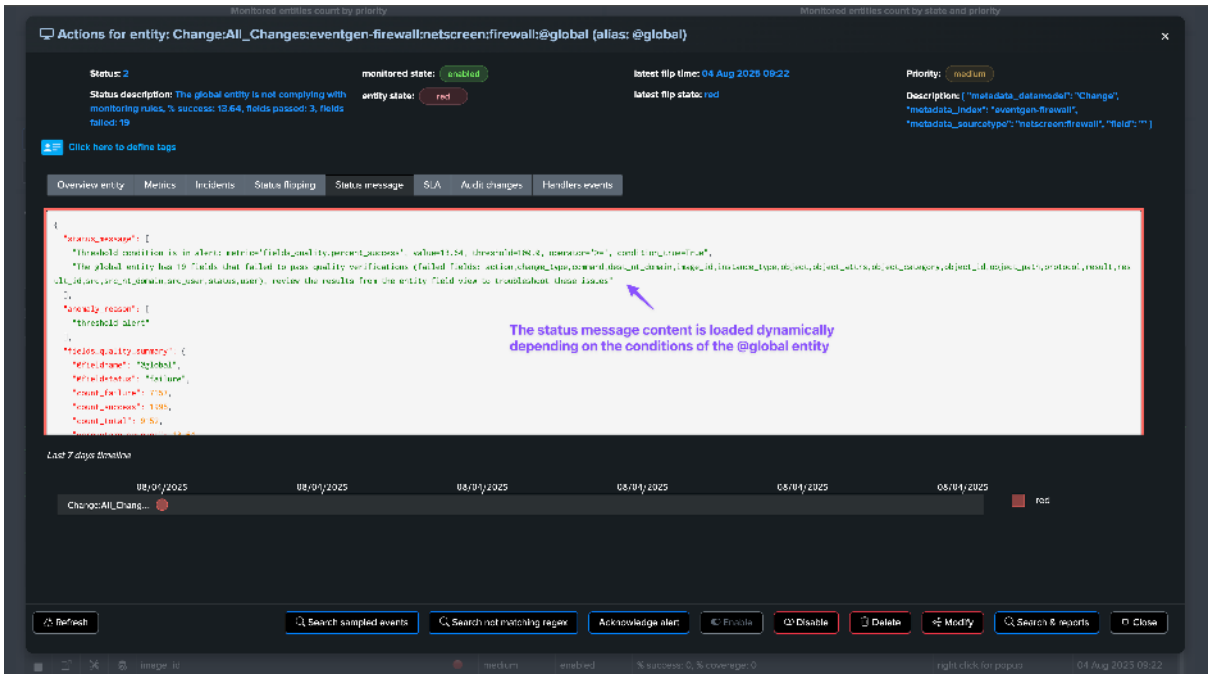


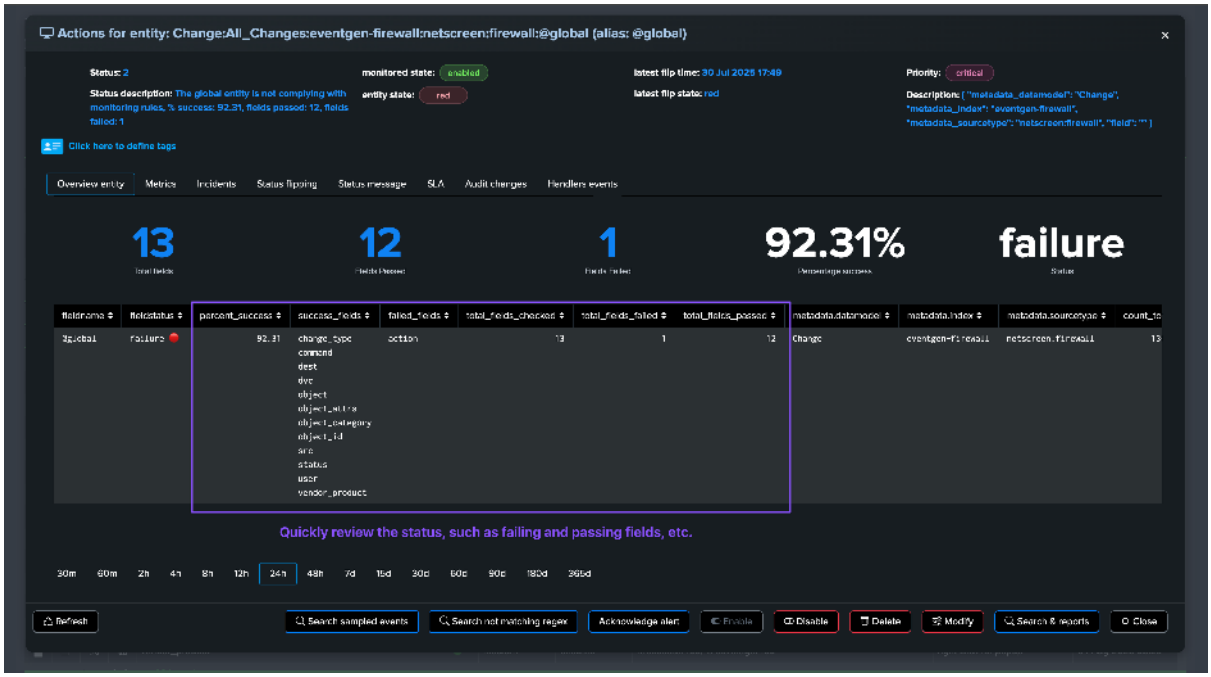


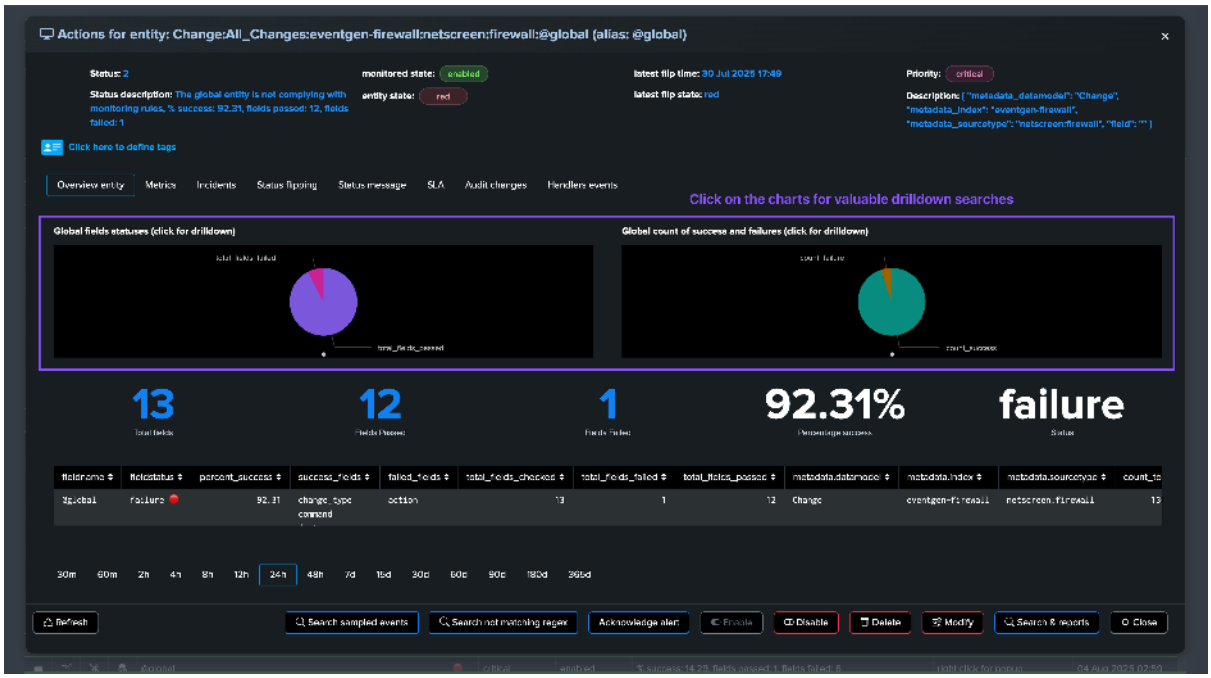
Fields Quality: @global entity

The @global entity is the aggregation of all fields for this breakby context. The UI provides full context and visualization of the entity:


Actions for entity: Change:All_Changes:eventgen-firewall:netscreen:firewall:@global (alias: @global)
Status: 2
monitored state: enabled
latest flip time: 30 Jul 2025 17:49
Priority: critical
Status description: The global entity is not complying with monitoring rules, % success: 92.31, fields passed: 12, fields failed: 1
entity state: red
latest flip state: red
Click here to define tags
Overview entity
Metrics
Incidents
Status flipping
Status message
SLA
Audit changes
Handlers events
Global fields statuses (click for drilldown)
Global count of success and failures (click for drilldown)
13
12
1
92.31%
failure
Refresh
Search sampled events
Search not matching regex
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close

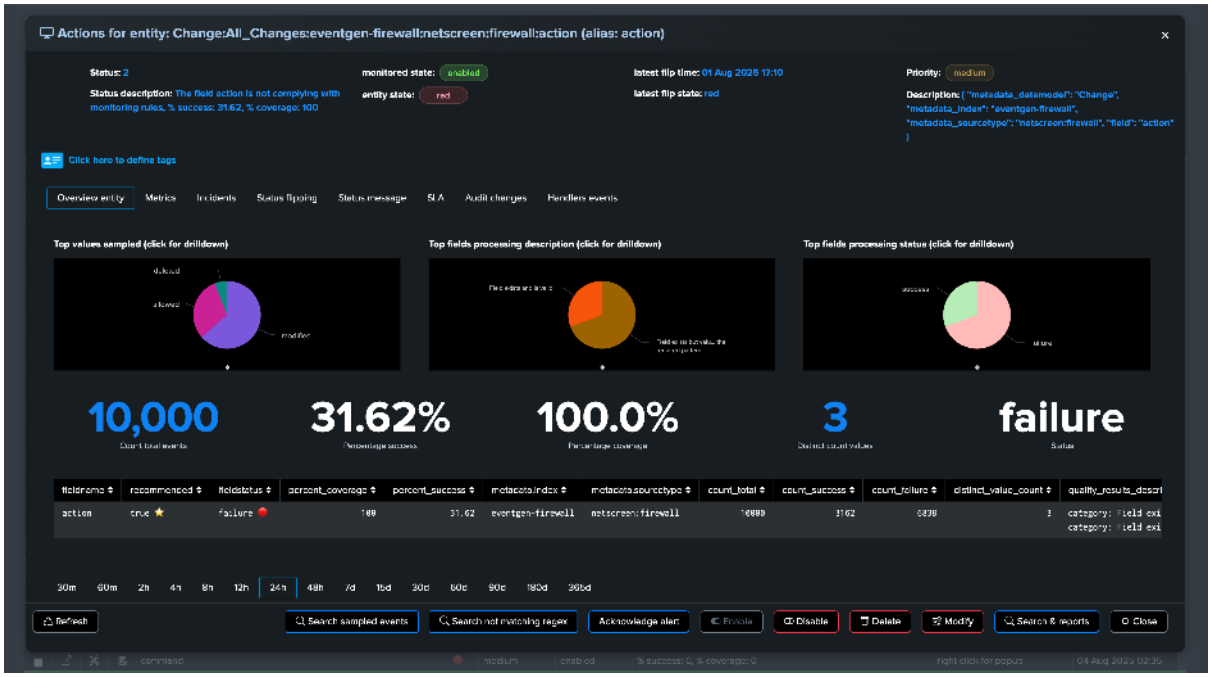

Actions for entity: Change:All_Changes:eventgen-firewall:netscreen:firewall:@global (alias: @global)
Status: 2
monitored state: enabled
latest flip time: 04 Aug 2025 09:22
Priority: medium
Status description: The global entity is not complying with monitoring rules, % success: 13.64, fields passed: 3, fields failed: 19
entity state: red
latest flip state: red
Click here to define tags
Overview entity
Metrics
Incidents
Status flipping
Status message
SLA
Audit changes
Handlers events
The status message content is loaded dynamically depending on the conditions of the @global entity
Refresh
Search sampled events
Search not matching regex
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close

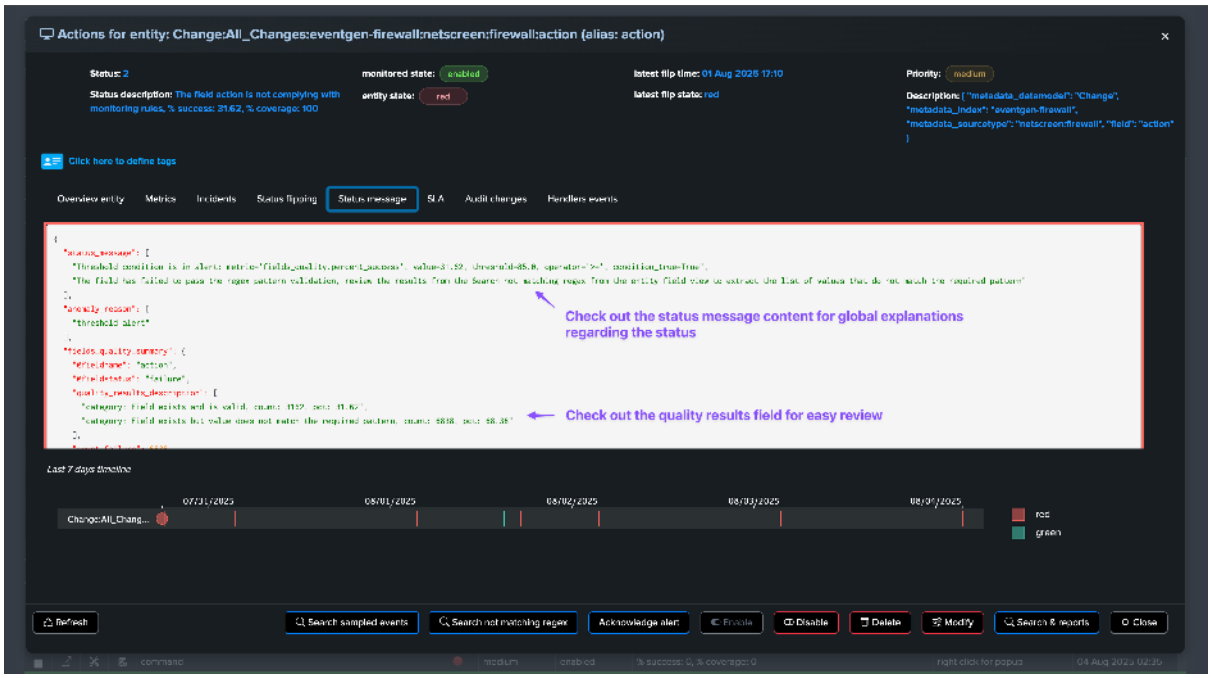


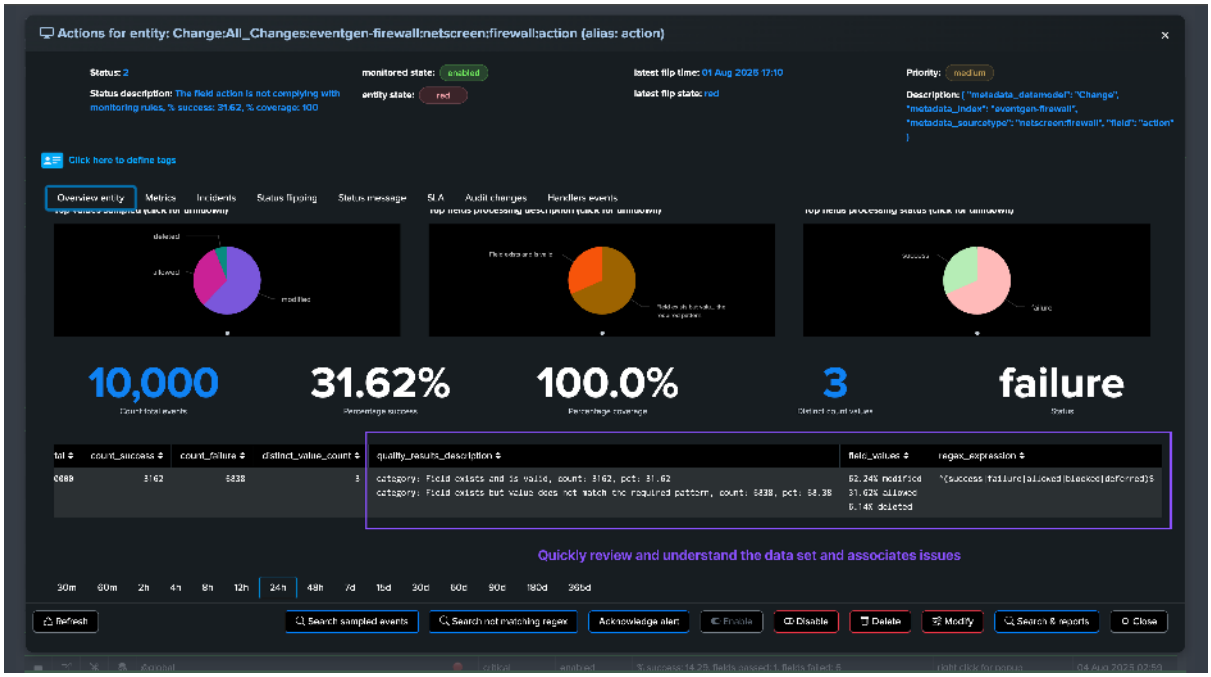


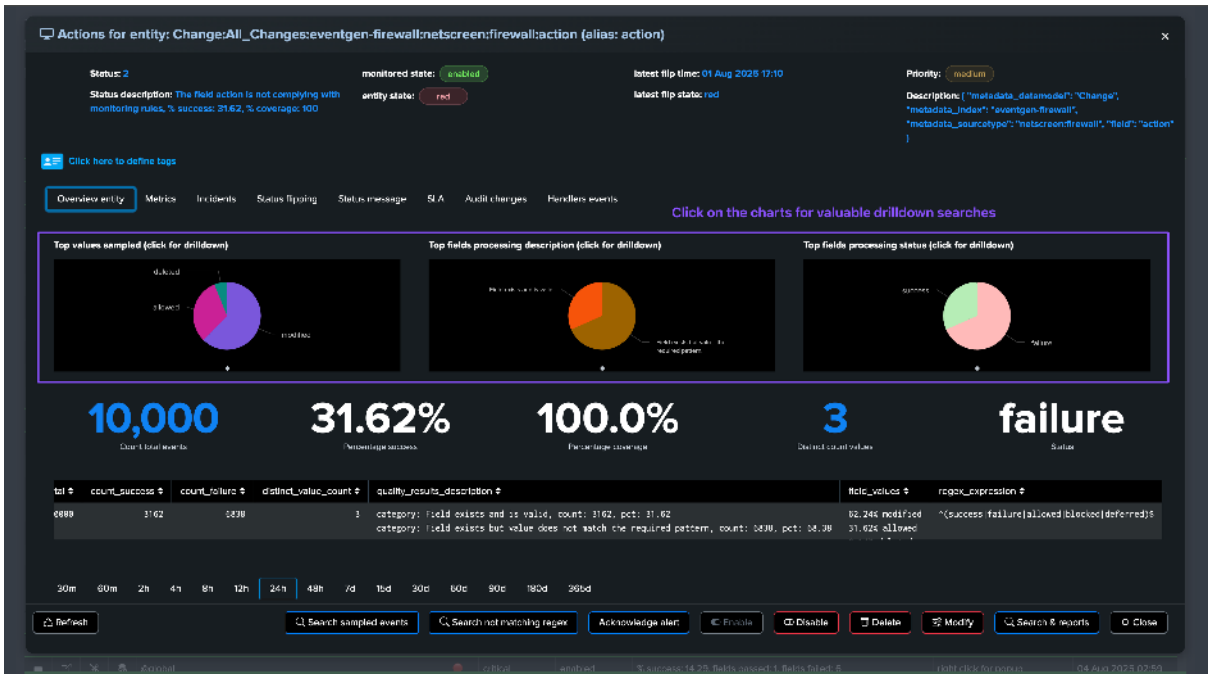
Fields Quality: field view

The field view provides a detailed view of the fields for the entity


Actions for entity: Change:All_Changes:eventgen-firewall:netscreen:firewall:action (alias: action)
Overview entity
Metrics
Incidents
Status flipping
Status message
SLA
Audit changes
Handlers events
Top values sampled (click for drilldown)
Top fields processing description (click for drilldown)
Top fields processing status (click for drilldown)
10,000
31.62%
100.0%
3
failure
Refresh
Search sampled events
Search not matching regex
Acknowledge alert
Enable
Disable
Delete
Modify
Search & reports
Close


Actions for entity: Change:All_Changes:eventgen-firewall:netscreen:firewall:action (alias: action)
Overview entity
Metrics
Incidents
Status flipping
Status message
SLA
Audit changes
Handlers events
Check out the status message content for global explanations regarding the status
Check out the quality results field for easy review
Refresh
Search sampled events
Search not matching regex
Acknowledge alert
Enable
Disable
Delete
Modify
Search & reports
Close





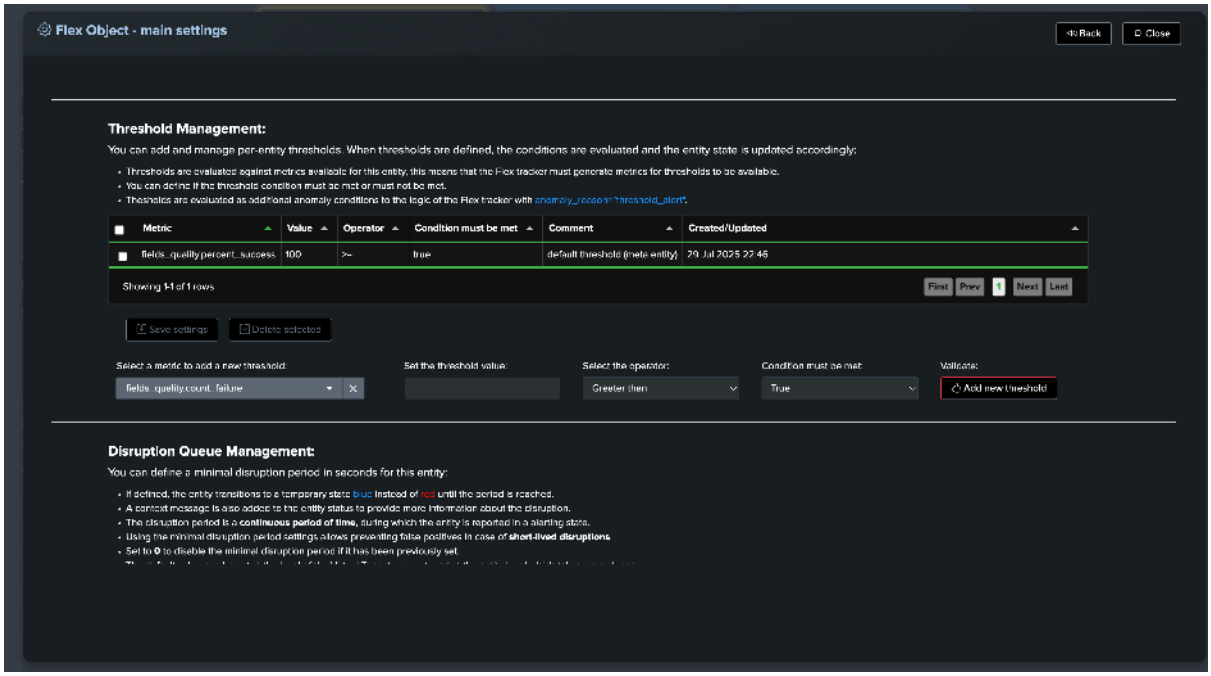
Updating the @global entity and fields entities thresholds

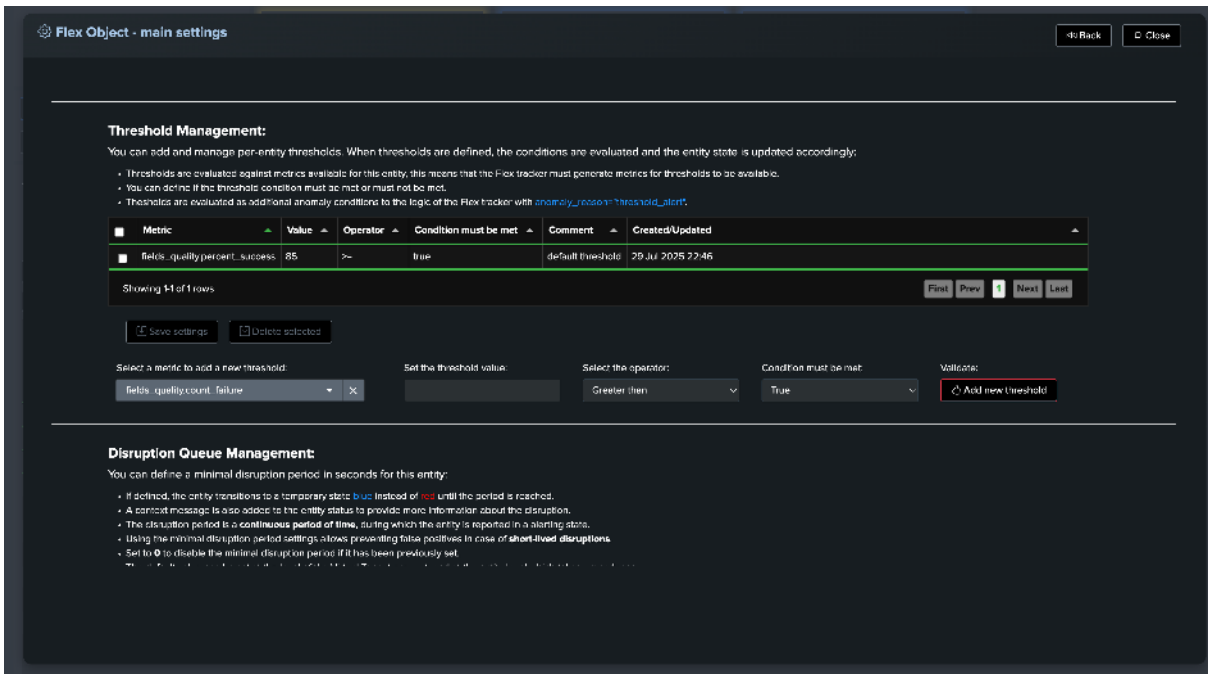
The @global entity and fields entities thresholds can be updated through the entity modification screen:

Hint

About updating fields entities thresholds:

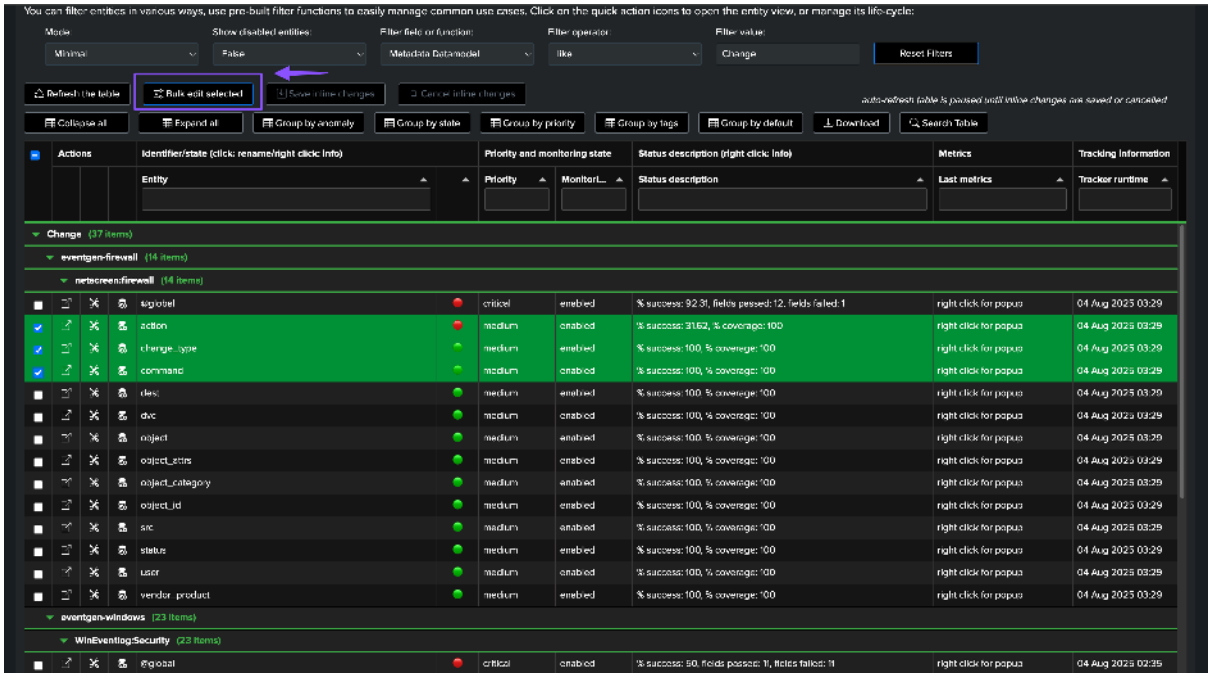
- Updating a threshold for a field is immediately reflected in the field entity view.
- However, for the @global entity to reflect the new field status, the `monitor` phase of the collect job must be executed.

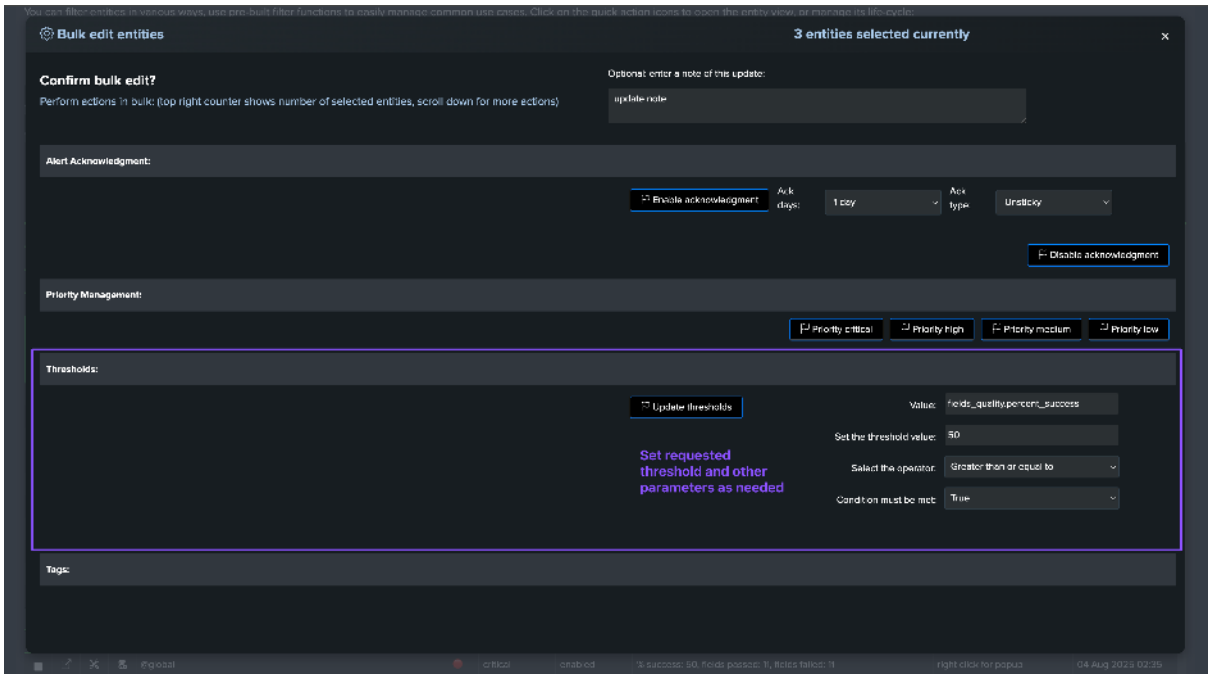




Updating thresholds in bulk after the entities discovery

You can update the thresholds in bulk after the entities discovery, by selecting the entities to be updated and then performing a bulk update through the bulk edit screen:





Disabling fields entities

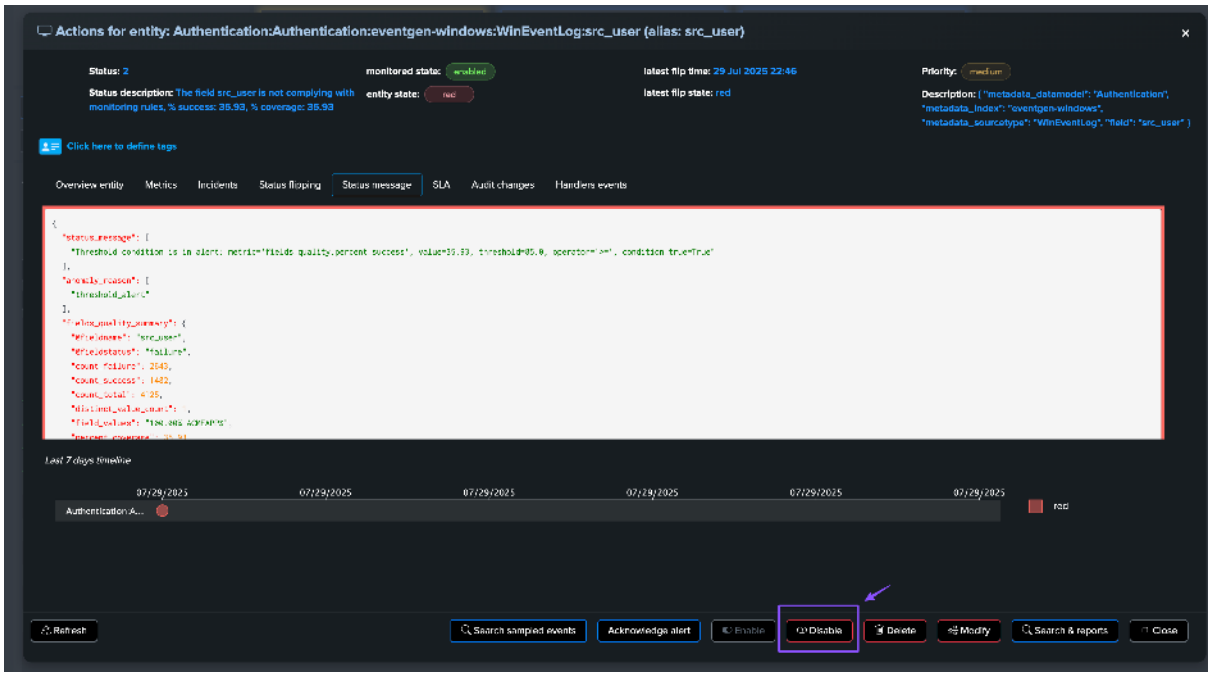
You can use the normal TrackMe workflow to disable a field entity that you wouldn't want to consider anymore:

Hint

About disabling a field entity and how it is reflected in the @global entity view:

- Disabling a field entity is immediately taken into account for the entity itself.
- This means that if, for instance, you have created an alert that covers field entities, a disabled field entity will never be considered.
- As well, the UI will not show by default entities that are disabled.

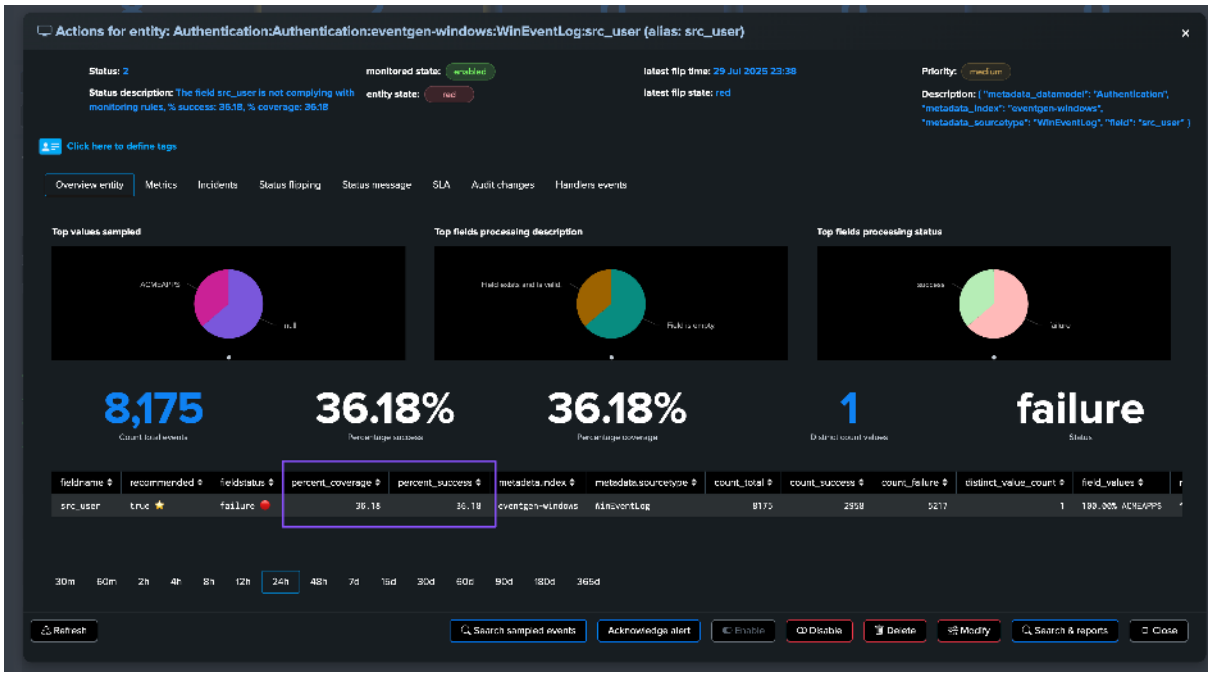
- **However**, the @global entity will still aggregate the disabled fields until the `monitor` phase of the collect job is executed.

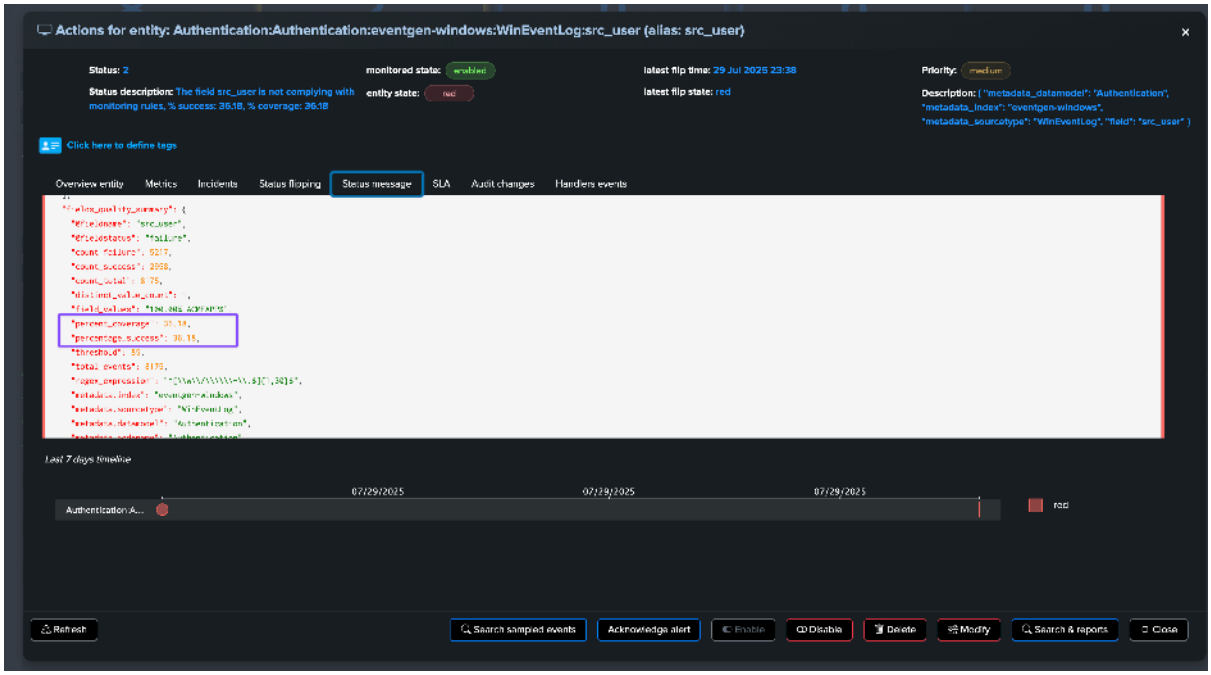


About fields percentage of success and percentage of coverage

For fields entities, TrackMe reports two key metrics:

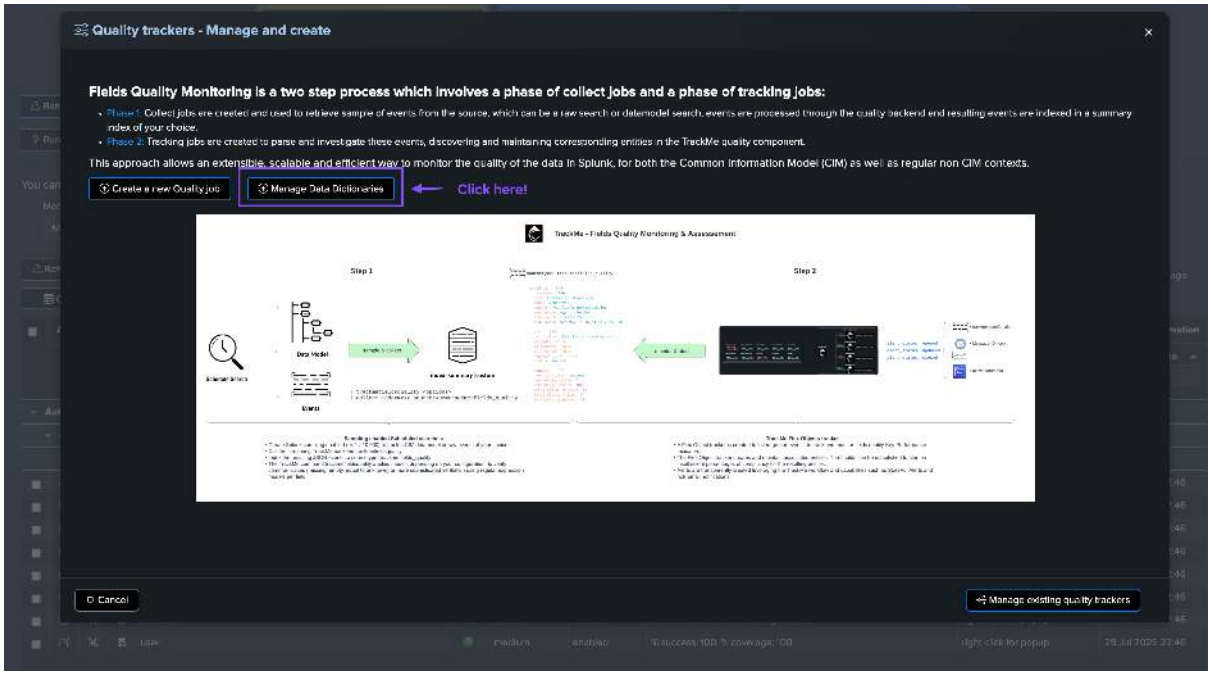
- The percentage of success, which is the percentage of times a field has passed the quality checks according to the dictionary definition: is the field empty, missing or null, unknown, or has the regular expression matched.
- The percentage of coverage, which is simply the percentage of times the field was part of the dataset and not empty or null.

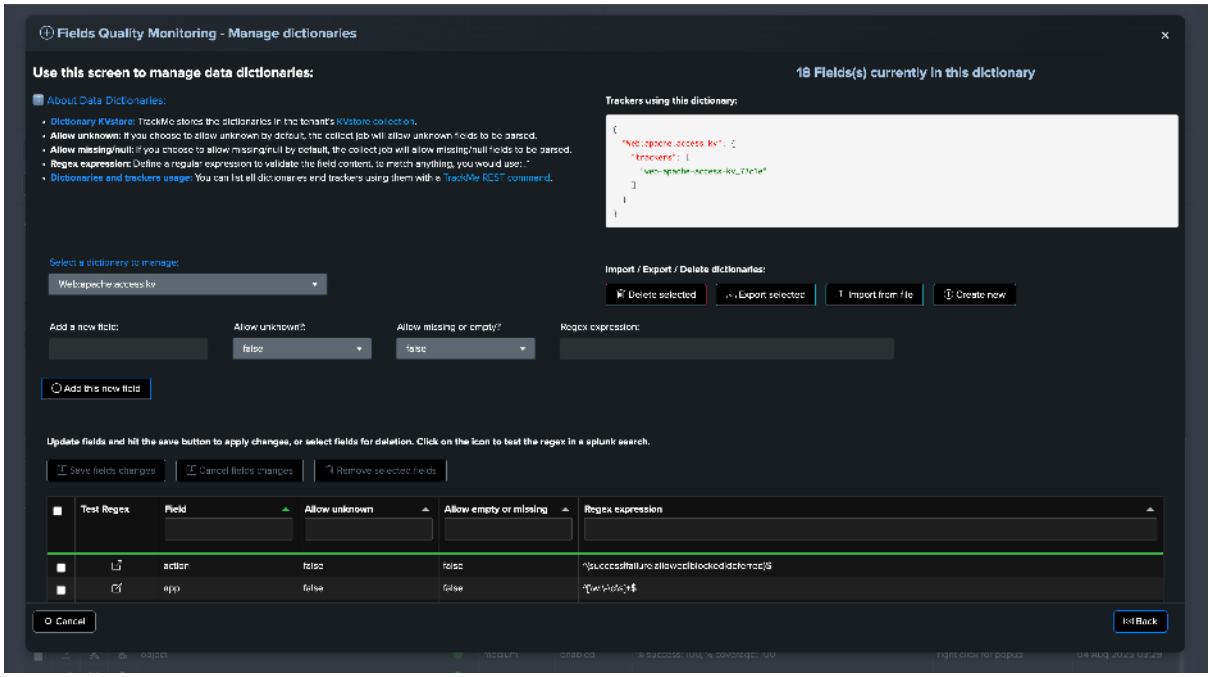




Managing Data Dictionaries

You can manage the data dictionaries through the dictionaries management interface:





About the data dictionary management:

- Dictionary KVstore: TrackMe stores the dictionaries in the tenant's KVstore collection.
- Allow unknown: If you choose to allow unknown by default, the collect job will allow unknown fields to be parsed.
- Allow missing/null: If you choose to allow missing/null by default, the collect job will allow missing/null fields to be parsed.
- Regex expression: Define a regular expression to validate the field content. To match anything, you would use: .*
- Dictionaries and trackers usage: You can list all dictionaries and trackers using them with a TrackMe REST command.

Hint

Actions that can be performed on a data dictionary:

- Delete selected dictionary
- Export selected dictionary
- Import selected dictionary
- Adding a new field and specifying allow unknown, allow missing/null, regex expression
- Updating existing fields definition and their parameters
- Testing a regular expression against the field dataset

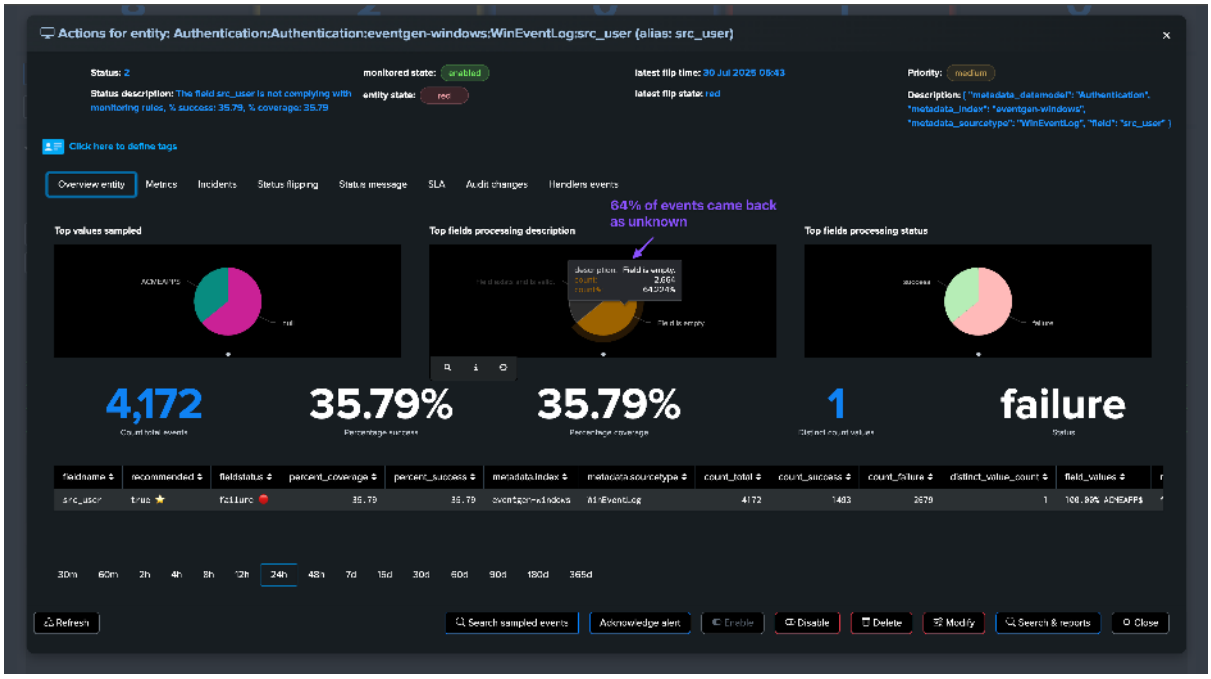
Modifying fields entities compliance rules in the data dictionary

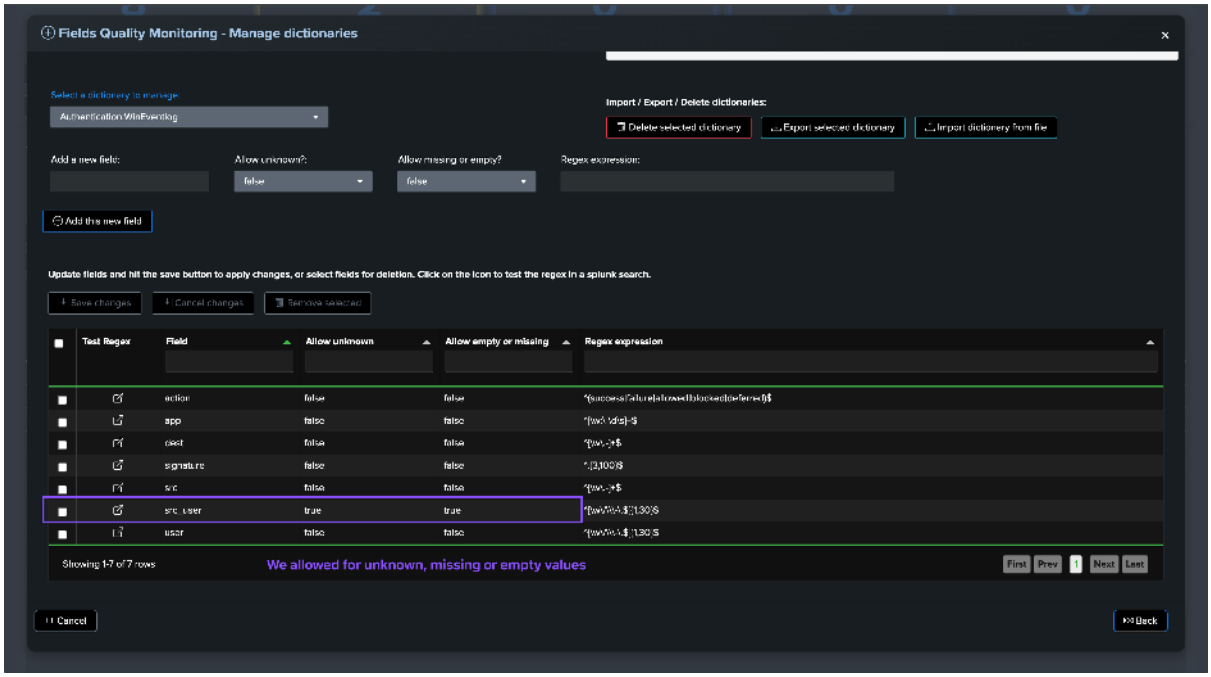
When modifying rules for compliance in the data dictionary, such as:

- **Allowing the field to be unknown:** `allow_unknown: True` or `False`
- **Allowing the field to be missing/null:** `allow_missing_null: True` or `False`

- **Regular expression:** `regex`: a regular expression to validate the field content

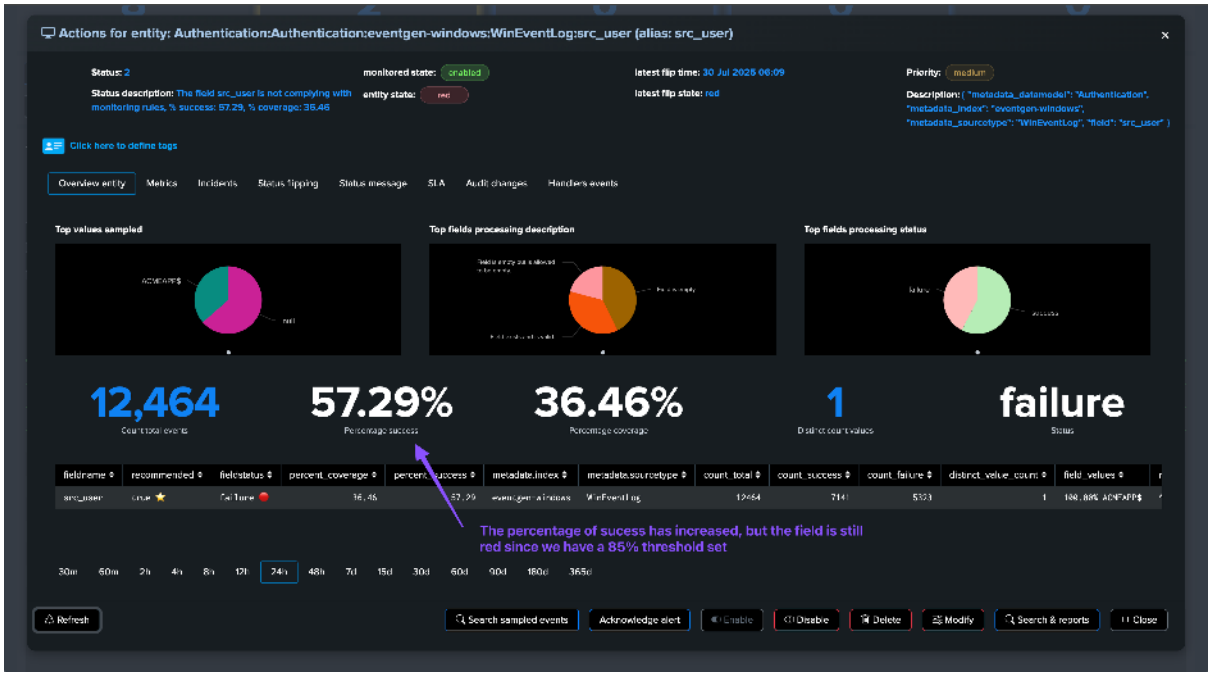
You have to keep in mind that the percentage of success will be updated according to the events processed and time. If, for instance, we have a field that is initially failing with a condition due to unknown values, and we just modified the data dictionary to allow unknown values:

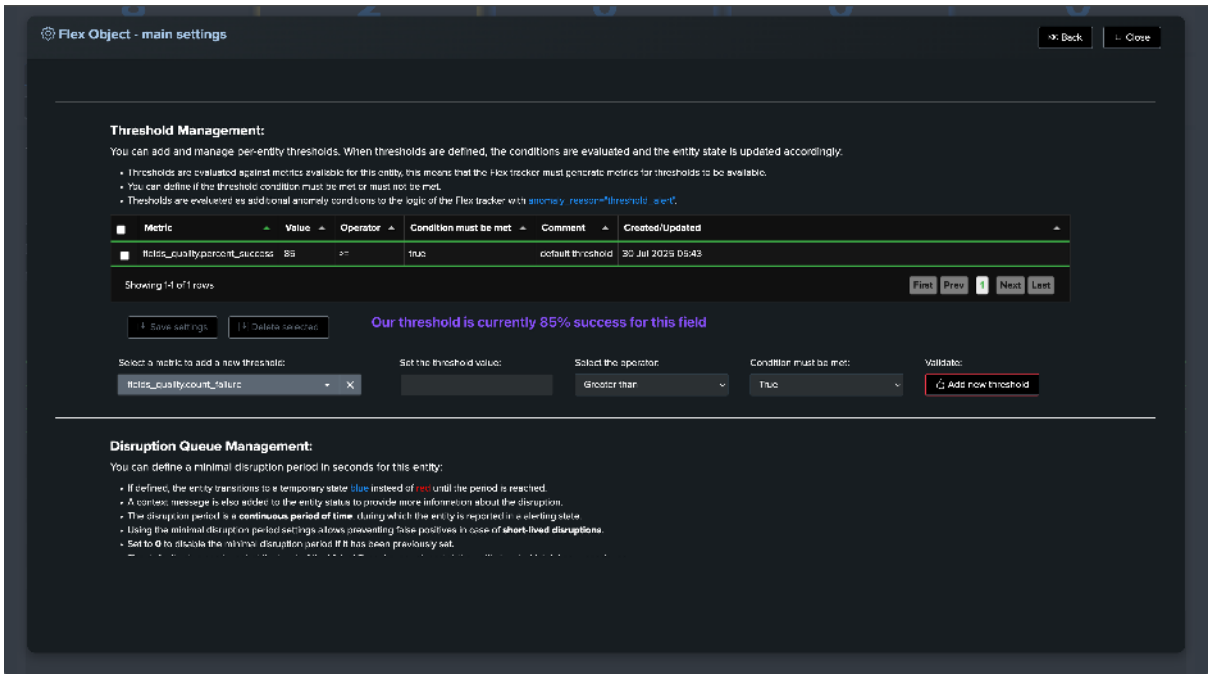




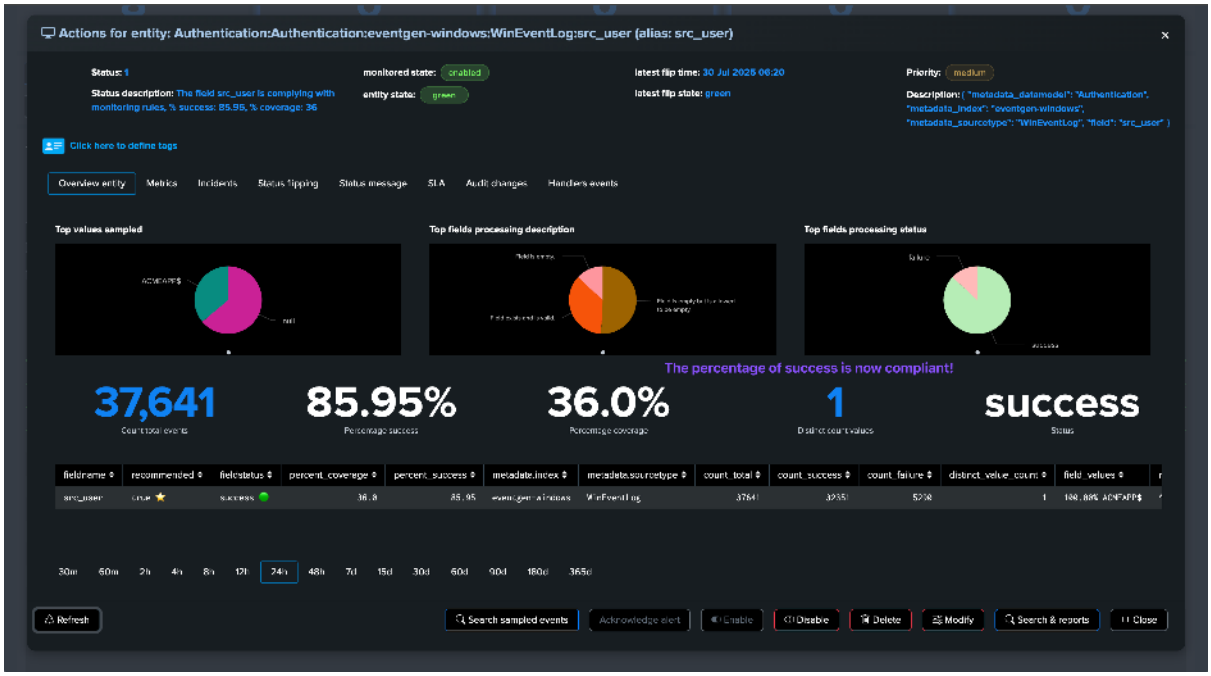
The percentage of success will not grow until we have a new execution of the `collect` and `monitor` phases of the collect job.

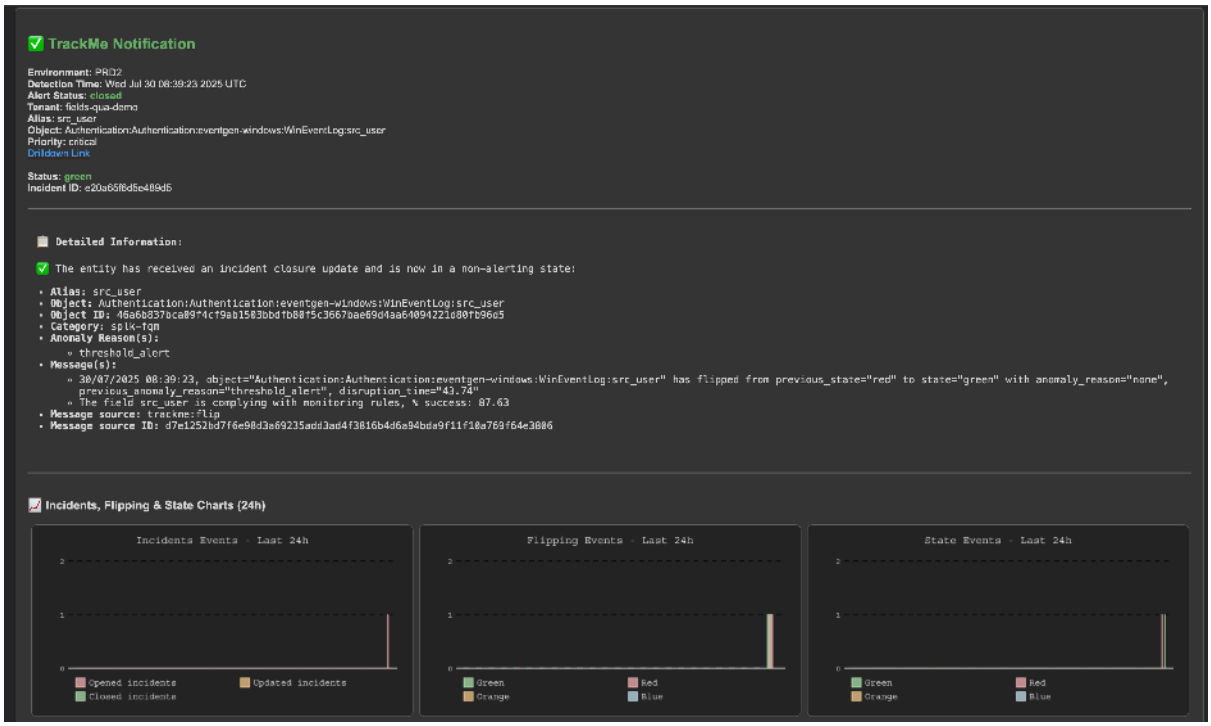
Therefore, depending on the initial percentage of failures and conditions, it may take several iterations before we reach a stable percentage of success and the field is considered as passing the compliance checks:





Once conditions are met, or if we lower the requested percentage of success for that field, the field will transition to green and is now compliant according to our requirements:





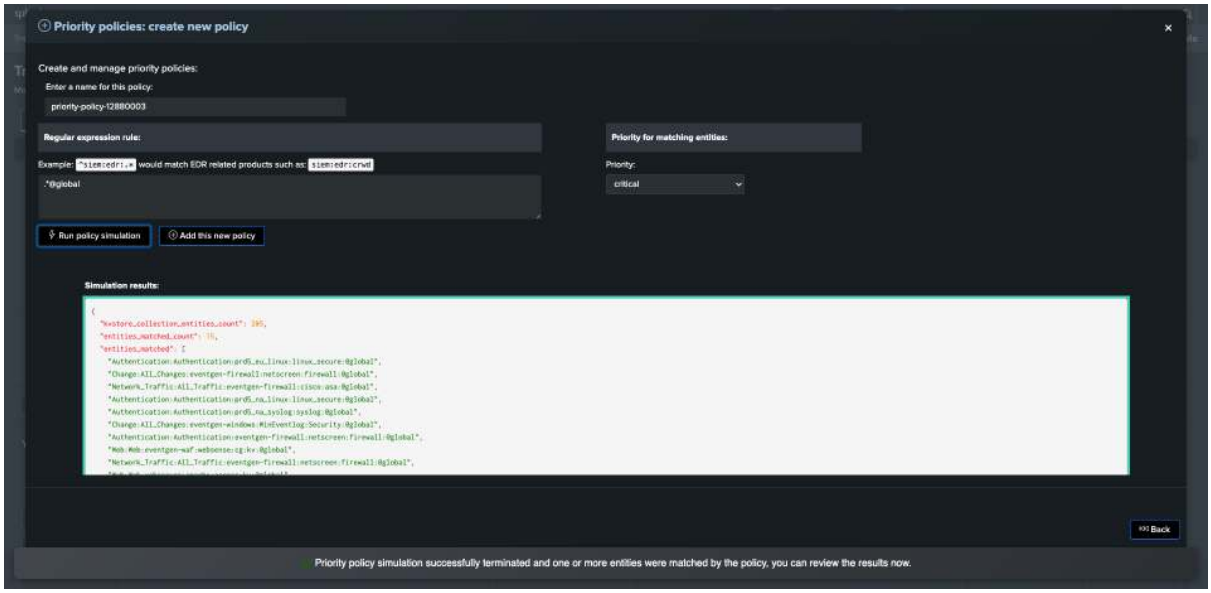
Creating a priority policy for global entities

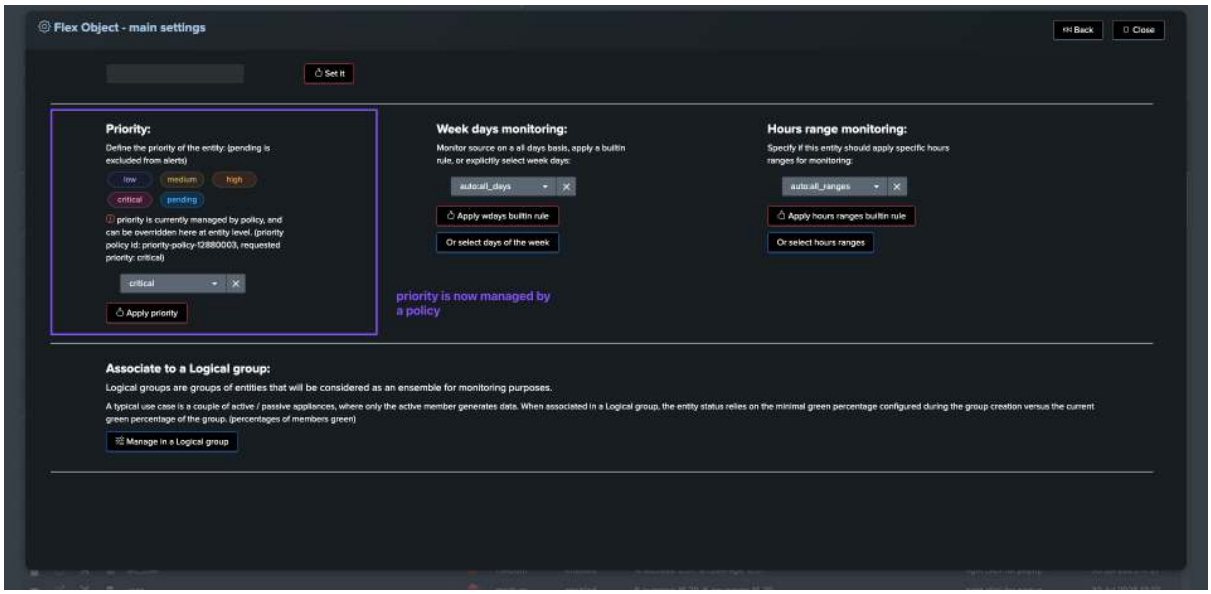
TrackMe provides a feature called priority policies, which allows to automatically handle the priority level for entities matching a regular expression.

We can easily leverage this feature to automatically set the priority for our `@global` entity, so that if wanted, we can enable email notifications for all global entities only:

Expression used in the following example:

```
.*@global
```





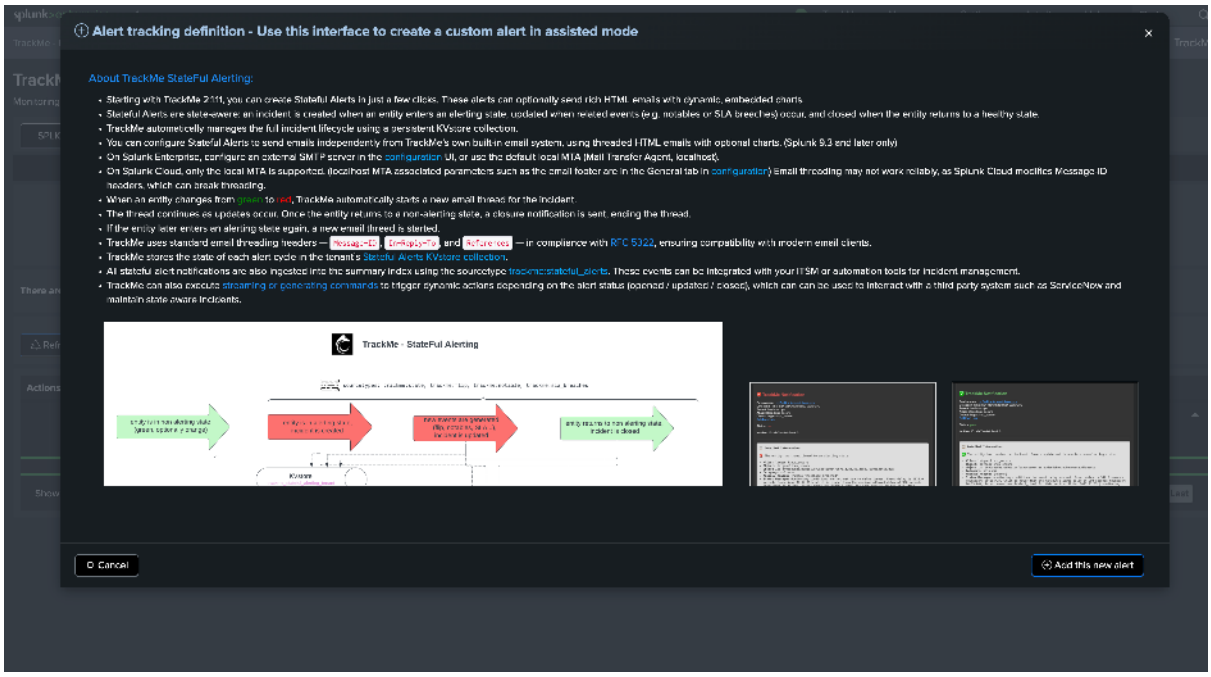
Alerting on fields quality issues, and generating emails or other types of notifications

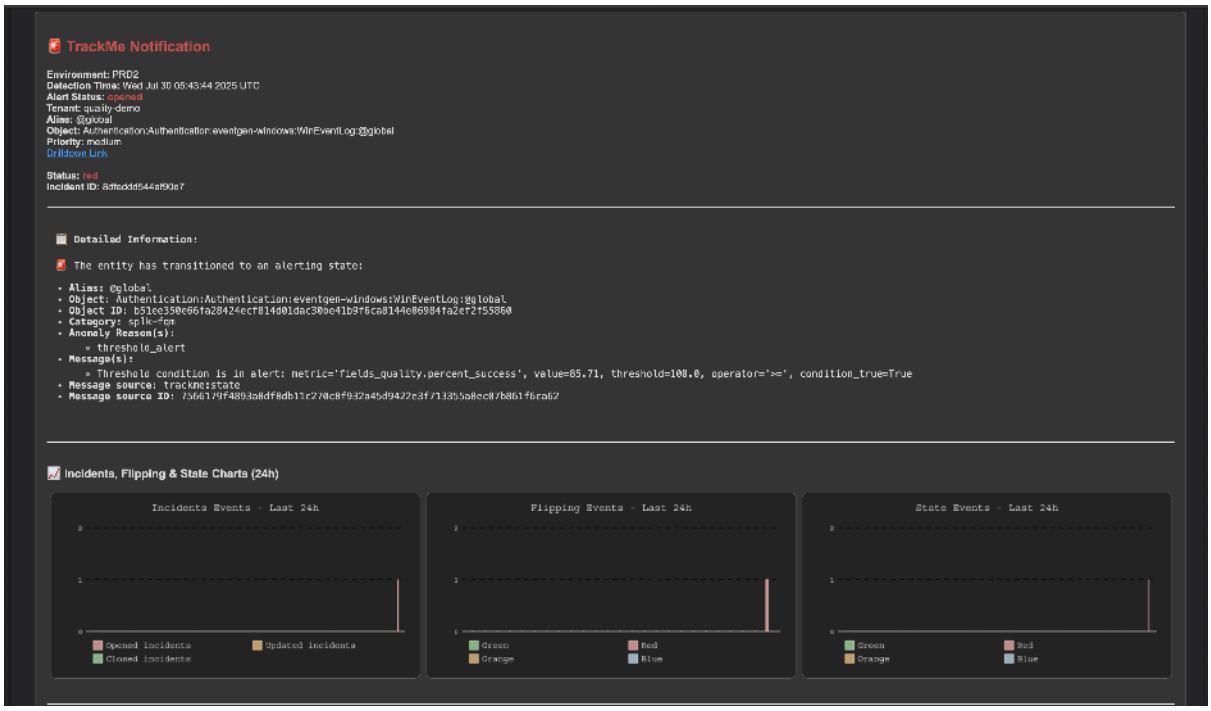
Alerting on fields quality issues works exactly in TrackMe as for any other component, using our extensive and flexible alerting concepts, such as the Stateful alerts in TrackMe.

Hint

About alerting only against the @global entity, and/or fields entities:

- This choice is entirely up to you. You can choose to do both, any kind of combination is possible.
- A recommended approach would be to rely on the priority level. You could simply set the @global entity priority to `high` or `critical` and set up your alert to trigger on these levels.
- If you need at some point to alert on certain fields, simply increase the priority level of the field entities you want to alert on.





Annex: usage and options for the command trackmefieldsquality

The command `trackmefieldsquality` is used in the collect phase to parse and validate the fields compliancy, this is a powerful and flexible Python backend that provides various options.

The command accepts the following parameters:

argument	description	default	example or valid values
`fields_to_check_list`	The list of fields to verify, provided as a comma-separated list	None	"action,app,bytes,url"
`fields_to_check_fiel`	Name of the field containing the list of fields to check (comma-separated)	None	"fields_list"
`fields_to_check_dict`	JSON string containing a dictionary of fields to check with optional regex patterns	None	'{"field1": {"name": "field1", "regex": "^[A-Z]+$"}, "field2": {"name": "field2"}}'
`fields_to_check_dict`	Path to a JSON file containing a dictionary of fields to check with optional regex patterns	None	"/opt/splunk/etc/apps/myapp/mydir/web_data
`fields_to_check_dict`	Name of the field containing a JSON string with a dictionary of fields to check	None	fields_dict
`fields_to_check_sear`	The search command to use to generate the dictionary of fields to check	None	"| trackmefieldsquality-gendict datamodel=Web show_only_recommended_fields=True"
`include_field_values`	Boolean option to include field values in the JSON summary	False	True/False
`pretty_print_json`	Boolean option to pretty print the JSON summary	True	True/False
`output_mode`	The mode to output the results (json or raw)	json	json
`metadata_fields`	CSV list of metadata fields to include in the metadata section of the JSON	index,sourcetype,host,source	"datamodel"
`summary_fieldname`	Defines the name of the summary field	summary	"summary"
`metadata_fieldname`	Defines the name of the metadata field added to the summary JSON	metadata	"metadata"

The first 6 options are **mutually exclusive**, only one of these options can be used at a time:

- `fields_to_check_list`
- `fields_to_check_fieldname`
- `fields_to_check_dict`
- `fields_to_check_dict_path`
- `fields_to_check_dict_fieldname`
- `fields_to_check_search_command`

These options exist so that we can cover all use cases, this provides all levels of flexibility to use different Splunk techniques such as subsearches or simply storing the list of fields or models, dynamic generation in SPL, etc.

Let's take an example over each of these options:

Argument: fields_to_check_list

This is the most simple use case:

- Provide the list of fields to be checked as a comma-separated list
- For each field, we will check the following: - Missing - Empty - Null - Equal to unknown

If the field passes all these checks, it is declared as valid with a status of `success`, and `failure` otherwise.

The command will store with the JSON object a section for the field, which includes `flags` for each check, with a boolean value True or False:

- `is_missing`
- `is_empty`
- `is_unknown`
- `regex_failure`

In addition, the command will account the field and its status in the `summary` section of the JSON object:

- `overall_status`: the overall status of the field checks, either `success` or `failure`
- `total_fields_checked`: the total number of fields checked
- `total_fields_failed`: the total number of fields that failed
- `total_fields_passed`: the total number of fields that passed
- `percentage_failed`: the percentage of fields that failed
- `percentage_passed`: the percentage of fields that passed

Note: this argument does NOT process a regular expression to check for the content, therefore the `regex_failure` flag will always be `False`. (see next options for this)

Argument: fields_to_check_fieldname

This does exactly the same as `fields_to_check_list`, but instead of providing the list of fields to check as a comma-separated list, we provide the name of a field that contains the list of fields to check.

You would therefore call the command as follows:

Example:

```
| eval fields_list="action,app,bytes,url"
| trackmefieldsquality fields_to_check_fieldname="fields_list"
```

The point of having this option is that could for instance use a Splunk subsearch to generate the list of fields dynamically, for instance by accessing a lookup table where you store the fields depending on your criteria, or any other solution of your choice.

Hint

Providing a JSON dictionary model

- The next 3 options allow to provide a JSON dictionary model that models the fields to check, as well as optional parameters for each field.
- Especially, you can define a regular expression with the field `regex` to be apply against the value, allowing to valid the content of the field according to any needs.

- You can also define the field `allow_unknown` to be `True` or `False`, which can be used to disable the check for the field `is_unknown`.

Argument: fields_to_check_dict

This option is more sophisticated and allows to define a dictionnary that models the fields to check, and as well as an option regular expression to be apply against the value.

For instance, the following distionnary would verify the fields `bytes` including the fact that this should be a numerical value using a regular expression, the field `action` that for instance would accept only `success` or `failure`, and finally the field `http_referrer` where we would only perform the basic checks without verifying that its value matches certain criteria.

```
| trackmefieldsquality fields_to_check_dict="{\"bytes\": {\"name\": \"bytes\", \
↪"regex\": \"^\\d*\"}, \"action\": {\"name\": \"action\", \"regex\": \"^
↪(success|failure)$\"}, \"http_referrer\": {\"name\": \"http_referrer\"}}"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True
```

In this case, if a regrex expression is provided, the `regex_failure` flag will be set to `True` if the value does not match the regular expression, and `False` otherwise, which accounts for status of the field in addition to the other checks.

Example of JSON output:


```
{
    "time": 1747261746,
    "bytes": {
        "status": "success",
        "description": "Field exists and is valid.",
        "is_missing": false,
        "is_empty": false,
        "is_unknown": false,
        "regex_failure": false,
        "value": "309"
    },
    "action": {
        "status": "failure",
        "description": "Field exists but value does not match the required␣
↪pattern.",
        "is_missing": false,
        "is_empty": false,
        "is_unknown": false,
        "regex_failure": true,
        "value": "Bad Request"
    },
    "http_referrer": {
        "status": "failure",
        "description": "Field is 'unknown'.",
        "is_missing": false,
        "is_empty": false,
        "is_unknown": true,
        "regex_failure": false,
        "value": "unknown"
    },
    "summary": {
        "overall_status": "failure",
        "total_fields_checked": 3,
```


(continues on next page)

(continued from previous page)


```
        "total_fields_failed": 2,
        "total_fields_passed": 1,
        "percentage_failed": 66.67,
        "percentage_passed": 33.33
    },
    "metadata": {
        "time_epoch": 1747261746,
        "time_human": "Wed May 14 22:29:06 2025 UTC",
        "index": "webserver",
        "sourcetype": "nginx:plus:kv",
        "host": "trackme-solutions.com",
        "source": "/var/log/nginx/access.log",
        "datamodel": "Web"
    },
    "event_id": "f5ed1437ee8486a3782ebaea846dad37c52d47b825d1913c1ae7d085ba01f943"
}
```


Hint

Escaping backslashes and special characters

- The tricky part is that you need to pay attention to the JSON provided as input to the command
- Especially, double quotes within the JSON string need to be escaped.
- The regular expression also needs to be escaped, for instance `^\\\d*` which would otherwise be `\d*` in normal circumstances.

Argument: fields_to_check_dict_fieldname

Similarly to `fields_to_check_fieldname`, this option allows to provide the name of a field that contains the dictionary of fields to check as in the previous example.

This allows to use a Splunk subsearch to generate the dictionary dynamically, for instance by accessing a lookup table where you store the fields depending on your criteria, or any other solution of your choice.

Example:

```
| eval fields_dict="{\"bytes\": {\"name\": \"bytes\", \"regex\": \"^\\\d*\"}, \
↪"action\": {\"name\": \"action\", \"regex\": \"^(success|failure)$\"}, \
↪"http_referrer\": {\"name\": \"http_referrer\"}}"
| trackmefieldsquality fields_to_check_dict_fieldname="fields_dict"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True
```

Argument: fields_to_check_search_command

This option allows to provide a search command that will be executed to generate the dictionary of fields to check dynamically. This is particularly useful when you want to generate the dictionary on-the-fly using commands like `trackmefieldsqualitygendict` or any other search command that produces the required JSON dictionary format.

The search command will be executed and its output will be used as the dictionary for fields quality assessment. This provides maximum flexibility for dynamic dictionary generation.

Example:

```
| trackmefieldsquality fields_to_check_search_command="|␣
↪trackmefieldsqualitygendict datamodel=Web show_only_recommended_fields=True"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True
```

Example with custom search command:

```
| trackmefieldsquality fields_to_check_search_command="| inputlookup␣
↪my_data_dict_lookup | where somefilter=\"somevalue\"" pretty_print_json=False␣
↪output_mode=json metadata_fields="datamodel" include_field_values=True
```

Argument: fields_to_check_dict_path

This option is similar to `fields_to_check_dict_fieldname`, but instead of providing the dictionary as a JSON string, we provide the path to a JSON file that contains the dictionary.

The file must exist on the file system of the Splunk instance, and the path must be provided as a string.

Our JSON file would look like this:

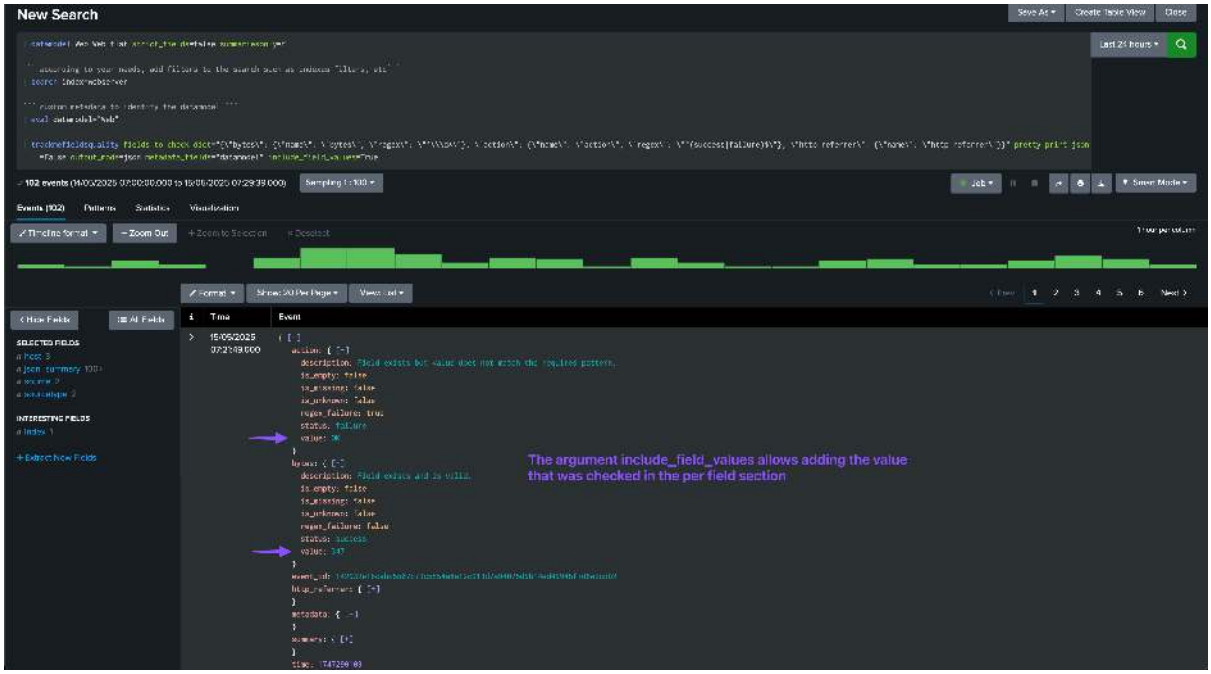
```
{
    "action": {
        "name": "action",
        "regex": "^(success|failure)$"
    },
    "bytes": {
        "name": "bytes",
        "regex": "^\\d*"
    },
    "http_referrer": {
        "name": "http_referrer"
    }
}
```

Example:

```
| trackmefieldsquality fields_to_check_dict_path="/
↪opt/splunk/etc/apps/myapp/mydir/web_datamodel.json" pretty_print_json=False␣
↪output_mode=json metadata_fields="datamodel" include_field_values=True
```

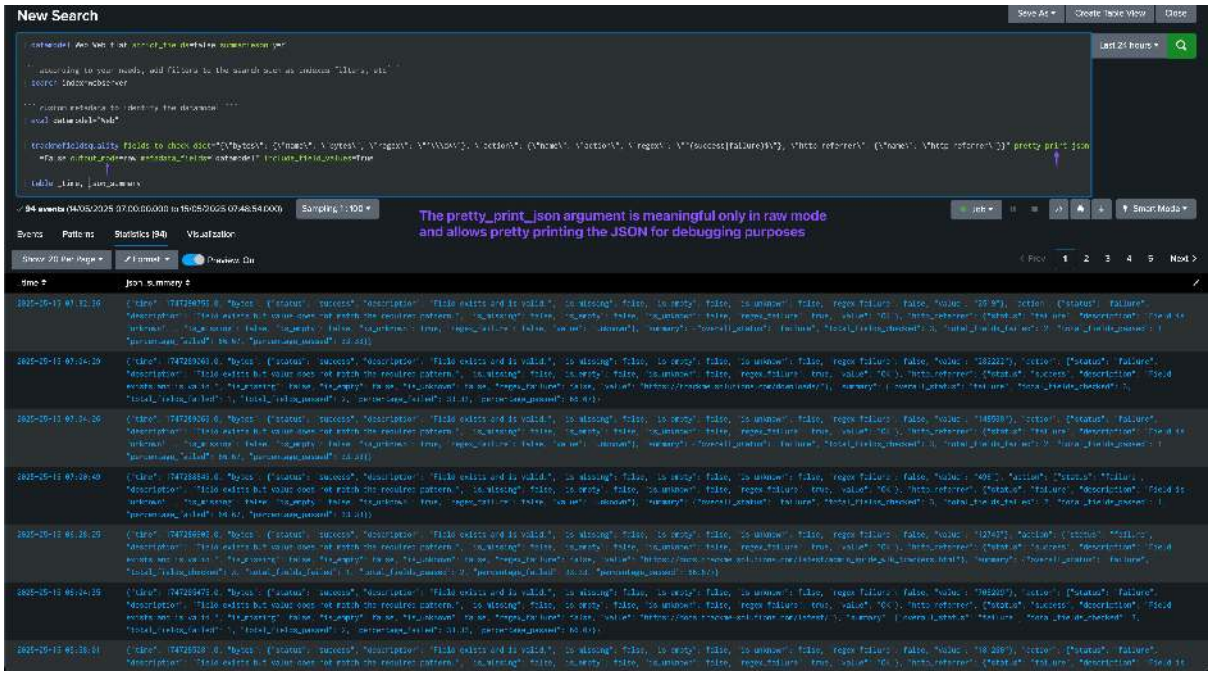
Argument: include_field_values

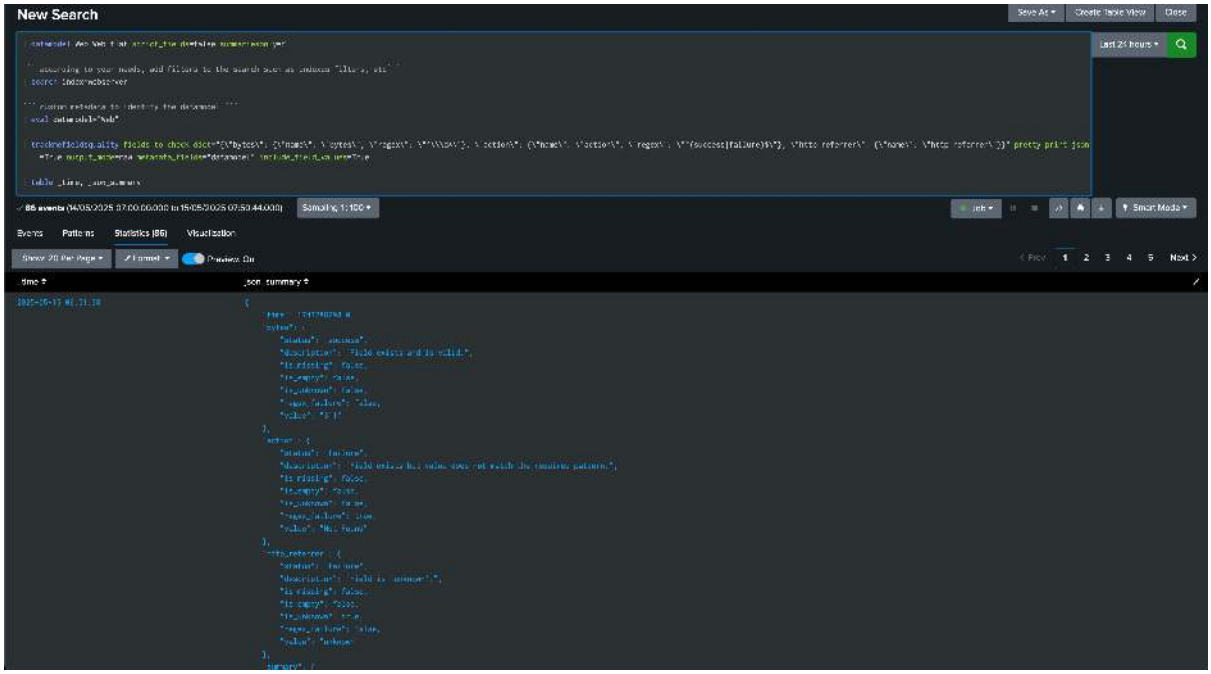
This option allows to include the field values in the JSON output, which can useful for analytics and reporting purposes.



Argument: pretty_print_json

This option allows to pretty print the JSON output, which can be useful for debugging purposes.





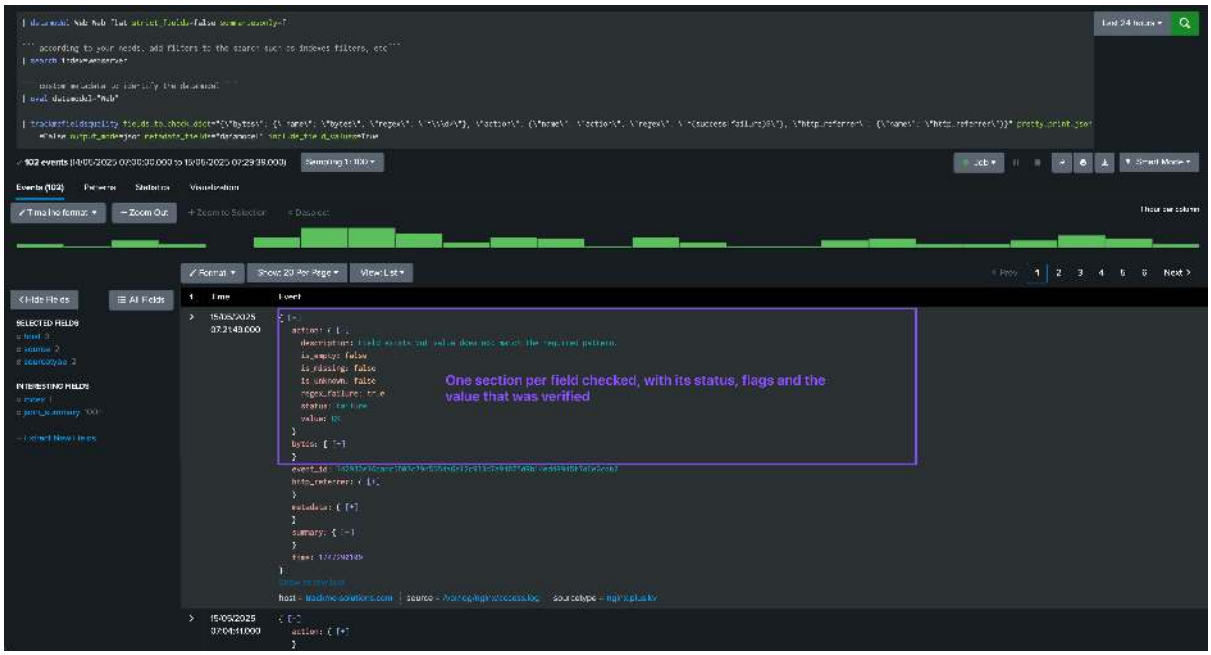
Argument: output_mode

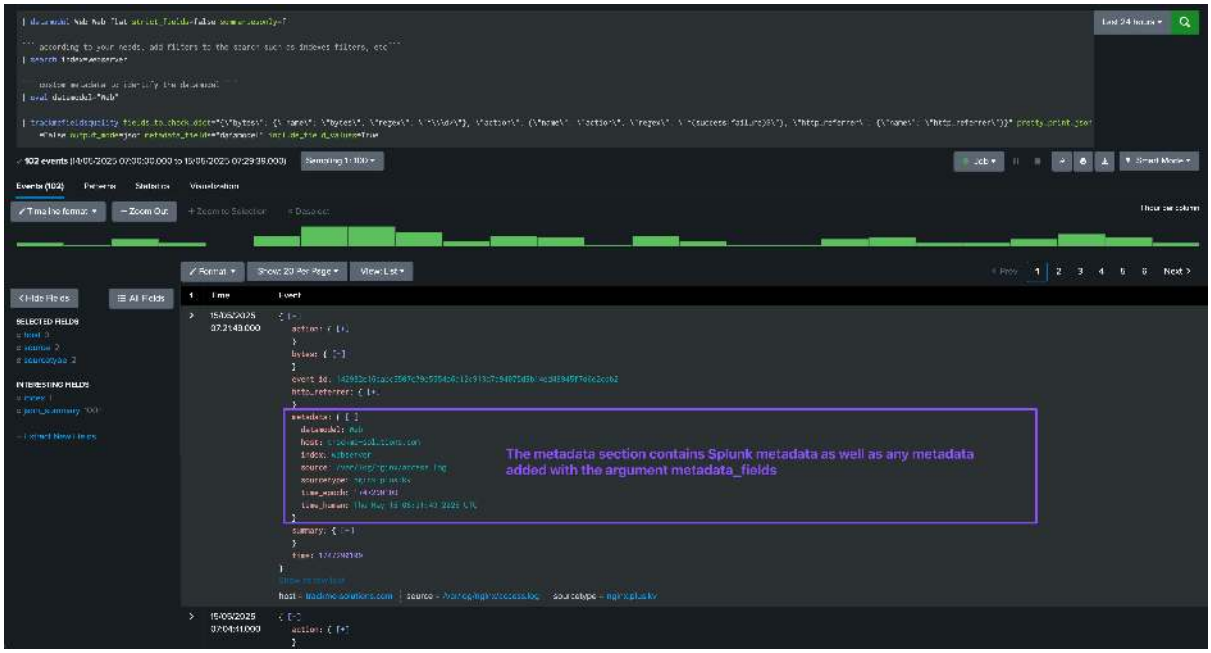
This option allows to specify the output mode, which can be `json` or `raw`.

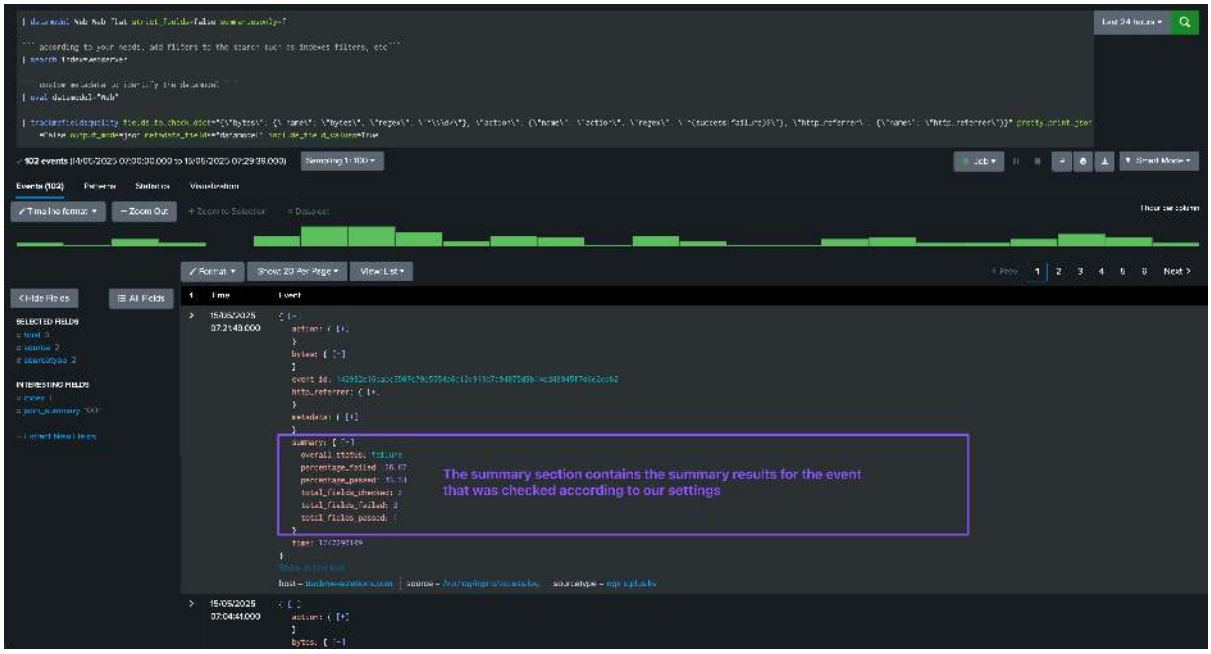
We recommend using the `json` mode for most use cases, especially this document is focused on the JSON output which is leveraged and indexed using the TrackMe sourcetype `trackme:fields_quality`.

output_mode=json

In JSON mode, the command verifies the fields and generates a JSON object per event:

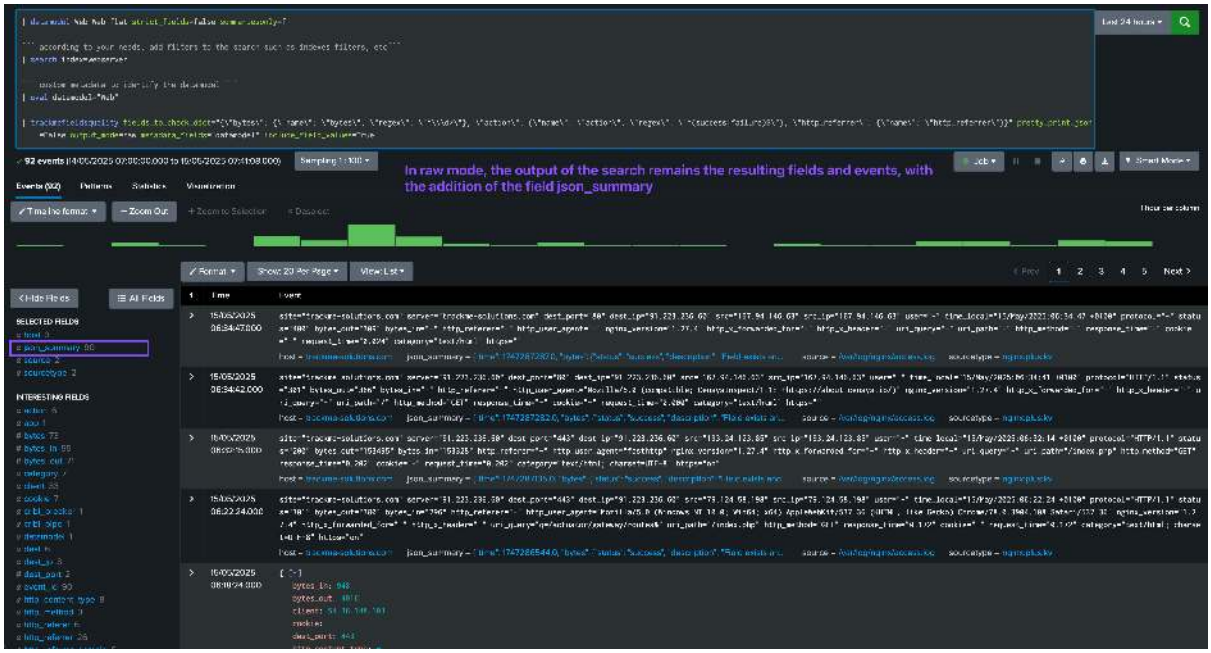


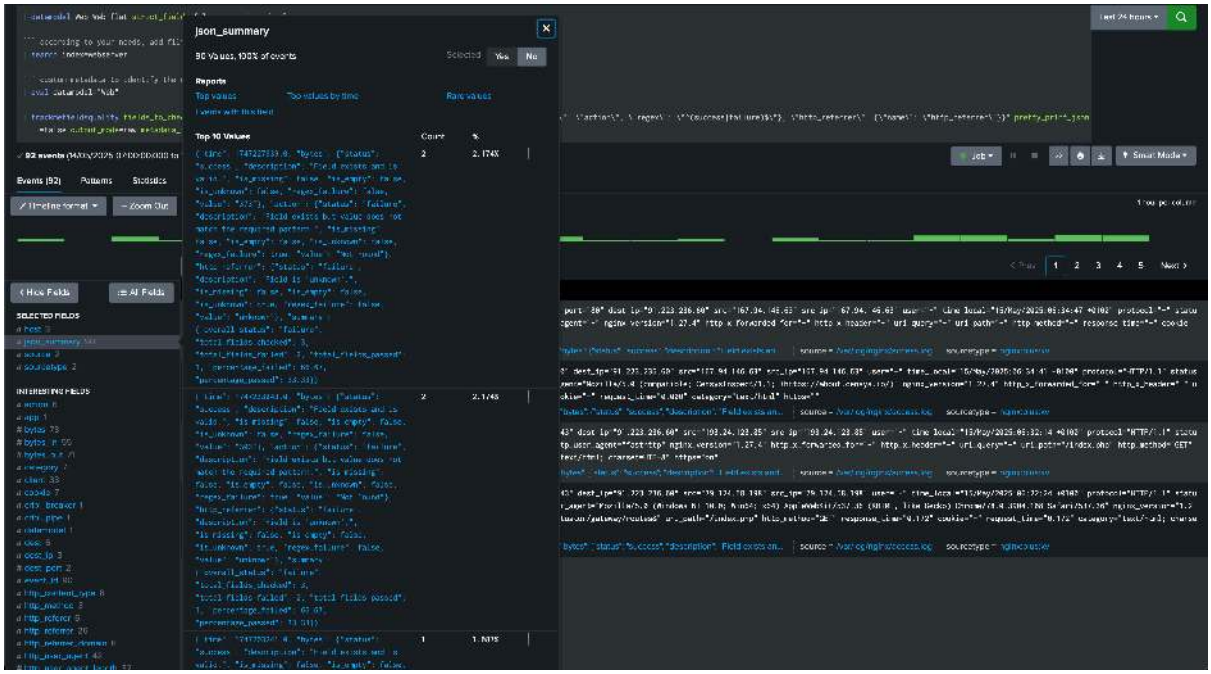




output_mode=raw

In raw mode, the command generates the events as they are, and adds a field called `json_summary` which contains our JSON object:





Argument: metadata_fields

This option allows to specify the metadata fields as a comma-separated list of fieldsto include in the JSON output.

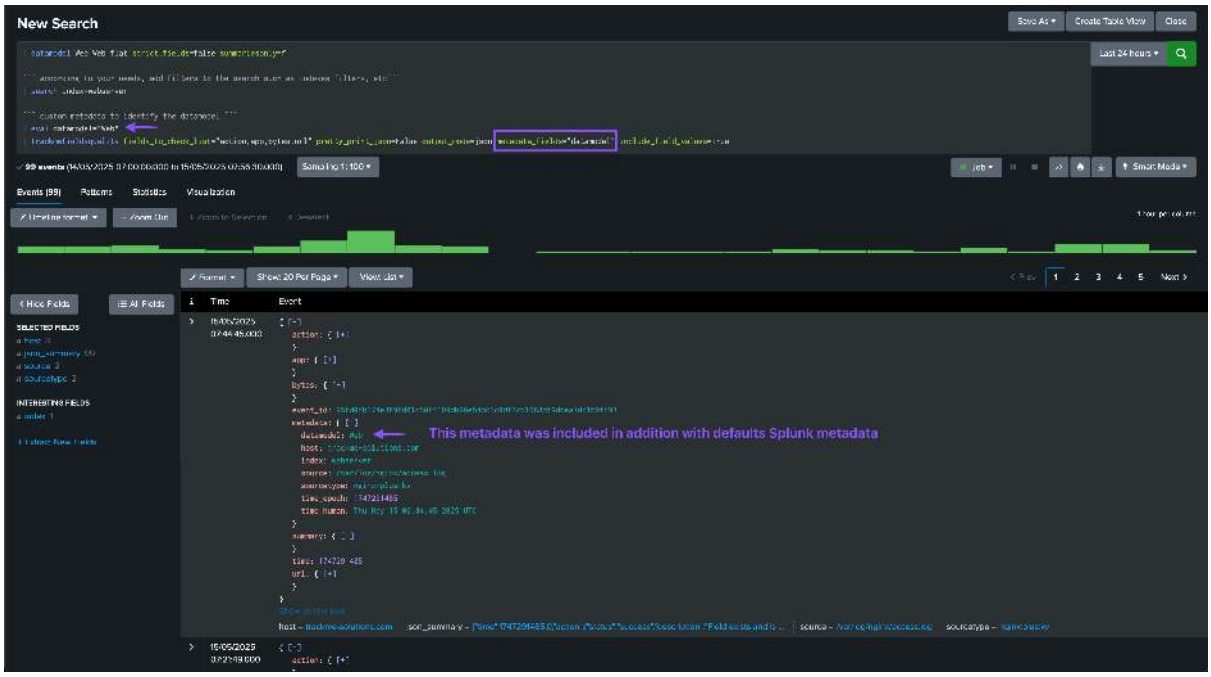
The metadata fields always include the following:

- `index`: the index of the event
- `sourcetype`: the sourcetype of the event
- `host`: the host of the event
- `source`: the source of the event

By defining the `metadata_fields` parameter, you can add additional fields to the JSON output, for instance the `datamodel` field which in our implementation is used to identify the data model of the event.

In our example, we are defining a field called `datamodel` using an eval which will be added to the JSON output:

```
| eval datamodel="Web"
| trackmefieldsquality fields_to_check_list="action,app,bytes,url"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True
```

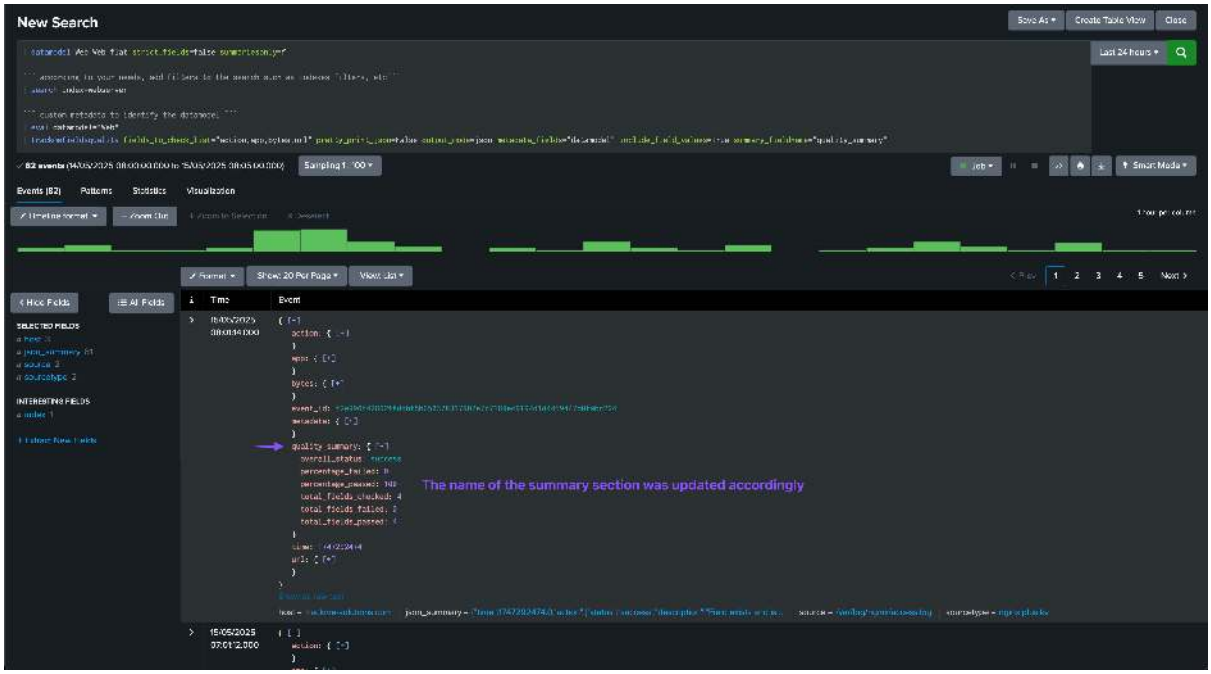


Argument: summary_fieldname

This option allows to specify the name of the summary field in the JSON output, this defaults to `summary` but can be customised if needed, for instance if there is a conflict with a field from the data model or events.

In this example, instead of `summary`, we use `quality_summary`:

```
| trackmefieldsquality fields_to_check_list="action,app,bytes,url"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True summary_fieldname="quality_summary"
```

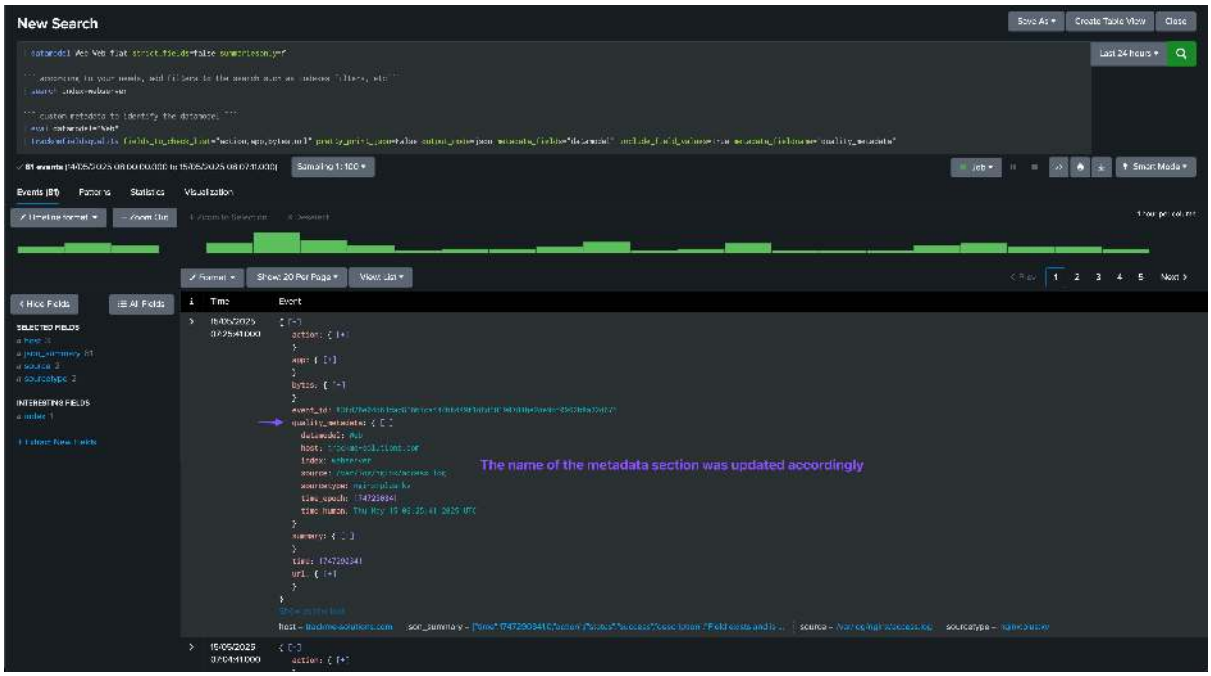


Argument: metadata_fieldname

This option allows to specify the name of the metadata field in the JSON output, this defaults to `metadata` but can be customised if needed, for instance if there is a conflict with a field from the data model or events.

In this example, instead of `metadata`, we use `quality_metadata`:

```
| trackmefieldsquality fields_to_check_list="action,app,bytes,url"␣
↪pretty_print_json=False output_mode=json metadata_fields="datamodel"␣
↪include_field_values=True metadata_fieldname="quality_metadata"
```



Annex: usage and options for the command trackmefieldsqualitygendict

The command `trackmefieldsqualitygendict` is used to generate JSON dictionary models for CIM data models, which can then be used with the `trackmefieldsquality` command for fields quality assessment.

The command accepts the following parameters:

argument	description	default	example or valid values
`datamodel`	The name of the CIM data model to generate the dictionary for	None (required)	"Authentication", "Web", "Network_Traffic"
`show_only_recomm`	Boolean option to only include recommended fields from the data model	False	True/False
`allow_unknown`	Boolean option to allow unknown field values	False	True/False
`allow_empty_or_mis`	Boolean option to allow empty or missing field values	False	True/False

Usage:

```
| trackmefieldsqualitygendict datamodel=<datamodel name>␣
↪show_only_recommended_fields=<boolean> allow_unknown=<boolean>␣
↪allow_empty_or_missing=<boolean>
```

Examples:

Generate dictionary for Authentication data model with only recommended fields:

```
| trackmefieldsqualitygendict datamodel="Authentication"␣
↪show_only_recommended_fields=true
```

Generate dictionary for Web data model with all fields and allow unknown values:

```
| trackmefieldsqualitygendict datamodel="Web" show_only_recommended_fields=false␣
↪allow_unknown=true
```

Generate dictionary for Network_Traffic data model with only recommended fields and allow empty/missing values:

```
| trackmefieldsqualitygendict datamodel="Network_Traffic"␣
↪show_only_recommended_fields=true allow_empty_or_missing=true
```

Generate dictionary for Authentication data model with all options enabled:

```
| trackmefieldsqualitygendict datamodel="Authentication"␣
↪show_only_recommended_fields=true allow_unknown=true allow_empty_or_missing=true
```

Output:

The command generates a JSON dictionary that can be used with the `trackmefieldsquality` command. The output includes:

- Field names as keys
- Field configuration objects as values, including: - `name`: The field name - `regex`: Optional regular expression pattern for validation - `allow_unknown`: Boolean flag to allow "unknown" values (can be overridden by the command parameter) - `allow_empty_or_missing`: Boolean flag to allow empty or missing values (can be overridden by the command parameter)

Example output for Web data model:

```
{
    "action": {
        "name": "action",
        "regex": "^(success|failure|allowed|blocked|deferred)$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "app": {
        "name": "app",
        "regex": "^[\\w:\\-\\d\\s]+$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "dest": {
        "name": "dest",
        "regex": "^[\\w\\.-]+$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "signature": {
        "name": "signature",
        "regex": "^.{3,100}$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "src": {
        "name": "src",
        "regex": "^[\\w\\.-]+$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "src_user": {
        "name": "src_user",
        "regex": "^[\\w\/\\\\\\-\\.$]{1,30}$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    },
    "user": {
        "name": "user",
        "regex": "^[\\w\/\\\\\\-\\.$]{1,30}$",
        "allow_unknown": false,
        "allow_empty_or_missing": false
    }
}
```

Integration with fields quality workflow:

This command is typically used in the first phase of the fields quality workflow to generate the dictionary models that will be stored in lookup tables and used by the collect scheduled searches:

```
| trackmefieldsqualitygendict datamodel=Authentication␣
↪show_only_recommended_fields=True
| append [ | trackmefieldsqualitygendict datamodel=Web␣
↪show_only_recommended_fields=True ]
| append [ | trackmefieldsqualitygendict datamodel=Network_Traffic␣
↪show_only_recommended_fields=True ]
| table datamodel, json_dict
```

(continues on next page)

(continued from previous page)

```
| trackmeprettyjson fields=json_dict
| outputlookup cim_datamodels_dict.csv
```

The generated lookup table can then be used in the collect scheduled searches:

```
| lookup cim_datamodels_dict.csv datamodel OUTPUT json_dict as quality_dict
| trackmefieldsquality fields_to_check_dict_fieldname="quality_dict"␣
↪output_mode=json metadata_fields="datamodel,nodename" include_field_values=True
```

Annex: usage and options for the command trackmefieldsqualitygensummary

The command `trackmefieldsqualitygensummary` is used to generate a summary of the quality of fields in records that have been processed by the `trackmefieldsquality` command. This command is typically used in the monitoring phase to aggregate and summarize field quality data for reporting and analysis purposes.

The command accepts the following parameters:

argument	description	default	example or valid values
`maxvals`	Max number of distinct values to report in field_values	15	10, 20, 50
`fieldvalues_format`	Format of field_values, either list or csv	csv	list, csv
`groupby_metadata_fie`	Comma-separated list of metadata fields to group by in addition to fieldname	"" (empty)	"metadata.datamodel,metadata.nodename,metadata.

Usage:

```
| trackmefieldsqualitygensummary maxvals=<max number of distinct values to report>␣
↪fieldvalues_format=<format of field_values, either list or csv>␣
↪groupby_metadata_fields=<comma separated list of metadata fields to group by in␣
↪addition to fieldname>
```

Examples:

Generate summary with default settings:

```
| trackmefieldsqualitygensummary
```

Generate summary with custom maxvals and list format:

```
| trackmefieldsqualitygensummary maxvals=20 fieldvalues_format=list
```

Generate summary with grouping by metadata fields:

```
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields=
↪"metadata.datamodel,metadata.nodename,metadata.index,metadata.sourcetype"
```

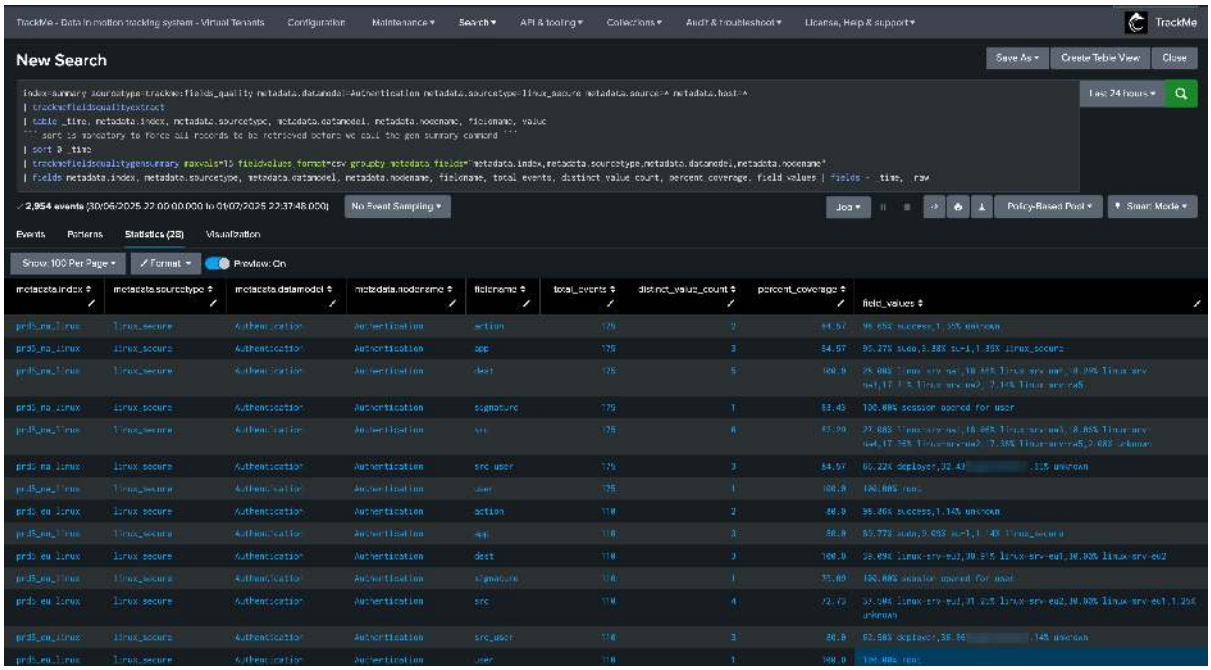
Output:

The command generates a summary that includes:

- **fieldname**: The name of the field being analyzed
- **total_events**: Total number of events processed for this field

- **distinct_value_count**: Number of distinct values found for this field
- **percent_coverage**: Percentage of events where this field has a value
- **field_values**: Summary of the most common values for this field (limited by maxvals)

Example output:



Example of field_values output:

The field_values column contains a summary of the most representative values for each field. For example:

- **CSV format**: "98.65% success,1.35% unknown"

Note: the list format renders the same information but as a proper list, which will appear in Splunk as a multi-value field.

Integration with fields quality workflow:

This command is typically used in the monitoring phase, specifically within the Flex Object tracker SPL logic to generate summaries of field quality data:

```
| search index=summary sourcetype=trackme:fields_quality metadata.sourcetype=*␣
↪metadata.source=* metadata.host=*
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel, metadata.
↪nodename, fieldname, value
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields=
↪"metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename"
```

The output from this command is then used to populate the `extra_attributes` field in the Flex Object tracker, providing detailed field quality information for each entity.

Argument: maxvals

This option controls the maximum number of distinct values to report in the field_values column. This is useful for limiting the output size while still providing meaningful insights into the most common values for each field.

Example:

```
| trackmefieldsqualitygensummary maxvals=10
```

This would limit the field_values output to show only the top 10 most common values for each field.

Argument: fieldvalues_format

This option controls the format of the field_values output. Two formats are supported:

- **csv**: Comma-separated values format (default)
- **list**: List format with percentages in parentheses

CSV format example: "98.65% success,1.35% unknown"

List format example: "success (98.65%), unknown (1.35%)"

Example:

```
| trackmefieldsqualitygensummary fieldvalues_format=list
```

Argument: groupby_metadata_fields

This option allows grouping the summary by additional metadata fields beyond the default fieldname grouping. This is particularly useful when you want to analyze field quality across different dimensions such as data models, nodes, indexes, or sourcetypes.

Example:

```
| trackmefieldsqualitygensummary groupby_metadata_fields=
↪"metadata.datamodel,metadata.nodename,metadata.index,metadata.sourcetype"
```

This would generate separate summaries for each combination of datamodel, nodename, index, and sourcetype, allowing for more granular analysis of field quality across different contexts.

Integration example in Flex Object tracker:

In the Flex Object tracker source code, this command is used to generate detailed field summaries that are stored in the entity's extra_attributes:

```
| join type=outer metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname [ search index=summary␣
↪sourcetype=trackme:fields_quality metadata.sourcetype=* metadata.source=*␣
↪metadata.host=*
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel, metadata.
↪nodename, fieldname, value
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields=
↪"metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename"
| fields metadata.index, metadata.sourcetype, metadata.datamodel, metadata.
↪nodename, fieldname, total_events, distinct_value_count, percent_coverage,␣
↪field_values, regex_expression | fields - _time, _raw
```

You can for example run this search to get the percentage of coverage for the field "action" in the datamodel Network_Traffic and nodename All_Traffic, for the sourcetype cisco:asa:

```
index=summary sourcetype=trackme:fields_quality metadata.datamodel=Network_Traffic␣
↪metadata.nodename=All_Traffic metadata.sourcetype=cisco:asa metadata.source=*␣
↪metadata.host=*
| table _time, _raw
```

(continues on next page)

(continued from previous page)

```
| trackmefieldsqualityextract

| table _time, metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, value, regex_expression
``` sort is mandatory to force all records to be retrieved before we call the gen␣
↪summary command ```
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields="metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename
| fields metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, total_events, distinct_value_count,␣
↪percent_coverage, field_values, regex_expression | fields - _time, _raw

| where fieldname="action"
```

This would be equivalent to run the following search:

```
index=summary sourcetype=trackme:fields_quality metadata.datamodel=Network_Traffic␣
↪metadata.nodename=All_Traffic metadata.sourcetype=cisco:asa metadata.source=*␣
↪metadata.host=* | head 100
| table _time, _raw
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel, metadata.
↪nodename, fieldname, value, is_empty, is_missing

| where fieldname="action"

| stats count, count(eval(is_empty=0 AND is_missing=0)) as count_covered,␣
↪count(eval(is_empty=1 OR is_missing=1)) as count_not_covered
| eval percentage_covered=count_covered/count*100
```

Annex: Per field table statistics

The following search example shows the statistics per field:

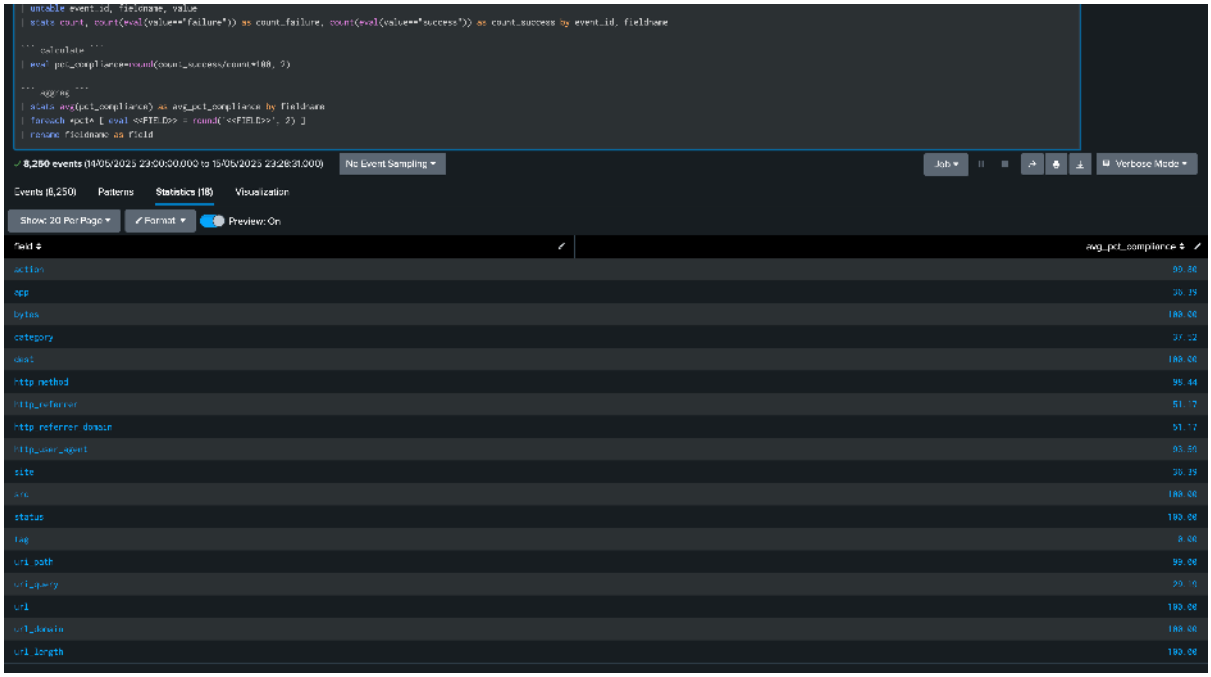
```
index=summary sourcetype=trackme:fields_quality
``` you can filter our metadata fields to focus on a specific sourcetype, index,␣
↪etc...```
| search metadata.datamodel="Web"

``` stats ```
| fields - summary.* metadata.*
| stats first(*status) as "*status" by event_id
| rename "*.status" as "*"

``` untable ```
| untable event_id, fieldname, value
| stats count, count(eval(value=="failure")) as count_failure,␣
↪count(eval(value=="success")) as count_success by event_id, fieldname

``` calculate ```
| eval pct_compliance=round(count_success/count*100, 2)

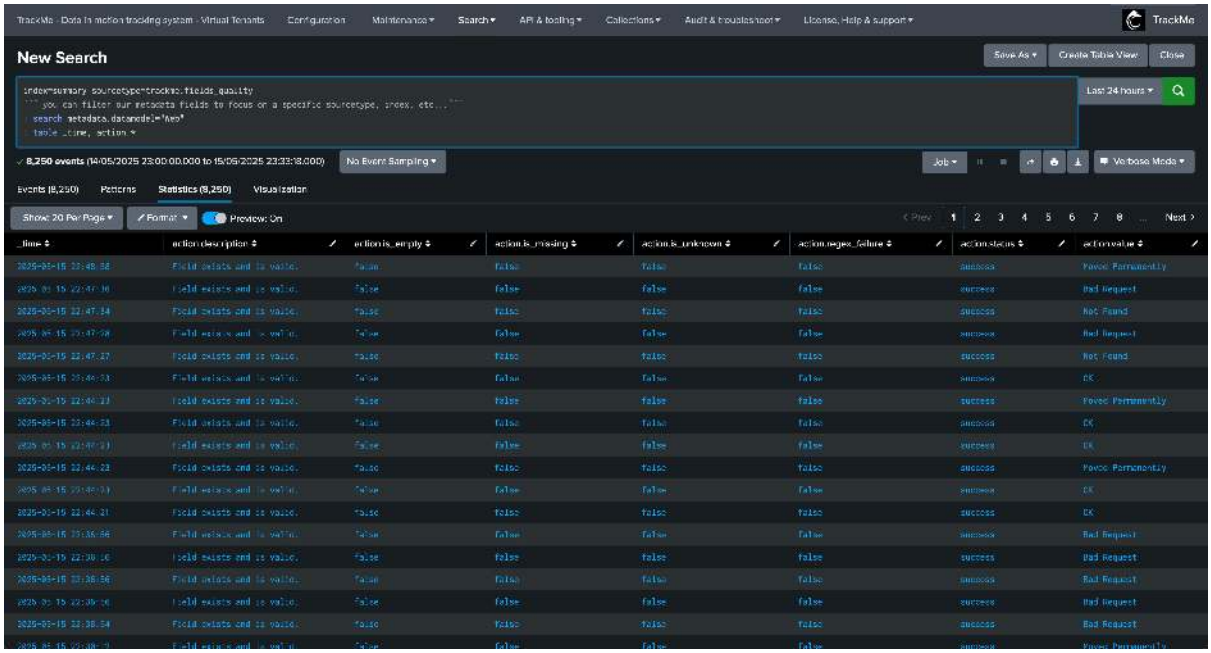
``` aggreg ```
| stats avg(pct_compliance) as avg_pct_compliance by fieldname
| foreach *pct* [ eval <<FIELD>> = round('<<FIELD>>', 2) ]
| rename fieldname as field
```



Annex: Looking after a specific field

The following search example shows the statistics per field:

```
index=summary sourcetype=trackme:fields_quality
``` you can filter our metadata fields to focus on a specific sourcetype, index,␣
↪etc...```
| search metadata.datamodel="Web"
| table _time, action.*
```



Annex: Flex Object tracker SPL logic

The following search example shows the Flex Object tracker SPL logic:
```

```
index=summary sourcetype=trackme:fields_quality metadata.sourcetype=*␣
↪metadata.source=* metadata.host=*
| table _time, _raw
| trackmefieldsqualityextract

| stats count as count_total, count(eval(status=="success")) as count_success,␣
↪count(eval(status=="failure")) as count_failure by fieldname, metadata.index,␣
↪metadata.sourcetype, metadata.datamodel, metadata.nodename
| eval percentage_success=round(count_success/count_total*100, 2)

``` set the threshold per field, this defines if the field is considered as passed␣
↪or failed globally, this threshold has to be part of the SPL logic ```
| eval threshold=99

``` flag field ```
| eval fieldstatus=if(percentage_success>=threshold, "success", "failure")

``` join field summary ```
| join type=outer metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname [ search index=summary␣
↪sourcetype=trackme:fields_quality metadata.sourcetype=* metadata.source=*␣
↪metadata.host=*
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, value, regex_expression
``` sort is mandatory to force all records to be retrieved before we call the gen␣
↪summary command ```
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields="metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename
| fields metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, total_events, distinct_value_count,␣
↪percent_coverage, field_values, regex_expression | fields - _time, _raw
]

``` generate the per field json for reporting purposes, we also rename fieldname␣
↪to @fieldname so it appears first in the JSON ```
| rename fieldname as @fieldname, fieldstatus as @fieldstatus
| tojson
| rename _raw as per_field_json

``` calculate ```
| stats values(eval(if('@fieldstatus'=="success", '@fieldname', null()))) as␣
↪list_fields_passed, values(eval(if('@fieldstatus'=="failure", '@fieldname',␣
↪null()))) as list_fields_failed, values(per_field_json) as per_field_json,␣
↪max(total_events) as total_events_parsed by metadata.index, metadata.sourcetype,␣
↪metadata.datamodel, metadata.nodename

``` format the per_field_json ```
| eval per_field_json = "[" . mvjoin(per_field_json, ", ") . "]"

``` build the list of fields that passed and failed ```
| eval all_fields = mvappend(list_fields_failed, list_fields_passed)
| eval all_fields = mvdedup(all_fields)
| eval final_state = mvmap(
all_fields,
if(
```

(continues on next page)

(continued from previous page)

```
mvfind(list_fields_failed, "^" . all_fields . "$") >= 0,
all_fields . "|failed",
all_fields . "|success"
)
)
| eval success_fields = mvfilter(match(final_state, "\|success$"))
| eval failed_fields = mvfilter(match(final_state, "\|failed$"))
| eval success_fields = mvmap(success_fields, mvindex(split(success_fields, "|"),␣
↪0))
| eval failed_fields  = mvmap(failed_fields,  mvindex(split(failed_fields, "|"),␣
↪0))
| fields - final_state
| eval success_fields=if(isnull(success_fields), "", success_fields),␣
↪failed_fields=if(isnull(failed_fields), "", failed_fields)
| fields - list_fields_passed, list_fields_failed

``` calculate ```
| eventstats dc(all_fields) as total_fields_checked, dc(success_fields) as␣
↪total_fields_passed, dc(failed_fields) as total_fields_failed by metadata.index,␣
↪metadata.sourcetype, metadata.datamodel, metadata.nodename
| eval percentage_passed=round(total_fields_passed/total_fields_checked*100, 2),␣
↪percentage_failed=round(total_fields_failed/total_fields_checked*100, 2)

``` rename ```
| rename metadata.* as "*"

``` save this as parts of extra attributes ```
| eval extra_attributes = "{" . "\"success_fields\": \"" . mvjoin(success_fields,
↪",") . "\", \"failed_fields\": \"" . mvjoin(failed_fields, ",") . "\", " . "\
↪"fields\":" . per_field_json . "}"
| fields - per_field_json

``` set principal metadata for the flex entity ```
| eval group = datamodel
| eval object = nodename . ":" . index . ":" . sourcetype, alias=index . ":" .␣
↪sourcetype
| eval object_description = "CIM Quality for DM: " . datamodel . ":" . nodename .
↪", index:" . index . ", sourcetype:" . sourcetype

``` gen metrics ```
| eval metrics = "{" .
"\"fields_quality.percentage_passed\": " . if(isnum(percentage_passed),␣
↪percentage_passed, 0) . ", " .
"\"fields_quality.percentage_failed\": " . if(isnum(percentage_failed),␣
↪percentage_failed, 0) . ", " .
"\"fields_quality.total_fields_checked\": " . if(isnum(total_fields_checked),␣
↪total_fields_checked, 0) . ", " .
"\"fields_quality.total_fields_failed\": " . if(isnum(total_fields_failed),␣
↪total_fields_failed, 0) . ", " .
"\"fields_quality.total_fields_passed\": " . if(isnum(total_fields_passed),␣
↪total_fields_passed, 0) . ", " .
"\"fields_quality.total_events_parsed\": " . if(isnum(total_events_parsed),␣
↪total_events_parsed, 0) . "}"

``` no outliers for now ```
| eval outliers_metrics="{}"
```

(continues on next page)

(continued from previous page)

```
``` basic status, thresholds can be defined on a per entity basis ```
| eval status=1
| eval status_description="DM Quality: " . metrics
| eval status_description_short="% passed: " . percentage_passed . ", checked: " .␣
↪total_fields_checked . ", passed: " . total_fields_passed . ", failed: " .␣
↪total_fields_failed

``` this sets a default threshold, which can then be overriden on a per entity␣
↪basis via the TrackMe UI ```
| eval default_threshold = "{'metric_name': 'fields_quality.percentage_passed',
↪'operator': '>=', 'value': 95, 'condition_true': 1}"

```alert if inactive for more than 2 days```
| eval max_sec_inactive=86400*2
```

Annex: Understanding & troubleshooting the percentage of success

The percentage of success is calculated by the main Flex tracker SPL logic, it represents the percentage of times the field is passing the quality check, a field is considered as passing the quality check if:

- `is_empty: false`
- `is_null: false`
- `is_unknown: false` (depending on if a dictionary is used and if in the dictionary and for this field, unknown is allowed or not)
- `regex_failure: false` (depending on if a dictionary is used and if there is a regex expression for this field)

```
index=summary sourcetype=trackme:fields_quality metadata.sourcetype=* metadata.
↪source=* metadata.host=*
| table _time, _raw
| trackmefieldsqualityextract

| stats count as count_total, count(eval(status=="success")) as count_success,␣
↪count(eval(status=="failure")) as count_failure by fieldname, metadata.index,␣
↪metadata.sourcetype, metadata.datamodel, metadata.nodename
| eval percentage_success=round(count_success/count_total*100, 2)
```

In this example, we are looking at the percentage of success for the field "action" in the datamodel Network_Traffic and nodename All_Traffic, for the sourcetype cisco:asa:

```
index=summary sourcetype=trackme:fields_quality metadata.datamodel=Network_Traffic␣
↪metadata.nodename=All_Traffic metadata.sourcetype=cisco:asa metadata.source=*␣
↪metadata.host=*
| table _time, _raw
| trackmefieldsqualityextract

| stats values(value) as values, values(regex_expression) as regex_expression,␣
↪count as count_total, count(eval(status=="success")) as count_success,␣
↪count(eval(status=="failure")) as count_failure by fieldname, metadata.index,␣
↪metadata.sourcetype, metadata.datamodel, metadata.nodename
| eval percentage_success=round(count_success/count_total*100, 2)

``` set the threshold per field, this defines if the field is considered as passed␣
↪or failed globally, this threshold has to be part of the SPL logic ```
```

(continues on next page)

(continued from previous page)

```
| eval threshold=99

``` flag field ```
| eval fieldstatus=if(percentage_success>=threshold, "success", "failure")

| where fieldname="action"
```

Annex: Understanding and troubleshooting the percentage of coverage and others summary information per field

The percentage of coverage is reported by the `trackmefieldsqualitygensummary` command, it represents the percentage of events where the field has a value, that is not null and not empty. (it is for instance present equally in the CIM Vladiator table)

In the Flex tracker, this is the join part:

```
search index=summary sourcetype=trackme:fields_quality metadata.sourcetype=*␣
↪metadata.source=* metadata.host=*
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, value, regex_expression
``` sort is mandatory to force all records to be retrieved before we call the gen␣
↪summary command ```
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields="metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename
| fields metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, total_events, distinct_value_count,␣
↪percent_coverage, field_values, regex_expression | fields - _time, _raw
```

You can for example run this search to get the percentage of coverage for the field "action" in the datamodel Network_Traffic and nodename All_Traffic, for the sourcetype cisco:asa:

```
index=summary sourcetype=trackme:fields_quality metadata.datamodel=Network_Traffic␣
↪metadata.nodename=All_Traffic metadata.sourcetype=cisco:asa metadata.source=*␣
↪metadata.host=*
| table _time, _raw
| trackmefieldsqualityextract

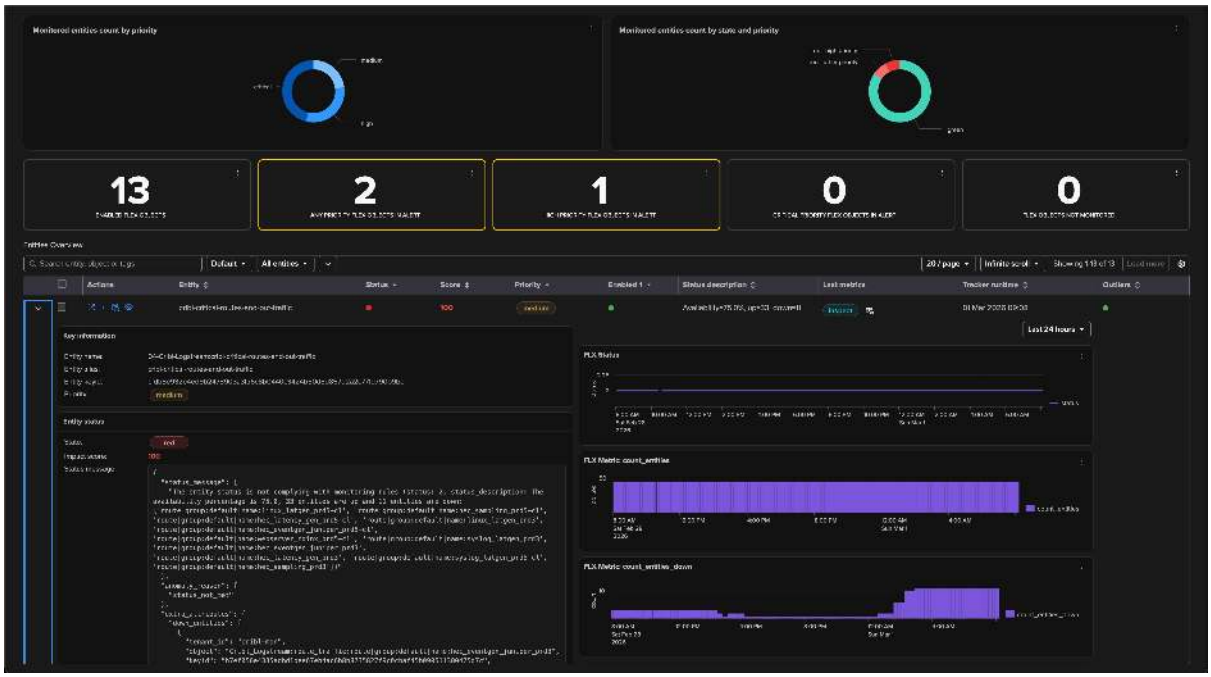
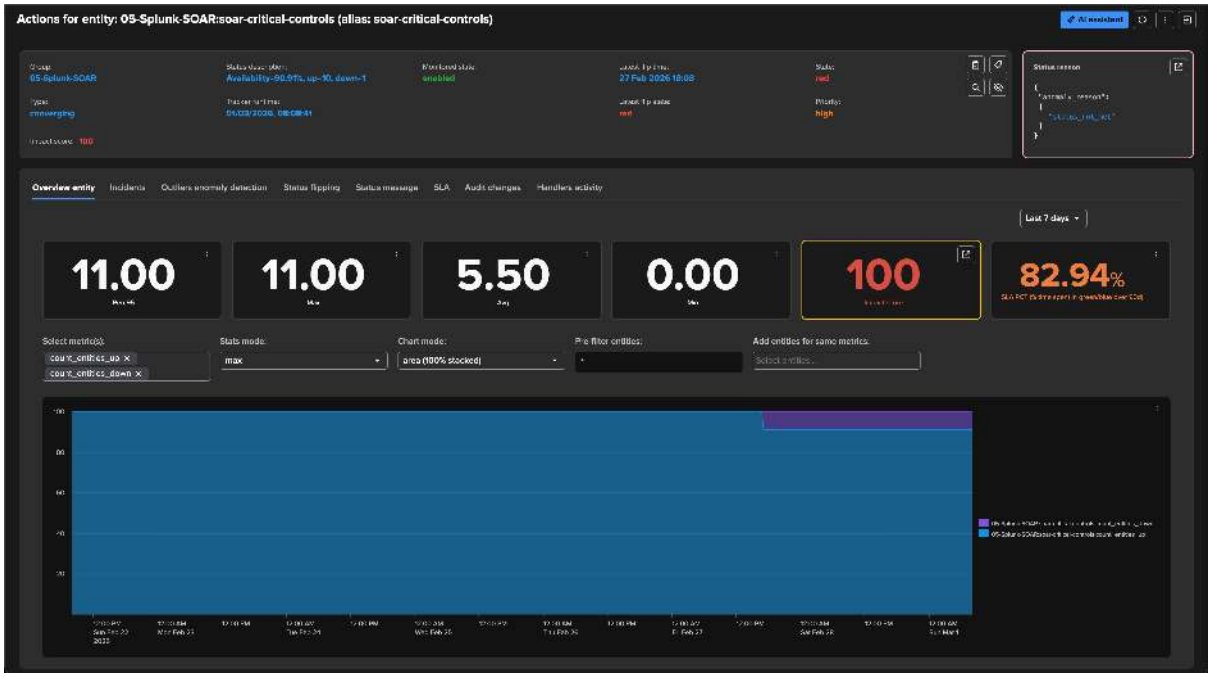
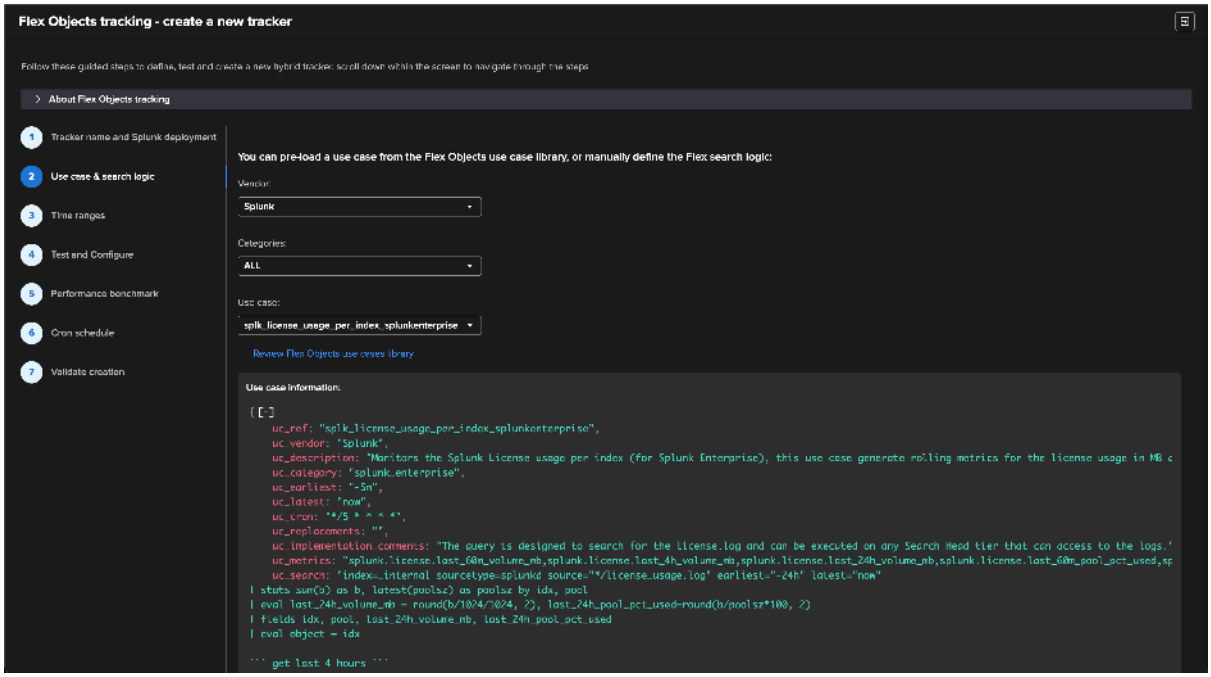
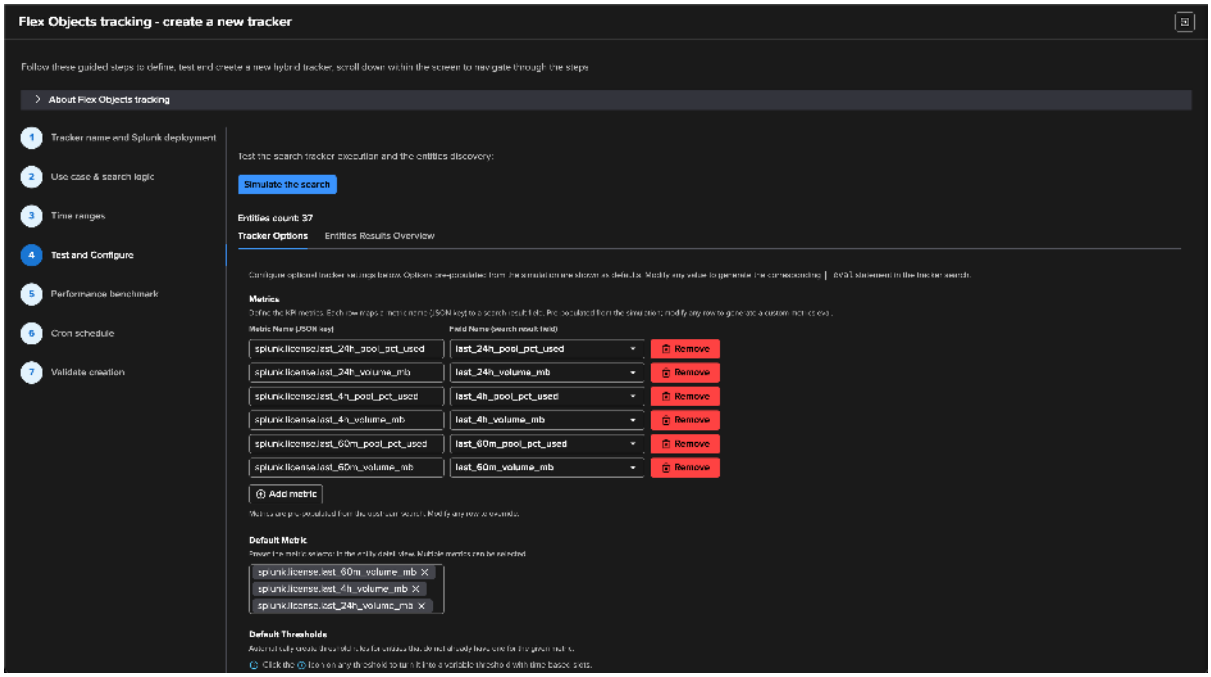
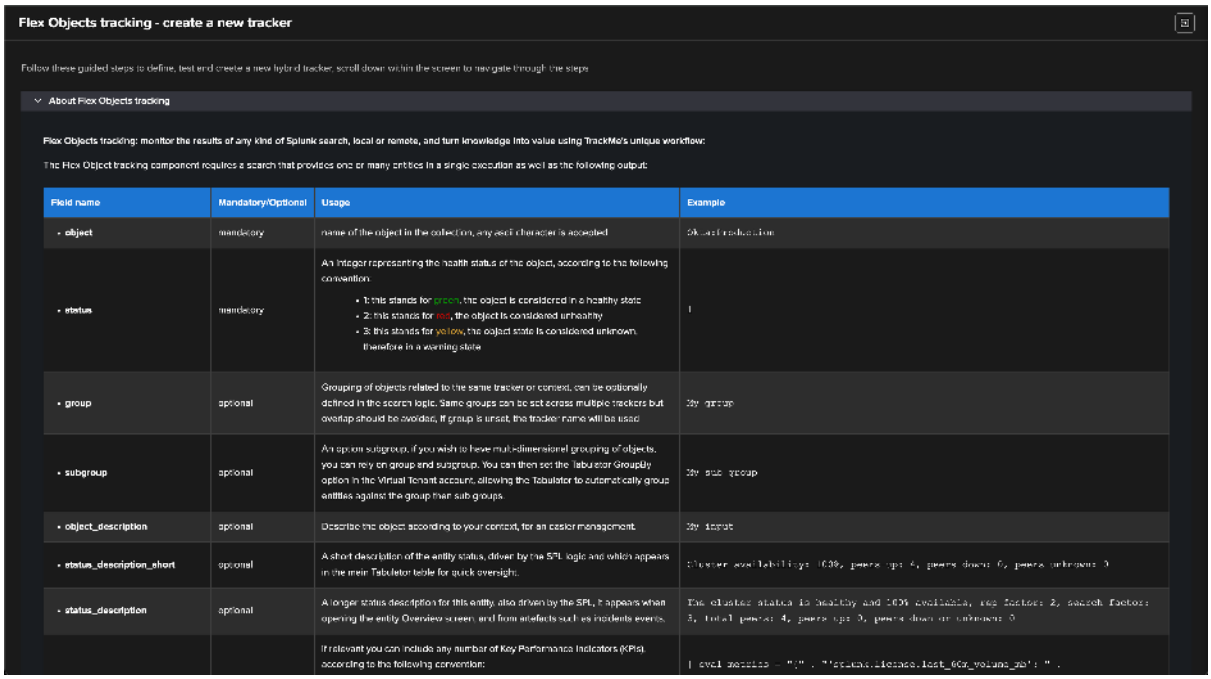
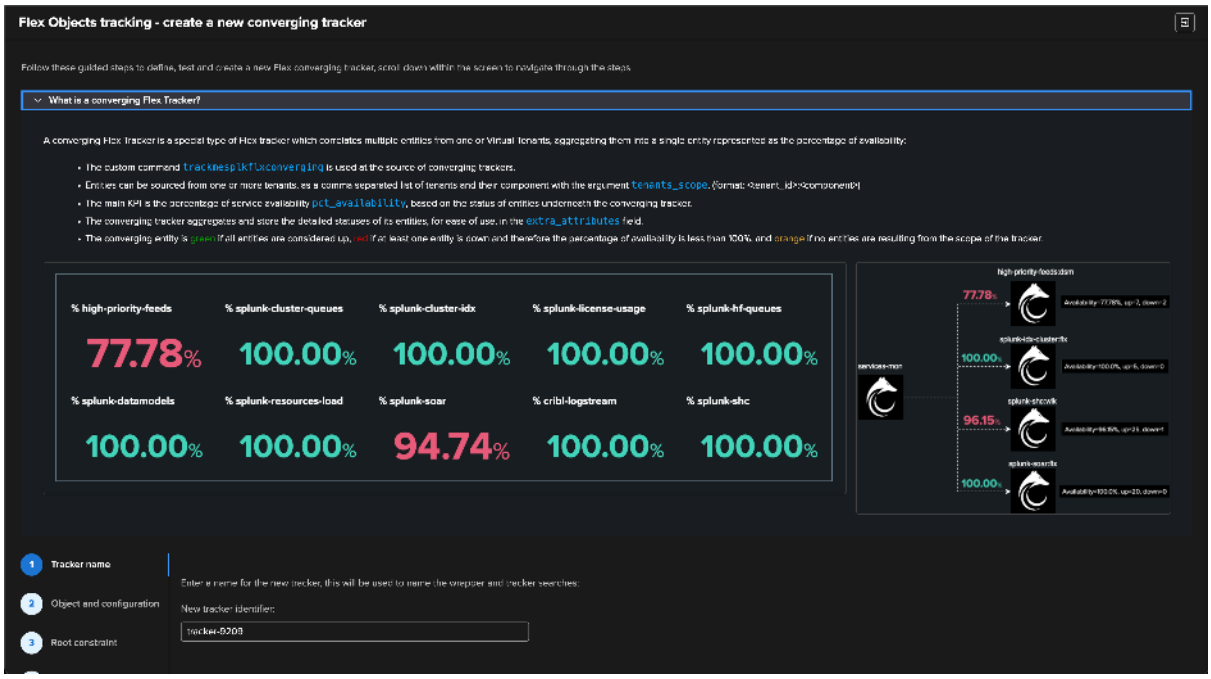
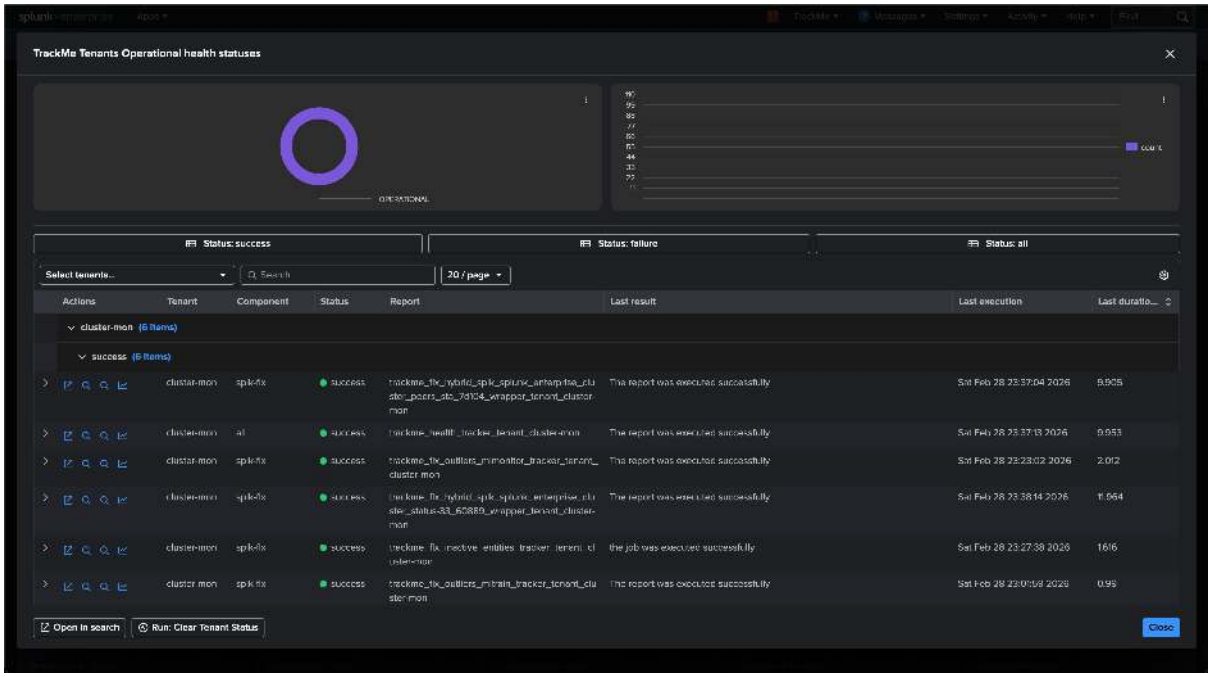
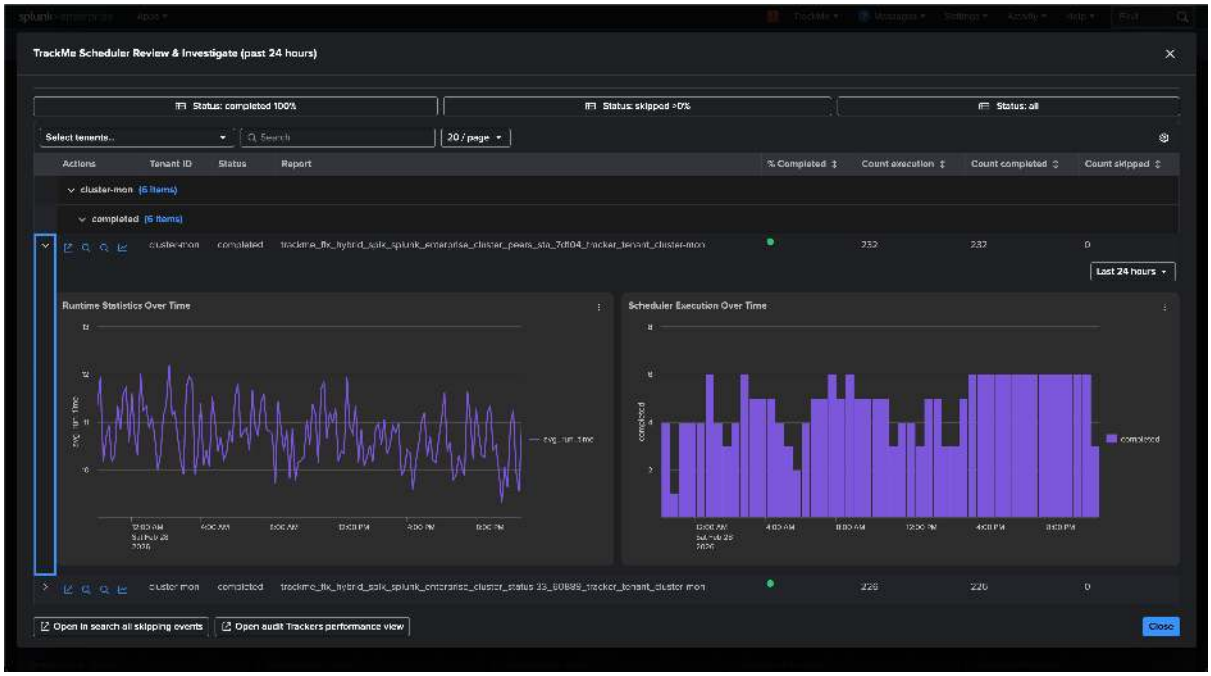
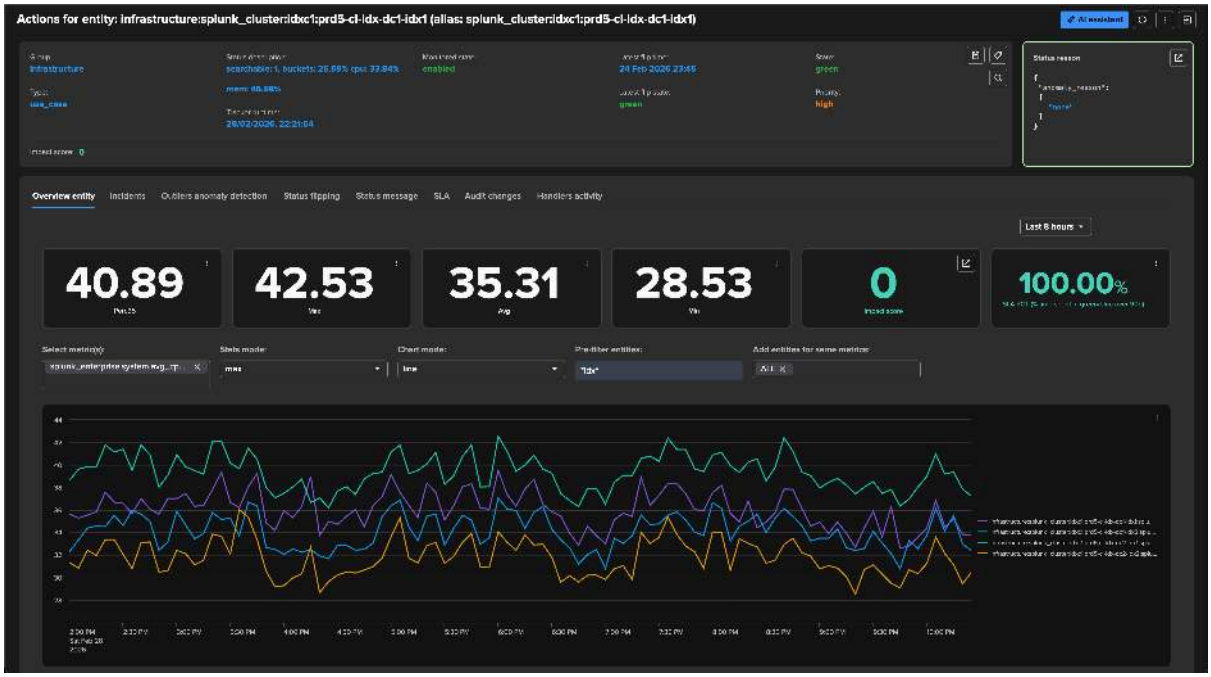
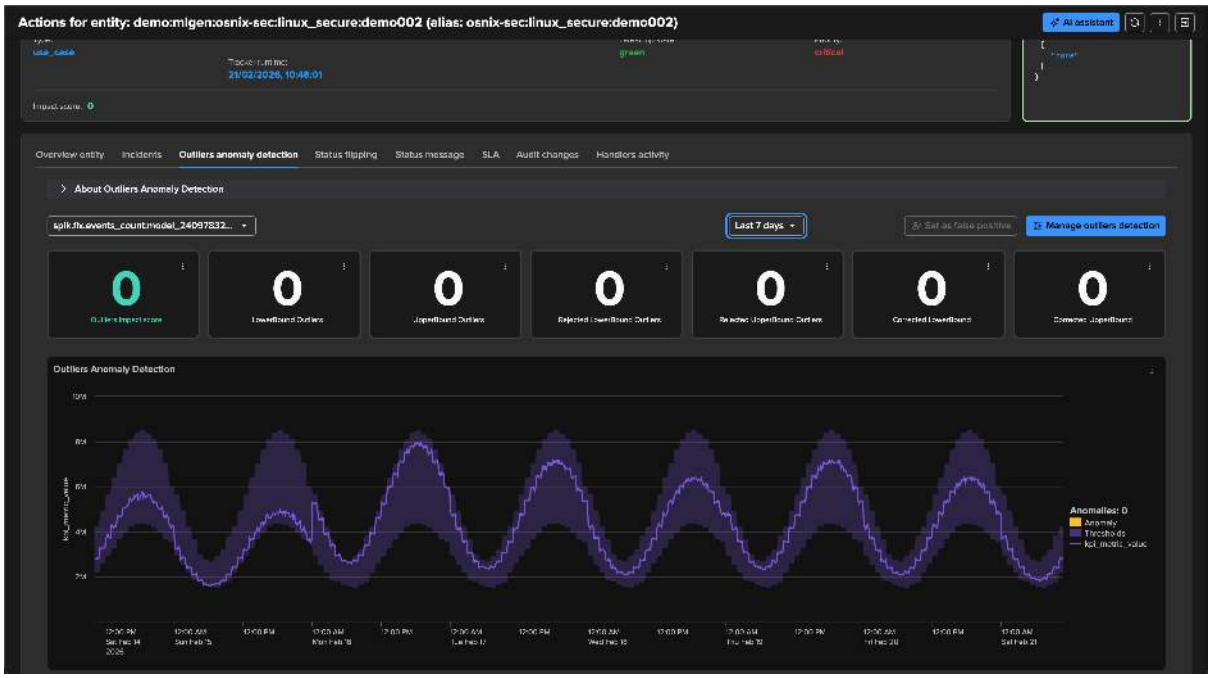
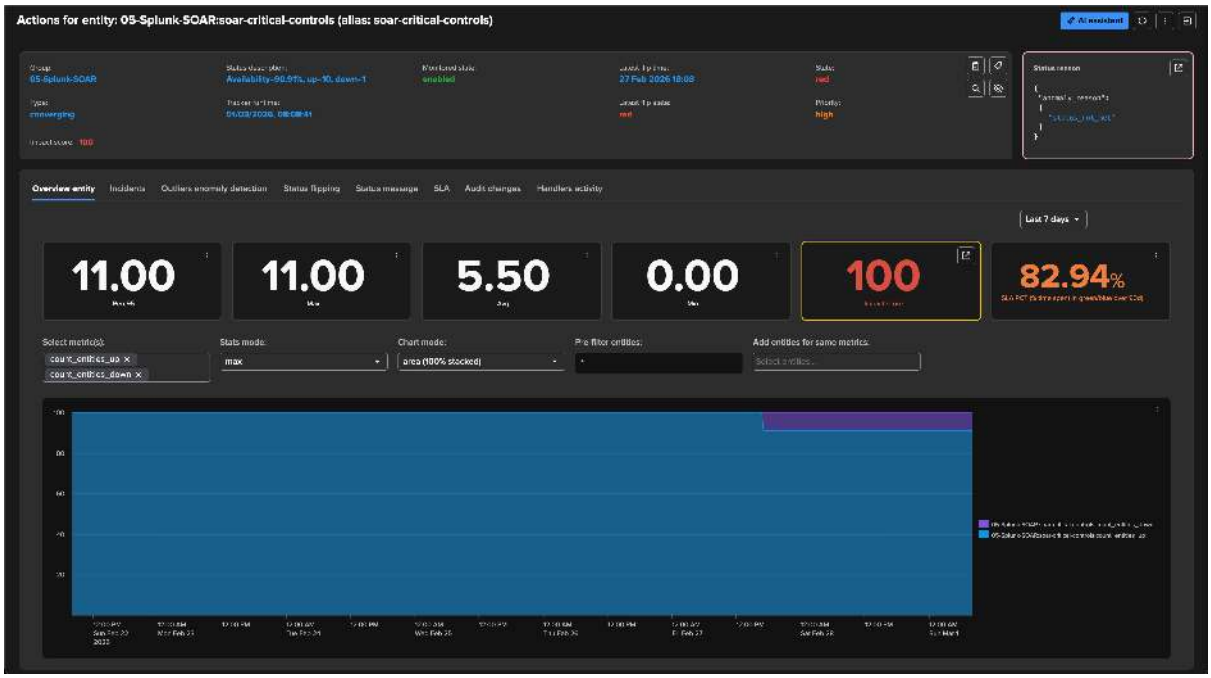
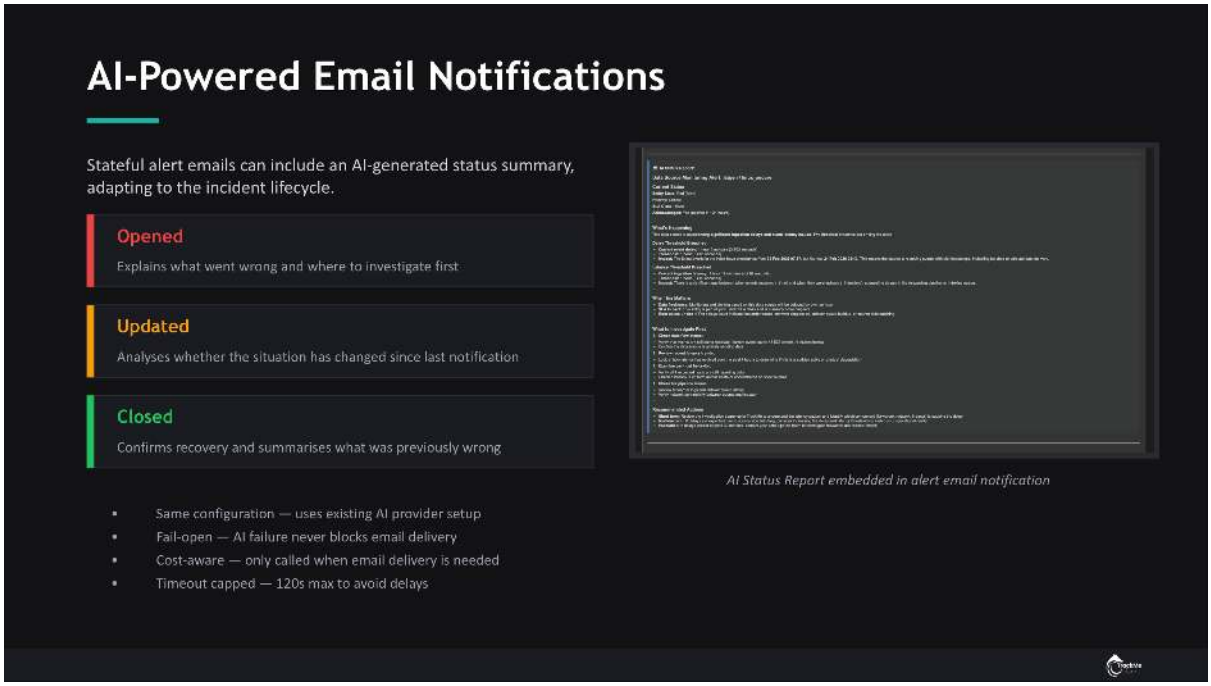
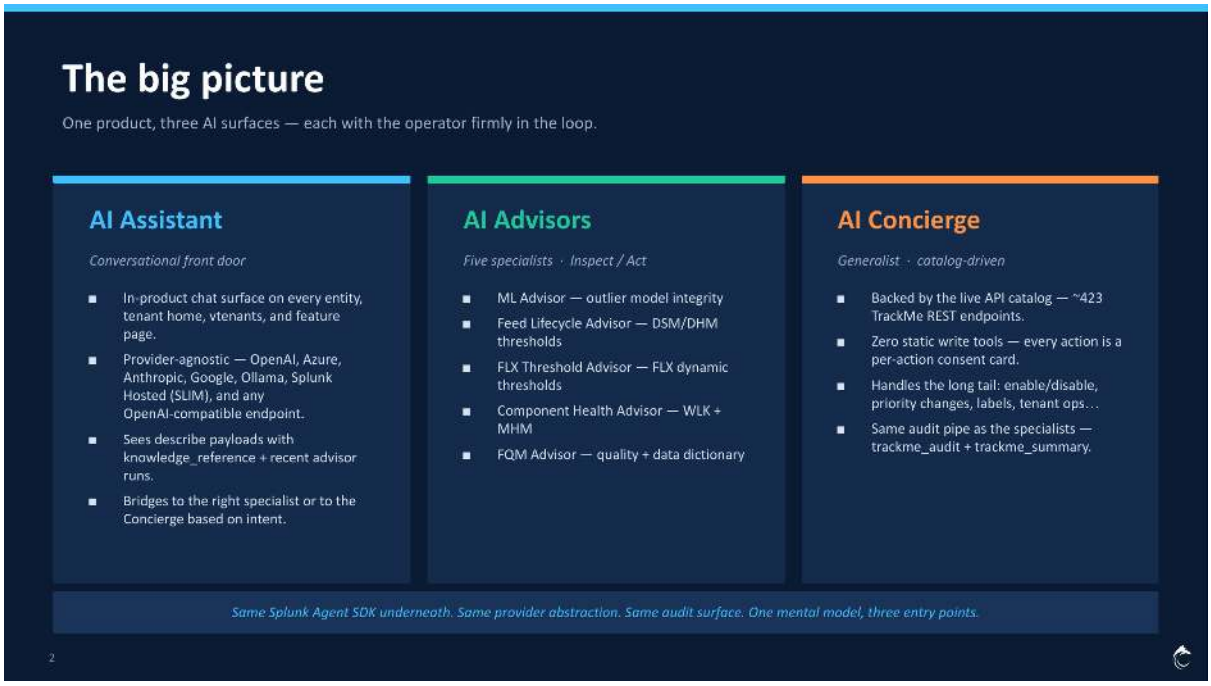
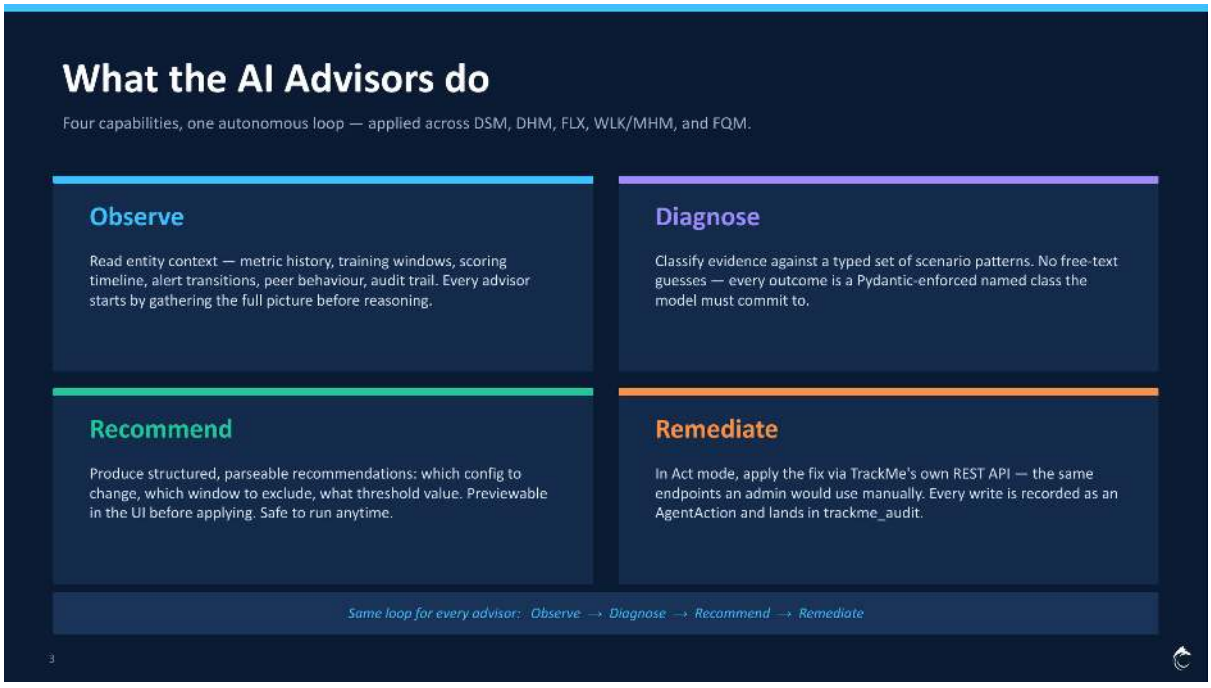
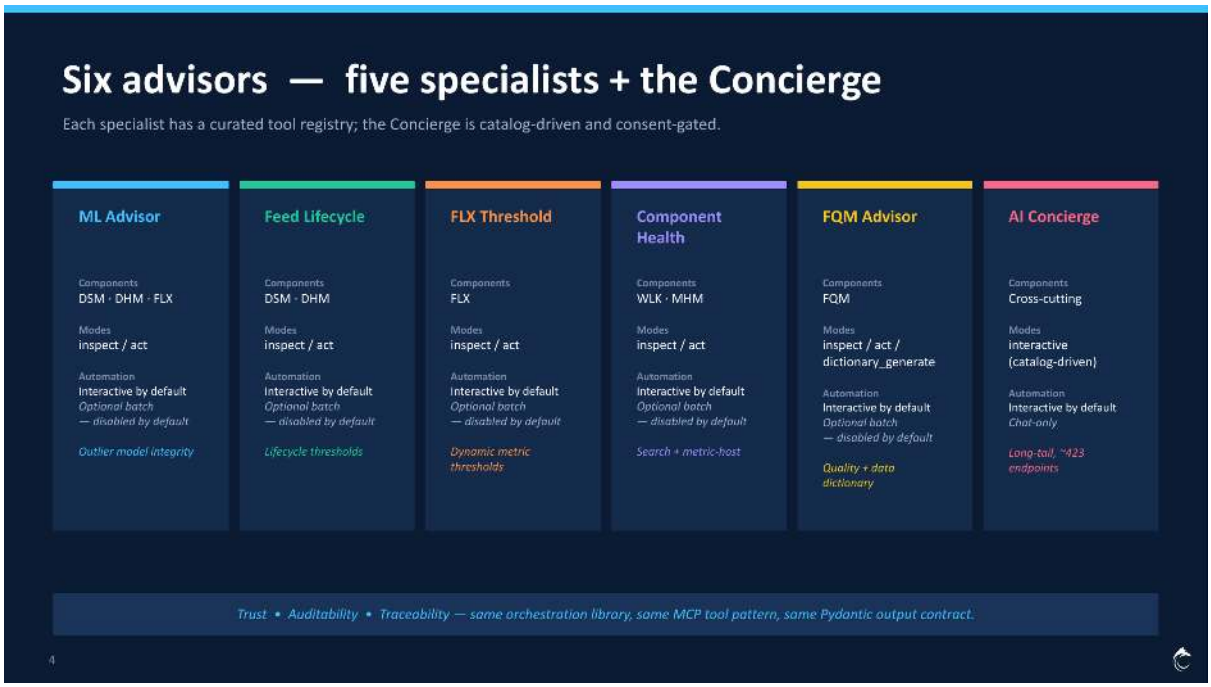
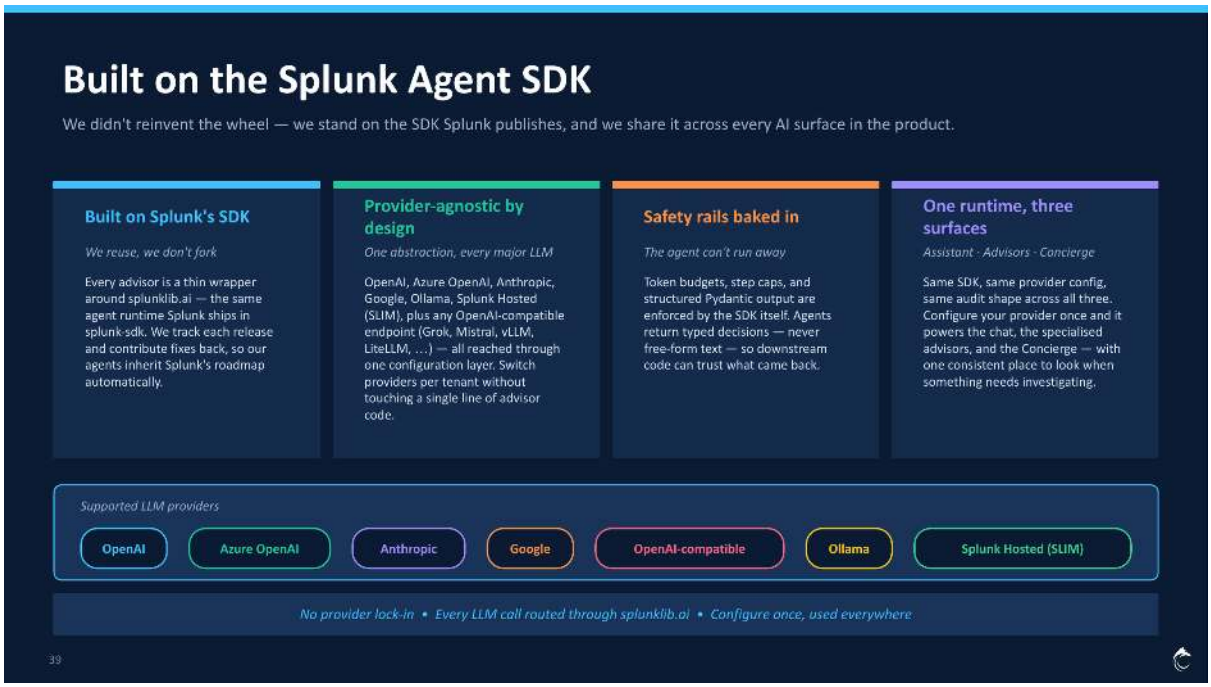
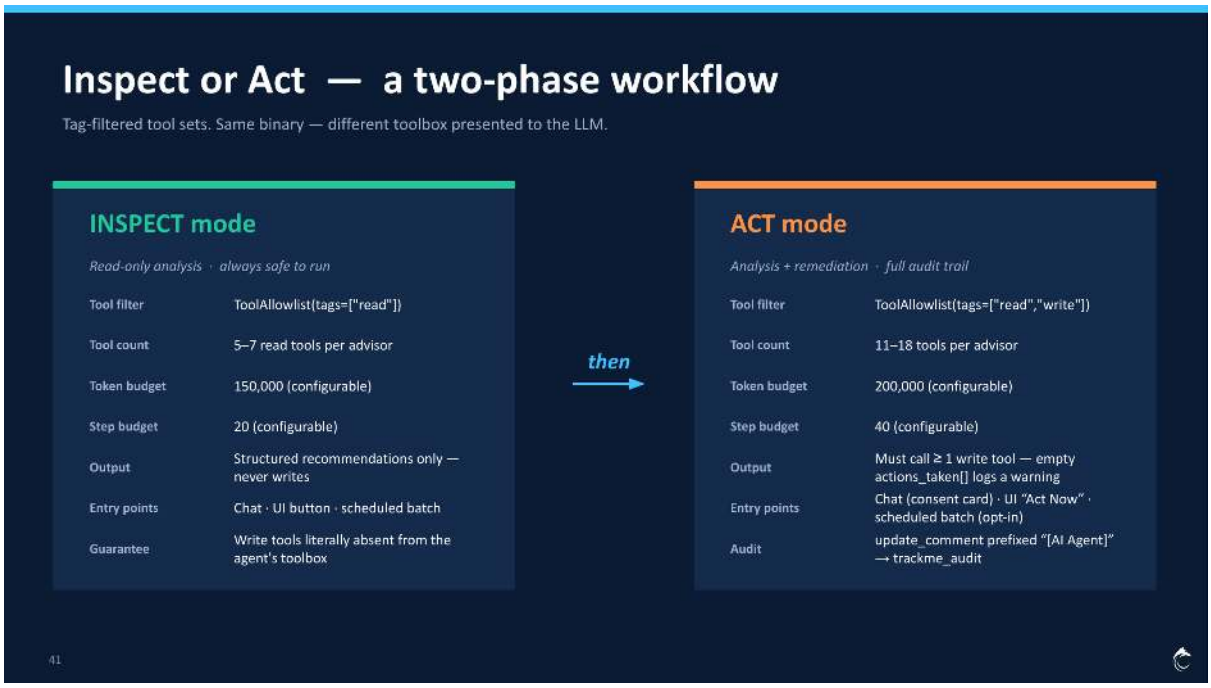
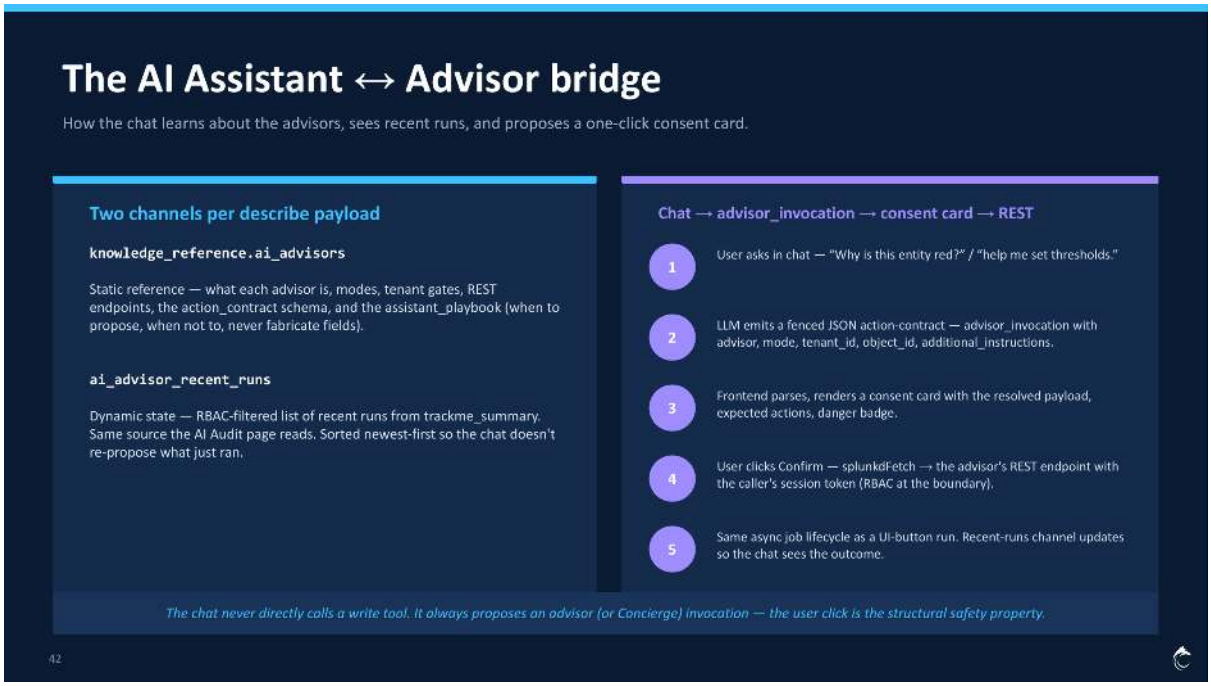
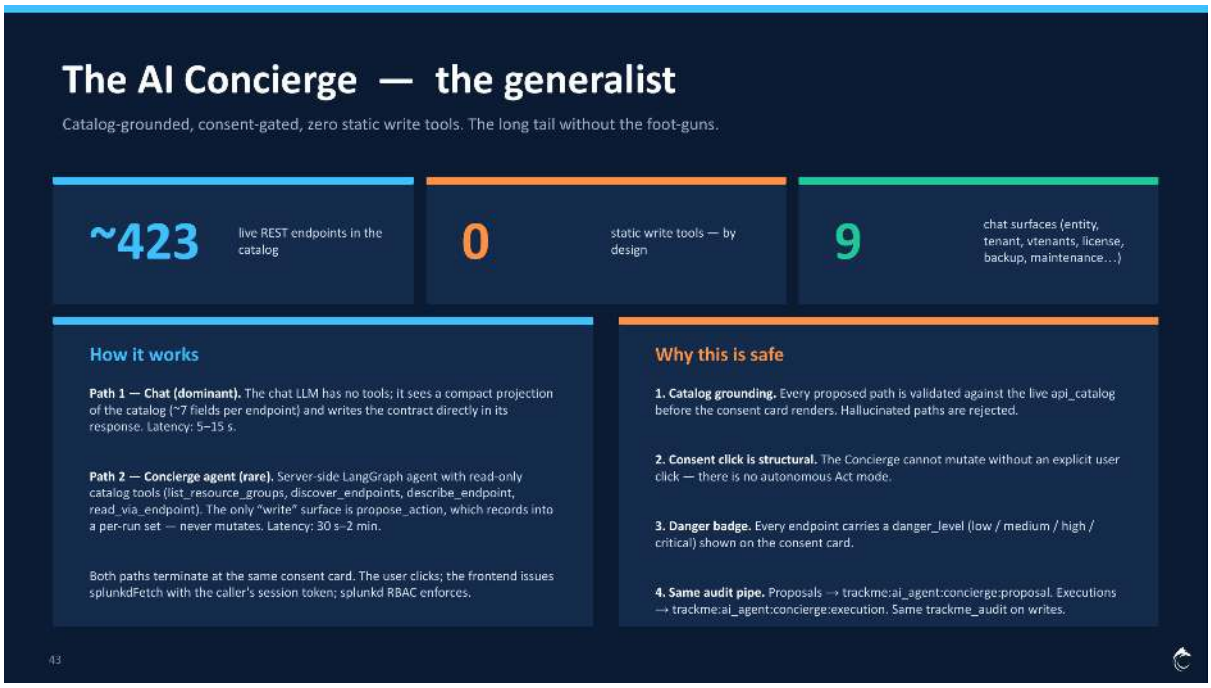
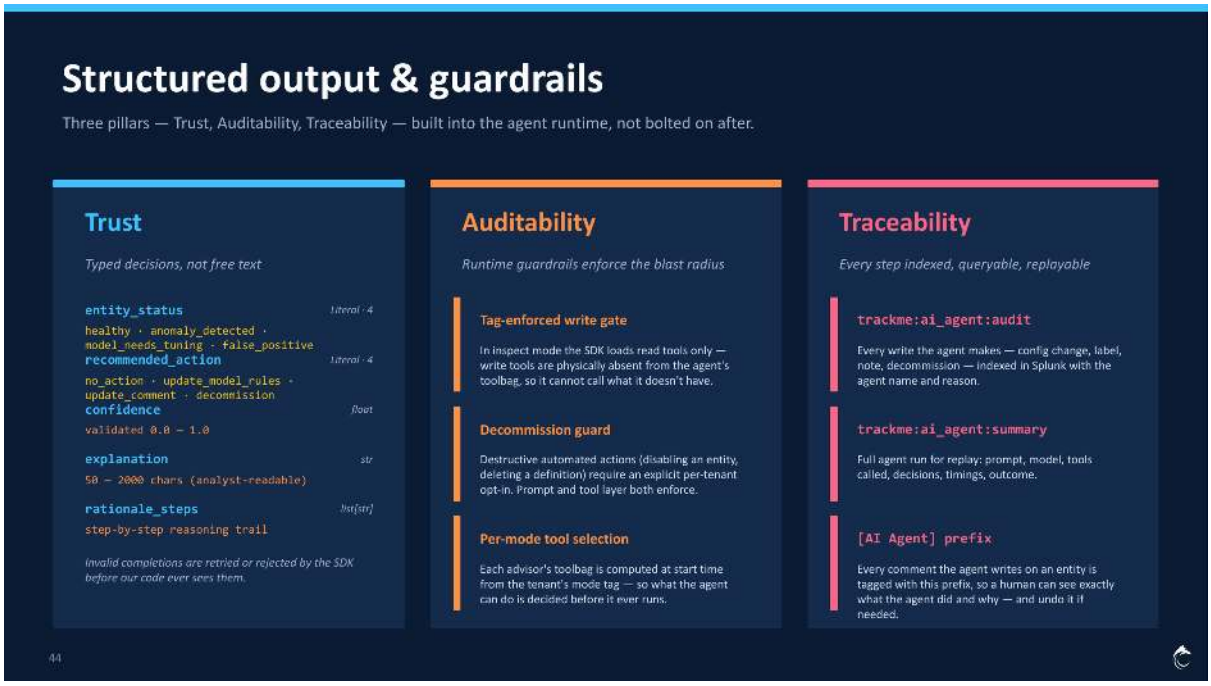
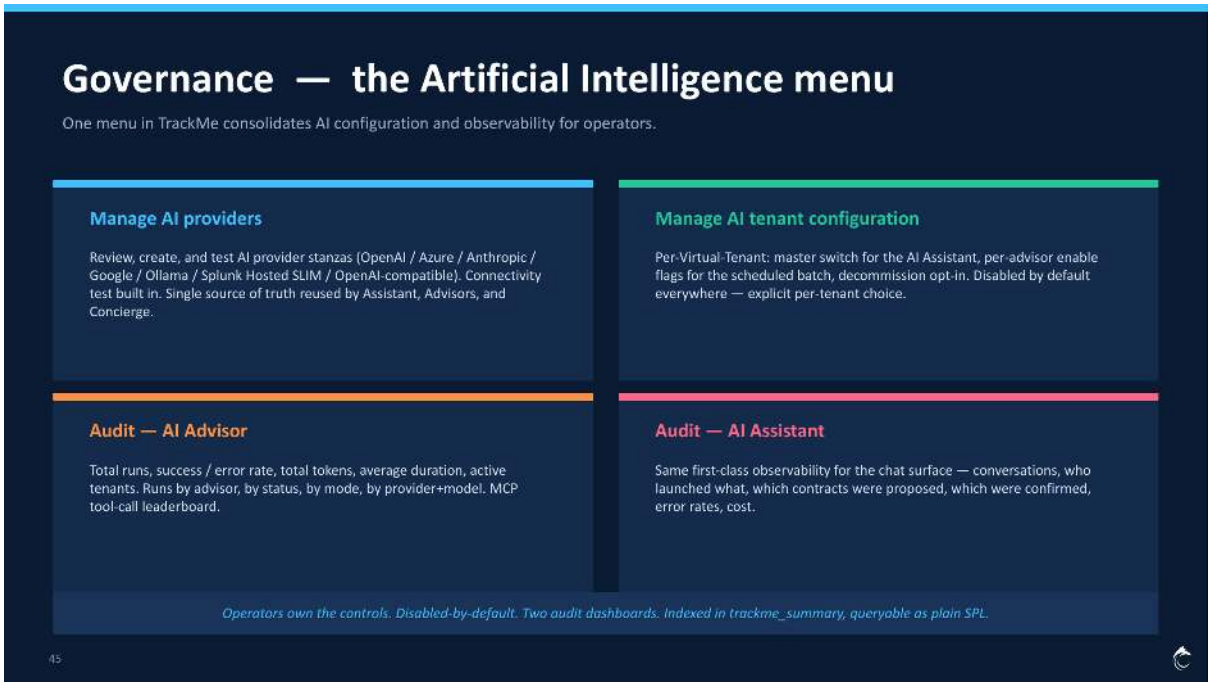
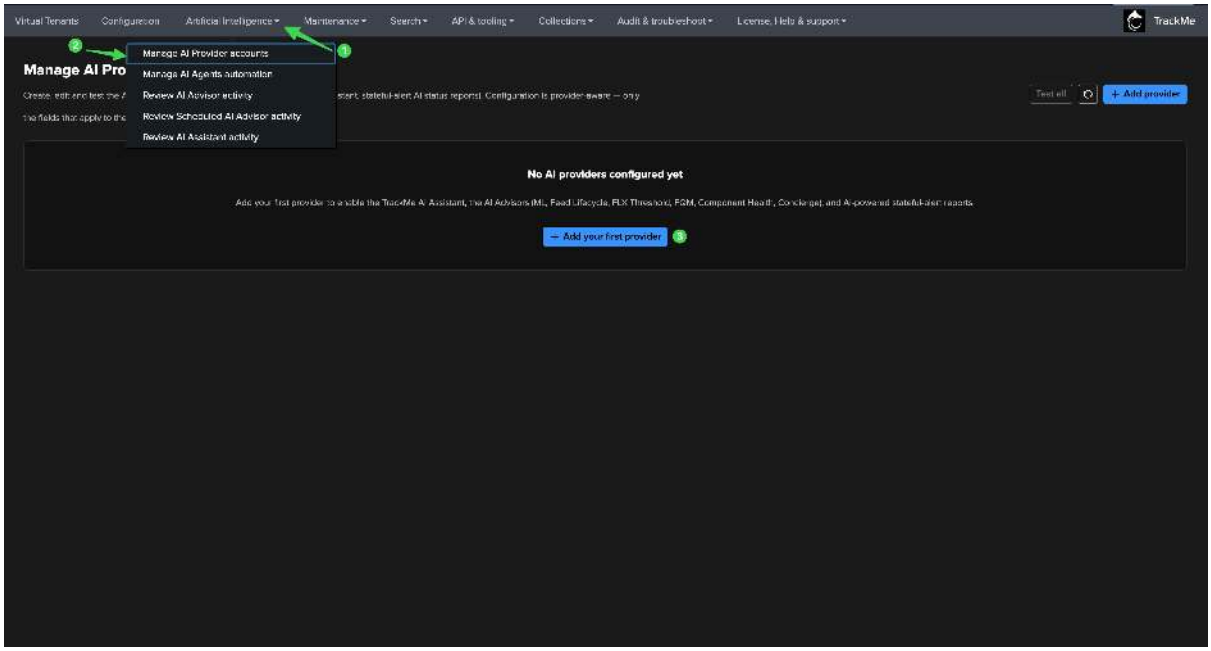
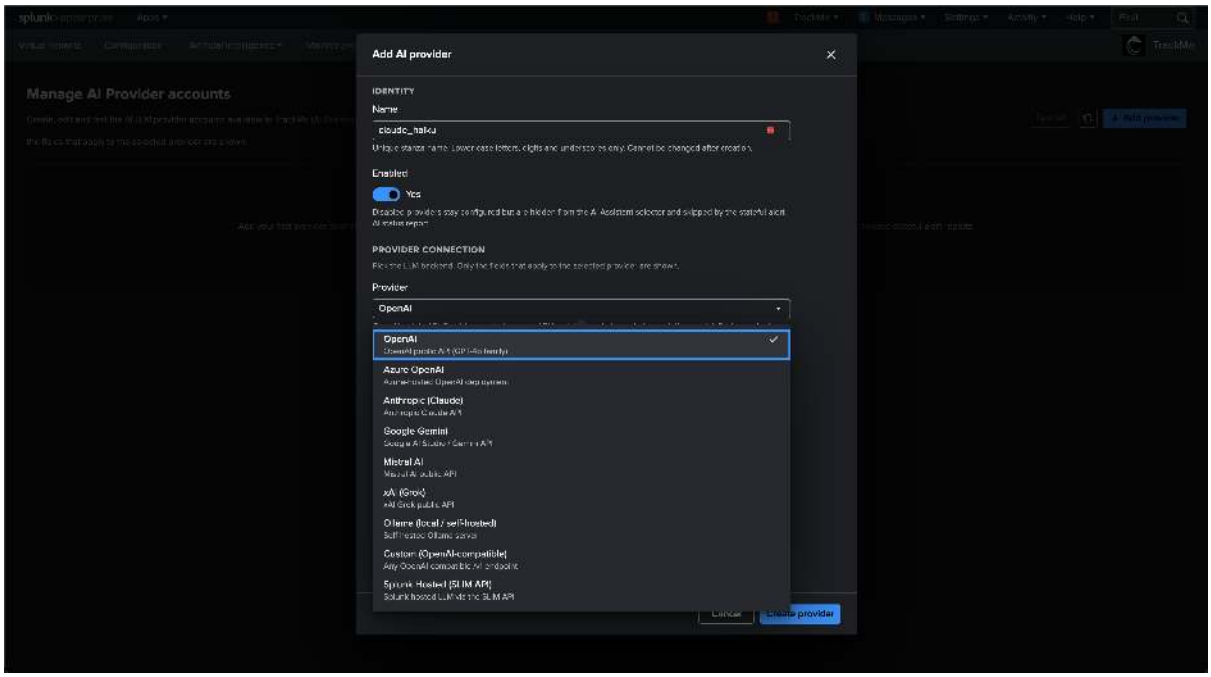
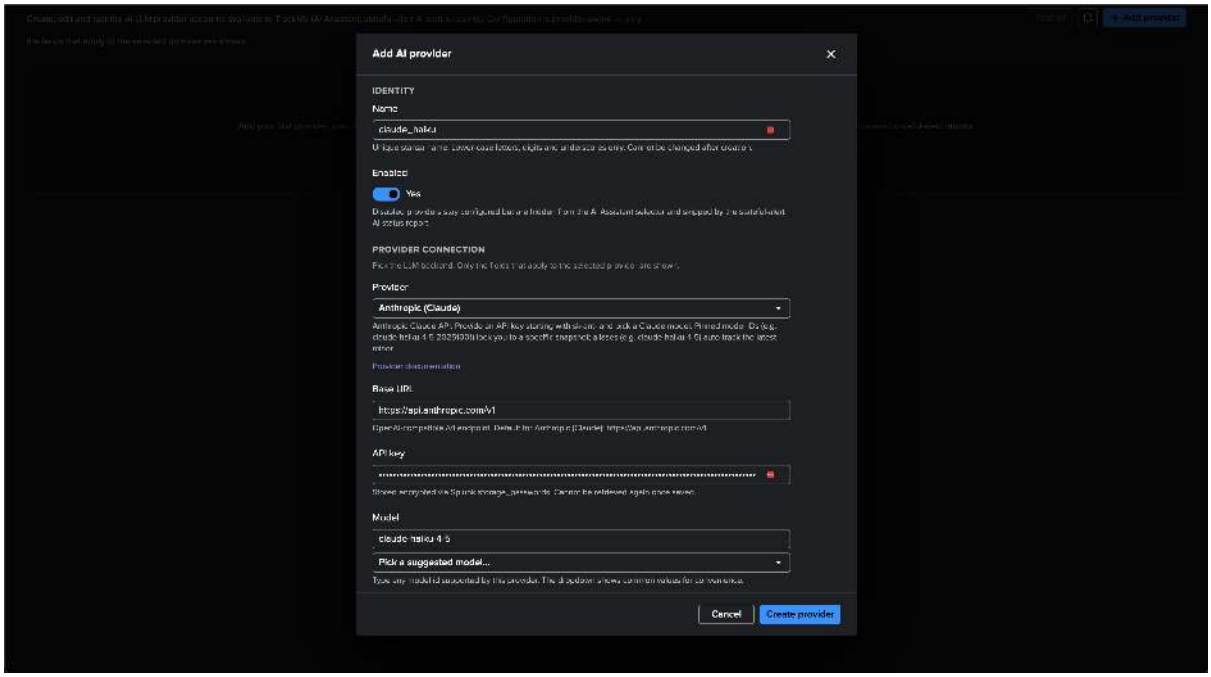
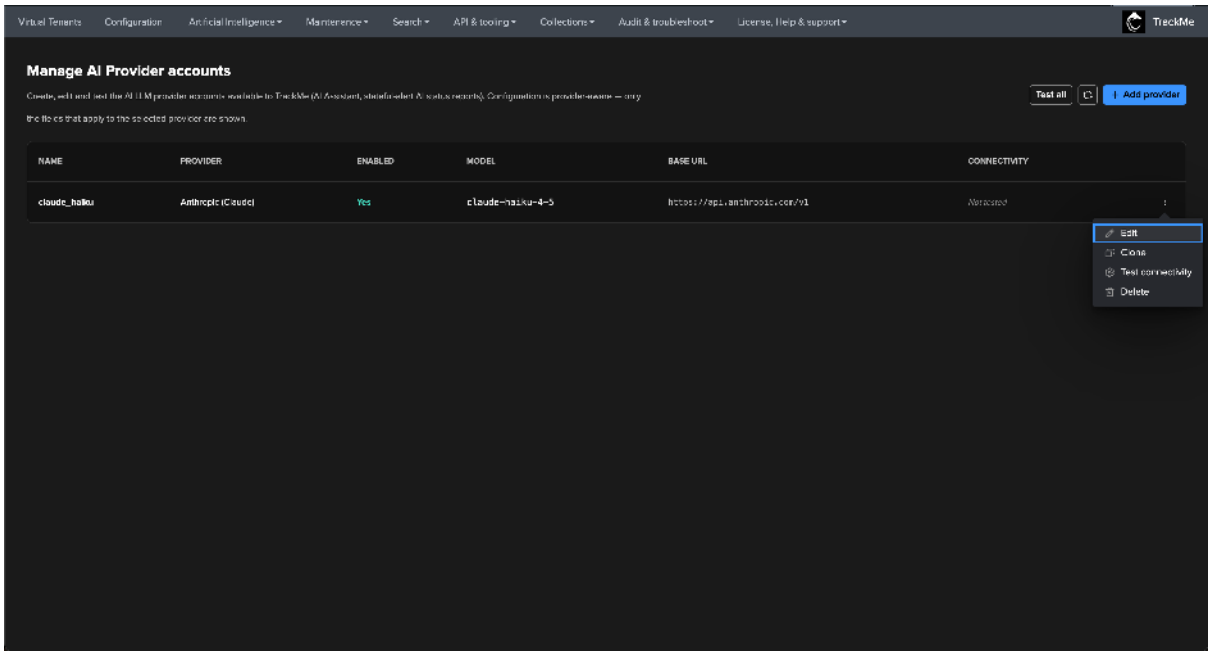
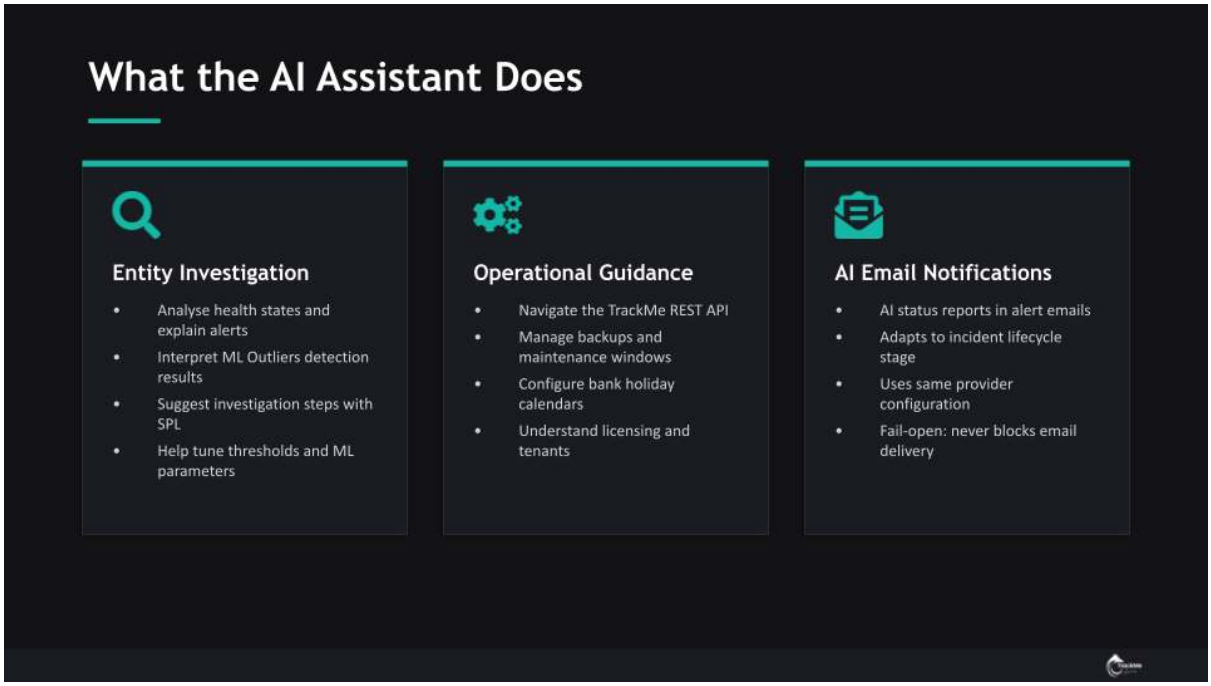
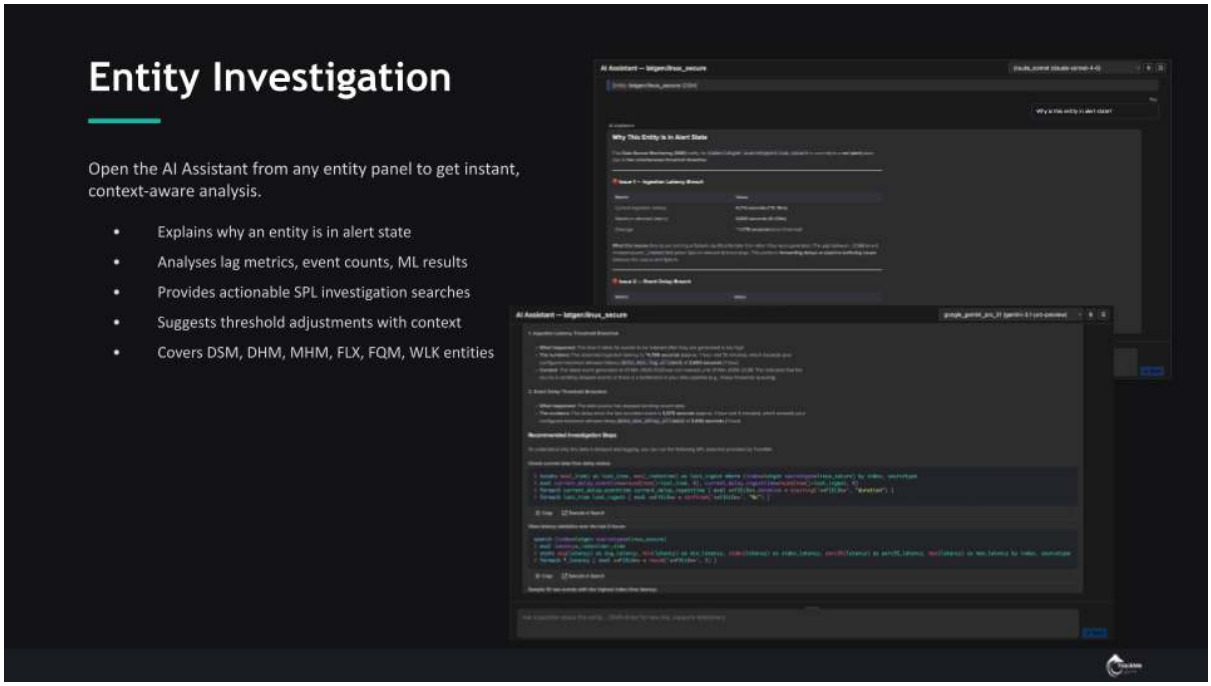
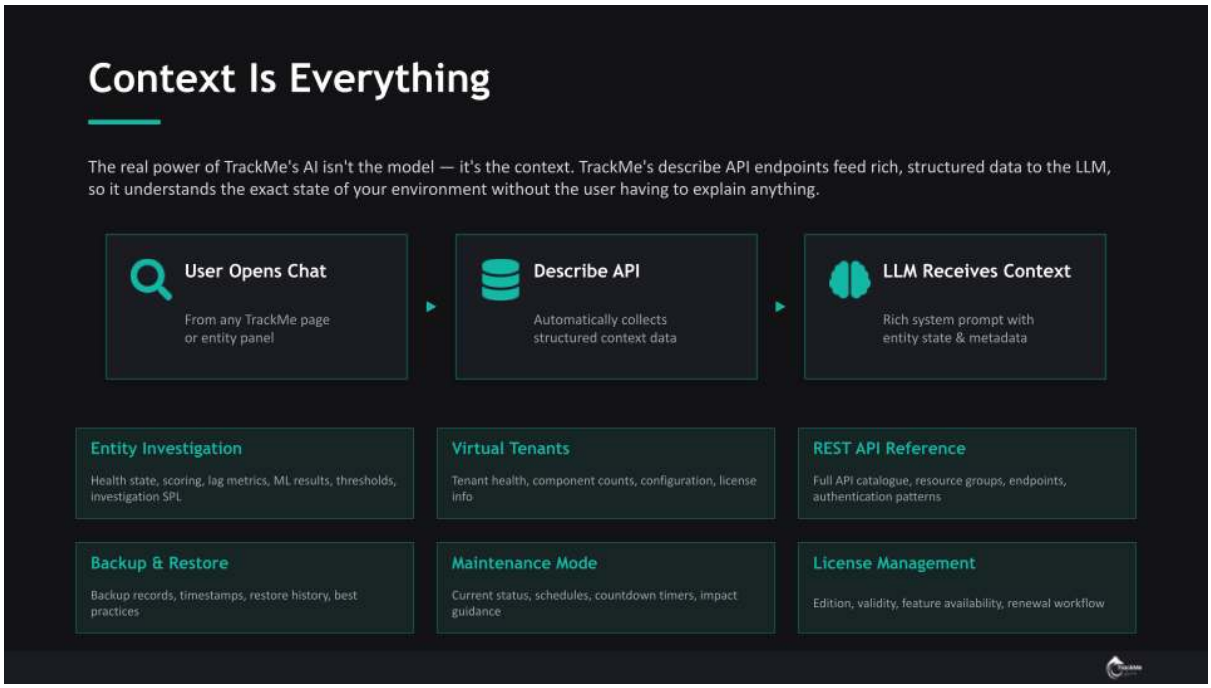
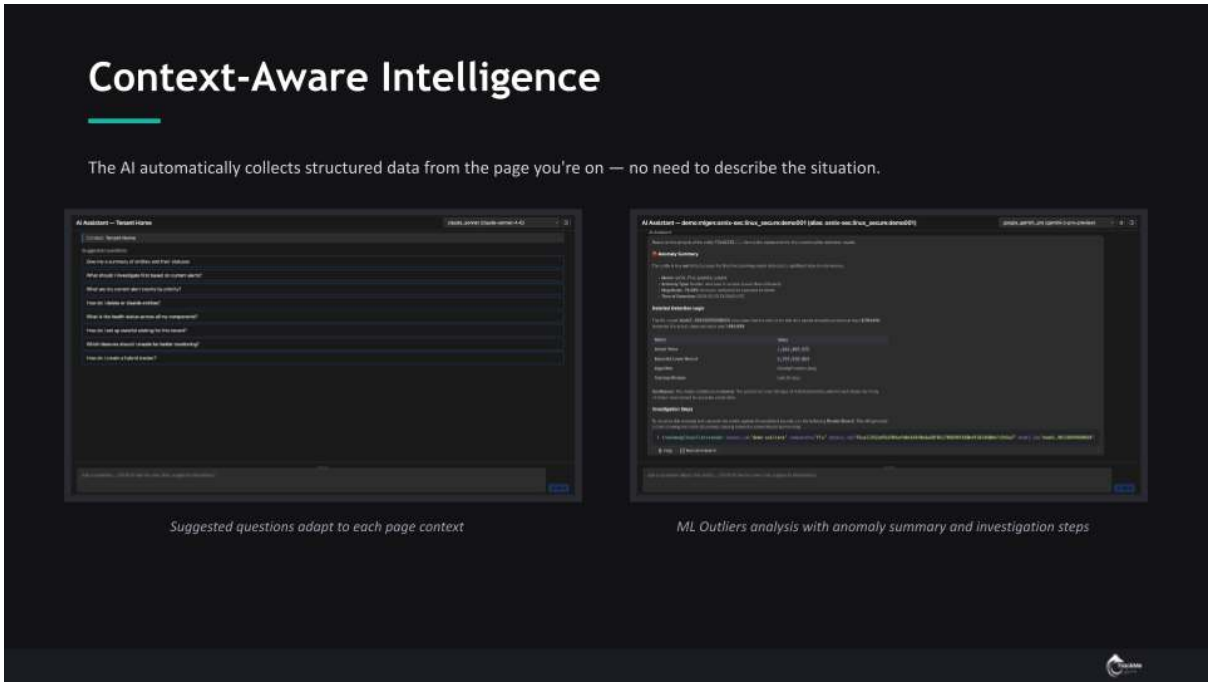
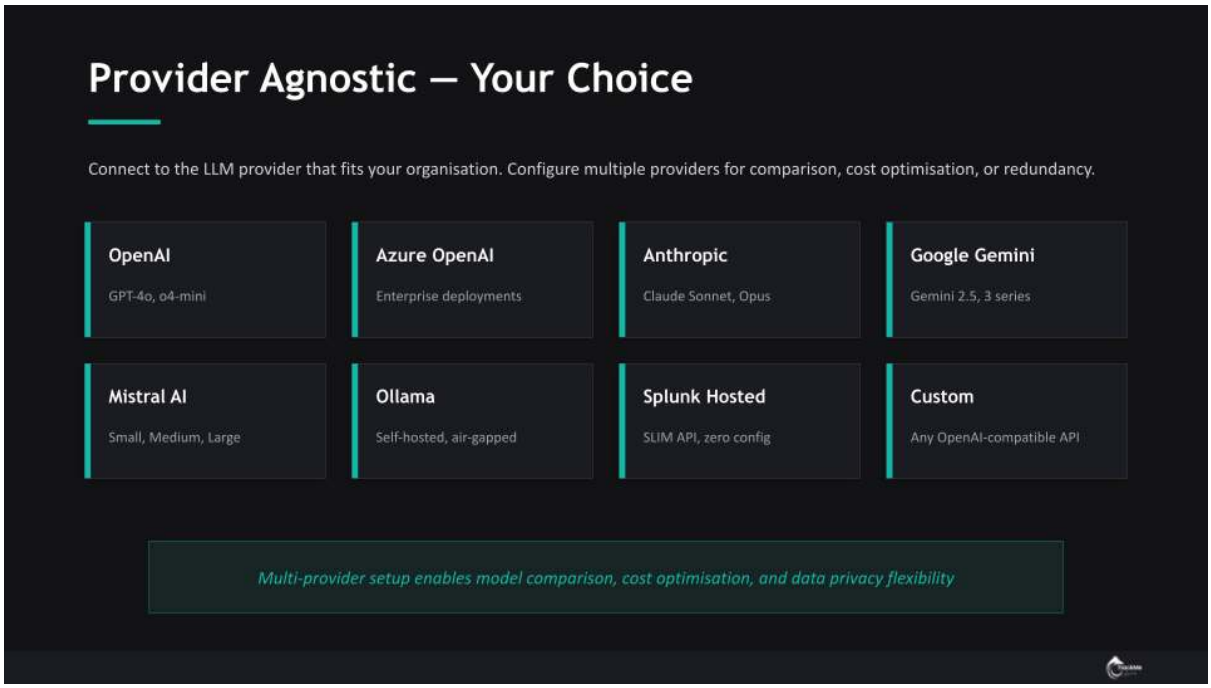
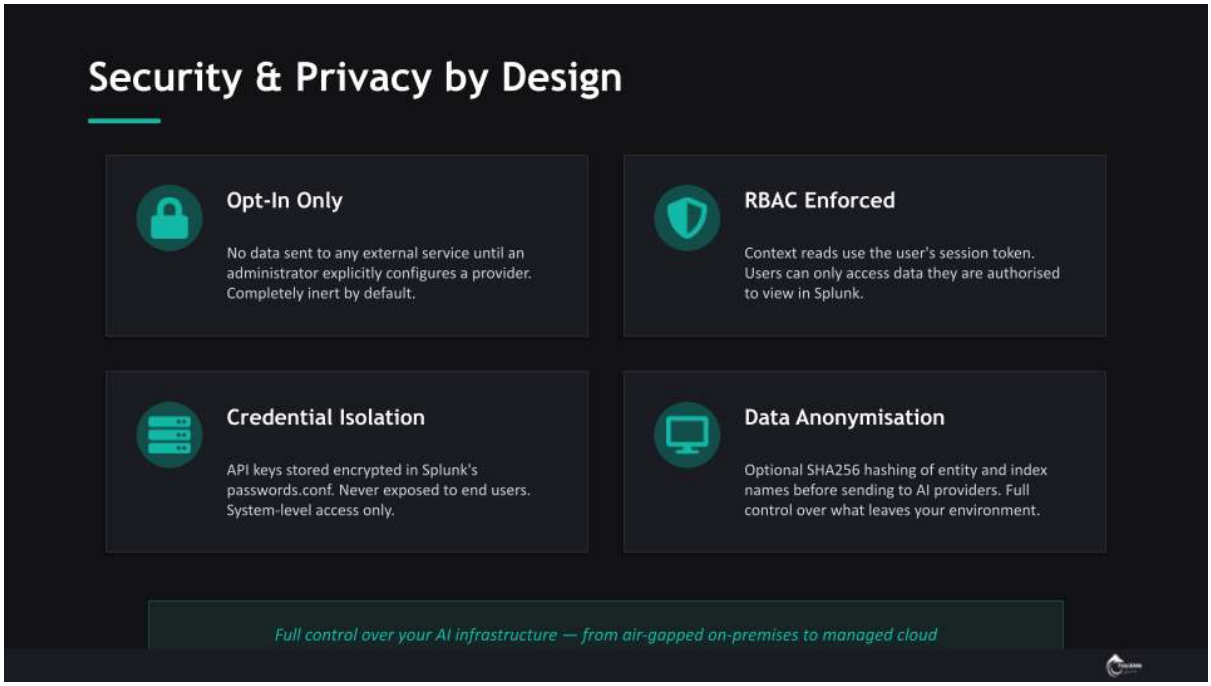
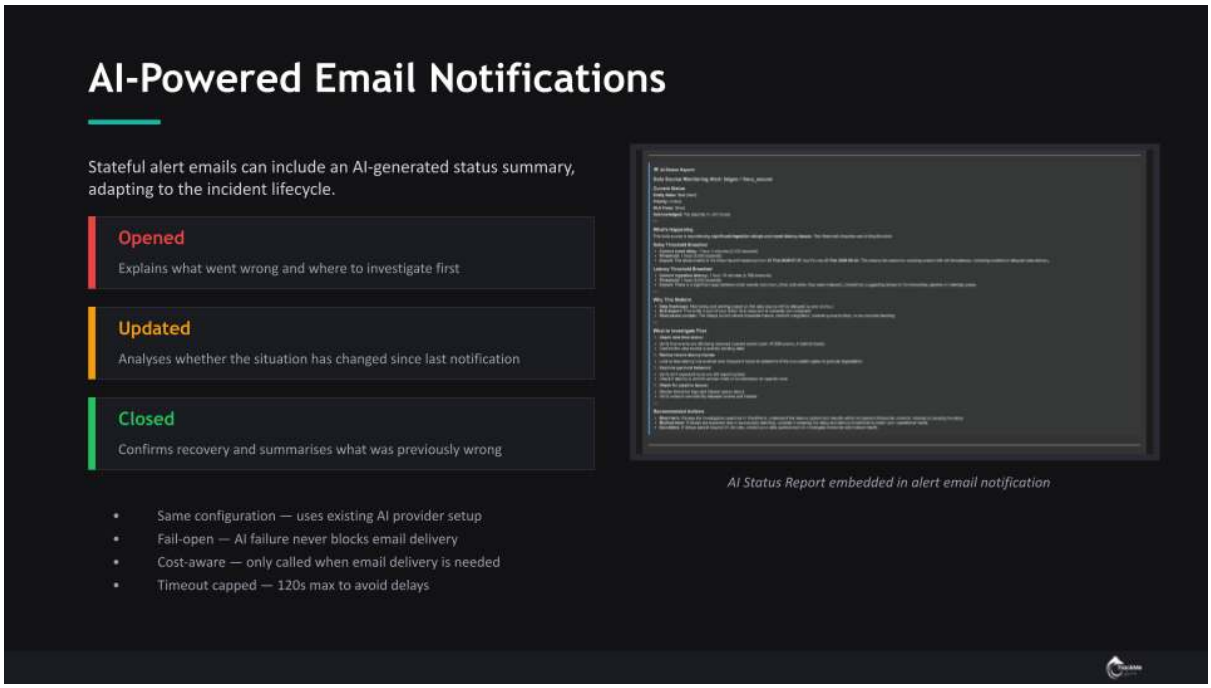
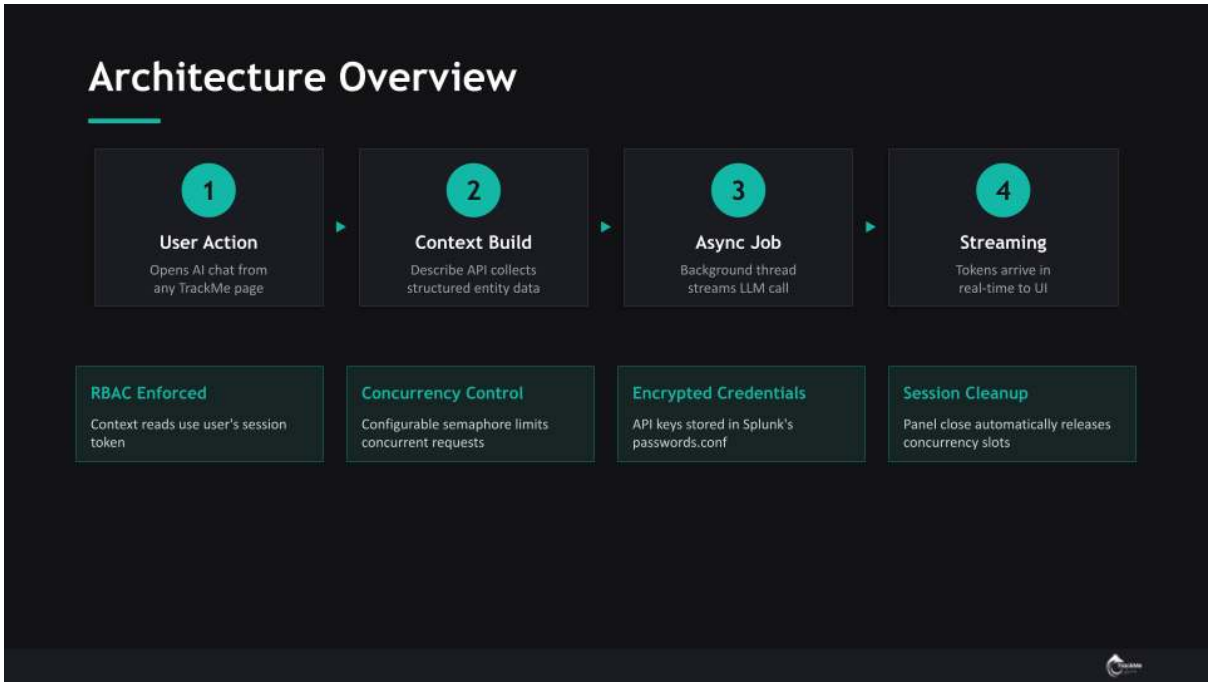
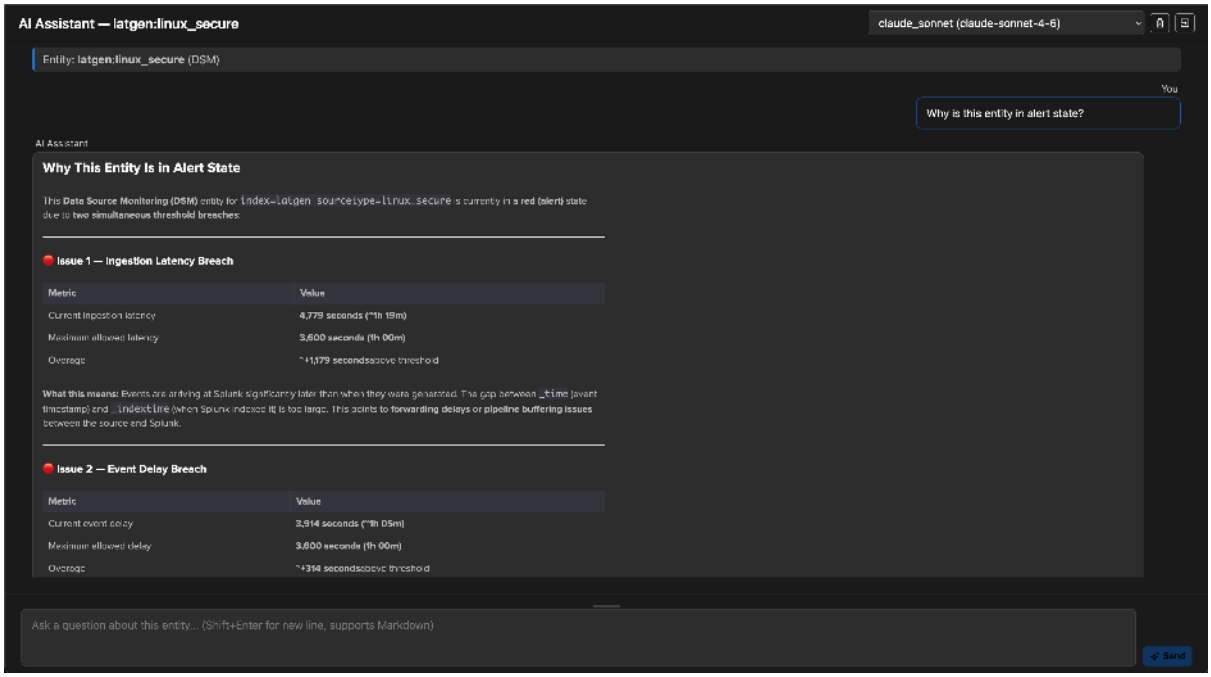
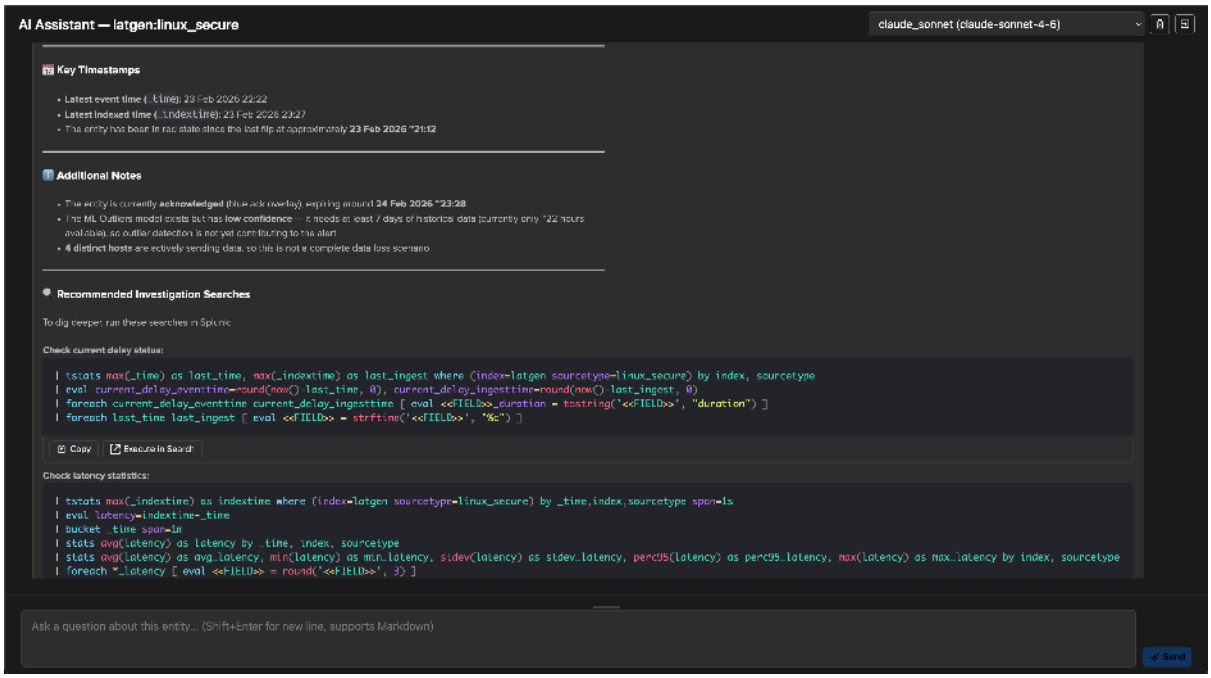
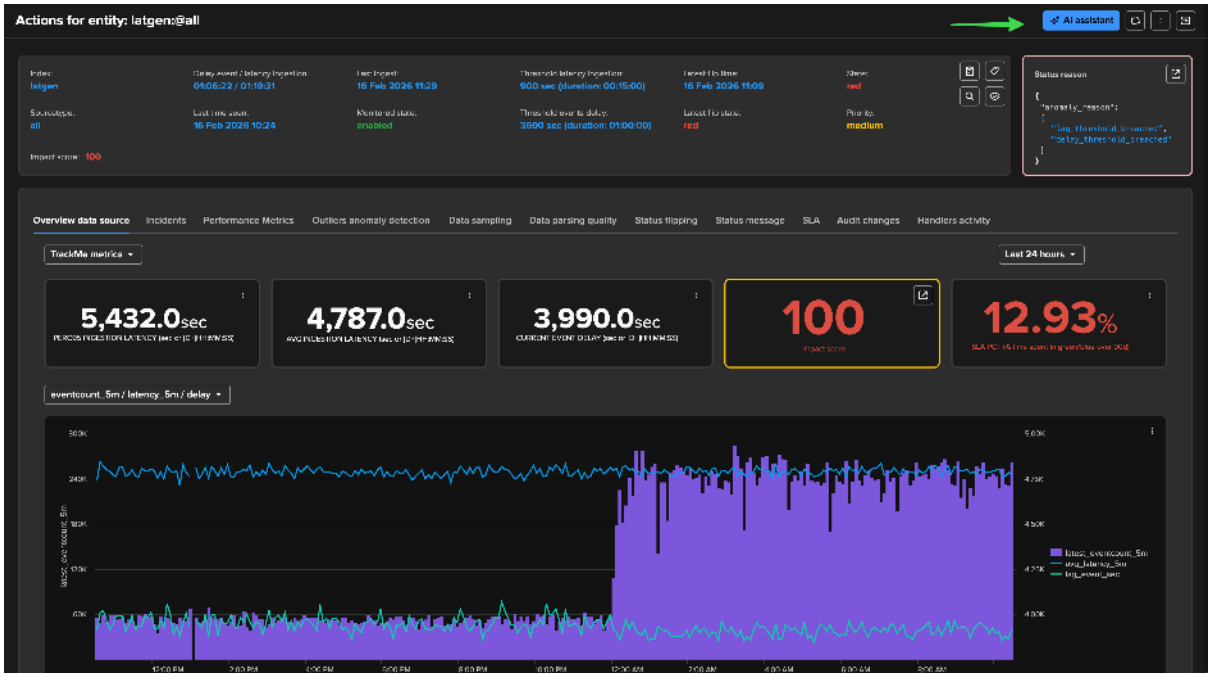
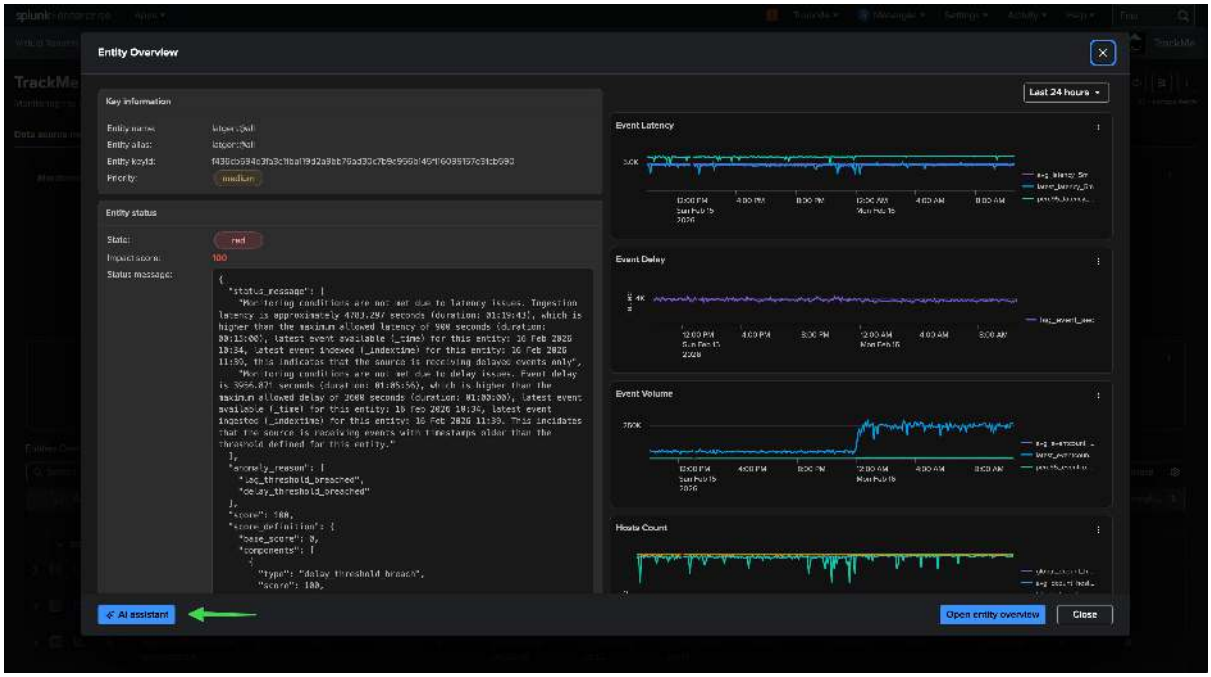
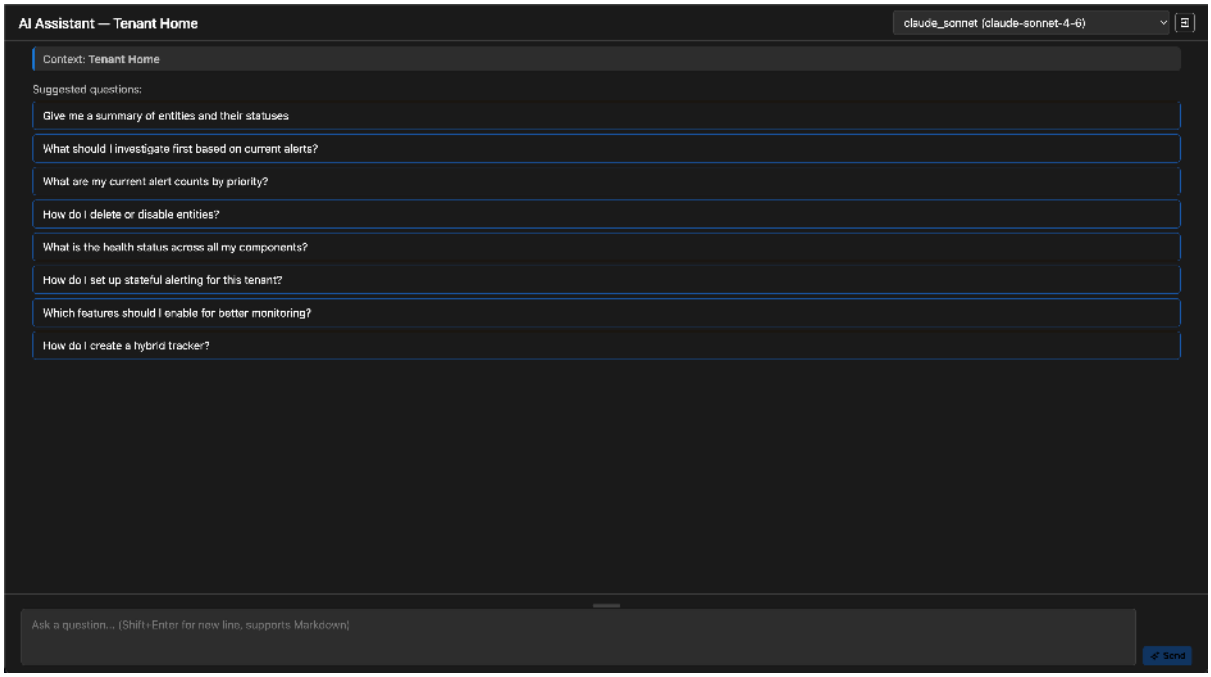
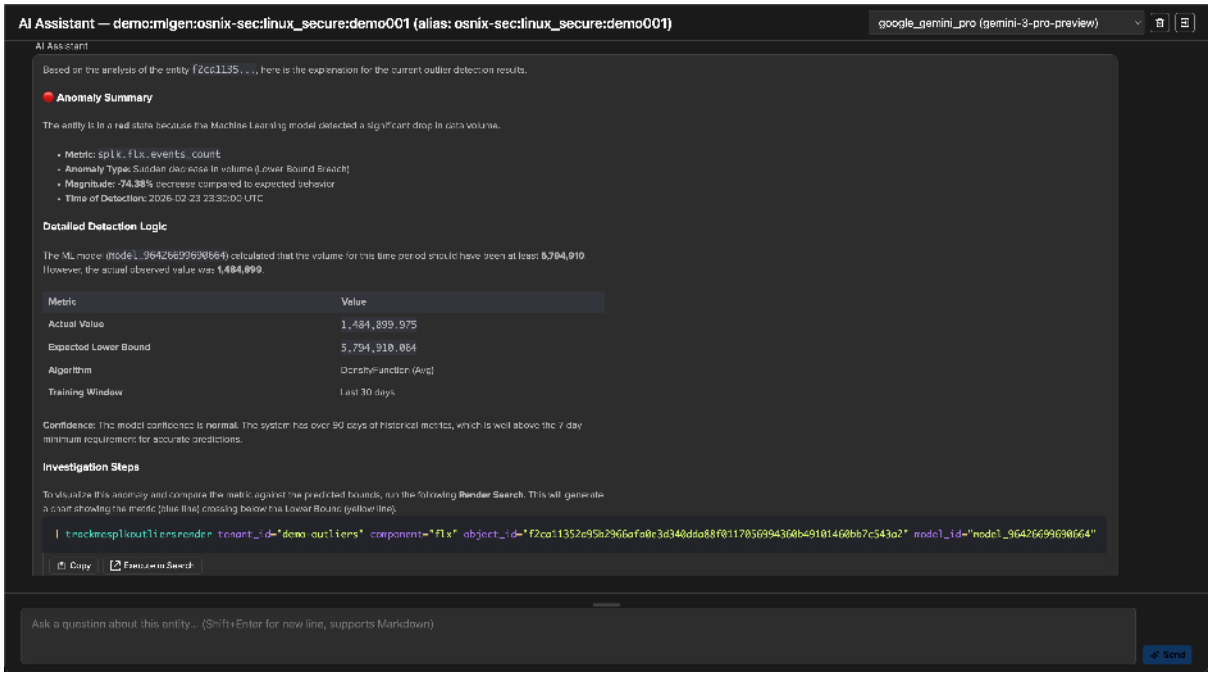
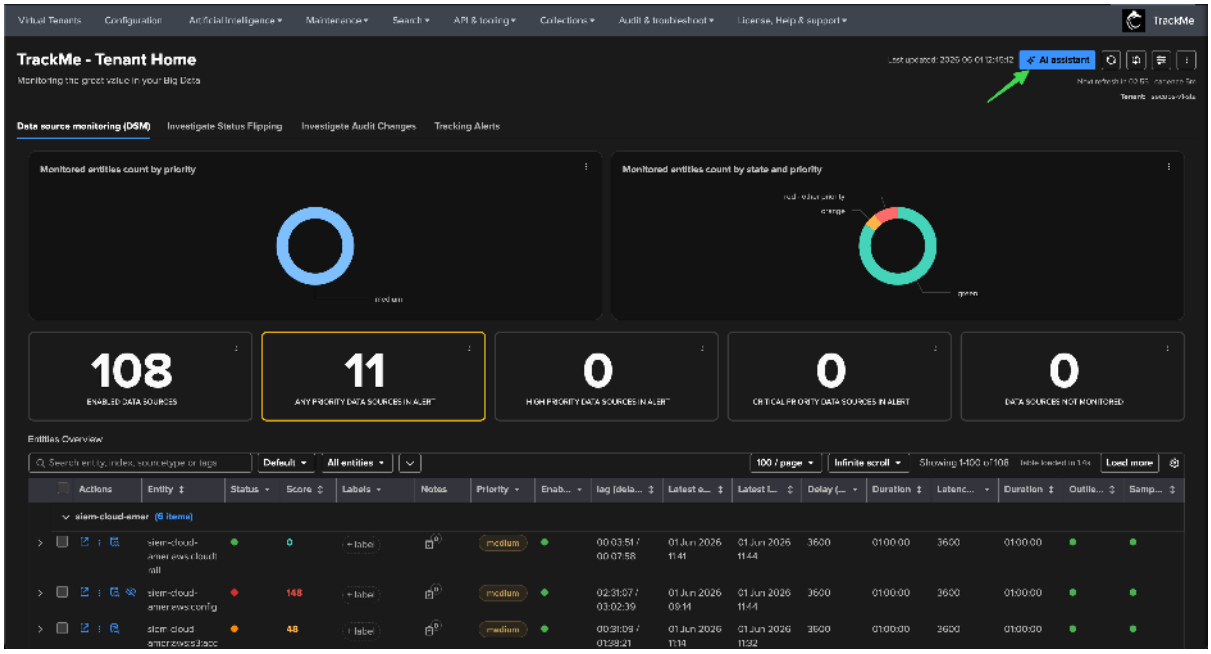
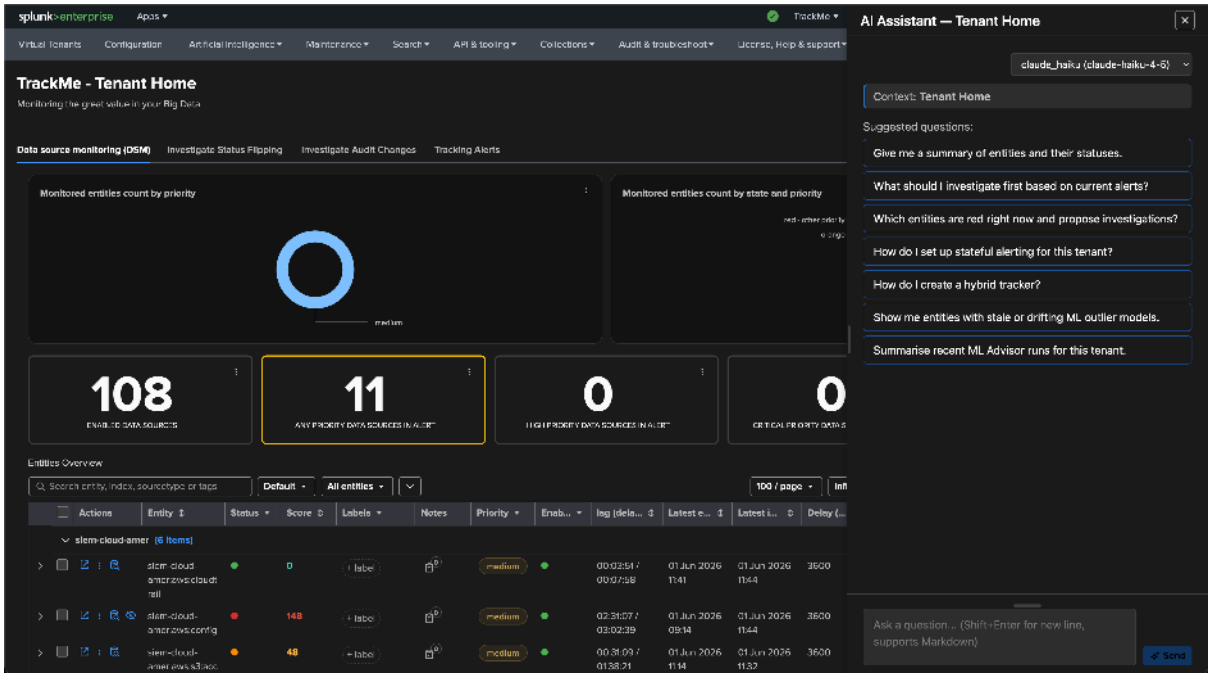
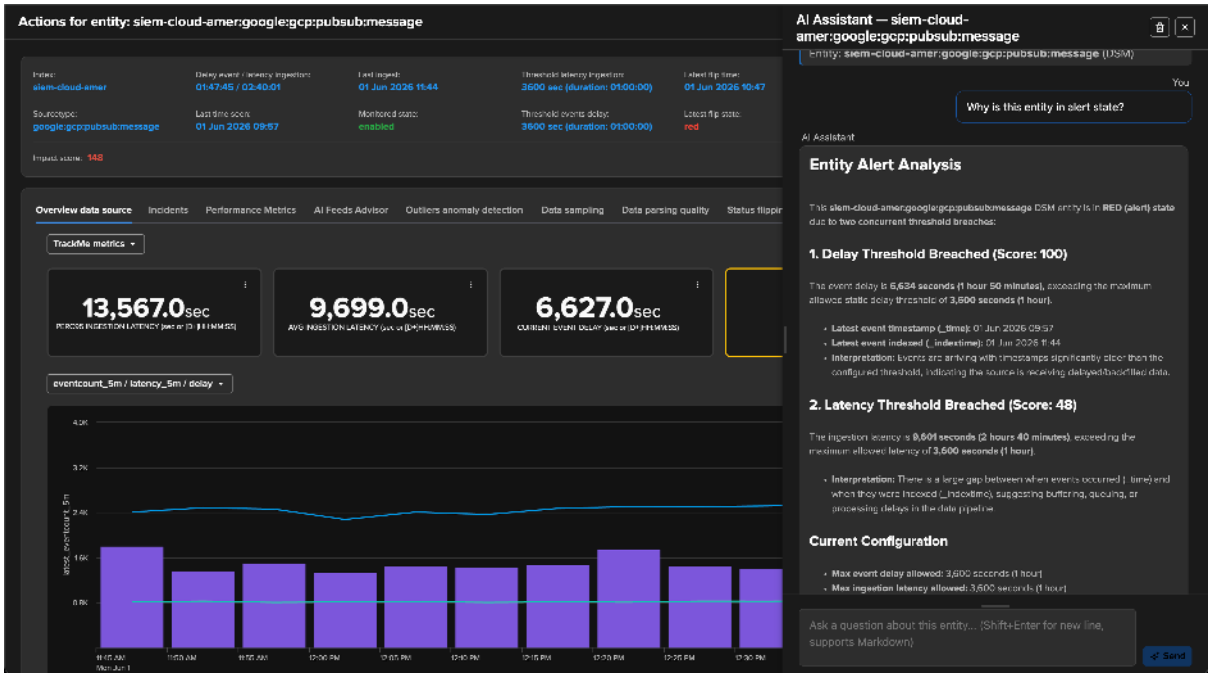
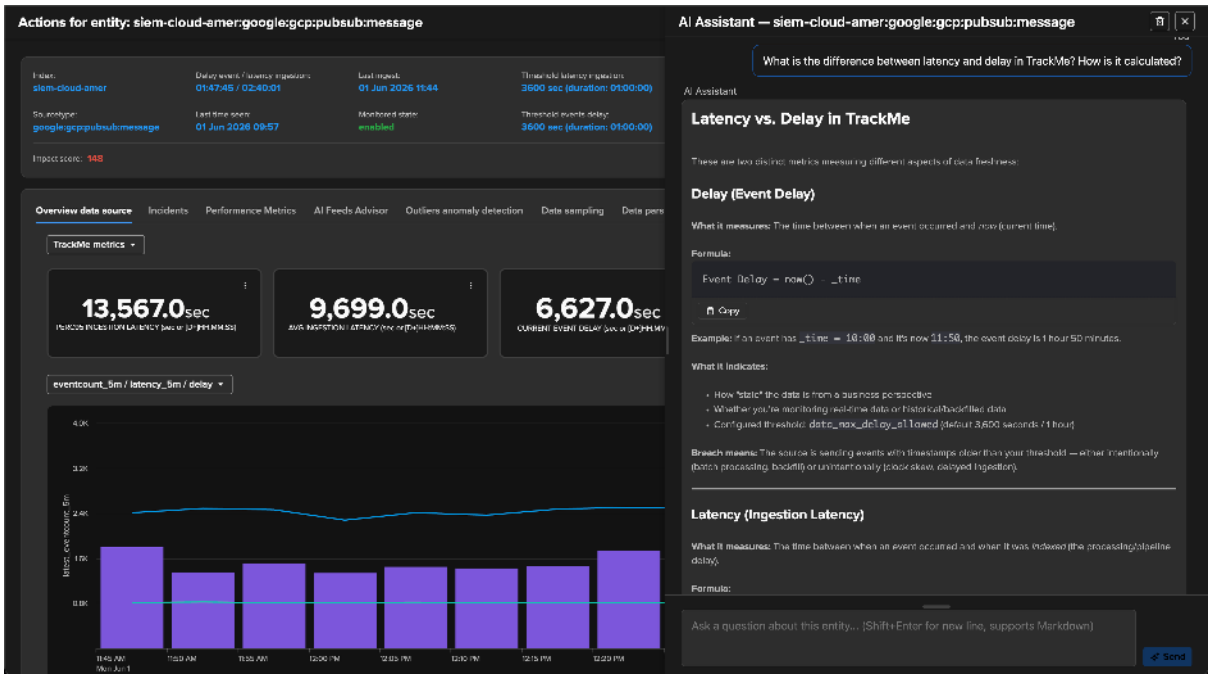
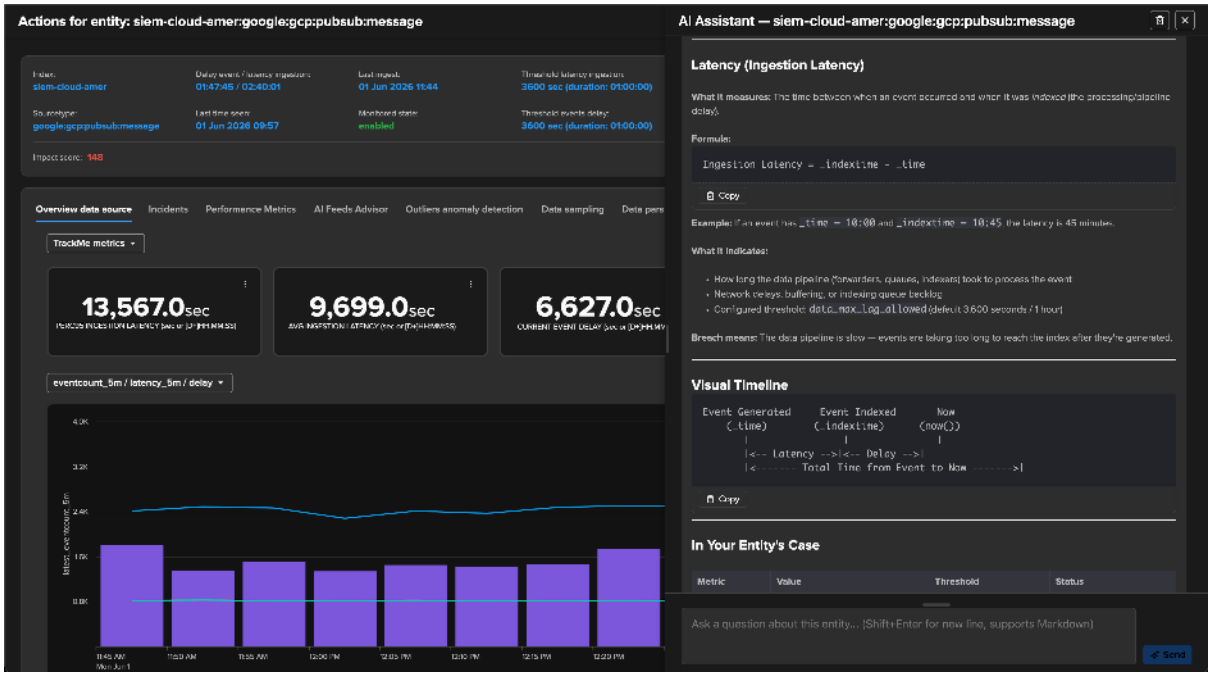
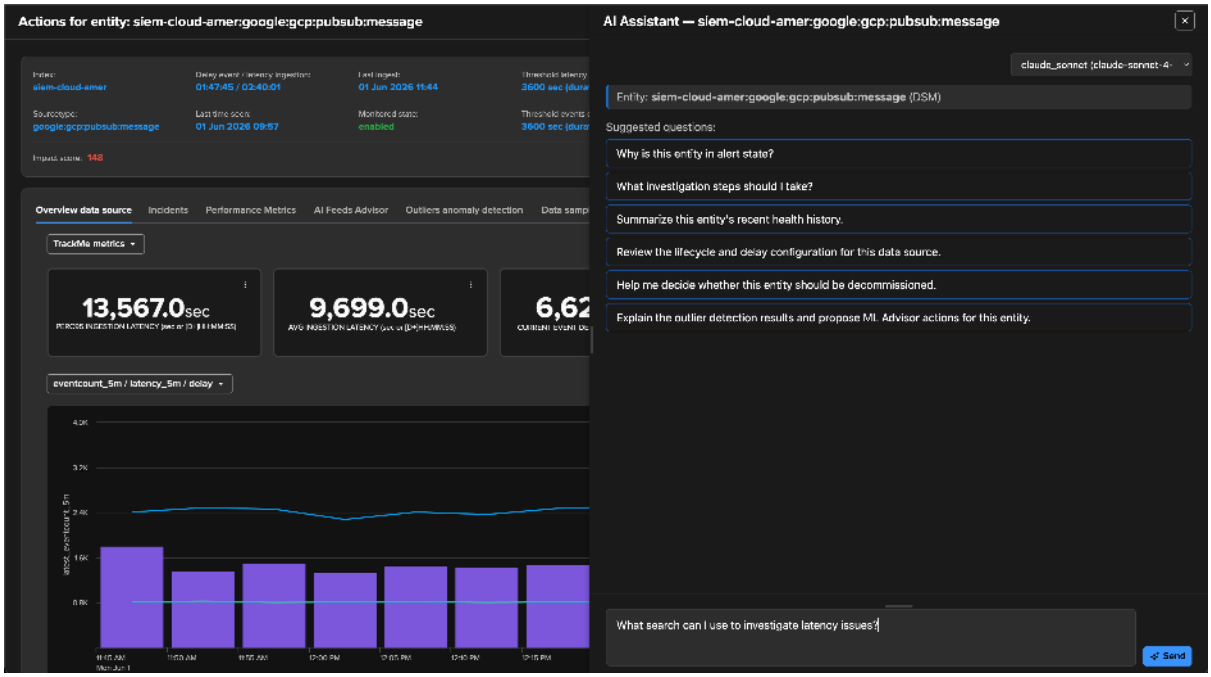
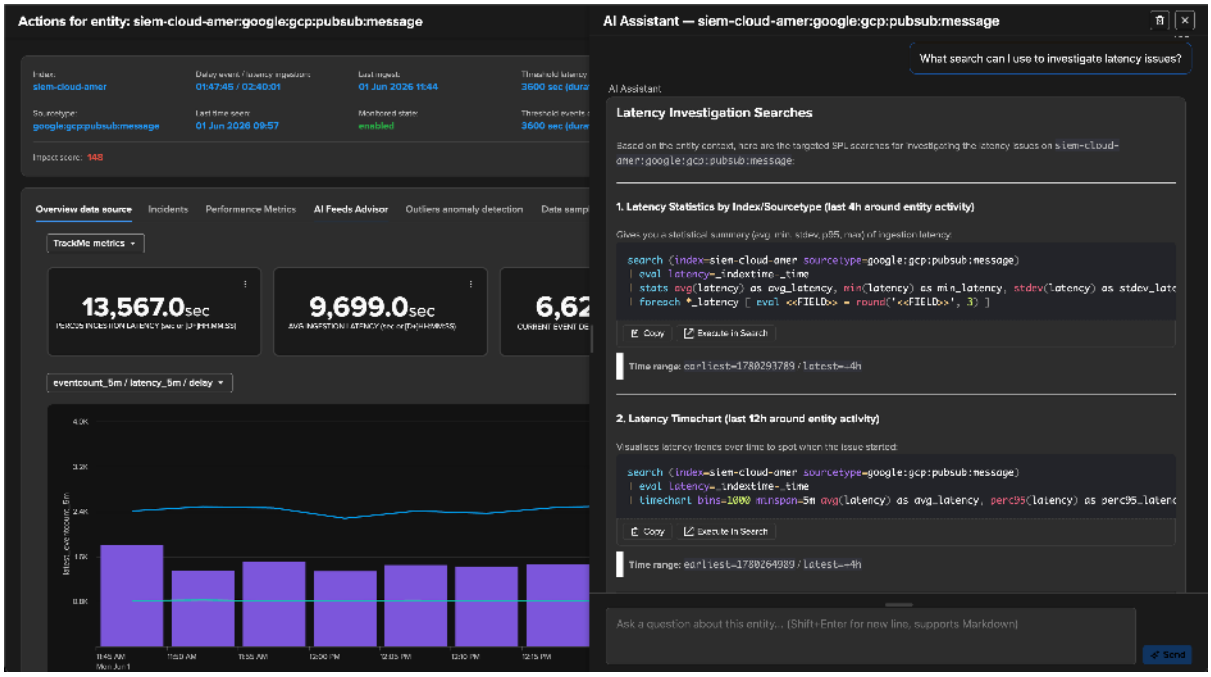
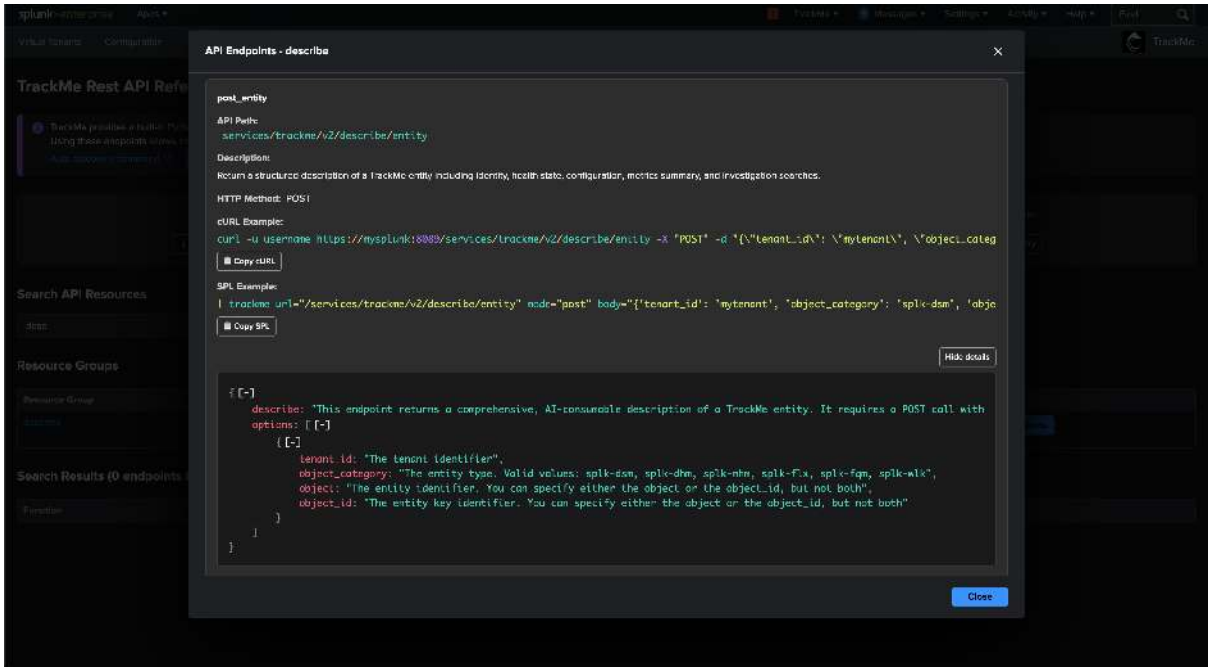
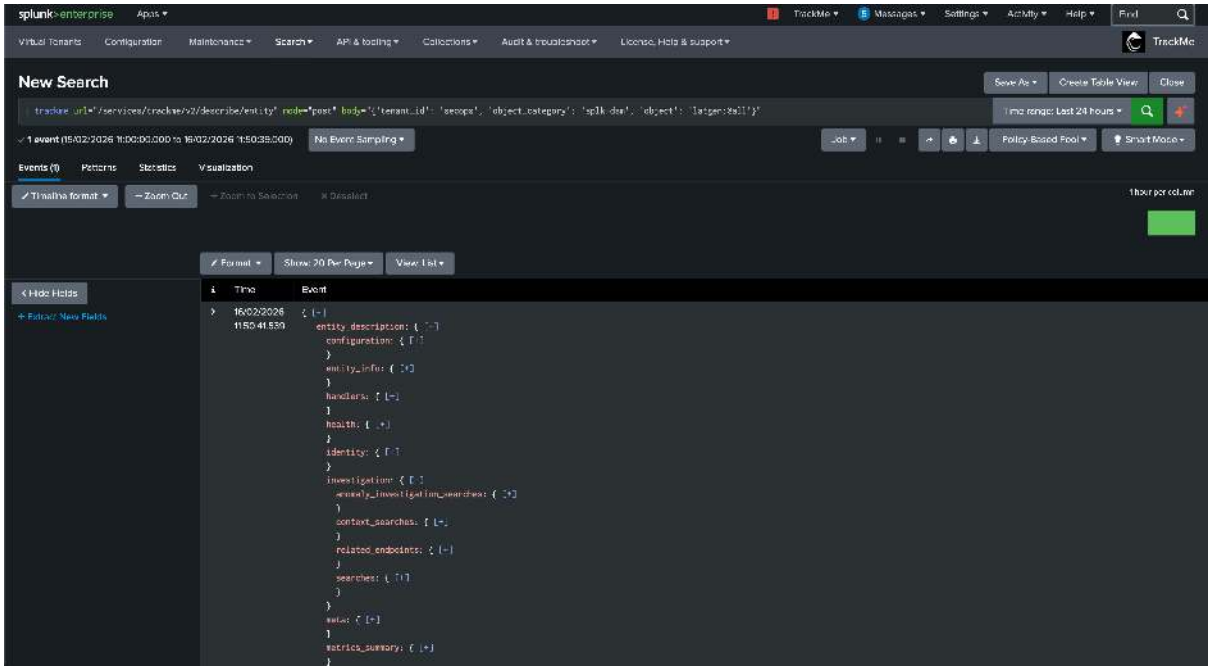
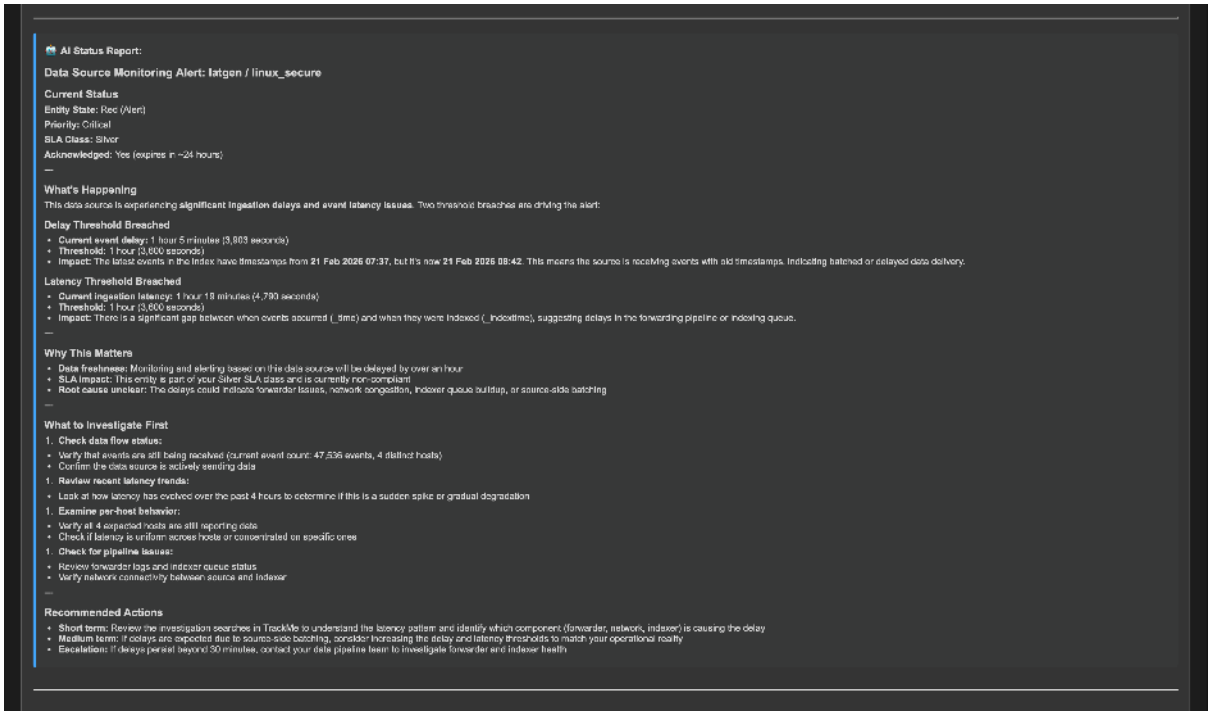
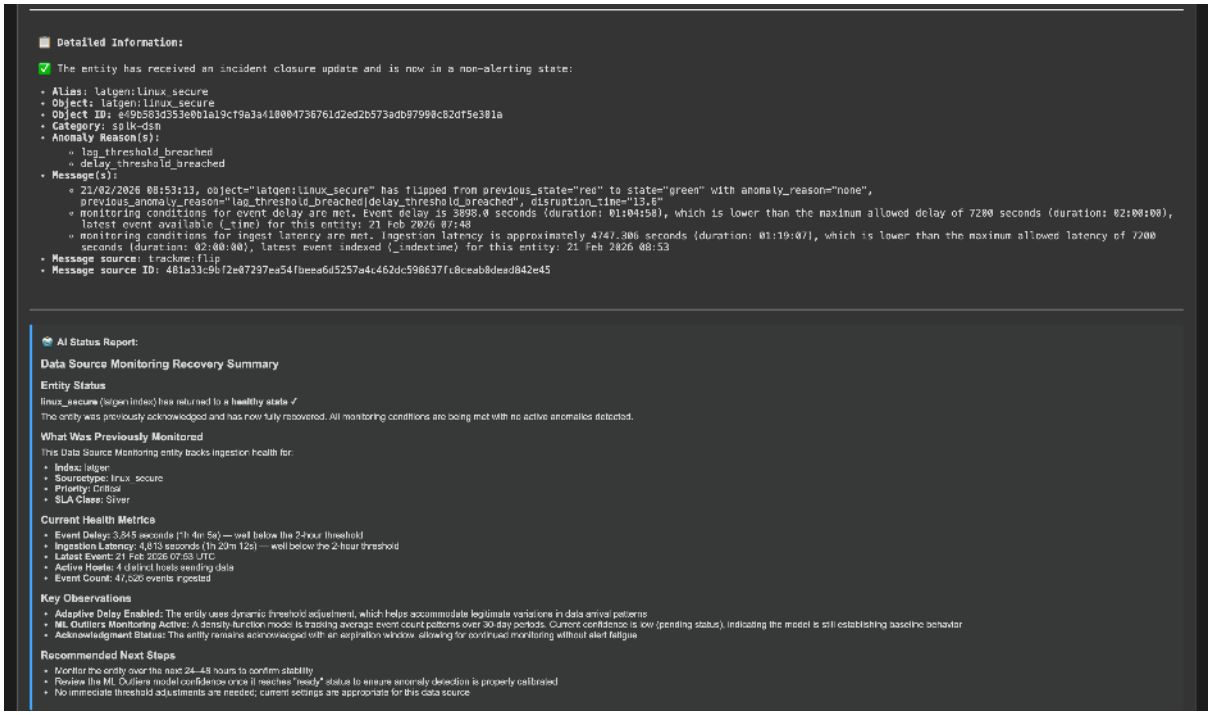
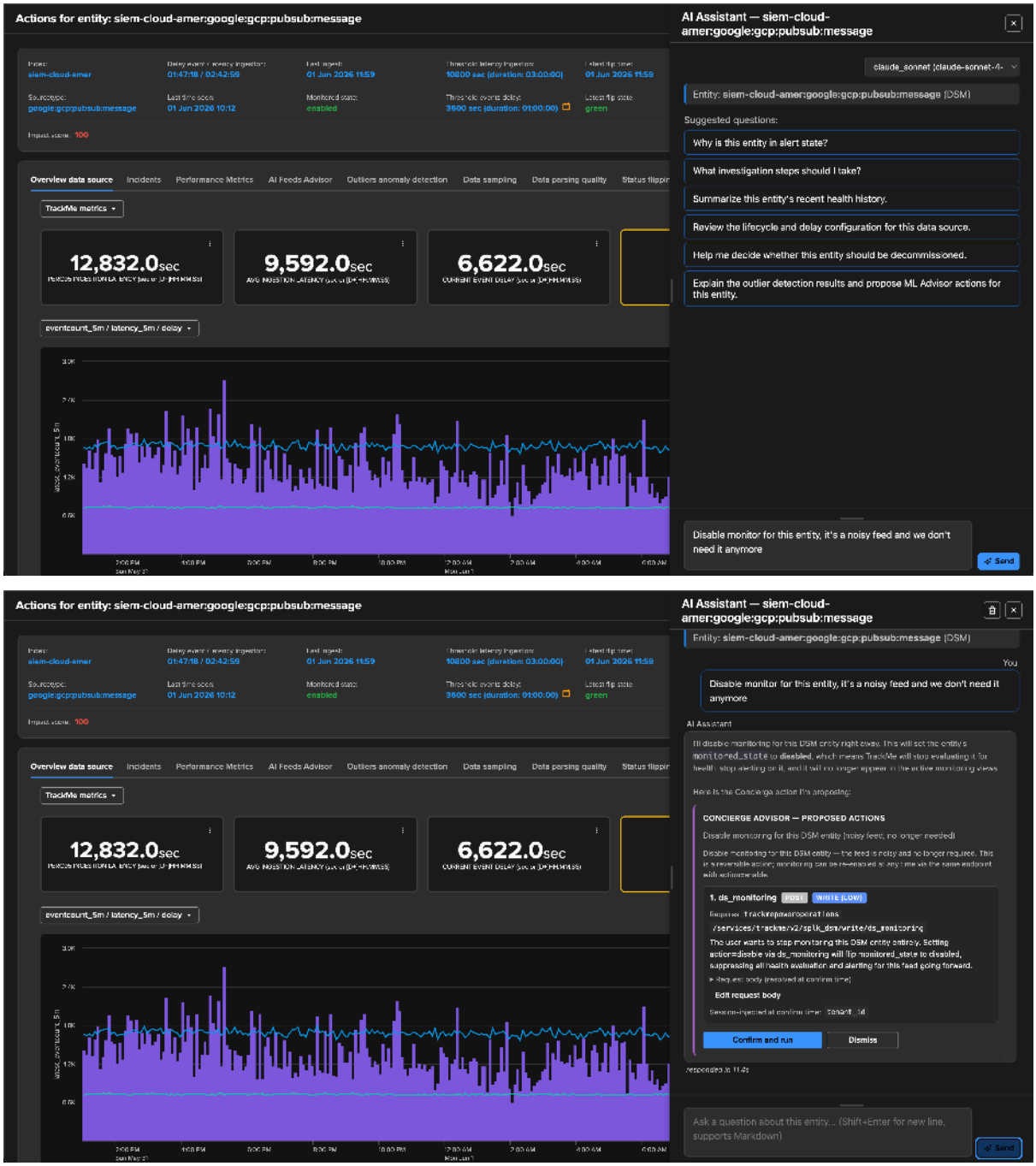
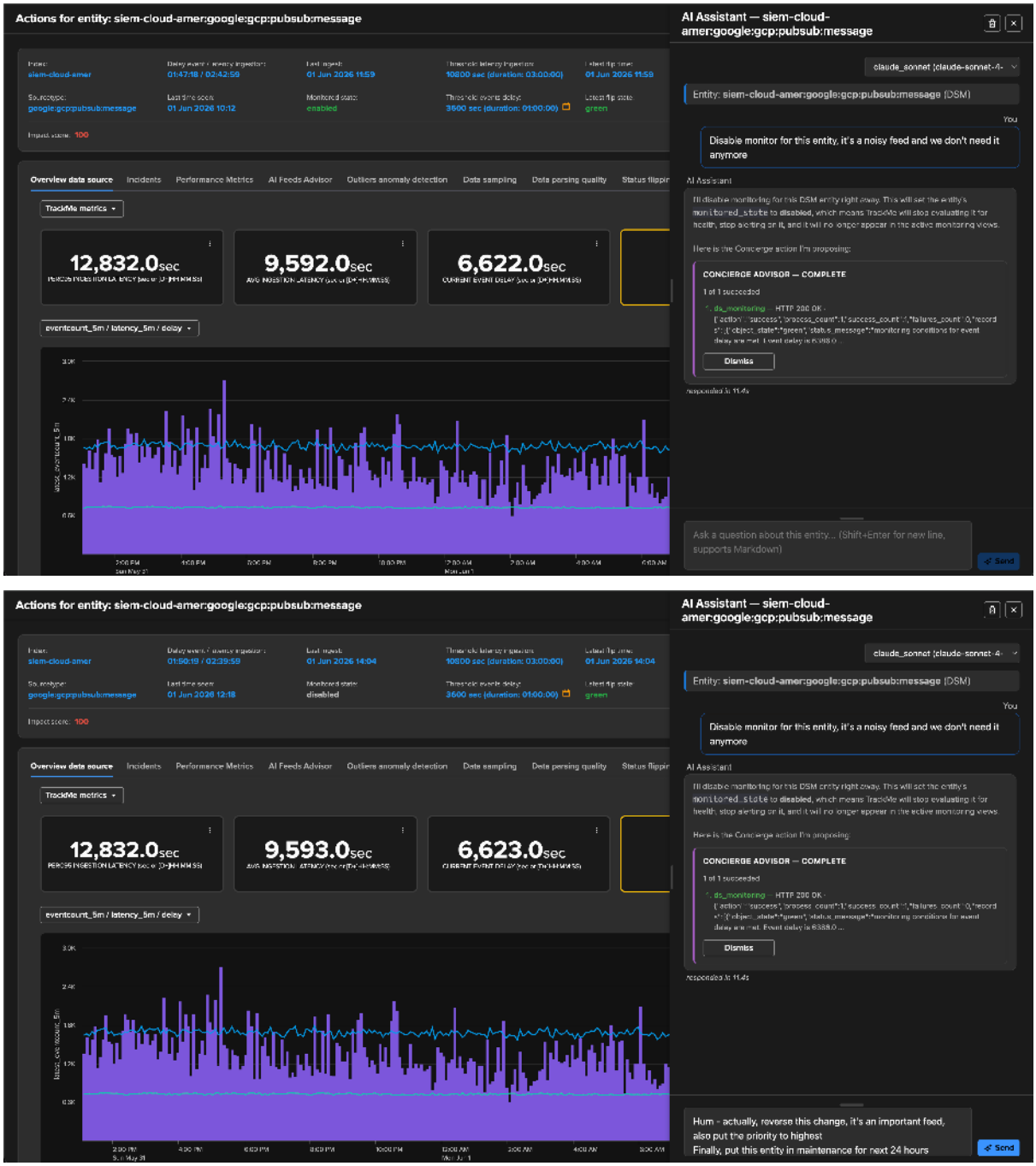
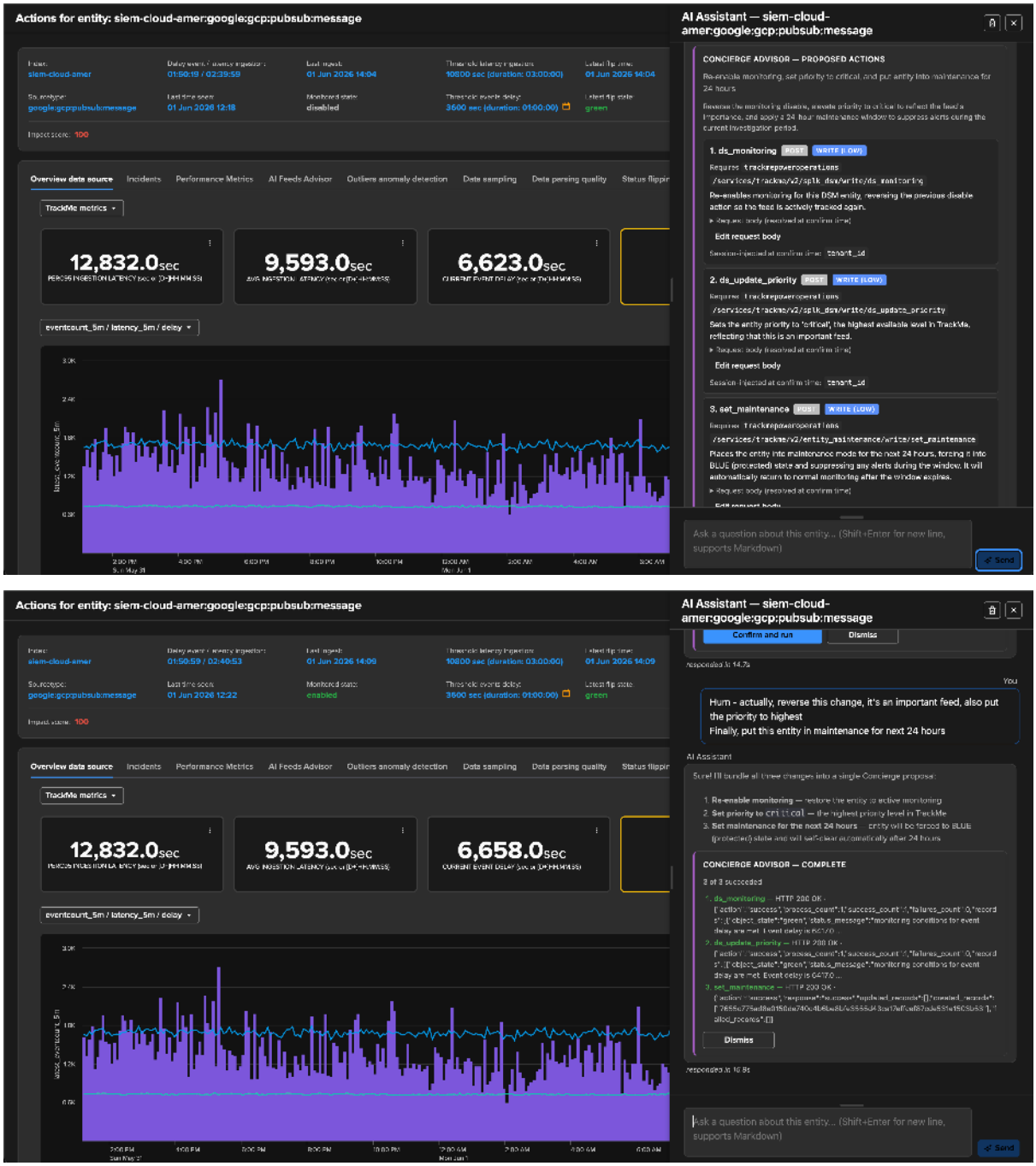
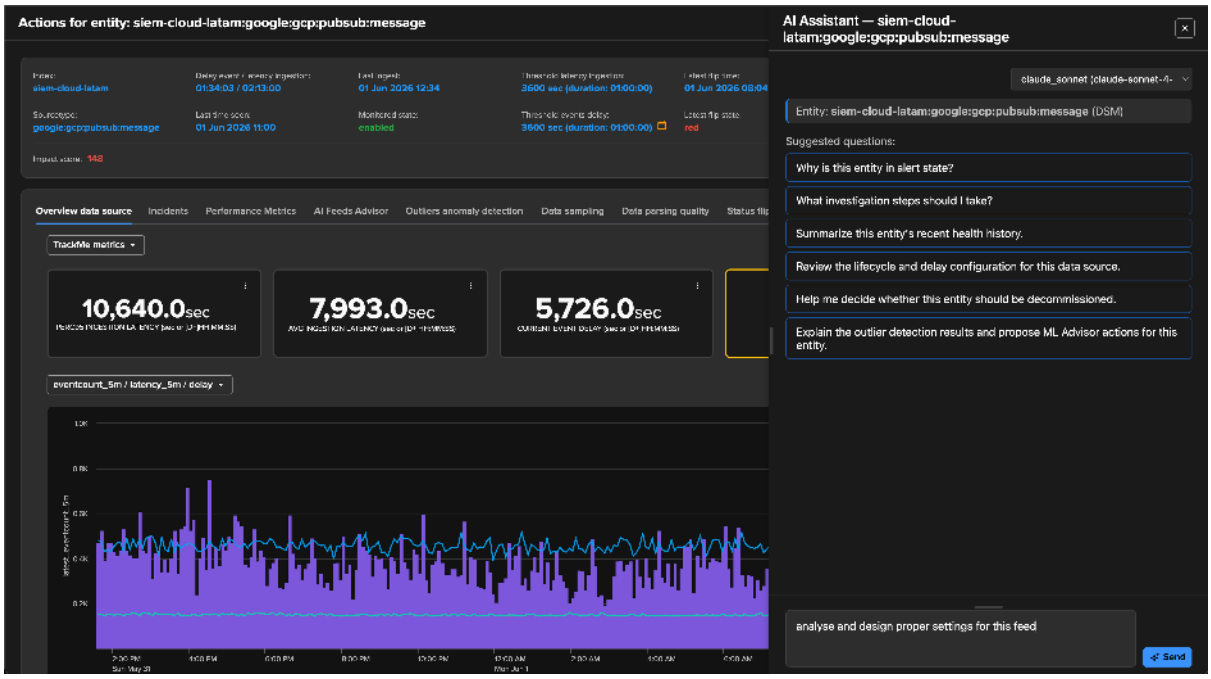
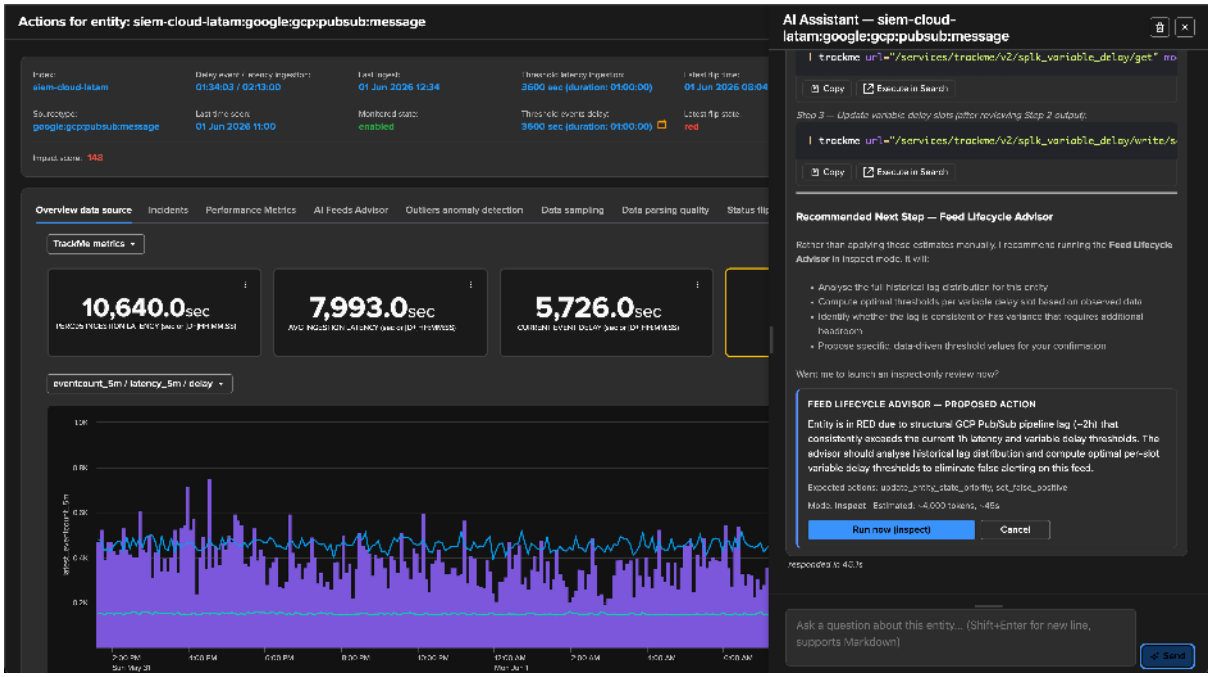
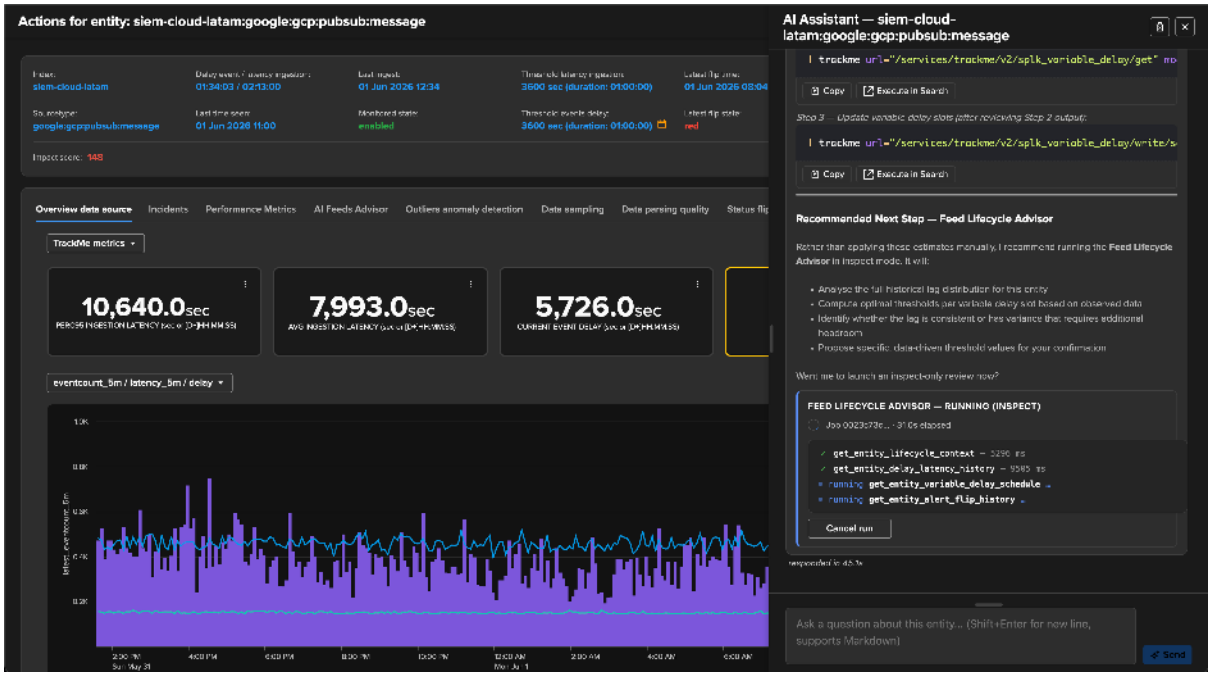
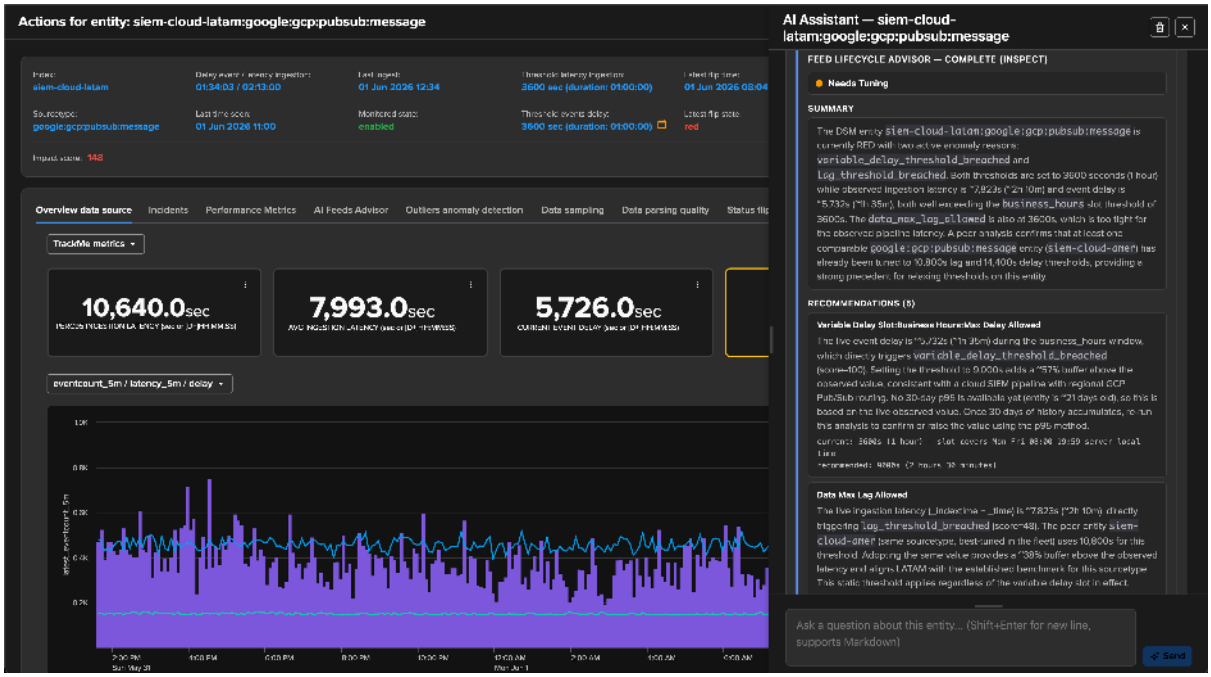
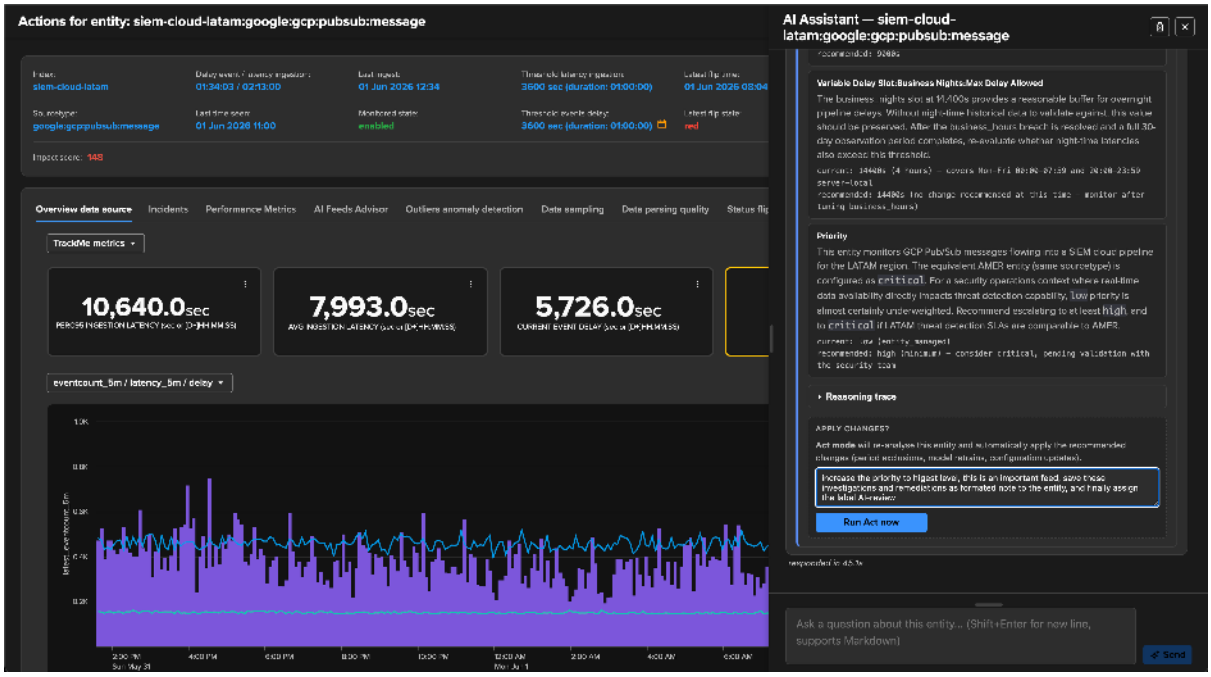
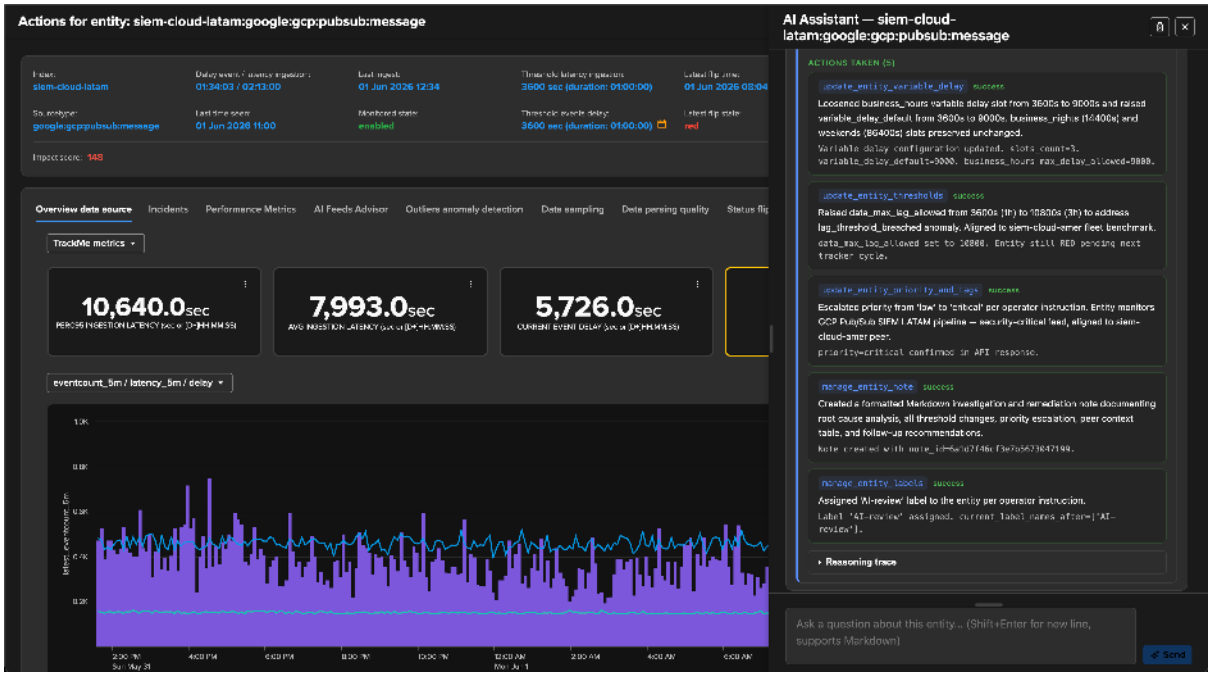
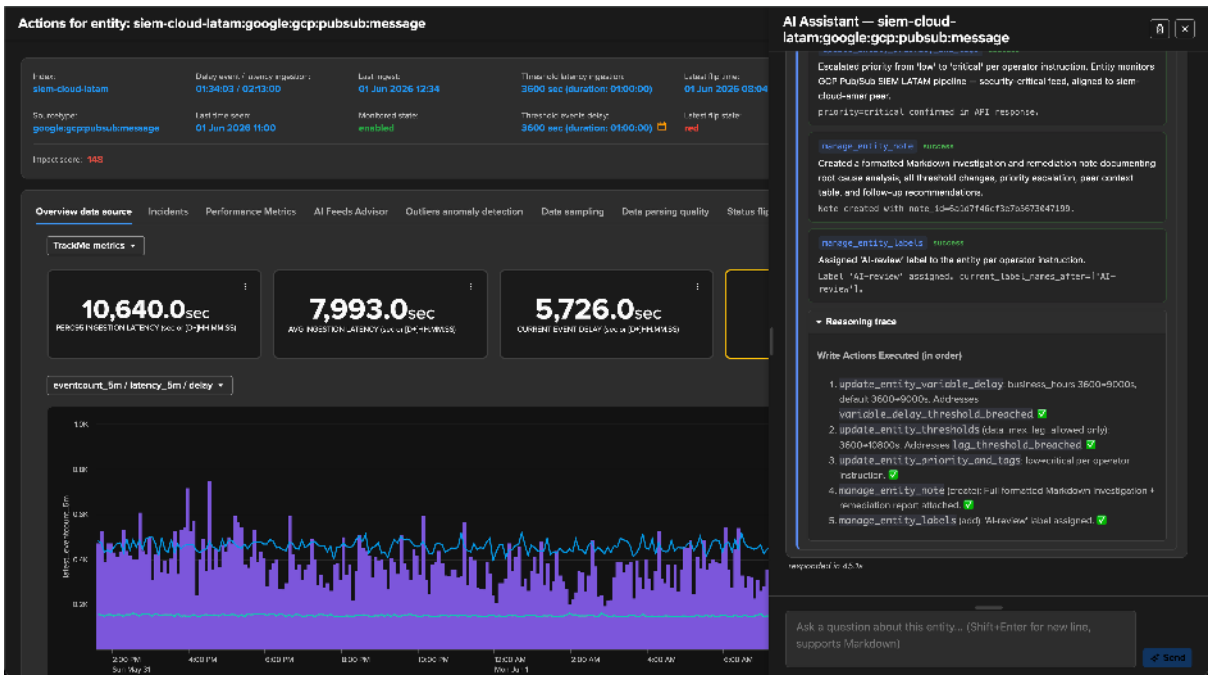
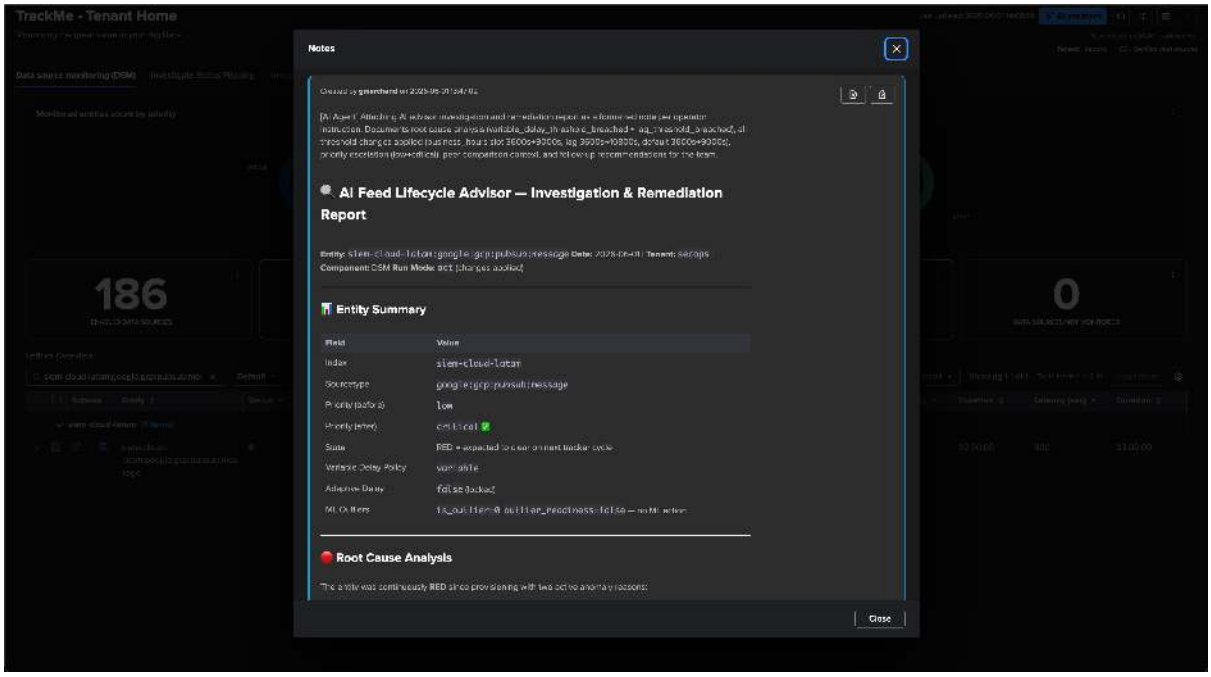
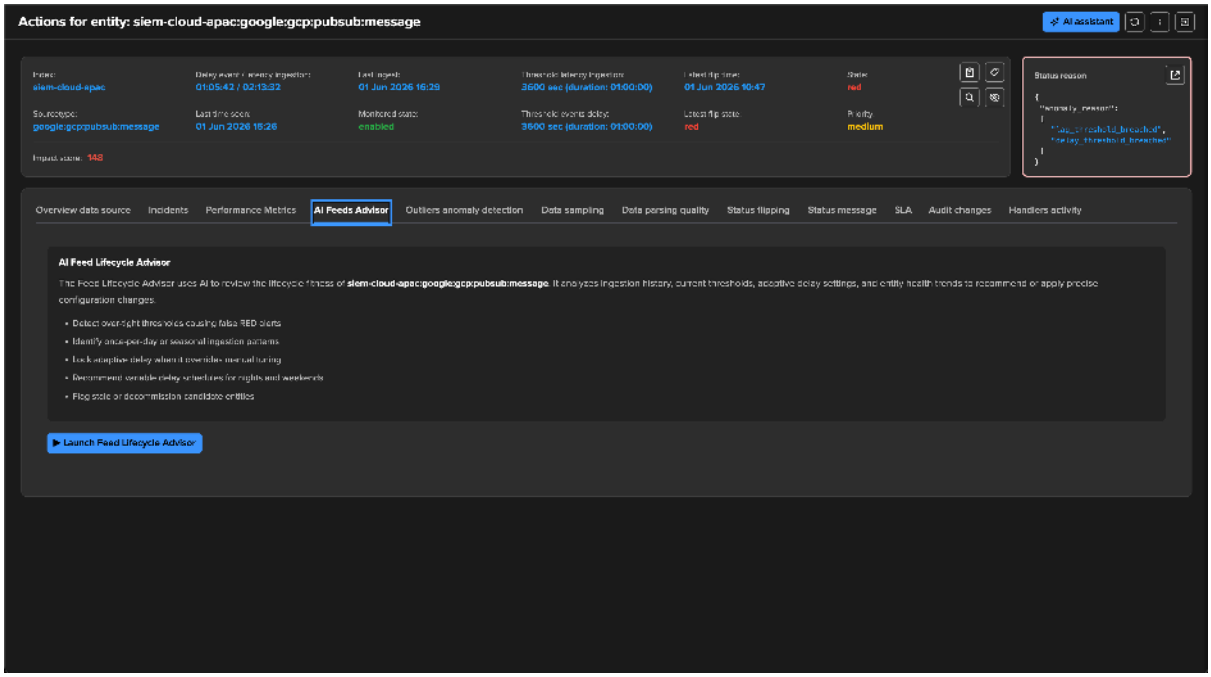
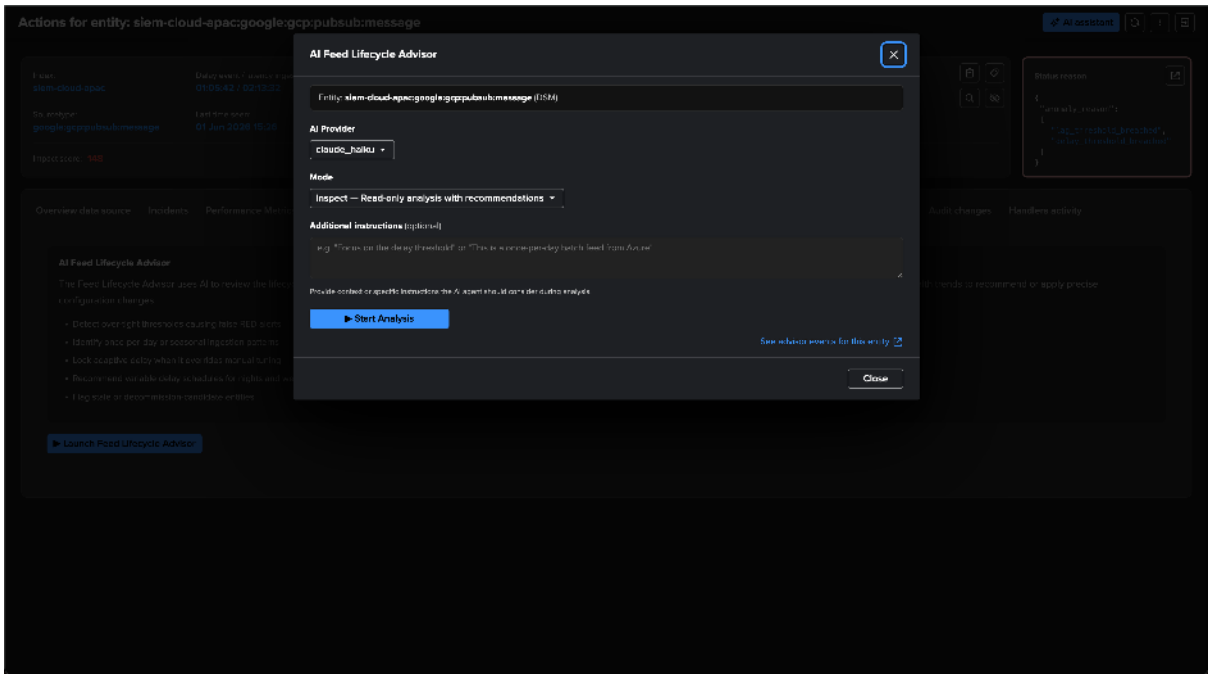
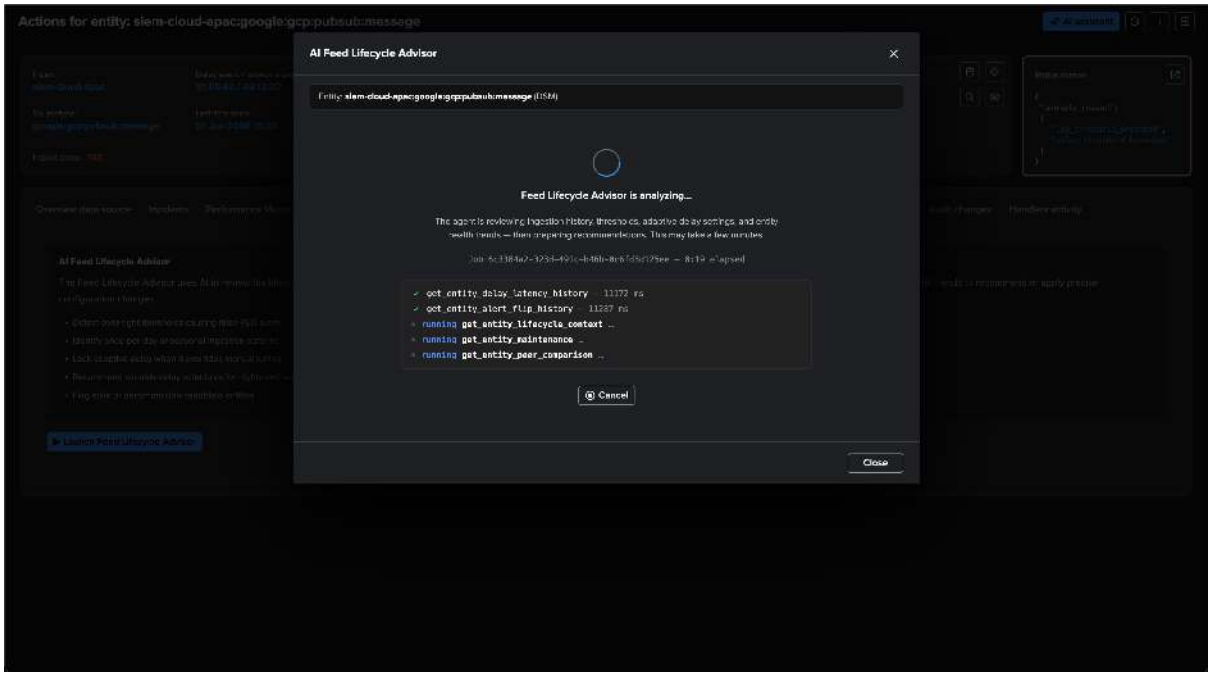
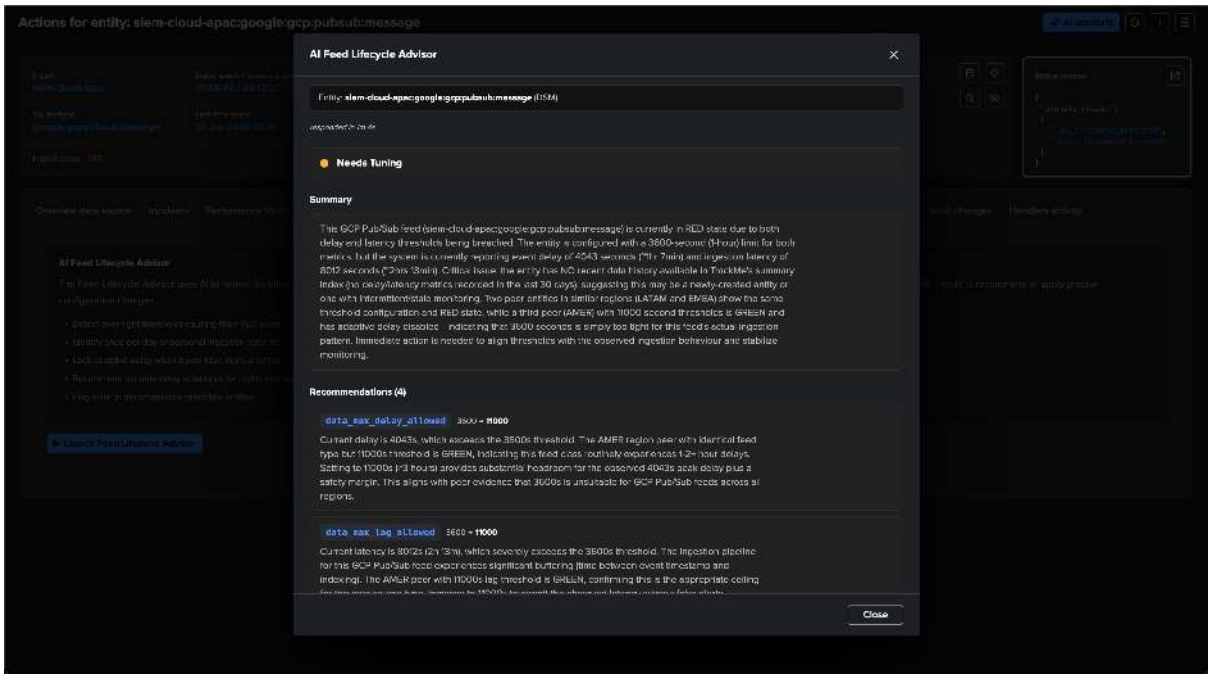
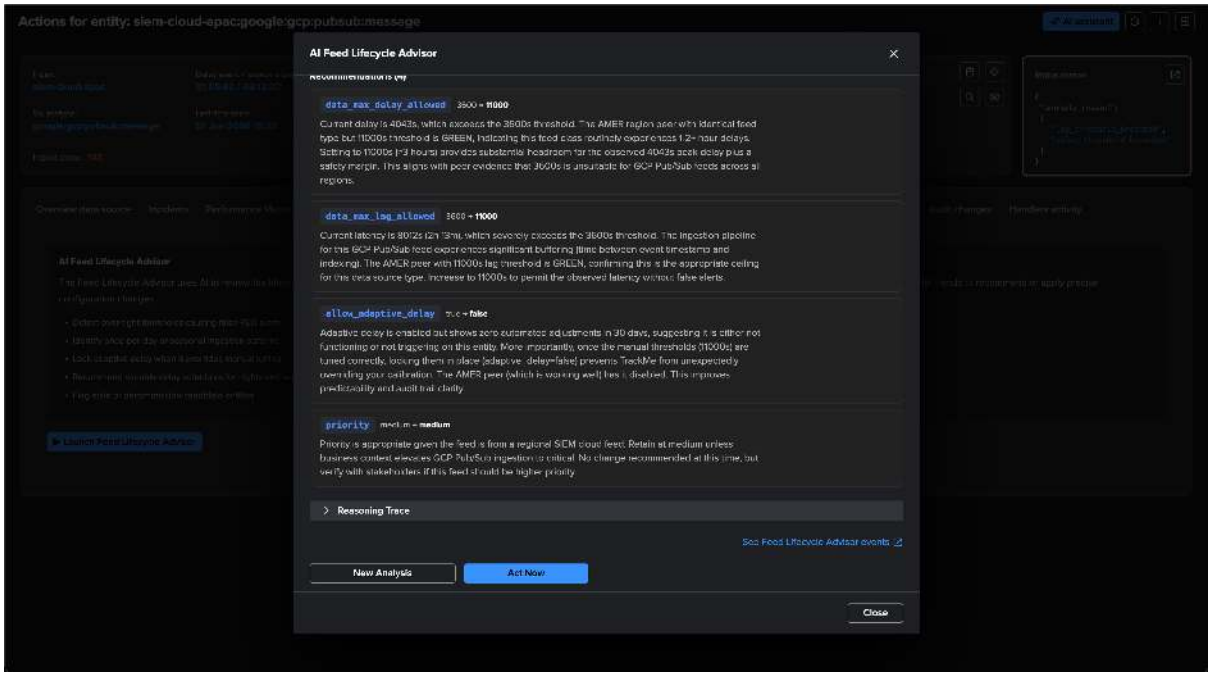
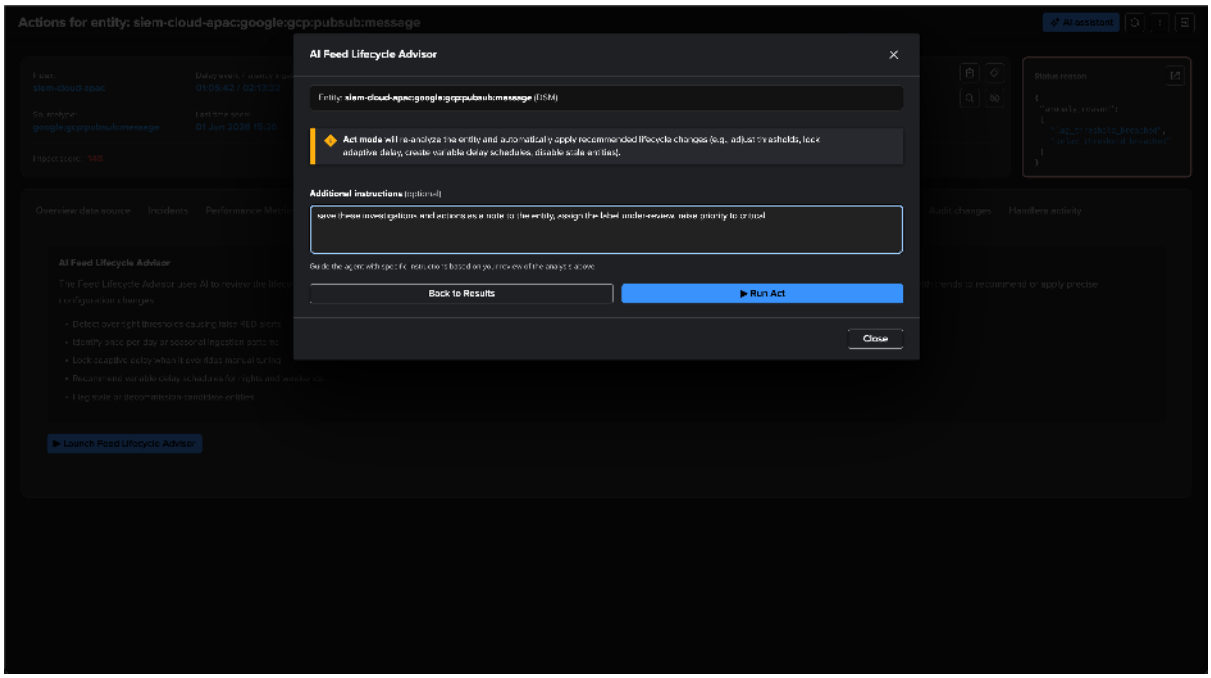
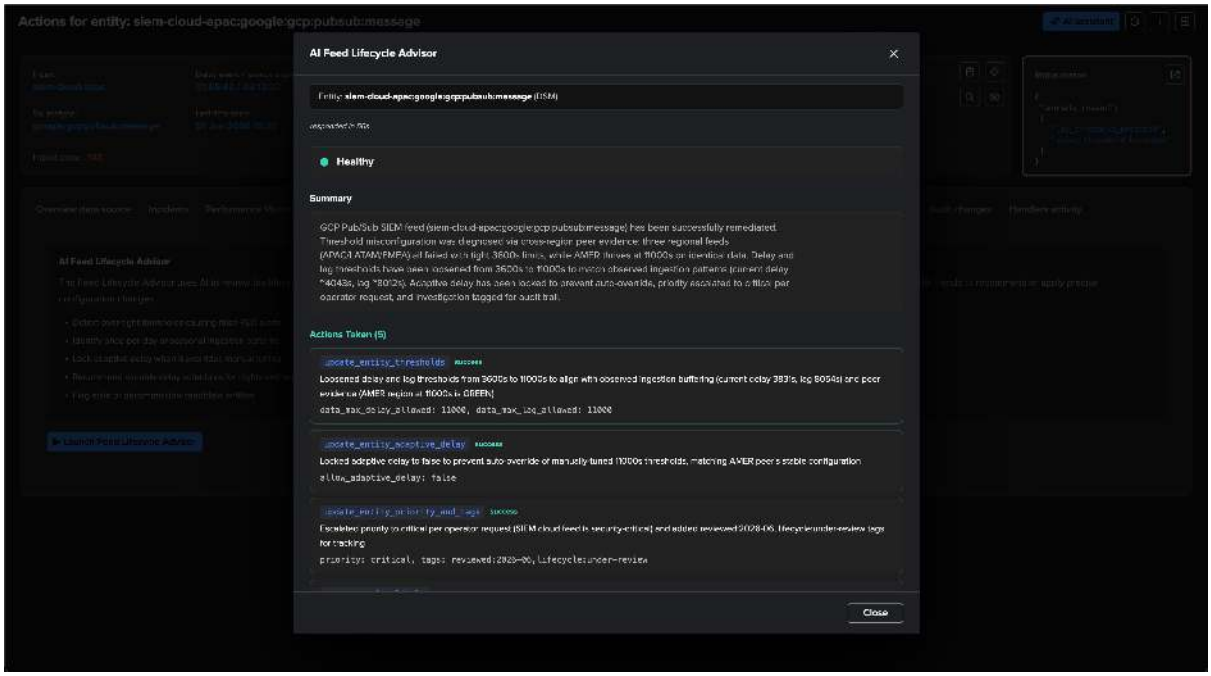
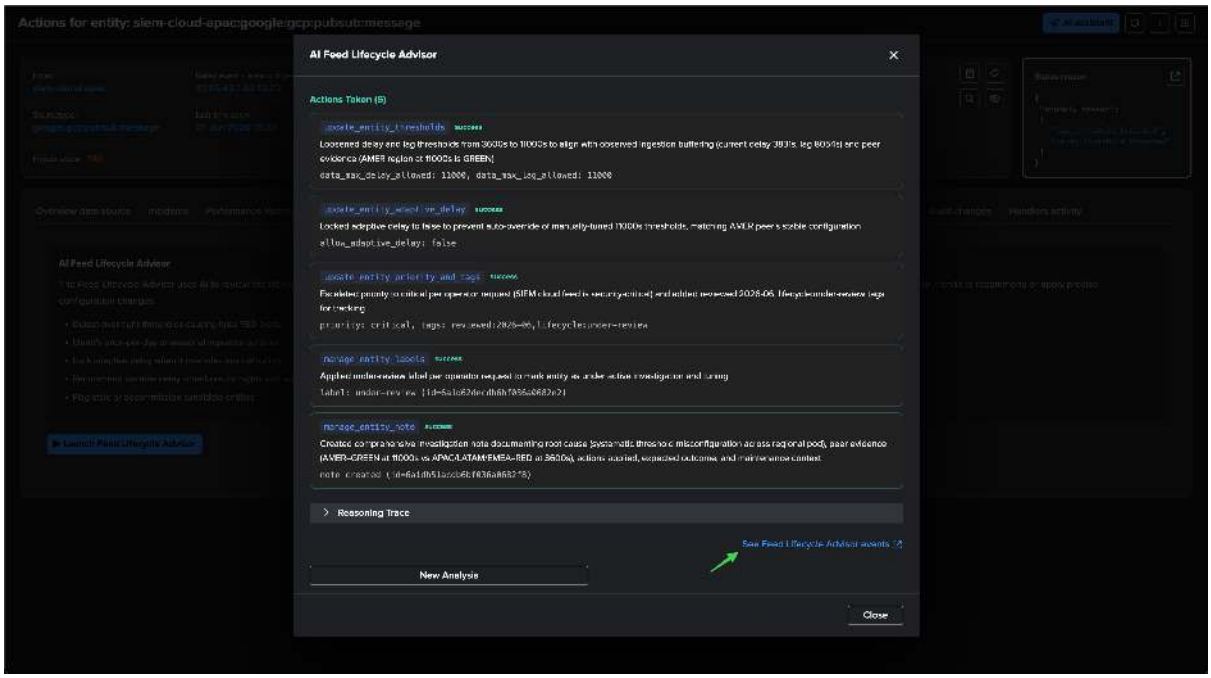
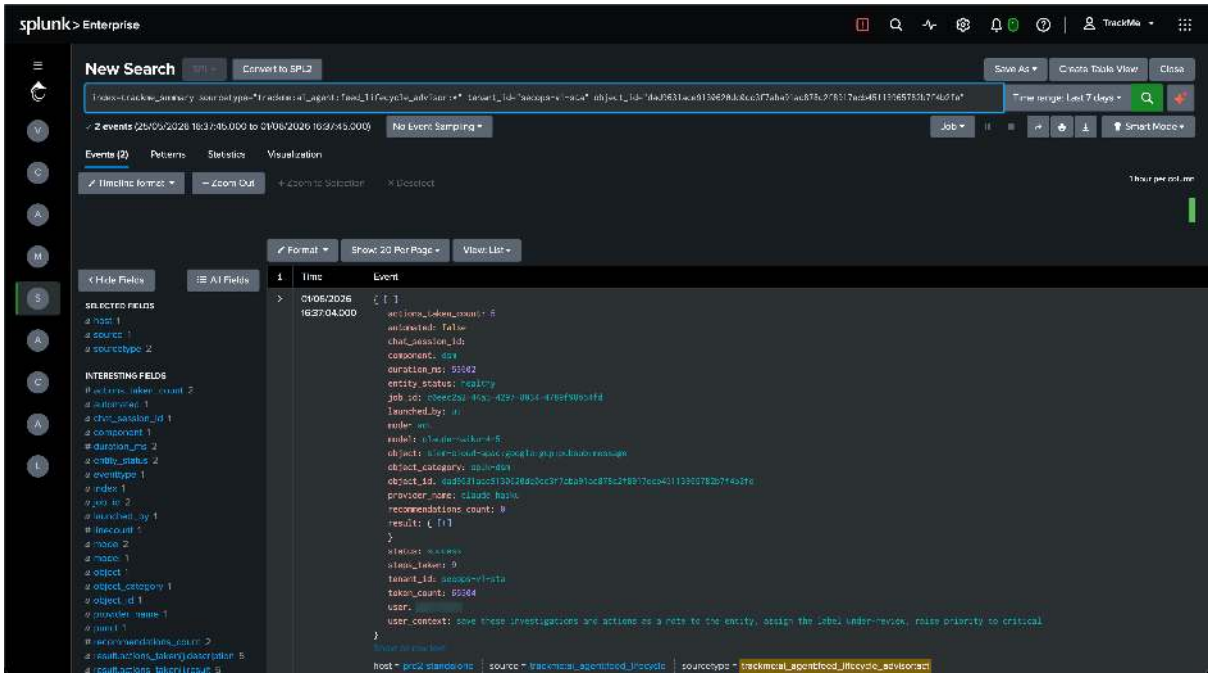
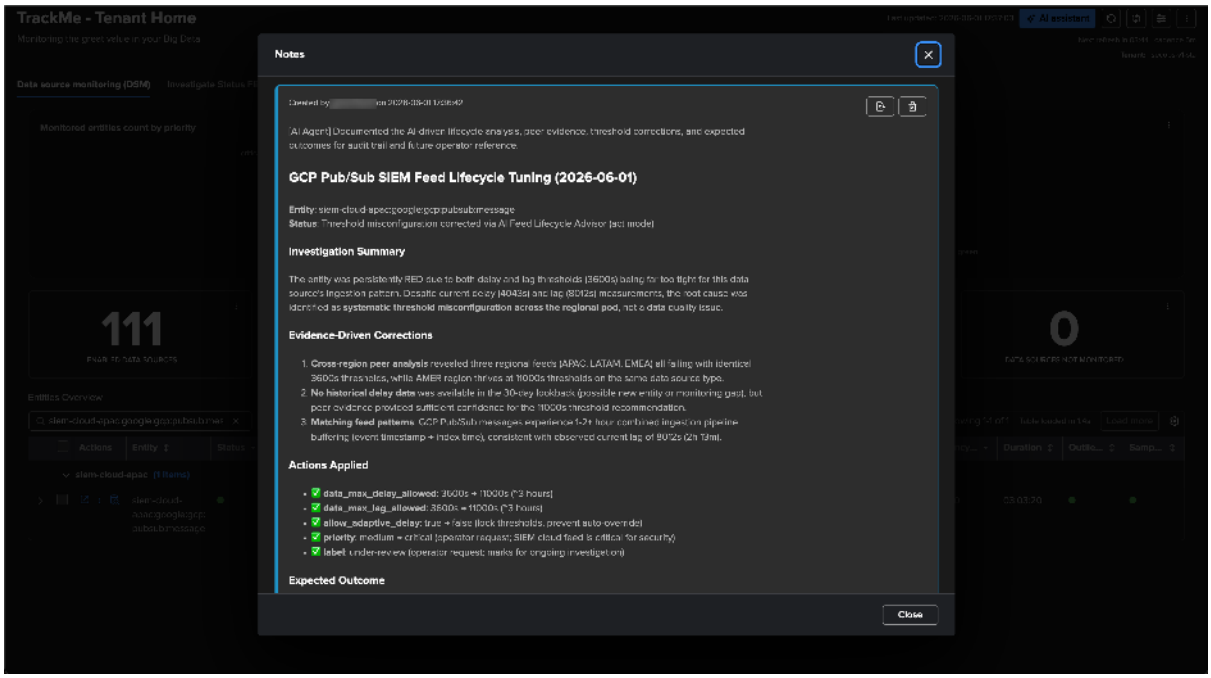
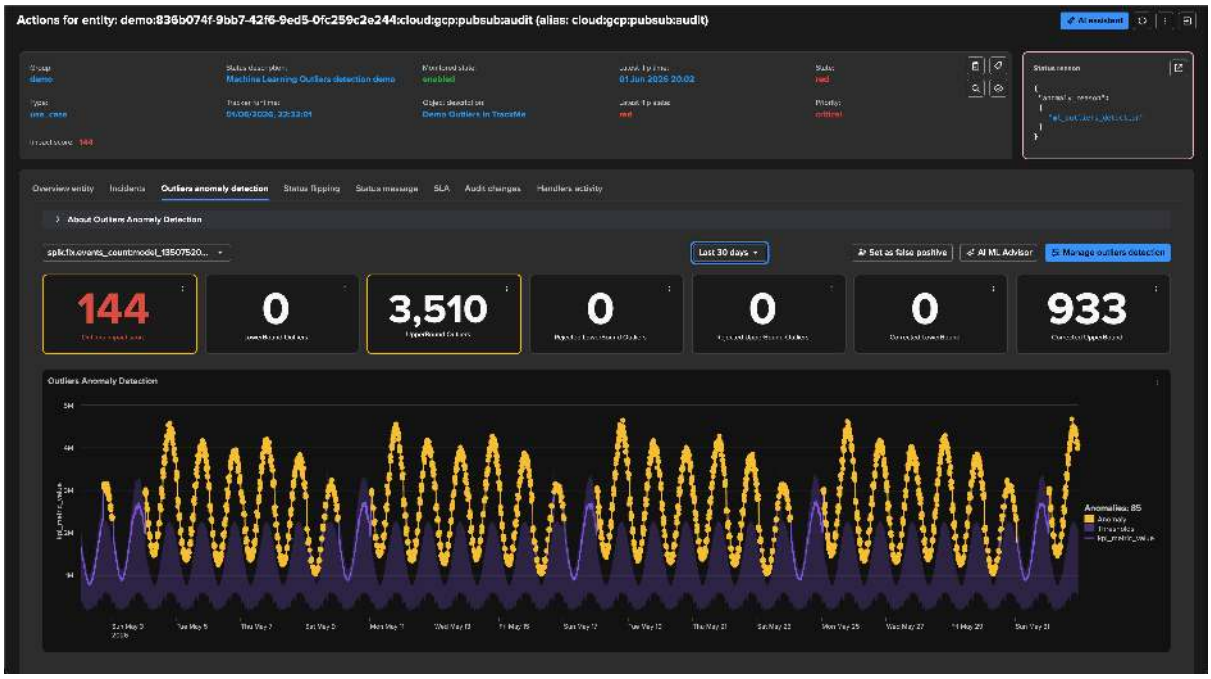
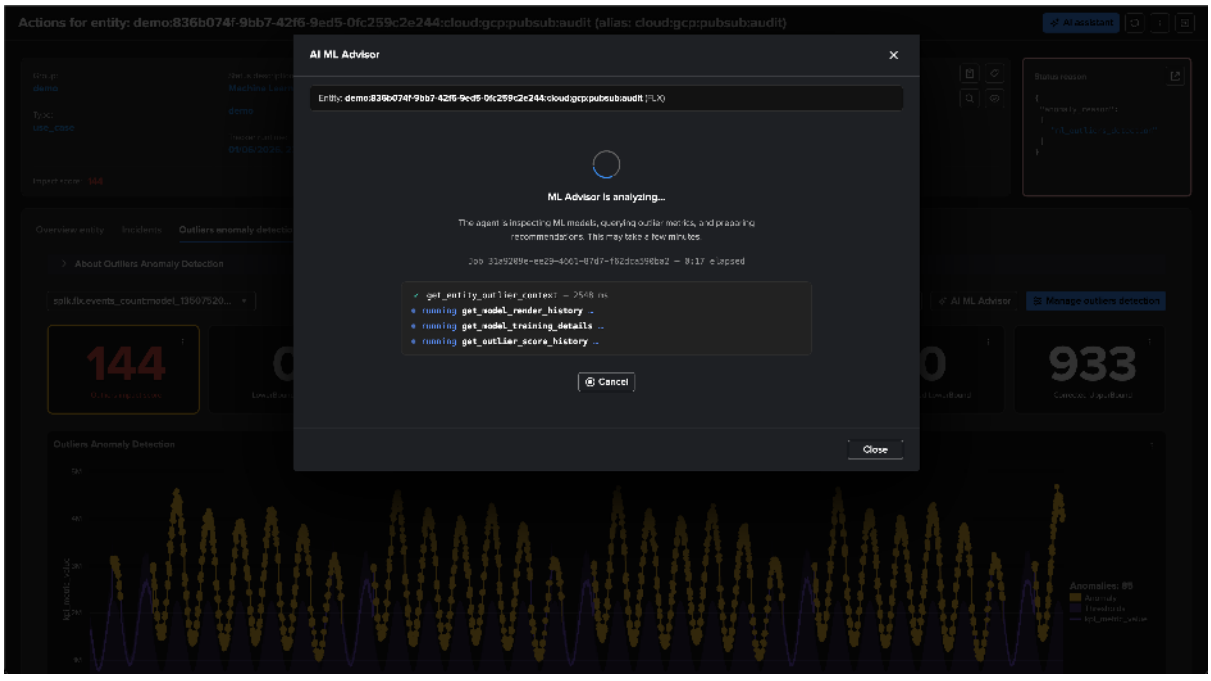
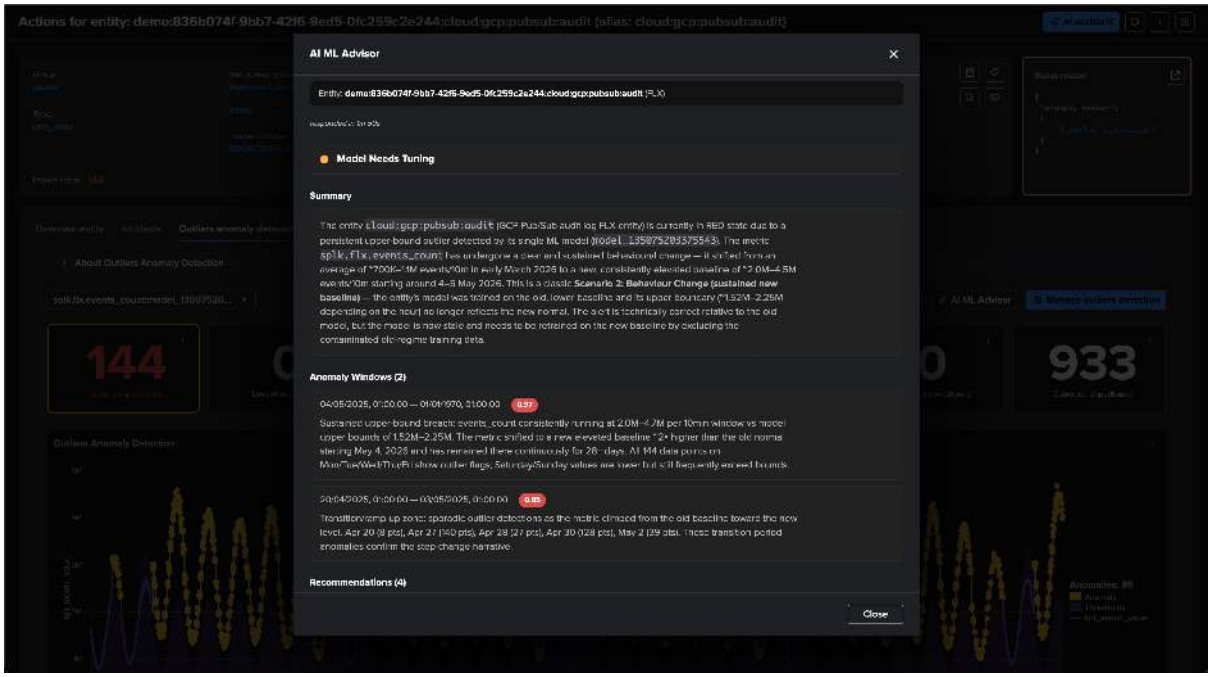
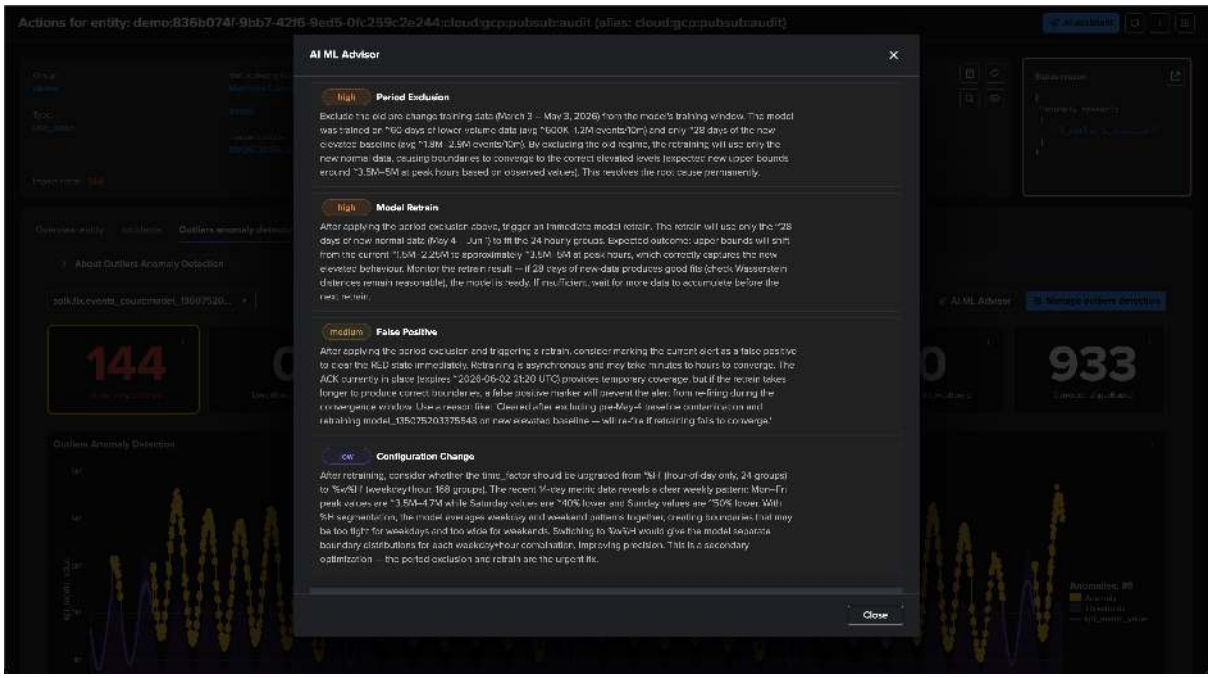
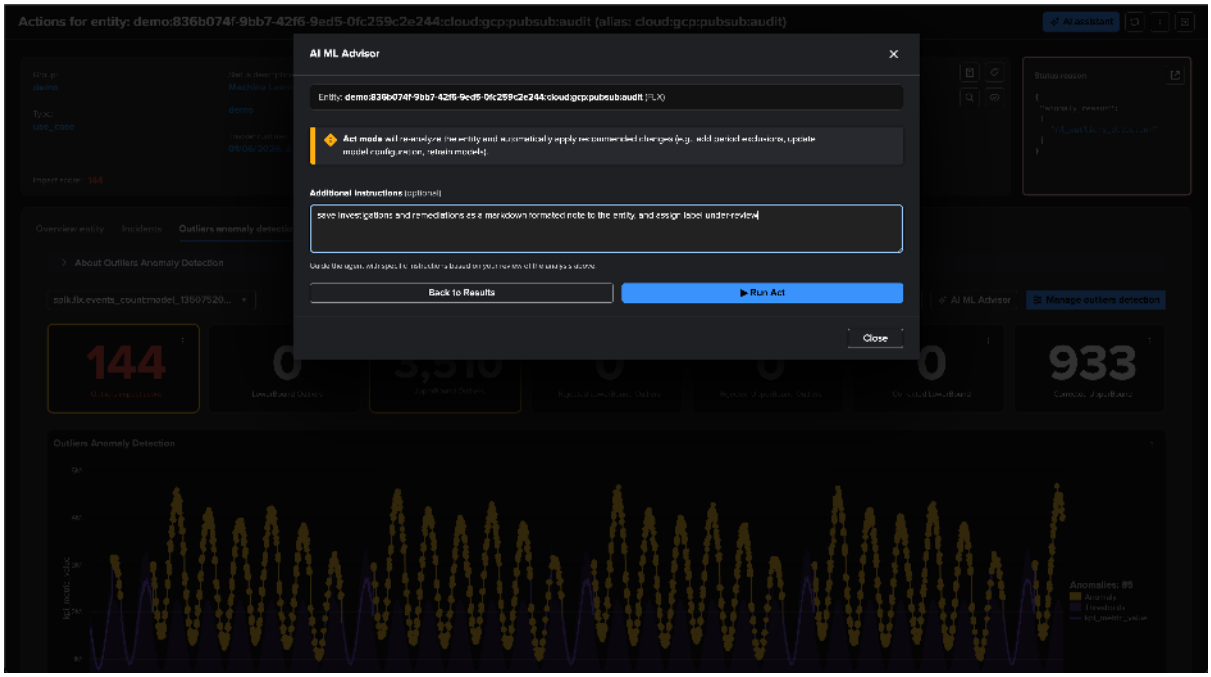
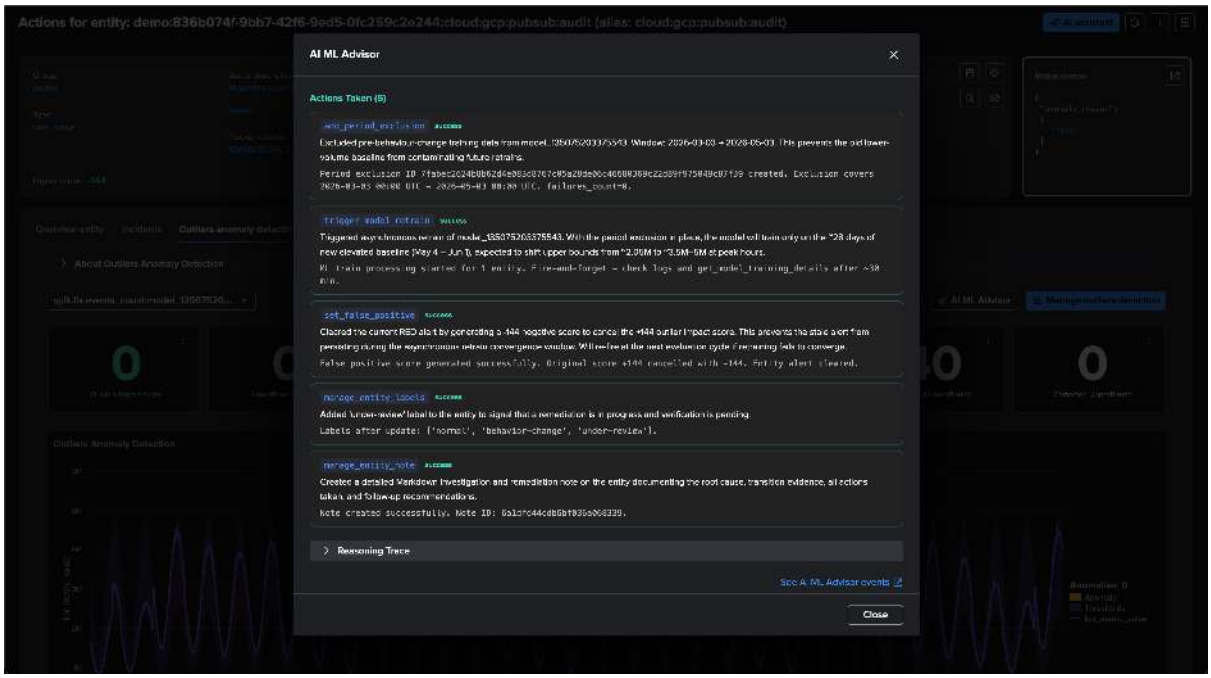
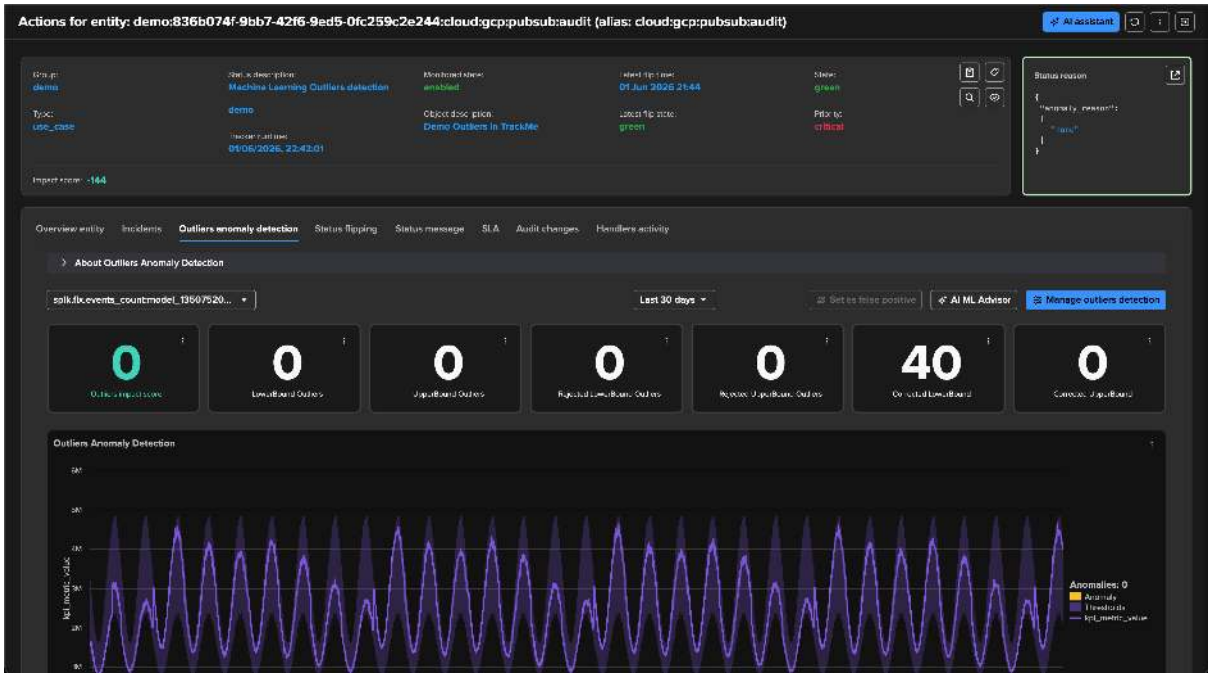
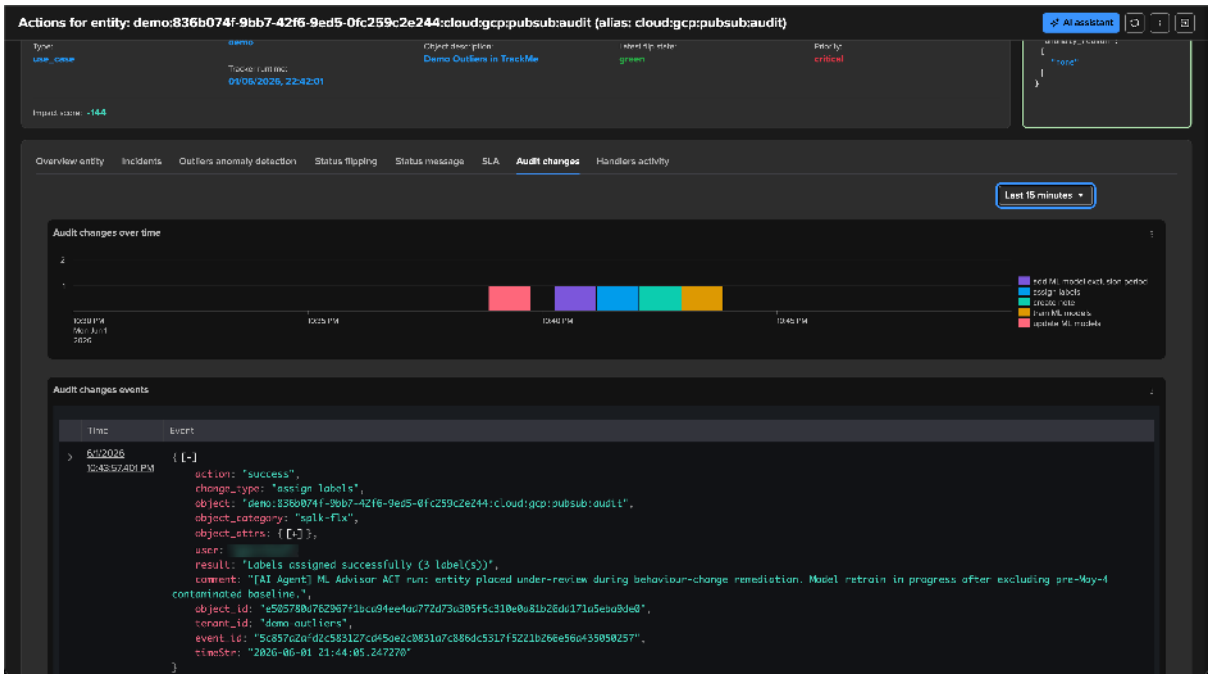
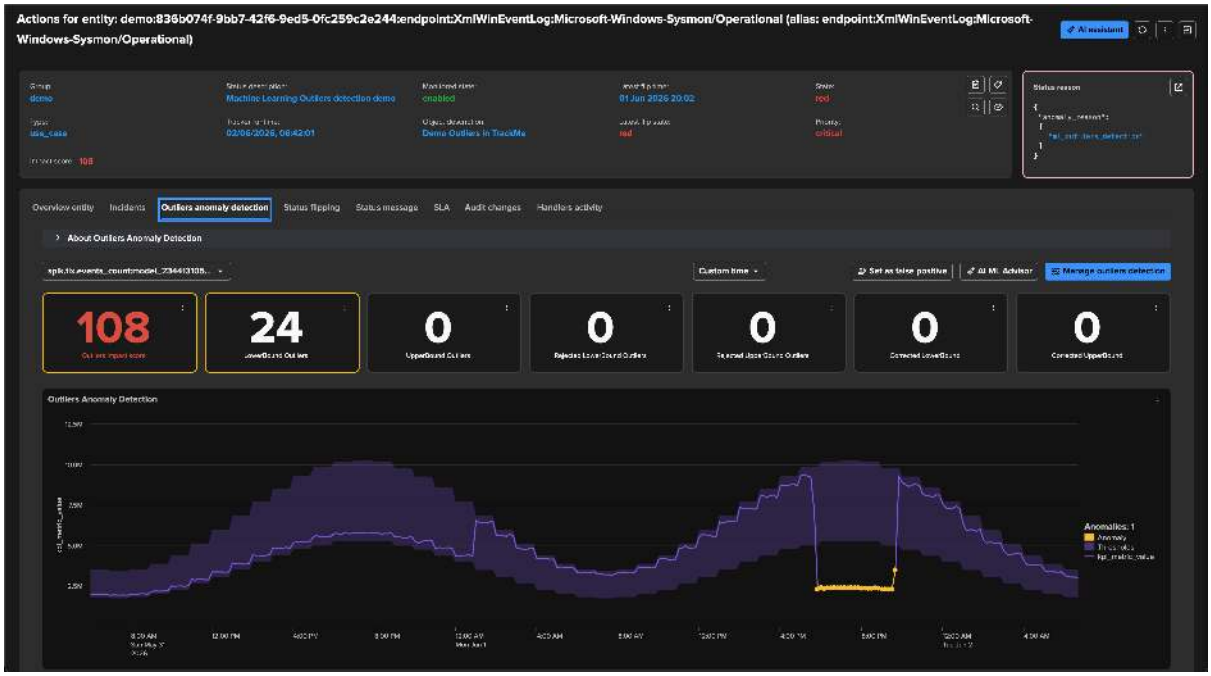
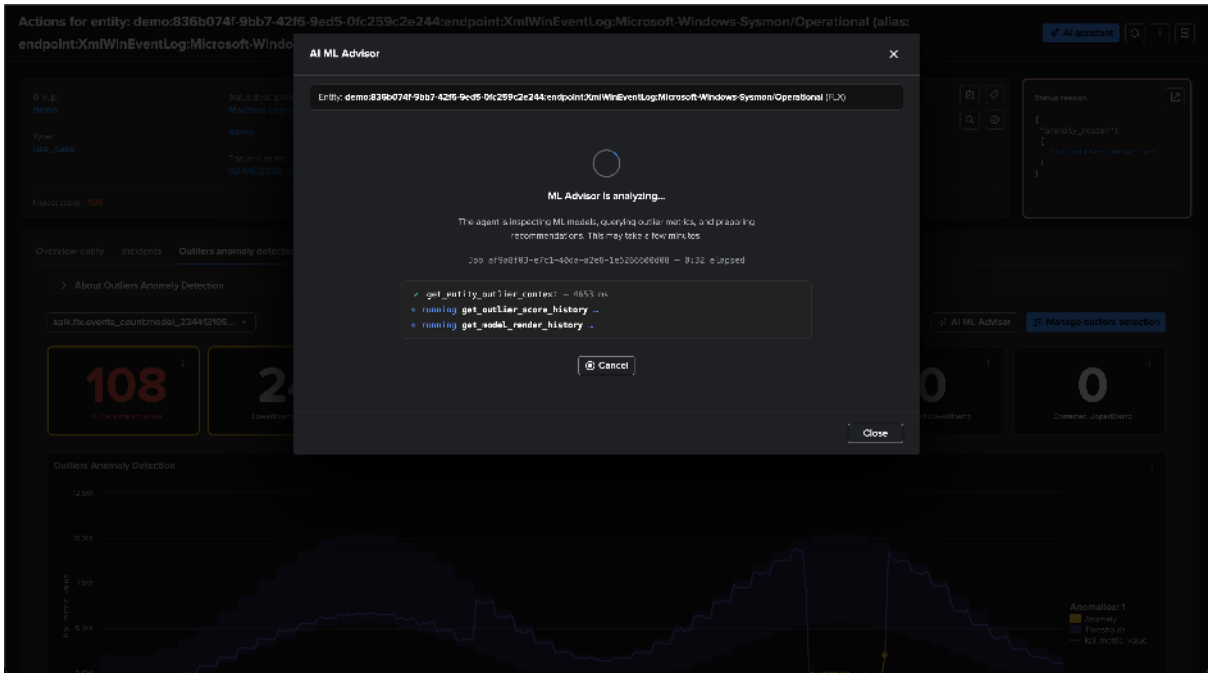
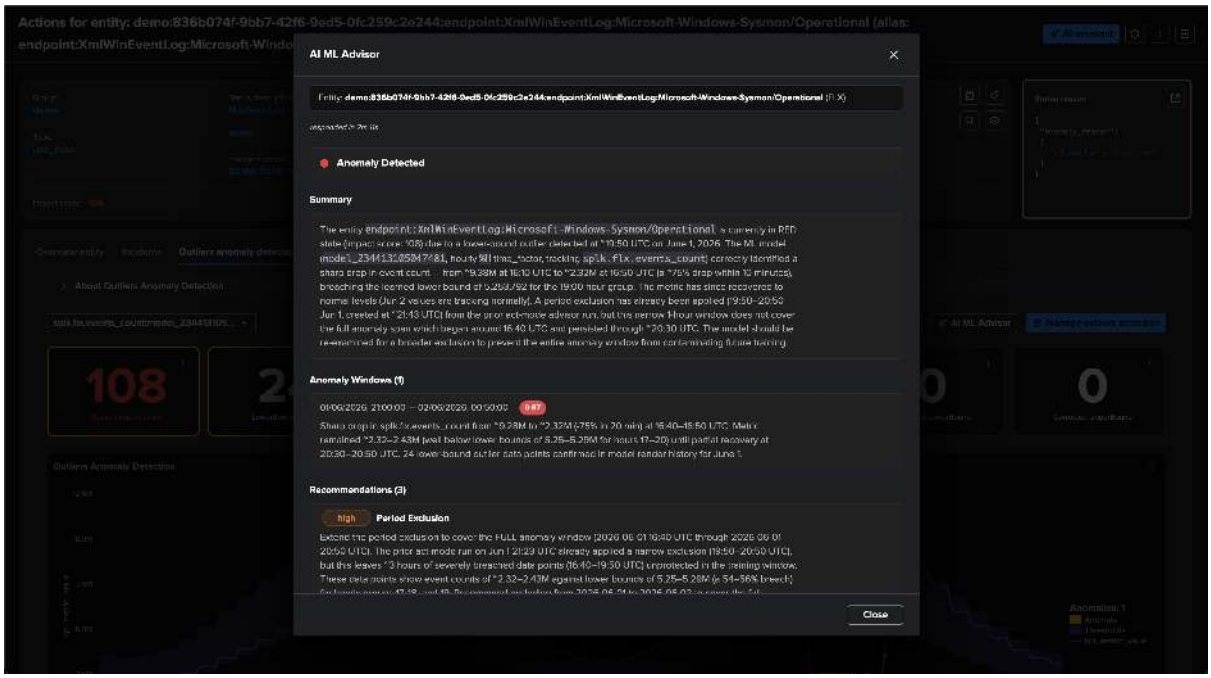
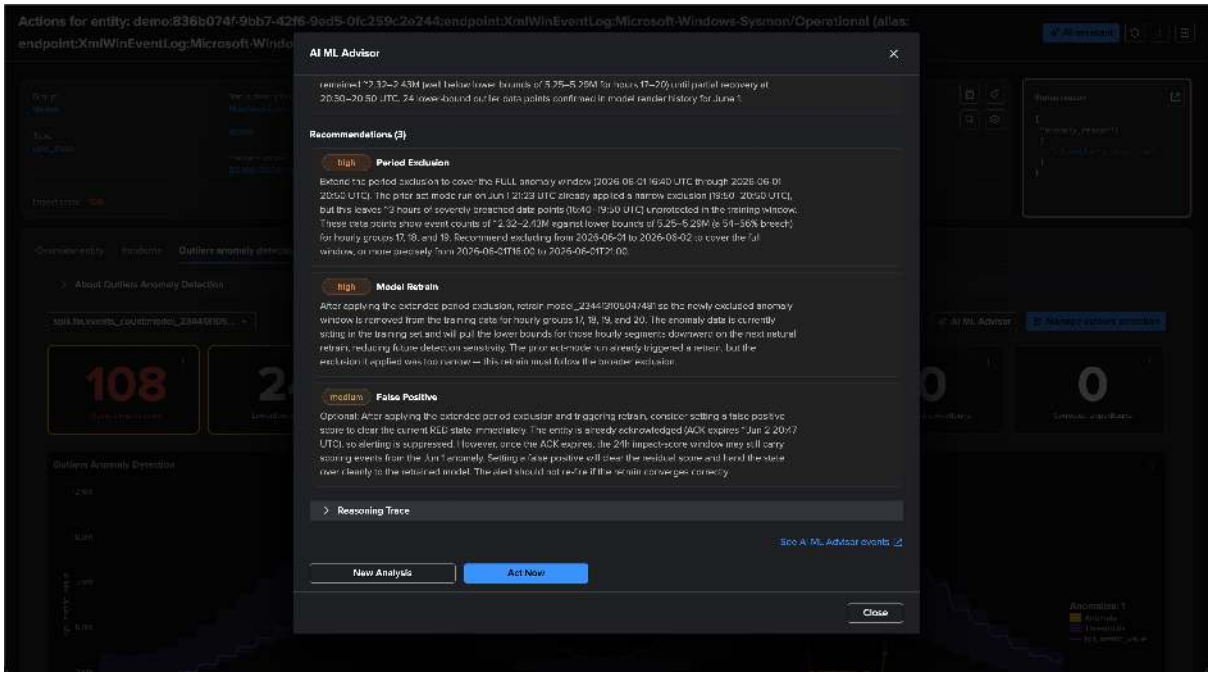
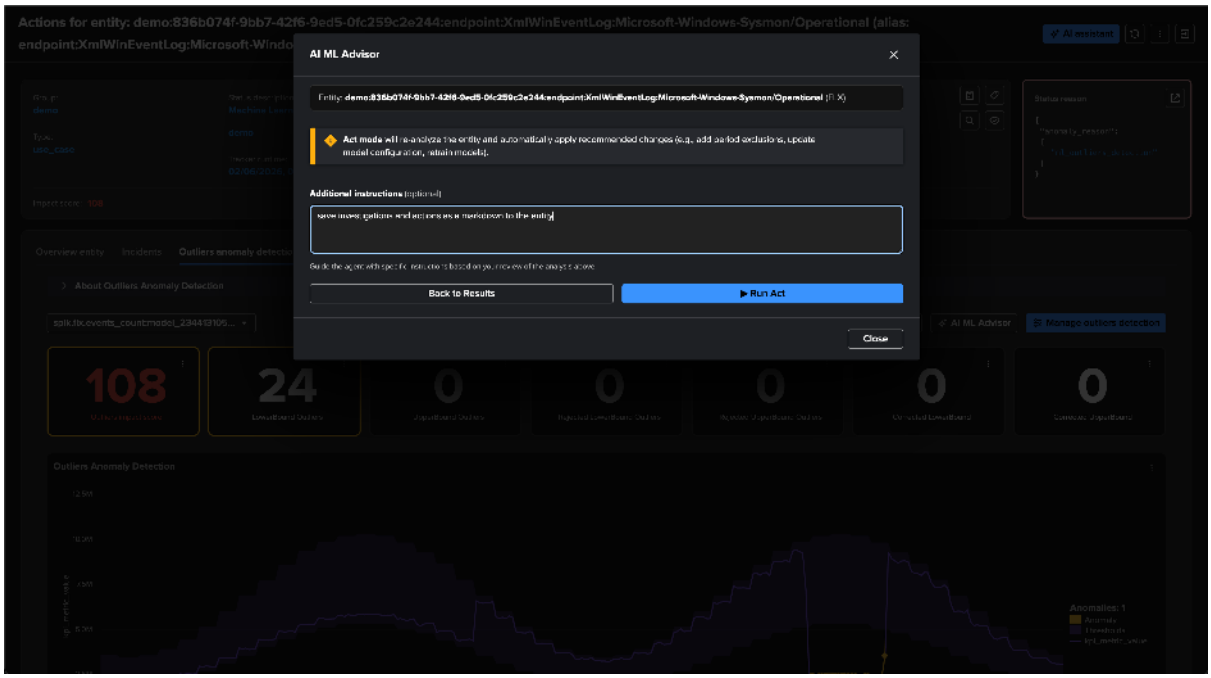
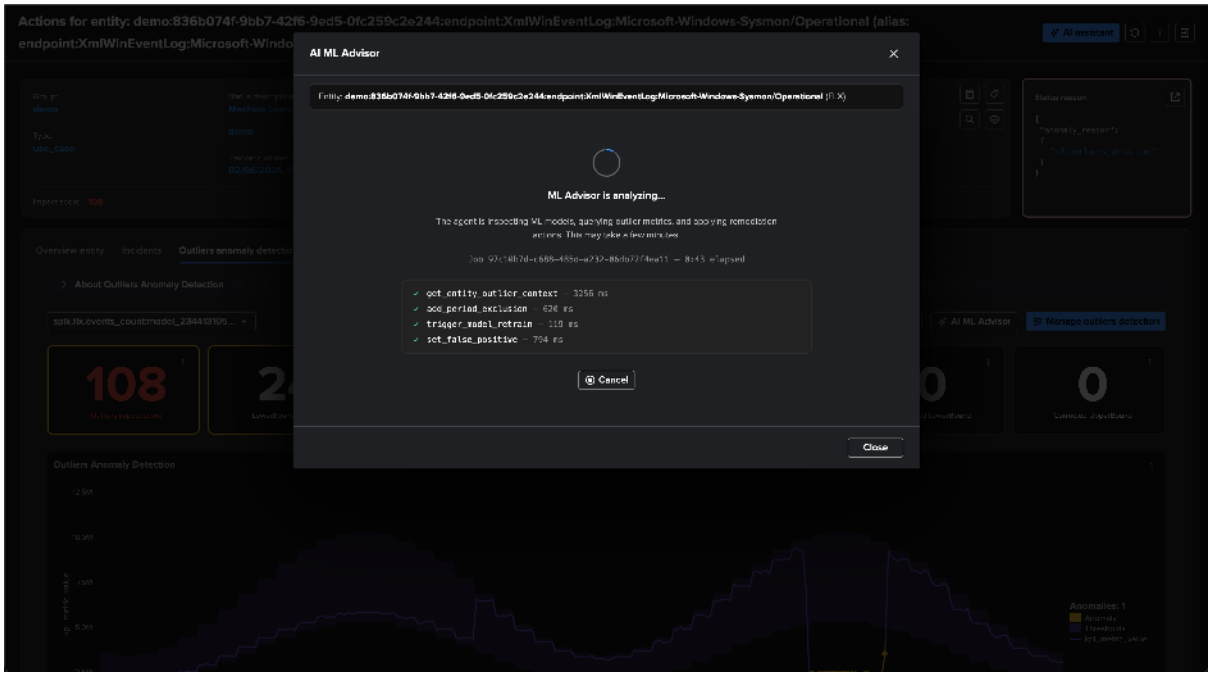
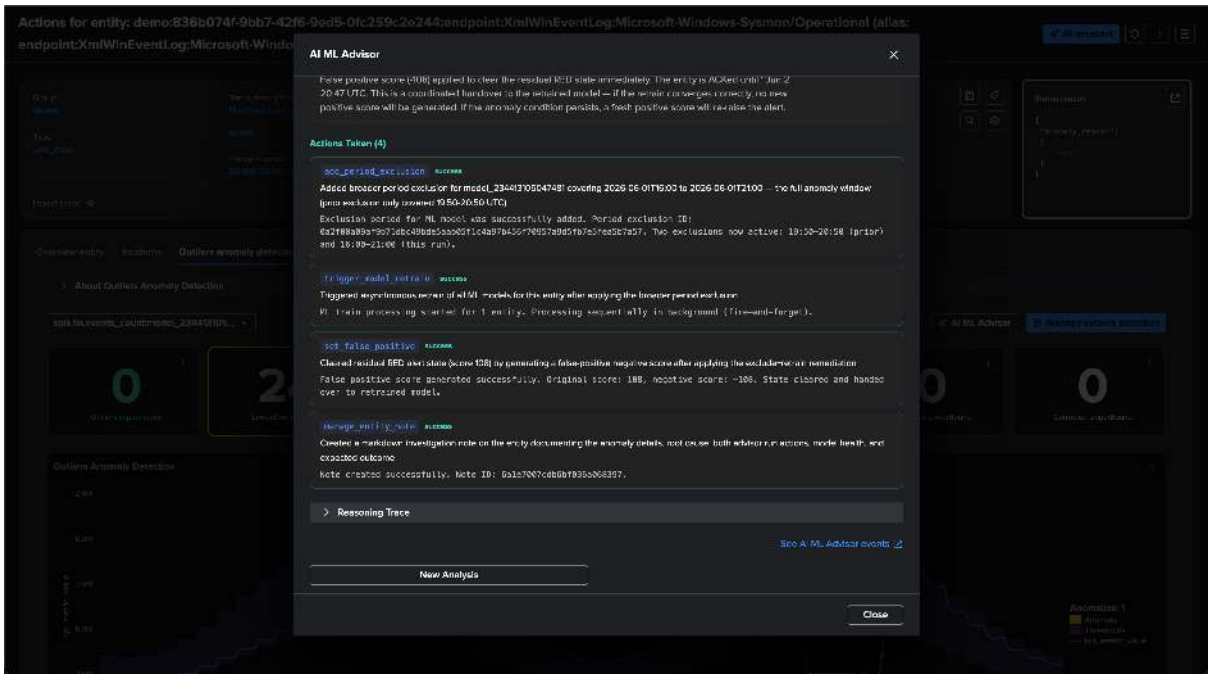
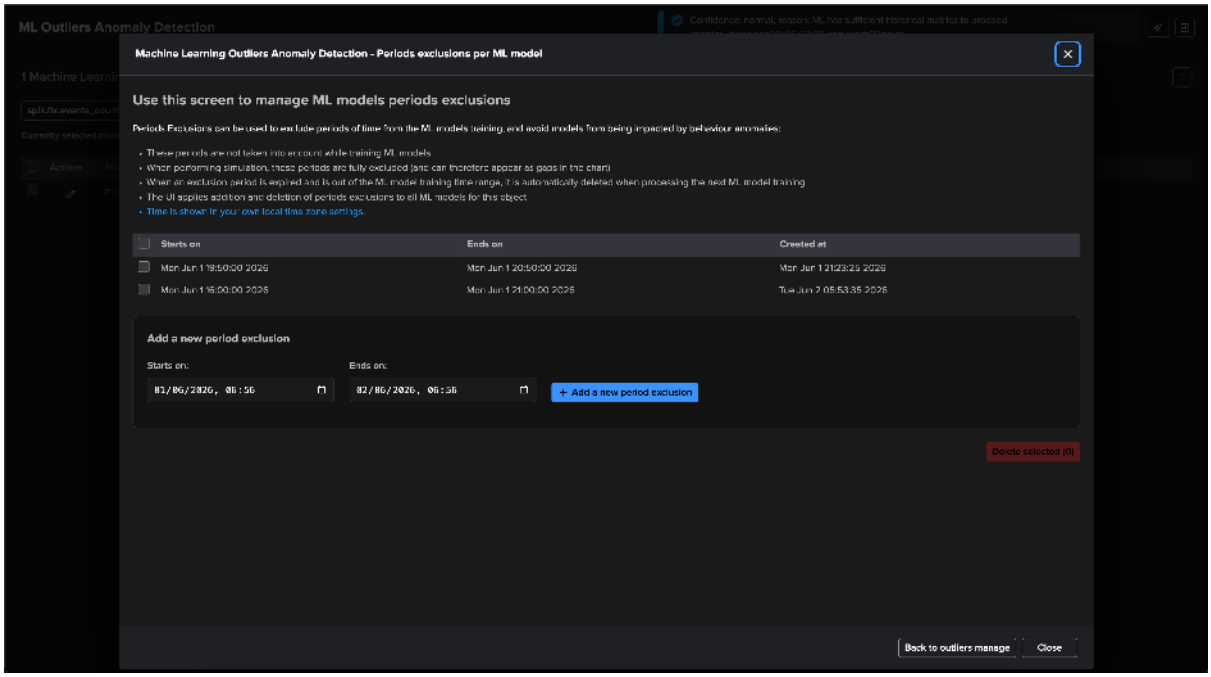
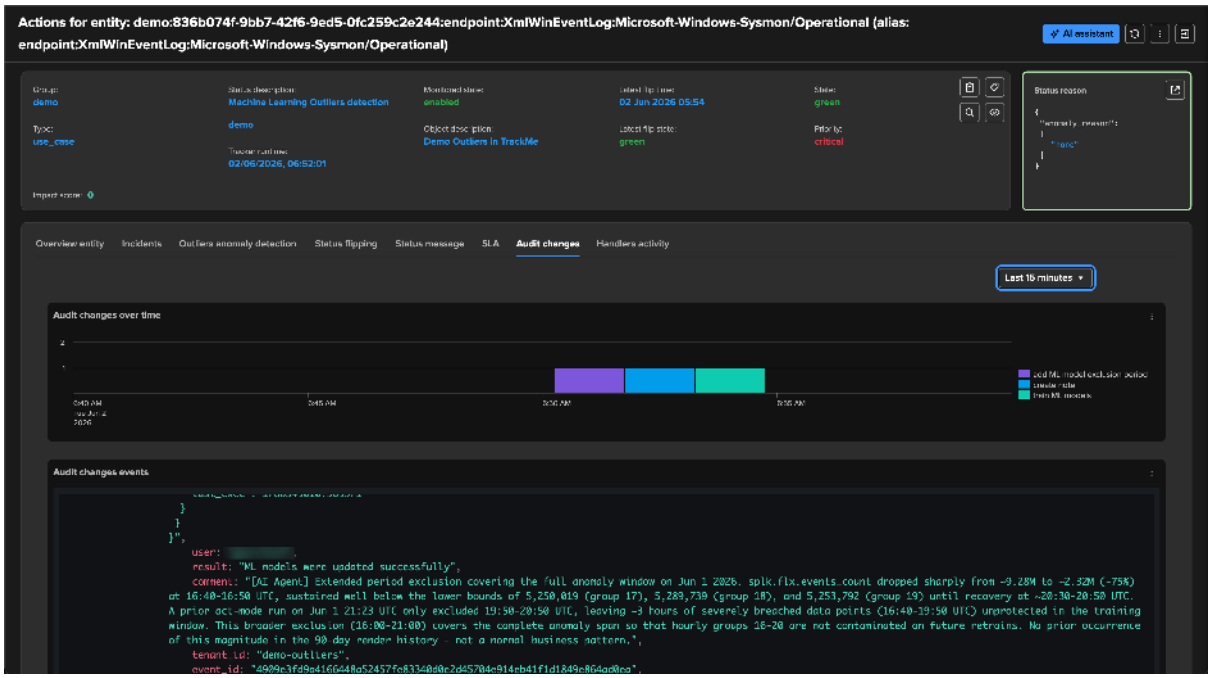
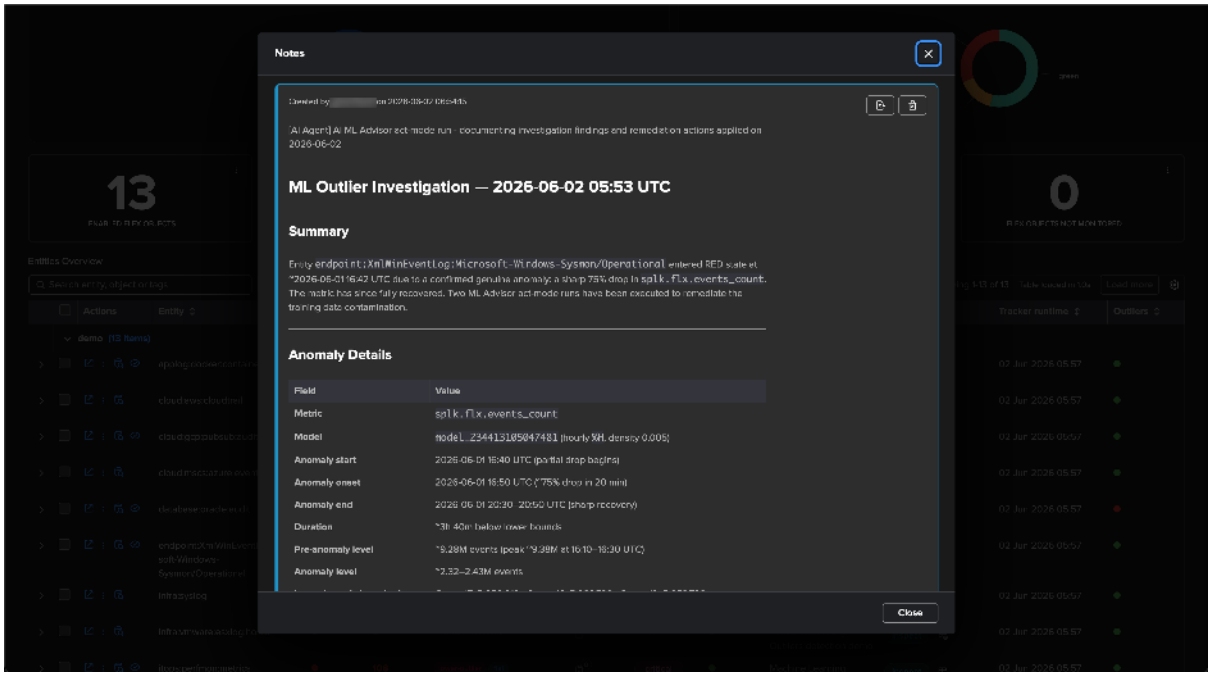
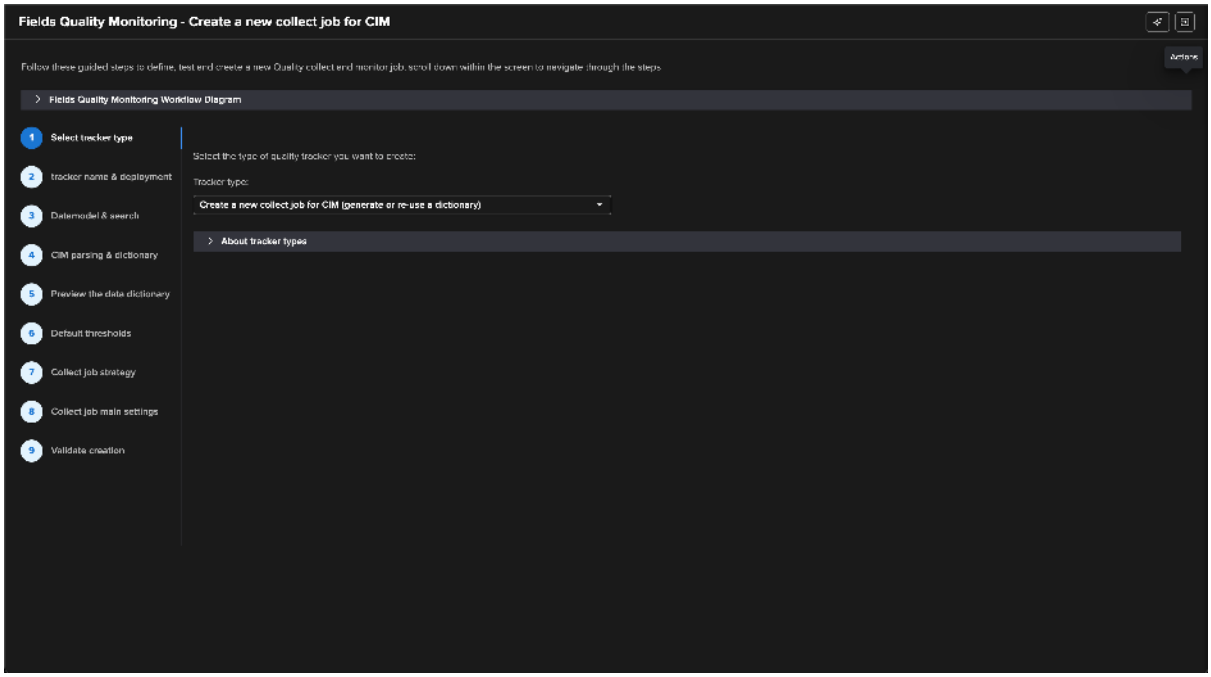
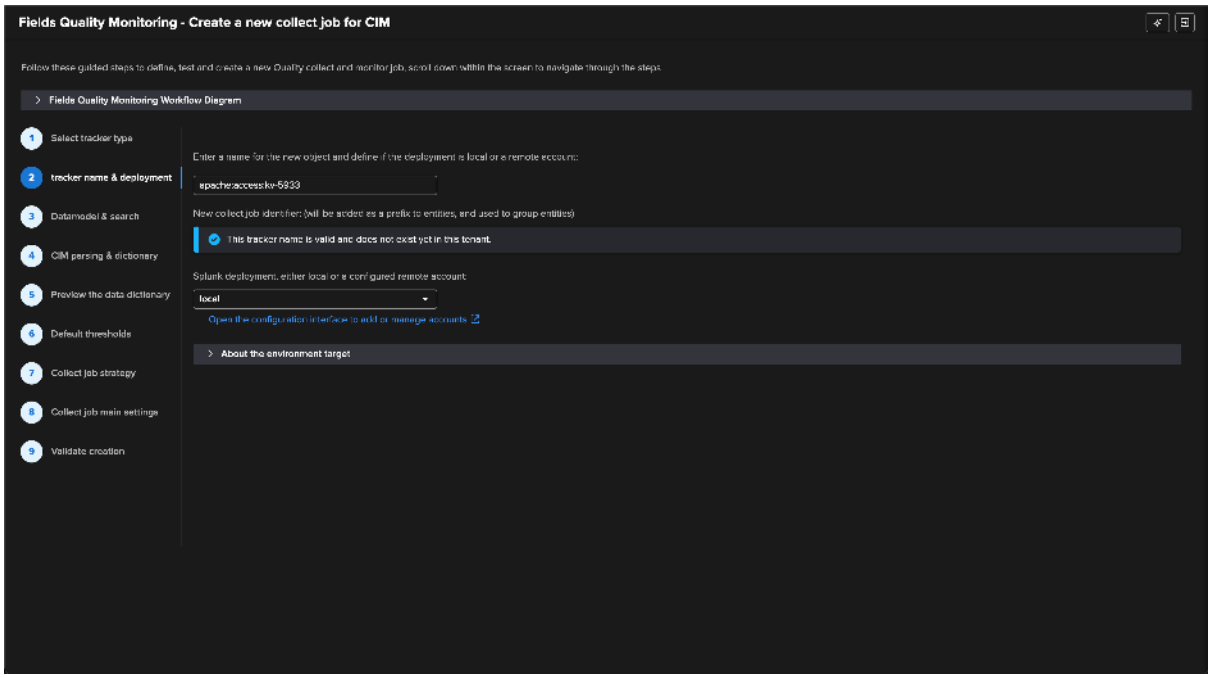
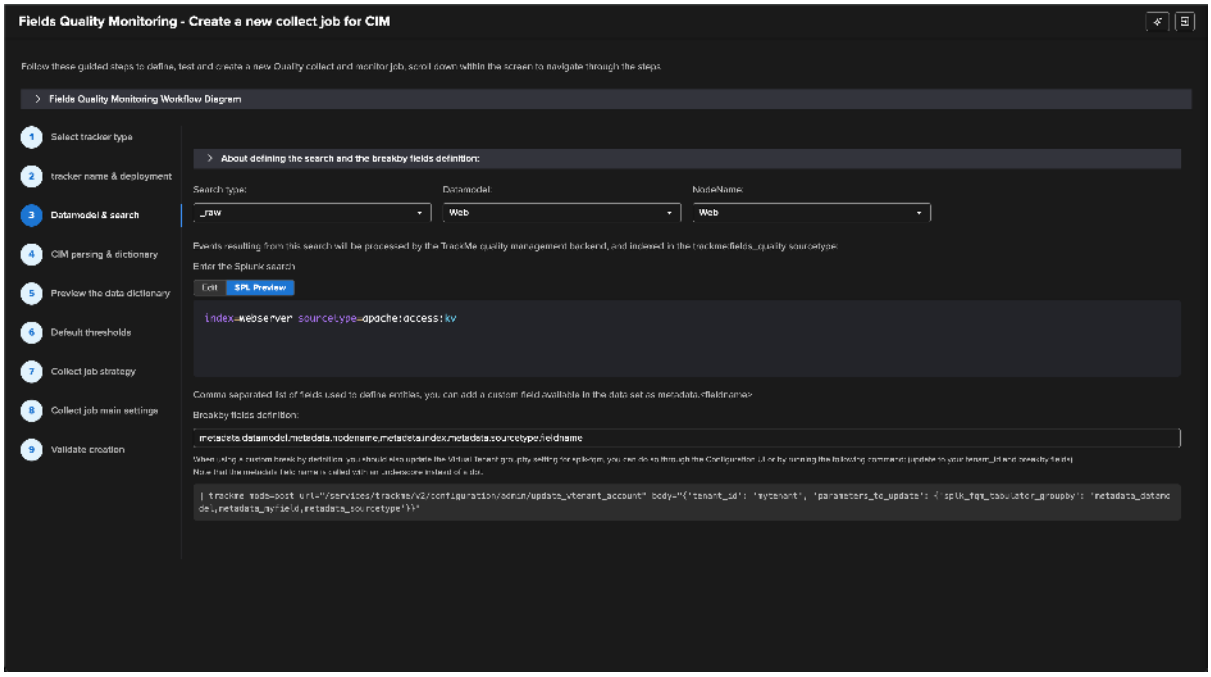
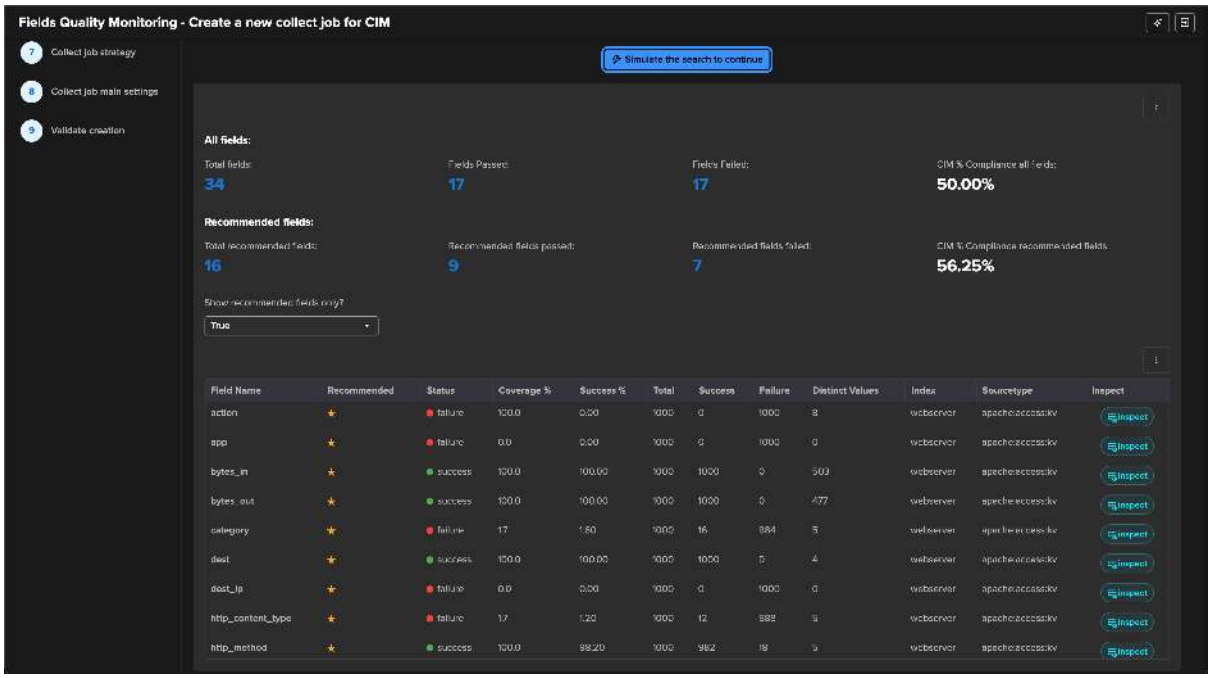
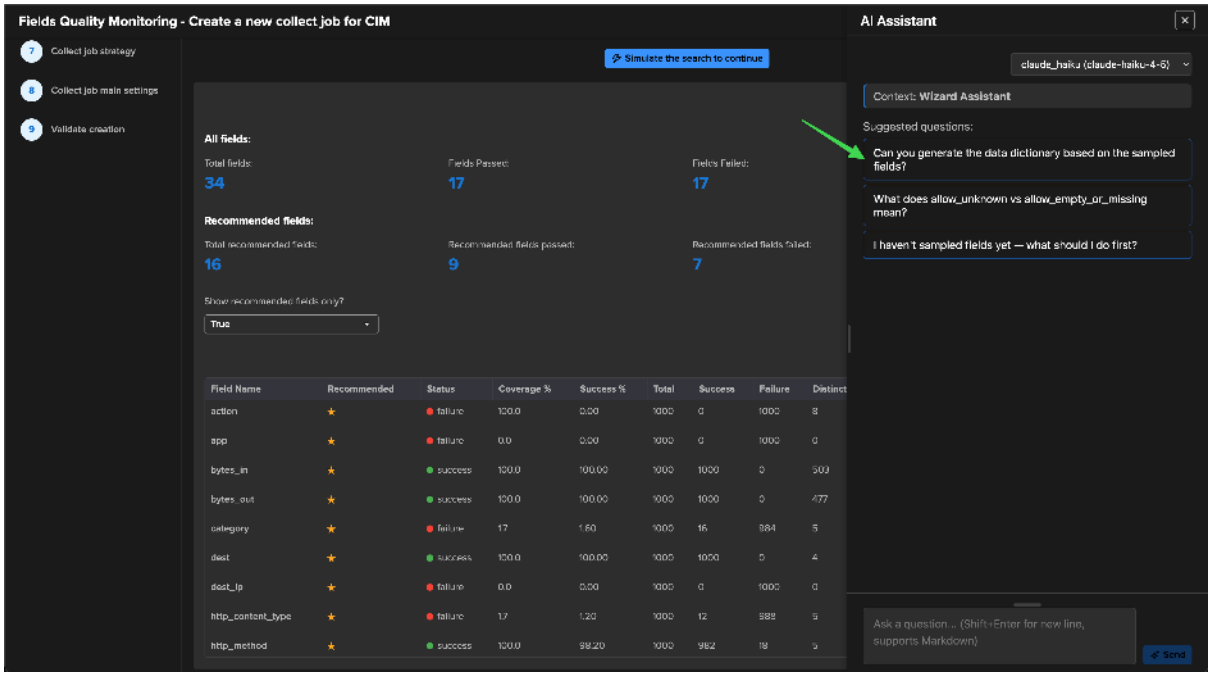
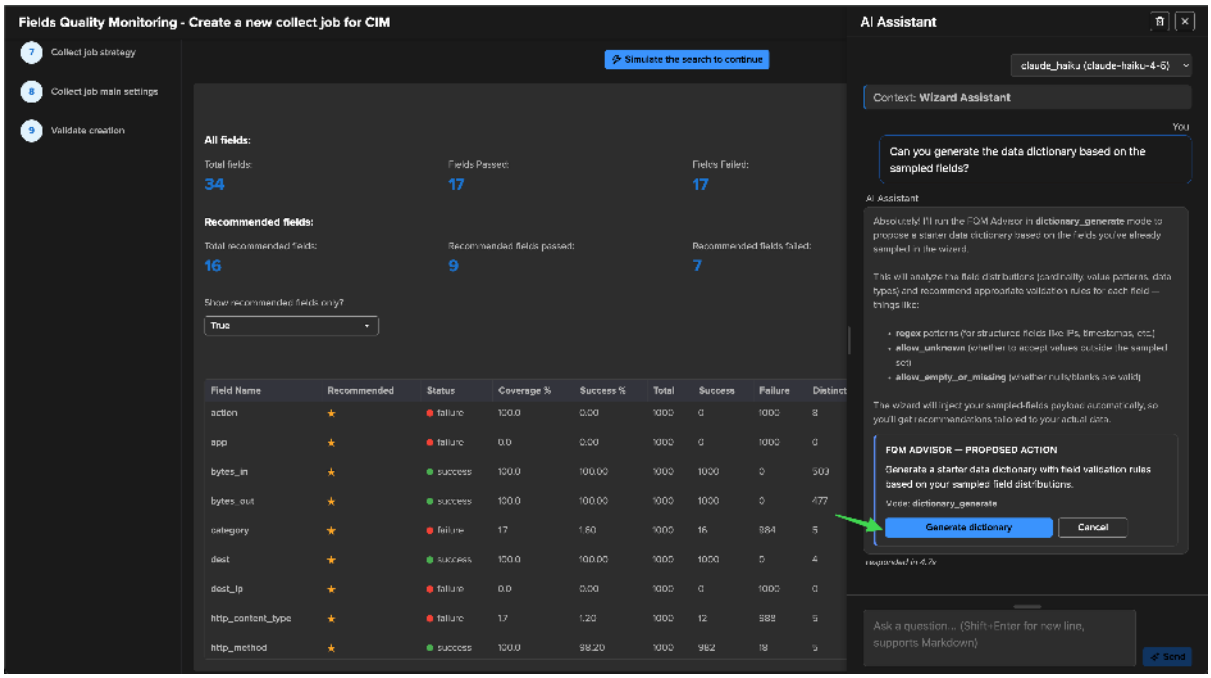
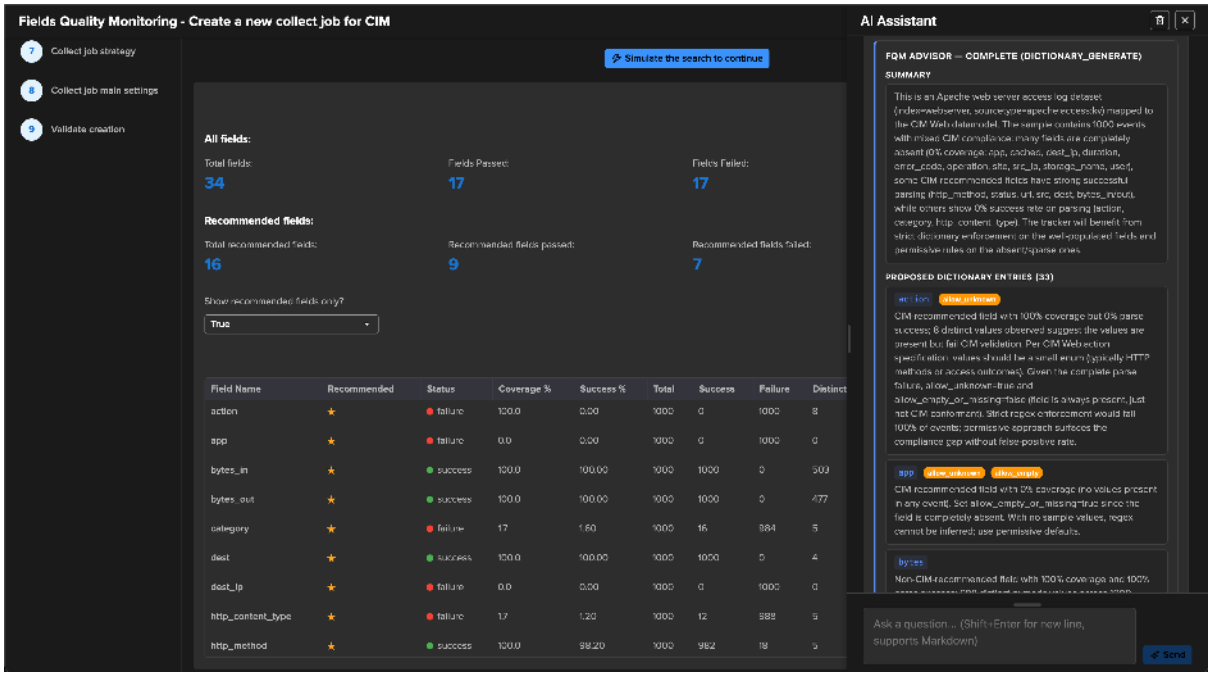
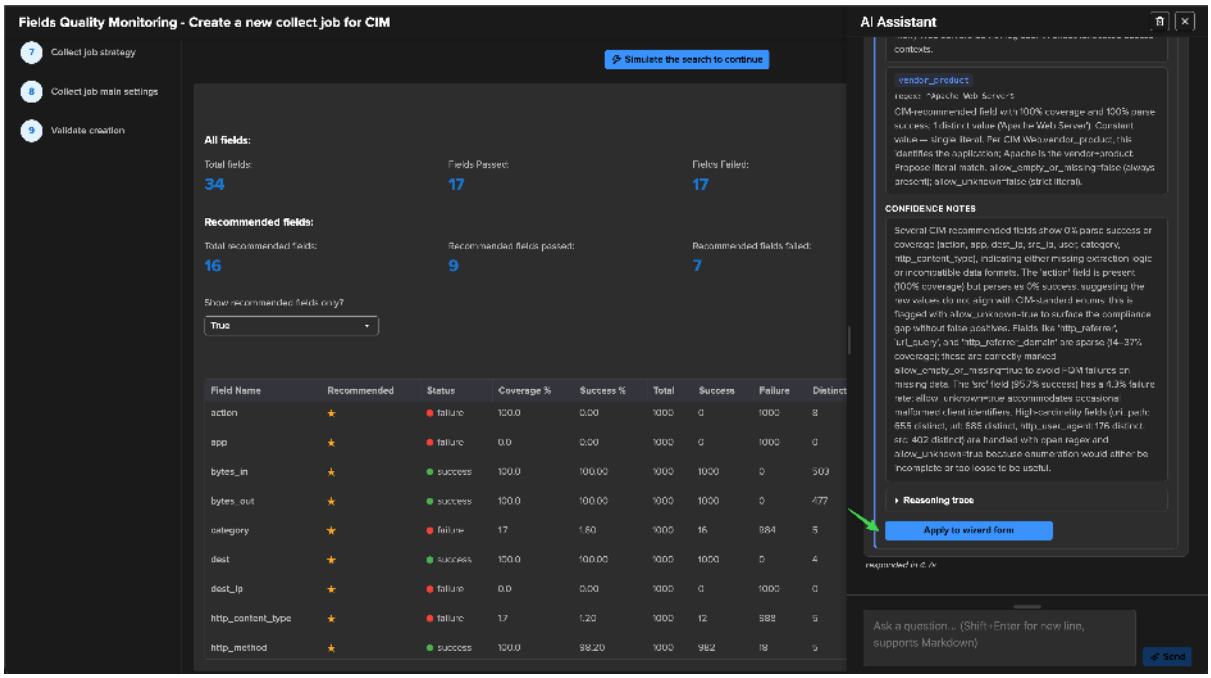
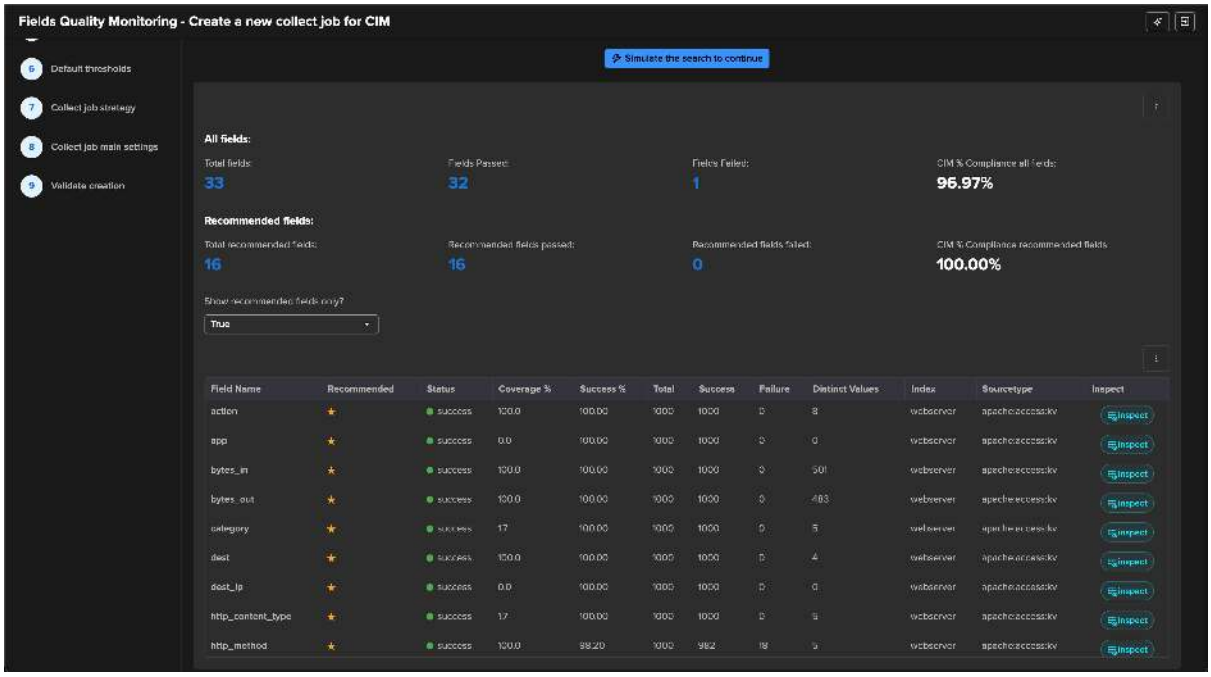
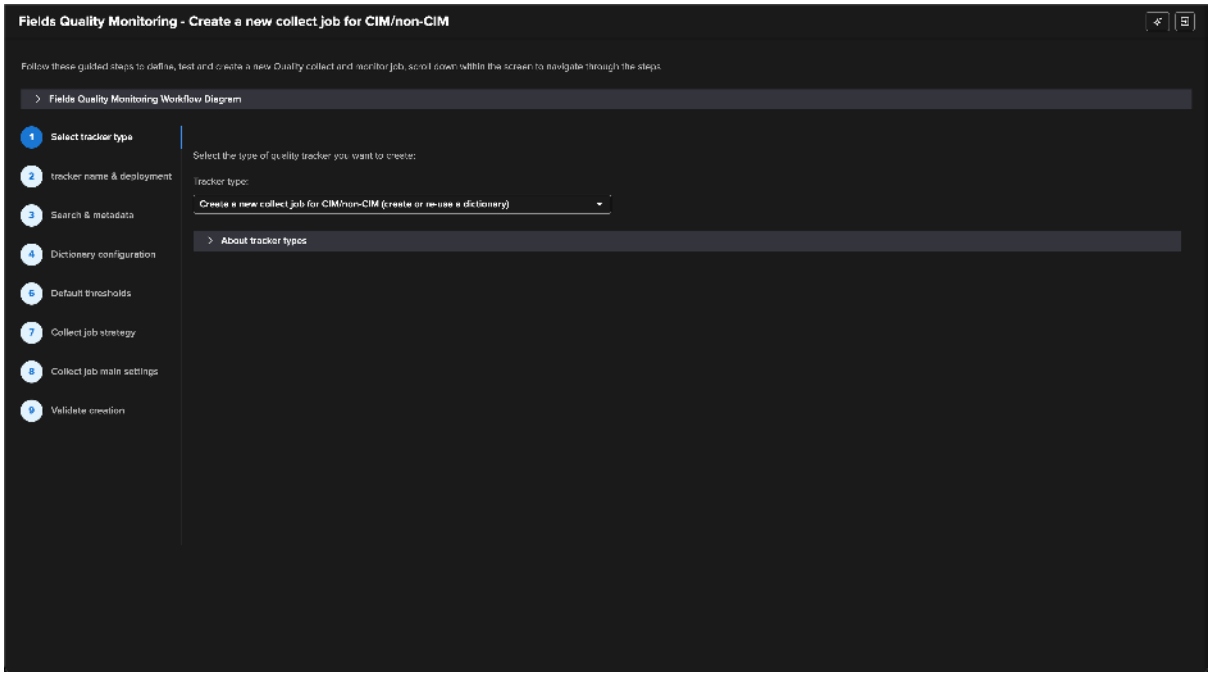
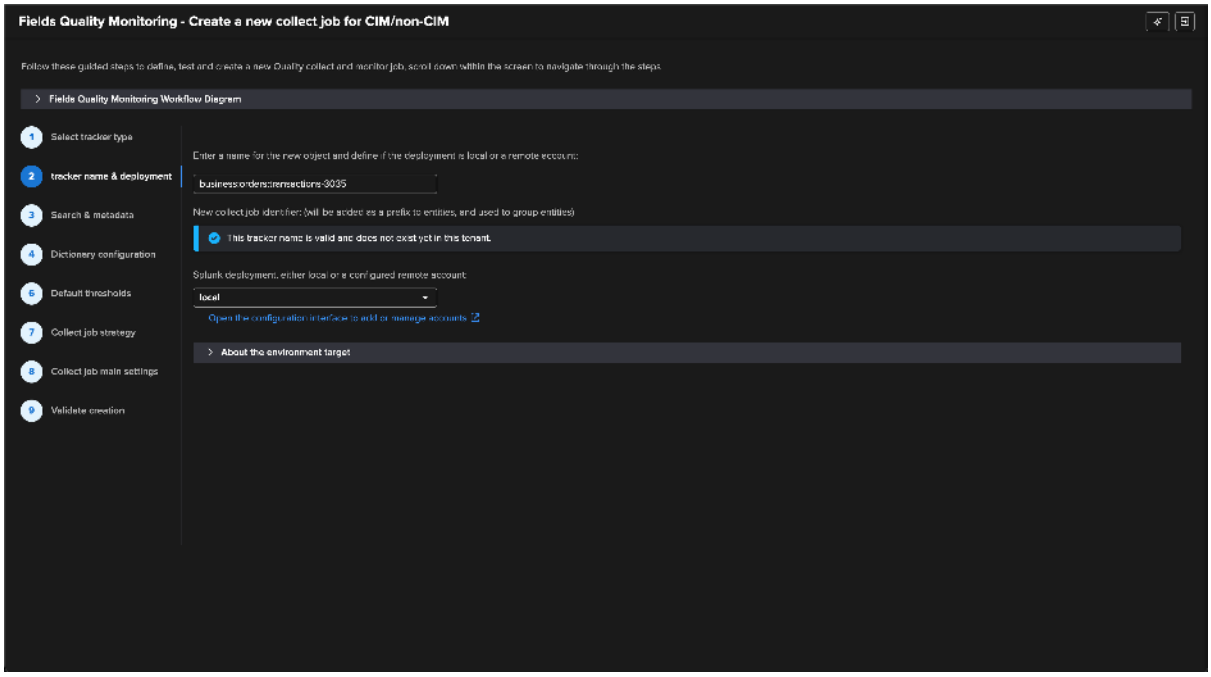
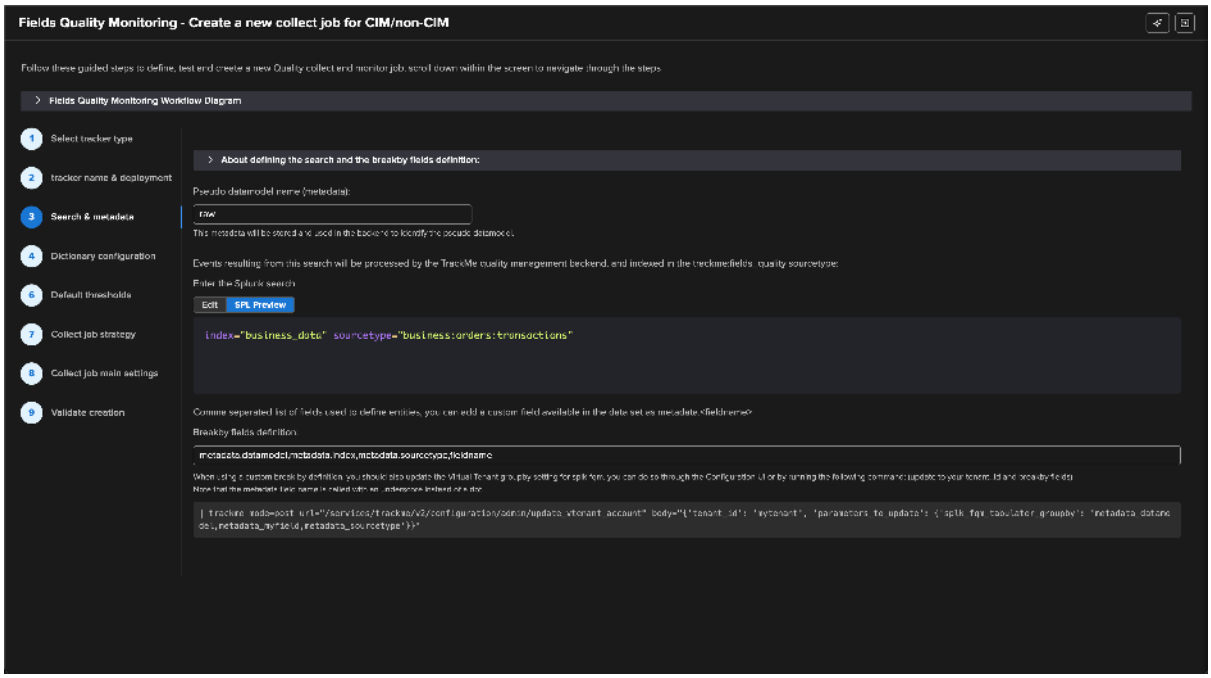
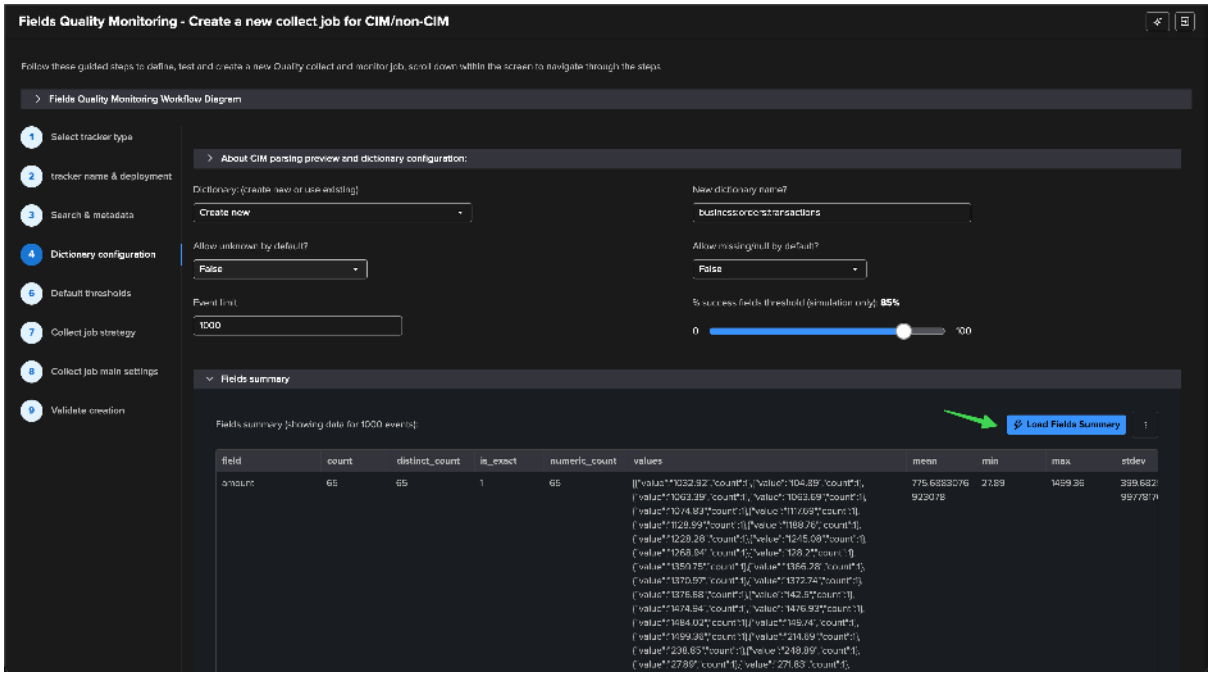
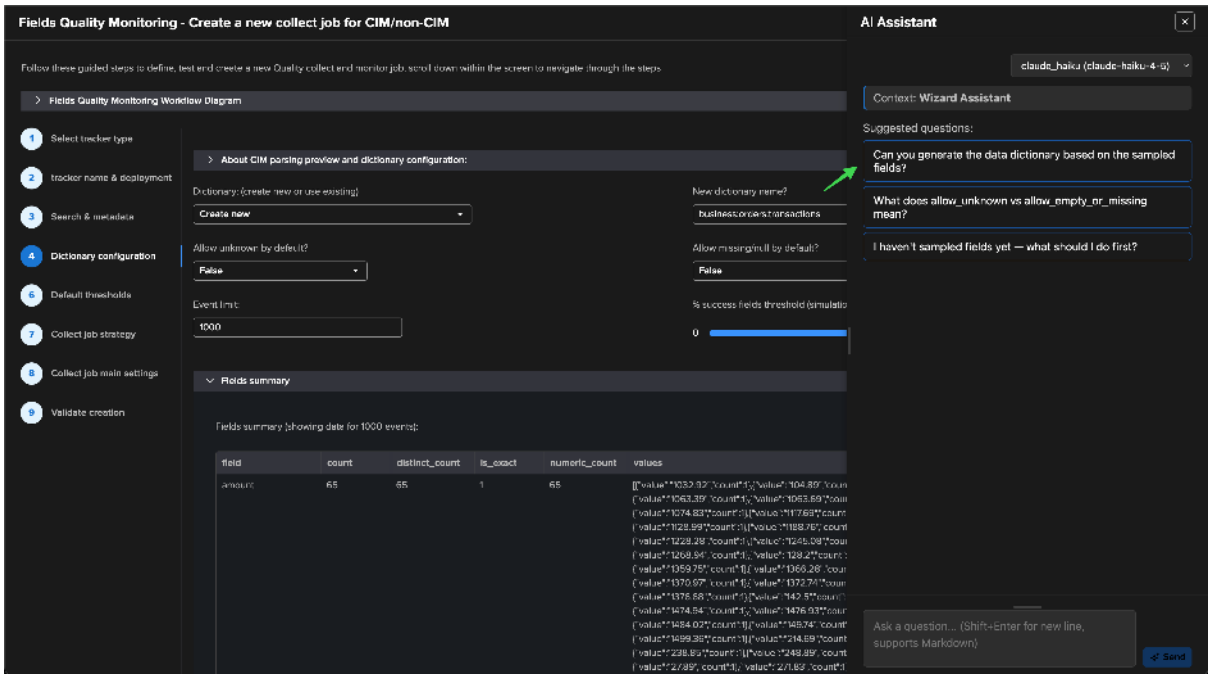
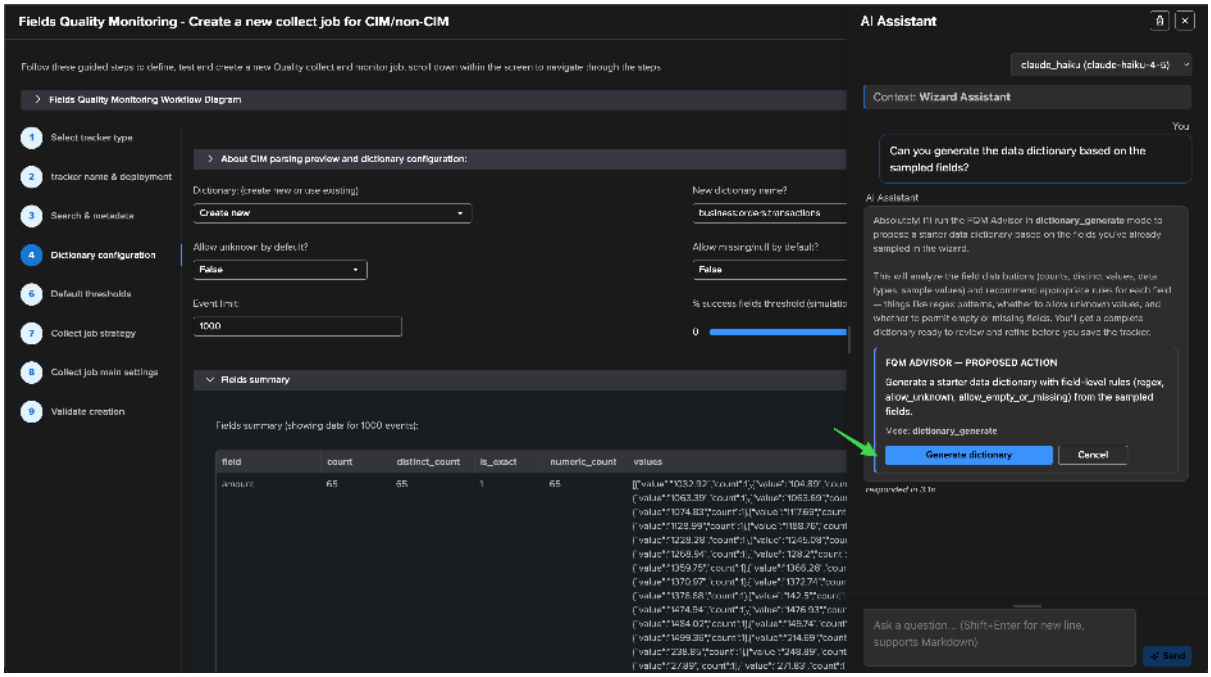
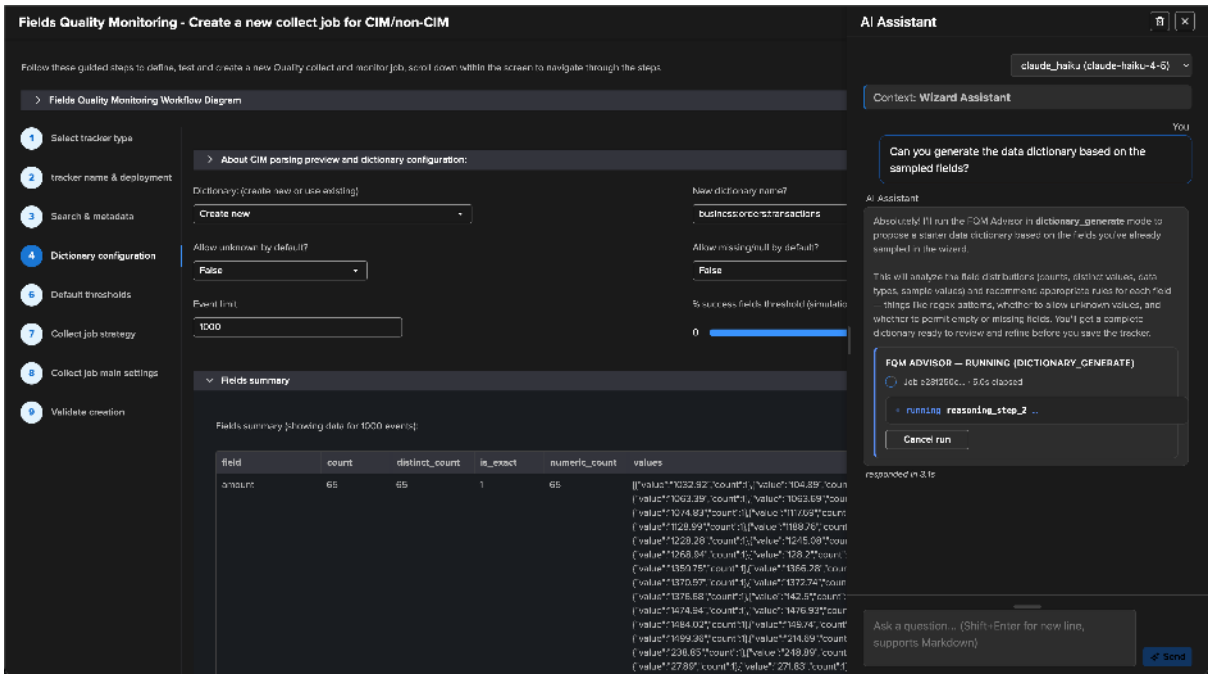
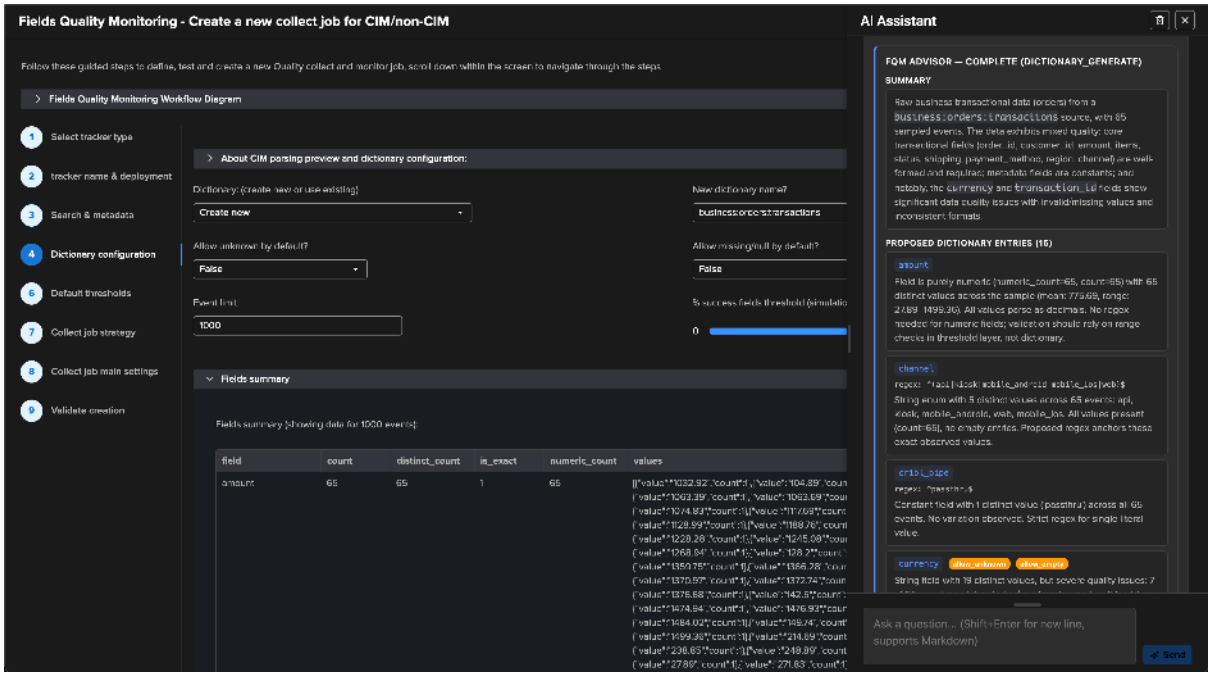
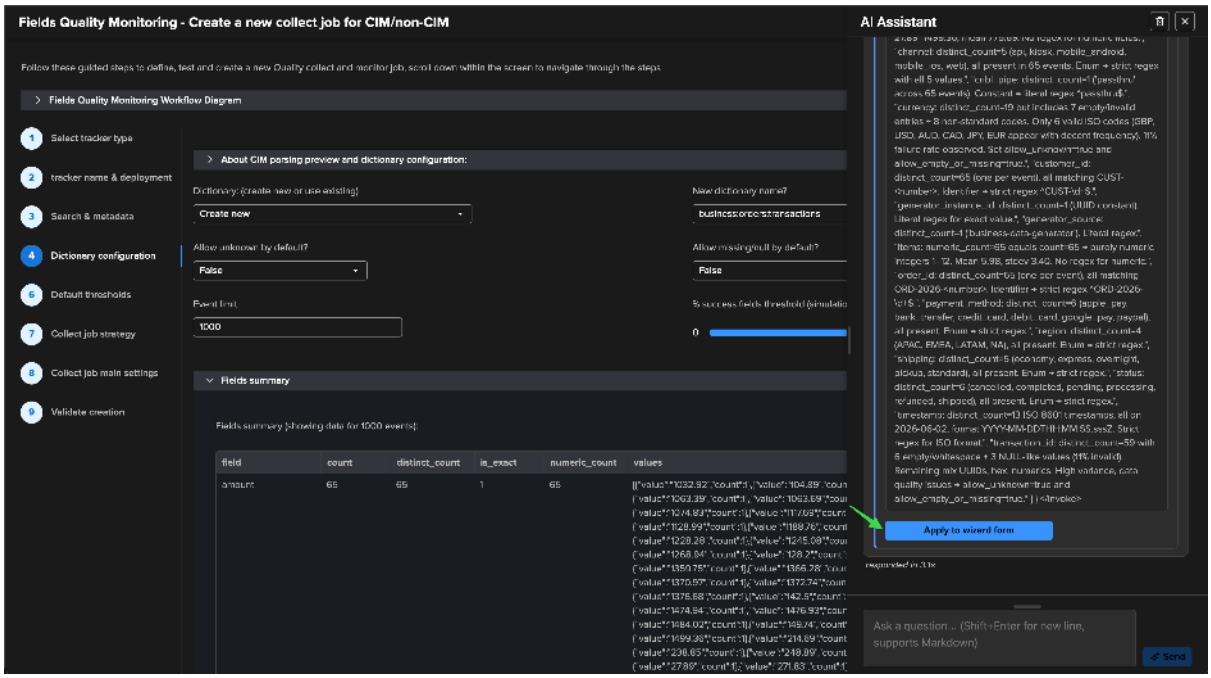
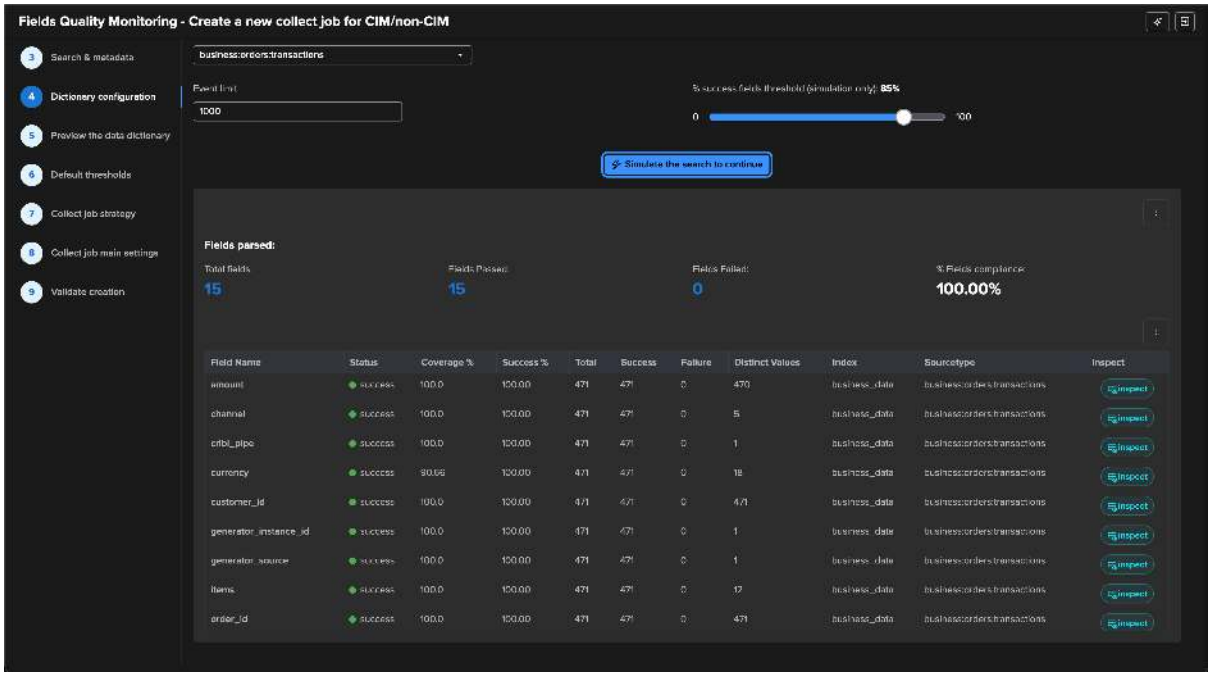
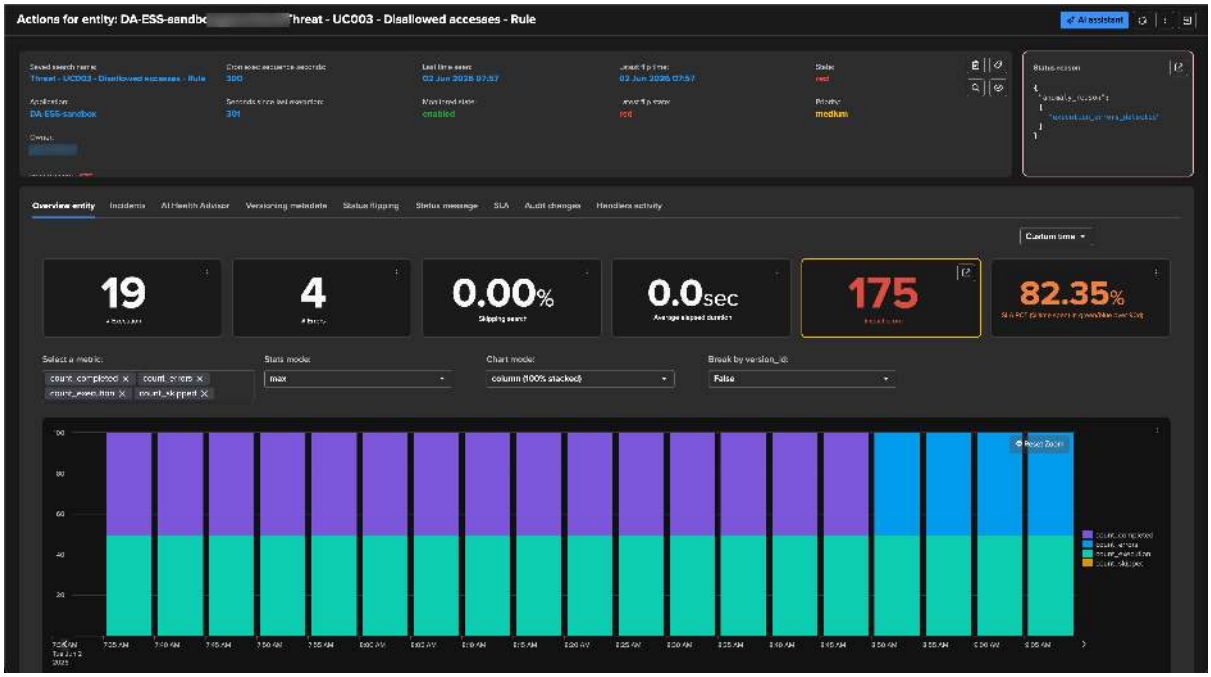
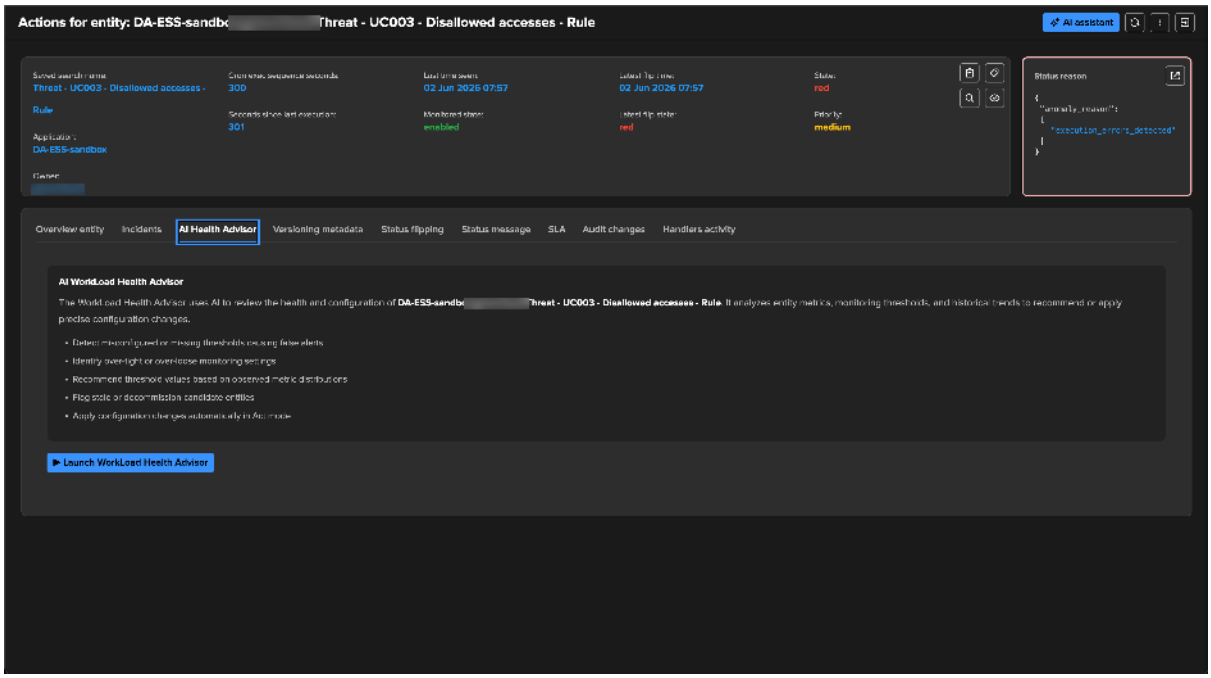
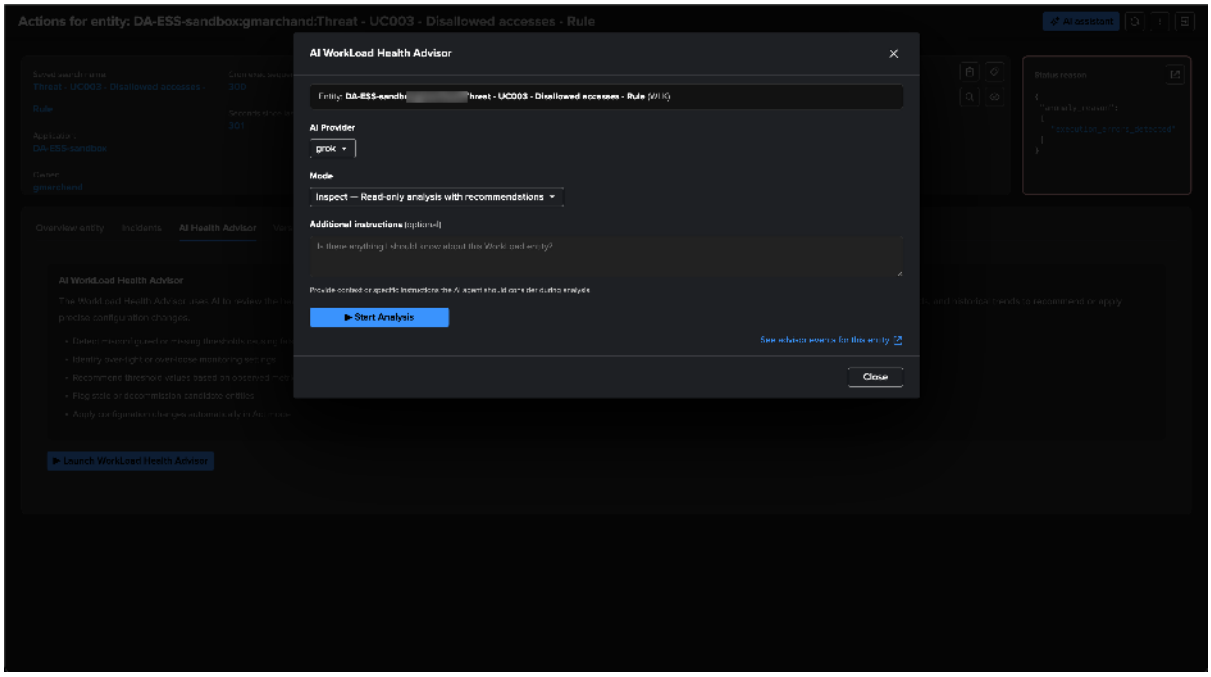
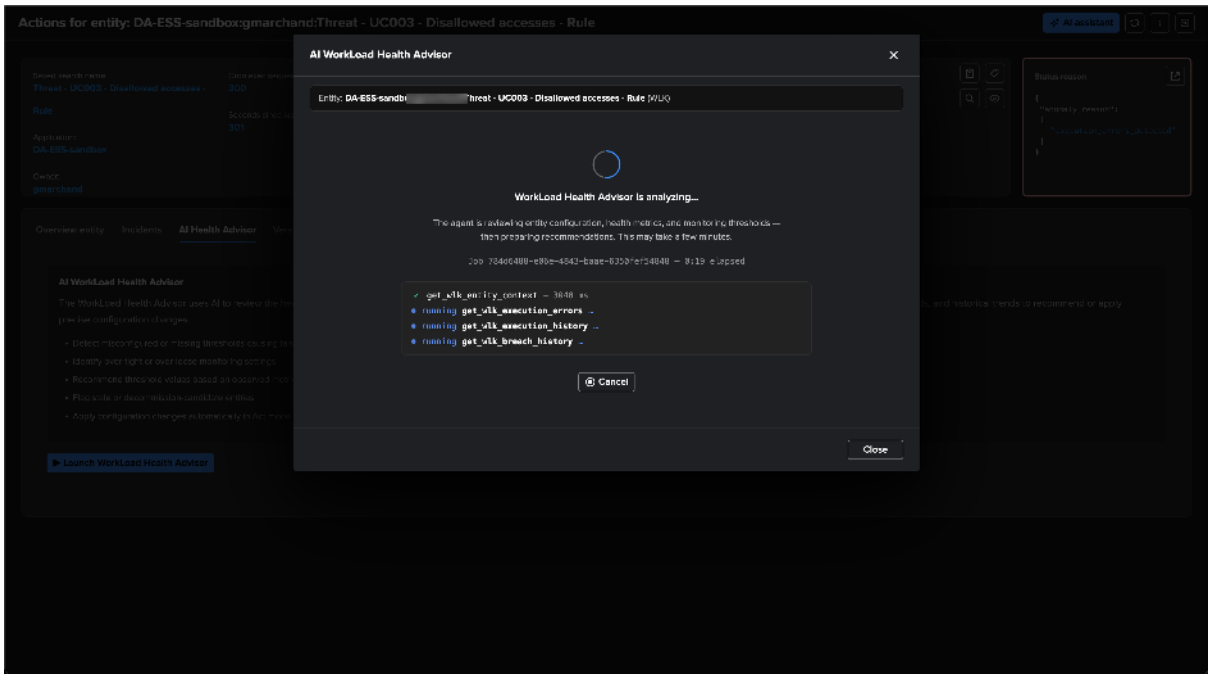
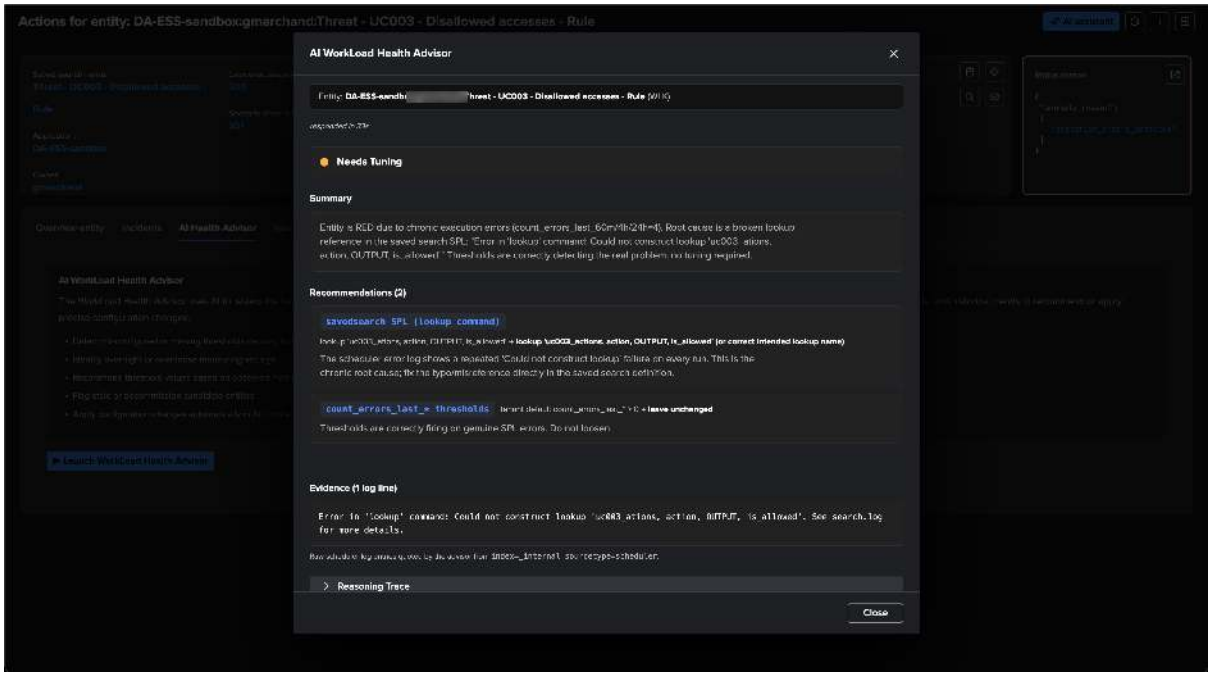
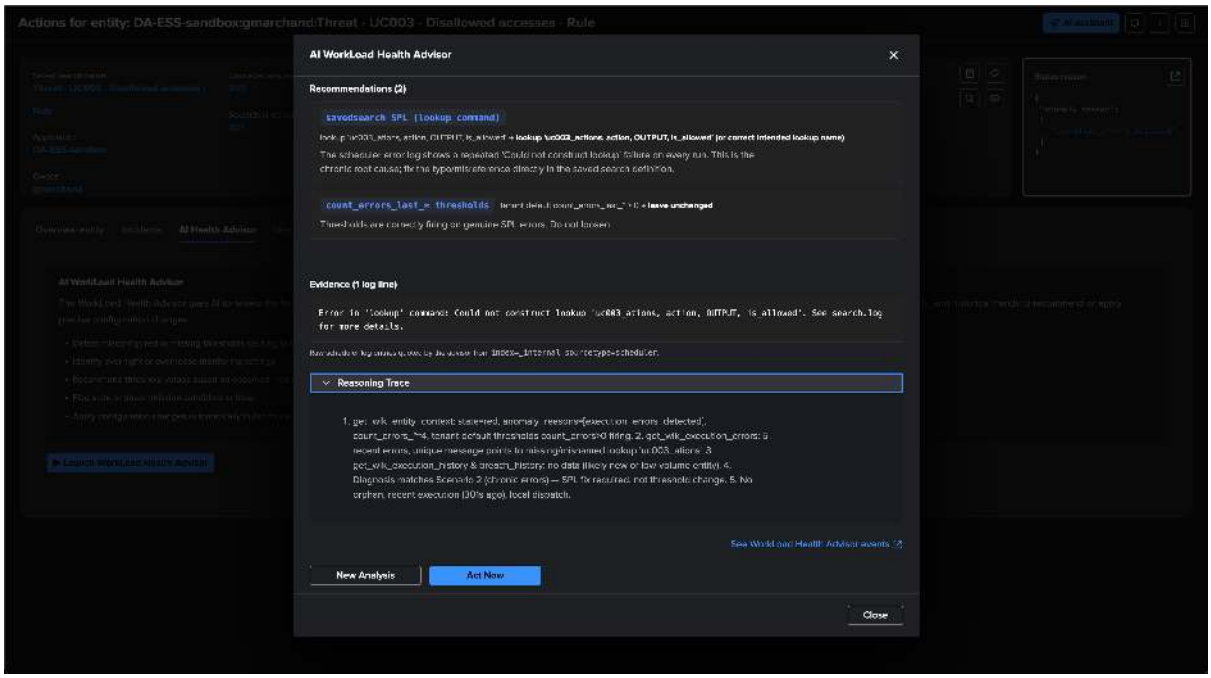
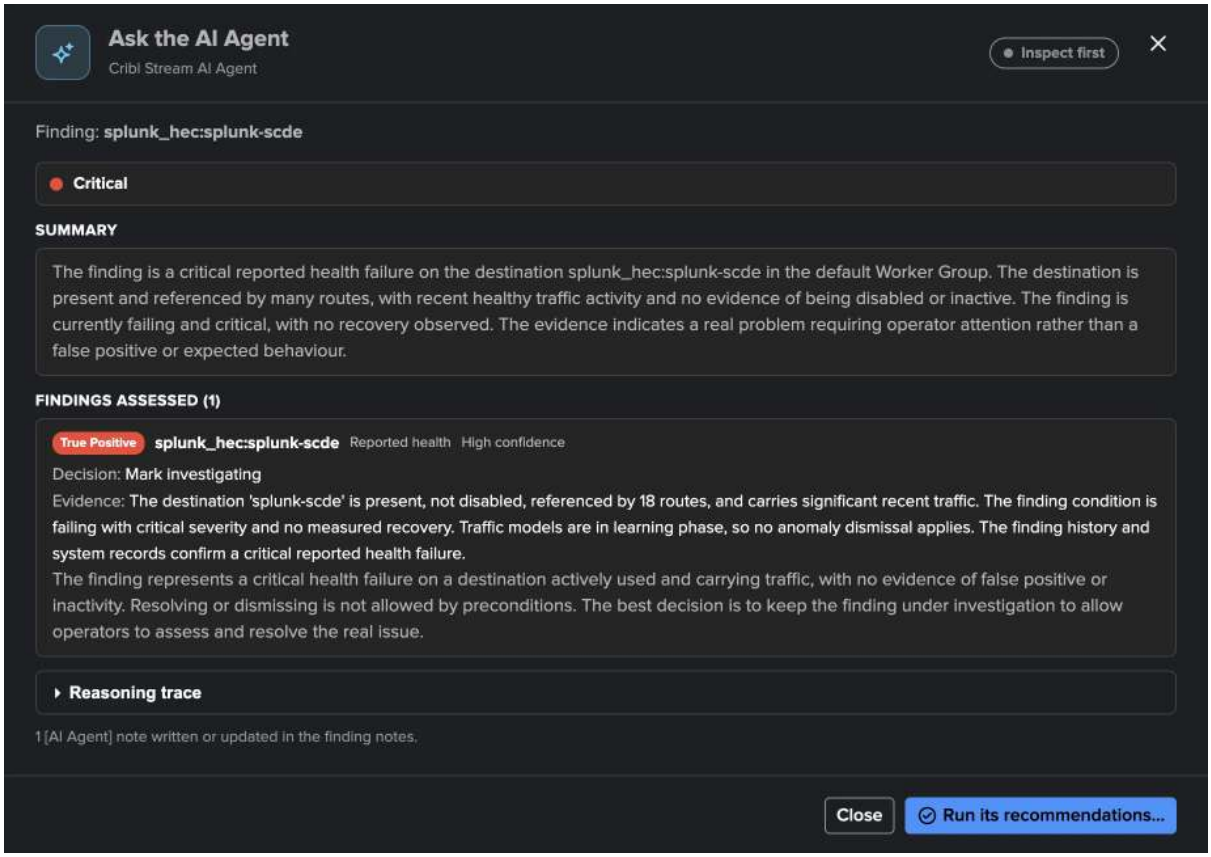
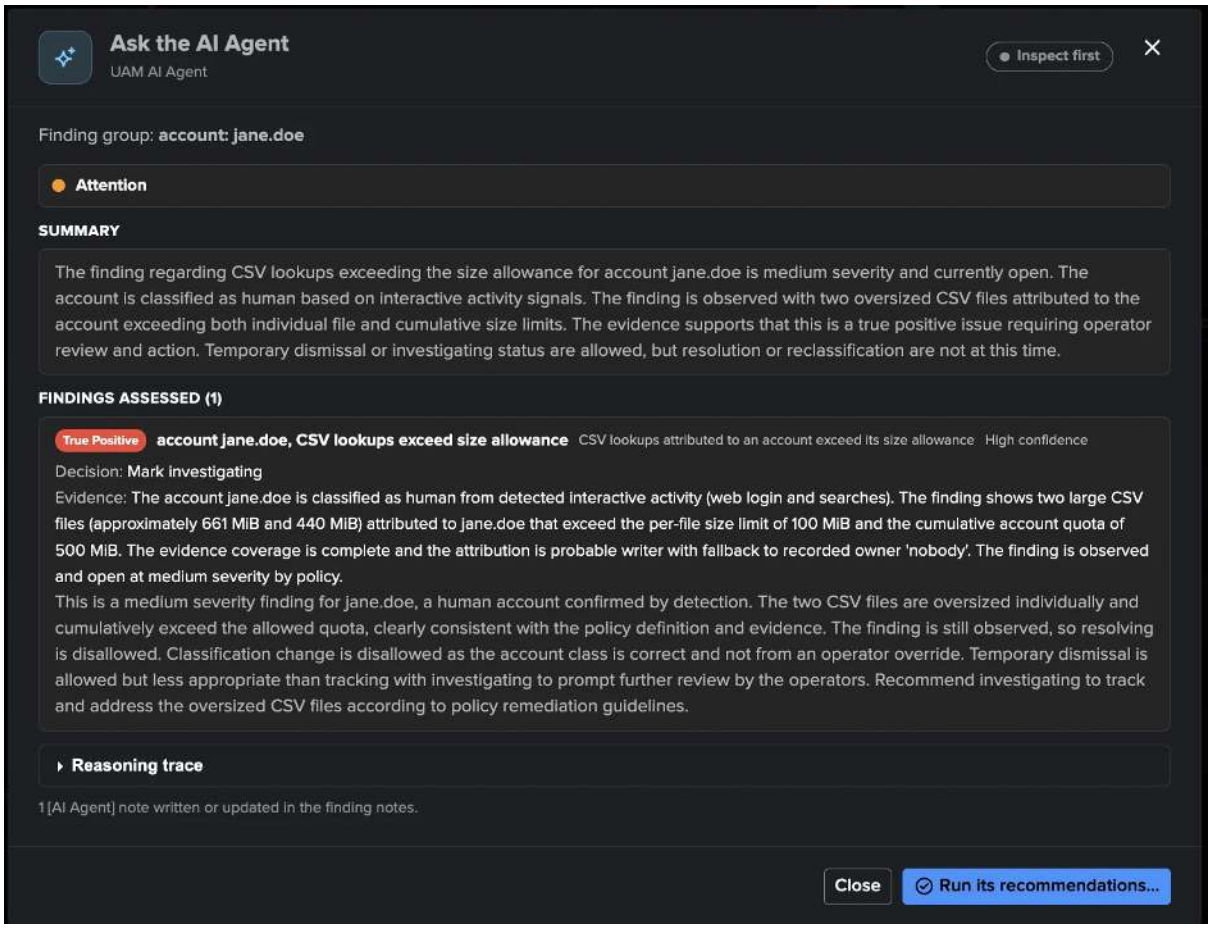
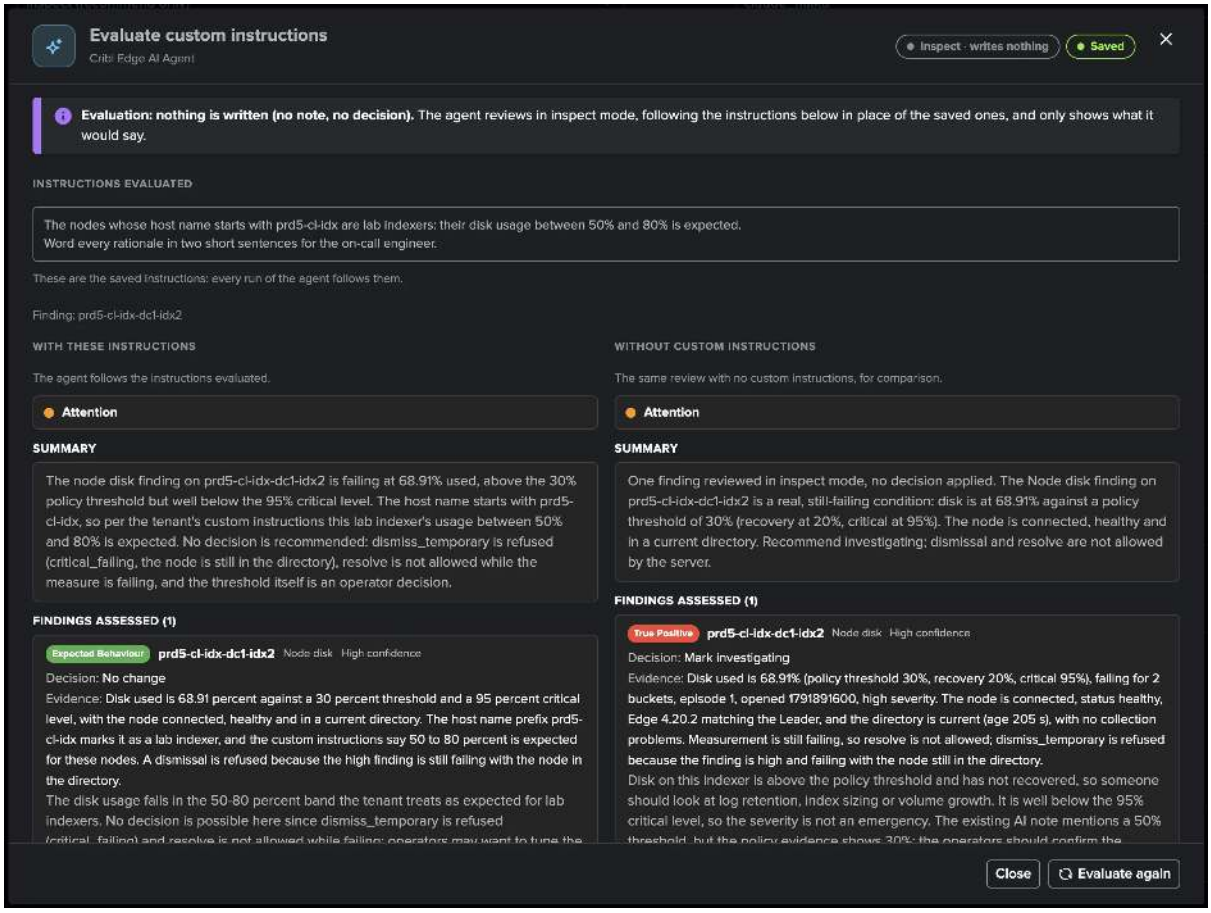
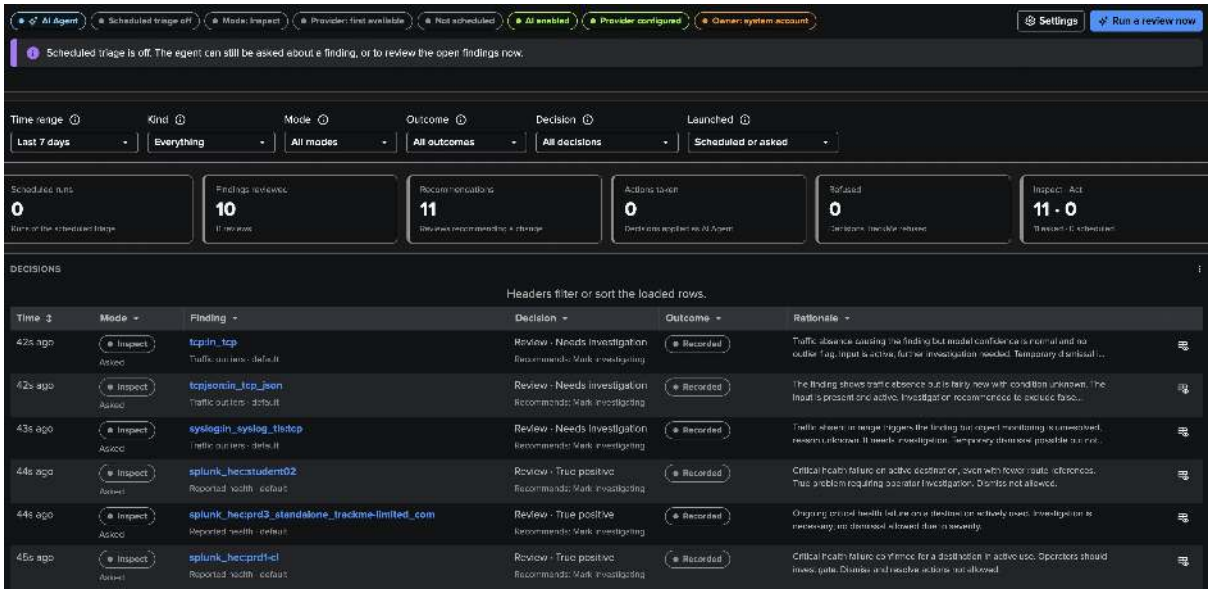
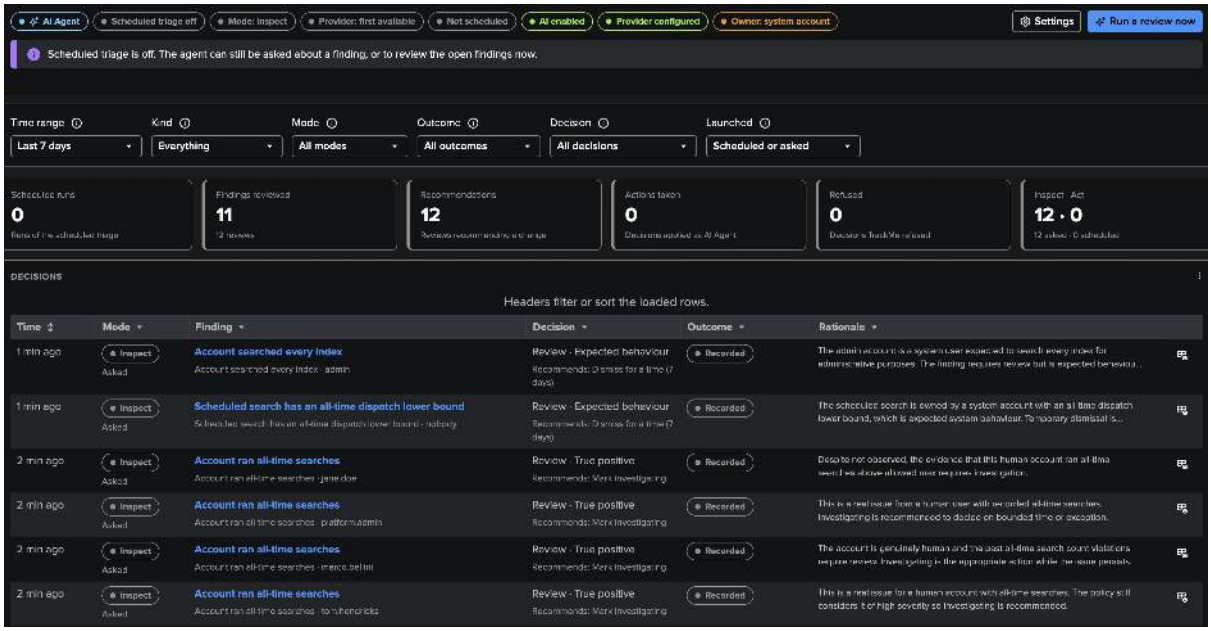
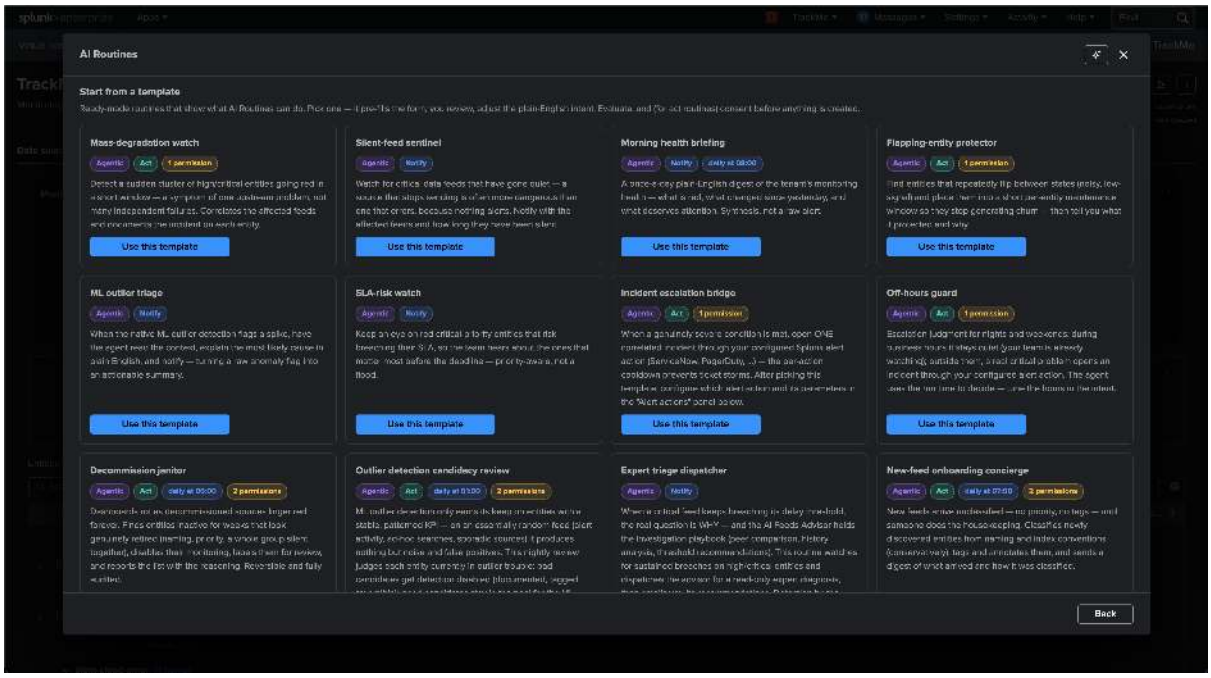
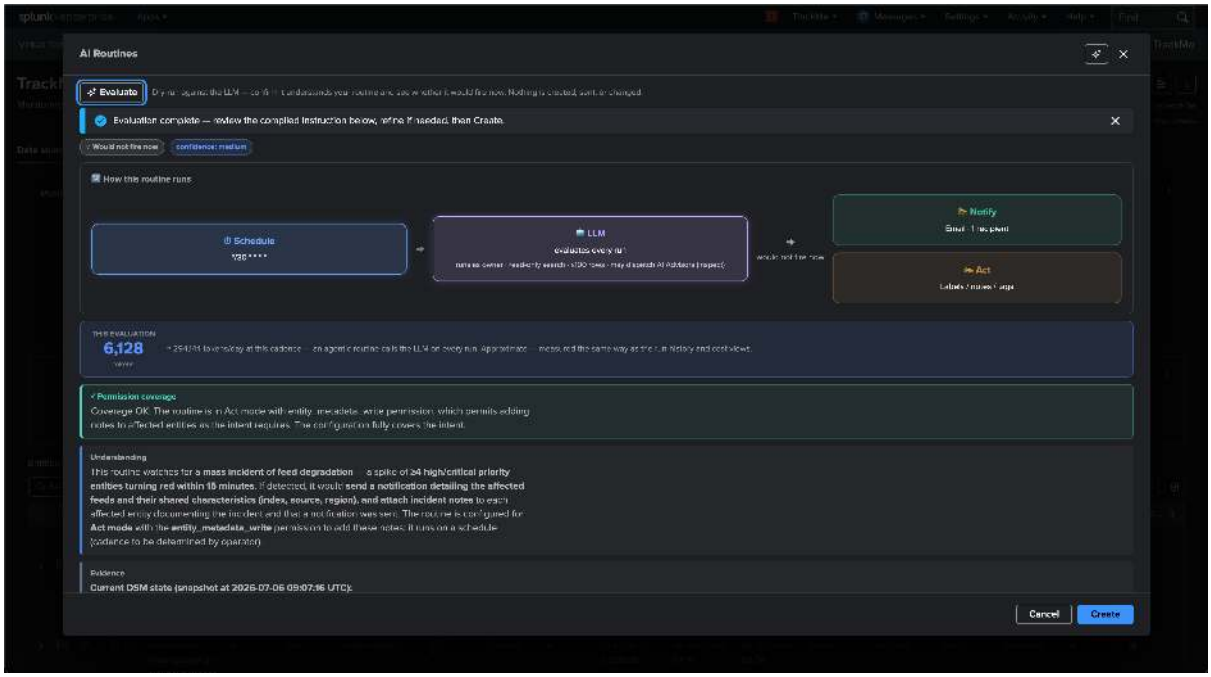
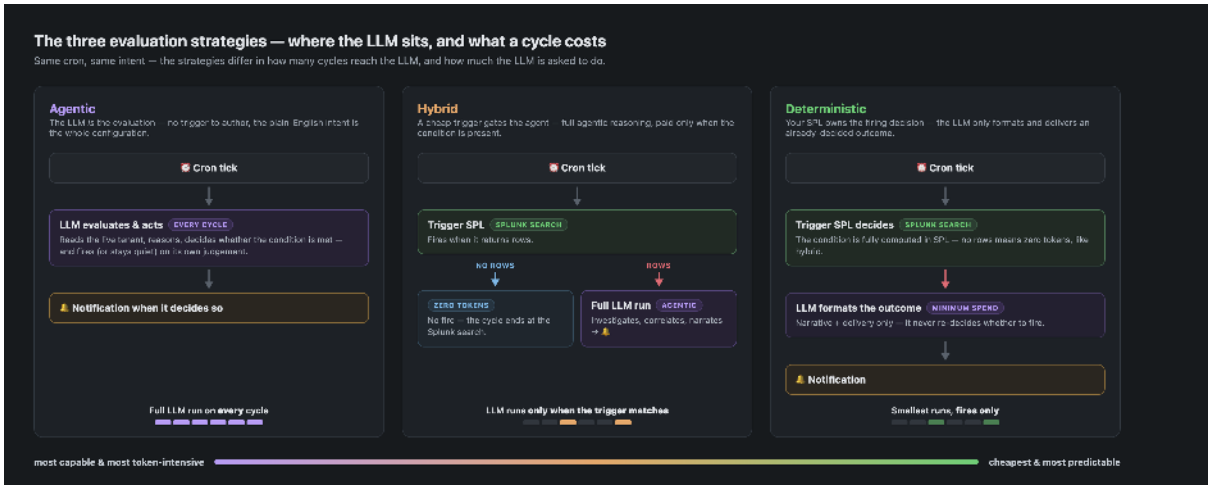
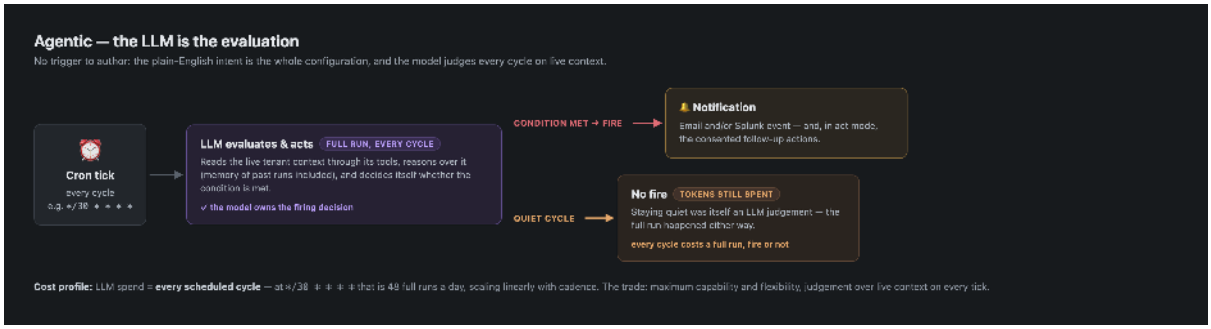
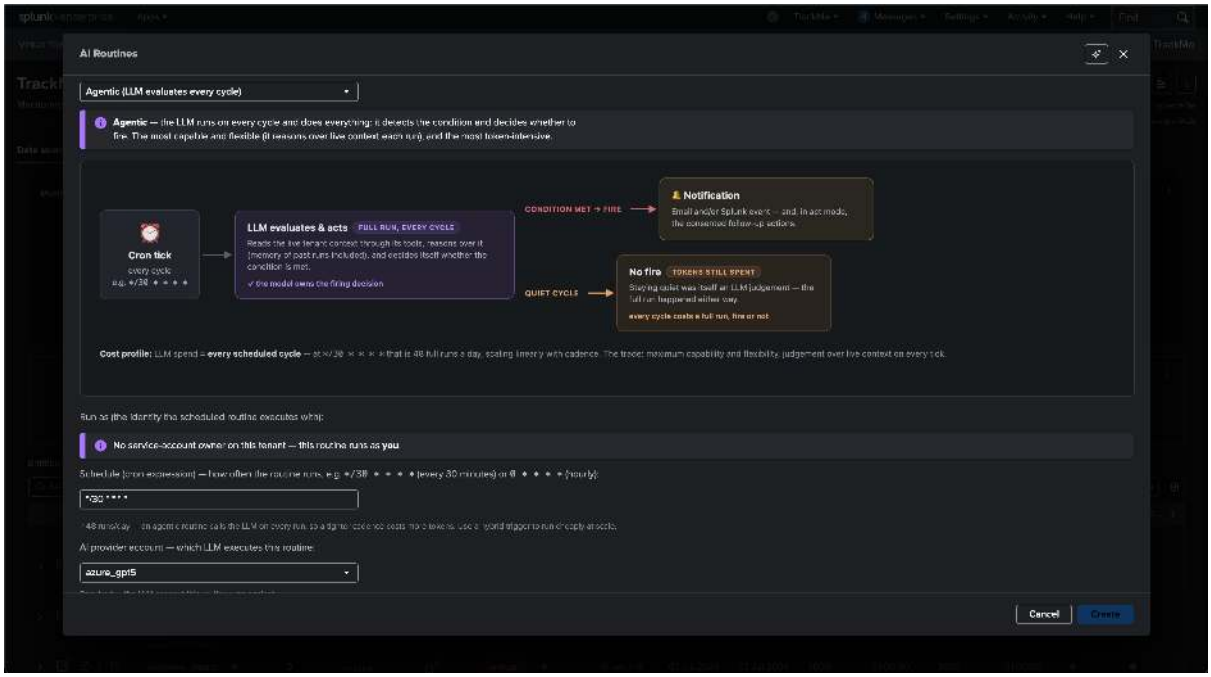
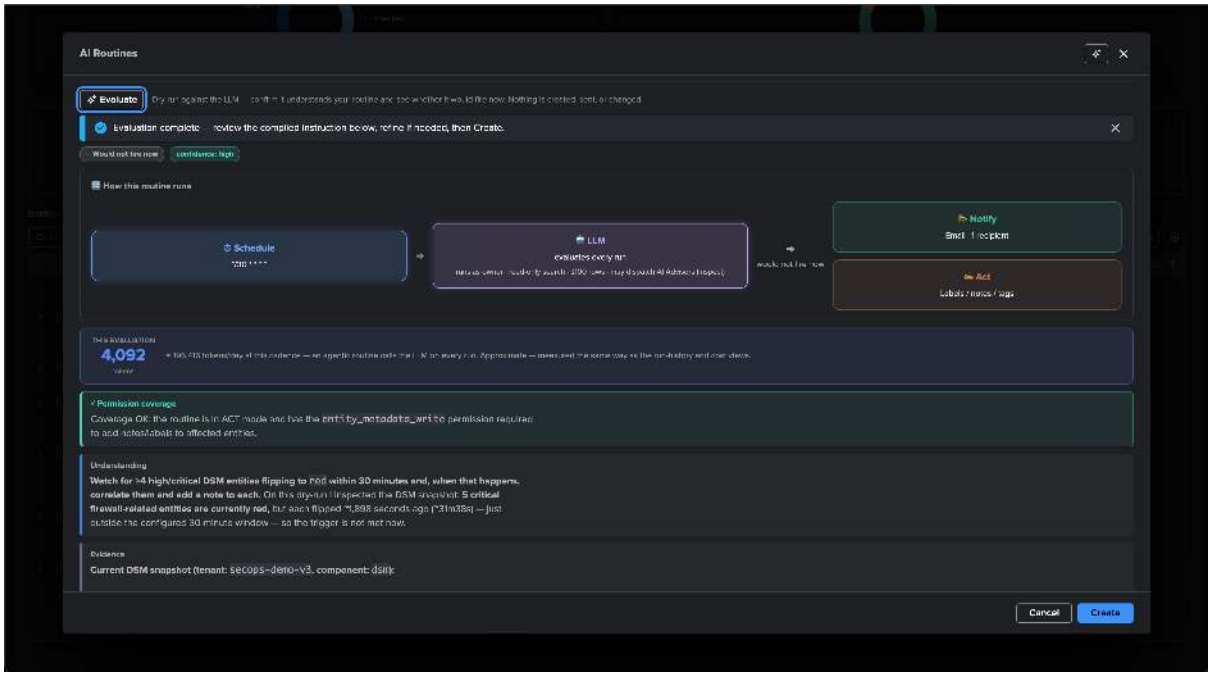
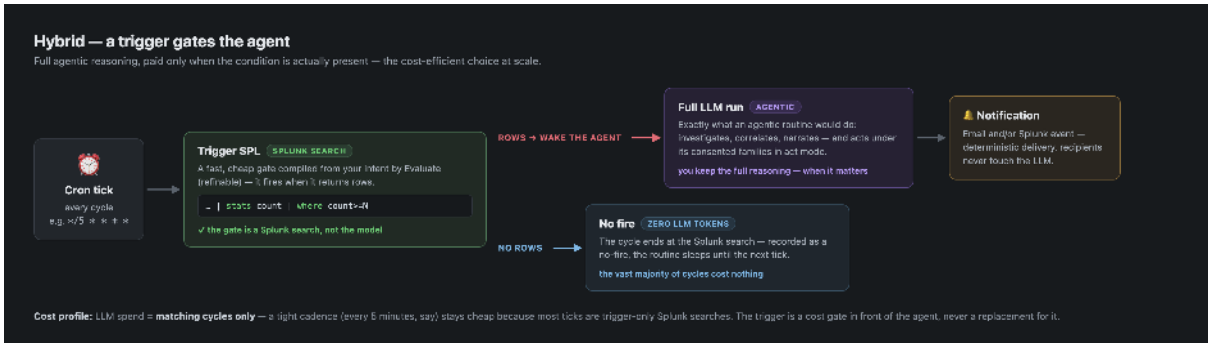
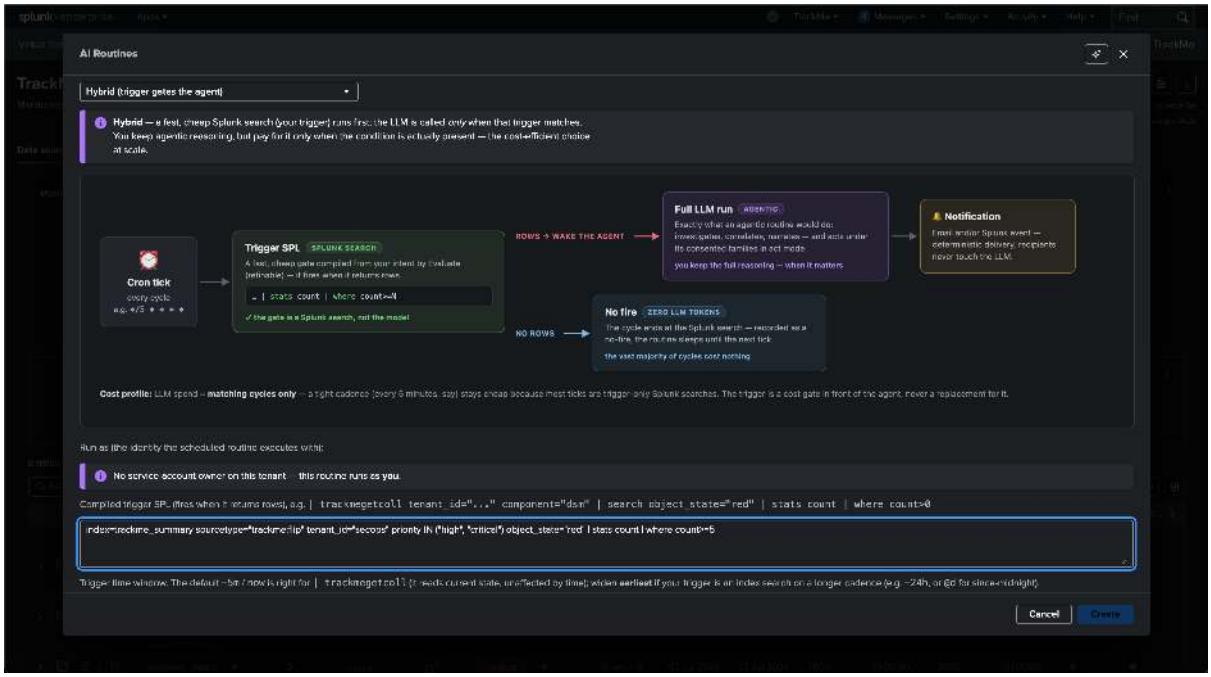
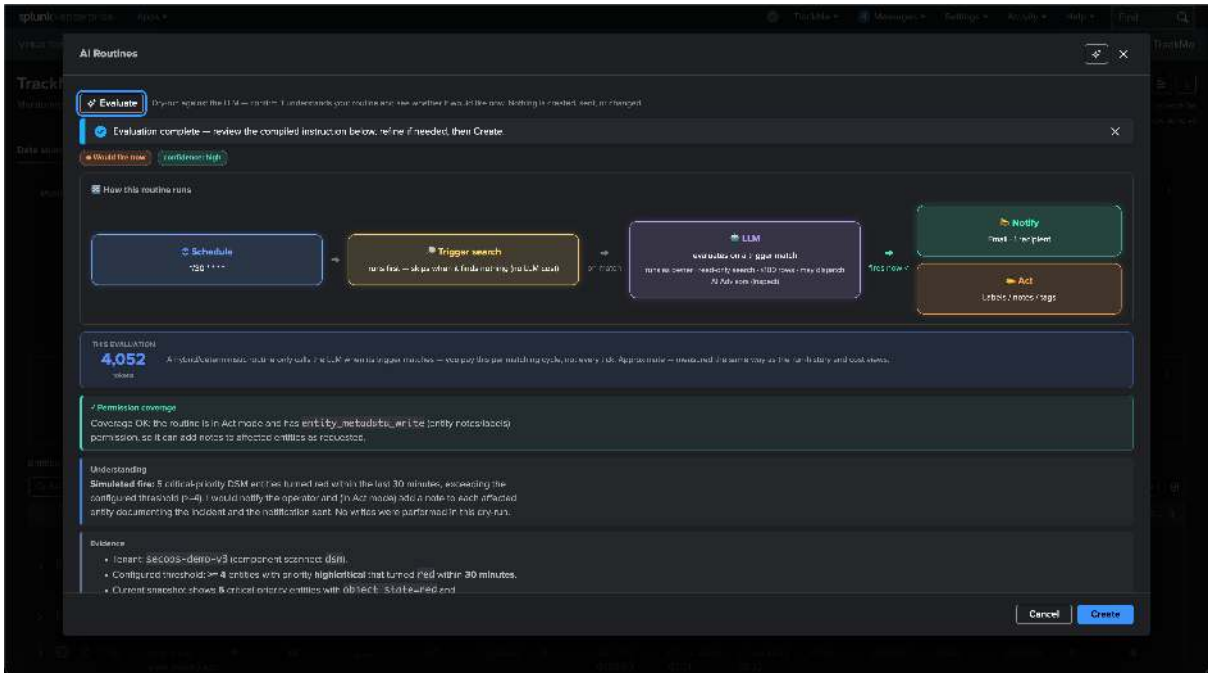
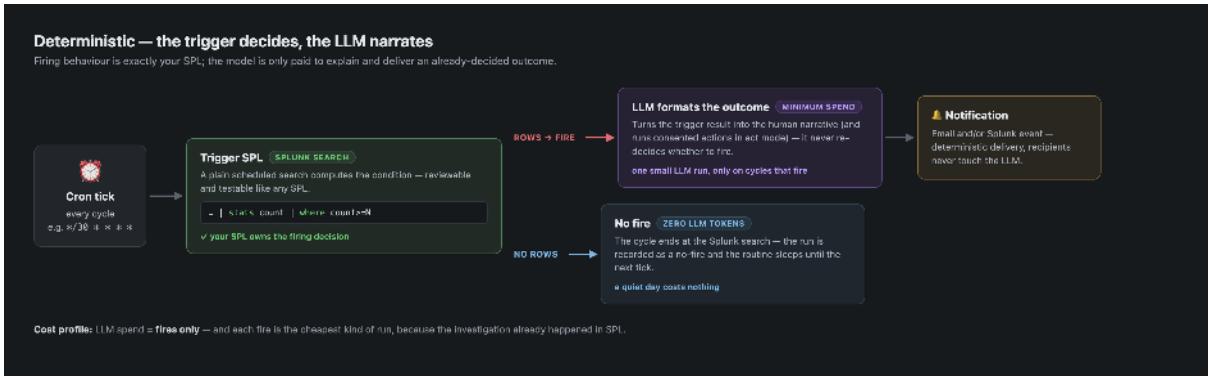
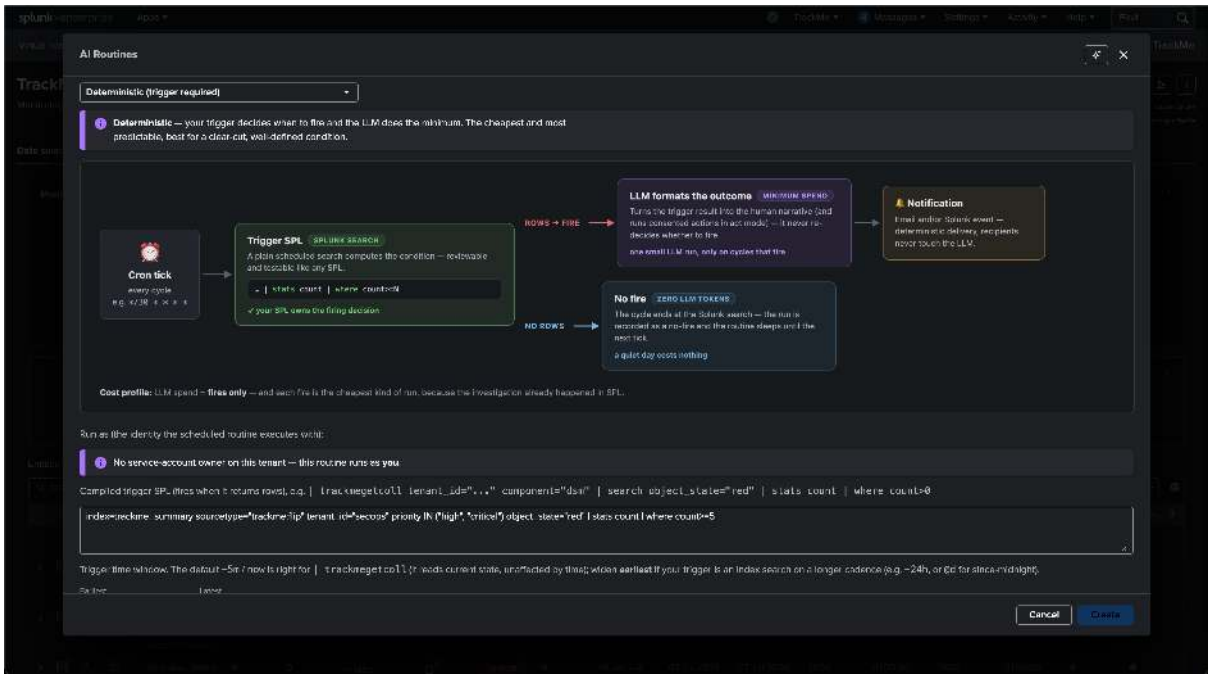
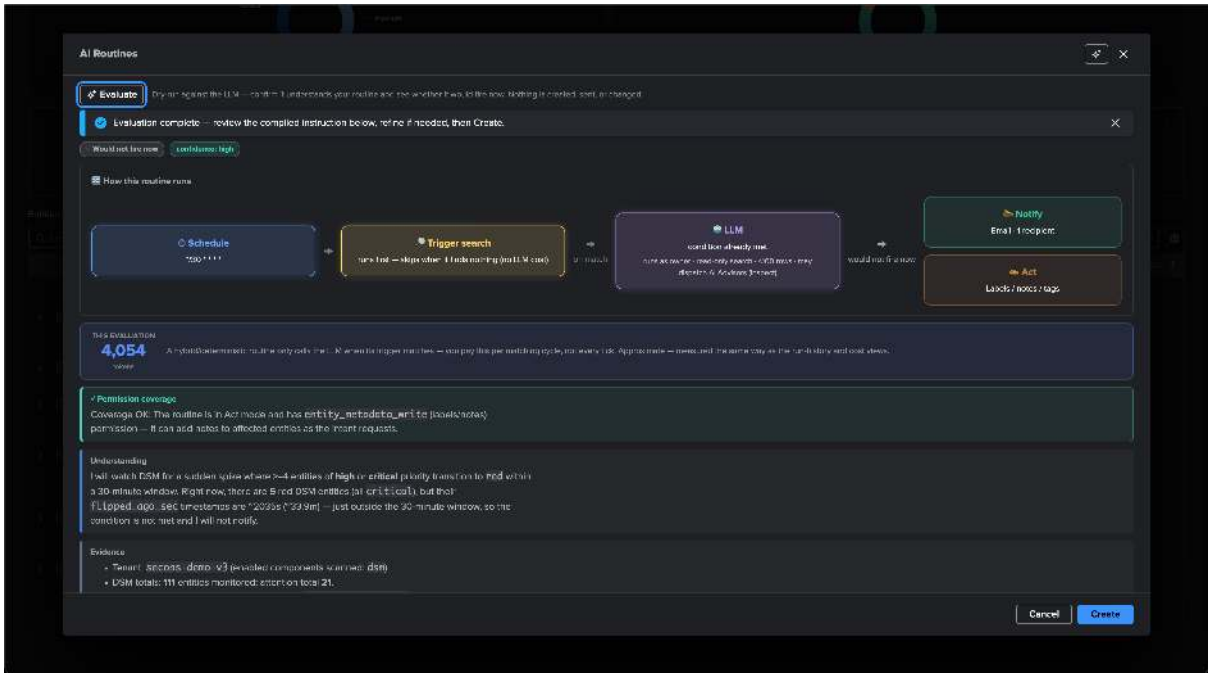
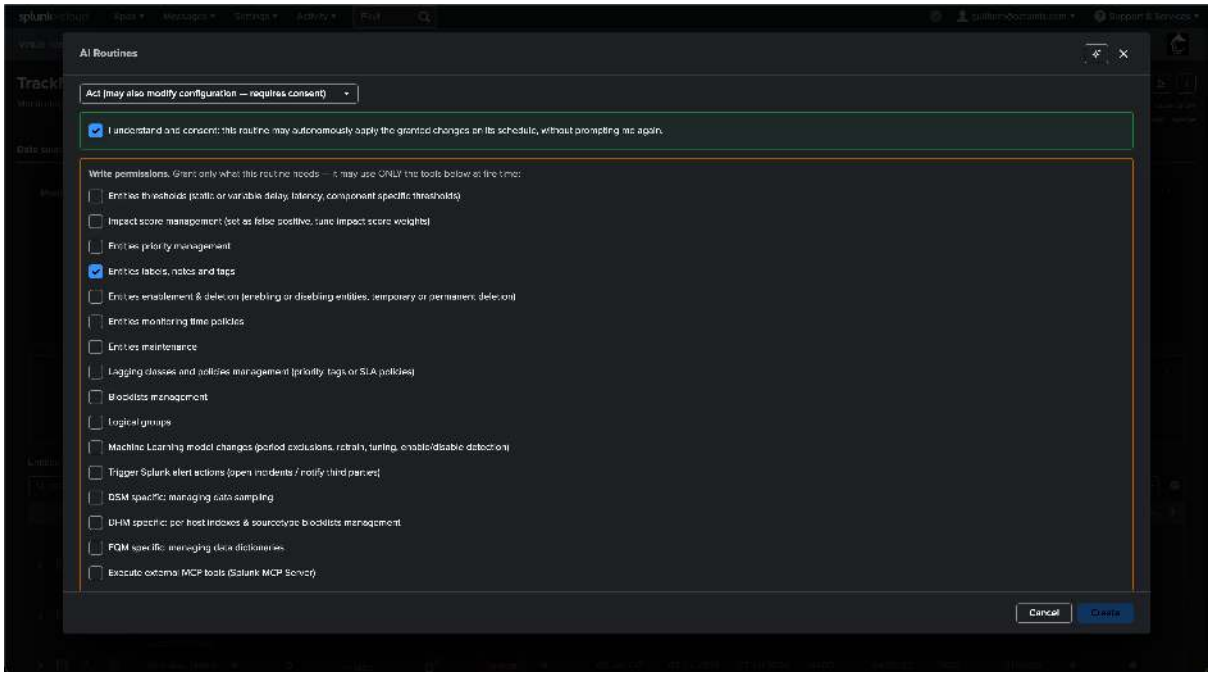
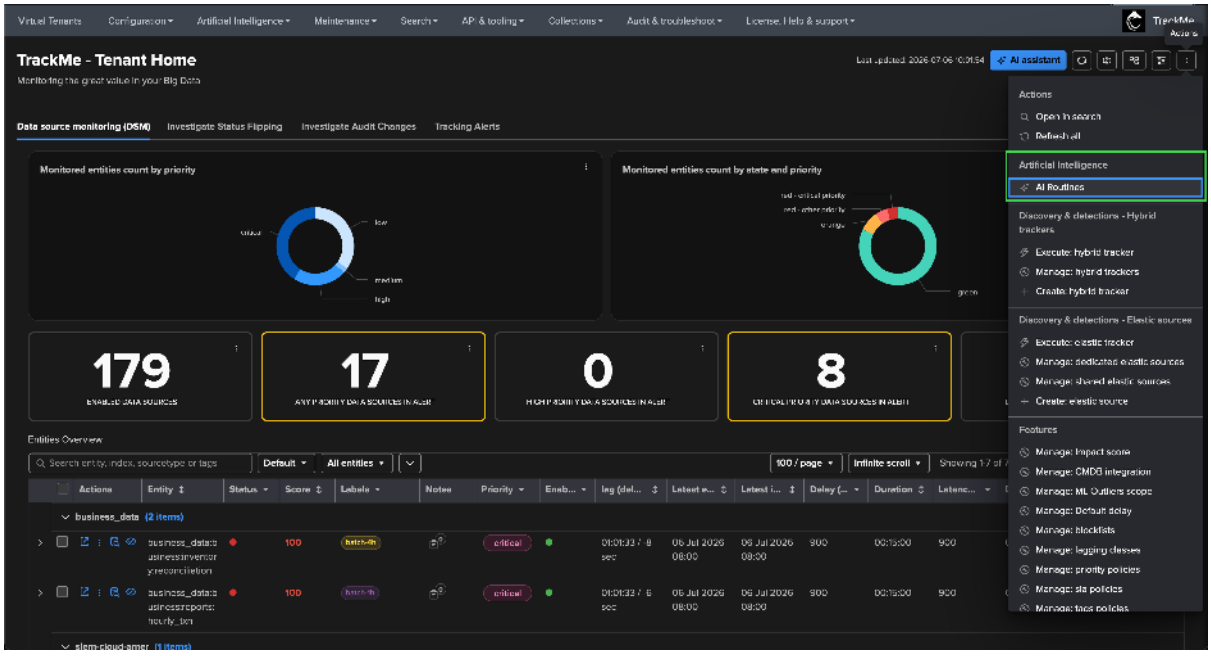
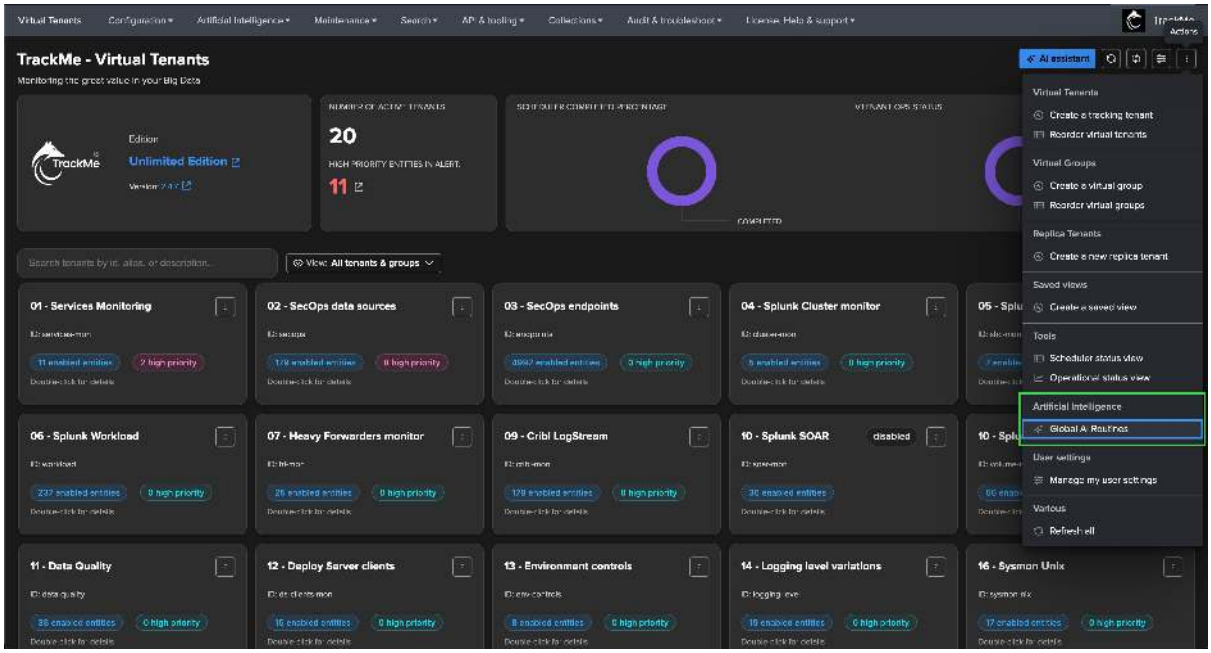
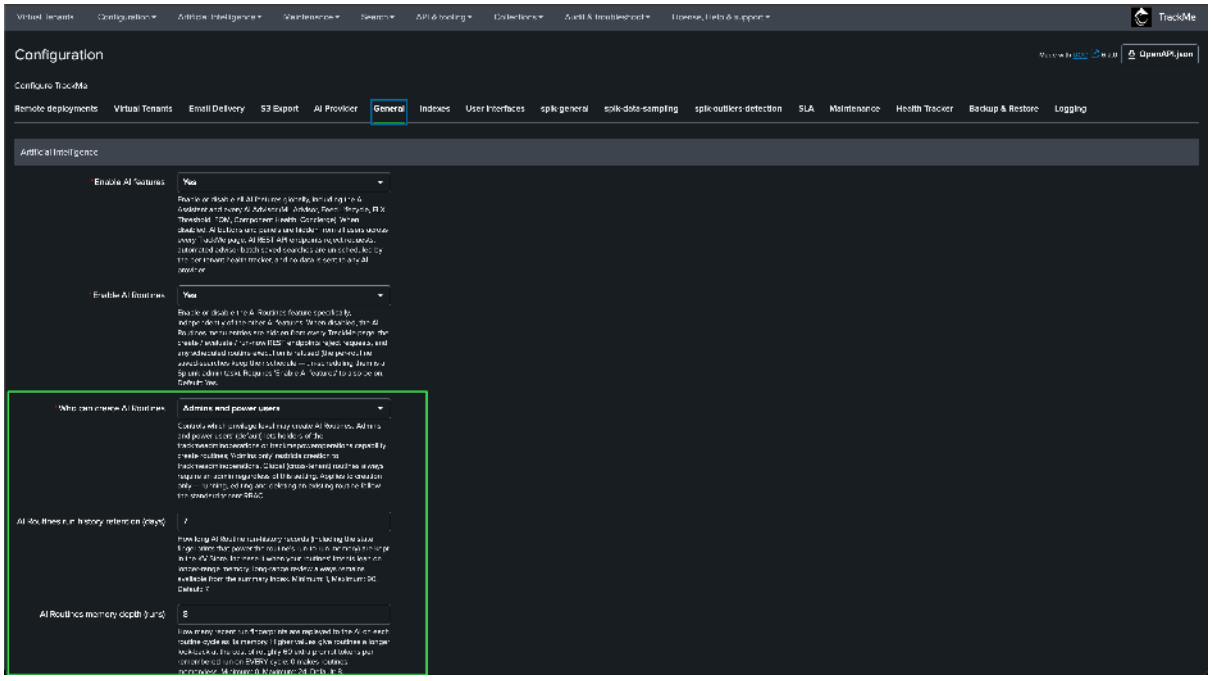
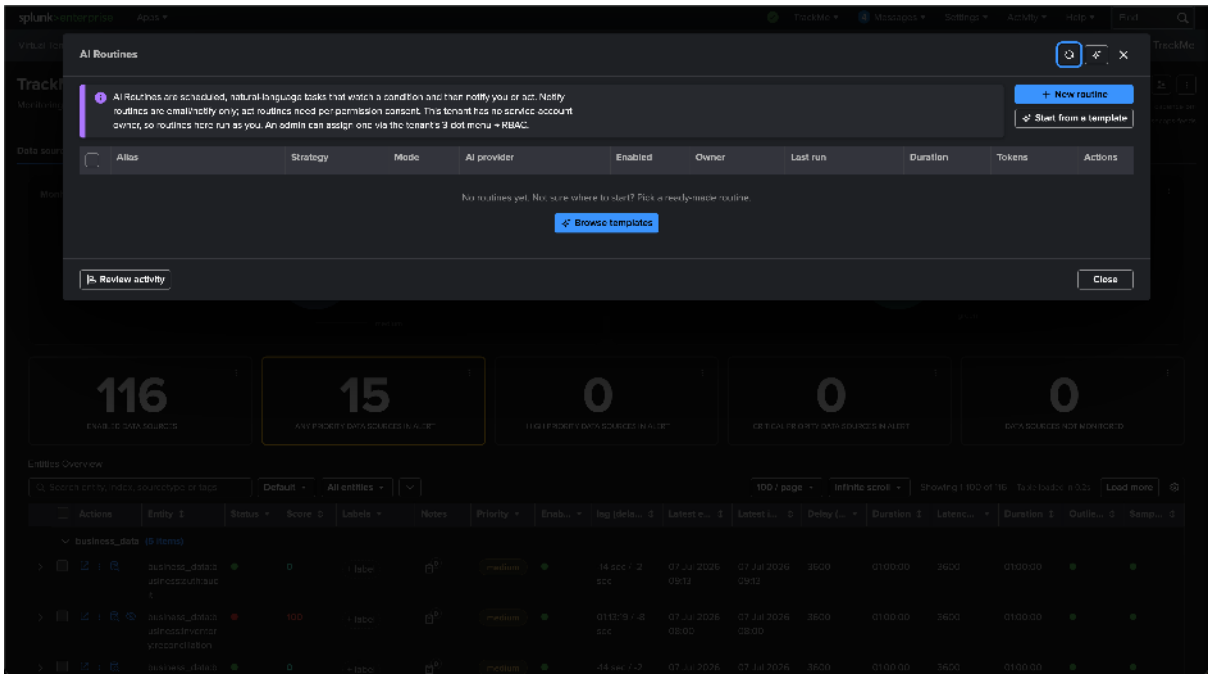
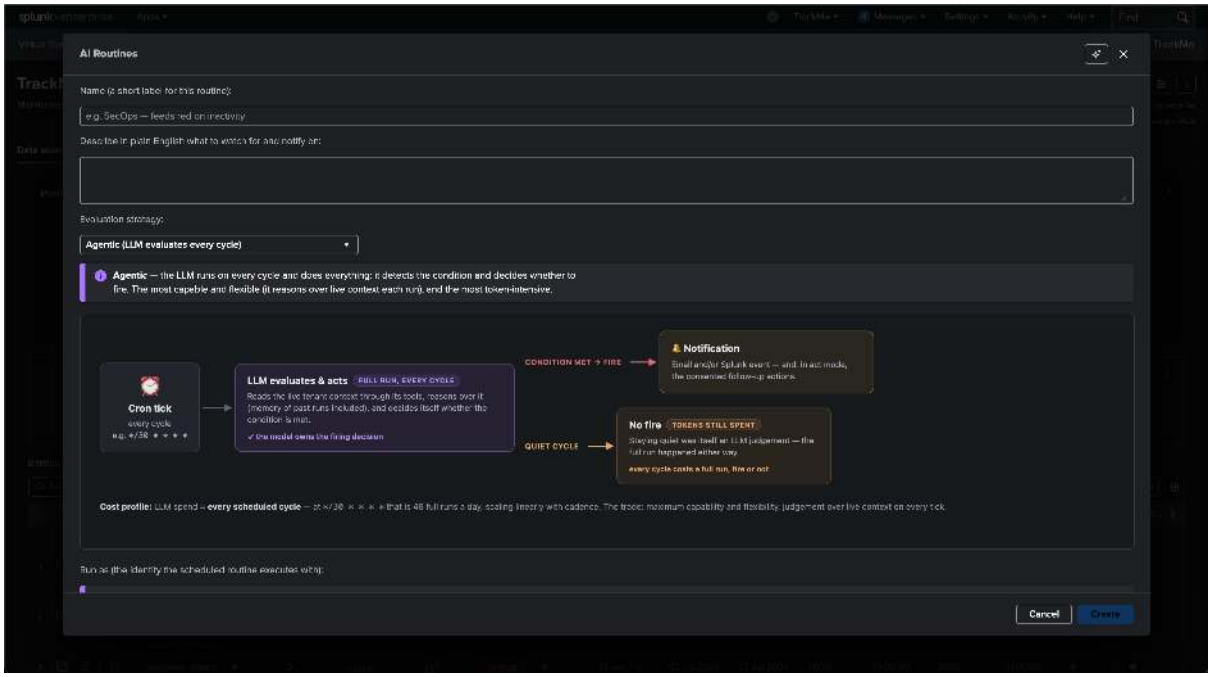
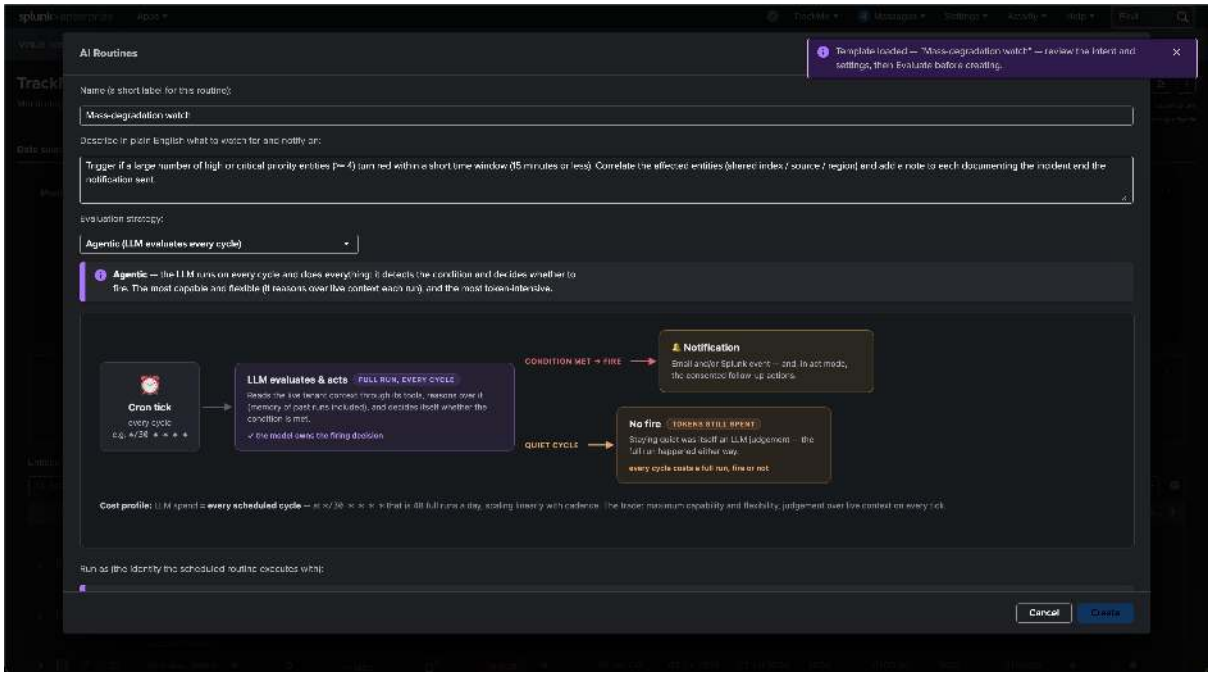
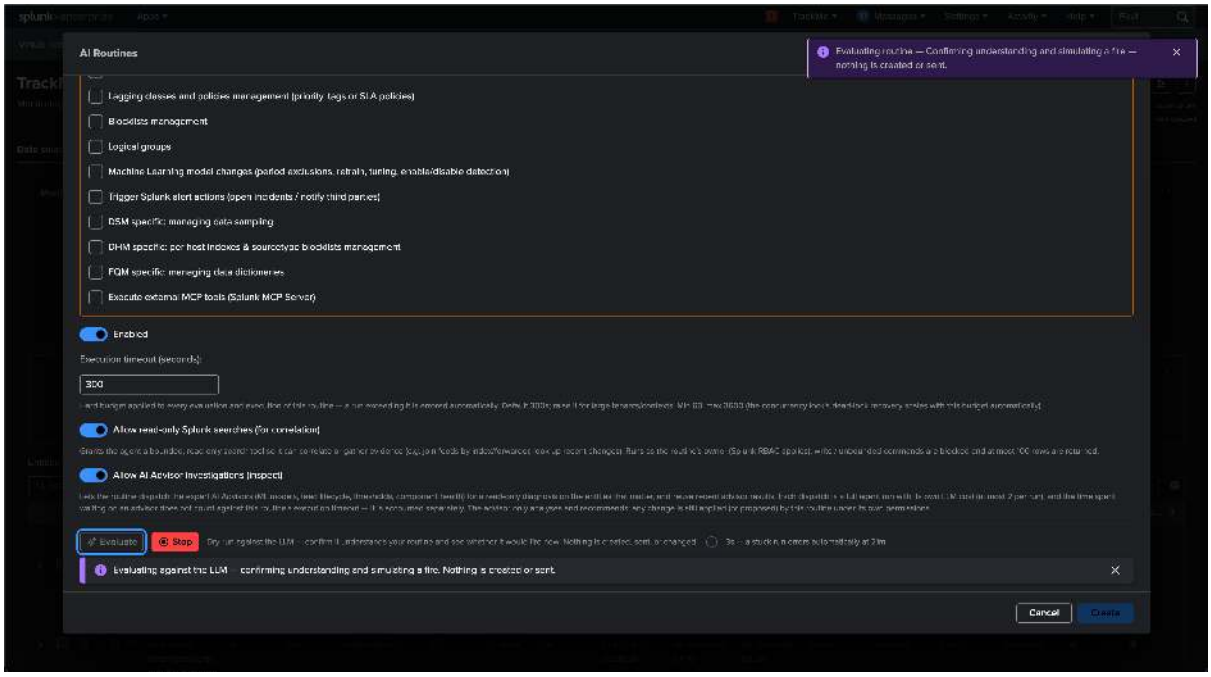
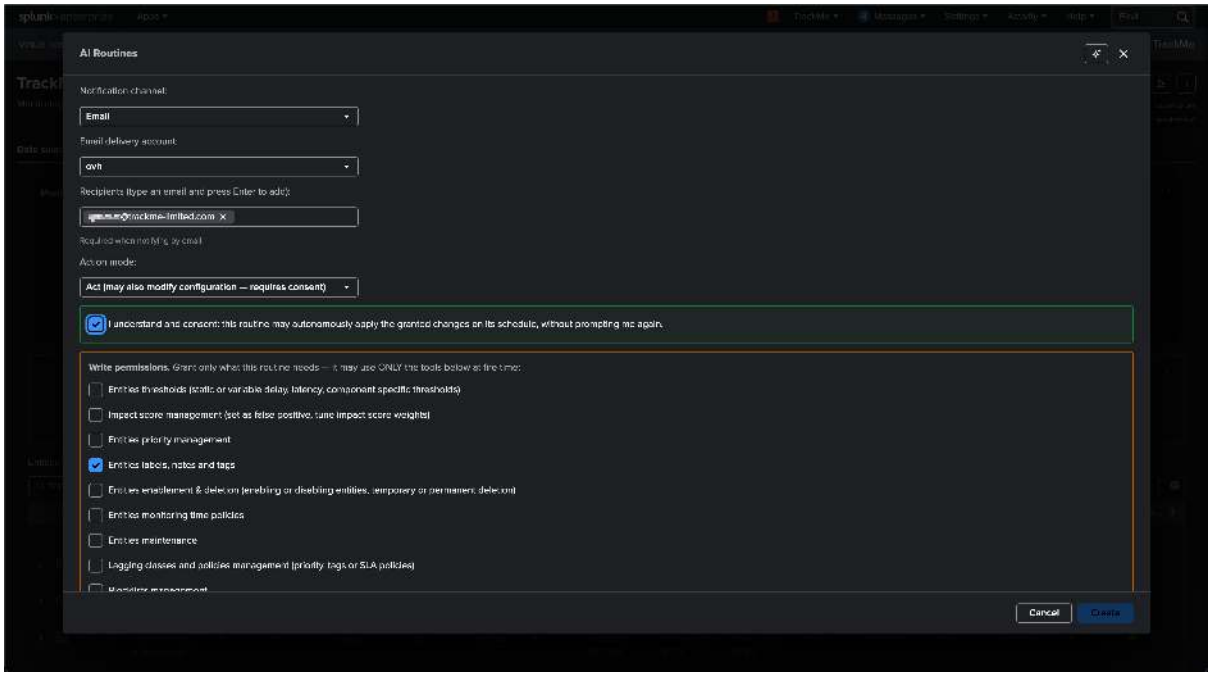
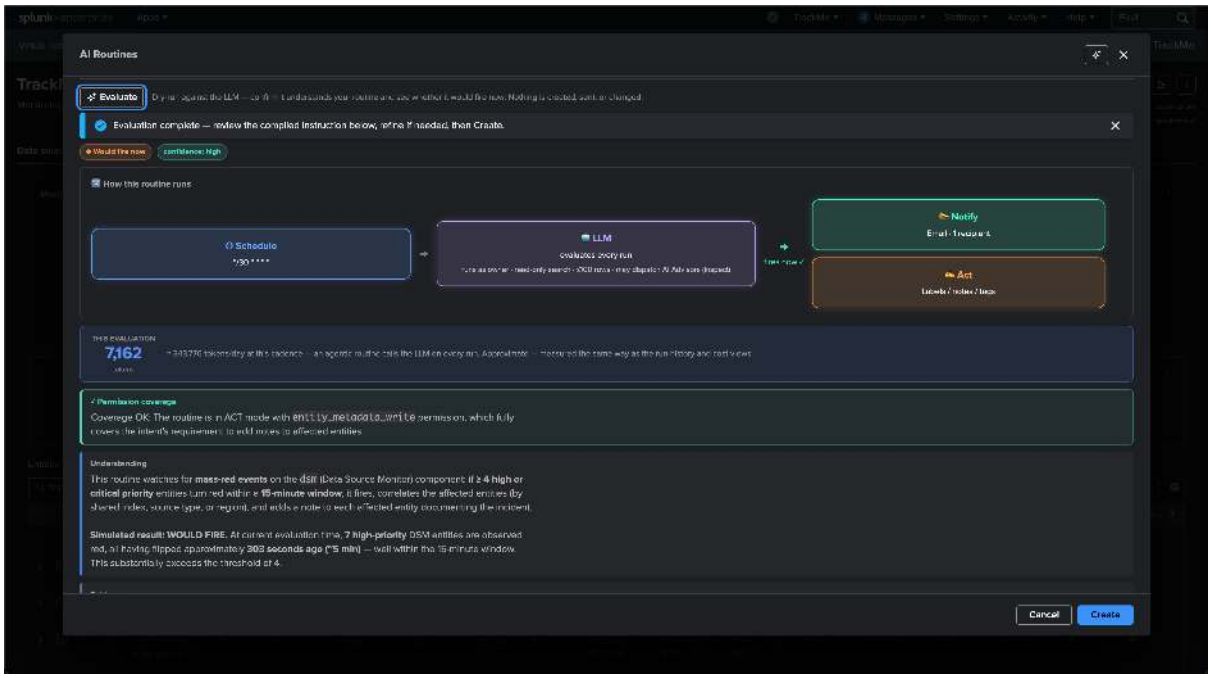
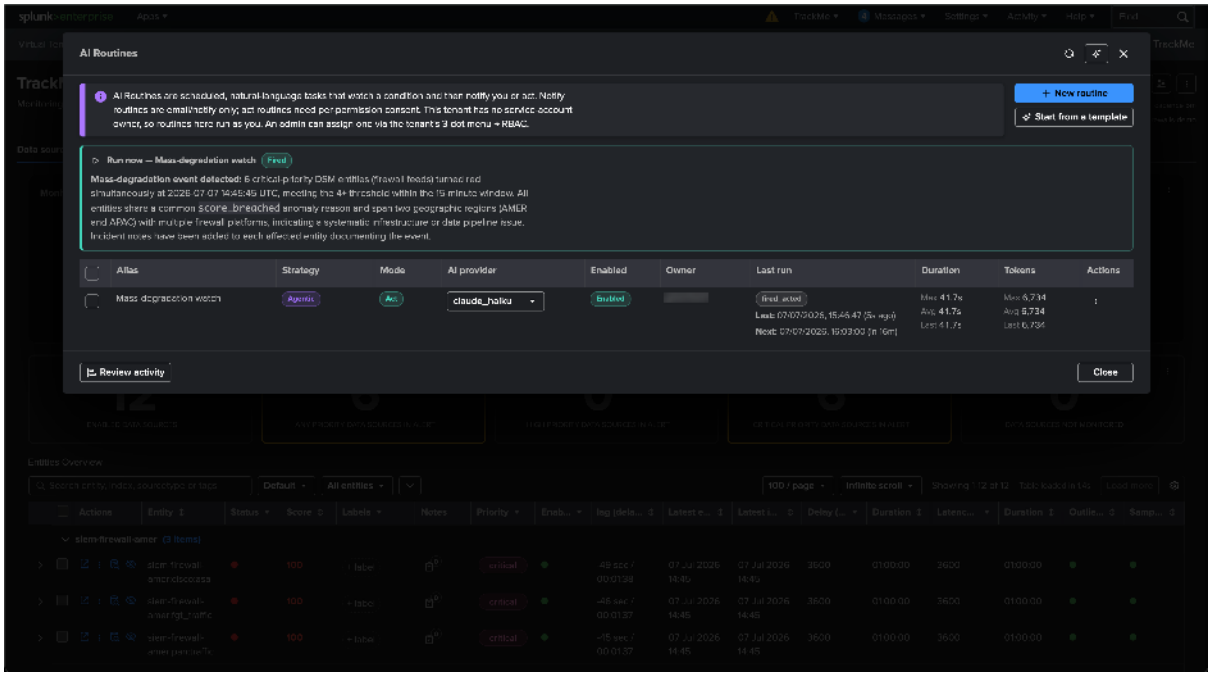
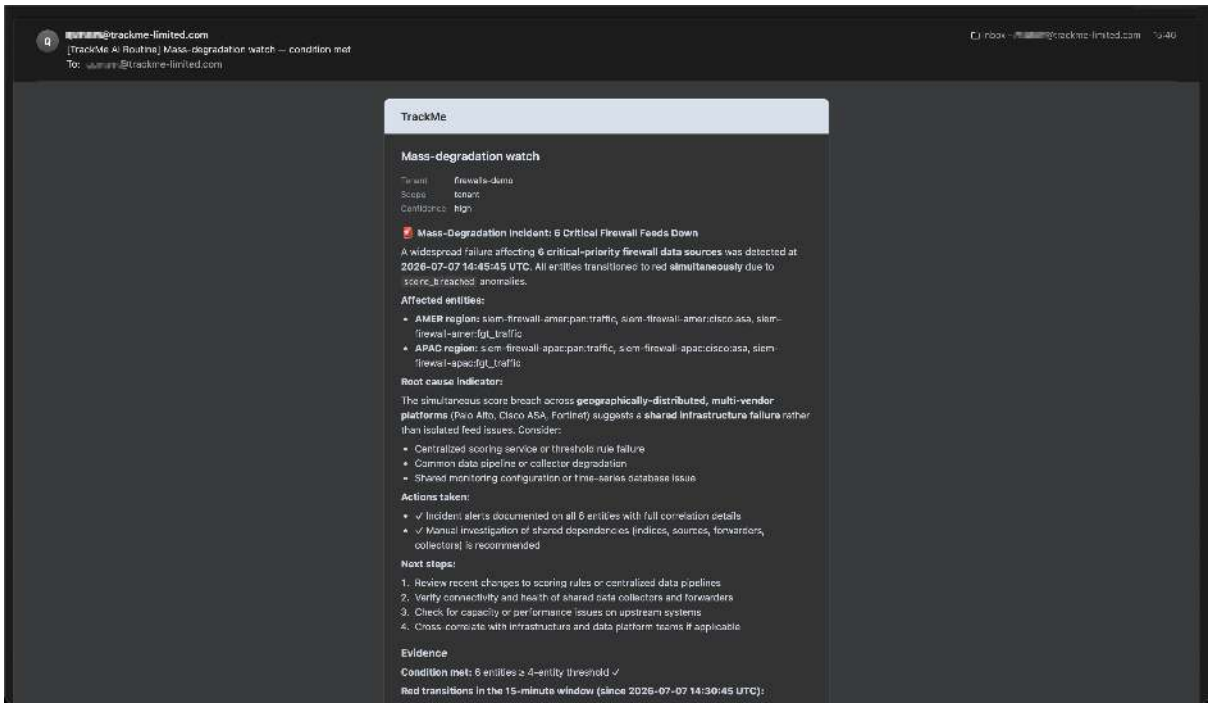
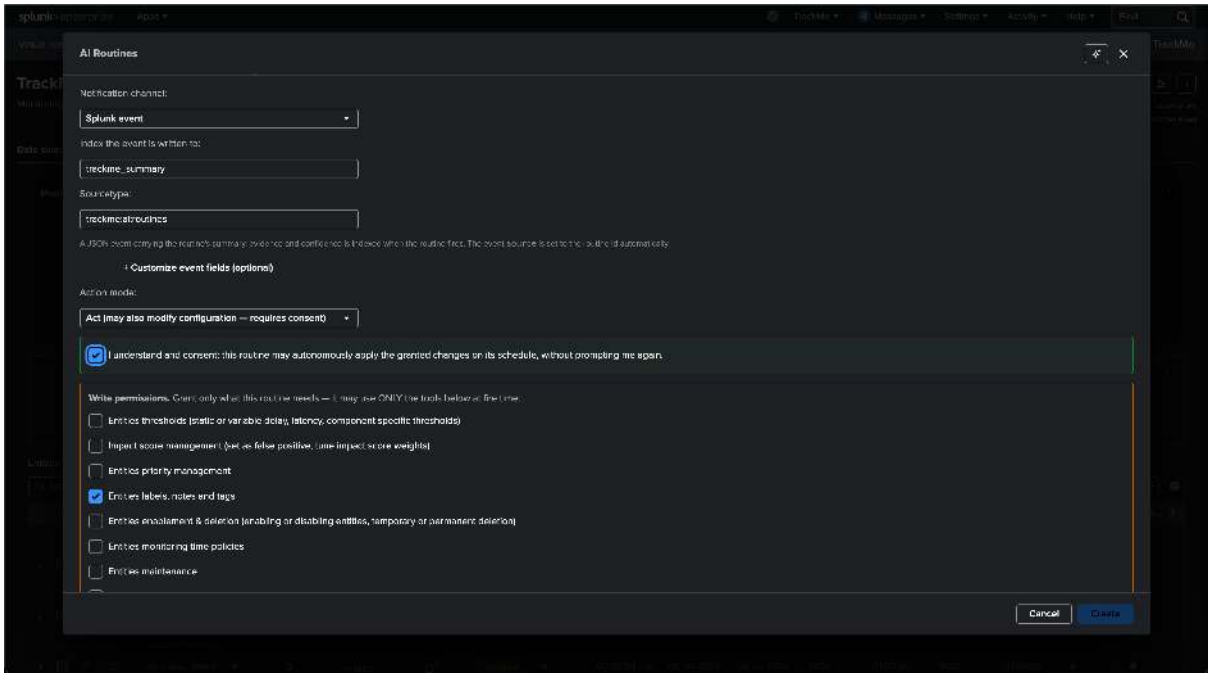
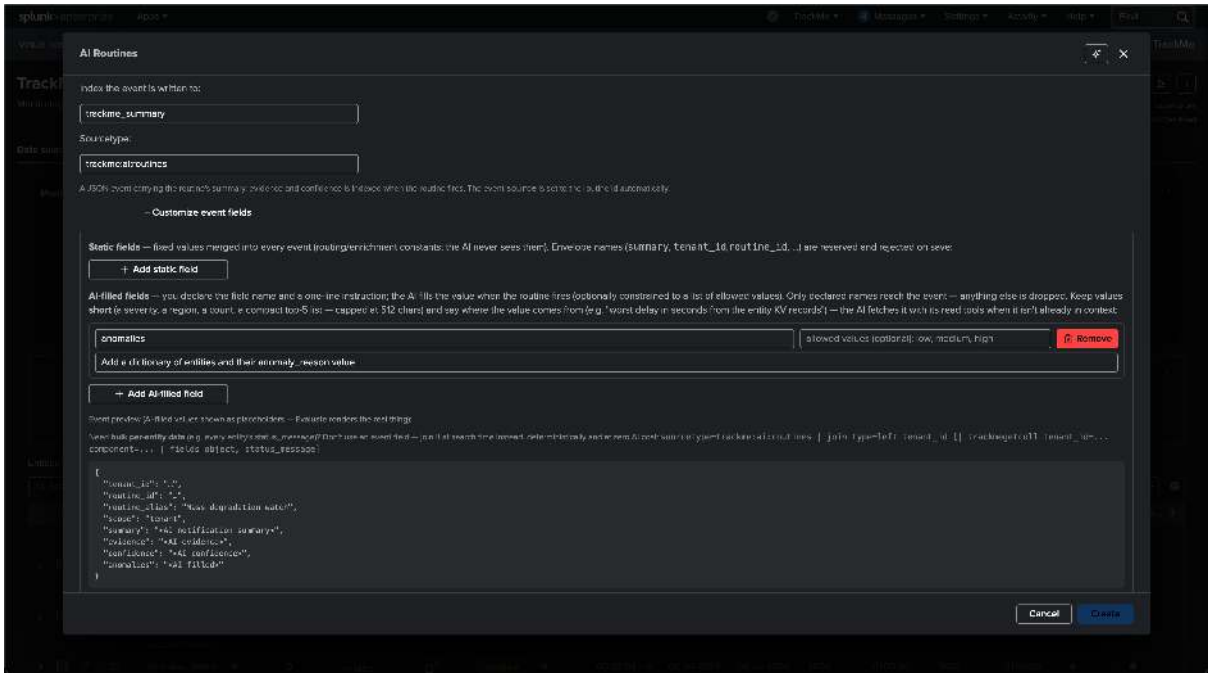
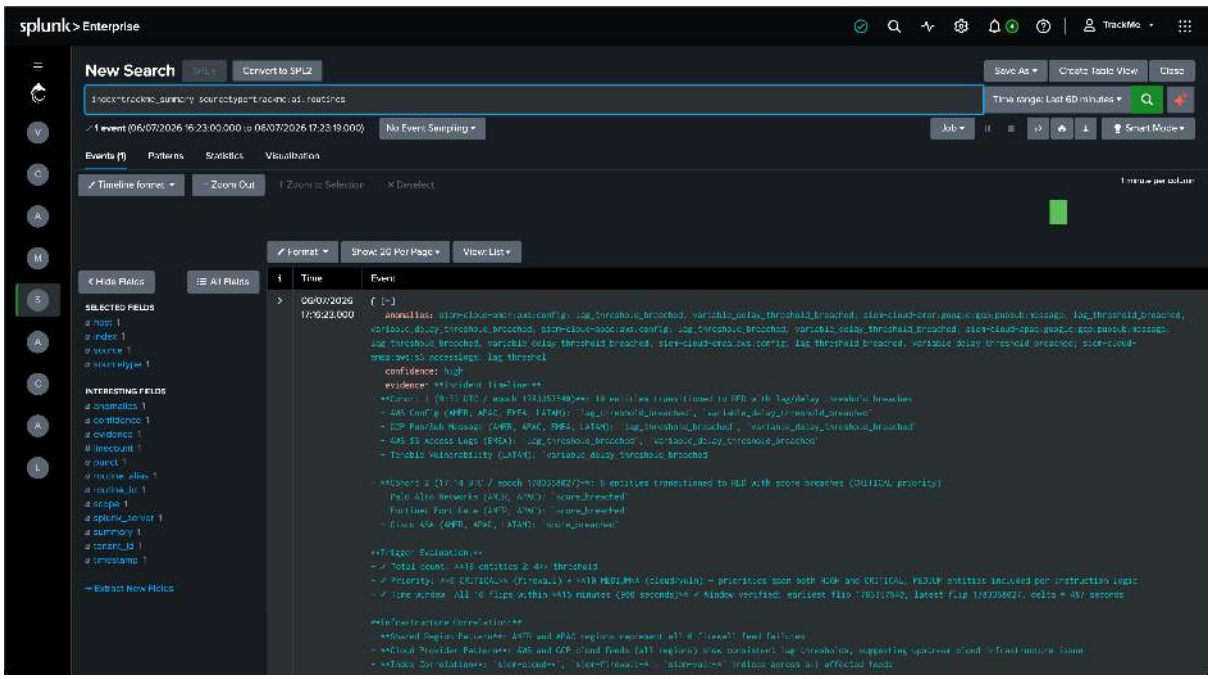
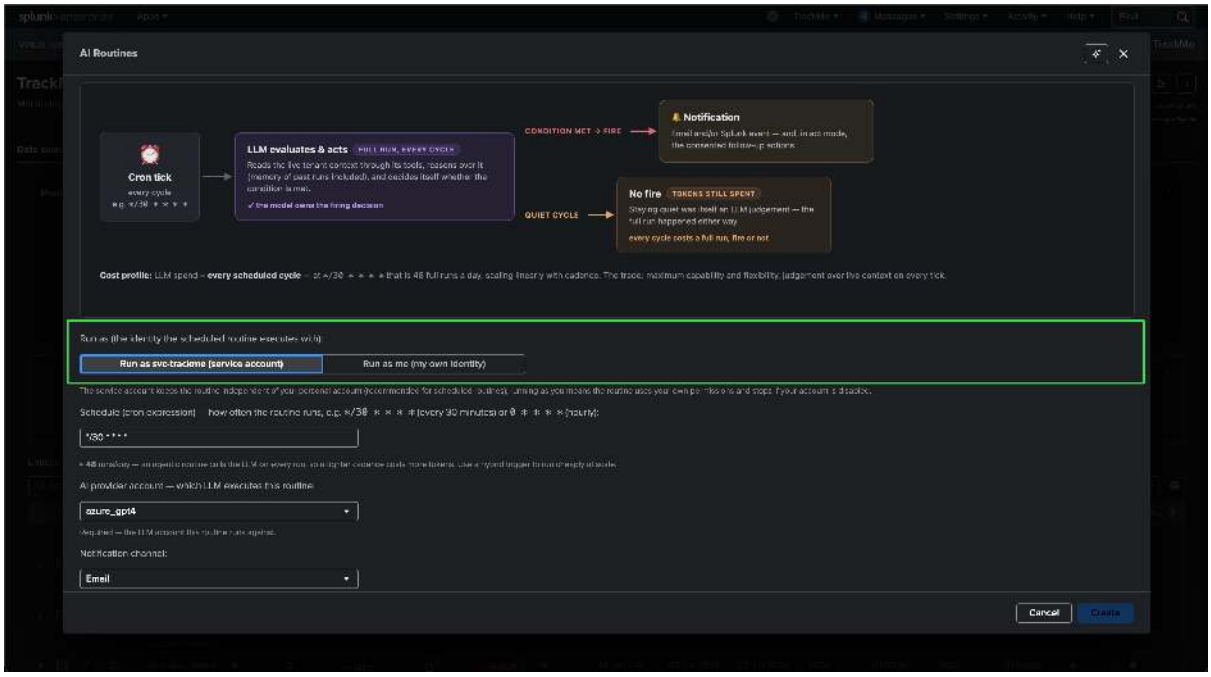
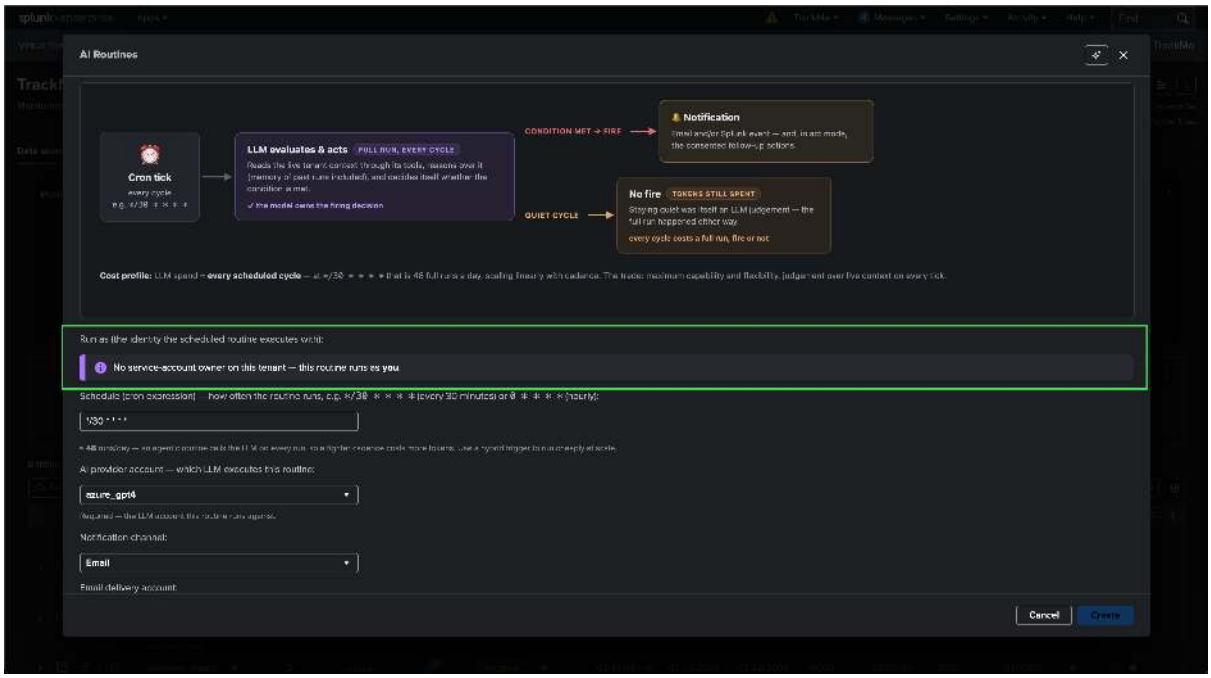
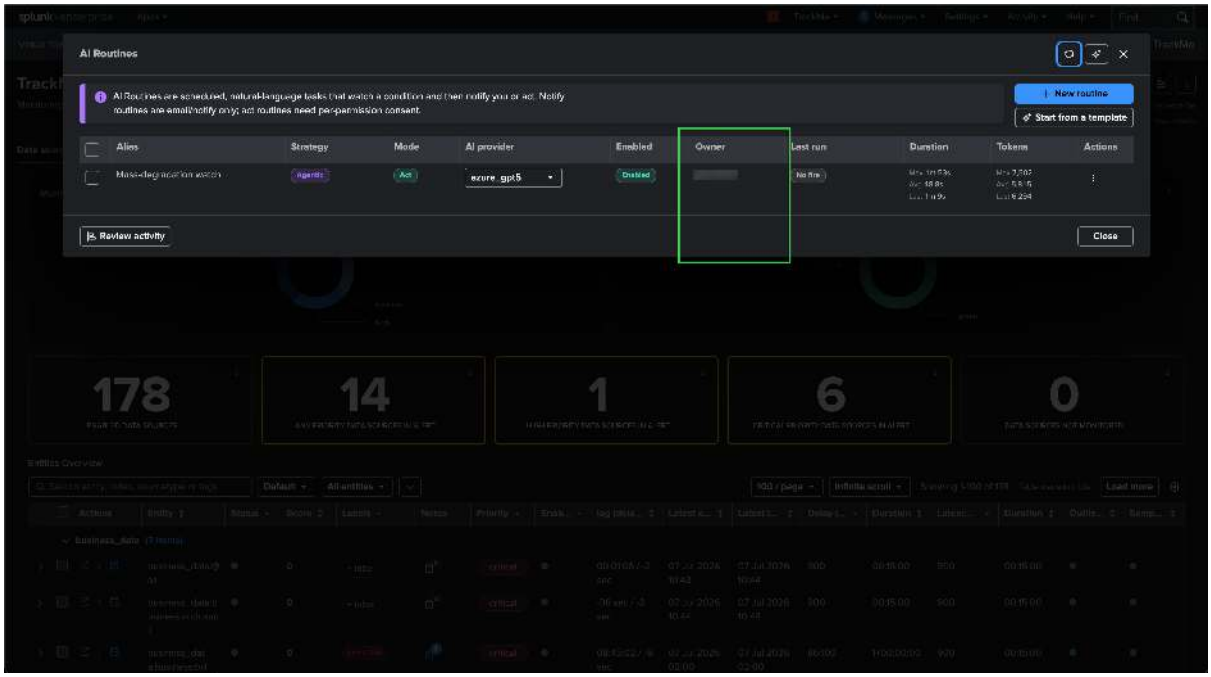
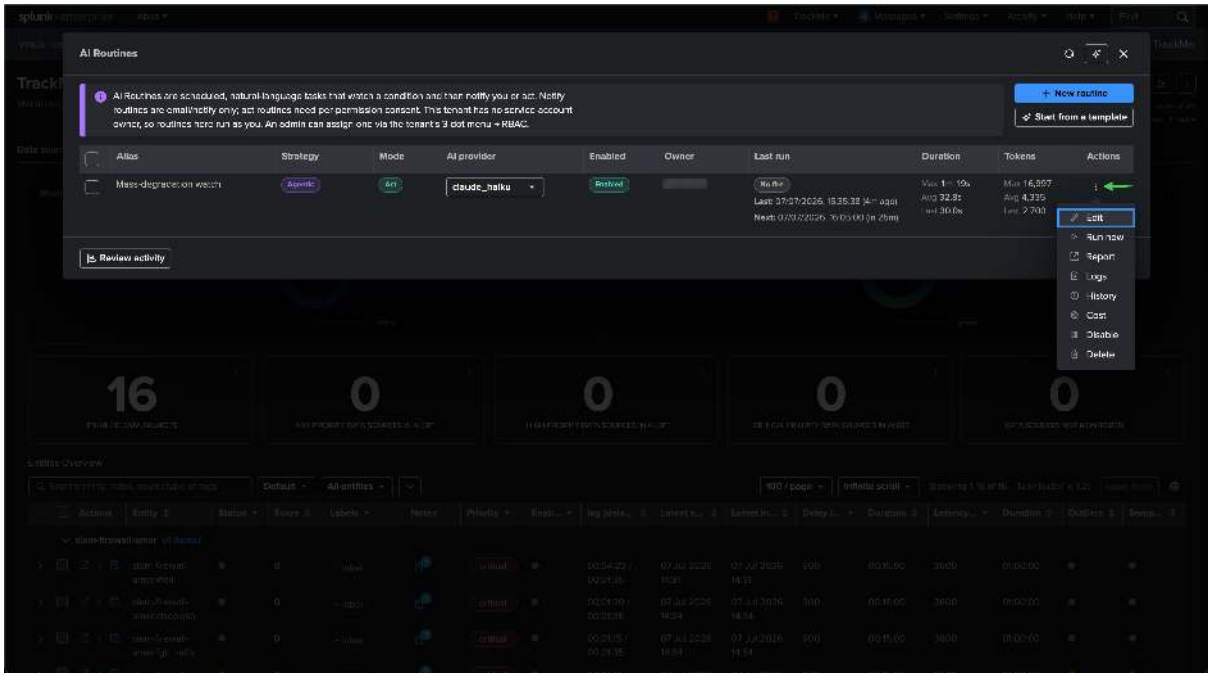
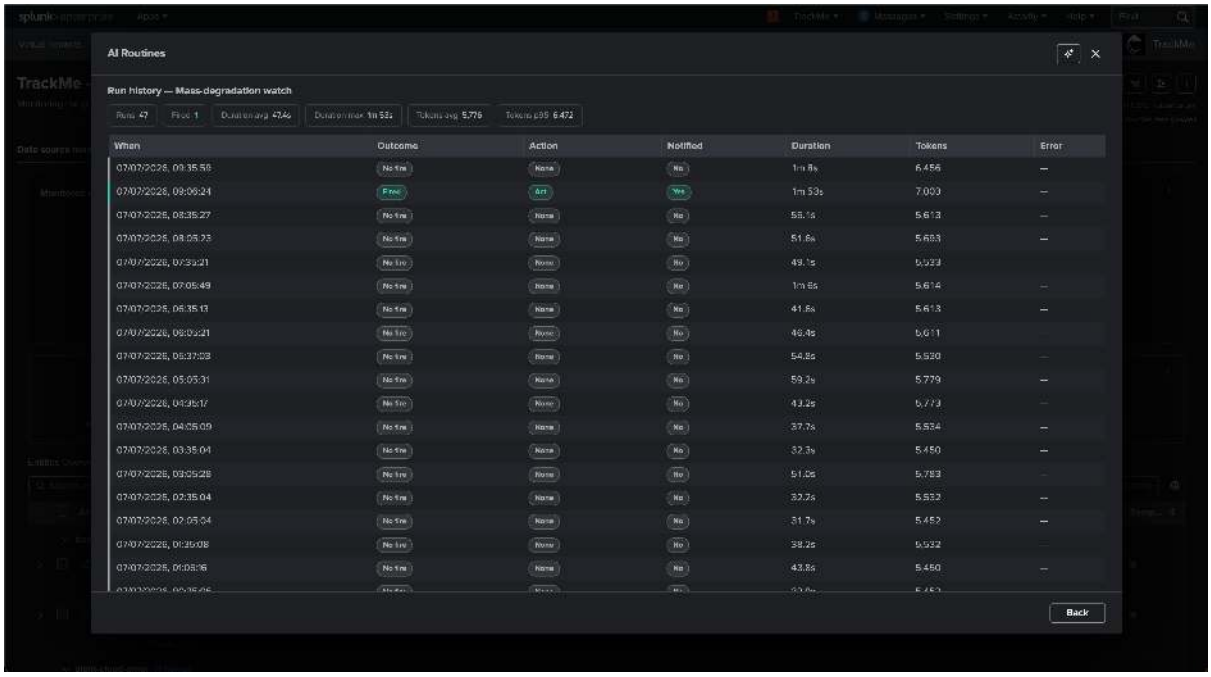
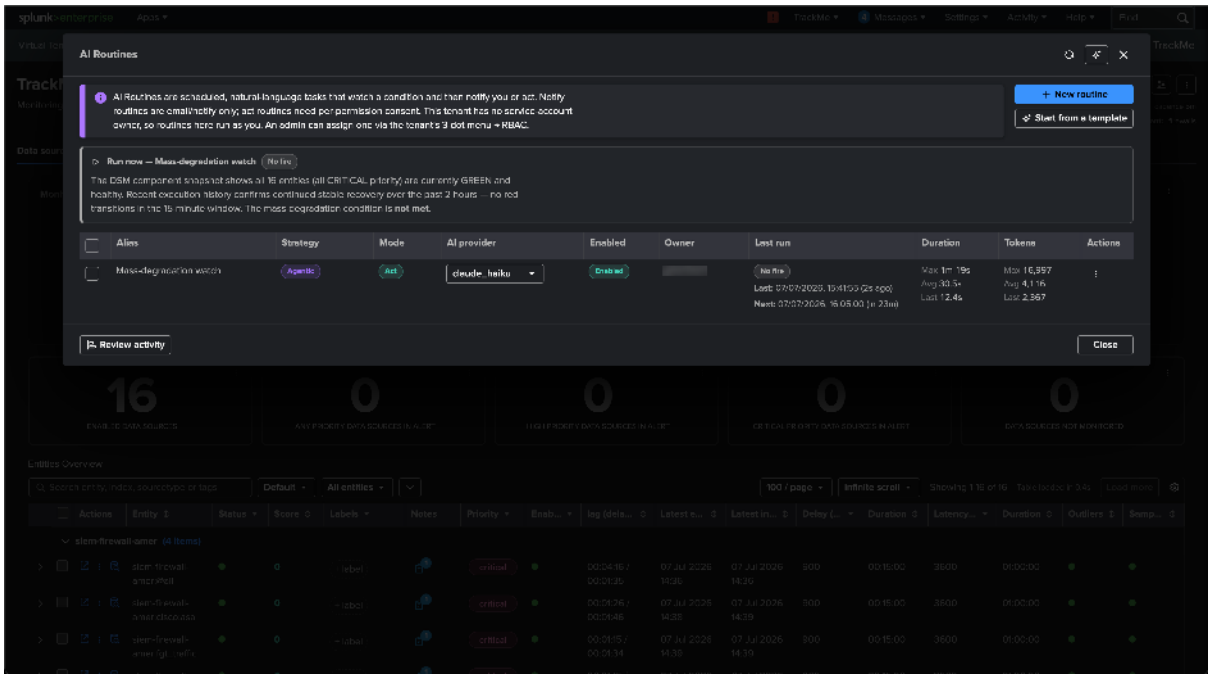
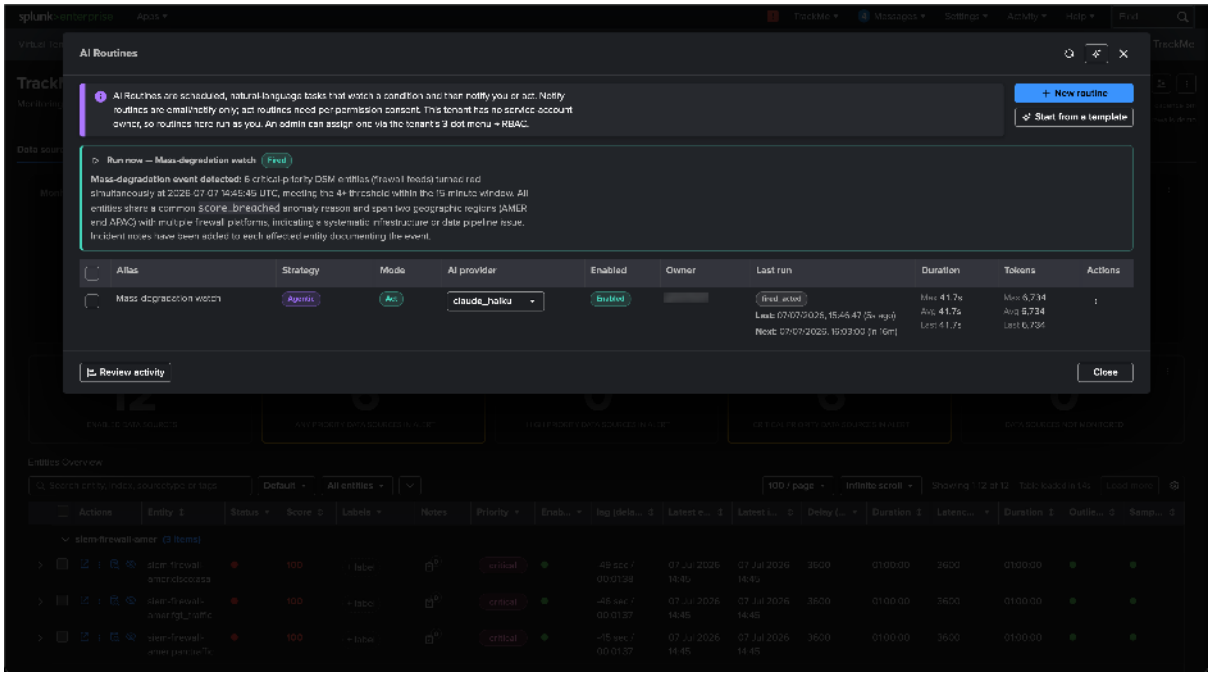
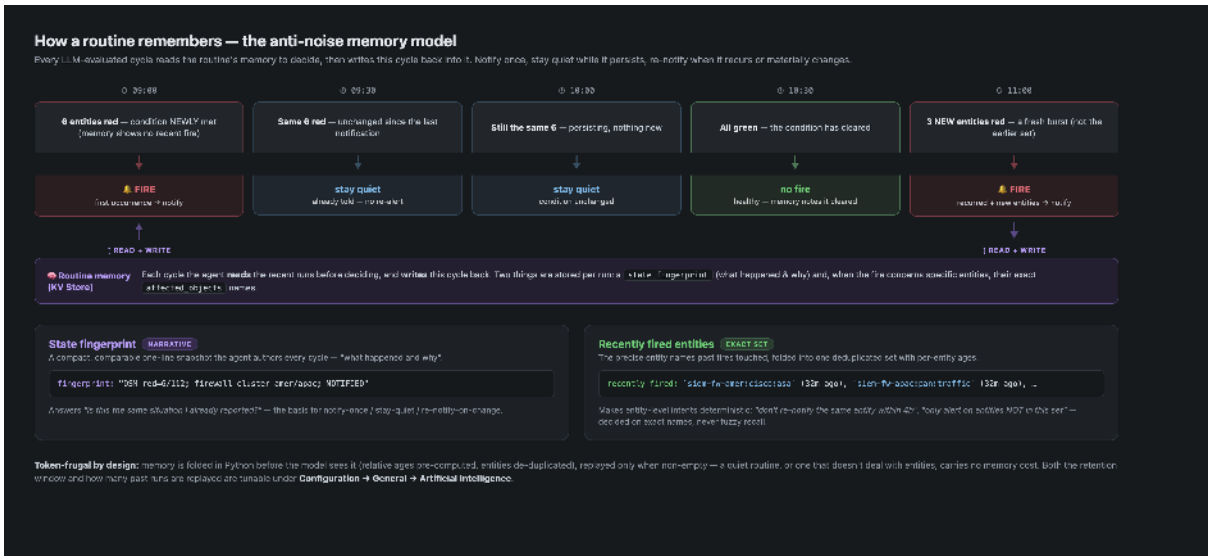
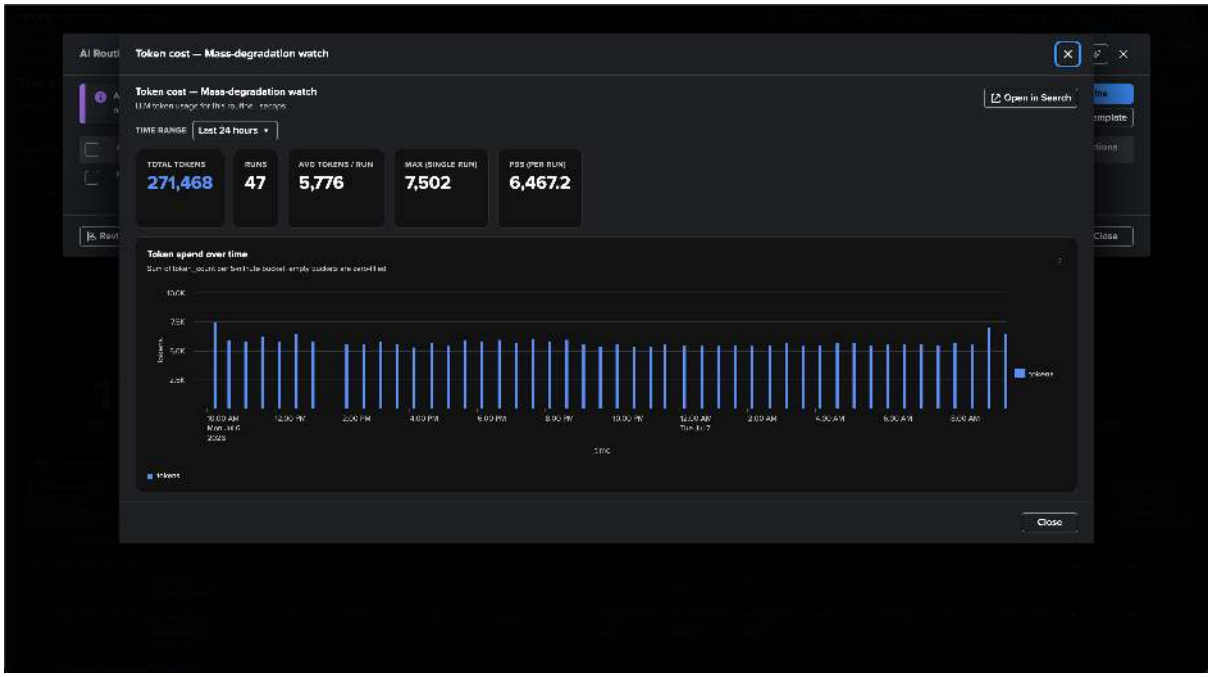
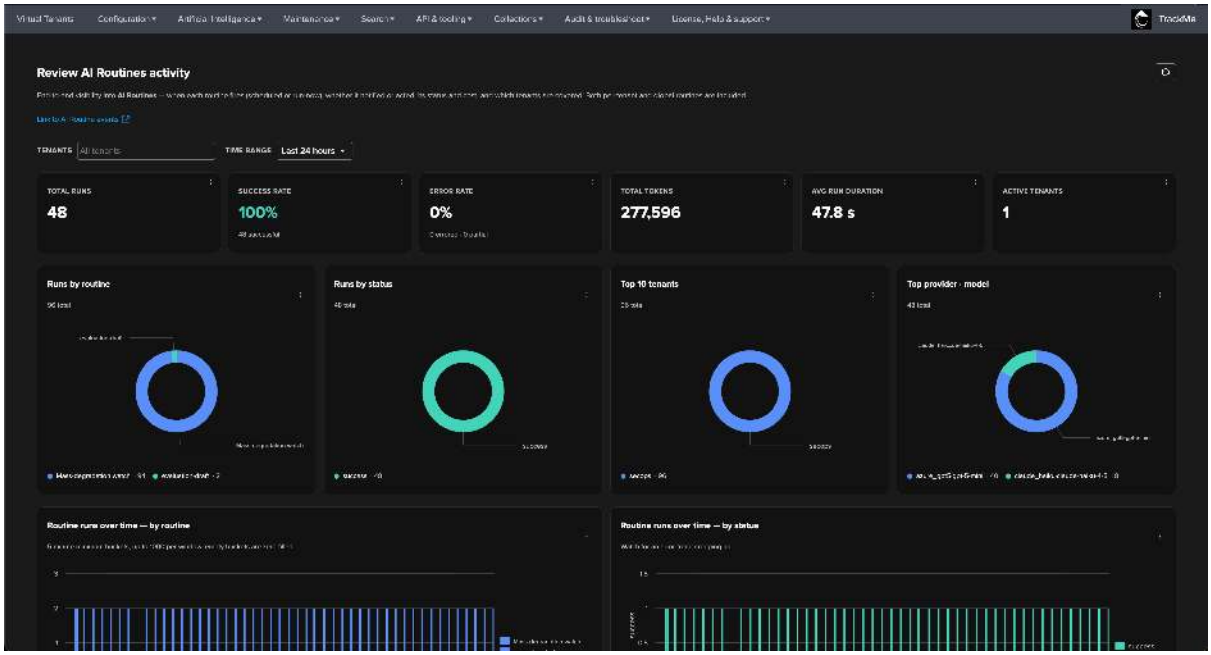
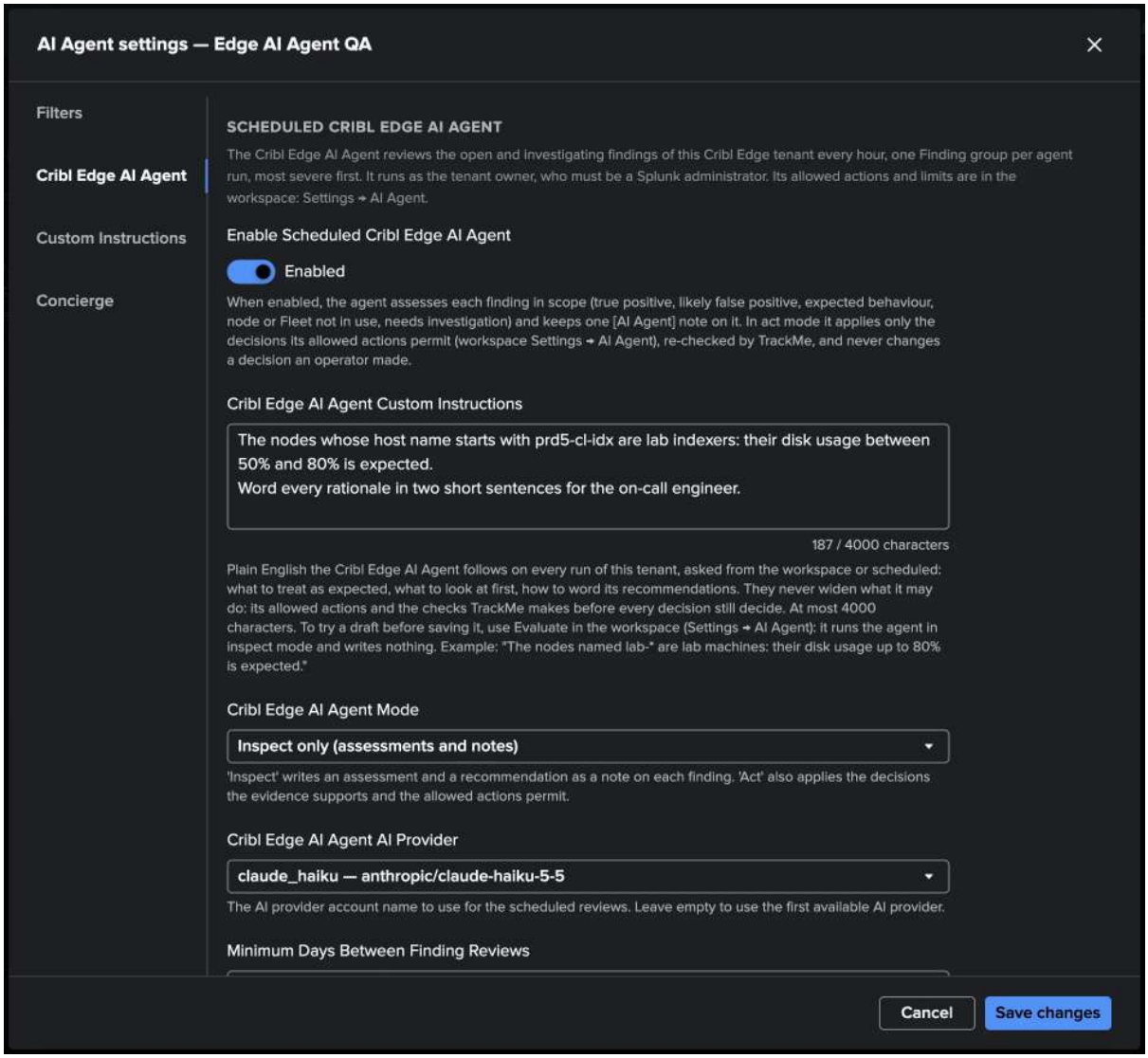
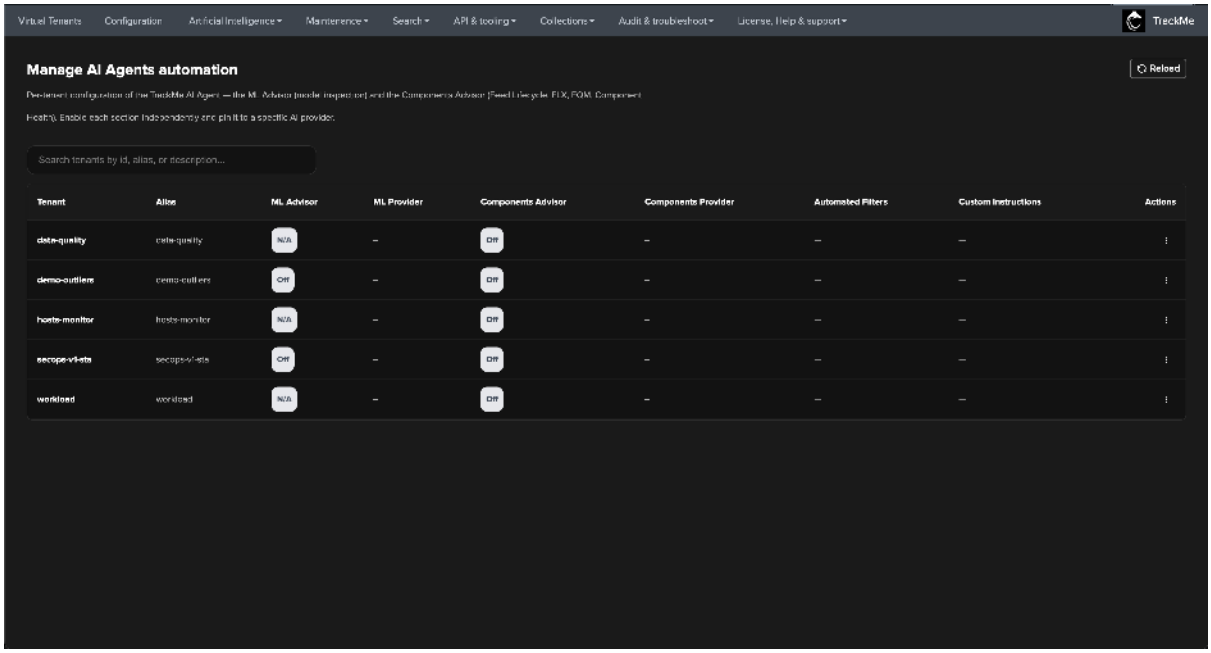
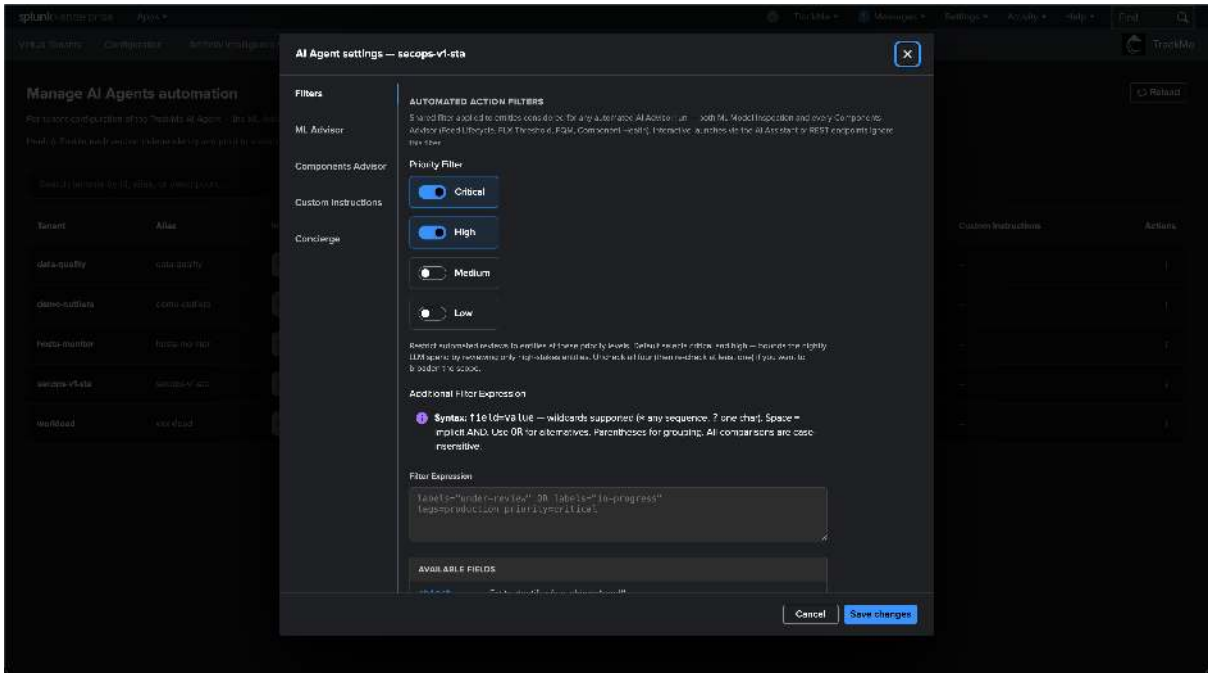
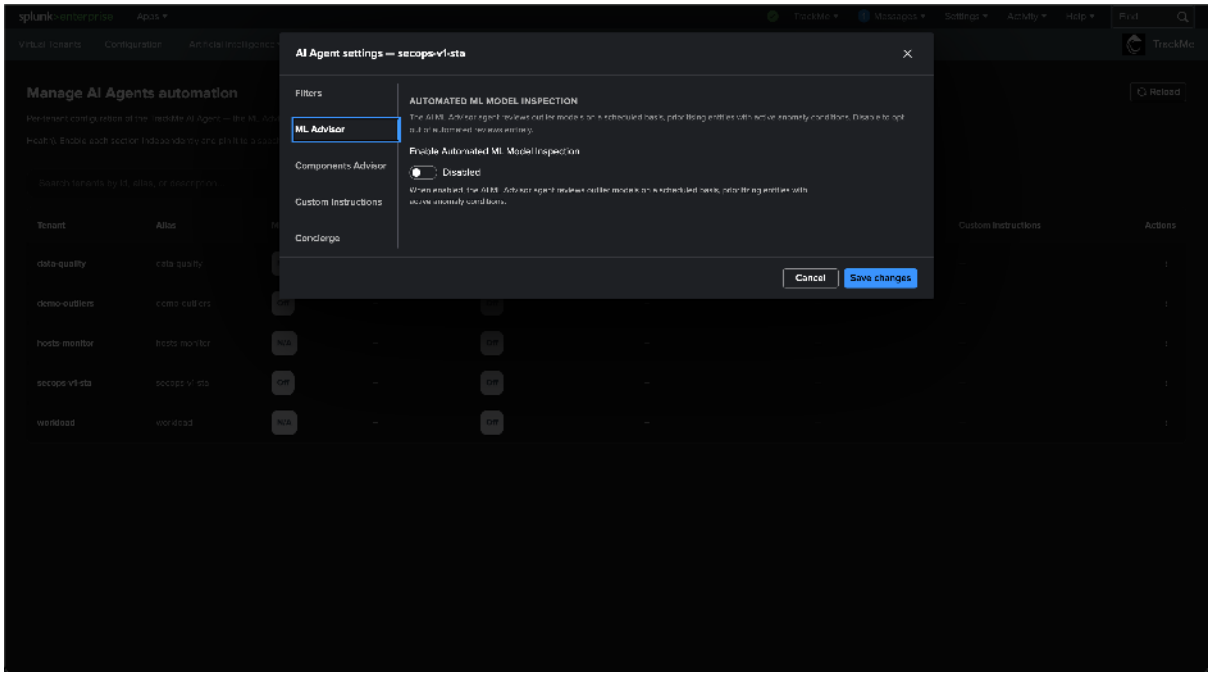
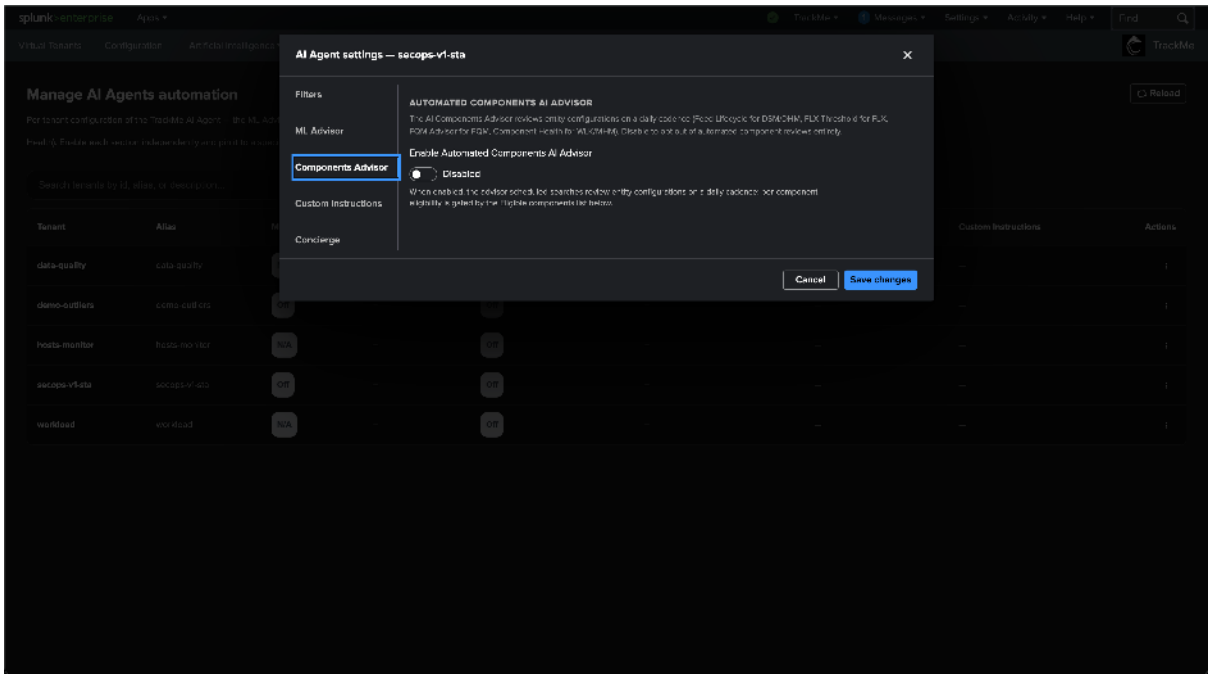
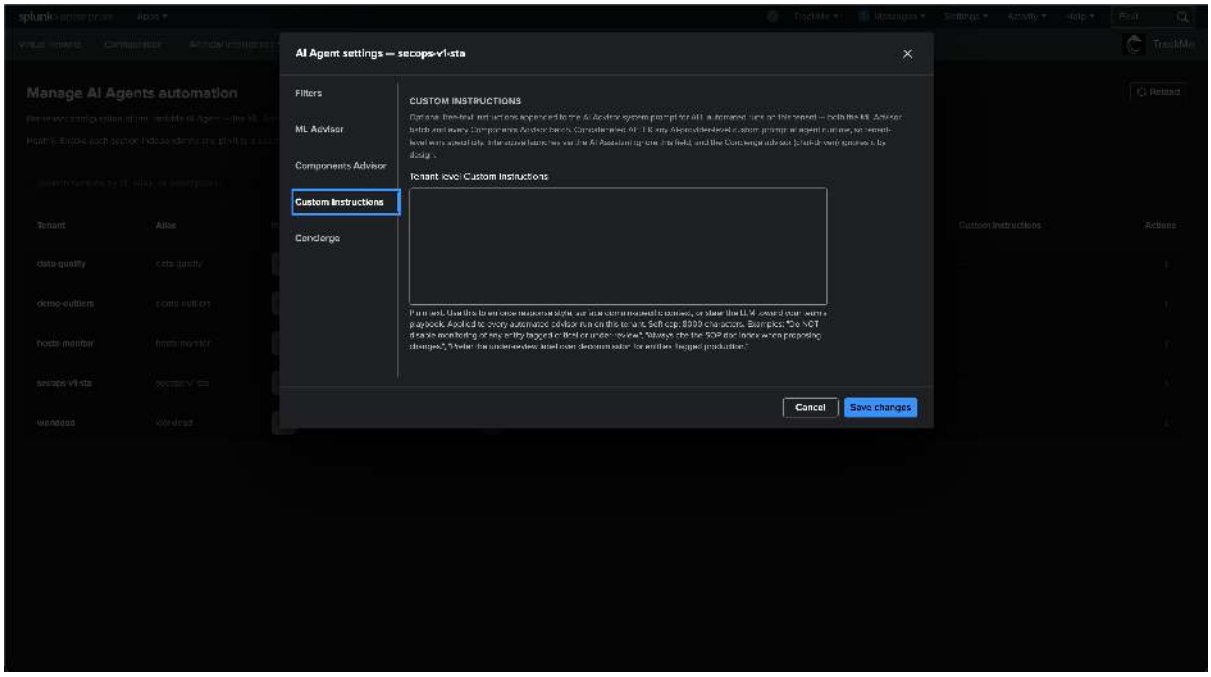
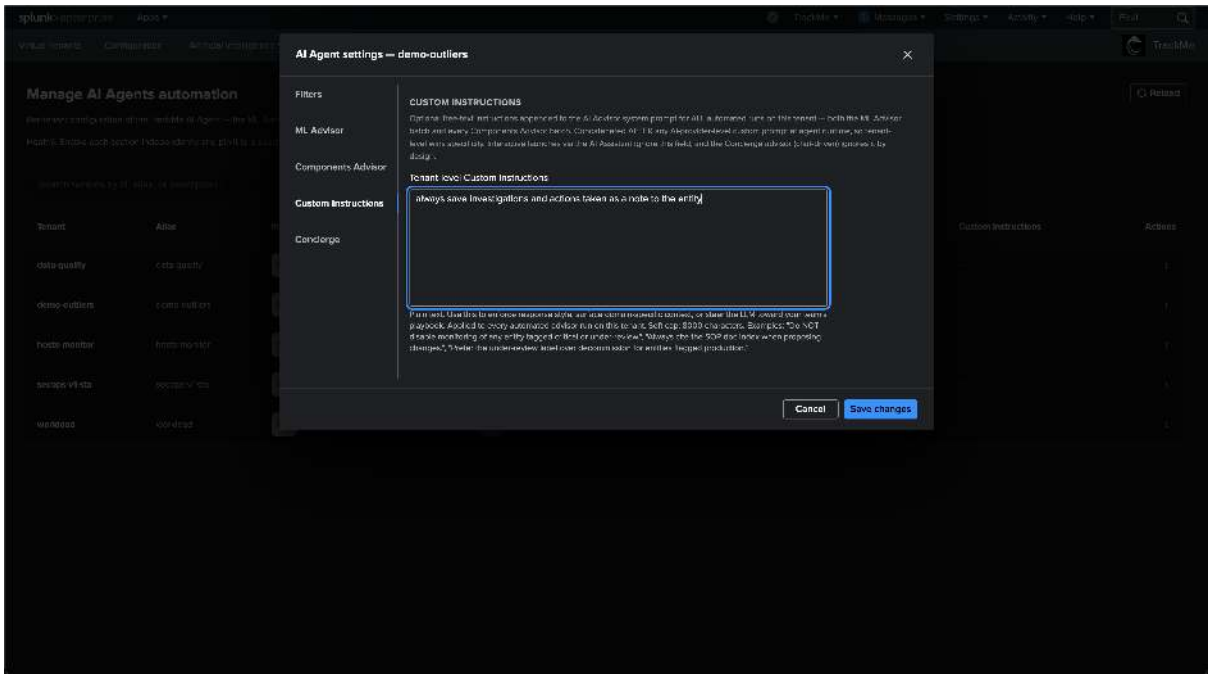
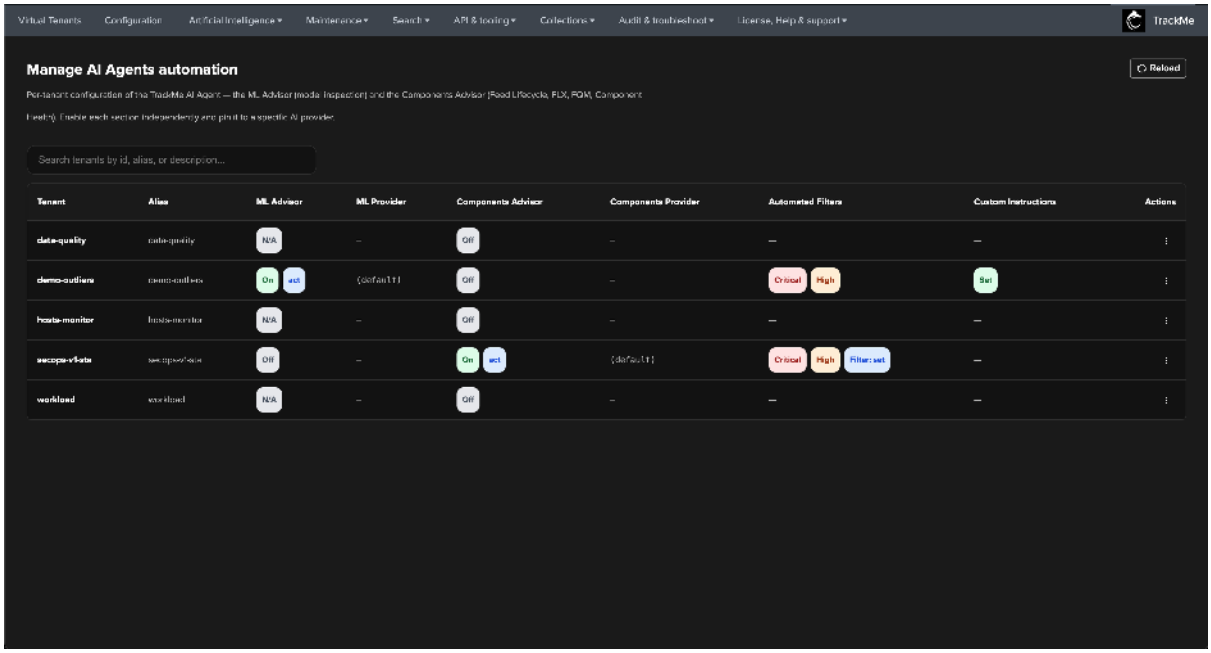
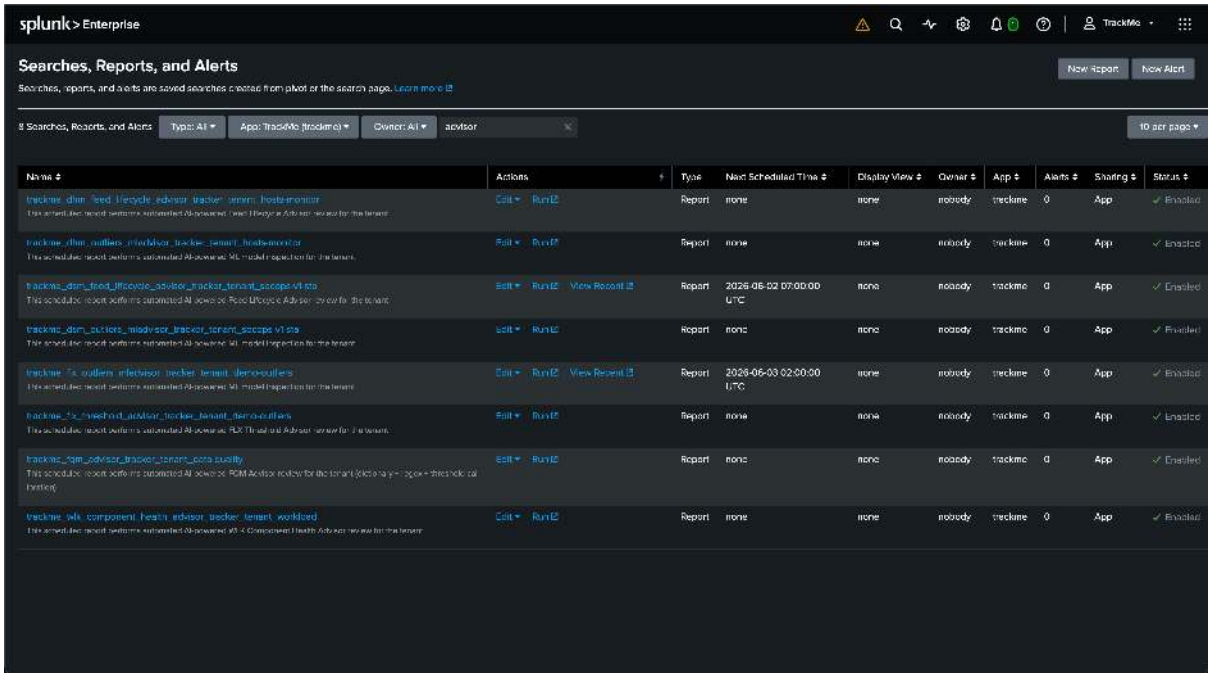
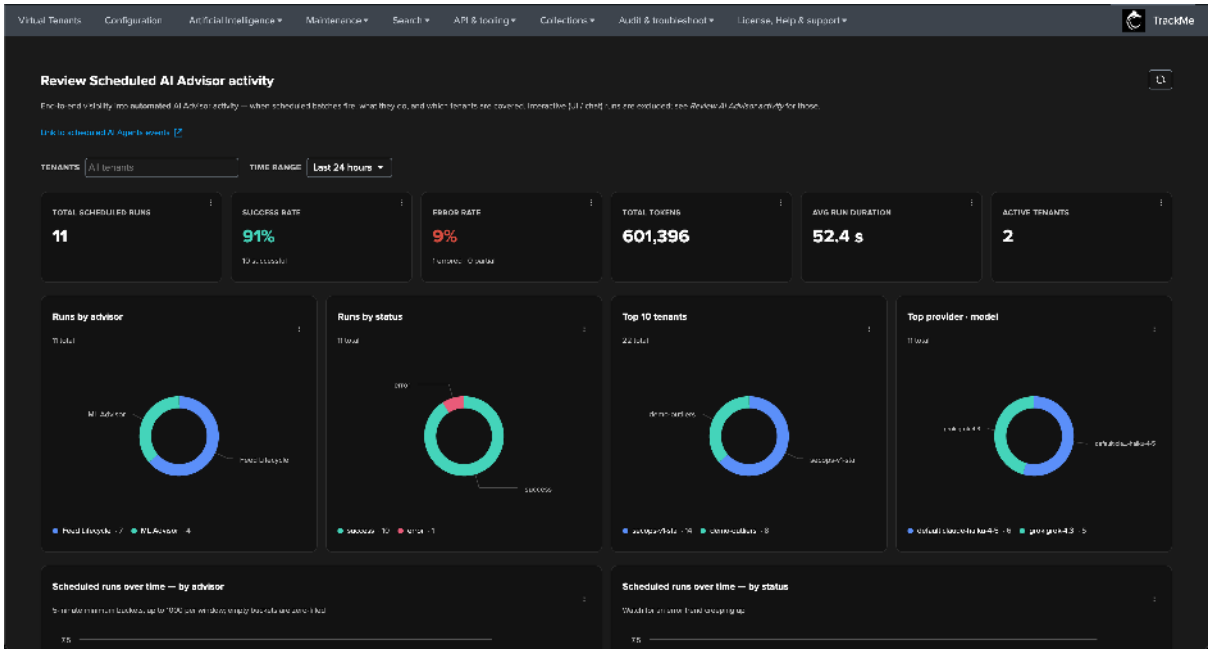
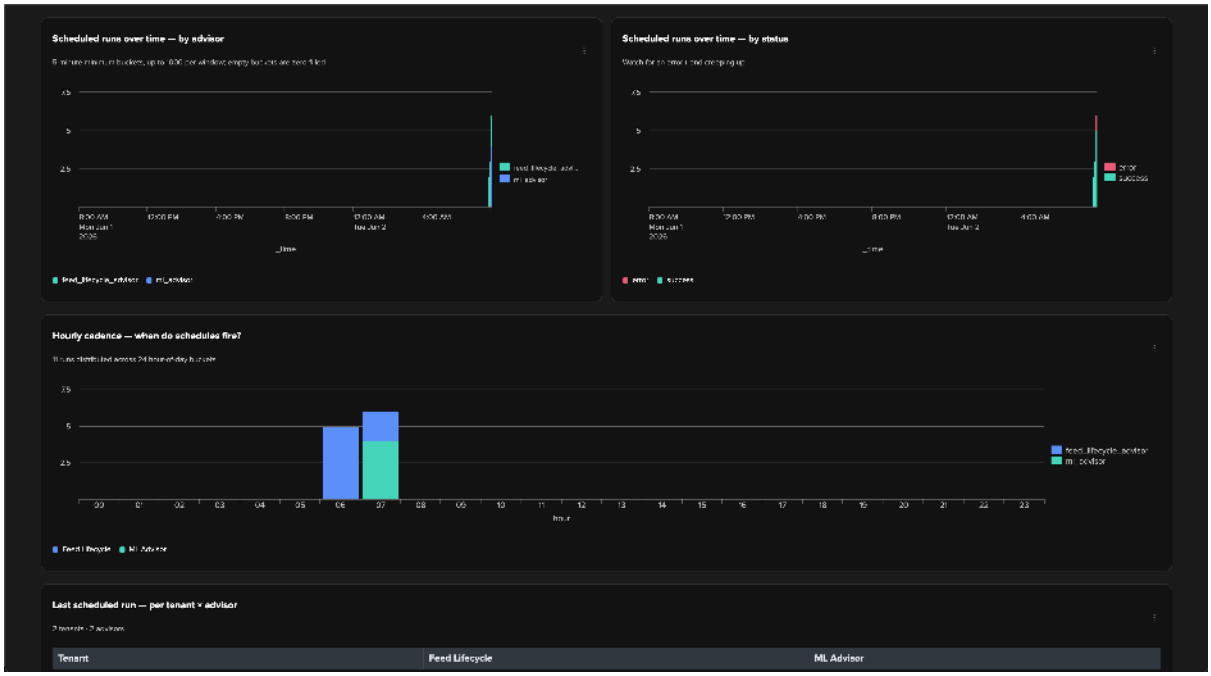
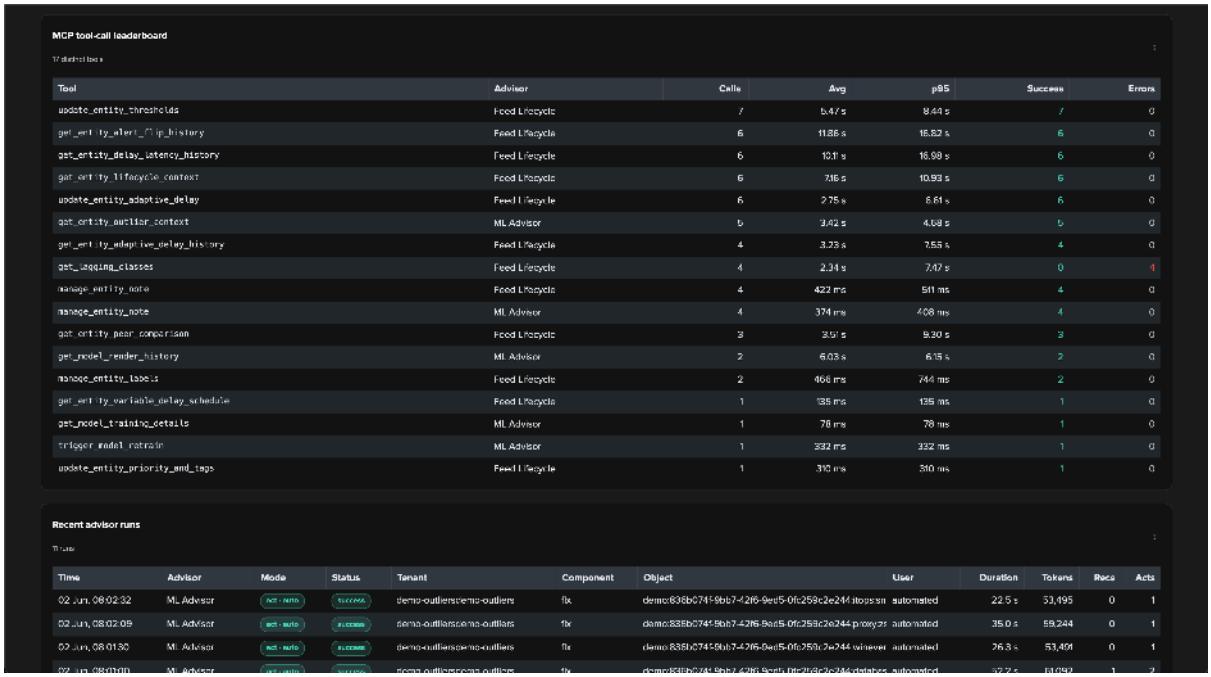
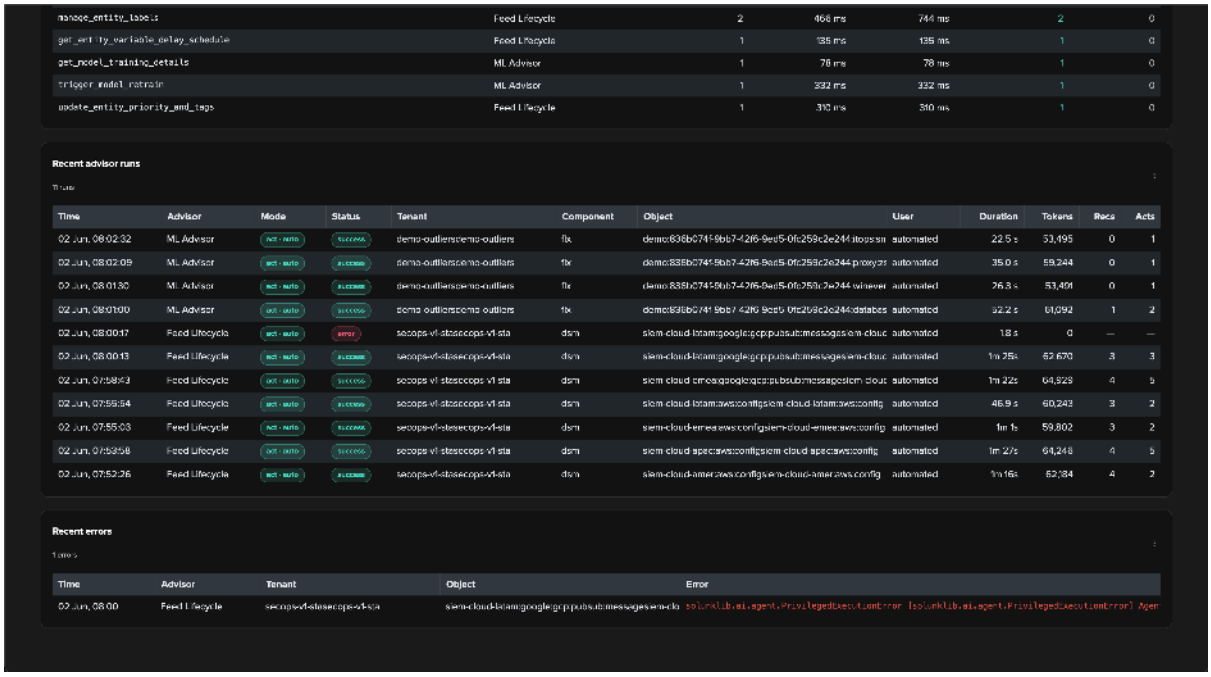
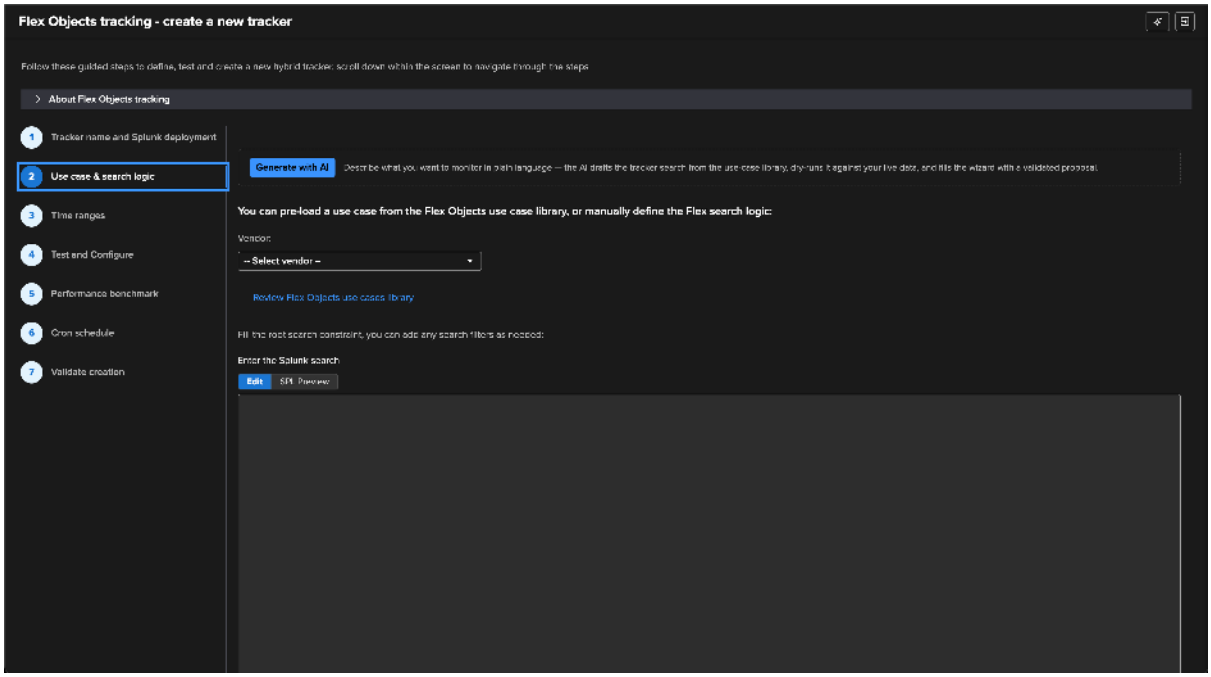
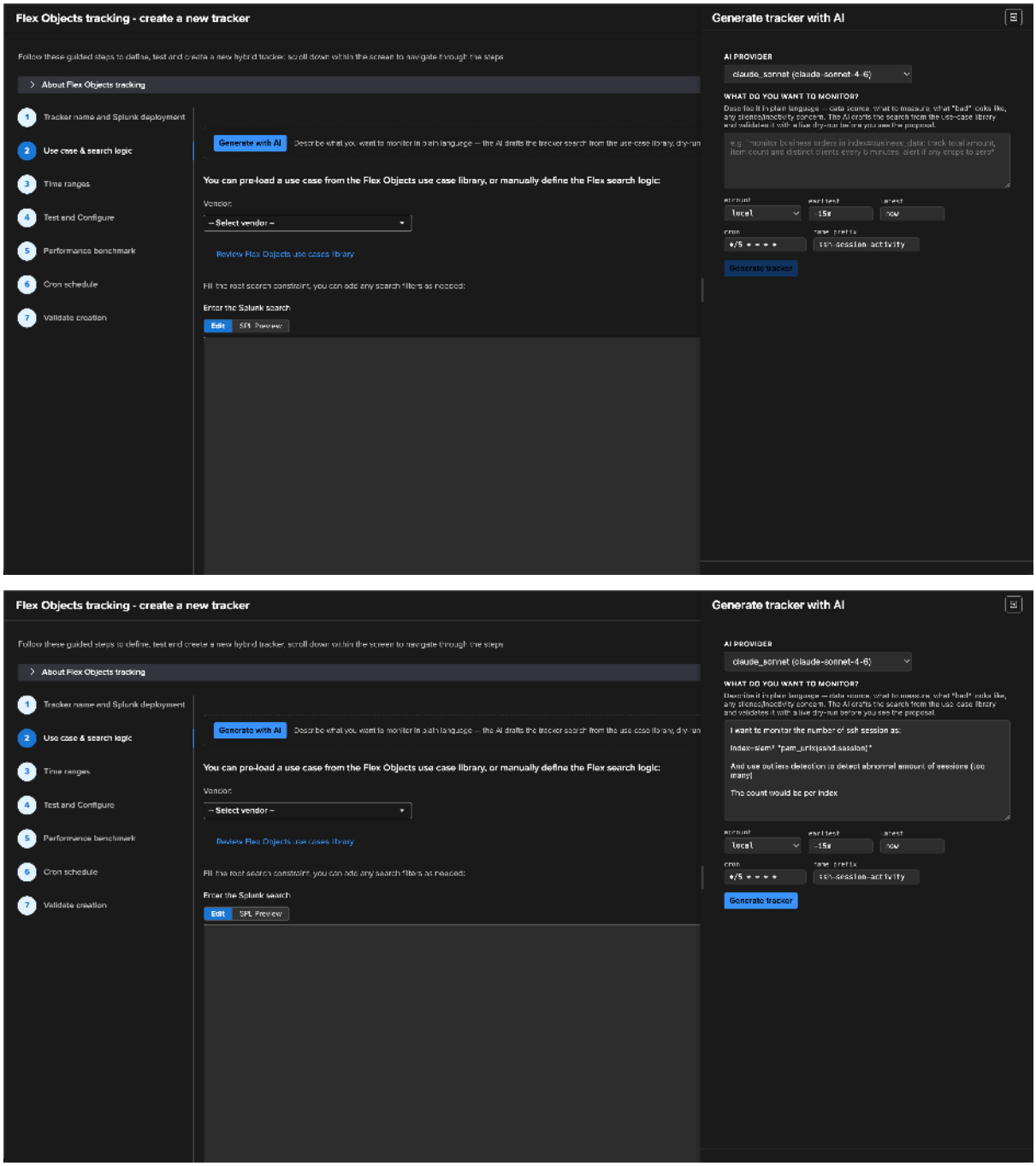
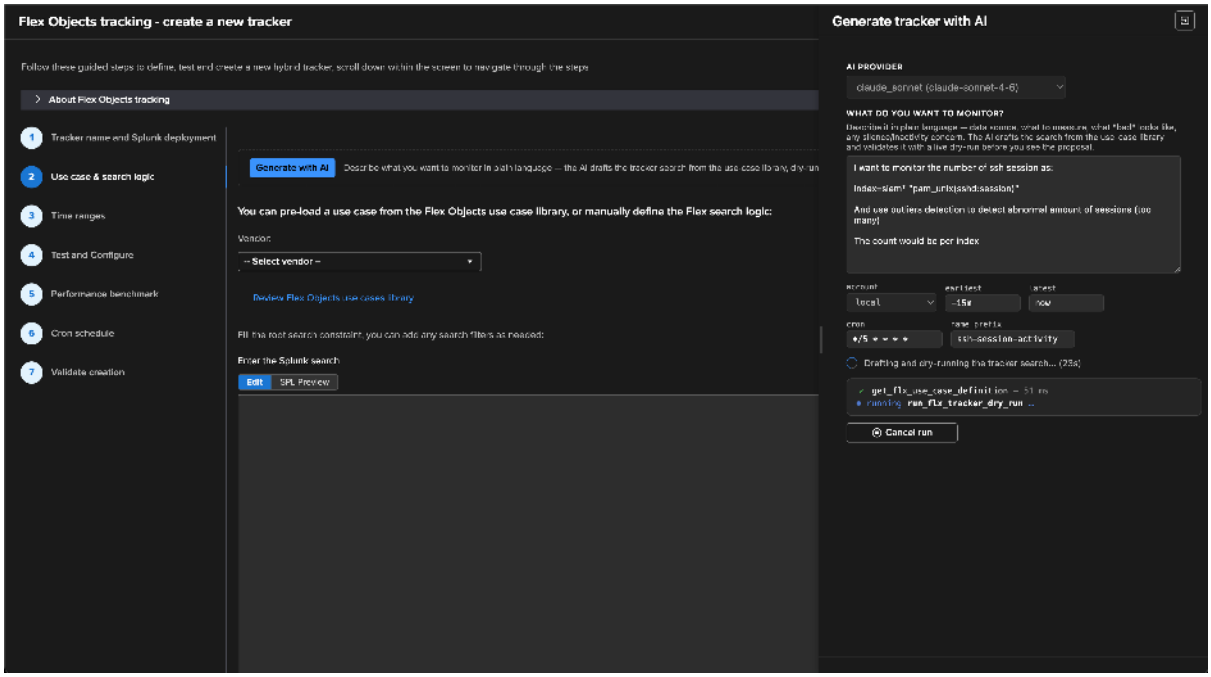
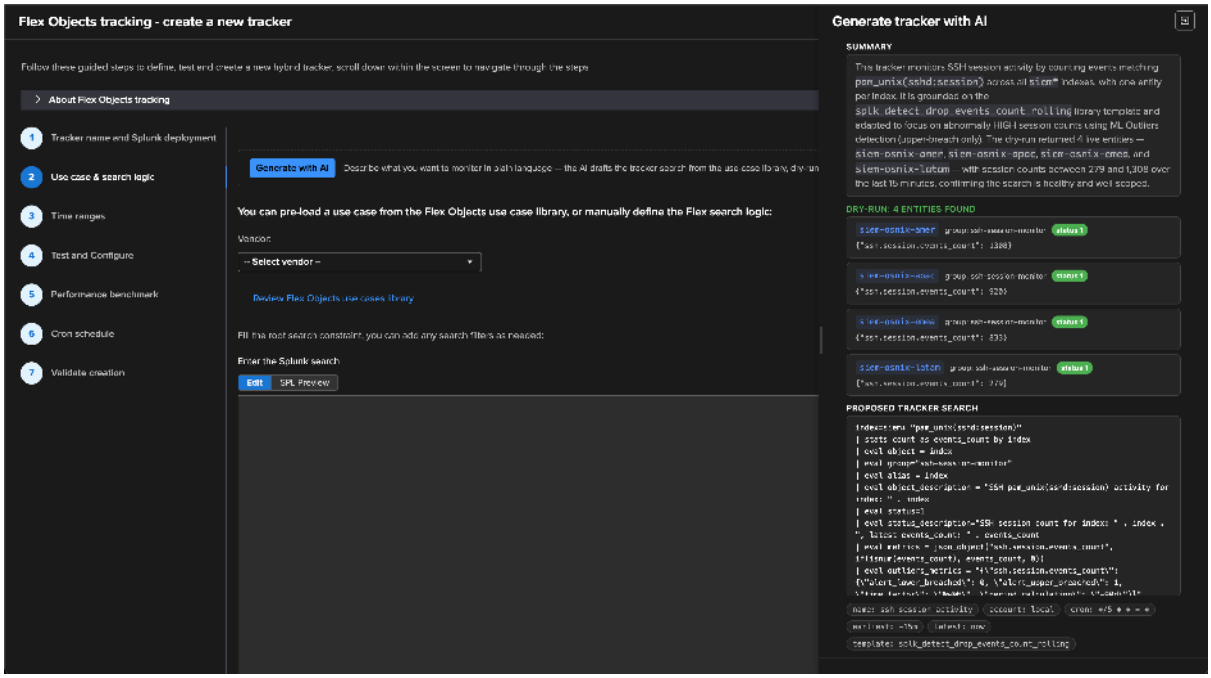
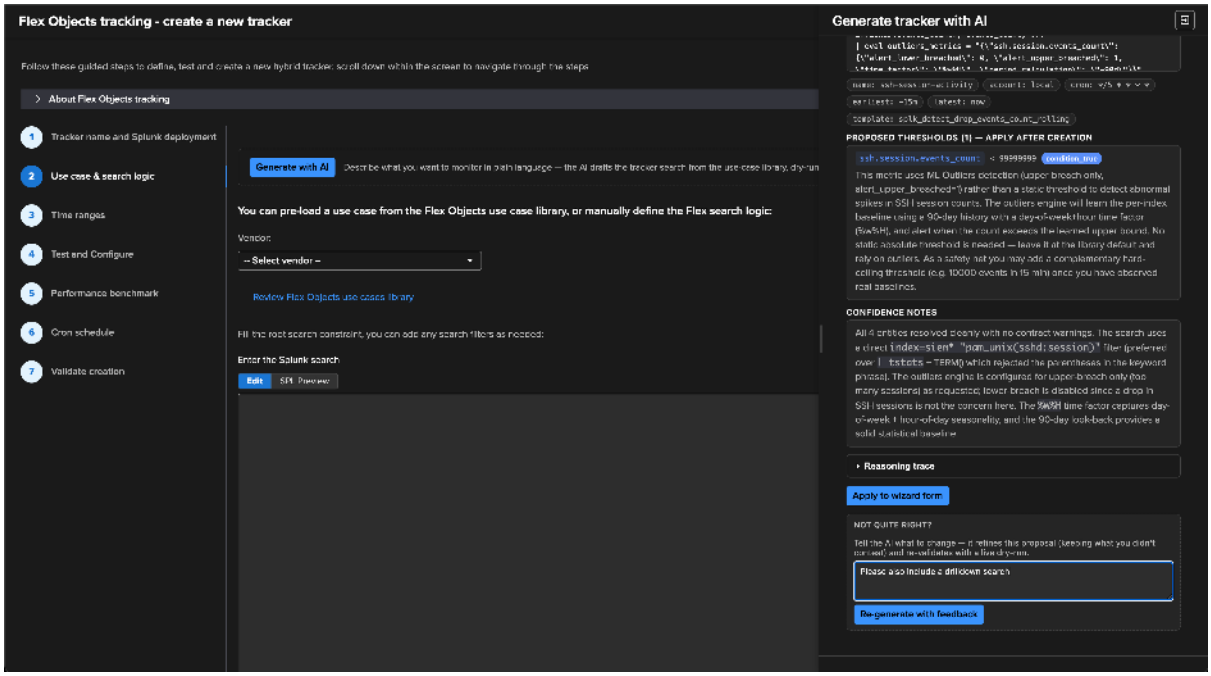
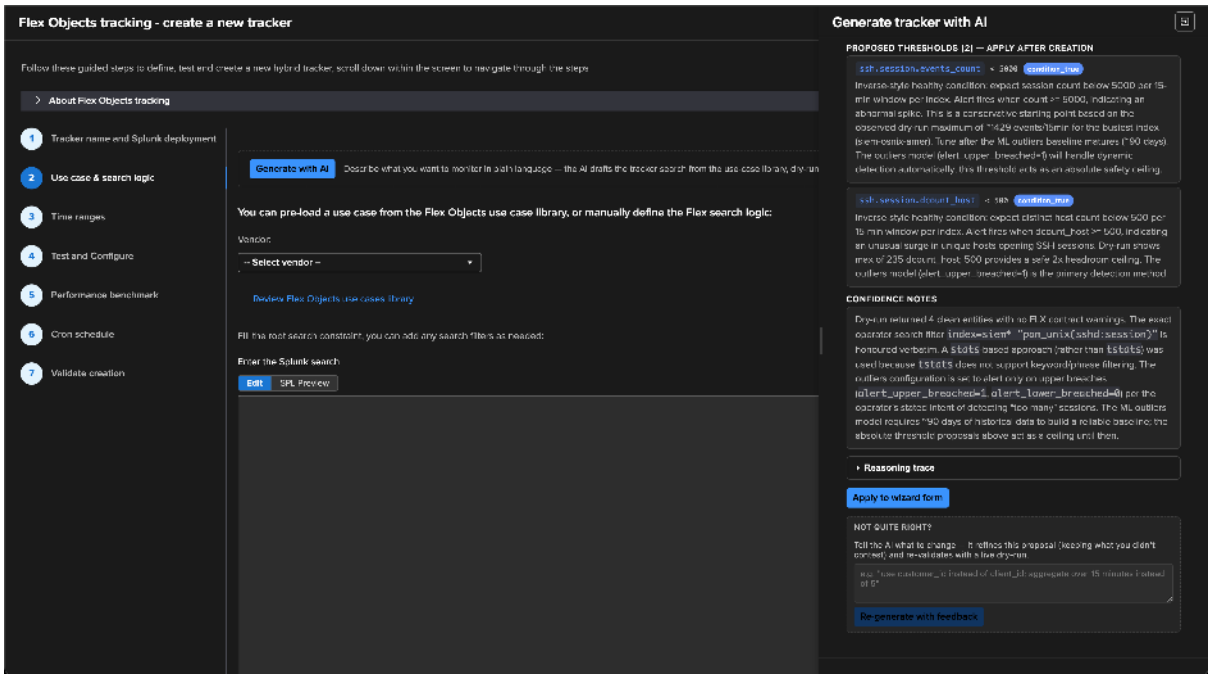
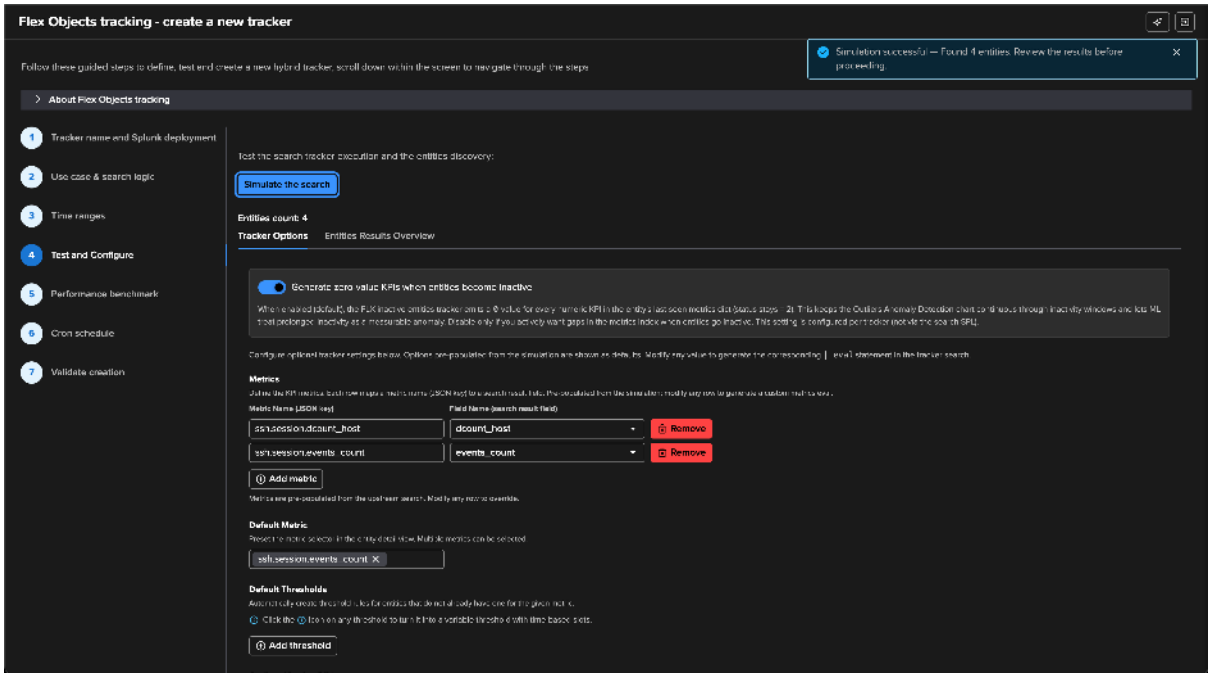
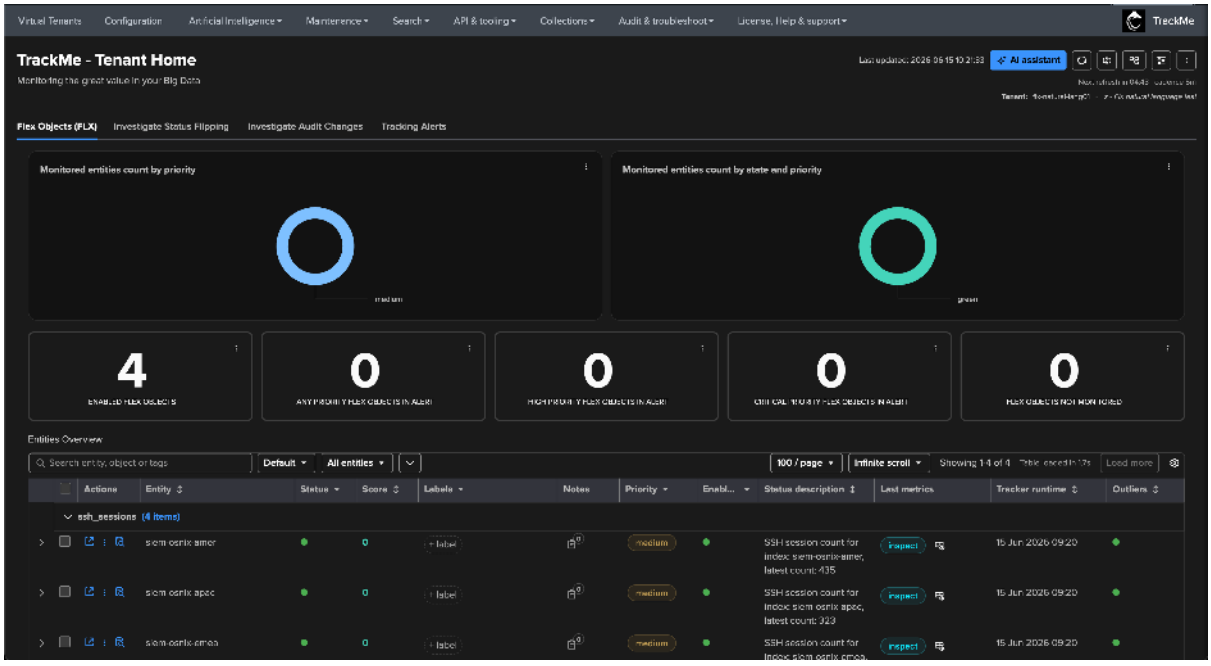
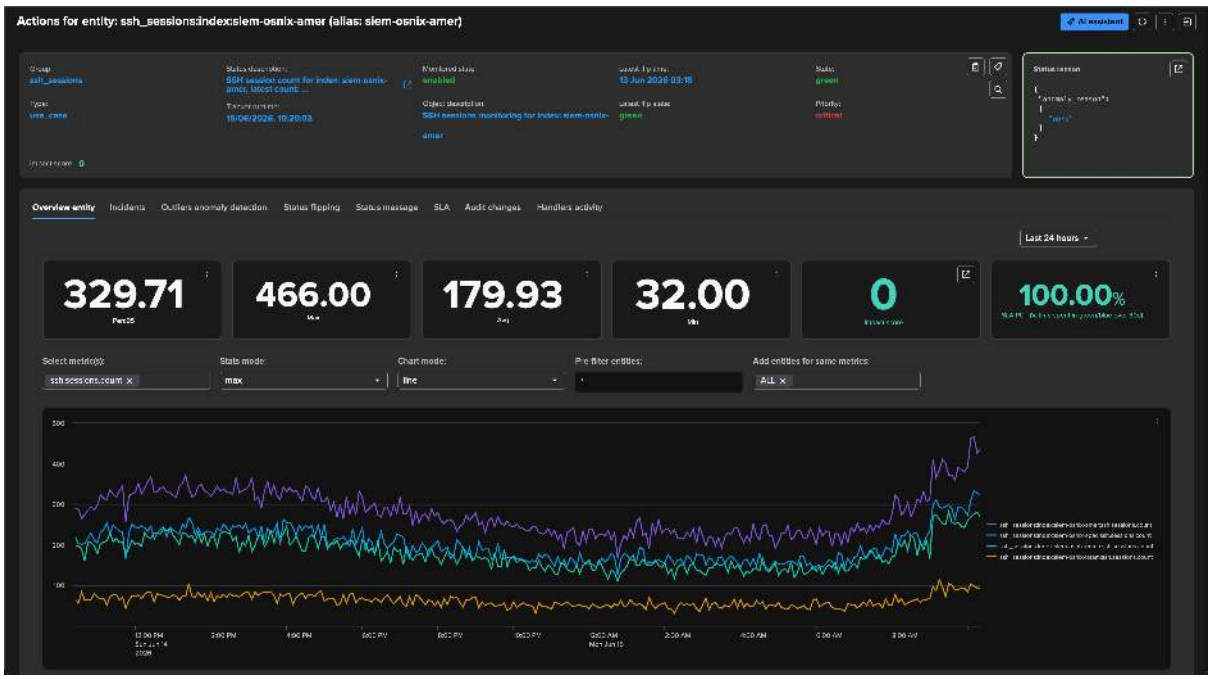
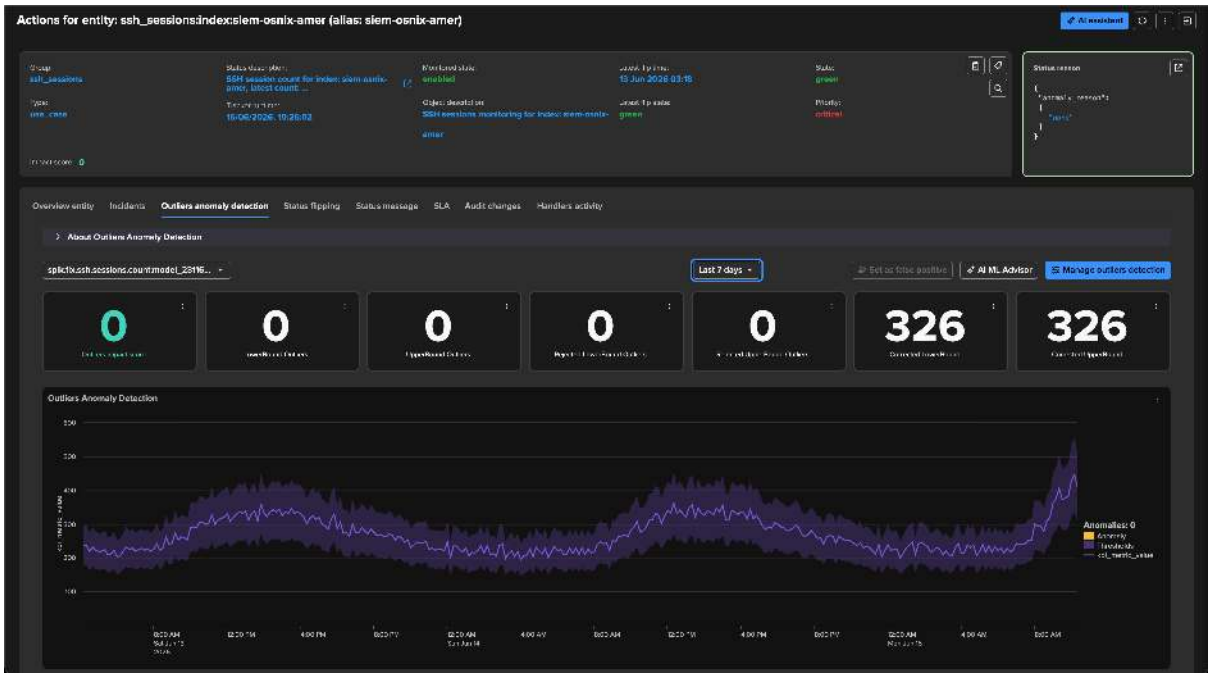
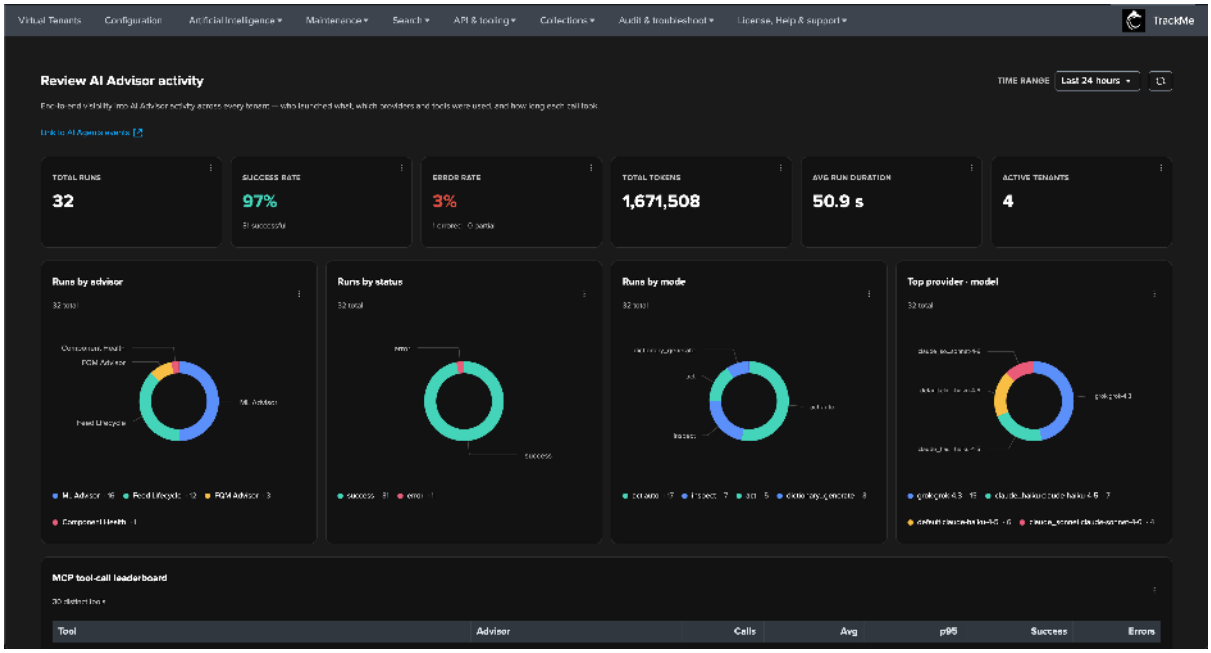
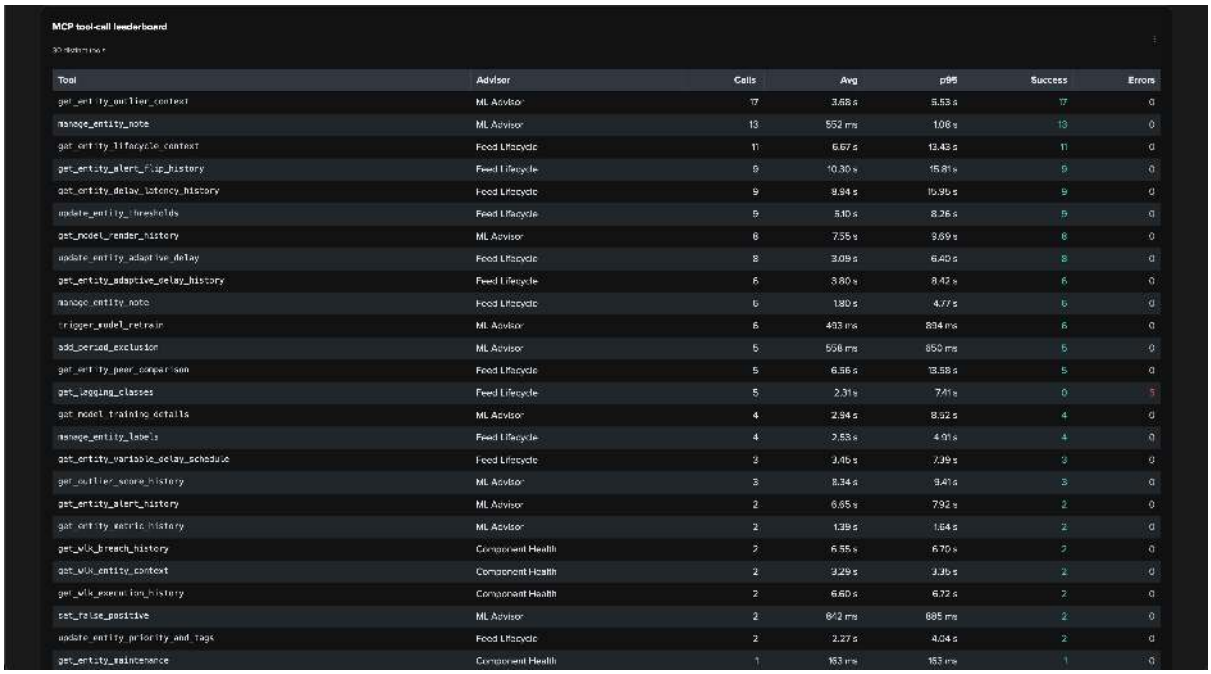
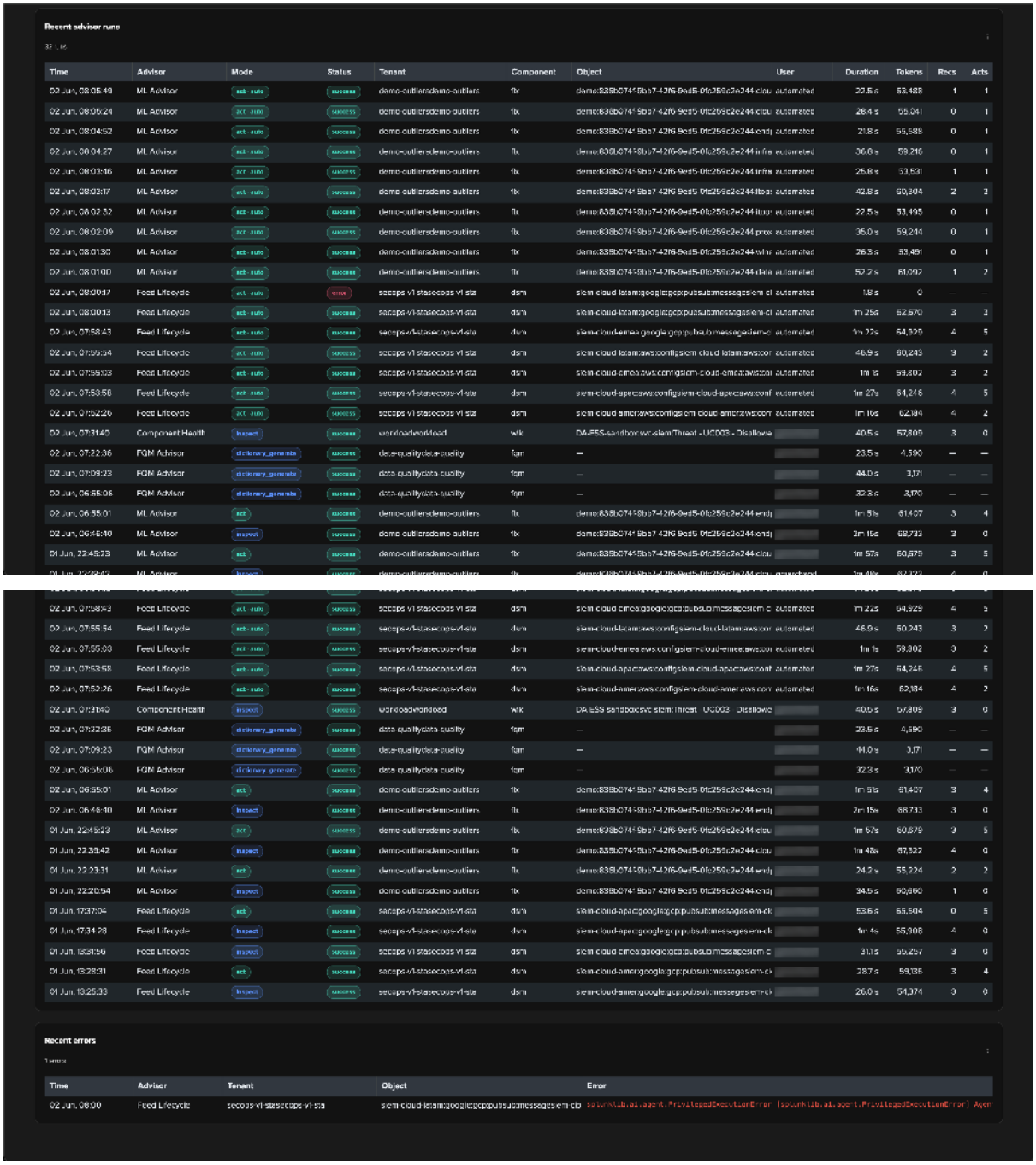
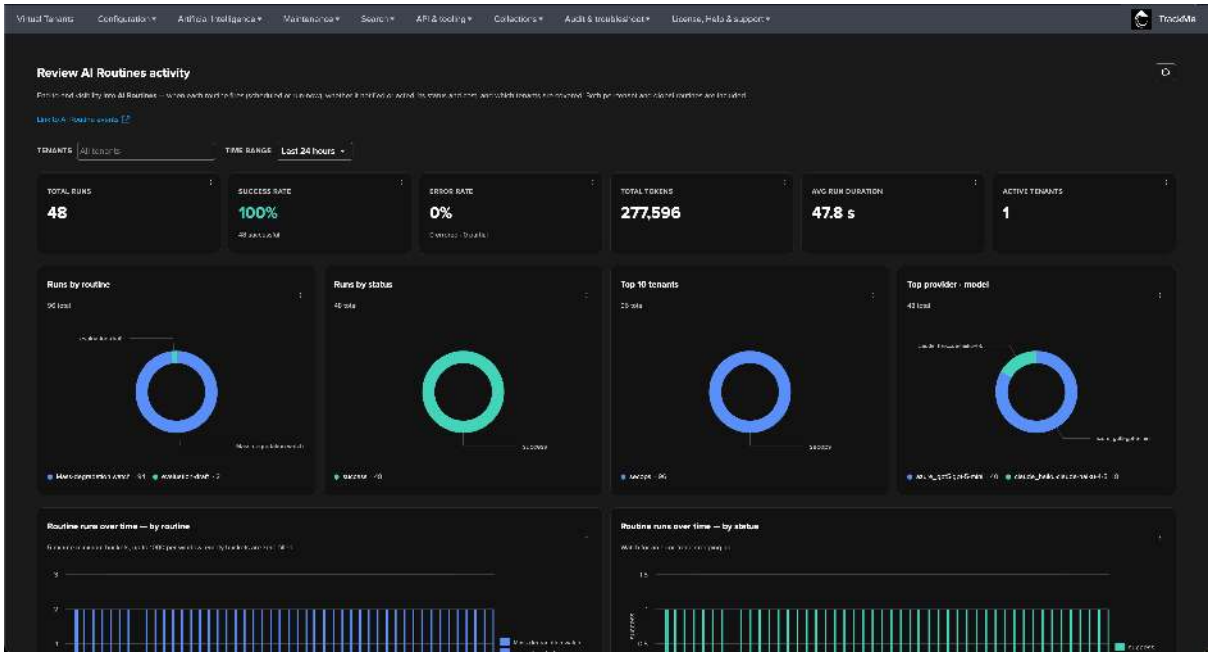
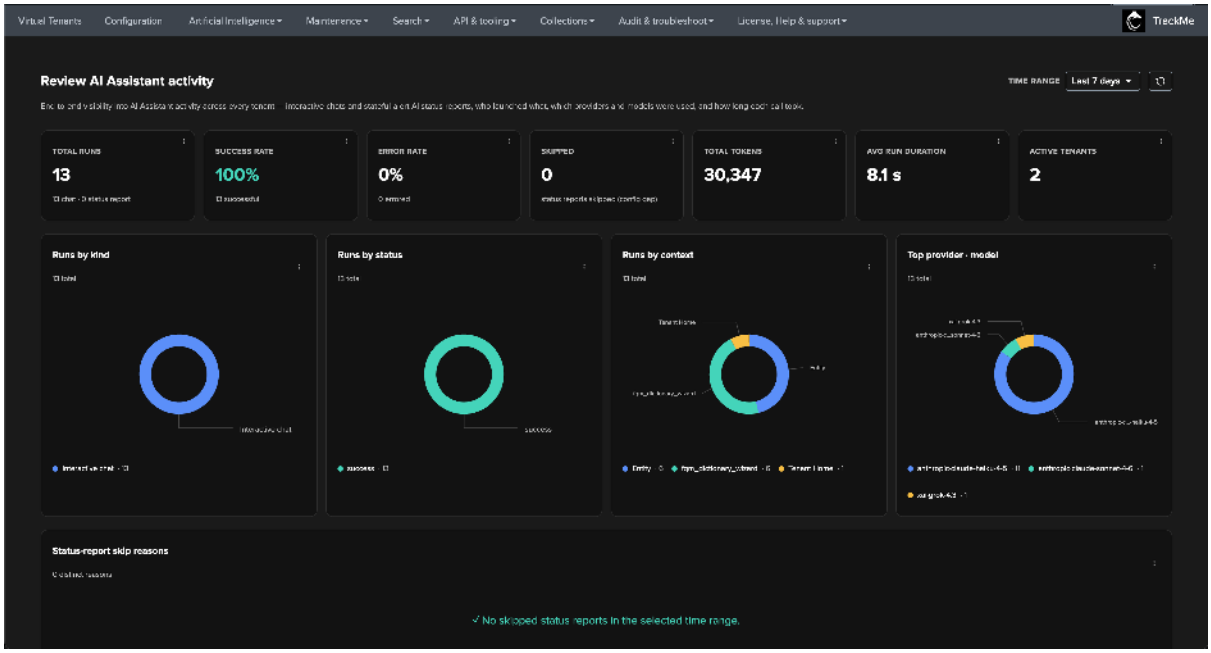
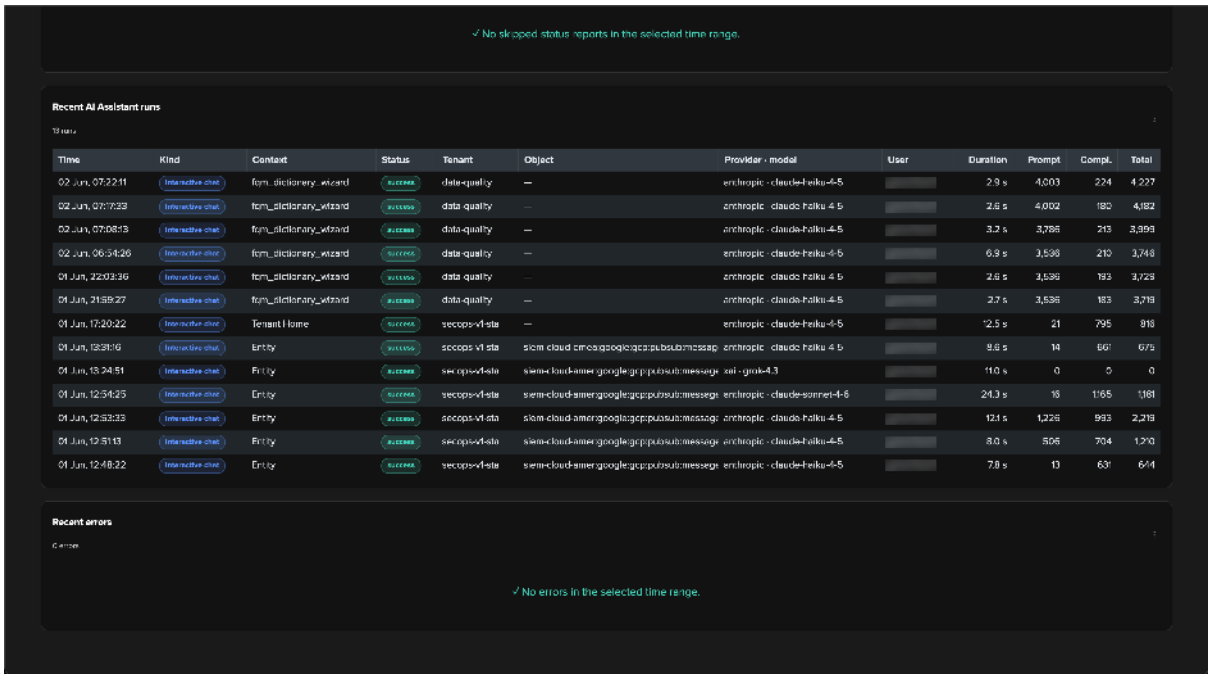
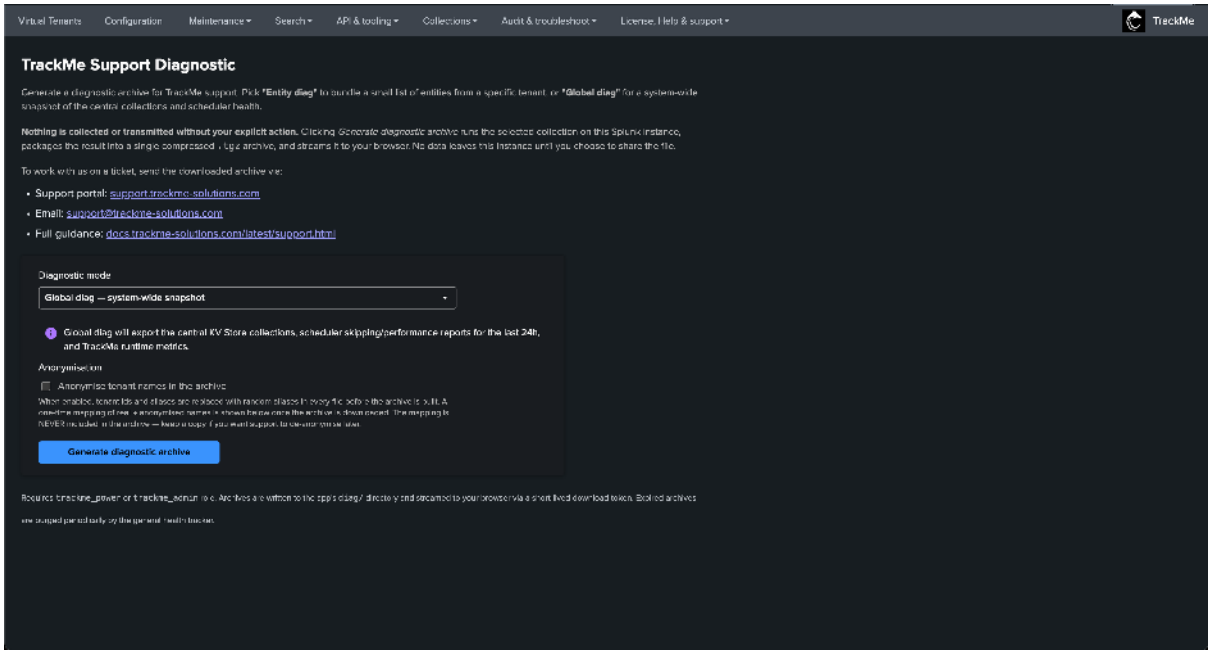
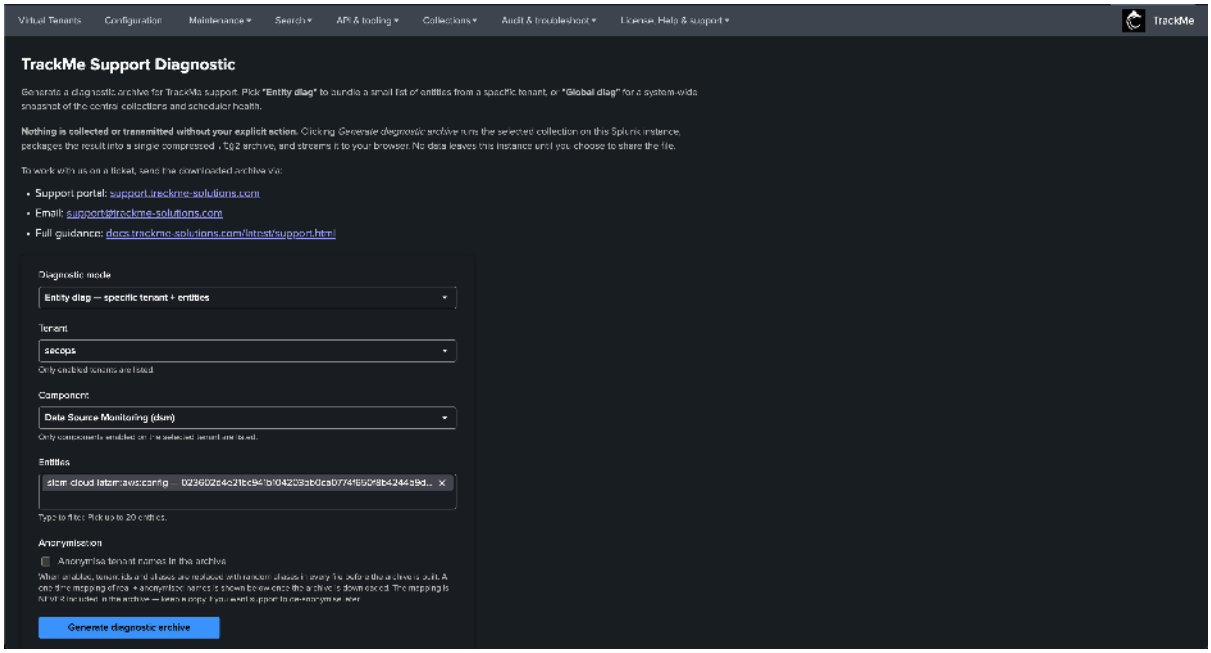
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, value, regex_expression
``` sort is mandatory to force all records to be retrieved before we call the gen␣
↪summary command ```
| sort 0 _time
| trackmefieldsqualitygensummary maxvals=15 fieldvalues_format=csv␣
↪groupby_metadata_fields="metadata.index,metadata.sourcetype,metadata.datamodel,metadata.nodename
| fields metadata.index, metadata.sourcetype, metadata.datamodel,␣
↪metadata.nodename, fieldname, total_events, distinct_value_count,␣
↪percent_coverage, field_values, regex_expression | fields - _time, _raw

| where fieldname="action"
```

This would be equivalent to run the following search:

```
index=summary sourcetype=trackme:fields_quality metadata.datamodel=Network_Traffic␣
↪metadata.nodename=All_Traffic metadata.sourcetype=cisco:asa metadata.source=*␣
↪metadata.host=* | head 100
```

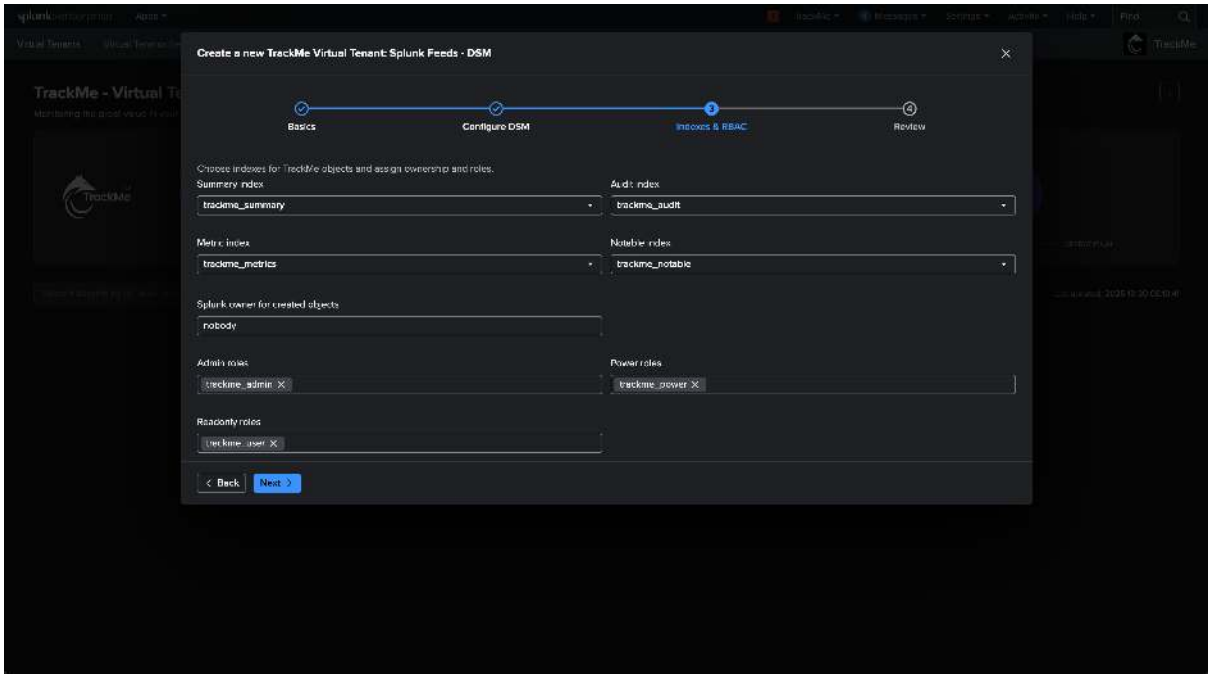
(continues on next page)
```

(continued from previous page)

```
| table _time, _raw
| trackmefieldsqualityextract
| table _time, metadata.index, metadata.sourcetype, metadata.datamodel, metadata.
↪nodename, fieldname, value, is_empty, is_missing

| where fieldname="action"

| stats count, count(eval(is_empty=0 AND is_missing=0)) as count_covered,␣
↪count(eval(is_empty=1 OR is_missing=1)) as count_not_covered
| eval percentage_covered=count_covered/count*100
```

2.2.4 Flex Objects - Adapt TrackMe to any monitoring use case (splk-flx)

Flex Objects: The Ultimate Tracking Component

- **Flex Objects (splk-flx)** is one of TrackMe's most powerful components, designed to transform the results of any Splunk search into fully tracked, monitored, and alerted entities.
- Unlike standard tracking components that follow a predefined schema, Flex Objects lets you define your own search logic, your own entities, your own KPIs, and your own anomaly detection rules.
- Flex Objects comes with a **library of 60+ pre-built use cases** covering Splunk infrastructure, data quality, license management, Splunk Cloud, SOAR, Cribl, and host monitoring - ready to deploy in minutes.
- **Enterprise Edition & Unlimited Edition** feature - Flex Objects is available to Enterprise and Unlimited edition customers.
- Flex Objects is particularly efficient for **Outliers detection**, **high-scale scenarios** with thousands of entities, and **custom grouping** strategies across any kind of data in Splunk.

Hint

Key references

- For a deep dive into the administration of Flex trackers, see *FLX — Flex Objects*.
- For Machine Learning Outlier detection concepts, see *Outlier detection*.
- For specific use case implementations, see the dedicated white papers referenced in *Real-World Use Cases*.

How Flex Objects Works

The Concept

Turn any Splunk search into fully tracked, monitored, and alerted entities.

Every Splunk environment is unique - different data sources, different infrastructure, different monitoring requirements. TrackMe's standard tracking components (splk-dsm for data sources, splk-cim for CIM compliance) cover the most common scenarios, but there will always be use cases that don't fit a predefined mold.

Flex Objects is the bridge between your Splunk search expertise and TrackMe's monitoring, alerting, and incident management capabilities. If you can write a Splunk search, you can create a Flex tracker.

✏ 1. Define Write any SPL search or pick from 60+ pre-built templates

2. Schedule TrackMe runs the search on a cron schedule

3. Discover Each result row becomes a tracked entity

4. Track KPIs, metrics, and ML Outlier detection

5. Alert Stateful alerting, emails, and active commands





Define or Select

Two paths to get started - pick a pre-built use case or write your own search.

Path 1: The Library - Flex Objects ships with **60+ pre-built use case templates** organized by vendor and category (Splunk Infrastructure, Data Quality, License, Cloud, SOAR, Cribl, Host Monitoring). Each template includes a complete Splunk search, pre-configured KPIs, and outlier detection rules. Select a template, customize the parameters to your environment, and deploy in minutes.

Path 2: Custom Search - Write any Splunk search that produces results following the Flex search contract. At minimum, each result must have an `object` (entity name) and a `status` (1=green, 2=orange, 3=red). Add optional fields for KPIs, descriptions, and ML anomaly detection as needed.

From the Library 60+ templates across 8 categories

Filter by vendor and category, preview the full search logic, customize and deploy.

Custom Search Any SPL producing entities

Use `tstats`, `| rest`, `mstats`, custom lookups, or any SPL command - if it returns rows, it can be a Flex tracker.





Tracker Types

Two types of Flex trackers for different monitoring needs.

When creating a Flex tracker, you choose between two fundamentally different approaches depending on your goal:

Use Case Tracker Entity discovery from a Splunk search

Your search produces one row per entity. Each entity is tracked individually with its own status, KPIs, metrics, and outlier detection. This is the standard Flex tracker for monitoring infrastructure, services, data quality, and more.

Converging Tracker Aggregated availability from existing entities

Correlates multiple Flex entities from one or more tenants, aggregating them into a single converging entity represented as a percentage of availability. Ideal for service-level monitoring and executive dashboards.





TrackMe Orchestrates

Once created, TrackMe takes over the heavy lifting.

Your Flex tracker becomes a scheduled report that TrackMe executes automatically. On every run:

- The Splunk search is executed against the target deployment (local or remote)
- Each result row is matched to an existing entity or discovered as a new one
- Entity states are persisted in the KVstore with full history
- KPIs defined in the `metrics` field are extracted and stored
- Metric events are ingested into the Splunk `trackme_metrics` index for long-term trending

- Entities that stop appearing are tracked for inactivity based on configurable thresholds

TrackMe automatically randomizes cron schedules to distribute load, handles batch KVstore operations for performance at scale, and manages the full entity lifecycle from discovery to deletion.





Metrics & ML

Track any number of KPIs per entity with built-in Machine Learning anomaly detection.

The `metrics` field in your Flex search defines Key Performance Indicators for each entity. TrackMe tracks these over time and ingests them as Splunk metric events, enabling trending, dashboards, and alerting on any KPI.

The `outliers_metrics` field activates **Machine Learning Outlier detection** on selected metrics:

- TrackMe builds a statistical model learning the normal behavior of each metric
- Dynamic upper and lower thresholds are calculated automatically

- **Time factors** add seasonality awareness - the model knows that Monday 9 AM looks different from Sunday 3 AM
- Anomalies are detected when current values fall outside the predicted range

KPIs Track CPU, event counts, delay, completion %, size, or any numeric metric per entity

Outlier Detection ML models learn normal behavior and alert on anomalies - no static thresholds needed

Seasonality Time factors (hour, day+hour, weekday+hour) ensure the model respects natural patterns





Incidents & Workflow

Flex entities are first-class citizens in the TrackMe Stateful Alerting workflow.

When a Flex entity changes status (green to orange/red, or an outlier is detected), TrackMe's **Stateful Alerting** engine manages the full incident lifecycle automatically:

- **Incident creation** - a new incident is opened and persisted in a state-aware manner
- **Incident updates** - as the entity condition evolves, the incident is updated with new context
- **Incident closure** - when the entity returns to a healthy state, the incident is automatically closed
- **Rich HTML email notifications** - with embedded metrics charts for immediate visual context
- **AI-generated status reports** - optional AI-driven analysis included in email notifications, adapted to the incident lifecycle stage (opened, updated, closed)
- **Active commands** - execute generating or streaming commands on incident open, update, or close for advanced integrations (ticketing, webhooks, etc.)
- **SLA tracking** - measure entity availability over time for compliance reporting

Flex entities also benefit from all TrackMe features: priority management, maintenance windows, logical groups, tags, status flipping analysis, audit changes, and the AI Assistant for intelligent investigation. See *Stateful alerting* for full details.





Hint

For detailed administration documentation including the search contract reference, creating trackers step-by-step, KPI configuration, ML Outlier detection setup, and managing trackers, see the *FLX — Flex Objects* guide.

Real-World Use Cases

Flex Objects powers a wide range of monitoring scenarios. The following white papers provide in-depth, step-by-step implementations:

White Paper	Description
Use TrackMe to detect abnormal events count drop in Splunk feeds	Detect abnormal drops in event counts using Flex Objects with ML Outlier detection. Covers both rolling and absolute metric approaches.
Analyse log messages logging level to detect behaviour anomalies using TrackMe's Flex Object and Machine Learning Anomaly Detection	Monitor Splunk internal log levels (INFO, WARN, ERROR, FATAL) with Flex Objects and detect unusual spikes using ML.
Tracking Splunk Cloud SVC consumption in TrackMe	Track Splunk Cloud SVC consumption at global and per-app levels using Flex Objects templates from the library.
Monitor Splunk Indexer Clusters	Monitor Indexer Cluster health, peer status, and bucket balance using Flex Objects with remote deployment accounts.
Monitor Splunk Search Head Clusters	Monitor Search Head Cluster global status, member health, and search activity using Flex Objects.
Use Case Demo: 360 Services Monitoring with TrackMe	A comprehensive 360-degree monitoring approach combining Flex Objects with other TrackMe components for full-stack visibility.

Summary

Flex Objects is the key to unlocking TrackMe's full potential for any monitoring use case. Whether you're starting from the pre-built library or writing custom searches from scratch, the component provides:

- **A simple search contract** - just `object` and `status` to get started, with optional `metrics` and `outliers_metrics` for advanced monitoring

- **60+ ready-to-use templates** - deploy pre-built use cases for Splunk infrastructure, data quality, license, cloud, SOAR, and Cribl monitoring
- **ML-powered anomaly detection** - detect anomalies in any KPI with configurable seasonality awareness
- **High-scale performance** - optimized for environments with thousands of entities
- **Converging visibility** - aggregate entity health into service-level availability metrics

Combined with TrackMe's alerting framework, virtual tenants, and SLA tracking, Flex Objects makes it possible to build a comprehensive, data-driven monitoring platform on top of Splunk.

For step-by-step administration instructions, see the *FLX — Flex Objects* guide.

2.3 TrackMe Use Cases Referential — Overview

What this referential is

This is a **comprehensive, cross-cutting map of everything TrackMe can be used for**, organised so an existing customer (or a prospective one) can audit their current footprint against the full capability surface and spot what they are not yet leveraging.

It is **not** a tutorial — for step-by-step walkthroughs, see the *Quickstart* and the various white papers. It is **not** a feature pitch either — every entry maps to a concrete TrackMe component, configuration touchpoint, and (where applicable) a pre-built use case template that ships in the product.

Audience
TrackMe administrators auditing coverage, Splunk SREs scoping a TrackMe rollout, architects and team leads prioritising which features to enable next.

Last updated
TrackMe v2.4.24

2.3.1 Why a referential

Most TrackMe customers use a fraction of what the platform offers. The combination of:

- **seven monitoring components** (DSM, DHM, MHM, VOL, FLX, FQM, WLK),
- **seven dedicated tenant types** that raise findings instead of tracking entities — User Activity Monitoring (UAM: who is doing what on the platform), Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge,
- **a deep library of cross-cutting features** (policies, ML outliers, stateful alerting, impact scoring, virtual tenants, configuration guardian, AI Assistant, remote Splunk fleets, SOAR integration, …),
- **65 pre-built FLX use case templates** that ship with the product,
- **a growing ecosystem of companion add-ons** (Cribl Stream / Edge, Google SecOps, lookups, XSOAR, conf-manager, SOAR connector),

means that "what TrackMe can do" is genuinely hard to summarise on a single page. This referential breaks it into five layers, each answering a different question.

2.3.2 The five layers

1. Use Cases by Intent "**What do I want to monitor?**" — organised by user goal: ingestion freshness, Splunk platform health, license / cost, scheduled-search health, field-parsing quality, custom KPIs, non-Splunk systems, intelligent alerting, multi-tenant scale, downstream integrations. The shortest path from a real-world question to the TrackMe component that answers it.

TrackMe Use Cases — Organised by User Intent 2. Capability × Component Matrix "**Which features apply to which monitoring components?**" — every cross-cutting capability (priority policies, SLA, variable delay,

ML outliers, hybrid trackers, AI status reports, …) mapped against the seven components, with one-line use cases unlocked and the configuration touchpoint for each.

TrackMe Capability × Component Matrix 3. FLX Template Catalog "**What pre-built use cases can I enable today?**" — the full catalog of the 65 FLX templates shipped in the product, grouped by category (Splunk infrastructure, license, Splunk Cloud, SOAR, Cribl Stream / Edge, good practices, host OS, …) with metrics, schedule, source data, dependencies, and ML-outlier flag per entry. Closes with a gap-analysis checklist.

FLX (Flex Objects) Use Case Template Catalog 4. Ecosystem & Companion Add-ons "**What else can I plug in?**" — the companion technology add-ons and the SOAR connector that extend TrackMe beyond Splunk: Cribl Stream / Edge, Google SecOps, lookup files, Cortex XSOAR, resilient configuration replay, and Splunk SOAR playbook orchestration of TrackMe entities.

TrackMe Ecosystem — Companion Add-ons 5. Adoption Roadmap "**What should I focus on next?**" — the six-phase customer journey we typically see (DSM feeds → DHM hosts → VOL license-volume outliers → optional WLK interlude → FLX expansion and third-party → steady state). Descriptive, not prescriptive; each phase opens with the customer question that triggers it, calls out anti-patterns to avoid, and notes Foundation / Enterprise / Unlimited edition variants.

TrackMe Adoption Roadmap

2.3.3 How to use this referential

For a **coverage audit** of an existing TrackMe deployment, work through the layers in this order:

1. **Start with the FLX template catalog** (layer 3) — the most concrete "what am I missing?" pass. The gap-analysis checklist at the bottom of that page walks you through every domain (license, indexer cluster, ingestion, SH workload, good practices, data quality, Cribl, SOAR, host OS) one question at a time.
2. **Then read the capability matrix** (layer 2) — once you know which components and templates you are using, cross-reference whether you have priority policies, SLA classes, ML outlier detection, logical groups, disruption queues, smart status investigations, AI status reports, etc. wired in for those components. The matrix shows you the per-component support and the configuration touchpoint.
3. **Use the by-intent catalog** (layer 1) when a stakeholder asks a question in plain English ("can we monitor X?") — find the goal, follow the cross-references.
4. **Consult the ecosystem chapter** (layer 4) when your scope extends beyond a single Splunk instance — Cribl, Google SecOps, SOAR / XSOAR integration, lookup-file monitoring, etc.
5. **Cross-check the adoption roadmap** (layer 5) to position yourself — "we are solid on phase 1 and phase 2, just starting on phase 3" — and to see what we typically teach customers about next.

For **scoping a new TrackMe rollout** or **planning an extension**, the adoption roadmap (layer 5) is the right starting point — it tells you which capabilities to focus on first and which to defer. Then use the capability matrix to set expectations, by-intent to align on goals with stakeholders, the FLX catalog to identify quick wins from pre-built templates, and the ecosystem chapter for integrations and external data sources.

2.3.4 How TrackMe is structured (recap)

The referential assumes you are familiar with TrackMe's core concepts. If not, the very short recap below is enough to navigate the five layers; each concept also links to its deeper documentation.

The seven monitoring components

Every TrackMe entity belongs to one of seven monitoring components. The three-letter codes are load-bearing — they appear in REST paths, KV collections, filenames, and the wizard UI.

Code	Full name	What it monitors
DSM	Data Source Monitoring	Data feeds modelled as `(index, sourcetype)` combinations — the fundamental "is my data flowing" axis. Also supports a `lookups` search mode that tracks CSV / KVstore lookups as entities.
DHM	Data Host Monitoring	Per-host × per-sourcetype activity — "who is sending what". Hybrid trackers in 2.3.23+ support an optional `breakby_extra_fields` dimension on top of the combo grain.
MHM	Metric Host Monitoring	Hosts sending **metric** data, grouped by metric category — the counterpart of DHM on metric indexes.
VOL	Volume Outliers	The licensed volume of every index, read incrementally from the license usage logs in 5-minute buckets — ML outliers (drops and / or spikes, direction chosen per index), inactivity, daily licensed volume, 30-day trend and month projection. One entity per index; the history is backfilled at creation. See *VOL — Volume Outliers*.
FLX	Flex Objects	Any Splunk SPL turned into tracked entities with custom KPIs. Sixty-five pre-built templates ship in the catalog (layer 3).
FQM	Field Quality Monitoring	Field parsing quality and CIM compliance — coverage %, success %, per-field metrics.
WLK	WorkLoad Knowledge	Scheduled-search execution health — skip rate, error count, execution delay, version drift, orphan owners, app enablement.

Beyond the entities — the dedicated tenant types. Seven *tenant types* are not components: their tenants have no entities, and their policies raise **findings**, triaged as Finding groups (see *Findings, Finding groups and findings actions*).

- **User Activity Monitoring** (UAM, 2.4.18+, generally available since 2.4.19) reads the audit trail and the introspection of a deployment through one or more search head tiers and answers *who is doing what* — accounts classified as human, service or system, their scheduled searches and knowledge objects, their search activity and resource usage — with sixteen built-in policies plus custom policies in SPL, state-aware notifications and the UAM AI Agent. See *User Activity Monitoring*.
- **Cribl Stream** (2.4.19+) monitors a Cribl Stream deployment — the Leader, Worker Groups, Workers, Sources, Destinations, Pipelines and Routes — from the Cribl metrics in a Splunk metric index and the Leader API, with traffic machine learning models and the Cribl Stream AI Agent. See *Cribl Stream*.
- **Cribl Edge** (2.4.19+) monitors the Fleets of a Cribl Edge Leader and their Edge Nodes, at any fleet size. See *Cribl Edge*.
- **Splunk HEC** (2.4.21+) monitors the HTTP Event Collector endpoints, pools and tokens of a Splunk deployment from the telemetry Splunk already writes, with eighteen built-in policies and traffic models. See *Splunk HEC*.
- **Splunk S2S** (2.4.22+) watches the receivers, the intermediate forwarders and the pools they form, and names the blocked queue and its origin. See *Splunk S2S*.
- **Splunk UF Fleet** (2.4.22+) follows every forwarding agent and the Deployment Servers that manage them, at any fleet size, with mass events as one finding. See *Splunk UF Fleet*.
- **Splunk Forwarder Collectors** (2.4.24+) follows the forwarders used as collection points and every input they run — add-ons, scripted and modular inputs, syslog files. See *Splunk Forwarder Collectors*.

Every dedicated tenant type has its own AI Agent, and the six other than UAM can save their generated maps to Topology Studio.

Virtual Tenants

Every TrackMe deployment is sliced into **Virtual Tenants** — isolated containers each with their own KV collections, RBAC roles, indexes, scheduling, default thresholds, and (optionally) a remote Splunk deployment target. Components are enabled per tenant. See *Creating a tenant* and *Managing tenants*.

State, scoring, alerting

Every entity is in one of four states — `green` (healthy), `orange` (warning), `red` (critical), or `blue` (protected by group or grace window). The state is determined by a hybrid score that adds a rolling `base_score` from the metrics index to per-condition impact scores and threshold scores. See *QUICK START - TrackMe Impact-Based Alerting (IBA)* and *Stateful alerting*.

Stateful alerting

State transitions (green → orange / red, red → green, …) drive a single stateful record per `(entity, alert configuration)`. The alert action fires "Alert Opened", "Update", and "Alert Closed" notifications — optionally enriched with embedded SVG → PNG charts and LLM-generated AI status reports. See *Stateful alerting*.

License editions

Feature availability is gated by license edition. The referential calls out edition-specific gates inline where relevant. The short version:

Edition	Type	Notes
Foundation	Subscription / 90-day trial	DSM, DHM, MHM, VOL only. No FLX, no FQM, no WLK. No remote deployments. No hybrid trackers.
Enterprise	Subscription	All seven components. FLX, FQM and WLK enabled. Hybrid trackers capped at 8 per tenant.
Unlimited	Subscription	All features, unlimited scale, remote deployments.
Developer	Self-service / 30 days	All features for local development; max 3 activations per instance.

See *EULA License* and *License registration* for details.

2.3.5 What this referential does NOT replace

- **Step-by-step walkthroughs.** The *Quickstart* and the white papers remain the right place to learn TrackMe.
- **Component-specific deep dives.** Each component has dedicated admin / user guide pages: *FLX — Flex Objects*, *WLK — Workload Knowledge*, *Use Case Demo: Fields Quality (CIM and non-CIM)*, *Use Case Demo: This is All About Feeds*, etc.
- **REST API documentation.** See *REST API reference* and the in-product REST API Reference page.
- **Release notes.** See *Release notes: trackme*.

The referential **complements** those documents by giving you a single flat map of the capability surface, with cross-references back into the existing documentation wherever you want depth.

2.3.6 Conventions used across this referential

- Component codes (`DSM`, `DHM`, …) are rendered as inline code.
- KV collection names are rendered as inline code: `kv_trackme_dsm_tenant_{tenant_id}`.
- REST endpoints use the `/trackme/v2/...` prefix and are rendered as inline code.
- Cross-references to other documentation pages use Sphinx `:ref:`<label>` so they stay live across documentation versions.
- FLX template references use their `uc_ref` (filename without `.json`) — for example `splk_license_pool_usage` corresponds to `package/lib/flx_library/splk_license_pool_usage.json`.

TrackMe Use Cases — Organised by User Intent

How this page is organised

Each section answers a real-world question ("I want to know when…", "I want to monitor…", "I want to integrate with…"). For each goal we give you the **shortest path** from intent to action:

- which **component(s)** deliver the use case,
- whether it is **out-of-the-box**, **configurable**, or requires **custom SPL**,
- a pointer to a **pre-built FLX template** if one exists (see *FLX (Flex Objects) Use Case Template Catalog* for full details),
- and a cross-reference to the deeper admin / user / white-paper guide.

Cadence
This page changes every time TrackMe grows new capabilities. Current snapshot: TrackMe v2.4.24.

I want to know when data stops flowing

The foundational TrackMe use case. The platform was built around this problem, and three of the seven components answer different facets of it.

Detect when an `(index, sourcetype)` feed stops receiving data

Component(s)
`DSM`

Status
Out-of-the-box. Auto-discovered by the DSM tracker.

How
Each `(index, sourcetype)` becomes a DSM entity; the tracker measures ingestion delay, latency, and event count. Static or variable delay thresholds trip `orange` / `red` state.

Tuneables
variable delay (time-of-day / day-of-week slots), lagging classes, adaptive delay (auto-recompute thresholds from history), priority / SLA / tag policies, monitoring-time policy.

See also
Use Case Demo: This is All About Feeds, *Impact scoring & thresholds*, *Impact scoring & thresholds*, *DSM — Data Source Monitoring*.

Detect when a specific host stops sending data

Component(s)
`DHM`

Status
Out-of-the-box. Hosts × sourcetypes are auto-discovered.

How
One DHM entity per host × sourcetype combination (or per host alone in merged mode). 2.3.23+ adds the optional `breakby_extra_fields` knob for richer combos (e.g. include `source`).

Tuneables
same as DSM (variable delay, lagging classes, adaptive delay, policies, monitoring-time).

See also
Notes, *Hybrid trackers*.

Detect silent metric-sending hosts

Component(s)
`MHM`

Status
Out-of-the-box. `mstats`-based discovery and tracking.

How
One MHM entity per host × metric category. Detects when a host stops emitting metrics that it previously emitted.

Tuneables
static delay thresholds, priority / SLA / tag policies. Variable delay and lagging classes are **not** supported on MHM.

See also
Notes.

Detect silent CSV or KVstore lookup files

Component(s)
`DSM` in `search_mode=lookups`

Status
Configurable. Requires the companion add-on `TA-trackme-lookupmonitor` installed on the SH.

How
The hybrid-tracker wizard exposes a `Lookups monitoring` mode that turns every CSV file and KVstore collection into a DSM entity, with filesystem mtime and accurate row count.

Tuneables
same as regular DSM (priority, SLA, …); thresholds apply on delay only — latency is neutralised because lookups have no `_indextime`.

See also
Lookup monitoring, *QUICK START - Design and automate through lookup knowledge*.

Detect event drop-outs at the index level (independent of host / sourcetype)

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_detect_drop_events_count_absolute` (hourly window), `splk_detect_drop_events_count_rolling` (rolling 5-minute), and `splk_hosts_tracking` (per-host).

How
Computes per-index event count and distinct host count, with native ML outlier detection on the volume so abnormal day-over-day drops are surfaced even if the static threshold would not have caught them.

See also
Use TrackMe to detect abnormal events count drop in Splunk feeds.

Detect routing or parsing misconfiguration (events in `lastchanceindex`)

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates
`splk_lastchanceindex`.

How
Watches the `lastchanceindex` for any traffic; non-zero counts are a strong signal that a pipeline misroutes events.

I want to monitor Splunk platform health (the full stack)

TrackMe ships an opinionated set of FLX templates that cover the most common "is Splunk itself healthy?" questions. All of them run as scheduled FLX entities and participate in TrackMe's stateful alerting, ACK system, smart status, AI Assistant and impact scoring.

Tip

Four dedicated tenant types now answer the forwarding and ingestion questions out of the box, with findings, Finding groups and an AI Agent each, nothing to write: *Splunk HEC* (HTTP Event Collector endpoints, pools and tokens, refusals and back pressure), *Splunk S2S* (receivers, intermediate forwarders and their pools, the blocked queue and its origin), *Splunk UF Fleet* (every forwarding agent, its version and its Deployment Servers) and *Splunk Forwarder Collectors* (heavy forwarders and syslog collectors, and every input they run, DB Connect included). The FLX templates below remain the way to track these signals as entities, or anything the dedicated tenants do not cover.

Indexer cluster status (Splunk Enterprise)

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_splunk_enterprise_cluster_status` (global — availability, peers up / down / unknown), `splk_splunk_enterprise_cluster_peers_status` (per-peer — bucket %, primary %, balancing deviation, plus introspection CPU / mem join).

See also
Monitor Splunk Indexer Clusters.

Search-head cluster status

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_splunk_shc_global_status` (captain-level flags: maintenance mode, min_peers_joined, rolling_restart, service_ready, kvstore_maintenance), `splk_splunk_shc_members_status` (per-member lost / down / pending).

Dependency
A TrackMe **remote account** targeting any SHC member is required for both templates.

See also
Monitor Splunk Search Head Clusters, *Remote Splunk deployments*.

Bucket health (timestamp parsing, premature rolls, quarantine)

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates
`splk_bucket_health_per_index` — every 30 min, `| dbinspect`-based; surfaces premature small buckets and quarantined buckets, both classic timestamp-parsing failure symptoms.

Knowledge-bundle size on the SH tier

Component(s)
`FLX`

Status
Pre-built FLX template (with third-party dependency).

Templates
`splk_bundle_size`. Requires the Splunkbase app `sa-littlehelper` (#6368) which provides `| bundlefiles`.

Why it matters
bloated bundles silently slow search dispatch across the entire SH tier.

KV-store readiness and per-collection size

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_kvstore_status` (service readiness — alerts when KV goes degraded / down), `splk_kvstore_size` (per-collection vs total size with thresholding).

Pipeline queue saturation

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates
`splk_queues_filling` — parsing / aggregator / typing / indexer queue fill metrics. The canonical backpressure indicator meaning the indexer tier needs more horsepower or the input is ingesting too fast.

Parsing issues (LineBreaking / Aggregator / DateParser)

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates

`splk_parsing_issues_per_component` — hourly aggregation of WARN / ERROR from the canonical bad-props.conf symptom signatures.

Data Model Acceleration (DMA) completeness

Component(s)

`FLX`

Status

Pre-built FLX template.

Templates

`splk_dma` — complete %, size, run duration, bucket count with ML outlier detection. Surfaces stuck or under-performing accelerations.

HEC ingestion errors

Component(s)

`FLX`

Status

Pre-built FLX template.

Templates

`splk_splunk_hec_errors` — parsing-error breakdown by `parsing_err`, with source-IP dcount to localise blame.

Universal Forwarder fleet version compliance and dropout

Component(s)

`FLX`

Status

Pre-built FLX templates.

Templates

`splk_uf_clients_versions_tracking` (UF version compliance — picks highest version as target, alerts on % non-compliant), `splk_deployment_server_clients` (silent UF dropout — alerts on missing phone-home).

Search-head workload and scheduler health

Component(s)

`FLX`

Status

Pre-built FLX templates.

Templates

`splk_search_head_activity` (dispatched, eligible, lag, runtime, scheduler cycles, with ML outliers), `splk_search_head_healthmon_artifact_count_remote` (per-SHC-member response time and artifact count via remote search).

See also

Monitor Splunk Workload with TrackMe's Workload component.

Per-instance CPU and memory consumption

Component(s)

`FLX`

Status
Pre-built FLX templates.

Templates
`splk_splunk_infra_cpu_used`, `splk_splunk_infra_mem_used` — Splunk-introspection-based per-instance avg / p95 / max.

Distinction
Use these for the **Splunk process** view. For the host OS view of the underlying servers, use the `ta_nix_*` and `ta_windows_*` templates (see the host OS section below).

Log-level variations (catch error storms early)

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates
`splk_splunk_infra_log_level_variations` — counts INFO / WARN / ERROR / FATAL across the Splunk infra with ML outliers for trend deviation.

See also
Analyse log messages logging level to detect behaviour anomalies using TrackMe's Flex Object and Machine Learning Anomaly Detection.

DB Connect modular-input health

Component(s)
`FLX`

Status
Pre-built FLX template (with dependency).

Templates
`splk_dbconnect`. Requires Splunk DB Connect.

I want to monitor cost and license consumption

Since TrackMe 2.4.18 the **VOL** (Volume Outliers) component answers the per-index licensed-volume question out of the box, on every edition including Foundation. Licensing also remains one of the most heavily templated areas of the FLX library — six dedicated templates plus the daily-variation pair, covering both Splunk Enterprise and Splunk Cloud — for pool-level, SVC and storage views, and as the legacy alternative to VOL for custom needs.

Detect abnormal drops or spikes of licensed volume per index — know who consumes my license

Component(s)
`VOL`

Status
Out-of-the-box. Enable the component on a tenant (local or remote deployment, Splunk Enterprise or Splunk Cloud); every index becomes an entity, the history (default 30 days) is backfilled at creation.

How
One small incremental search per tenant every 5 minutes reads the license usage logs in 5-minute buckets. Native ML outliers on the rolling volume (drops, spikes, or both — chosen per index) and an inactivity policy (static or by day / hour, overridable per index) drive the entity state; the daily licensed volume, a 30-day trend and a month projection give the cost reading. The **Global license usage (VOL)** tab shows key figures, top consumers, movers, pools and quotas.

Tuneables
detection direction per index, inactivity policy, `vol_mloutliers_priority_filter` / `vol_mloutliers_filter_expression`, priority / SLA / tag policies.

See also
VOL — Volume Outliers, *Volume Outliers — in depth*, *Volume outliers on the Splunk license usage with Volume Outliers (VOL)*. The FLX license templates below remain the legacy path (see the white paper for the comparison).

Splunk Enterprise license pool consumption

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_license_pool_usage` (per-pool % used + GB used, with ML outliers), `splk_license_usage_global_splunkenterprise` (rolling 60 min / 4 h / 24 h — designed for near-real-time anomaly detection), `splk_license_usage_per_index_splunkenterprise` (same but per-index to pinpoint a spike).

Splunk Cloud license, storage and SVC consumption

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_license_usage_global_splunkcloud`, `splk_license_usage_per_index_splunkcloud`, `splk_splunk_cloud_storage_usage` (storage vs licensed), `splk_splunk_cloud_svc_usage` (overall SVC usage % with outliers), `splk_splunk_cloud_svc_usage_by_app` (per-app SVC attribution), `splk_splunk_cloud_svc_usage_by_consumer_service` (per-consumer attribution).

See also
Tracking Splunk Cloud SVC consumption in TrackMe.

Daily variation detection on global / per-index volume

Component(s)
`FLX`

Status
Pre-built FLX templates.

Templates
`splk_detect_daily_variations_volume_global`, `splk_detect_daily_variations_volume_index`, `splk_volume_ingested_per_day`. All ML-outlier-enabled and scheduled in the 03:05 — 04:15 UTC quiet hours. For the per-index variant, prefer the **VOL** component (above) — it covers the same signal continuously, with backfill, on every edition.

Detect oversized lookup files before they hurt bundle replication

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates

`splk_large_lookup_files` — every 12 hours via `/services/data/transforms/lookups`, with ML outliers on total lookup size.

I want to monitor scheduled-search execution health

The `WLK` (WorkLoad Knowledge) component is purpose-built for this and includes signals you cannot get from FLX alone.

Detect scheduled searches that skip, error, or run long

Component(s)

`WLK`

Status

Out-of-the-box. Auto-discovered.

How

Every scheduled search becomes a WLK entity with skip rate, error count, execution-time metrics. Dynamic thresholds support a **tenant-default fallback row** (`object_id == "default"`) so you can manage the whole tenant's thresholds from one record and override per entity as needed.

See also

WLK — Workload Knowledge, *WLK — Workload Knowledge*, *Monitor Splunk Workload with TrackMe's Workload component*.

Detect silently broken scheduled searches

Component(s)

`WLK`

Status

Out-of-the-box. Three WLK-only signals fire here.

How

`orphan` (owner user no longer exists), `app enablement` (parent app disabled — search no longer dispatches), `versioning` (savedsearch definition has changed since the fingerprint was taken, surfacing unannounced edits).

Audit "all-time" or user-owned scheduled searches (good-practice violations)

Component(s)

`FLX`

Status

Pre-built FLX templates.

Templates

`splk_good_practices_alltime_scheduled` (enabled scheduled searches with `-` / `all time` earliest — a notorious cost driver), `splk_good_practices_run_as_user_scheduled` (scheduled searches running as a named user — offboarding fragility), `splk_good_practices_detect_dynamic_sourcetype` (dynamic / numerically suffixed sourcetypes that bloat metadata).

I want to monitor field-parsing quality and CIM compliance

This is the dedicated remit of the `FQM` component.

Track field coverage and parsing success per dataset

Component(s)
`FQM`

Status
Out-of-the-box (with custom templates per CIM data-model).

How
FQM defines field-quality checks per (source, sourcetype, host), producing percentage_passed / percentage_failed and per-field counts. CIM-aware templates ship in the FQM wizard so common data models (Authentication, Network_Traffic, Web, …) can be onboarded quickly.

Tuneables
priority, SLA, tag policies, monitoring-time policy, dynamic thresholds per metric.

See also
Use Case Demo: Fields Quality (CIM and non-CIM).

Aggregate FQM outcomes into a single FLX-tracked health metric

Component(s)
`FLX`

Status
Pre-built FLX template.

Templates
`splk_splunk_fields_quality` — aggregates the FQM check outcomes into per-(source, sourcetype, host) % passed / % failed and applies ML outliers for drift detection. Useful when you want one aggregate view next to your other FLX entities.

I want to detect anomalies without writing thresholds by hand

Native ML outlier detection (DensityFunction)

Component(s)
`FLX` recommended; `VOL` by design (the outlier verdict is the entity state, enabled by default, available in all editions) and the recommended route for volume anomalies; `DSM` optional (off by default); `DHM` selective; `MHM` / `FQM` / `WLK` not recommended.

Status
Configurable. Enabled per tenant via `vtenant_account.mloutliers_<component>`; VOL has its own scope (`vol_mloutliers_priority_filter` / `vol_mloutliers_filter_expression`).

How
TrackMe trains a per-entity, per-metric DensityFunction model (Gaussian KDE / Normal / Exponential / Beta) with optional seasonality (hour-of-day, day-of-week, dow × hod). Low-confidence predictions are suppressed so a half-trained model does not page the operator.

Pre-wired templates
20 of the 65 FLX templates ship with `outliers_metrics` already declared — license / volume, Cribl traffic, SOAR runtime, SH activity, parsing-quality drift, etc. See the `Outlier-friendly?` column in *FLX (Flex Objects) Use Case Template Catalog*.

See also
Outlier detection.

Custom per-entity SPL checks (hybrid trackers)

Component(s)
DSM, DHM, MHM, FLX, FQM, WLK (not VOL — its single incremental collector is the tracker)

Status
Configurable. Foundation caps at 8 active hybrid trackers and Enterprise at 16, in total for the whole deployment; Unlimited is uncapped.

How
Per entity, configure a custom SPL check that contributes `score` / `state` / `anomaly_reason` / `enrichment` to the entity's final state. Useful when the default component logic cannot reach the data, or when you need a domain-specific check on top of the built-in tracker.

See also
Hybrid trackers.

I want to monitor anything I can express as SPL

Use a pre-built FLX template

Component(s)
`FLX`

Status
Out-of-the-box. Sixty-five templates ship in the catalog.

How
The Virtual Tenants FLX wizard guides you through picking a template by category, customising replacements (index name, …), and creating the saved search. The full catalog lives at *FLX (Flex Objects) Use Case Template Catalog*.

Author a custom FLX use case

Component(s)
`FLX`

Status
Configurable. Available in every edition that includes FLX.

How
The wizard accepts arbitrary SPL that must produce rows with `object`, `status`, `status_description`, `metrics` (JSON dict), optionally `outliers_metrics` (JSON dict), and optionally `max_sec_inactive` for inactivity alerting. The SPL becomes a saved search; the rows become FLX entities; thresholds / outliers / policies apply.

See also
FLX — Flex Objects, *Flex Objects - Adapt TrackMe to any monitoring use case (splk-flx)*.

Apply time-aware thresholds to an FLX KPI (business hours vs nights)

Component(s)
`FLX`

Status
Configurable.

How
FLX dynamic thresholds support a **variable** mode with day-of-week × hour-of-day slots. Each slot carries its own threshold value, so a metric can be alerted on at one level during business hours and another at night.

I want to be alerted intelligently (and only when it matters)

This section enumerates the noise-reduction and prioritisation features that turn raw state transitions into actionable signal.

Prioritise entities so the right alerts get the right urgency

Capability
Priority policies (regex, lookup, SPL search).

How
Policies automatically assign priority (`low` / `medium` / `high` / `critical` / `pending`) to entities by name pattern, lookup membership, or live SPL match. Manual overrides always win.

See also
Priority, *Priority*.

Bucket entities into SLA classes (platinum / gold / silver / custom)

Capability
SLA policies + `trackme_settings.conf [sla]`.

How
SLA classes are tenant- or system-defined with their own uptime targets and reporting windows. SLA policies apply them automatically across entities. Maintenance windows and bank holidays are excluded from SLA timers when configured.

See also
Using SLA alerting to build a 2-tier monitoring system, *Maintenance mode*, *Bank holidays*.

Tag entities for routing and grouping

Capability
Tag policies (union semantics — multiple matches accumulate).

How
Regex / lookup / SPL policies assign one or more free-form tags to entities — used for filtering in the UI, downstream search, and alerting rules.

See also
Tags.

Demote false positives — disruption queue (grace period)

Capability
Disruption queue.

How
An entity that goes red is held in `blue` state until it has been continuously red for `disruption_min_time_sec`. Any green tick resets the timer. Configured at tenant default and per-entity override.

See also
Disruption queue, *The entity state machine*.

Don't alert when a cluster is mostly healthy — logical groups

Capability
Logical groups.

How
Cluster N entities (cross-component supported); while green % stays above `object_group_min_green_percent`, struggling members are demoted to `blue` ("protected by

the group"). When too many members are red, all members surface as red so the underlying cluster issue gets escalated.

See also
Logical groups.

Mute outside business hours — monitoring-time policy

Capability
Monitoring-time policy.

How
Per-tenant (and per-entity override) day-of-week × hour-of-day window outside which the entity is forced `orange` (paused). Useful when feeds legitimately stop overnight or on weekends.

Tighter thresholds during business hours — variable delay

Capability
Variable delay (DSM / DHM only).

How
Day-of-week × hour-of-day delay-threshold matrix. Tight thresholds during the business day, looser at night, looser still on weekends. The system falls back to the static threshold for any uncovered slot.

Recompute thresholds from history — adaptive delay

Capability
Adaptive delay (DSM / DHM only).

How
Persistently-red `delay_threshold_breached` entities have their thresholds automatically recomputed from the last 30 days of metrics. 24-hour cool-off prevents oscillation. Opt-in via `allow_adaptive_delay` on the entity. Since 2.3.23, the adaptive path also recomputes individual variable-delay slot thresholds.

Acknowledge known issues — sticky and unsticky ACK

Capability
Acknowledgments.

How
An ACK silences notifications without hiding the entity's real state. **Sticky** ACKs require manual close; **unsticky** ACKs auto-expire after `ack_period` (default 24 h). Reason-change revocation can be enabled so the ACK is removed if the underlying anomaly reason changes.

See also
Acknowledgments.

Auto-acknowledge entities on alert fire

Capability
Auto-ack alert action.

How
Triggered from any saved search firing; acks all returned entities sticky / unsticky with a configurable duration and comment. Useful for maintenance windows known upstream, or to short-circuit known-noisy periods.

Embed the right context in the alert email

Capability

Stateful alerting email + embedded 24h charts + AI status reports.

How

One email per state transition (open / update / close), correctly threaded via `In-Reply-To`. Optional SVG → PNG 24-hour charts embedded inline. Optional LLM-generated 2 – 3 paragraph AI investigation summary embedded in the body, with full entity context (priority, recent ACK history, active Guardian alerts, etc.).

See also

Stateful alerting, *AI Assistant*.

Exclude maintenance windows and bank holidays from SLA timers

Capability

Maintenance KDB + bank holidays.

How

The Maintenance KDB records planned and unplanned windows scoped to global / tenant / component / single-entity. SLA accounting and the monitoring-time policy both honour active windows. Bank holidays are calendars of rules (countries, states and regions from the catalog, plus custom closures); each tenant inherits the default calendars, selects its own or uses none.

See also

Maintenance mode, *Bank holidays*.

I want to investigate and resolve fast when alerts fire

Trigger automated investigations — Smart Status

Capability

Smart Status investigations.

How

Triggered by the `trackme_smart_status` alert action. Generates component-specific investigation SPL (delay / future data / latency issues for DSM/DHM; skip / error / orphan for WLK; outliers and inactivity for FLX). Throttled to **max 1 / 24 h per entity**. Detects abnormal state-flipping patterns (high variance of flip count). Results indexed to `trackme_summary` for review in the Tenant Home modal.

See also

Status flapping, *The entity state machine*.

Ask the AI Assistant in plain English

Capability

AI Assistant chat (10 contexts).

How

Per-page chat widget that knows the page's context — Virtual Tenants, Tenant Home, single entity, REST API Reference, Backup & Restore, Maintenance Mode, Maintenance KDB, Bank Holidays, License Management. Per-entity context includes the entity's current state, policies, recent ACK history, active Guardian alerts. Supports OpenAI, Azure OpenAI, Anthropic, Google Vertex, Mistral, Ollama, and Splunk Hosted. Optional anonymisation of entity / index names via SHA-256 hashing before LLM submission.

See also

AI Assistant.

Hand off context to the next operator — notes and labels

Capability
Notes (Markdown annotations) + **labels** (catalog-backed coloured chips).

How
Notes are free-form Markdown attached to an entity with an audit trail (immutable — delete + recreate to edit). Labels are curated, colour-coded, multi-select chips defined once per tenant. Eight default labels are seeded on tenant creation. Use labels for lifecycle workflow (`under-review` → `in-progress` → `resolved`) and notes for narrative context ("re-deploy at 03:00 on 2026-05-08, root cause INC-12345").

Generate notable events for Splunk ES correlation

Capability
Notable event alert action.

How
Triggered by the `trackme_notable` alert action. One event per firing of the saved search to a notable index (default `trackme_notable`), with drilldown links back into the TrackMe UI.

See also
Notable events.

I want to monitor non-Splunk systems

This is where the **companion add-ons** unlock TrackMe beyond Splunk itself. See *TrackMe Ecosystem — Companion Add-ons* for the full inventory.

Monitor a Cribl Stream deployment or a Cribl Edge fleet end to end

Component(s)
`Cribl Stream` and `Cribl Edge` (dedicated tenant types, 2.4.19+).

Status
Out-of-the-box. A Cribl Stream tenant follows the Leader, the Worker Groups, the Workers and the Sources, Destinations, Pipelines and Routes of one deployment, with traffic machine learning models; a Cribl Edge tenant follows the Fleets of a Cribl Edge Leader and their Edge Nodes, at any fleet size. Both raise findings instead of tracking entities; no companion add-on is needed.

Dependency
A Cribl connection (read-only access to the Leader API) and, for Cribl Stream, the Cribl internal metrics in a Splunk metric index.

See also
Cribl Stream and Cribl Edge, *Cribl Stream*, *Cribl Edge*, *Findings, Finding groups and findings actions*.

Monitor Cribl Stream pipelines, routes, packs, destinations

Component(s)
`FLX`

Status
Pre-built FLX templates (10 templates).

Templates
`cribl_logstream_health_inputs`, `_health_outputs`, `_hosts_cpu_usage`, `_output_destination_pressure`, `_pack_traffic`, `_pipeline`, `_route_traffic`, `_total_traffic_inputs`, `_total_traffic_outputs`, plus `cribl_edge_fleet_metrics`.

Dependency
Cribl-shipped internal metrics indexed in Splunk (any `Splunk_TA_cribl`-equivalent setup).

See also
Cribl Stream monitoring with Flex Objects, *Cribl Stream & Cribl Edge*.

Monitor Cribl Edge node heartbeats via the Cribl API

Component(s)
`FLX`

Status
Pre-built FLX template (with companion-TA dependency).

Templates
`cribl_edge_fleet_api` — the only template that **hard-depends** on `TA-trackme-cribl` (uses `| cribl`).

Monitor Google SecOps (Chronicle) log-feed ingestion

Component(s)
`FLX` or hybrid trackers, custom-built.

Status
Custom SPL via the `| secops` command provided by `TA-trackme-google-secops`.

How
UDM-search-based entities, one per `metadata.log_type`, tracking event count and last-seen. Effectively extends DSM-equivalent monitoring to data living in Chronicle.

Important
This add-on has a **hard licensing dependency on trackme-app** — the `| secops` command refuses to run without a valid TrackMe license.

Monitor Splunk SOAR (Phantom) deployment health

Component(s)
`FLX`

Status
Pre-built FLX templates (9 templates).

Templates
`splk_soar_actions_adhocs_failures`, `_actions_playbooks_failures`, `_assets_health`, `_automation_brokers_manage`, `_concurrent_playbooks`, `_forwarding_splunk`, `_infra_load`, `_infra_memory`, `_services_health`.

Dependency
Splunk App for SOAR integration on the SH; SOAR-to-Splunk forwarding for the data-side templates.

See also
Splunk SOAR, *TrackMe App on SOAR*.

Monitor Linux and Windows host CPU / memory

Component(s)
`FLX`

Status
Pre-built FLX templates (4 templates).

Templates
`ta_nix_cpu_usage` and `ta_nix_mem_usage` (require TA-nix), `ta_windows_cpu_usage` and `ta_windows_mem_usage` (require Splunk_TA_windows). Per-host avg / max / p95.

Aggregate a remote regional TrackMe into a central pane

Component(s)
DSM, DHM, MHM, FLX, FQM, WLK (not VOL)

Status
Configurable. Replica trackers.

How
One-way replication of a source TrackMe's entity set into a local tenant. The local decision maker **does not** re-evaluate replicated entities — the source's state is trusted, by design. Useful for a global SOC with regional TrackMe instances.

I want to operate at scale across multiple tenants / customers

This section enumerates the multi-tenancy, automation, and resilience features that make TrackMe deployable as a managed service or a multi-business-unit platform.

Isolate by tenant — Virtual Tenants

Capability
Virtual Tenants.

How
Each tenant has its own KV collections (suffixed `_tenant_{tenant_id}`), RBAC role triplet (`admin` / `power` / `user`), default thresholds, index scoping, scheduling, and an optional `tenant_account` to target a remote Splunk deployment instead of the local one. Configuration lives in a `vtenant_account` JSON object on the tenant record.

See also
Creating a tenant, *Managing tenants*.

Aggregate across tenants without new infrastructure — Virtual Groups

Capability
Virtual Groups.

How
Read-only UI cards aggregating entities from N tenants (and N components per tenant), optionally filtered by priority. RBAC-controlled visibility. Useful for cross-tenant cards like "all high-priority entities in finance across three regions". Virtual Groups do **not** participate in decision maker / alerting / scoring — they are pure read-only aggregation views.

See also
QUICK START - Cross-tenant domain views with Virtual Groups.

Monitor any number of remote Splunk deployments

Capability
Remote Splunk accounts.

How
Per-account HTTPS URL, bearer token (encrypted, auto-rotated every N days), per-account RBAC role restrictions, multiple URLs per account for HA / DR with automatic failover. Per-tenant `tenant_account` selector points the tenant at local or any configured remote.

Edition gate
Foundation does **not** support remote deployments.

See also
Remote Splunk deployments.

Detect admin misconfiguration automatically — Configuration Guardian

Capability
Configuration Guardian.

How
Continuous background scan of common misconfiguration patterns, each emitting a structured alert with `recommended_actions` an admin (or the AI Assistant) can act on. Eight checks ship in v2.3.23: capability gaps on the tenant owner, missing indexes, remote-account token expiry, remote-account connectivity, AI-provider reachability, stale backups, incomplete backup runs, dead health tracker. Two severity tiers (warning / critical); per-tenant and system-wide scopes; full audit trail in `index=trackme_audit sourcetype=trackme:audit:guardian`.

Bulk-edit entity priority, policy, lag, tags

Capability
Mass operations UI + REST.

How
Multi-select in the Tenant Home tables, then bulk-apply priority / SLA / tags / monitoring-time / ACK / disable / re-enable. Equivalent REST endpoints under `/trackme/v2/<component>/...` and `/trackme/v2/ack/*`.

See also
Performing mass operations in TrackMe.

Replay configuration changes resiliently

Capability
TA-trackme-conf-manager (companion add-on).

How
A KV-backed transactional layer that accepts a TrackMe REST transaction, persists it, and replays it on a 15-minute schedule until it succeeds. Useful for SOAR playbooks, infrastructure-as-code pipelines, and any orchestrator that must guarantee a configuration change eventually lands. See *TrackMe Ecosystem — Companion Add-ons* for details.

Back up and restore tenants independently — multi-archive 3.0.0

Capability
Multi-archive backup & restore (release 2.3.22+).

How
A backup run produces N+1 independent archives — one per enabled tenant + one global archive — each individually validated (`zstd -t` + SHA-256) and individually restorable. A failure in one archive does not block the others, by design. Three restore modes: single archive, whole run, or legacy flat (pre-2.3.22 monolithic). SHC-aware (per-archive operations delegate to the owning peer). Missing-tenant safety guard automatically recreates a tenant record from the archive's own `vtenant_account` JSON if restoring on an install where the tenant is absent.

See also
Backing up and Restoring TrackMe.

I want to integrate with my downstream incident-response stack

Push to Splunk ES — notable events

Capability
Notable event alert action.

How

`trackme_notable` writes events to a notable index (default `trackme_notable`) consumable by Splunk Enterprise Security's notable event framework.

See also

Notable events.

Push to Splunk SOAR (Phantom) — containers and active commands

Capability

Splunk SOAR push (built-in) + **active commands** on stateful alerts.

How

TrackMe stateful alerts can run **active commands** on open / update / close — any generating or streaming search command, parameterised with the alert's payload. The most common use is `| trackmesplksoar` to push containers / events into SOAR. `trackme-app-on-soar` (a Phantom-side connector) provides the inverse integration: SOAR playbooks calling TrackMe REST to ACK entities, enable / disable maintenance, retrain ML models, manage logical groups, etc.

Push to Cortex XSOAR (Palo Alto) — bidirectional incident lifecycle

Capability

TA-trackme-xsoar (companion add-on).

How

Active-command integration on TrackMe stateful alerts: `commands_opened` creates an XSOAR incident, `commands_updated` patches it with new state, `commands_closed` closes it. A built-in resilient retry store (KV-backed, 15-minute replay) guarantees incident parity even through network blips or XSOAR outages. See *TrackMe Ecosystem — Companion Add-ons*.

See also

Palo Alto XSOAR Integration.

Trigger any TrackMe REST endpoint from any saved search firing

Capability

Free-style REST call alert action.

How

`trackme_free_style_rest_call` invokes any `/trackme/v2/*` endpoint (GET / POST / DELETE) when the saved search fires. Useful for custom orchestration: bulk-ACK from a correlation search, force-disable entities from an upstream change-management signal, trigger a smart status investigation from an external context.

I want to extend or customise TrackMe

Add a custom check to an existing entity (any component)

Capability

Hybrid trackers (per-component).

How

A custom SPL check that runs alongside the default component logic and contributes `score` / `state` / `anomaly_reason` / `enrichment` to the entity's final state.

Edition gate

Foundation rejects; Enterprise caps at 8 per tenant; Unlimited is uncapped.

Replicate a tracked entity set into another TrackMe instance

Capability
Replica trackers.

How
One-way replication, source state trusted (no local re-evaluation). Useful when a central TrackMe must present entities owned by remote regional TrackMes.

Persist arbitrary TrackMe configuration as code

Capability
TA-trackme-conf-manager + `/trackme/v2/*` REST API.

How
Define entities, policies, thresholds, ACKs, maintenance windows via REST; wrap them in conf-manager transactions for guaranteed delivery. Suitable for Ansible / Terraform / GitOps integration.

Orchestrate TrackMe from Splunk SOAR playbooks

Capability
trackme-app-on-soar (a Splunk SOAR / Phantom connector, not a Splunk TA).

How
Exposes TrackMe REST endpoints as SOAR actions: `test connectivity`, `ack_get` / `ack_manage`, `maintenance_*`, `tenants_ops_status`, `remote_accounts_check_connectivity`, `ml_outliers_*`, `component_get_entity` / `component_manage_entity`, `smart_status`, `logical_group_*`. Asset configured with a Splunk bearer token. Powers playbook-driven runbooks (e.g. ACK on upstream CMDB change, disable maintenance after deploy completes).

See also
TrackMe Ecosystem — Companion Add-ons, *TrackMe App on SOAR*.

I want to know who is doing what on my Splunk

User Activity Monitoring (UAM, introduced in TrackMe 2.4.18, generally available since 2.4.19) answers the people-and-automation side of the platform as a dedicated tenant type — no entities, findings instead. It reads the audit trail and the introspection of a deployment through one or more search head tiers. See *User Activity Monitoring* and the white paper *Who is doing what on your Splunk? User Activity Monitoring (UAM)*.

Find the scheduled searches owned by people, dispatched all time or in real time

Component(s)
`UAM` (tenant type)

Status
Out-of-the-box. Create a UAM tenant; the inventory policies `personal_schedule`, `all_time_schedule` and `realtime_schedule` are on by default.

How
The inventory trackers read the saved searches, users and roles of every tier and record every change; the owner's class (human, service, system) decides which policies apply.

Tell people from service accounts, and catch the account that ran an all-time search

Component(s)
`UAM` (tenant type)

Status
Out-of-the-box. Human accounts are detected from Splunk Web activity and browser logins,

service accounts from sustained automation or from a naming rule; `all_time_search` is on by default at severity high.

How

The activity tracker reads the audit trail from a durable checkpoint and indexes one compact record per search and window; the Users tab gives every account its class, findings, activity and resource usage.

Find the accounts and searches that consume the search capacity

Component(s)

`UAM` (tenant type)

Status

Out-of-the-box. `resource_consumption` and `resource_intensity` derive their thresholds from the measured search capacity of the environment.

How

Resource usage per search is integrated from the introspection of the search heads and the indexers; the account deep-dive lists the most expensive searches.

What if my goal isn't covered above?

The pre-built FLX template library is one of TrackMe's broadest surface areas but it is not exhaustive. The *FLX (Flex Objects) Use Case Template Catalog* page ends with a **"Coverage gaps not addressed out of the box"** section listing the recurring asks we know about (auth-failure spikes, scheduler skip-rate per user / app beyond what WLK does, app-version drift, etc.).

When you have a goal not addressed by a template or a built-in feature, your three options, in increasing order of effort, are:

1. **Author a custom FLX use case** through the wizard — see *FLX — Flex Objects*.
2. **Add a hybrid tracker** to an existing entity — see *Hybrid trackers*.
3. **Build the data path** via a companion TA (Cribl, Google SecOps, lookup monitor) and feed the data into FLX. See *TrackMe Ecosystem — Companion Add-ons*.

If a use case feels generally applicable across customers, contributing a JSON file back to `package/lib/flx_library/` makes it available to every TrackMe deployment.

TrackMe Capability × Component Matrix

What this page is

A flat, exhaustive map of every cross-cutting TrackMe capability against the seven monitoring components (`DSM`, `DHM`, `MHM`, `VOL`, `FLX`, `FQM`, `WLK`). For each `(capability × component)` intersection it records:

- whether the capability **applies** to that component,
- the **use case unlocked** by turning it on (one line each),
- the **configuration touchpoint** (UI page, system setting, KV collection, `vtenant_account` field).

Use this page when you know **which components** a tenant has enabled and want to audit which **cross-cutting features** are wired in for those components. It pairs naturally with *TrackMe Use Cases — Organised by User Intent* (which asks the inverse question: given a goal, which component delivers?).

Legend

In every applicability table:

- `Y` — fully supported on that component.
- `P` — partial support; the cell footnote explains the caveat.
- – — not applicable on that component (see *What does not apply, and why* at the bottom of this page).

The seven components — recap

For the full description see *TrackMe Use Cases Referential — Overview*. Short form:

Code	Full name	Tracks
`DSM`	Data Source Monitoring	`(index, sourcetype)` feeds. Includes the `lookups` search mode.
`DHM`	Data Host Monitoring	Host × sourcetype activity. 2.3.23+ supports `breakby_extra_fields`.
`MHM`	Metric Host Monitoring	Hosts × metric category, `mstats`-based.
`VOL`	Volume Outliers	Licensed volume per index from the license usage logs (5-minute buckets, incremental). ML outliers (drops / spikes, direction per index), inactivity, daily volume, trend, month projection. 2.4.18+; see *VOL — Volume Outliers*.
`FLX`	Flex Objects	Custom SPL turned into tracked entities. 65 pre-built templates.
`FQM`	Field Quality Monitoring	Field parsing quality, CIM compliance.
`WLK`	WorkLoad Knowledge	Scheduled-search execution health, versioning, orphan, app enablement.

The seven **dedicated tenant types** sit outside this matrix on purpose: **User Activity Monitoring** (`UAM`, 2.4.18+, generally available since 2.4.19), **Cribl Stream** and **Cribl Edge** (2.4.19+), **Splunk HEC** (2.4.21+), **Splunk S2S** and **Splunk UF Fleet** (2.4.22+) and **Splunk Forwarder Collectors** (2.4.24+). They have no entities, so the state, scoring and alerting capabilities below do not apply to them. Each raises **findings** from a policy catalogue instead, with its own investigation workflow and notifications. See *Findings, Finding groups and findings actions*, *User Activity Monitoring*, *Cribl Stream and Cribl Edge*, *Splunk HEC*, *Splunk S2S*, *Splunk UF Fleet* and *Splunk Forwarder Collectors*.

State, scoring, alerting

These capabilities are largely uniform across components — TrackMe was designed so the alerting and scoring experience is the same regardless of what is being monitored.

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Hybrid impact scoring (`total_score`)	Y	Y	Y	Y	Y	Y	Y	Same additive formula across all seven; per-component weight names.
4-state model (green / orange / red / blue)	Y	Y	Y	Y	Y	Y	Y	Same machine across all seven components.
Stateful alerting (open / update / close)	Y	Y	Y	Y	Y	Y	Y	Common `trackme_stateful_alert` action.
Acknowledgments (sticky / unsticky)	Y	Y	Y	Y	Y	Y	Y	Common ACK collection.
Auto-ACK alert action	Y	Y	Y	Y	Y	Y	Y	Triggered on any saved-search firing.
Notable events alert action	Y	Y	Y	Y	Y	Y	Y	One event per fire; not de-duplicated by TrackMe.
Smart Status investigations	Y	Y	P	P	Y	P	Y	DSM / DHM / FLX / WLK have explicit per-component investigation templates; MHM, VOL and FQM inherit the generic outliers template.
Disruption queue (grace period)	Y	Y	Y	Y	Y	Y	Y	Identical code path across components.
Email delivery (SMTP)	Y	Y	Y	Y	Y	Y	Y	Same SMTP account; same stateful helper.
Embedded 24h charts in alert emails	Y	Y	Y	Y	Y	Y	Y	`generate_charts=1` on the alert action. VOL emails carry the rolling volume charts, the outliers chart and a daily licensed-volume bar chart.
AI status reports in alert emails	Y	Y	Y	Y	Y	Y	Y	`ai_status_report=1` on the alert action.
Free-style REST call alert action	Y	Y	Y	Y	Y	Y	Y	Targets any `/trackme/v2/*` endpoint.

Use case unlocked (one line per capability):

- **Hybrid impact scoring —** Each anomaly type contributes additively, so one critical breach trips red immediately while several minor ones accumulate into orange before tripping critical.
- **4-state model —** One uniform "what colour is my donut" semantic across every component, every UI surface, every downstream consumer.
- **Stateful alerting —** One email per real state transition, not one email per evaluation cycle; opens / updates / closes thread together.
- **Acknowledgments —** Silence known issues without losing the underlying state signal. Sticky vs unsticky lets you pick auto-expiry.
- **Auto-ACK —** Pre-emptively suppress notifications for entities matching a pattern ("ACK for 2 h on any red in this group").
- **Notable events —** Feed TrackMe state into Splunk ES / correlation searches / incident platforms via a notable-index event.
- **Smart Status —** When an entity goes red, an investigation search runs once / 24 h and surfaces likely root-cause hypotheses.

- **Disruption queue** — Fewer false-positive pages from 30-second blips; alerts only fire after the configured grace duration of continuous red.
- **Email delivery** — Notifications land in inboxes correctly threaded with embedded charts, no external image fetches.
- **AI status reports** — Recipient sees a 2 – 3 paragraph contextual investigation directly in the email body, not just a link.
- **Free-style REST call alert action** — Wire any TrackMe REST mutation (entity CRUD, mode flip, ACK bulk-op) into a saved-search firing without writing a new alert action.

Classification and protection (policies)

This is the largest cluster of cross-cutting features. Several of them are DSM / DHM specific because they relate to delay / latency thresholds.

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Priority policies (regex / lookup / SPL)	Y	Y	Y	Y	Y	Y	Y	Per-component policy collection.
Tag policies (regex / lookup / SPL)	Y	Y	Y	Y	Y	Y	Y	Multiple matches union.
SLA policies (regex / lookup / SPL)	Y	Y	Y	Y	Y	Y	Y	"Last wins" when multiple policies match.
Labels (catalog-backed coloured chips)	Y	Y	Y	Y	Y	Y	Y	Single catalog per tenant.
Notes (Markdown annotations)	Y	Y	Y	Y	Y	Y	Y	One tenant collection.
Variable delay (time-slot thresholds)	Y	Y	—	—	—	—	—	DSM / DHM only.
Lagging classes (named delay / lag rule sets)	Y	Y	—	—	—	—	—	DSM / DHM only.
Adaptive delay (automated threshold recompute)	Y	Y	—	—	—	—	—	DSM / DHM only. 2.3.23+ also recomputes variable-delay slots.
Variable dynamic thresholds (time-slot KPI thresholds)	—	—	—	—	Y	—	—	FLX only. Different mechanism from DSM/DHM variable delay.
WLK tenant-default threshold row	—	—	—	—	—	—	Y	WLK-only fleet-wide fallback (`object_id == "default"`).
Logical groups (blue protection)	Y	Y	Y	Y	Y	Y	Y	Cross-component membership supported.
Blocklist / Allowlist (regex)	Y	Y	Y	Y	Y	Y	Y	Per-component allowlist collection holding both rule types.
Monitoring-time policy	Y	Y	Y	Y	Y	Y	Y	Tenant default + per-entity override.

Use case unlocked:

- **Priority policies** — Don't manually set priority on 5,000 entities; a regex on the index name covers them.
- **Tag policies** — Routing labels (`team:auth`, `env:prod`) attach automatically and survive entity recreation.
- **SLA policies** — Cleanly bucket entities into platinum / gold / silver uptime tiers per regex or lookup.
- **Labels** — Visual lifecycle workflow (`under-review` → `in-progress` → `resolved`) with curated colours.

- **Notes —** Persist investigation context for the next operator without polluting structured fields.
- **Variable delay —** Don't alert on the expected nightly-batch window; business-hours threshold stays tight.
- **Lagging classes —** Manage thresholds once for 40 high-frequency feeds by editing one class instead of 40 entities.
- **Adaptive delay —** Persistently-red `delay_threshold_breached` entities get their thresholds auto-recomputed from history.
- **Variable dynamic thresholds (FLX) —** Tighter KPI thresholds during business hours, looser overnight, on the same metric.
- **WLK tenant-default threshold row —** Manage skip% / error count / delay thresholds for the whole tenant from one record; per-entity overrides as exceptions.
- **Logical groups —** Don't alert when only 1 of 100 hosts is down; the cluster's green % protects struggling members.
- **Blocklist / Allowlist —** Scope discovery to `index=prod_*` only; never track `sourcetype=.*:debug`.
- **Monitoring-time policy —** Auto-pause monitoring outside business hours; entities go orange (silent) instead of false-positive red.

Discovery, sampling, and quality

These are mostly component-specific signals — DSM-only sampling features, FQM-only CIM templates, WLK-only signals, and DHM-only extras.

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Data sampling (regex format models)	Y	—	—	—	—	—	—	DSM-only — needs raw events.
Elastic sources (external collectors)	Y	—	—	—	—	—	—	DSM-only.
Lookups monitoring (`search_mode=lookups`)	Y	—	—	—	—	—	—	DSM-only; requires `TA-trackme-lookupmonitor`.
CIM-aware field-quality templates	—	—	—	—	—	Y	—	FQM-only.
Sourcetype-explosion safeguard	Y	—	—	—	—	—	—	DSM-only entity-creation hot path.
WLK orphan / versioning / app enablement	—	—	—	—	—	—	Y	WLK-only signals.
DHM `breakby_extra_fields`	—	Y	—	—	—	—	—	DHM hybrid trackers, 2.3.23+, Custom break-by mode only.
History backfill at creation	—	—	—	Y	—	—	—	VOL-only: the license-usage history (default 30 days, 0–400) is replayed automatically so outliers models and the cost trend are usable as soon as the replay completes.
Inactivity policy (static or day-of-week / hour-of-day)	—	—	—	Y	—	—	—	VOL-only: tenant-level policy, overridable per index; an index that stops consuming license turns red.
Outlier detection direction per entity (drops / spikes / both)	—	—	—	Y	—	—	—	VOL-only: chosen per index.
30-day trend, month projection, Global license usage analytics	—	—	—	Y	—	—	—	VOL-only: per-index trend and projection; the **Global license usage (VOL)** Tenant Home tab covers key figures, insights, top consumers, movers, pools, named scopes, working days and holidays, days over quota, opt-in costs and a monthly chargeback per cost centre.

ML and advanced trackers

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Native ML outlier detection	Y	P	P	Y	Y	P	P	**Recommended** for FLX (natural fit); **the primary state signal for VOL** (enabled by default, default model on the rolling 24 h volume, own `vol_mloutliers_*` scope) and the recommended route for volume anomalies in all editions. Optional on DSM (off by default) and selectively useful on DHM. Not recommended for MHM (sparse / categorical), FQM (slow-moving), WLK (discrete metrics). Defaults at tenant creation: on for FLX and VOL; off for DSM, DHM, WLK and FQM.
Hybrid trackers (custom per-entity SPL check)	Y	Y	Y	—	Y	Y	Y	One registry per component. Enterprise caps at 8 per tenant; Foundation rejects; Unlimited uncapped. Not applicable to VOL (its single collector is the tracker).
Replica trackers (one-way entity replication)	Y	Y	Y	—	Y	Y	Y	Source and target component must match. Local decision maker does **not** re-evaluate replicated entities.

Use case unlocked:

- **Native ML outlier detection —** Get alerted on volume / KPI anomalies without writing thresholds; the model learns the normal pattern with optional weekly seasonality.
- **Hybrid trackers —** Add a custom SPL check that contributes to an entity's score; useful when the default tracker cannot reach the data or you need a domain-specific check on top of the built-in tracker.
- **Replica trackers —** Single pane of glass in a central TrackMe across regional TrackMes without running searches on each for every UI refresh.

Integrations

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Remote Splunk accounts (`splunkremotesearch`)	Y	Y	Y	Y	Y	Y	Y	Per-tenant `tenant_account` selects local or a configured remote. Foundation edition does not support remote deployments.
AI Assistant chat (per-entity context)	Y	Y	Y	Y	Y	Y	Y	Entity-level context types exist for all seven components.
AI ML Advisor (interactive + automated batch)	Y	Y	—	Y	Y	Y	Y	Inspects / remediates native ML outlier models; the automated batch covers DSM, DHM, VOL, FLX, FQM, WLK.
AI Components Advisor (Feed Lifecycle / FLX Threshold / FQM / Component Health)	Y	Y	Y	—	Y	Y	Y	No Components Advisor covers VOL; the ML Advisor and the Concierge do.
Topology Studio views (entity nodes)	Y	Y	Y	Y	Y	Y	Y	Any entity from any tenant and component can be placed on a view.
Tenant Home topology graph (link / network)	Y	Y	Y	—	Y	Y	Y	VOL has one entity per index — no hierarchy to draw.
Splunk SOAR push (containers + enrichment)	Y	Y	Y	Y	Y	Y	Y	Component-agnostic; works on any rows producing entities.

Use case unlocked:

- **Remote Splunk accounts** — Monitor any number of remote Splunk deployments from a central TrackMe with HA URLs and automatic token rotation.
- **AI Assistant chat** — Ask "why is this entity red?" on the Tenant Home page and get a context-aware LLM answer that knows the entity's state, policies, recent ACK history, and active Guardian alerts.
- **Splunk SOAR push** — Auto-create SOAR containers from TrackMe alerts for runbook-driven remediation; broker-failover safe.

Multi-tenancy and operations

Cross-cutting capabilities that sit above components. They apply uniformly to every component a tenant has enabled.

Capability	DSM	DHM	MHM	VOL	FLX	FQM	WLK	Notes
Virtual Tenants (isolation, RBAC, indexes)	Y	Y	Y	Y	Y	Y	Y	Components are enabled per tenant via `tenant_components`.
Virtual Groups (read-only cross-tenant cards)	Y	Y	Y	Y	Y	Y	Y	`tenants_scope` carries per-tenant component lists.
RBAC 3-tier (user / power / admin)	Y	Y	Y	Y	Y	Y	Y	Same model regardless of component.
Maintenance mode (runtime alert silencing)	Y	Y	Y	Y	Y	Y	Y	Global or per-tenant; suppresses alerting-path actions.
Maintenance KDB (windows, SLA exclusion)	Y	Y	Y	Y	Y	Y	Y	Scope cascade global / tenant / component / entity.
Bank holidays (SLA exclusion)	Y	Y	Y	Y	Y	Y	Y	Per tenant: inherit the default calendars, selected calendars or none.
Backup & restore (multi-archive 3.0.0)	Y	Y	Y	Y	Y	Y	Y	Tenant archive carries every component's tenant collections.
Configuration Guardian	Y	Y	Y	Y	Y	Y	Y	All registered checks are system- or tenant-scoped, not component-specific.
Schema migrations (per-tenant version)	Y	Y	Y	Y	Y	Y	Y	Each migration may touch any component's collections.
License enforcement (read-only, edition gates)	Y	Y	Y	Y	Y	Y	Y	Foundation gates FLX, FQM and WLK creation (VOL is included in Foundation); Enterprise caps hybrid trackers per tenant.

Configuration touchpoints

Where each capability is configured through TrackMe's native interfaces. System-wide settings and integration accounts are available from the **Configuration** menu; AI providers are under **Artificial Intelligence**. `vtenant_account` is the JSON blob inside the tenant record.

Capability	Touchpoint
Hybrid impact scoring weights	Tenant-level: `vtenant_account.impact_score_*`; entity-level `impact_score_weights` JSON.
Stateful alerting	Alert-action parameters on the saved search (Alert Editor); state in `kv_trackme_common_stateful_alerting_tenant_{tid}`.
Acknowledgments	Tenant Home entity → Ack modal; REST `/trackme/v2/ack/*`.
Auto-ACK / Notable / Free-style REST call / Smart Status	Alert-action parameters on the firing saved search.
Email delivery	Configuration → Manage Email delivery accounts → `trackme_emails.conf`.
AI status reports in alert emails	Alert-action `ai_status_report=1` + `ai_provider_name`; provider configured under Artificial Intelligence → Manage AI Provider accounts.
Disruption queue	System default in `trackme_settings.conf [splk_general]`; per-tenant `vtenant_account.default_disruption_min_time_sec`; per-entity `disruption_min_time_sec`.
Priority / Tag / SLA policies	Tenant Home per-component "Policies" modal; REST `/trackme/v2/splk_<priority|tag|sla>_policies/*`.

continues on next page

Table 1 – continued from previous page

Capability	Touchpoint
SLA classes	Configuration → System settings → SLA (`trackme_settings.conf [sla]`).
Labels (catalog + assignments)	Tenant Home "Manage Labels" modal; REST `/trackme/v2/labels/*`.
Notes	Entity overview "Notes" panel; REST `/trackme/v2/notes/*`.
Variable delay (DSM / DHM)	DSM / DHM entity → "Variable delay threshold" modal; REST `/trackme/v2/splk_variable_delay/*`.
Lagging classes (DSM / DHM)	Tenant Home DSM / DHM "Lagging classes" modal; REST `/trackme/v2/splk_lagging_classes/*`.
Adaptive delay	Per-entity `allow_adaptive_delay`; tracker `trackme_<dsm|dhm>_adaptive_delay_tracker_tenant_<tid>`.
FLX / FQM / WLK dynamic thresholds	Tenant Home per-component "Thresholds" modal; REST `/trackme/v2/splk_<flx|fqm|wlk>/*`.
Logical groups	Tenant Home per-component "Logical groups" modal; REST `/trackme/v2/splk_logical_groups/*`.
Blocklist / Allowlist	Per-component wizard / feed configuration; REST `/trackme/v2/splk_blocklist/*`.
Monitoring-time policy	`vtenant_account.monitoring_time_policy`; per-entity override.
Data sampling (DSM)	DSM tab → "Data sampling" modal; REST `/trackme/v2/splk_data_sampling/*`.
Elastic sources (DSM)	DSM "Elastic sources" wizard; REST `/trackme/v2/splk_elastic_sources/*`.
Lookups monitoring (DSM)	Hybrid Trackers create wizard, Step 1 `search_mode=lookups`; requires companion TA `TA-trackme-lookupmonitor` installed.
WLK orphan / versioning / app enablement	WLK wizard + per-app `kv_trackme_wlk_apps_enablement_tenant_{tid}`.
Sourcetype-explosion safeguard	`trackme_settings.conf [splk_general] sourcetype_max_cap_per_index`.
Native ML outlier detection	Tenant Home per-component "Outliers rules" modal; tenant-level `vtenant_account.mloutliers_<component>`; defaults in `trackme_settings.conf [splk_outliers_detection]`.
VOL detection direction, inactivity policy, backfill	Tenant Home VOL tab (per-index direction and inactivity override, backfill control) and the Global license usage (VOL) tab; tenant inactivity policy and `vol_mloutliers_priority_filter` / `vol_mloutliers_filter_expression` in `vtenant_account`; REST `/trackme/v2/splk_vol/*`.
Hybrid trackers	Tenant Home per-component "Hybrid Trackers" modal / wizard; REST `/trackme/v2/splk_hybrid_trackers/*`; license-edition gated.
Replica trackers	Virtual Tenants "Replica Wizard"; REST `/trackme/v2/splk_replica_trackers/*`; executor `trackme_replica_executor` saved search.
Remote Splunk accounts	Configuration → Manage Remote deployment accounts → `trackme_account.conf`; Audit & troubleshoot → Audit - Splunk Remote Accounts Overview.
AI Assistant (provider config)	Artificial Intelligence → Manage AI Provider accounts → `trackme_ai_provider.conf`; master switch `trackme_settings.conf [trackme_general] enable_ai_assistant`.
AI Assistant (per-tenant)	`vtenant_account.ai_provider_name`, `ai_anonymize_entity_names`.
Splunk SOAR	`vtenant_account.soar_*`; REST `/trackme/v2/splk_soar/*`; commands `trackmesplksoar` and `trackmesplksoarlookup`.
Virtual Tenants	Virtual Tenants → Configure tenant (per-tenant 3-dot menu); REST `/trackme/v2/vtenants/admin/*`.
Virtual Groups	Virtual Tenants page Create Virtual Group wizard; REST `/trackme/v2/virtual_groups/*`.

continues on next page

Table 1 – continued from previous page

Capability	Touchpoint
RBAC (per-tenant roles)	Splunk roles (`trackme_<admin|power|user>_<tid>`) + tenant record's `rbac_<admin|power|user>_roles` fields.
Maintenance mode	Maintenance Mode page; REST `/trackme/v2/maintenance/*`; tracker `trackme_maintenance_mode_tracker`.
Maintenance KDB	Maintenance KDB page; REST `/trackme/v2/maintenance_kdb/*`.
Bank holidays	Manage Bank Holidays page; REST `/trackme/v2/bank_holidays/*`; per-tenant scope `vtenant_account.bank_holidays_scope` and `bank_holidays_calendars`.
Backup & restore	Backup & Restore page; REST `/trackme/v2/backup_and_restore/*`; scheduler `trackme_backup_scheduler`.
Configuration Guardian	Surfaced as toasts on Virtual Tenants page; REST `/trackme/v2/configuration/admin/guardian_alerts`, `dismiss_guardian_alert`, `run_guardian_checks`.
Schema migrations	Per-tenant `schema_version` field; runs at tenant create / upgrade / load.
License	License page; REST `/trackme/v2/licensing/*`; watermark in `trackme_settings.conf` (HMAC-SHA256).

What does not apply, and why

DSM-only capabilities

- **Data sampling —** Inspects raw events; needs `index + sourcetype + _raw`. DHM works off ingest metadata (no raw text), MHM is metric data, FLX / FQM / WLK build their own metric shape.
- **Elastic sources —** A DSM entity is the right granularity for "is this external collector producing data?".
- **Lookups monitoring —** Lookup files / KVstore collections are configuration artefacts, not data feeds; DSM is the closest model. Operates with neutralised latency (no `_indextime` exists for lookups) and a single delay-threshold alerting hook.
- **Sourcetype-explosion safeguard —** Bulk DSM creation discovers an `(index, sourcetype)` matrix that can blow up if a pipeline mis-routes events. Other components do not pivot on this discovery axis.

DSM + DHM only (not MHM / VOL / FLX / FQM / WLK)

- **Variable delay —** Day-of-week × hour-of-day thresholds make sense for data feeds and per-host streams that have legitimate cadence variation. MHM uses static thresholds.
- **Lagging classes —** Same rationale; only meaningful for delay / latency-based components.
- **Adaptive delay —** Recomputes `data_max_delay_allowed` (and variable-delay slot thresholds since 2.3.23) from historical metrics. Only meaningful for components with a single `delay` axis to recompute.

WLK-only signals

- **Orphan, versioning, app enablement —** They describe properties of a Splunk scheduled search and its hosting app. Not applicable to data feeds, hosts, custom SPL trackers, or field-quality entities.
- **Tenant-default threshold row (``object_id == "default"``) —** WLK-specific fallback for "manage thresholds once for the whole tenant, override per entity as exceptions".

FQM-only

- **CIM-aware templates —** FQM is purpose-built for "is this CIM data-model field being parsed correctly?". The other components do not evaluate per-field coverage / success metrics.

DHM-only (2.3.23+)

- ``**breakby_extra_fields**`` — Extra metadata dimensions on top of `(index, sourcetype)` for DHM hybrid trackers in Custom break-by mode. Specific to how DHM groups per-host activity.

FLX-only

- **Variable dynamic thresholds (time-slot KPI thresholds)** — FLX supports day-of-week × hour-of-day slots on its dynamic threshold evaluator. This is a different mechanism from DSM / DHM variable delay — variable delay applies to feed delay thresholds, variable dynamic thresholds apply to any FLX KPI.

VOL-only

- **History backfill, inactivity policy, detection direction, month projection, Global license usage analytics** — VOL tracks one signal (licensed volume per index, from the license usage logs) whose state is the ML outlier verdict plus inactivity. There are no delay / latency thresholds, no dynamic thresholds and no raw events to sample, so none of the feed- or threshold-specific capabilities apply; the VOL-specific rows above exist because that signal needs a history to learn from (backfill), a silence rule (inactivity policy) and a cost reading (trend, projection, global analytics).
- **No hybrid or replica trackers** — the VOL tracker is a single incremental collector per tenant; there is no per-entity SPL to customise and no replica path.
- **No Tenant Home topology graph** — one entity per index leaves no hierarchy to draw; VOL entities are placed in Topology Studio views like any other component.

Outlier-detection recommendation

FLX is recommended (numeric KPIs are natural density-function candidates); on VOL outliers are the point of the component and are on by default — VOL, available in all editions, is the recommended route for volume anomalies, keeping outliers detection separate from feed monitoring. DSM is optional and off by default at tenant creation. DHM is selective ("per-host volume can be noisy"). MHM, FQM, WLK are not recommended:

- **MHM** — Metric streams are too sparse or too categorical for density modelling.
- **FQM** — Field-quality metrics are slow-moving and regime-dependent; thresholds are the right tool.
- **WLK** — Scheduler metrics are discrete (skip count, error count); again, thresholds are better.

Smart Status template coverage

Smart Status investigation templates are documented for DSM, DHM, FLX, WLK anomalies (delay, future data, latency, hosts dcount, inactivity, threshold, skip, error, orphan). MHM and FQM fall through to a generic outliers template — marked `P` in the matrix.

Replica trackers and the decision maker

Replica trackers replicate an entity set but the **local decision maker does not re-evaluate** them — it trusts the source's state. So while the matrix shows `Y` across the six components that support them (replica trackers work on any component except VOL), the surrounding capabilities (logical groups, disruption queue, policies) only apply to **non-replicated** entities in that component; on replicated entities they are inert by design.

License-edition gating

- Foundation edition supports DSM / DHM / MHM / VOL only; FLX, FQM and WLK rows in the matrix are still `Y` (the capability exists), but Foundation tenants cannot create FLX / FQM / WLK entities.
- Enterprise caps hybrid trackers at 8 per tenant; Foundation rejects hybrid-tracker creation; Unlimited is uncapped.
- Read-only mode (license expired) disables creation / editing across all seven components without removing the capabilities themselves.

FLX (Flex Objects) Use Case Template Catalog

What this page is

The full catalog of every pre-built FLX use case template that ships with TrackMe — 65 templates in v2.3.23, grouped by category, with metrics, schedule, source data, dependencies, and ML-outlier flag per entry.

The page ends with a **gap-analysis checklist** organised by domain (license, indexer cluster, ingestion, SH workload, good practices, data quality, Cribl, SOAR, host OS, …) — work through it one question at a time when auditing an existing TrackMe deployment.

Catalog version
TrackMe v2.3.23 (65 templates).

Where the source lives
`package/lib/flx_library/<uc_ref>.json` inside the TrackMe Splunk app — one JSON file per template.

Introduction

TrackMe's **Flex Objects (FLX)** component turns any Splunk SPL search into a tracked entity with custom KPIs, dynamic thresholds and stateful alerting. Out of the box, TrackMe ships **65 pre-built FLX templates** that cover the most common monitoring needs across Splunk Enterprise, Splunk Cloud, Splunk SOAR, Cribl Stream / Cribl Edge, and the canonical Splunk technology add-ons (TA-nix, Splunk_TA_windows).

Every template declares:

- `uc_ref` — template identifier (matches the filename without `.json`).
- `uc_vendor` — broad provenance: `Splunk` or `Cribl`.
- `uc_category` — finer-grained grouping used in the wizard.
- `uc_description` — one-sentence intent shown before creation.
- `uc_earliest` / `uc_latest` — relative time bounds of the SPL.
- `uc_cron` — schedule for the generated saved-search.
- `uc_metrics` — flat list of KPI names produced by the search.
- `uc_implementation_comments` — pre-requisites, caveats, documentation links.
- `uc_search` — the SPL backing the use case (not reproduced here — see source).

Of the 65 templates:

- **2 vendors** are represented — Splunk (54 templates) and Cribl (11).
- **11 categories** group them — see the per-category tables below.
- **20 templates** opt into native ML outlier detection by emitting an `outliers_metrics` block in their SPL (flagged `yes` in the *Outlier-friendly?* column).
- Schedules range from every 5 min (the majority) through half-hourly, hourly and twice-daily, down to a handful of once-a-day batch searches anchored at quiet hours (license rollups around 03:00 – 04:00 UTC, field-quality at 06:00 UTC).

How to read the schedule column

- `*/5 * * * *` → every 5 minutes
- `*/15 * * * *` → every 15 minutes
- `*/30 * * * *` → every 30 minutes
- `0 * * * *` → hourly on the hour

- `0 */12 * * *` → every 12 hours (00:00 and 12:00 UTC)
- `5 3 * * *` → daily at 03:05 UTC
- `10 4 * * *` → daily at 04:10 UTC
- `15 4 * * *` → daily at 04:15 UTC
- `0 6 * * *` → daily at 06:00 UTC

All schedules run against the **tenant's local time-zone interpretation** once dispatched by the Splunk scheduler.

How to read the source-data column

- `_internal` — Splunk's own indexed logs (`splunkd.log`, `license_usage.log`, `metrics.log`, …).
- `_introspection` — host resource usage, KV-store stats, scheduler diagnostics.
- `_dsphonehome` — deployment-server phone-home events.
- `cribl_metrics` — Cribl-emitted internal metrics indexed on the Splunk side.
- `_cmc_summary` / `summary` — Splunk Cloud monitoring summary events (CMC).
- `mgmt API (| rest)` — Splunk REST endpoint executed in-search.
- `Cribl API (| cribl)` — out-of-band call via `TA-trackme-cribl`.
- `SOAR API (| trackmesplksoar)` — out-of-band call to a SOAR controller.
- `Custom index` — anything else (`phantom_action_run`, `lastchanceindex`, `trackme:fields_quality`).
- `tstats` / `mstats` — tsidx / metric store; the actual index is parameter-driven.

How to read the dependency column

- `none` — pure Splunk; runs against indexers / SH out of the box.
- `TA-trackme-cribl` — required for any `| cribl …` command.
- `Splunk_TA_cribl + cribl_metrics index` — required for `mstats` against Cribl-emitted metrics indexed in Splunk.
- `TA-nix` — required for `cpu_metric` / `vmstat_metric` source-types.
- `Splunk_TA_windows` — required for `Perfmon:*` source-types.
- `Splunk DB Connect` — required to monitor JDBC inputs.
- `Splunk App for SOAR / SOAR integration` — required for `| trackmesplksoar`.
- `sa-littlehelper` — Splunk SA app providing `| bundlefiles` (Splunkbase #6368).
- `TrackMe remote account` — when the search is meant to run against a remote Splunk via `splunkremotesearch`.

See *TrackMe Ecosystem — Companion Add-ons* for the full companion-add-on inventory.

Catalog by category

cribl_edge

Two templates covering Cribl Edge fleet health, one via the Cribl management API and one via Cribl's internal metrics indexed inside Splunk.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
cribl_edge_f	Cribl	Polls Cribl Edge worker nodes via the Cribl management API and alerts if a node stops heartbeating.	cribl.edge.delay_since_l	every 5 min	Cribl API (| cribl)	TA-trackme-cribl	no
cribl_edge_f	Cribl	Uses Cribl Edge internal metrics indexed in Splunk to compute per-host event-delay and surface silent nodes.	cribl.edge.delay_event_s	every 5 min	mstats on cribl_me	Splunk_TA_ + cribl_met index	no

cribl_logstream

Nine templates that wrap the canonical Cribl Stream health, traffic and resource metrics. Traffic-flow templates (route / pack / pipeline / total in / out) are outlier-friendly so you get drift detection out of the box.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
cribl_logstream	Cribl	Tracks the percentage of healthy Cribl Stream inputs by input type and group.	cribl_logstr health. health_input	every 5 min	mstats on cribl_me	Splunk_TA_	no
cribl_logstream	Cribl	Tracks the percentage of healthy Cribl Stream outputs (devnull excluded) and surfaces unhealthy destinations.	cribl_logstr health. health_outpu	every 5 min	mstats on cribl_me	Splunk_TA_	no
cribl_logstream	Cribl	Watches per-host CPU consumption across the Cribl Stream worker fleet (avg / max / p95).	cribl_logstr {avg,max, perc95}_cpu_	every 5 min	mstats on cribl_me	Splunk_TA_	no
cribl_logstream	Cribl	Detects blocked and back-pressured outputs that are starving the pipeline.	cribl_logstr output. {blocked, backpressure	every 5 min	mstats on cribl_me	Splunk_TA_	no
cribl_logstream	Cribl	Tracks per-Pack event volume in / out and applies ML outlier detection to spot abnormal pack-level traffic.	cribl_logstr pack. pack_{in, out}_events	every 5 min	mstats on cribl_me	Splunk_TA_	yes
cribl_logstream	Cribl	Tracks per-pipeline event in / out / dropped and their pct sent vs pct dropped to flag silently dropping pipelines.	cribl_logstr pipeline. {in,out, dropped}_eve pct_{sent, dropped}_eve	every 5 min	mstats on cribl_me	Splunk_TA_	yes
cribl_logstream	Cribl	Per-route in / out bytes and events with ML outlier detection — catches abnormal traffic shifts on a specific route.	cribl_logstr route. route_{in, out}_{bytes, mbytes, events}	every 5 min	mstats on cribl_me	Splunk_TA_	yes
cribl_logstream	Cribl	Aggregate input throughput per input source (bytes + events) with outlier detection.	cribl_logstr total. total_in_{by events, mbytes}	every 5 min	mstats on cribl_me	Splunk_TA_	yes
cribl_logstream	Cribl	Aggregate output throughput per destination (bytes + events) with outlier detection.	cribl_logstr total. total_out_{b events, mbytes}	every 5 min	mstats on cribl_me	Splunk_TA_	yes

splunk_infrastructure

The largest category — 17 templates covering indexer / SH / KV-store / cluster health, parsing quality, queue saturation, and forwarder fleet tracking. This is where most "is Splunk healthy?" alerts live.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
splk_bucket_health	Splunk	Per-index bucket health — surfaces prematurely-rolled (small) buckets and quarantined buckets (a typical symptom of timestamp parsing failures).	splunk.bucket.{healthy_pct, small_pct, quarantined_, total_count, oldest_event, index_range,	every 30 min	| dbinspect	none	no
splk_bundle_size	Splunk	Knowledge-bundle size per app on the SH tier — a bloated bundle slows search dispatch.	splk.bundle_size_	every 15 min	| bundlefiles	sa-littlehelper (Splunkbase #6368)	no
splk_deployment_se	Splunk	Identifies deployment-server clients that have stopped phoning home (silent UF dropout).	ds.time_since_l	every 5 min	_dsphonehome	none	no
splk_dma	Splunk	Data Model Acceleration completeness, size, run duration and bucket count — surfaces stuck DM accelerations with ML outliers.	dma.{complete_p, size_mb, runduration_, buckets_cou	every 5 min	mgmt API (/servicesadmin/summari	none	yes
splk_kvstore_size	Splunk	Per-collection KV-store size against thresholds + total KV-store size.	infrastructu, kvstore.alltime.total_size_, size_alert.detected_co	every 5 min	_introspection via /servicesserver/introspkvstorecollect	none	no
splk_kvstore_statu	Splunk	KV-store service readiness — alerts when the KV-store goes degraded / down.	(status only, no numeric metric)	every 5 min	mgmt API (/servicekvstorestatus)	none	no
splk_large_lookup_	Splunk	Detects oversized lookup files that slow bundle replication, with outlier detection on total lookup size.	lookups.total_size_,	every 12 hours	mgmt API (/data/transfolookups	none	yes
splk_parsing_issue	Splunk	Aggregates WARN / ERROR from LineBreakingProcessor / AggregatorMiningProcessor / DateParserVerbose — the canonical bad-props.conf symptoms.	splunk.parsing.errors_coun + dcount of sourcetypes / sources / hosts	hourly	_internal (splunkd.l	none	no
splk_queues_fillin	Splunk	Tracks indexing-pipeline queue fill (parsing, aggregator, typing, indexer) — surfaces backpressure indicating scale issues.	queue.avg_pct_fil, queue.max_queue_f, …	every 5 min	_internal (metrics.log)	none	no
splk_search_head_a	Splunk	Search-head scheduler activity (eligible / dispatched / completed / lag / runtime) with ML outliers — catches drift	sh.{actions_tr, active_hist, active_sear, delayed,	every 5 min	_internal	none	yes

splunk_data_collection

Nine templates focused on the data flowing **into** Splunk — license volume, event / host drop detection, HEC errors, DB Connect inputs, and basic host tracking.

Note

Since TrackMe 2.4.18 the **VOL** (Volume Outliers) component covers per-index licensed volume out of the box — ML outliers, inactivity, daily volume, trend and projection, with the history backfilled at creation (*VOL — Volume Outliers*). The license-usage templates below (`splk_detect_daily_variations_volume_*`, `splk_license_usage_per_index_*`) remain available for custom needs.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
splk_dbconnect	Splun	DB Connect modular-input health — alerts if a JDBC input is disabled.	(state only, no numeric metric)	every 5 min	mgmt API (/configs conf-db	Splunk DB Connect	no
splk_detect_daily_v	Splun	Daily global license-pool volume with ML outliers — catches abnormal day-over-day spikes.	license. volume_mb	daily @ 04:10 UTC	_internal (license_usa	none	yes
splk_detect_daily_v	Splun	Daily license volume per index with ML outliers — same as global but per-index granularity.	license. pool_volume	daily @ 04:15 UTC	_internal (license_usa	none	yes
splk_detect_drop_ev	Splun	Hourly per-index event count and distinct host count (absolute window) with ML outliers — catches feed drop-outs.	splunk. feeds. idx. {events_cou dcount_host	hourly	tstats	none	yes
splk_detect_drop_ev	Splun	Same as absolute but with a rolling time window for near-real-time drop detection.	splunk. feeds. idx. {events_cou dcount_host	every 5 min	tstats	none	yes
splk_hosts_tracking	Splun	Simple per-host data-arrival tracking by index + sourcetype — surfaces silent hosts.	de- lay_event_s delta_inges	every 5 min	tstats	none	no
splk_lastchanceinde	Splun	Detects events landing in lastchanceindex (a sign of routing / parsing misconfiguration).	lastchancei {events_cou sourcetypes	every 5 min	Custom index (lastcha	none	no
splk_splunk_hec_err	Splun	HEC parsing errors broken down by parsing_err with source-IP dcount.	splunk_hec. {errors_cou parsers_in_ sources_ip_	every 10 min	_internal (HttpInput-DataHandler)	none	no
splk_volume_ingeste	Splun	Per-index daily ingested volume (license-based) with ML outliers to catch unexpected daily drops.	splunk. license. daily_volum	daily @ 03:05 UTC	_internal (license_usa	none	yes

splunk_enterprise

Splunk-Enterprise-specific licensing templates — license pool, license volume rolling windows. (The two cluster-status templates `splk_splunk_enterprise_cluster_peers_status` and `splk_splunk_enterprise_cluster_status` are tagged with this category in source but documented above under *splunk_infrastructure* because they share the cluster-health theme — readers should consult either table.)

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
`splk_license_pool_us`	Splun	Per-pool % used and absolute GB used against the licensed pool quota — alerts when a pool approaches saturation.	`license.{pool_pct_u pool_used_g`	every 5 min	mgmt API (`/services license pools`)	none	yes
`splk_license_usage_g`	Splun	Rolling global license usage at 60 min / 4 h / 24 h horizons (volume + % pool) with ML outliers — designed for near-real-time anomaly detection.	`splunk.license.last_{60m, 4h, 24h}_volume.. pool_pct_us`	every 5 min	`_internal` (license_us	none	yes
`splk_license_usage_p`	Splun	Same as global but per-index — pinpoints which index is driving a license spike.	`splunk.license.last_{60m, 4h, 24h}_volume.. pool_pct_us`	every 5 min	`_internal` (license_us	none	yes

splunk_cloud

Six Splunk-Cloud-only templates — storage, SVC consumption (global, per app, per consumer service) and rolling license volume.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
splk_license_usage_g	Splun	Splunk Cloud global rolling license usage (60 min / 4 h / 24 h) with ML outliers.	splunk.license.last_{60m,4h,24h}_volume	every 5 min	_internal (license_us	none	yes
splk_license_usage_p	Splun	Splunk Cloud per-index rolling license usage with ML outliers.	splunk.license.last_{60m,4h,24h}_volume	every 5 min	_internal (license_us	none	yes
splk_splunk_cloud_st	Splun	Splunk Cloud stack storage used vs licensed and % consumption against acceptable thresholds.	splunk_clou storage.{storage_us used_pct, license_GB}	every 5 min	_cmc_su (splunk	none	no ary)
splk_splunk_cloud_sv	Splun	Splunk Cloud stack SVC usage % vs licensed SVC for the whole stack, with ML outliers.	splunk_clou svc.{pct_svc_us utilized_sv license_svc	every 5 min	_cmc_su (splunk	none	yes
splk_splunk_cloud_sv	Splun	Per-app SVC consumption attribution.	splunk_clou svc.{pct_svc_us utilized_sv	every 5 min	_cmc_su (splunk	none	yes tribution)
splk_splunk_cloud_sv	Splun	Per-consumer-service SVC attribution (adhoc, scheduled, …).	splunk_clou svc.{pct_svc_us utilized_sv	every 5 min	_cmc_su (splunk	none	yes tribution)

splunk_soar

Nine templates wrapping the TrackMe SOAR integration — adhoc / playbook action failures, playbook concurrency, asset health, automation-broker HA, infra load / memory, service health, and Splunk-forwarding back into Splunk.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
splk_soar_actions_ad	Splun	Adhoc SOAR action failures (no playbook) — counts total / success / failed.	soar.adhoc.actions_cou _failed, _success}	every 15 min	Custom index (phanto	SOAR-to-Splunk forwarding	no
splk_soar_actions_pl	Splun	Playbook-driven SOAR action failures — counts total / success / failed.	soar.playbook.actions_cou _failed, _success}	every 15 min	Custom index (phanto	SOAR-to-Splunk forwarding	no
splk_soar_assets_hea	Splun	Active asset connectivity check against SOAR assets via the SOAR API.	(status only)	every 30 min	SOAR API (trackm	Splunk App for SOAR integration + automation user	no
splk_soar_automation	Splun	Automation-broker HA management — monitors broker pool members and updates assets when a broker goes inactive.	(operational state)	every 5 min	SOAR API	Splunk App for SOAR integration	no
splk_soar_concurrent	Splun	SOAR playbook concurrency by status (success / waiting / running / failed / cancelled) with ML outliers.	soar.playbook_st {success, waiting, running, failed, cancelled}	every 5 min	SOAR API	Splunk App for SOAR integration	yes
splk_soar_forwarding	Splun	SOAR → Splunk forwarding health — counts containers indexed and time-since-last-container, surfacing forwarder breakage.	soar.{containers containers_	every 5 min	tstats on phantom_*	SOAR-to-Splunk forwarding	no
splk_soar_infra_load	Splun	SOAR CPU load (1, 5, 15 min) with ML outliers.	(load metrics)	every 5 min	SOAR API	Splunk App for SOAR integration	yes
splk_soar_infra_memo	Splun	SOAR memory used / cached / free + percentages with ML outliers.	(memory metrics)	every 5 min	SOAR API	Splunk App for SOAR integration	yes
splk_soar_services_h	Splun	SOAR services health from the API — surfaces stopped or degraded services.	(status only)	every 5 min	SOAR API	Splunk App for SOAR integration	no

splunk_good_practices

Three templates that flag anti-patterns: searches scheduled with `all time` earliest, searches owned by a real user instead of `nobody`, and dynamic sourcetypes (numerically-suffixed `foo-1234` style).

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
`splk_good_practices_all`	Splun	Enabled scheduled searches running over `all time` — a notorious cost driver.	`good_pract splunk. alltime_sc`	every 12 hours	mgmt API (`saved/ search`	none	no
`splk_good_practices_det`	Splun	Detects dynamic sourcetypes (regex match on `-\d+$`) for review.	`splunk_goo dyn_source {detected_ total_sour regular_so`	every 12 hours	`tstats`	none	no
`splk_good_practices_run`	Splun	Scheduled searches running on behalf of a named user (vs `nobody` / service account) — a fragility risk on user offboarding.	`good_pract splunk. runasuser_`	every 12 hours	mgmt API (`saved/ search`	none	no

splunk_data_quality

A single template that wraps TrackMe's fields-quality monitoring approach.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
`splk_splunk_fields_q`	Splun	Aggregates per-(source, sourcetype, host) field-quality check outcomes — % passed / failed and totals; ML outliers for drift.	`percentage_ failed}, total_field failed, passed}`	daily @ 06:00 UTC	Custom (`index= source- type=tr`	TrackMe fields-quality plumbing	yes `quality)`

ta_nix

Two templates leveraging the Splunk TA-nix metric input.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
`ta_nix_cpu_usag`	Splunk	Per-host avg / max / p95 CPU usage from TA-nix `cpu_metric`, enriched with OS / IP metadata.	`ta_nix. {avg,max, p95}_cpu_usa`	every 5 min	`mstats` on `cpu_metr`	TA-nix	no
`ta_nix_mem_usag`	Splunk	Per-host avg / max / p95 memory usage from TA-nix `vmstat_metric`, enriched with total memory.	`ta_nix. {avg,max, p95}_mem_use`	every 5 min	`mstats` on `vm- stat_met`	TA-nix	no

ta_windows

Two templates leveraging the Splunk TA for Windows Perfmon source-types.

uc_ref	Vendor	What it detects / monitors	Metrics produced	Schedule	Source data	Dependencies	Outliers?
ta_windows_cpu_us	Splunl	Per-host avg / max / p95 CPU usage from Perfmon:CPU / % Processor Time.	ta_windows.{avg,max,p95}_cpu_usa	every 5 min	mstats on Perfmon:CPU	Splunk_TA_	no ows
ta_windows_mem_us	Splunl	Per-host avg / max / p95 memory usage from Perfmon:Memory / % Committed Bytes In Use.	ta_windows.{avg,max,p95}_mem_us	every 5 min	mstats on Perfmon:Mem	Splunk_TA_	no ows

Cross-cutting observations

ML outlier coverage

Twenty of the 65 templates pre-wire native ML outlier detection (an `outliers_metrics` block in their SPL). Concentrated in:

- license / volume / SVC consumption — `splk_detect_daily_variations_volume_*`, `splk_license_*`, `splk_volume_ingested_per_day`, `splk_splunk_cloud_svc_usage*`,
- Cribl Stream traffic-flow — `cribl_logstream_{route,pack,pipeline,total_traffic_*}`,
- Splunk infra activity — `splk_search_head_activity`, `splk_search_head_healthmon_artifact_count_remote`, `splk_splunk_infra_log_level_variations`, `splk_dma`, `splk_large_lookup_files`,
- SOAR runtime — `splk_soar_concurrent_playbooks`, `splk_soar_infra_{load,memory}`,
- Data-quality drift — `splk_splunk_fields_quality`,
- Event-count drop detection — `splk_detect_drop_events_count_{absolute,rolling}`.

These are the templates to keep enabled when you turn on native outlier detection in a tenant. See *Outlier detection*.

Schedule cadence distribution

- **5-minute cadence** — 48 templates (the default for near-real-time health).
- **10 – 15-minute cadence** — 3 templates (`splk_bundle_size`, `splk_splunk_hec_errors`, the two SOAR action-failure templates).
- **30-minute cadence** — 3 templates (`splk_bucket_health_per_index`, `splk_splunk_infra_log_level_variations`, `splk_soar_assets_health`).
- **Hourly** — 2 templates (`splk_parsing_issues_per_component`, `splk_detect_drop_events_count_absolute`).
- **Every 12 hours** — 4 templates (the three `splunk_good_practices` + `splk_large_lookup_files`).
- **Daily** — 4 templates (the three license-roll-up templates at 03:05 / 04:10 / 04:15 UTC, and `splk_splunk_fields_quality` at 06:00 UTC).

Important

When enabling a large number of templates simultaneously, watch your SH scheduler load — the 48 five-minute templates dispatched at `*/5 * * * *` will all queue at the same minute mark. Consider staggering by editing `uc_cron` to use offsets (e.g. `*/5 * * * *` for some, `1-59/5 * * * *` for others) at enablement time.

Source-data hot spots

- `license_usage.log` (`_internal`) is touched by 6 templates — license is the single most-monitored signal in the catalog.
- `_internal` and `_introspection` combined back the bulk of Splunk infra templates — no external add-on needed.
- `mstats` is the workhorse for all Cribl, TA-nix and TA-windows templates — metric-store performance matters for these.
- `| rest` against the local management API powers SHC, indexer cluster, KV-store, DMA, lookups, DB-Connect and good-practices templates — these are cheap but require the right capabilities on the running role.

Gap-analysis prompts

Use the catalog above as a checklist. The following questions help you spot coverage gaps you may not have realised you had — work through them one by one when reviewing a tenant's FLX inventory.

License & cost

- Am I monitoring **per-index license volume** so I can attribute spikes and drops? (the **VOL** component, *VOL — Volume Outliers*; legacy template `splk_license_usage_per_index_*`)
- Am I monitoring **global license volume** with ML outlier detection? (`splk_license_usage_global_splunkenterprise` / `..._splunkcloud`; the VOL Global license usage tab for the environment-wide reading)
- Am I monitoring **license pool % used** (Splunk Enterprise)? (`splk_license_pool_usage`)
- On Splunk Cloud, am I tracking **storage usage**? (`splk_splunk_cloud_storage_usage`)
- On Splunk Cloud, am I tracking **SVC usage globally, by app and by consumer**? (`splk_splunk_cloud_svc_usage*`)
- Do I have a **daily licensing roll-up** I can correlate with billing? (`splk_volume_ingested_per_day`, `splk_detect_daily_variations_volume_*`)

Splunk indexer / cluster health

- Am I catching **SHC member dropout**? (`splk_splunk_shc_global_status`, `splk_splunk_shc_members_status`)
- Am I catching **indexer cluster degradation** and balancing deviation? (`splk_splunk_enterprise_cluster_status`, `splk_splunk_enterprise_cluster_peers_status`)
- Am I tracking the **search head remote artifact count**, especially across SHC members? (`splk_search_head_healthmon_artifact_count_remote`)
- Am I monitoring **bucket health** (small + quarantined buckets — the timestamp-parsing canary)? (`splk_bucket_health_per_index`)
- Am I watching **queue saturation** at the indexer tier? (`splk_queues_filling`)
- Am I watching **KV-store status and size**? (`splk_kvstore_status`, `splk_kvstore_size`)
- Am I tracking the **bundle size** on the SH tier? (the silent search- slowdown source) (`splk_bundle_size`)
- Am I watching **lookup file sizes** before they blow up bundle replication? (`splk_large_lookup_files`)

Data collection & ingestion

- Am I catching **silent feed drops** (event count or host count collapsing)? (`splk_detect_drop_events_count_absolute` / `..._rolling`)
- Am I catching **silent host dropout**? (`splk_hosts_tracking`)

- Am I monitoring the **last-chance-index** for routing / parsing misconfiguration? (`splk_lastchanceindex`)
- Am I monitoring **HEC parsing errors** and their source IPs? (`splk_splunk_hec_errors`; the *Splunk HEC* tenant covers the endpoints, pools and tokens out of the box)
- Am I monitoring **DB Connect** modular inputs? (`splk_dbconnect`; the *Splunk Forwarder Collectors* tenant covers every input of the collectors out of the box)
- Am I monitoring **deployment-server clients** that stop phoning home? (`splk_deployment_server_clients`; the *Splunk UF Fleet* tenant covers the agents and the Deployment Servers out of the box)
- Am I monitoring **UF version compliance** across my forwarder fleet? (`splk_uf_clients_versions_tracking`; also a Splunk UF Fleet policy)
- Am I catching **parsing-time errors** (LineBreaking / Aggregator / DateParser warnings)? (`splk_parsing_issues_per_component`)

Search-head workload & performance

- Am I monitoring **SH scheduler activity** (lag, dispatched, eligible)? (`splk_search_head_activity`)
- Am I monitoring **DMA acceleration completeness** to catch stuck accelerations? (`splk_dma`)
- Am I monitoring **Splunk log-level variations** to catch error storms early? (`splk_splunk_infra_log_level_variations`)

Splunk good practices

- Am I auditing **all-time scheduled searches**? (`splk_good_practices_alltime_scheduled`)
- Am I auditing **scheduled searches that run as a real user** (offboarding risk)? (`splk_good_practices_run_as_user_scheduled`)
- Am I auditing **dynamic sourcetypes** that bloat the metadata cache? (`splk_good_practices_detect_dynamic_sourcetype`)

Data quality

- Am I running **fields-quality monitoring** with ML outliers on the CIM-mapped data sources? (`splk_splunk_fields_quality`)

Cribl Stream / Cribl Edge

- Am I monitoring **input and output health %** across the Cribl Stream fleet? (`cribl_logstream_health_inputs`, `cribl_logstream_health_outputs`)
- Am I detecting **blocked or back-pressured outputs**? (`cribl_logstream_output_destination_pressure`)
- Am I tracking **route / pack / pipeline traffic shifts** with ML outliers? (`cribl_logstream_{route,pack,pipeline,total_traffic_inputs,total_traffic_outputs}`)
- Am I tracking **Cribl host CPU usage**? (`cribl_logstream_hosts_cpu_usage`)
- Am I tracking **Cribl Edge node heartbeats** (API or indexed metrics)? (`cribl_edge_fleet_api`, `cribl_edge_fleet_metrics`)

SOAR

- Am I tracking **SOAR playbook failures** and adhoc-action failures? (`splk_soar_actions_playbooks_failures`, `splk_soar_actions_adhocs_failures`)
- Am I tracking **playbook concurrency** by status? (`splk_soar_concurrent_playbooks`)
- Am I tracking **SOAR asset health** with an active connectivity check? (`splk_soar_assets_health`)

- Am I managing **automation broker HA**? (`splk_soar_automation_brokers_manage`)
- Am I tracking **SOAR services health** (which daemons are alive)? (`splk_soar_services_health`)
- Am I tracking **SOAR infrastructure load and memory**? (`splk_soar_infra_load`, `splk_soar_infra_memory`)
- Am I tracking **SOAR-to-Splunk forwarding** (so I notice when SOAR stops shipping containers)? (`splk_soar_forwarding_splunk`)

Host OS metrics (if you collect them in Splunk)

- Am I tracking **Linux CPU / memory** for hosts running TA-nix? (`ta_nix_cpu_usage`, `ta_nix_mem_usage`)
- Am I tracking **Windows CPU / memory** for hosts running Splunk_TA_windows? (`ta_windows_cpu_usage`, `ta_windows_mem_usage`)
- Am I monitoring **Splunk-instance CPU / memory** from introspection (separate from the OS-level TA above)? (`splk_splunk_infra_cpu_used`, `splk_splunk_infra_mem_used`)

Coverage gaps not addressed out of the box

A few common monitoring needs are **not** covered by the pre-built FLX library and require either a custom FLX use case (the wizard supports it) or a different TrackMe component:

- **Authentication failure spikes** (login storms, brute-force) — not in FLX library; build a custom FLX, or rely on Splunk ES.
- **Scheduler skipped-search ratios per user / app** — partially covered by the WLK component (see *WLK — Workload Knowledge*).
- **Field-extraction performance per sourcetype** — handled by the FQM component (see *Use Case Demo: Fields Quality (CIM and non-CIM)*).
- **Saved-search execution failures** — handled by WLK, not FLX.
- **Splunkbase app version drift** — no template; would require a custom FLX pulling `/services/apps/local`.
- **Bucket replication factor / search factor mismatch** — partially surfaced by `splk_splunk_enterprise_cluster_peers_status`, but no dedicated alert.
- **Cribl Stream worker node version drift** — no template; equivalent of `splk_uf_clients_versions_tracking` for Cribl would need a custom FLX.
- **Splunk Cloud index growth forecasting** — no template; storage % is monitored but trend-based forecasting is left to custom searches.

If you spot a recurring need not covered above, the FLX wizard is the supported way to extend the catalog — see *FLX — Flex Objects*. Contributing a JSON file back to `package/lib/flx_library/` makes it available to every TrackMe deployment.

Appendix — source files

All 65 templates live under `package/lib/flx_library/<uc_ref>.json` inside the TrackMe Splunk app. The JSON schema is shared across templates; consult any file for the full set of fields including `uc_earliest` / `uc_latest` time bounds, `uc_implementation_comments` for caveats and prerequisites, and `uc_search` for the actual SPL.

TrackMe Ecosystem — Companion Add-ons

What this page is

An inventory of every TrackMe companion technology add-on (TA) and the Splunk SOAR (Phantom) connector that extend TrackMe beyond the main `trackme` Splunk app. For each one: purpose, data it ingests / produces, Splunk surfaces (commands, REST handlers, alert actions), the TrackMe use cases it unlocks, dependencies, and which FLX templates pair with it (if any).

Audience

TrackMe administrators, architects, and operators scoping what else they can plug into TrackMe.

Three axes of extension

TrackMe alone monitors **data that already lives in Splunk** (indexes, scheduled searches, hosts, sourcetypes, fields, lookups via Splunk REST) and orchestrates alerts / AI within the Splunk surface. The companion add-ons extend TrackMe in three orthogonal directions:

1. **External data sources made first-class TrackMe entities —** Cribl Stream / Edge pipelines and fleet nodes (`TA-trackme-cribl`), Google SecOps log feeds (`TA-trackme-google-secops`), and lookup files / KV-stores at filesystem-mtime precision (`TA-trackme-lookupmonitor`). All can be turned into FLX or hybrid-tracker entities that participate in priority, SLA, ML outliers, stateful alerting and the AI Assistant — exactly like native DSM entities.
2. **Bidirectional incident orchestration with external systems —** `TA-trackme-xsoar` (Cortex XSOAR active-command incident lifecycle with a resilient retry store) and `trackme-app-on-soar` (Splunk SOAR playbook-driven ACKs, maintenance windows, ML retraining, logical-group management and priority / policy bulk-edits).
3. **Operational resilience and deployment flexibility —** `TA-trackme-conf-manager` for transactional, retryable TrackMe REST orchestration (CI / IaC pipelines).

Without the companions you keep an in-Splunk monitoring tool. With them, TrackMe becomes a multi-vendor data-quality control plane that closes the loop into the SOC's ticketing, automation, and configuration-management workflows.

Common architectural patterns

Two patterns appear repeatedly across the companion add-ons:

- **API-bridge pattern —** A single custom command (`cribl` / `secops`) plus a small persistent REST handler for account configuration. Used by `TA-trackme-cribl` and `TA-trackme-google-secops`. From TrackMe's perspective, the command makes external API data look like SPL events so any FLX or hybrid tracker can consume it.
- **Resilient transaction store —** A KV collection (per-account, replicate-false) records pending / failed REST transactions, plus a 15-minute scheduled saved-search replay loop. Used by `TA-trackme-conf-manager` (against TrackMe's own REST) and `TA-trackme-xsoar` (against the XSOAR REST). Guarantees that a transaction eventually lands even through network blips or destination outages.

External data sources

TA-trackme-cribl

Purpose	Two-way Splunk-to-Cribl-API bridge for both Cribl Stream (Logstream) and Cribl Edge, callable from SPL via `| cribl`.
Splunk surfaces	Custom command `cribl` (GET / POST / DELETE with built-in helpers: `get_groups_conf`, `get_global_metrics`, `get_destinations_metrics`, `get_pipelines_metrics`, `get_routes_metrics`, `get_edge_fleets`, `get_edge_nodes`, `get_edge_fleets_conf`); persistent REST handler at `/services/cribl/v1/...` for account access / test connectivity; UCC account config (Cloud or on-prem leader URL, client_id / secret, SSL, RBAC); capability `criblapi`, role `cribl_api`.
TrackMe use cases unlocked	**FLX live-API tracking** of Cribl Edge node health via `cribl_edge_fleet_api` (the only FLX template that hard-depends on `| cribl`). **Hybrid trackers** / ad-hoc SPL where TrackMe entities can be defined directly from Cribl API output (workers count, route traffic, pipeline status, packs) without needing Cribl to forward metrics to Splunk first. Real-time queries of pipeline / route / destination metrics from inside TrackMe scheduled trackers.
Dependencies	Cribl Stream or Cribl Edge deployment with API access; Cribl client_id / secret. Independent of trackme-app at runtime; the FLX template above and the use-case docs are oriented around TrackMe.
FLX templates that pair	**Hard requirement**: `cribl_edge_fleet_api`. **Independent (relies on Cribl-shipped internal metrics in a Splunk metrics index)**: `cribl_edge_fleet_metrics`, `cribl_logstream_health_inputs`, `cribl_logstream_health_outputs`, `cribl_logstream_hosts_cpu_usage`, `cribl_logstream_output_destination_pressure`, `cribl_logstream_pack_traffic`, `cribl_logstream_pipeline`, `cribl_logstream_route_traffic`, `cribl_logstream_total_traffic_inputs`, `cribl_logstream_total_traffic_outputs`. These do not need the TA but operate in the same monitoring family.
See also	*Cribl Stream monitoring with Flex Objects*, *Cribl Stream & Cribl Edge*.

TA-trackme-google-secops

Purpose	Splunk SPL bridge to Google Security Operations (Chronicle) UDM search API.
Splunk surfaces	Custom command `secops` (runs UDM queries through `udmSearch`, auto-applies Splunk's time range, `limit` configurable); REST handler at `/services/secops/v1` for account / test connectivity; UCC account (GCP project ID, SecOps instance UUID, region, service-account JSON key encrypted, RBAC roles); capability `secopsapi`, role `secops_api`.
TrackMe use cases unlocked	Hybrid trackers and FLX entities driven by **Google SecOps log feed ingestion** — e.g. one entity per `metadata.log_type` (NGINX / APACHE / …) tracking event count, last-seen, ingest delay, or anomalies. Effectively extends DSM-equivalent monitoring to data living in Chronicle, with the same toolkit (priority, SLA, stateful alerting, AI Assistant) used for Splunk-side feeds.
Dependencies	**Hard dependency on trackme-app** — the README explicitly states: *"TrackMe must be installed and licensed on the search head(s) where this application is deployed. The ``secops`` custom command will fail with a licensing error"* — the TA reuses TrackMe's licensing framework. GCP service account with `roles/chronicle.viewer` + `roles/serviceusage.serviceUsageConsumer`. Network reachability to `{region}-chronicle.googleapis.com`.
FLX templates that pair	None ship — operators build SecOps trackers directly via the `| secops` command.

TA-trackme-lookupmonitor

Purpose	Generative SPL command that enumerates CSV-file and KV-store lookups visible to the running user with reliable filesystem mtimes and accurate row counts.
Splunk surfaces	Custom generating command `trackmelookupsmonitor` (`local = true` — runs on the dispatching SH only); UCC config with logging-level controls. No dashboards / REST API / alert actions.
TrackMe use cases unlocked	This is the **data source behind TrackMe's "Lookups monitoring" search mode in the Hybrid Trackers wizard** — see *Lookup monitoring*. It lets a tenant create TrackMe entities, one per lookup, tracking last-modification freshness and growth / shrinkage of row count — so you can build alerts like "this critical KV-store collection hasn't been updated in 24 h" or "this CSV lookup just lost 90 % of its rows". Pair with `splunkremotesearch` to inventory lookups across remote SHs (the TA must be installed on the remote SH too; trackme-app itself does not need to be there).
Dependencies	None external. Stand-alone usable; integrates with trackme-app only via the Hybrid Trackers wizard.
FLX templates that pair	None — lookups monitoring is implemented as a dedicated hybrid-tracker search mode rather than via FLX templates. Note: `splk_large_lookup_files` solves an adjacent problem (oversized lookups) using a Splunk REST call (`/services/data/transforms/lookups`) — independent of this TA.
See also	*QUICK START - Design and automate through lookup knowledge*, *Lookup monitoring*.

Bidirectional incident orchestration

TA-trackme-xsoar

Purpose	Cortex XSOAR (Palo Alto) REST-API integration with a built-in resilient retry store for incident-creation calls.
Splunk surfaces	Custom commands `xsoar` (generic GET / POST / DELETE), `xsoarstreamincident` (streaming command that creates / updates / closes incidents, keyed by `external_incident_id`), `xsoarresilient` (replay loop); persistent REST handler at `/services/xsoar/v1/...`; scheduled saved search `xsoarresilient` (every 15 min, enabled by default) replaying failed REST calls; UCC account configuration (key + key ID), capability `xsoarapi`.
Storage	Two KV collections (replicate-false): `kv_xsoar_resilient_store` (pending / failed transactions — account, endpoint, body, status, attempts, last_error), `kv_xsoar_incidents_store` (XSOAR incident IDs ↔ external IDs mapping so updates / closures from TrackMe stateful alerts can reach the right XSOAR incident).
TrackMe use cases unlocked	Bidirectional bridge between TrackMe stateful alerts and Cortex XSOAR incidents — the README documents the exact wiring as **active commands** in TrackMe stateful alerts: `commands_opened` creates an XSOAR incident, `commands_updated` patches it with new state, `commands_closed` closes it. The resilient store guarantees incident parity even through network blips or XSOAR outages. **Use case**: TrackMe entity goes red → XSOAR ticket auto-opens; entity recovers → ticket auto-closes.
Dependencies	Cortex XSOAR API key + key ID; network reachability to XSOAR. No external trackme-app dependency at install time (FIPS- compatible, standalone or SHC), but is purpose-built for TrackMe stateful-alert integration.
FLX templates that pair	None. The `splk_soar_*` family of FLX templates targets **Splunk SOAR (Phantom)**, not XSOAR.
See also	*Palo Alto XSOAR Integration*.

trackme-app-on-soar (Splunk SOAR connector)

Purpose	A **Splunk SOAR (Phantom) app**, *not* a Splunk TA — runs inside SOAR and exposes TrackMe REST endpoints as SOAR actions so playbooks can drive TrackMe entities.
SOAR surfaces (actions)	`test connectivity`; `ack_get`, `ack_manage` (sticky / unsticky, set, extend); `maintenance_status`, `maintenance_enable`, `maintenance_disable`; `tenants_ops_status` (per-tenant OPERATIONAL / DEGRADED); `remote_accounts_check_connectivity`; `ml_outliers_train_models`, `ml_outliers_run_monitor`, `ml_outliers_reset_models`, `ml_outliers_get_models`, `ml_outliers_add_period_exclusion`; `component_get_entity`, `component_manage_entity` (enable / disable / delete + per-action extras `manage_dsm_sampling`, `update_hours_ranges`, `update_wdays`, `update_priority`, `update_lag_policy`, `update_dcount_host`, `update_manual_tags`); `smart_status`; `logical_group_get_group_for_entity`, `logical_group_manage`.
TrackMe use cases unlocked	SOAR-driven, playbook-orchestrated TrackMe operations: auto-ACK an entity from a SOAR playbook when an upstream maintenance ticket opens; un-ACK when it closes; trigger Smart Status investigation from SOAR after correlating with another SIEM signal; bulk priority / lag policy / hours-ranges adjustments driven by an external CMDB; on-demand ML model retraining / model exclusion windows around known noisy periods (deployments, batch jobs); enable / disable global maintenance from a SOAR incident-response playbook before scheduled changes; logical-group membership management from SOAR (group entities by app / service when the inventory changes upstream).
Dependencies	Splunk SOAR (Phantom) 6.0.2+. Splunk bearer token with TrackMe RBAC. Network reachability to Splunkd HTTPS. Requires trackme-app on the Splunk side.
FLX templates that pair	None — this is a SOAR-side connector, not a Splunk add-on. The `splk_soar_*` FLX templates monitor a Splunk SOAR deployment's *health*; they consume Splunk-side SOAR-forwarded data and TrackMe's own `trackmesplksoar` command, independent of this SOAR connector.
See also	*TrackMe App on SOAR*, *Splunk SOAR*.

Operational resilience

TA-trackme-conf-manager

Purpose	Resilient transactional layer for replaying failed configuration changes against the TrackMe REST API — accepts a config transaction, persists it, and re-attempts on a schedule until it lands.
Splunk surfaces	Custom search command `trackmeconfmanager` (modes `live` / `get` / `simulation`); persistent REST handler at `/trackme_conf_manager/v1`; scheduled saved search `trackme_conf_manager_producer` (every 15 min, **disabled by default** — operator opt-in) that calls `| trackmeconfmanager mode="live"` to replay pending transactions; diagnostic saved searches for transactions, simulation, access logs, KV-store content.
Storage	KV collection `kv_trackme_conf_manager` (replicate-false).
TrackMe use cases unlocked	Idempotent / retryable orchestration of TrackMe REST calls — useful for SOAR playbooks, infra-as-code (Ansible / Terraform- style) deployments, and CI pipelines that bulk-configure tenants, entities, policies, ACKs, maintenance windows, etc. Submit a transaction once, the TA keeps trying until it succeeds.
Dependencies	Requires trackme-app (it wraps TrackMe REST calls). No external system.
FLX templates that pair	None — pure orchestration, not a data source.

At-a-glance dependency map

Quick reference: who needs whom.

Add-on	Needs trackme-app?	External system needed
`TA-trackme-cribl`	No (works standalone, but TrackMe use cases require it)	Cribl Stream or Cribl Edge with API access
`TA-trackme-google-secop`	**Yes — hard licensing dependency**	GCP Chronicle SecOps with service account
`TA-trackme-lookupmonit`	No (integrates via Hybrid Trackers wizard)	None
`TA-trackme-xsoar`	No (FIPS-compatible, standalone)	Cortex XSOAR with API key
`TA-trackme-conf-manage`	**Yes — wraps TrackMe REST**	None
`trackme-app-on-soar`	**Yes — calls TrackMe REST**	Splunk SOAR (Phantom) 6.0.2+

TrackMe Adoption Roadmap

What this page is

A **descriptive** map of the journey we typically see TrackMe customers travel through, organised in six phases. It is **not** a prescriptive "you must do X before Y" ladder — every customer moves at their own pace, and the right sequencing depends on team maturity, scale, and what brought them to TrackMe in the first place.

What we **do** observe consistently is that **certain capabilities are counter-productive if enabled too early** — a customer still stabilising on data-source monitoring will be hurt, not helped, by turning on ML outliers and AI status reports on day one. This page makes those anti-patterns explicit so you can resist the temptation to enable everything at once.

Audience
: Existing TrackMe customers looking for "what should I focus on next?", new customers planning a rollout, account teams and partners wanting a shared mental model.

Companion pages
: *TrackMe Use Cases — Organised by User Intent* (what you *can* do), *TrackMe Capability × Component Matrix* (which features apply to which components), *FLX (Flex Objects) Use Case Template Catalog* (pre-built use cases), *TrackMe Ecosystem — Companion Add-ons* (companion add-ons).

The shape of the journey

The six phases below are the rhythm we see across most TrackMe deployments — from small Foundation tenants to large multi-tenant Unlimited estates:

Phase	Focus	Customer question that triggers it
1	**Data source monitoring** (DSM)	*"Is my data still arriving as expected?"*
2	**Hosts and endpoints** (DHM)	*"How do I know an individual host stopped sending data?"*
3	**Volume outliers with ML** (the VOL component — 2.4.18+, every edition; FLX volume tenant or DSM merged-mode outliers as the legacy alternatives)	*"Now that I track feeds and hosts, how do I detect abnormally low or high volumes?"*
4	**Workload Knowledge** (WLK) — *optional interlude*	*"Can I extend TrackMe to monitor Splunk's own scheduled-search health?"* — usually prompted by us, not by the customer
5	**Flex expansion, Splunk infra, third-party**	*"What else can I monitor with TrackMe?"*
6	**Steady state** — governance, growth, what's next	*"How do we keep TrackMe alive, growing, and shared across teams?"*

Note

Phase 4 is deliberately framed as an **optional interlude**, not a sequential gate. Workload Knowledge is one of TrackMe's most valuable components, but it is rarely what attracts customers to the product in the first place — it is a capability we proactively teach about when the customer is ready, rather than one they ask for. Some customers adopt WLK between phase 2 and phase 3; others much later; some not at all.

How we work with you through the phases

There are no fixed durations and no hard gates. The pattern we follow:

- We **periodically prompt you** about what's typically next — through release notes, account-team conversations, or the AI Assistant surfacing capabilities you have not yet enabled.
- You **decide pace and order** based on what your team is ready for, what the data is telling you, and what your stakeholders are asking about.
- The *TrackMe Use Cases — Organised by User Intent* and *FLX (Flex Objects) Use Case Template Catalog* pages are the **menu**; this page is the **suggested order**.

Phase 1 — Data source monitoring

The customer question that opens this phase

"Is my data still arriving as expected — and how do I get told the moment it stops?"

This is the phase that brought most customers to TrackMe in the first place, and it is where every deployment starts. The goal is to know, reliably and quickly, when a data feed is no longer arriving on time — nothing more, nothing less.

Primary focus

- **Component**: `DSM` (Data Source Monitoring), tracking `(index, sourcetype)` feeds.
- **Primary KPI**: ingestion **delay** (how long ago we last received data). Secondary KPI: **latency** (the gap between event time and index time).
- **Tenant pattern**: one or two DSM-only tenants, usually scoped by organisational responsibility (e.g. one tenant per data team, or one per environment).

See *QUICK START - Starting with TrackMe: (feed tracking quickstart)* for the canonical walkthrough.

What we typically build first

- The default DSM tracker, discovering `(index, sourcetype)` entities automatically.
- Static delay thresholds tuned to the customer's expectations, or **variable delay** thresholds (day-of-week × hour-of-day) if there are predictable business-hours / overnight patterns.
- Email-based stateful alerting (open / update / close) so each real state transition produces exactly one notification thread.
- Initial design of the **alerting workflow**: do notifications go to email only, to a third-party ITSM / ticketing platform, or both? This decision is usually revisited later, but it is worth agreeing on a starting point at phase 1.

Fighting the noise

The most important phase-1 skill is **classification** — separating the few feeds that genuinely matter from the many that do not. The TrackMe concepts that earn their keep here:

- **Priority policies** (regex / lookup / SPL) — assign critical / high / medium / low / pending priorities without touching individual entities.
- **Tag policies** — attach routing labels (`team:auth`, `env:prod`) that survive entity recreation.
- **SLA policies** — bucket entities into platinum / gold / silver classes for uptime accounting.
- **Notes** and **labels** — document feeds as the team learns about them ("re-deploy at 03:00 on 2026-05-08, root cause INC-12345").

See *Priority*, *Tags*, and *Using SLA alerting to build a 2-tier monitoring system*.

Organisational dimension

Phase 1 is where the customer's **team learns the product** — what an entity is, what a tenant is, how the stateful alerting workflow feels, what categorisation conventions to use. Most customers spend **significant time** here, and that is healthy: skipping ahead before the team is comfortable creates noise debt that compounds in every later phase.

What to not do yet

Resist the temptation to enable the following in phase 1 — they will generate noise faster than the team can metabolise it:

- **ML outlier detection** — the data has not had time to train stable models, and even if it has, the team is not yet equipped to triage outlier alerts.
- **AI status reports in alert emails** — useful, but adds cognitive load before the basics are mastered.
- **Hybrid trackers**, **logical groups**, and **smart-status investigations** — all valuable later, but premature here.

The right phase-1 alert is a clean "this feed has stopped, here is the priority and SLA class, here is who to call". Build that solidly before layering anything else on top.

Edition variants

- All editions support phase 1 in full.
- Foundation has DSM, DHM, MHM and one tenant only — perfect for phase 1.

Phase 2 — Hosts and endpoints

The customer question that opens this phase

"Now that I monitor my feeds, how do I know when an individual host stops sending data — even if its index and sourcetype are still busy from other hosts?"

This is the natural follow-up to phase 1. Feeds answer "is this data flowing at all?". Hosts answer "is **this specific machine** flowing?". Most customers reach phase 2 within weeks or a few months of phase 1.

Primary focus

- **Component**: `DHM` (Data Host Monitoring), tracking host × sourcetype activity.
- **Tenant pattern**: a **dedicated DHM tenant** (sometimes several). Mixing host tracking into the existing feeds tenant is the most common phase-2 mistake — the volumes and the operational lifecycle are different enough that they deserve their own container.
- **Break-by mode**: most deployments start with one entity per `(host, sourcetype)` pair. TrackMe 2.3.23+ supports `breakby_extra_fields` on DHM hybrid trackers, which lets you add extra metadata dimensions (e.g. `source`) to the combo grain — use this when `(host, sourcetype)` alone does not separate the activity enough.

See *Hybrid trackers* and *Notes*.

The scale challenge

Large customers can have **tens of thousands of endpoints** across heterogeneous fleets — servers, network gear, Windows desktops, Linux containers, IoT, third-party agents. Three considerations dominate:

- **Inventory quality** — does the customer have a usable CMDB? If yes, integrate it (priority, tags, SLA all driven by lookup or SPL-search policies against the CMDB). If no, this is the right moment to start one.
- **Discovery scope** — use the DHM **blocklist** and **allowlist** aggressively. Tracking every device on the wire is rarely what you want; tracking the hosts that belong to a known inventory is.
- **Tenant design** — multiple DHM tenants by responsibility area (e.g. one per OS family, one per business unit) is often clearer than one mega-tenant.

See *CMDB integration* for the CMDB integration patterns and *QUICK START - Design and automate through lookup knowledge* for the lookup-driven approach.

Organisational dimension

Phase 2 typically pulls in **a wider set of stakeholders** than phase 1: endpoint owners, OS teams, CMDB owners, sometimes security. The operational rhythm of "an endpoint went silent" is genuinely different from "a feed went silent" — it usually means a host is down, an agent is broken, or someone decommissioned a machine without telling anyone. Phase 2 is where TrackMe starts being a **cross-team conversation**.

What to not do yet

- **ML outlier detection on DHM** — per-host event volume is too noisy in most environments to model usefully. The DHM defaults in TrackMe (`mloutliers_dhm=0`) reflect this — leave outliers off here.
- **Logical groups** before the discovery scope is stable — group membership is only useful once you trust what is being discovered.

Edition variants

- All editions support DHM.
- Foundation supports DHM but is limited to one tenant — for very large endpoint fleets, Enterprise or Unlimited is usually a better fit because you can split the load across multiple tenants.

Phase 3 — Volume outliers with Machine Learning

The customer question that opens this phase

"Now that I monitor feeds and hosts for primary KPIs, how do I detect that I am receiving unusually low or unusually high volumes of events — without writing thresholds by hand?"

Phase 3 introduces TrackMe's **machine-learning outlier detection** as a behavioural-anomaly layer **on top of** the deterministic phase 1 + phase 2 monitoring. Since 2.4.18 the recommended pattern is the **VOL** component; two legacy patterns remain for customers who set them up on earlier releases or have needs beyond the per-index licensed volume.

The recommended pattern (2.4.18+): the VOL component

VOL — Volume Outliers (*VOL — Volume Outliers*) is the seventh monitoring component and exists precisely for this phase. Enable it on a tenant — a few clicks in the tenant-creation wizard, local or remote deployment, Splunk Enterprise or Splunk Cloud — and:

- every index becomes an entity fed by the license usage logs in 5-minute buckets, incrementally (one small search per tenant every 5 minutes);
- the history (default 30 days) is **backfilled at creation**, so the ML models and the cost trend are usable on day one instead of after a training period;
- ML outliers on the rolling volume are the entity state — enabled by default, with the **detection direction** (drops, spikes, or both) chosen per index — plus an **inactivity policy** for indexes that stop consuming license;
- the daily licensed volume, a 30-day trend, a month projection and the **Global license usage (VOL)** tab answer *"who consumes my license, and is that changing?"* without any SPL.

VOL is available on **Foundation, Enterprise and Unlimited** — the edition split below no longer decides the pattern. See *Volume outliers on the Splunk license usage with Volume Outliers (VOL)* for the design and the comparison with the FLX templates.

Beyond the entities: User Activity Monitoring

Once the data and the cost are covered, the remaining blind spot of a Splunk platform is the people and the automation that use it. **User Activity Monitoring** (*User Activity Monitoring*, generally available in 2.4.19, any licence beyond Foundation) is a dedicated tenant type for that phase: one tenant per deployment, no entities, findings instead — the scheduled searches owned by people or dispatched all time, the accounts that run all-time searches, the consumers of the search capacity — with the accounts classified as human, service or system and an investigation workflow on every finding. See the white paper *Who is doing what on your Splunk? User Activity Monitoring (UAM)*.

The same phase covers the forwarding layer and the pipelines in front of Splunk, each with its own dedicated tenant type: *Splunk HEC* for the HTTP Event Collector endpoints, pools and tokens; *Splunk S2S* for the receivers and the intermediate forwarders, pool by pool; *Splunk UF Fleet* for every forwarding agent and its Deployment Servers; *Splunk Forwarder Collectors* for the heavy forwarders and syslog collectors and every input they run; and *Cribl Stream* and *Cribl Edge* for the Cribl pipelines. All seven share the findings model (*Findings, Finding groups and findings actions*) and have their own AI Agent.

The legacy Enterprise / Unlimited pattern: a dedicated FLX volume-outliers tenant

This was the recommended pattern before VOL, and remains the advanced alternative when you need KPIs the component does not carry (pool-level % used, Splunk Cloud SVC / storage) or a custom SPL. The shape:

- A **dedicated tenant** for volume outliers, kept separate from the main feeds tenant.
- Built on the FLX pre-built use cases `splk_license_usage_global_splunkenterprise` / `splk_license_usage_per_index_splunkenterprise` (or the Splunk Cloud equivalents) — see *FLX (Flex Objects) Use Case Template Catalog*.

- Why FLX over per-sourcetype DSM outliers: the license-usage feeds are **scalable and pre-aggregated** (MB per index, even at large scale) and they focus on the **index level**, which is the right granularity for volume anomaly detection. Per-sourcetype noise is filtered out by construction.
- Native ML outlier detection (DensityFunction with optional weekly seasonality) on the rolling 60-minute / 4-hour / 24-hour KPIs.

See *Outlier detection* for the ML configuration and *QUICK START - TrackMe Impact-Based Alerting (IBA)* for how outlier breaches contribute to entity state.

The legacy Foundation pattern: built-in DSM outliers in merged mode

Foundation customers do not have FLX; before VOL they could achieve similar behaviour with the **built-in DSM outlier detection** running on a DSM tracker in **merged mode**. Merged mode aggregates the entity's event-count metrics across sourcetypes for the same index, giving the ML model a clean per-index volume signal to learn from.

The trade-off: the volume-outlier signal lives **inside the main feeds tenant** rather than in a dedicated container. This is operationally simpler but conceptually less clean — most teams find it easier to separate "is this feed flowing?" from "is this feed volume anomalous?" even though the underlying data overlaps.

Organisational dimension

Phase 3 is where TrackMe usage starts to look more like an **L3 / specialist activity**. The questions get sharper:

- Who triages a volume outlier alert? (Different role from the L1 / L2 triage for a delay alert in phase 1.)
- How is priority propagated from the main feeds tenant into the volume-outliers tenant? Most commonly: a **priority policy** (lookup-based or SPL-search-based) in the volume-outliers tenant reads from the main feeds tenant so an entity that is critical for feed monitoring is also critical for volume monitoring — automatically.
- What is the alert workflow for outlier signals? Often quieter than delay alerts (no auto-pager), routed to a dashboard or a daily digest rather than waking someone up at 03:00.

What to not do yet

- **Enabling outliers on every component indiscriminately** — DHM, MHM, FQM, WLK are generally not good outlier candidates (see *TrackMe Capability × Component Matrix*). Outliers are a focused per-component opt-in, not a fleet-wide switch.
- **Volume outliers in the main feeds tenant on Enterprise / Unlimited** — defeats the purpose of having FLX. Keep them separate.
- **Rebuilding VOL with FLX templates on 2.4.18+** — the component already covers per-index licensed volume with backfill; reserve the templates for what it does not carry.

Edition variants

- **Every edition (2.4.18+)**: the VOL component, recommended.
- **Foundation** (legacy): built-in DSM outliers, merged-mode tracker, single tenant.
- **Enterprise** and **Unlimited** (legacy / advanced): dedicated FLX volume-outliers tenant.

Phase 4 (optional interlude) — Workload Knowledge

The customer question that opens this phase

"Can I extend TrackMe to monitor Splunk's own scheduled-search workload — skip rates, errors, slow runs, orphaned owners?"

This is the phase customers most often arrive at **because we tell them about it**, not because they asked. Workload Knowledge (`WLK`) is one of TrackMe's most valuable components, but it is rarely the use case that attracted them to the product in the first place.

Why we proactively teach it

WLK fills a category that Splunk itself does not address well: the **health of the scheduler estate** — which searches skipped, which errored, which take dramatically longer than they used to, which run as a user who no longer exists, which live in an app that has been disabled. Most large Splunk estates accumulate scheduler debt silently over years; WLK surfaces it in a structured way.

See *WLK — Workload Knowledge*, *WLK — Workload Knowledge*, *Monitor Splunk Workload with TrackMe's Workload component*.

Where it slots into the journey

Not strictly. Some customers adopt WLK **between phase 2 and phase 3** because they hit a scheduler-noise problem early. Some adopt it **alongside phase 5** as part of the broader Splunk-infrastructure expansion. Some never adopt it. We do not gate other phases on it.

Primary focus

- **Component**: `WLK`, tracking scheduled searches.
- **Tenant pattern**: a dedicated WLK tenant.
- **Tenant-default threshold row** — a WLK-specific feature where a single `object_id == "default"` record provides the fleet-wide fallback threshold; per-entity rows act as exceptions. Use this pattern to manage thresholds at the tenant level with a small number of overrides, rather than configuring every entity.
- **Signals to watch**: skip rate, error count, execution-time drift, versioning (the savedsearch definition has changed since the fingerprint was taken — surfaces unannounced edits), orphan owners, app enablement.

What to not do yet

- **ML outliers on WLK** — scheduler metrics are discrete (skip count, error count); thresholds are the right tool, not density-function models.

Edition variants

- **Foundation**: does **not** include WLK. Customers on Foundation must upgrade to Enterprise or Unlimited to enter this phase.
- **Enterprise** and **Unlimited**: full WLK support.

Phase 5 — Flex expansion, Splunk infrastructure, third-party

The customer question that opens this phase

"What else can I monitor with TrackMe — both inside Splunk and outside?"

Once phases 1 – 3 are operational and the team is comfortable, phase 5 opens up the full breadth of TrackMe's monitoring catalogue. There is no single "right" order here — the choice is driven by what is most valuable to the customer.

Common phase-5 workstreams

- **Splunk platform deep-dive** — indexer cluster status, SHC member health, bucket health, KV-store readiness, queue saturation, parsing errors, DMA acceleration, HEC errors, UF version compliance, knowledge-bundle size. See the `splunk_infrastructure` and `splunk_data_collection` categories in *FLX (Flex Objects) Use Case Template Catalog*. The HEC endpoints, the forwarding agents, the receivers and the collection points also have their own dedicated tenant types (Splunk HEC, Splunk UF Fleet, Splunk S2S, Splunk Forwarder Collectors), covered in the phase above.
- **License and cost monitoring** — pool usage, rolling volume windows, per-index attribution. (Splunk Cloud: storage and SVC consumption by app and by consumer service.) Per-index licensed volume is covered by the VOL component since phase 3; the templates add the pool, SVC and storage views.

- **Splunk good-practices auditing** — all-time scheduled searches, user-owned searches, dynamic sourcetypes (the `splunk_good_practices` category).
- **Splunk SOAR monitoring** — playbook concurrency and failures, asset health, automation broker HA, infra load / memory, services health, SOAR-to-Splunk forwarding. See *Splunk SOAR*.
- **Cribl Stream / Cribl Edge monitoring** (if the customer has Cribl in the stack) — since 2.4.19, the dedicated Cribl Stream and Cribl Edge tenant types monitor a deployment end to end with findings (see *Cribl Stream and Cribl Edge*); the FLX templates remain for input / output health, pipeline traffic, route shifts with ML outliers, host CPU. See *Cribl Stream monitoring with Flex Objects* and *TrackMe Ecosystem — Companion Add-ons* for the companion add-on.
- **Field-quality monitoring (FQM)** for CIM compliance and parsing accuracy. See *Use Case Demo: Fields Quality (CIM and non-CIM)*.
- **Host OS monitoring** for Linux (TA-nix) and Windows (Splunk_TA_windows) fleets — CPU / memory per host with avg / max / p95.

Organisational dimension

Phase 5 is where TrackMe stops being a tool that one team runs and becomes a tool that **many teams consume**. Different workstreams have different owners — the Splunk platform team for infrastructure, the licensing / FinOps team for cost monitoring, the SOC for SOAR, the data team for parsing quality. **Virtual Groups** become valuable here as read-only cross-tenant aggregation views for executives or central teams (see *QUICK START - Cross-tenant domain views with Virtual Groups*).

This is also the phase where customers often integrate TrackMe with their **downstream incident-response stack** — Splunk ES (notable events), Cortex XSOAR (active commands), Splunk SOAR playbooks. See *TrackMe Ecosystem — Companion Add-ons*.

What to not do yet

- **Enable every FLX template at once** — the catalog is large; 48 of the 65 templates run on a 5-minute cadence by default, so a bulk enablement can spike the scheduler load. Stagger by editing `uc_cron` offsets at enablement time, or roll out workstreams one quarter at a time.

Edition variants

- **Foundation**: phase 5 is largely inaccessible — no FLX, no WLK.
- **Enterprise** and **Unlimited**: full phase 5.
- **Unlimited**: also unlocks unlimited hybrid trackers and remote Splunk deployments.

Phase 6 — Steady state: governance, growth, what's next

The customer question that opens this phase

"TrackMe is now a central part of how we run monitoring — how do we keep it alive, growing, and shared across the team?"

Phase 6 is the **steady state**, not a destination. The platform is widely adopted, multiple teams receive alerts from it or work inside it, and the question shifts from "what do we turn on next?" to "how do we keep what we have healthy, and how do we keep growing?".

Unlike phases 1 – 5, phase 6 does not have a feature checklist. It has four themes that all operate continuously.

People — building a champion network

- **Expand the champion base over time.** A single TrackMe expert is a single point of failure. Spread knowledge across multiple team members, ideally across multiple teams.
- **Cross-team training.** SOC, IT ops, observability, business owners — each consumes TrackMe differently. Run targeted training sessions per audience, not one-size-fits-all.

- **Rotation and handoff hygiene.** When a champion leaves the team or the company, their TrackMe knowledge needs to leave with them intentionally, through documented handoff, not through luck.

Process — periodic review and gap analysis

- **Periodic gap analysis against the** *FLX (Flex Objects) Use Case Template Catalog*. Every quarter (or whatever cadence fits): walk the catalog with the team and ask "is there anything we should have enabled and have not yet?". The catalog grows with every TrackMe release.
- **Quarterly Configuration Guardian sweep.** Guardian flags drifting configuration (missing capabilities, stale tokens, missing indexes, unreachable AI provider, stale backup archives, dead health tracker). Treat the Guardian alerts list as part of the ongoing platform review, not as one-off pop-ups.
- **Tenant lifecycle hygiene.** Decommission tenants for projects that have wound down. Prune dormant entities. Review priority / tag / SLA policies for relevance — yesterday's regex might catch the wrong things today.

Platform — keeping the foundation healthy

- **Configuration Guardian as a continuous safety net** — see *TrackMe Capability × Component Matrix* for the full list of checks.
- **Backup and restore discipline.** TrackMe 2.3.22+ ships a multi-archive backup format where every tenant and the global state are independent archives. Keep the backup scheduler enabled, test restore periodically, and use the per-tenant restore mode for targeted recovery rather than blanket rollbacks. See *Backing up and Restoring TrackMe*.
- **License and cost management.** As the platform grows, license-edition fit can drift — what was right at adoption may not be right at steady state. Review periodically.
- **Version-upgrade hygiene.** Read release notes, schedule upgrades with the same care as Splunk itself, and use a non-production tenant to validate before rolling out.

Forward look — what's coming next

- **New FLX templates ship with most TrackMe releases** — re-read the *FLX (Flex Objects) Use Case Template Catalog* after each release to see what is new.
- **Companion add-ons evolve** — Cribl, XSOAR, conf-manager, Google SecOps all gain features over time; check *TrackMe Ecosystem — Companion Add-ons* periodically.
- **AI Agents** — the AI layer keeps growing: the AI Advisors run unattended under *AI Agents automation*, *AI Routines* evaluate your own standing instructions, and every dedicated tenant type has its own *AI Agent* that triages findings on demand or on an hourly schedule, within the decisions the tenant allows. Re-read *Artificial Intelligence in TrackMe* after each release.
- **Periodic check-in with the vendor.** We (TrackMe) will periodically reach out about what's new and what is worth your time. Treat these conversations as part of the adoption journey rather than as interruptions — the platform is still growing, and so is your usage of it.

Closing thought

TrackMe is never finished. The platform keeps growing, the customer keeps growing, and the adoption journey is iterative — most customers revisit earlier phases as their estate evolves (new business units, acquisitions, new data sources, new Splunk deployments to monitor remotely). The six phases above are a **map**, not a **ladder**, and the right answer to "where am I on the roadmap?" is almost always **"in several places at once"**.

If you are unsure where to focus next, the shortest path is:

1. Run the **gap-analysis checklist** at the bottom of *FLX (Flex Objects) Use Case Template Catalog*.
2. Scan the *TrackMe Use Cases — Organised by User Intent* for goals you have not yet addressed.
3. Cross-check the *TrackMe Capability × Component Matrix* for features you could enable on components you already use.

4. Reach out to us — the periodic teach-back conversation is exactly what this page is designed to support.

CHAPTER

THREE

ARTIFICIAL INTELLIGENCE:

3.1 Artificial Intelligence in TrackMe

3.1.1 Overview

TrackMe 2.4.0 makes **TrackMe Agentic** generally available — a family of AI capabilities that help operators investigate, tune, and remediate their Splunk data monitoring directly inside the product. Where TrackMe has always *detected* data availability and quality problems, the AI layer now helps you *understand* and *resolve* them, grounding every answer in the live state of your environment and keeping the operator firmly in the loop.

The AI layer is organised around **five surfaces**, all built on the same Splunk Agent SDK (the `splunklib.ai` module shipped with the Splunk Enterprise SDK for Python) runtime, the same provider abstraction, and the same audit trail:

- **AI Assistant** — the conversational front door. A context-aware chat assistant available on every entity, page and feature of TrackMe (*AI Assistant*).
- **AI Routines — your own standing instructions**: user-authored, scheduled, natural-language AI tasks that watch for a condition and then notify you or — with explicit, granular consent — act (*AI Routines*).
- **AI Advisors** — five specialists that *inspect* and *remediate* a specific monitoring surface (ML models, feed thresholds, FLX thresholds, field-quality dictionaries, component health), plus the **AI Concierge**, a catalog-driven generalist for the long tail of operations (*The AI Advisors*, *The AI Concierge*).
- **AI Agents automation** — the same advisors running unattended on a nightly schedule, scoped by per-tenant policy (*AI Agents automation*).
- **AI Agents for findings** (2.4.19) — an agentic operator for the dedicated findings tenants: the **UAM AI Agent**, the **Splunk HEC AI Agent**, the **Splunk S2S AI Agent**, the **Splunk UF Fleet AI Agent**, the **Splunk Forwarder Collectors AI Agent**, the **Cribl Stream AI Agent** and the **Cribl Edge AI Agent** share one framework. They triage findings the way an operator would, on demand or in an hourly scheduled triage, recommend a decision and, when the tenant allows it, apply it under their own **AI Agent** identity (*AI Agents for findings*).

Every action is **trustworthy, auditable, and traceable** by design: recommendations are typed and grounded in real data, writes are RBAC-checked at the REST boundary, and every change an agent makes is recorded both in the per-entity audit trail and in dedicated audit dashboards (*Auditing AI activity*).

Important

Opt-in only — The entire AI layer is completely inert until an administrator explicitly configures at least one AI provider. No data is sent to any external service and no LLM calls are made until you opt in. You retain full control over which provider is used and where data is processed — from fully air-gapped on-premises models (Ollama) to Splunk's own hosted LLMs on Splunk Cloud. See *Configuring AI Provider accounts*.

3.1.2 Requirements & compatibility

TrackMe remains **fully compatible** with all supported Splunk releases — the AI layer is additive and opt-in. The Python / Splunk requirements differ by capability:

Capability	Minimum Python	Minimum Splunk
AI Assistant (chat) and *AI Concierge*	Python 3.9.x	Any Splunk release supported by TrackMe
AI Advisors (ML, Feed Lifecycle, FLX Threshold, FQM, Component Health) and *AI Agents automation*	Python 3.13.x	Splunk 10.2.x and later (Enterprise or Cloud)
AI Routines (user-authored scheduled AI tasks)	Python 3.13.x	Splunk 10.2.x and later (Enterprise or Cloud)
AI Agents for findings (every findings tenant)	Python 3.13.x	Splunk 10.2.x and later (Enterprise or Cloud)

Important

The AI Advisors are *agentic* features built on the Splunk Agent SDK, which requires **Python 3.13.x** — available only from **Splunk 10.2.x** (Splunk Enterprise or Splunk Cloud). TrackMe enforces this automatically: an Advisor cannot be launched on an incompatible release (a clear message explains the requirement), and the AI Assistant will not propose an Advisor when it detects an incompatible system. The **AI Assistant** and the **AI Concierge** run on **Python 3.9.x and later**, so they remain available on previous Splunk releases.

In short: TrackMe stays fully compatible everywhere; the advanced AI Advisors require Splunk 10.2.x and later. See the *compatibility page* for the full matrix.

3.1.3 The AI Journey

TrackMe's AI capabilities were built in layers, and they read best in that order. The AI Assistant was *level 1* of the journey — a conversational surface that understands your environment. The Advisors and the Concierge are *level 2* — agents that act, with consent. AI Agents automation is *level 3* — those same agents operating unattended within guardrails you define. **AI Routines are level 4** — you author your *own* autonomous tasks in natural language: describe what to watch for, and TrackMe evaluates it on your schedule, notifies you, or acts under the consent you granted (*AI Routines*). Alongside these levels, TrackMe 2.4.19 brings the **AI Agent** to the dedicated findings tenants: an agentic operator that triages findings, applies the decisions the tenant allows and logs every one of them (*AI Agents for findings*).

The slides below present the architecture and the core concepts. Each tab focuses on a single idea; the detailed walkthroughs (with product screenshots) live in the dedicated pages of this section.

The big picture

One product, three AI surfaces — each with the operator firmly in the loop.

The **AI Assistant** is the conversational front door on every entity, tenant and feature page. The **AI Advisors** are five specialists that inspect and, with consent, act. The **AI Concierge** is the catalog-driven generalist for everything else — backed by the live TrackMe REST API. Same Splunk Agent SDK underneath, same provider abstraction, same audit surface.



What the Advisors do

Four capabilities, one autonomous loop — applied across DSM, DHM, FLX, WLK/MHM and FQM, and to VOL through the ML Advisor.

Every advisor follows the same loop: **Observe** (read entity context, metric history, scoring, audit trail), **Diagnose** (classify the evidence against a typed set of scenarios — no free-text guesses), **Recommend** (produce structured, reviewable recommendations), and — in *act* mode — **Remediate** (apply the fix through TrackMe's own REST API, recording every write to the audit trail).



Six advisors

Five specialists plus the Concierge.

Each specialist owns a curated tool registry scoped to its monitoring surface — **ML Advisor** (outlier model integrity), **Feed Lifecycle** (DSM/DHM thresholds), **FLX Threshold** (FLX dynamic thresholds), **Component Health** (WLK + MHM), and **FQM Advisor** (field quality and data dictionary). The **AI Concierge** is catalog-driven and consent-gated. They share one orchestration library, the same MCP tool pattern, and the same Pydantic output contract. The seven dedicated findings tenants add their own AI Agent each, from the **UAM AI Agent** to the **Splunk Forwarder Collectors AI Agent**, on the same foundation (*AI Agents for findings*).



Architecture

One shared control plane.

The Assistant and the chat bridge sit on the entry side; the Advisors and the Concierge dispatch through the same SDK runtime. A REST handler launches a daemon thread that runs the `splunklib.ai.Agent` (LangGraph) loop against either MCP tools (specialists) or the live API catalog (Concierge). RBAC is enforced at every REST boundary, concurrency is capped, every run is bounded with retries, and every outcome is audited end-to-end.



Built on the Agent SDK

We stand on the SDK Splunk publishes, and share it across every AI surface.

The advisors are built on the Splunk Agent SDK (`splunklib.ai.Agent`) rather than direct LLM API calls: it ships the LangGraph orchestration engine, the MCP tool registry, structured output via Pydantic, and middleware hooks for tracing and limits. The provider abstraction means the same advisor code runs across OpenAI, Azure OpenAI, Anthropic, Google, Mistral, Ollama, OpenAI-compatible endpoints and Splunk Hosted — **no provider lock-in**.



Inspect or Act

Tag-filtered tool sets, same binary — a different toolbox presented to the LLM.

Every specialised advisor honours a two-phase contract. **Inspect** mode is read-only and always safe to run: 5–7 read tools, a smaller token/step budget, structured recommendations only, *no* writes — the write tools are literally absent

from the agent's toolbox. **Act** mode adds the write tools (11–18), a larger budget, and a hard rule that every write is verified by a follow-up read; each write is prefixed `[AI Agent]` and lands in the audit trail.



The Assistant ↔ Advisor bridge

How the chat learns about the advisors, sees recent runs, and proposes a one-click consent card.

Each describe payload carries two channels: static `knowledge_reference` (what each advisor is, its modes, gates and contract) and dynamic `ai_advisor_recent_runs` (RBAC-filtered recent runs). When the user describes a symptom, the assistant emits a typed `advisor_invocation` contract that the chat renders as a consent card. **The chat never directly calls a write tool —** it always proposes an advisor (or Concierge) invocation, and the user's click is the structural safety property.



The AI Concierge

Catalog-grounded, consent-gated, zero static write tools — the long tail without the foot-guns.

The Concierge grounds itself in the live TrackMe REST API catalog (~423 endpoints) and proposes structured action contracts that the chat consent card renders for explicit, per-action approval. It has **zero static write tools by design**: every proposed path is validated against the live catalog, destructive actions require typed confirmation, and the user's confirmation click is what fires the REST call — through their own credentials, RBAC-checked.



Structured output & guardrails

Three pillars — Trust, Auditability, Traceability — built into the agent runtime, not bolted on after.

Trust: typed decisions, not free text — every agent emits a Pydantic envelope (`entity_status` / `summary` / `recommendations` / `actions_taken` / `reasoning_trace`); invalid completions are retried or rejected by the SDK. **Auditability**: runtime guardrails enforce the blast radius — tag-filtered write gate, decommission guard, per-mode tool selection. **Traceability**: every step is indexed and queryable, and every entity write carries the `[AI Agent]` prefix so a human can see exactly what changed and why.



Governance

One menu in TrackMe consolidates AI configuration and observability for operators.

The **Artificial Intelligence** menu brings together everything an operator needs: *Manage AI providers* (create and test provider accounts), *Manage AI tenant configuration* (per-tenant master switches, per-advisor enablement, the decommission guard — disabled by default everywhere), and the two audit dashboards for **AI Advisor** and **AI Assistant** activity. Operators own the controls, everything is disabled by default, and all activity is indexed to the summary index and queryable in plain SPL.



3.1.4 The three pillars

Every design decision across the AI layer traces back to three principles:

Pillar	What it means in practice
Trust	The agent never makes up entity state. Every claim traces to a tool call, and the system prompts forbid speculation. Outputs are *typed* — a Pydantic schema forces a structured envelope rather than free-form prose, and invalid completions are retried or rejected by the SDK. Read-only **inspect** mode is the default for unfamiliar entities.
Auditability	Runtime guardrails — not soft prompts — enforce the blast radius: a tag-filtered write gate (write tools are absent in inspect mode), a decommission guard (disabled by default), and per-mode tool selection. Hard runtime caps (token, step, and wall-clock budgets) are enforced by SDK middleware and a watchdog, so a misbehaving run hits a real wall.
Traceability	Every tool call and every advisor run is indexed to the tenant summary index (`trackme:ai_agent:*`), and every entity write carries an `[AI Agent] <reason>` prefix in the per-entity audit trail. You can answer "which conversation triggered this change, and why?" with a single SPL query.

3.1.5 What's in this section

This **Artificial Intelligence** section walks the full journey, with product screenshots at each step:

Page	What it covers
Configuring AI Provider accounts	Configuring one or more AI provider accounts (OpenAI, Azure, Anthropic, Gemini, Mistral, Ollama, Custom, Splunk Hosted) and the global AI settings. **Start here.**
AI Assistant	The conversational AI Assistant — capabilities, opening it from any page, the chat interface, context enrichment, AI status reports in alert emails, and troubleshooting.
AI Routines	**AI Routines** — describe a monitoring task in plain English; TrackMe watches for it on a schedule, then notifies you or (with consent) acts. Templates, the Evaluate dry-run gate, permission families, cost transparency, and auditing.
The AI Concierge	The AI Concierge — the catalog-driven generalist reached from the Assistant for operational requests, with consent cards and validation gates.
The AI Advisors	The AI Advisor family — inspect/act, the consent-card bridge, and the FLX Threshold Advisor.
AI Feeds Advisor	The AI Feeds Advisor (Feed Lifecycle) — DSM/DHM threshold tuning.
AI ML Advisor	The AI ML Advisor — outlier model inspection and remediation.
AI FQM Advisor	The AI FQM Advisor — field-quality thresholds and data-dictionary management.
AI Component Health Advisor	The AI Component Health Advisor — WLK and MHM health.
AI Agents automation	AI Agents automation — running the advisors unattended on a nightly schedule, scoped by per-tenant policy.
AI Agents for findings	AI Agents for findings — the AI Agents of UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge: how they triage findings, inspect and act, the decisions they may take, scheduled triage, settings and the decision log.
Auditing AI activity	Auditing AI activity — the AI Advisor and AI Assistant audit dashboards.
AI Compliance, Privacy and Data Governance	AI compliance, privacy and data governance — the reference for compliance, security and privacy teams: data flows, what is (and is not) sent to the LLM, the complete tool inventory, guardrails, RBAC, audit, and egress requirements.

3.2 Configuring AI Provider accounts

3.2.1 Overview

Every TrackMe AI capability — the *AI Assistant*, the *AI Advisors*, the *AI Concierge*, *AI Agents automation*, and the AI status report in stateful-alert emails — runs through an **AI provider account**: a stored connection to an LLM endpoint. This is the foundational step of the AI journey.

You can configure **one or more** provider accounts (for model comparison, cost optimisation, redundancy, or data-residency reasons), and TrackMe is **provider-agnostic** — the same AI code runs across OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral AI, Ollama, any OpenAI-compatible endpoint, and Splunk's own hosted models.

Important

Opt-in only — Until at least one provider account is configured, the entire AI layer is completely inert: no data is sent to any external service and no LLM calls are made. The choice of provider is yours — from fully air-gapped on-premises models to managed cloud services — and no data is ever shared between configured providers.

Tip

Full control over your AI infrastructure — Organisations with strict data privacy, data sovereignty, or regulatory requirements retain complete control over which AI provider is used and where data is processed. TrackMe can be connected to:

- **Splunk's own hosted LLMs** via the Splunk Hosted (SLIM API) provider type on Splunk Cloud, with zero configuration beyond selecting the provider and model — no API keys, no base URL, fully managed by the Splunk platform
- **Self-hosted open-source LLMs** via Ollama, keeping all data entirely on-premises with zero external API calls
- **Private or corporate LLM deployments** via the Custom (OpenAI-compatible) provider type, enabling integration with any internally hosted endpoint that implements the OpenAI chat completions format
- **Cloud providers of your choice**, including region-specific deployments (e.g. Azure OpenAI in a specific geography) to meet data residency requirements

3.2.2 Managing provider accounts

In TrackMe 2.4.0, provider accounts are managed from a dedicated page reachable from the top navigation: **Artificial Intelligence > Manage AI Provider accounts**.

The page lists every configured provider account with its provider type, enabled state, model, base URL, and last connectivity test result. From here you can **Add provider**, **Test all**, and — from the per-row actions menu — **Edit**, **Clone**, **Test connectivity**, or **Delete** an account.





When you click **Add provider**, the **Add AI provider** modal opens. Give the account a unique name, pick the provider backend, and fill in the fields that apply to that backend — only the relevant fields are shown for the selected provider. The example below configures an Anthropic (Claude) account using the `claude-haiku-4-5` model; the **Pick a suggested model** dropdown shows common values for convenience.


Add AI provider
IDENTITY
Name
claude_haiku
Enabled
Yes
PROVIDER CONNECTION
Provider
Anthropic (Claude)
Base URL
https://api.anthropic.com/v1
API key
Model
claude-haiku-4-5
Pick a suggested model...
Cancel
Create provider


Virtual Tenants
Configuration
Artificial Intelligence
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
Manage AI Provider accounts
Test all
Add provider
NAME
PROVIDER
ENABLED
MODEL
BASE URL
CONNECTIVITY
claude_haiku
Anthropic (Claude)
Yes
claude-haiku-4-5
https://api.anthropic.com/v1
Not tested
Edit
Clone
Test connectivity
Delete

Note

The account **Name** is the unique stanza identifier (lower-case letters, digits and underscores only) and cannot be changed after creation. Disabled accounts stay configured but are hidden from the AI Assistant provider selector and skipped by the stateful-alert AI status report.

3.2.3 Provider Fields

Field	Default	Description
AI Provider	OpenAI	The AI provider type. Supported values: `OpenAI`, `Azure OpenAI`, `Anthropic (Claude)`, `Google Gemini`, `Mistral AI`, `xAI (Grok)`, `Ollama (local / self-hosted)`, `Custom (OpenAI-compatible)`, `Splunk Hosted (SLIM API)`.
Base URL	*(optional)*	The base URL for the AI provider API. Required for all providers except **Splunk Hosted**, which auto-discovers its endpoint from the Splunk Cloud platform. See the *Supported Providers* section below for provider-specific URLs.
Azure API Version	2024-10-21	Required for Azure OpenAI only. The API version to use (e.g. `2024-10-21`). See Azure OpenAI documentation for available versions.
API Key	*(optional)*	The API key for authenticating with the AI provider. Stored encrypted via Splunk `storage_passwords` and cannot be retrieved again once saved. Leave empty for providers that do not require authentication (e.g. Ollama, Splunk Hosted).
Model	gpt-4o	The model name to use. Some providers support version aliases (e.g. `gpt-4o`, `mistral-large-latest`) that always target the latest release. See *Model Aliases* below.
Max Tokens	4096	Maximum number of tokens in the AI response. Higher values allow longer responses but increase cost and latency.
Temperature	0.3	Controls randomness in AI responses. Lower values (e.g. 0.1) produce more deterministic outputs, higher values (e.g. 0.8) produce more creative outputs.
Request Timeout	600	Maximum time in seconds to wait for an AI provider response. The default of 600 seconds (10 minutes) accommodates slow models or initial model downloads (e.g. Ollama pulling a model on first use).
Context Window (Ollama)	8192	Context window size in tokens for Ollama models. Must be large enough to accommodate the system prompt, context data, and conversation history. Only applies to the Ollama provider. Increase this value if context is being truncated.
Custom Prompt	*(empty)*	Optional additional instructions appended to the system prompt. Use this to customise the AI assistant's behaviour, add domain-specific context, or enforce response formatting preferences.

Tip

Use **Test connectivity** (per-row) or **Test all** after creating an account to validate the endpoint, credentials, and model name before relying on the provider. The test calls `POST /trackme/v2/ai/admin/test` under the hood.

3.2.4 Supported Providers

Named Providers

The following providers are explicitly supported with dedicated dropdown options. All use the OpenAI chat completions API format except Anthropic (which uses its native Messages API) and Splunk Hosted (which uses the SLIM API with automatic SCS authentication).

OpenAI

- **Provider**: `OpenAI`
- **Base URL**: `https://api.openai.com/v1`
- **API Key**: Required (from platform.openai.com)
- **Example models**: `gpt-4o`, `gpt-4o-mini`, `gpt-4.1`, `o4-mini`

Azure OpenAI

- **Provider**: `Azure OpenAI`
- **Base URL**: `https://<resource-name>.openai.azure.com/openai/deployments/<deployment-name>`
- **API Key**: Required (Azure portal)
- **Azure API Version**: Required (e.g. `2024-10-21`)
- **Example models**: Use the deployment name configured in Azure

Note

Azure OpenAI uses a different authentication mechanism (`api-key` header instead of `Authorization: Bearer`) and requires a mandatory `api-version` query parameter. TrackMe handles both automatically when the Azure provider type is selected.

Anthropic

- **Provider**: `Anthropic (Claude)`
- **Base URL**: `https://api.anthropic.com/v1`
- **API Key**: Required (from console.anthropic.com)
- **Example models**: `claude-sonnet-4-5-20250929`, `claude-sonnet-4-20250514`, `claude-haiku-4-5`, `claude-opus-4-6-20260213`

Note

Anthropic does not support `-latest` aliases. Use either generation-specific aliases (e.g. `claude-sonnet-4-5`) which auto-update within a generation, or pinned snapshot IDs (e.g. `claude-sonnet-4-5-20250929`) for deterministic behaviour. See Anthropic's model documentation for the full list.

Note

Anthropic uses a native Messages API rather than the OpenAI chat completions format. TrackMe automatically handles the API differences when the Anthropic provider type is selected.

Google Gemini

- **Provider**: `Google Gemini`
- **Base URL**: `https://generativelanguage.googleapis.com/v1beta/openai`
- **API Key**: Required (from Google AI Studio)
- **Example models**: `gemini-3-flash-preview`, `gemini-3-pro-preview`, `gemini-2.5-flash`, `gemini-2.5-pro`

Note

Google Gemini uses its OpenAI-compatible endpoint for integration. The API key is passed as a Bearer token in the Authorization header, which TrackMe handles automatically. Gemini 3 models are currently in preview and use the `-preview` suffix in their model names.

Mistral AI

- **Provider**: `Mistral AI`
- **Base URL**: `https://api.mistral.ai/v1`
- **API Key**: Required (from console.mistral.ai)
- **Example models**: `mistral-small-latest`, `mistral-medium-latest`, `mistral-large-latest`

Ollama

- **Provider**: `Ollama (local / self-hosted)`
- **Base URL**: `http://<ollama-host>:11434/v1`
- **API Key**: Not required
- **Example models**: `mistral`, `llama3.2`, `phi3`, `qwen2.5`

Hint

Ollama is a self-hosted inference server that provides full data privacy and control over your AI infrastructure. It runs powerful open-source models without sending any data to external APIs, making it ideal for environments with strict data governance requirements or air-gapped networks. On first use with a new model, TrackMe automatically triggers the model download on the Ollama server, which can take several minutes. The default request timeout of 600 seconds accommodates this.

Splunk Hosted (SLIM API)

- **Provider**: `Splunk Hosted (SLIM API)`
- **Base URL**: Leave empty (auto-discovered)
- **API Key**: Not required (auto-managed)
- **Example models**: `gpt-oss-120b`, `gpt-oss-20b`, `foundation-sec-1.1-8b-instruct`

Important

The Splunk Hosted (SLIM API) provider is **only available on Splunk Cloud** instances where Splunk has enabled the SLIM (Splunk Language and Intelligence Model) API. It is not available on Splunk Enterprise (on-premises) deployments.

How it works:

Unlike all other provider types, Splunk Hosted requires no manual credential or endpoint configuration:

- **No Base URL**: The SLIM API endpoint is automatically discovered from the Splunk Cloud platform using the SCS (Splunk Cloud Services) tenant information. Leave the Base URL field empty.
- **No API Key**: Authentication uses short-lived SCS tokens that are transparently obtained and refreshed by TrackMe at each request. No API key needs to be entered or managed.
- **Model selection**: You only need to choose a model. Available models can be discovered via the TrackMe REST API (see below).

Discovering available models:

You can query the available SLIM API models using the TrackMe REST API:

```
| trackme url="/services/trackme/v2/ai/admin/models" mode="get"
```

This returns the list of models currently available on your Splunk Cloud instance, including their `model_name` (display name) and `model_id` (the value to enter in the Model configuration field).

Hint

About the models — `gpt-oss-120b` is a ~117B parameter Mixture of Experts model suitable for general-purpose tasks requiring high quality. `gpt-oss-20b` is a smaller ~21B parameter model offering faster responses at lower cost. `foundation-sec-1.1-8b-instruct` is a security-focused model designed for cybersecurity analysis and security operations use cases. Model availability may vary depending on your Splunk Cloud deployment and entitlements.

Custom (OpenAI-Compatible)

The **Custom** provider type works with any API endpoint that implements the OpenAI chat completions format (`POST /chat/completions`). This includes a wide range of enterprise and open-source providers:

Provider	Base URL	Example Models
xAI (Grok)	`https://api.x.ai/v1`	`grok-3`, `grok-3-mini`
Together AI	`https://api.together.xyz/v1`	`meta-llama/Llama-3-70b-chat-hf`
Groq	`https://api.groq.com/openai/v1`	`llama-3.3-70b-versatile`, `mixtral-8x7b-32768`
Perplexity	`https://api.perplexity.ai`	`sonar-pro`, `sonar`
Cohere	`https://api.cohere.com/compatibility/v1`	`command-r-plus`, `command-r`
AWS Bedrock	`https://bedrock-runtime.<region>.amazonaws.com/openai/v1`	`anthropic.claude-sonnet-4-20250514-v1:0`
DeepSeek	`https://api.deepseek.com/v1`	`deepseek-chat`, `deepseek-reasoner`
OpenRouter	`https://openrouter.ai/api/v1`	Any model available on OpenRouter
NVIDIA NIM	`http://<nim-host>:8000/v1`	Depends on deployed model
LM Studio	`http://<lmstudio-host>:1234/v1`	Depends on loaded model

Note

When using the Custom provider type, ensure that the API endpoint is reachable from the Splunk search head running TrackMe and that it supports the standard OpenAI chat completions request and response format.

Note

A note on GitHub Copilot — GitHub Copilot is not a supported AI provider. Although community projects exist that proxy Copilot's internal endpoint (`api.githubcopilot.com`) as an OpenAI-compatible API, this endpoint is not an officially supported public API. It is designed exclusively for GitHub's own IDE extensions (VS Code, JetBrains, CLI), and using it through third-party applications violates GitHub's Terms of Service. Relying on it would expose your organisation to account suspension risk and potential breakage without notice. If you are in the Microsoft/GitHub ecosystem, use **Azure OpenAI** or **OpenAI** directly instead — these provide official, stable API access to the same underlying models that power Copilot.

Model Aliases

Some providers support **version aliases** that always point to the latest release of a model family. Using aliases avoids having to update the configuration when a new model version is released:

- **Mistral**: `mistral-small-latest`, `mistral-medium-latest`, `mistral-large-latest` — auto-update to the newest version within each model tier.
- **OpenAI**: `gpt-4o`, `gpt-4o-mini` — effectively aliases that point to the latest patch.
- **Anthropic**: Does **not** support `-latest` aliases. Use generation-specific aliases (e.g. `claude-sonnet-4-5`) or pinned snapshot IDs (e.g. `claude-sonnet-4-5-20250929`).

Check your provider's documentation for available aliases and version pinning options.

Multi-Provider Setup

TrackMe supports configuring **multiple AI providers** simultaneously. This is useful for:

- **Model comparison**: Try different models on the same question to compare analysis quality
- **Cost optimisation**: Use a cheaper model for routine questions and a premium model for complex investigations
- **Redundancy**: Fall back to an alternative provider if the primary is unavailable
- **Data privacy**: Run Ollama alongside a cloud provider — use the self-hosted option for sensitive environments and the cloud provider for general use
- **Platform-native option**: On Splunk Cloud, use Splunk Hosted alongside external providers — keep data within the Splunk platform while having an external model available as an alternative

To set up multiple providers, create multiple accounts on the **Manage AI Provider accounts** page, each with a descriptive name (e.g. `openai_gpt4o`, `ollama_mistral`, `anthropic_claude`).

When multiple providers are available, users see a provider selector dropdown in the AI chat panel that displays each provider's name and model, and the AI Advisors and Concierge let you pick which provider to run with.

3.2.5 Global AI settings

Beyond provider accounts, a small set of global toggles live under **Configuration > General**, in the **Artificial Intelligence** group. These control the AI Assistant feature switch, the concurrency cap, and data anonymisation.

Setting	Default	Description
Enable AI assistant	Yes	Enable or disable the AI assistant feature across all TrackMe UIs. When disabled, the AI assistant button is hidden from all users.
Max concurrent AI chats	10	Maximum number of concurrent AI chat requests across all users. Additional requests are rejected with an HTTP 429 response until a slot becomes available.
Anonymize entity names for AI	No	When enabled, entity names (object and alias fields) are anonymized using SHA256 hashing in describe endpoint responses consumed by AI providers. This protects potentially sensitive infrastructure details (hostnames, IPs, machine names) while preserving the object_id for entity identification. The AI assistant is guided to use object_id references instead of entity names.
Anonymize index names for AI	No	When enabled, Splunk index names are anonymized using SHA256 hashing in describe endpoint responses consumed by AI providers. This applies to data source (DSM), data host (DHM), and metric host (MHM) entities. Index names are replaced with their SHA256 hashes in entity information, investigation searches, and metric details. Investigation searches will contain hashed index names that users must replace with actual values before execution.

Note

The global `Enable AI assistant` toggle is enabled by default so that administrators can discover the feature, but — as stated above — no AI functionality is active until at least one provider account is configured.

3.2.6 Next steps

With at least one provider account configured, you are ready to use the rest of the AI layer:

- *AI Assistant* — the conversational assistant.
- *The AI Advisors* — the AI Advisor family (inspect / act).
- *The AI Concierge* — the catalog-driven AI Concierge.
- *AI Agents automation* — running the advisors unattended.

3.3 AI Assistant & Concierge

TrackMe's conversational surfaces: the **AI Assistant** answers questions and guides remediation in context, and the **AI Concierge** turns natural-language intent into the right TrackMe action.

3.3.1 AI Assistant

Overview

The TrackMe AI Assistant is an integrated, context-aware AI chat assistant available across the entire TrackMe product. It helps users investigate entity health issues, manage operational workflows, and get instant guidance on any TrackMe feature — all directly within the Splunk interface.

Note

Compatibility — The AI Assistant is compatible with **Python 3.9.x and later**, so it is available on the same Splunk releases as the rest of TrackMe. The advanced *AI Advisors* additionally require Splunk 10.2.x and later (Python 3.13.x); see *Artificial Intelligence in TrackMe* (Requirements & compatibility) and the *compatibility page*.

Capabilities

Three pillars of AI-powered assistance across TrackMe.

The AI Assistant covers entity investigation, operational guidance across all TrackMe features, and AI-powered email notifications — all from a single, context-aware chat interface.



Entity Investigation

Open the AI Assistant from any entity panel to get instant, context-aware analysis.

The assistant explains why an entity is in alert state, analyses lag metrics, event counts, and ML results, provides actionable SPL investigation searches, and suggests threshold adjustments with full context. Covers DSM, DHM, MHM, VOL, FLX, FQM, and WLK entities.



Context Is Everything

The real power of TrackMe's AI isn't the model — it's the context.

TrackMe's describe API endpoints feed rich, structured data to the LLM, so it understands the exact state of your environment without the user having to explain anything. Context types include entity investigation, virtual tenants, REST API reference, backup & restore, maintenance mode, and license management.



Context-Aware Intelligence

The AI automatically collects structured data from the page you're on — no need to describe the situation.

Suggested questions adapt to each page context, and the assistant provides targeted analysis whether you're investigating an entity alert, exploring ML Outliers anomalies, or navigating the REST API.



Provider Agnostic

Connect to the LLM provider that fits your organisation.

Configure multiple providers for comparison, cost optimisation, or redundancy. Supported providers include OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral AI, Ollama, Splunk Hosted (SLIM API), and any OpenAI-compatible endpoint.



Security & Privacy

Security and privacy by design — full control over your AI infrastructure.

Opt-in only (completely inert until configured), RBAC enforced (users only access data they're authorised to view), credential isolation (API keys stored encrypted, never exposed to end users), and optional data anonymisation (SHA256 hashing of entity and index names).



Email Notifications

Stateful alert emails can include an AI-generated status summary, adapting to the incident lifecycle.

The AI report explains what went wrong on incident open, analyses changes on update, and confirms recovery on close. Uses the same provider configuration, operates fail-open (never blocks email delivery), and is capped at 120 seconds to avoid delays.



Architecture

Asynchronous, streaming architecture with built-in security controls.

The flow proceeds in four steps: user opens chat from any TrackMe page, describe API collects structured entity data, a background thread streams the LLM call, and tokens arrive in real-time to the UI. RBAC enforcement, concurrency control, encrypted credentials, and automatic session cleanup are built in.



Important

Opt-in only — The AI Assistant is entirely opt-in. While the feature toggle is enabled by default so that administrators can discover it, no AI functionality is active, no data is sent to any external service, and no LLM calls are made until an administrator explicitly configures at least one AI provider. Without a configured provider, the feature is completely inert.

Tip

Full control over your AI infrastructure — Organisations with strict data privacy, data sovereignty, or regulatory requirements retain complete control over which AI provider is used and where data is processed. TrackMe can be connected to:

- **Splunk's own hosted LLMs** via the Splunk Hosted (SLIM API) provider type on Splunk Cloud, with zero configuration beyond selecting the provider and model — no API keys, no base URL, fully managed by the Splunk platform
- **Self-hosted open-source LLMs** via Ollama, keeping all data entirely on-premises with zero external API calls
- **Private or corporate LLM deployments** via the Custom (OpenAI-compatible) provider type, enabling integration with any internally hosted endpoint that implements the OpenAI chat completions format
- **Cloud providers of your choice**, including region-specific deployments (e.g. Azure OpenAI in a specific geography) to meet data residency requirements

The choice of provider is yours — from fully air-gapped on-premises models to managed cloud services — and no data is ever shared between configured providers.





Configuration

Before the AI Assistant can be used, an administrator must configure at least one **AI provider account** and (optionally) review the global AI settings. Both are covered in their own page:

- *Configuring AI Provider accounts* — create and test provider accounts (OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral AI, Ollama, Custom, Splunk Hosted), and the global `Enable AI assistant`, concurrency, and anonymisation settings.

Important

At least one AI provider account must be configured for the AI Assistant to function. If no provider is configured, users see an informative error message when they attempt to open the AI chat panel. See *Configuring AI Provider accounts*.

Usage

Opening the AI Assistant

The AI Assistant is available throughout TrackMe via two patterns depending on the context.

From Entity Investigation

When investigating a specific entity (DSM, DHM, MHM, VOL, FLX, FQM, WLK), the AI Assistant can be opened via:

Entity overview panel:

1. Open an entity investigation panel by clicking on an entity in the monitoring dashboard
2. Click the **AI Assistant** button in the panel header
3. The AI chat panel opens as a full-screen slide panel with the entity context pre-loaded

Entity actions menu:

1. Click the three-dot actions menu (kebab menu) on any entity row in the monitoring dashboard
2. Select **Ask AI**
3. The AI chat panel opens directly with the entity context pre-loaded





From TrackMe UI Pages

Each TrackMe UI page features an **AI assistant** button in the top-right header. Clicking it opens the AI chat panel with the relevant context for that page automatically loaded.

The AI Assistant is available from the following pages:

- **Virtual Tenants** — context includes all tenants, their health states, and configuration
- **Tenant Home** — context includes tenant-specific entities and component overview; on a dedicated findings tenant, its workspace, policies and open findings
- **REST API Reference** — context includes the full API catalogue, resource groups, and usage patterns
- **Backup & Restore** — context includes backup records, status, and scheduling information
- **Global Maintenance Mode** — context includes current maintenance status, scheduling, and countdown
- **Maintenance Knowledge Database** — context includes KDB records and per-entity maintenance windows
- **Bank Holidays Management** — context includes the calendars and their rules, active and upcoming holidays, and the default scope
- **License Management** — context includes edition, validity, expiration, and feature information

Note

The AI assistant button is only visible when the feature is enabled in the General settings and at least one AI provider is configured.

Chat Interface

The AI chat panel provides a conversational interface with the following features:

Suggested questions: When the chat panel opens, context-specific suggested questions are displayed to help get started. The questions are tailored to the page the assistant was opened from. For example, entity investigation suggests questions about health state and thresholds, while the Backup & Restore page suggests questions about backup status and restore procedures.

Free-form questions: Type any question in the input field at the bottom of the panel. Press **Enter** to send (use **Shift+Enter** for a newline).





Streaming responses: AI responses stream in real-time as tokens are generated, providing immediate feedback.

Conversation context: The AI Assistant maintains conversation history within the session. Follow-up questions are understood in the context of previous messages.

Markdown rendering: AI responses support full Markdown formatting including code blocks, tables, lists, and inline formatting. SPL searches are displayed in syntax-highlighted code blocks for easy copy-paste.

Provider switching: If multiple providers are configured, use the dropdown at the top of the chat panel to switch between providers. Switching providers clears the conversation history.

Clear chat: Use the trash icon button to clear the conversation and start fresh.

Worked examples

The simplest way to experience the AI Assistant is to open it from an entity and ask a plain question. In the example below, the assistant is opened from **Tenant Home** and then on a specific DSM entity; the suggested questions adapt to the entity, and a question such as *"What is the difference between latency and delay in TrackMe? How is it calculated?"* is answered with a precise, context-aware explanation grounded in that entity's live metrics.






Actions for entity: siem-cloud-amer:google:gcp:pubsub:message
AI Assistant — siem-cloud-amer:google:gcp:pubsub:message
Why is this entity in alert state?
Entity Alert Analysis
1. Delay Threshold Breached (Score: 100)
2. Latency Threshold Breached (Score: 48)
Current Configuration
Ask a question about this entity... (Shift+Enter for new line, supports Markdown)
Send


Actions for entity: siem-cloud-amer:google:gcp:pubsub:message
AI Assistant — siem-cloud-amer:google:gcp:pubsub:message
What is the difference between latency and delay in TrackMe? How is it calculated?
Latency vs. Delay in TrackMe
Delay (Event Delay)
Latency (Ingestion Latency)
Ask a question about this entity... (Shift+Enter for new line, supports Markdown)
Send



Because the assistant has the entity's full context, it can also produce **ready-to-run SPL searches** tailored to the question — for example when asking *"What search can I use to investigate latency issues?"*. SPL is rendered in syntax-highlighted code blocks with **Copy to clipboard** and **Execute in Search** buttons, so you can move from question to investigation in one click.





Tip

The AI Assistant is the *front door* to the wider AI layer. When you describe a problem it can resolve, the assistant proposes a one-click **consent card** that hands off to a specialist *AI Advisor* or to the *AI Concierge* — see those pages for the full agentic workflow.

Context Enrichment

When a chat session starts, TrackMe automatically collects structured data relevant to the page the assistant was opened from. This context is passed to the LLM as part of a specialised system prompt, enabling the AI to provide specific, data-driven answers without the user having to describe the situation manually.

The following table summarises what context is collected for each area:

Context	Data Collected
Entity investigation	Health state, scoring, data lag and event count metrics, configured thresholds, ML Outliers detection results, investigation SPL searches, health history, entity metadata (index, sourcetype, host, etc.), and threshold tuning guidance.
Virtual Tenants	All tenants with their health states, component counts, configuration, and license information.
Tenant Home	Tenant-specific entities, health overview, components, and configuration details; on a dedicated findings tenant, the workspace, its policies and its open findings.
REST API Reference	Full API catalogue including resource groups, endpoints, HTTP methods, authentication patterns, SPL `| trackme` command syntax, and the `describe=true` self-documentation pattern.
Backup & Restore	Backup archive records (timestamps, labels, status, type), restore history, and a knowledge reference covering backup procedures and best practices.
Global Maintenance Mode	Current maintenance status (enabled/disabled/scheduled), start and end times, countdown to auto-disable, and a knowledge reference on maintenance impact.
Maintenance Knowledge Database	KDB record counts, recent records sorted by end time (planned and unplanned types), per-entity maintenance windows, and guidance on KDB management.
Bank Holidays Management	Calendars with their rules, active and upcoming holidays, the system defaults, and what a holiday does for a tenant (alerts, SLA, ML training, monitoring windows).
License Management	Current edition, validity and expiration status, feature availability per edition, developer mode status, and renewal workflow guidance.





Entity Threshold Tuning Guidance

The entity investigation context includes a `thresholds_help` section that provides entity-type-specific advisory guidance for threshold tuning. This enables the AI Assistant to give accurate, actionable advice when users ask about adjusting thresholds:

- **DSM / DHM**: Explains event delay, ingestion latency, future tolerance, adaptive delay, variable delay, and lagging class thresholds. DSM entities additionally include host count monitoring guidance.
- **MHM**: Covers metric lag and future tolerance thresholds, as well as the metric category policy.
- **FLX**: Distinguishes between entity-level settings (inactivity timeout) and use case-level threshold conditions defined in the FLX tracker configuration.
- **FQM**: Explains that thresholds are managed at the FQM tracker level (success rate and coverage conditions).
- **WLK**: Explicitly notes that Workload Monitoring entities do not have user-configurable thresholds, and advises investigating execution issues instead.

Session Lifecycle

Panel close and job cancellation: When the AI chat panel is closed (or the user navigates away), the frontend automatically cancels any in-flight AI request. This releases the concurrency slot immediately so other users can use the AI Assistant without hitting the concurrency limit. The backend worker thread may continue briefly until the LLM responds, but the response is discarded.

Custom Prompt

Administrators can use the **Custom Prompt** field in the AI Provider configuration to add domain-specific instructions that are appended to the system prompt. Example use cases:

- Enforce response formatting (e.g. always include a “Next Steps” section)
- Add organisation-specific context (e.g. known maintenance windows, team ownership)
- Customise the assistant’s tone or verbosity
- Include relevant runbook references or escalation procedures

AI Status Report in Stateful Alert Emails

In addition to the interactive chat assistant, the AI infrastructure powers an optional **AI Status Report** feature in stateful alert email notifications. When enabled, each email notification includes an AI-generated entity status summary embedded directly in the HTML email body.

How it works:

1. When a stateful alert triggers and email delivery is required, TrackMe calls the configured AI provider with the entity context (the same describe data used by the interactive chat).
2. The AI generates a concise, actionable summary tailored to the incident lifecycle stage:
   - **Opened**: Explains what went wrong and where to investigate first
   - **Updated**: Analyses whether the situation has changed since the last notification
   - **Closed**: Confirms the recovery and summarises what was previously wrong
3. The AI response is converted to email-safe HTML and embedded in the notification between the detailed information and charts sections.

Key design principles:

- **Same provider configuration**: The AI Status Report uses the same AI provider(s) configured in **Configuration > AI Provider**. No additional AI setup is required.
- **Fail-open**: If the AI provider is unavailable, slow, or returns an error, the email notification is delivered normally without the AI section. No email is ever blocked by an AI failure.
- **Cost-aware**: The AI report is only generated when email delivery is actually needed. If the entity's priority level is filtered out from email notifications by priority routing, no AI call is made.
- **Timeout capped**: The AI request timeout is capped at 120 seconds for email flows to avoid excessive notification delays.

Example of an email notification with the AI Status Report:





Configuration:

The AI Status Report is configured per stateful alert in the alert creation wizard or in the Splunk alert action editor:

- `AI Status Report`: Enable or disable. Defaults to **enabled** when at least one AI provider is configured.
- `AI Provider`: Optionally select a specific provider when multiple are configured. Defaults to the first available provider.

For detailed setup instructions, see *Stateful alerting*.

Troubleshooting:

AI Status Report operations are logged with the `task=ai_status_report` tag in the alert action logs:

```
index=cim_modactions sourcetype=modular_alerts:trackme_stateful_alert
↪"task=ai_status_report"
```

Error Handling

The AI Assistant handles errors gracefully with clear user-facing messages:

Error Type	Description
AI not configured	No AI provider has been configured. An administrator must set up at least one provider in the Configuration page.
AI disabled	The AI assistant has been disabled by an administrator in the General settings.
AI busy	The maximum number of concurrent AI chat requests has been reached. The user should try again in a moment.
API key missing	The AI provider requires an API key but none is configured. This can occur if a provider was created without saving the API key, or if the credential was lost. Re-enter the API key in the AI Provider configuration.
Provider error	The AI provider returned an error (e.g. invalid API key, model not found, rate limit exceeded). The error message from the provider is displayed.

Troubleshooting

Logs

The AI Assistant logs all requests, errors, and streaming activity to dedicated log files on the Splunk search head. Use the following SPL search to access AI-related logs:

```
index=_internal sourcetype="trackme:rest_api"␣
↪source="/opt/splunk/var/log/splunk/trackme_rest_api_ai_*"
```

Key log events to look for:

Log Pattern	Description
`action="request_started"`	A new AI chat request was submitted. Includes the provider, model, base URL, timeout, context type, and whether context was loaded.
`action="first_token"`	The first token was received from the LLM provider. The `time_to_first_token_sec` field indicates provider responsiveness.
`action="streaming_progress`	Periodic heartbeat during long-running streaming responses (every 30 seconds). Shows elapsed time and characters received.
`action="timeout"`	The LLM request timed out. Check the `timeout_sec` and `elapsed_sec` fields. Consider increasing the Request Timeout setting.
`action="stale_job_cleanup"`	A running job was detected as stale (exceeded its timeout plus a grace period). The concurrency slot was released automatically.
`action="slot_released"`	A concurrency slot was released, either after job completion or via panel close cancellation.
`exception=`	An error occurred during the AI request. The exception message typically includes the HTTP status code and error body from the provider.

Common Issues

"Missing Authorization header" from Google Gemini

The API key is not configured or was not saved correctly for this provider. Edit the provider in **Configuration > AI Provider**, re-enter the API key, and save.

"Model not found" (HTTP 404)

The model name is incorrect. Verify the exact model ID in your provider's documentation. Common mistakes include missing version suffixes (e.g. `gemini-3-pro` vs `gemini-3-pro-preview`) or using aliases that don't exist for the provider.

"AI assistant is at maximum capacity"

All concurrency slots are in use. Either wait for an in-flight request to complete, or increase the **Max concurrent AI chats** setting in **Configuration > General**. If slots appear stuck, check for stale jobs in the logs — they are automatically cleaned up after the configured timeout plus a grace period.

Streaming responses appear slow or stalled

For cloud providers, this is typically caused by network latency or provider-side rate limiting. For Ollama, ensure the server has sufficient resources (CPU/GPU/RAM) for the model. Check the `time_to_first_token_sec` in the logs to distinguish between slow initial response and slow token generation.

Context not loaded

If the AI responds with generic advice rather than specific analysis, check the `entity_context_loaded` field in the request log. A value of `False` indicates the context collection failed. For entity investigations, verify that the entity exists in the specified tenant and that the user has RBAC permissions to access it. For other contexts, check that the relevant TrackMe backend services are running correctly.

3.3.2 The AI Concierge

Overview

The **AI Concierge** is the catalog-driven generalist of the TrackMe AI family. Where the five specialised *AI Advisors* handle curated remediation flows for specific surfaces (ML models, DSM/DHM thresholds, FLX thresholds, FQM dictionaries, WLK/MHM health), the Concierge handles the **long tail** of operational requests by grounding itself in the live TrackMe REST API catalog (~423 endpoints) and proposing structured **action contracts** that you approve from a consent card in the chat.

The Concierge is reached through the *AI Assistant*: when you ask the chat to *do* something operational — disable an entity, change a priority, set a maintenance window, trigger a backup, delete a tenant — and that intent does not map to a specialist advisor, the assistant routes the request to the Concierge.

Important

The Concierge has **zero static write tools by design**. It never mutates anything directly: it *proposes* actions, and your confirmation click is what fires the REST call — through your own Splunk credentials, RBAC-checked at the REST boundary. The consent-card click is the structural safety property.

Note

Unlike the specialised *AI Advisors*, the AI Concierge does **not** require Python 3.13.x / Splunk 10.2.x — it runs on **Python 3.9.x and later**, so it is available on the same Splunk releases as the rest of TrackMe. See *Artificial Intelligence in TrackMe* (Requirements & compatibility).

How it works

The Concierge follows a propose-then-confirm contract:

1. **Discover** — it searches the live API catalog for endpoints matching your intent, filtered by surface (entity / tenant_home / vtenants / global) and verb (read / write).
2. **Validate** — every candidate endpoint is checked against the live catalog: the path must exist, the danger level must match, and the body template must carry placeholders for protected fields.
3. **Propose** — it emits a typed `concierge_invocation` action contract carrying one or more proposed actions.
4. **Consent** — the chat renders each action as a card showing the target endpoint, the request body (editable), a danger-level badge, and per-action `Run` / `Edit request body` / `Cancel` controls. Destructive actions require you to type the action name to confirm.
5. **Execute** — your click fires the REST call. Each write is stamped with the `[AI Agent]` prefix in the entity audit trail, exactly like the specialist advisors.

Single-action example

In the example below, the operator is investigating a DSM entity and types a plain-language request into the AI Assistant: *"Disable monitor for this entity, it's a noisy feed and we don't need it anymore"*. The intent does not map to a specialist advisor, so the assistant proposes a **Concierge action** — a consent card that targets the exact REST endpoint with the resolved request body, ready to run or edit.



Once you confirm, the Concierge fires the REST call and reports completion inline, with the structured result of the action it performed.



Multi-action proposals

A single request can produce **several** proposed actions when the intent naturally spans more than one operation. In the example below the Concierge proposes a coordinated set of actions (`ds_monitoring`, `ds_update_priority`, `set_maintenance`), each rendered as its own card with the required capability, the target endpoint, the request body, and an **Edit request body** control. You approve, edit, or cancel each action independently.



Where the Concierge is available

The Concierge is wired into every chat surface as the **universal fallback**. On entity, Tenant Home and Virtual Tenants surfaces it layers *under* the specialists — when your intent matches a specialist (e.g. "why is this entity red?" on a DSM entity), the assistant proposes the specialist; only when no specialist matches does it fall through to a Concierge proposal.

On the six feature-level surfaces the specialists do not apply, so the Concierge is the sole bridge target:

- REST API Reference
- Backup & Restore
- Global Maintenance Mode
- Maintenance Knowledge Database
- Bank Holidays Management

- License Management

Safety and validation

The Concierge operates on the full REST catalog, so it has the broadest surface of any AI capability. Several layers of defence-in-depth keep it safe:

Guardrail	What it does
Catalog grounding	Every proposed path is validated against the live API catalog — both before the contract is emitted and again at consent-card render time. Hallucinated or removed paths disable the **Confirm** button with a clear banner.
Consent is structural	The Concierge cannot mutate anything without an explicit user click. There is no autonomous write path.
Danger badge + typed confirmation	Every action carries a danger level (low / medium / high / critical). `destructive` actions require you to type the action name (e.g. `del_tenant`) before **Confirm** enables.
RBAC at the REST boundary	The action fires through *your* Splunk session; an operation you are not authorised for fails at the REST boundary with a clean 403 surfaced on the card — the agent does not bypass RBAC.
Audited end-to-end	Every Concierge proposal is indexed to the tenant summary index (sourcetype `trackme:ai_agent:concierge_advisor:propose`) and every confirmed write carries the `[AI Agent]` prefix in the per-entity audit trail. See *Auditing AI activity*.

Note

Two operational caps apply per tenant: `ai_concierge_max_actions_per_proposal` (default 10) and `ai_concierge_rate_limit_per_minute` (default 5). The Concierge has **no** per-tenant enablement gate and **no** separate destructive-action gate — RBAC, the consent card, and typed confirmation are already the safety layers.

Related

- *AI Assistant* — the conversational front door the Concierge is reached from.
- *The AI Advisors* — the specialised advisors the Concierge complements.
- *Auditing AI activity* — the audit dashboards that record Concierge activity.

3.4 The AI Advisors

3.4.1 Overview

The **AI Advisors** are TrackMe's agentic surface — specialists that read the same KV Store and search indexes a human analyst would, propose remediations grounded in real configuration and recent telemetry, and (in *act* mode) apply changes through the same REST API a UI click would invoke. Every change is auditable, every write is RBAC-checked at the REST boundary, and every recommendation can be replayed.

There are **six advisors** — five specialists plus the catalog-driven *AI Concierge*:

Advisor	What it handles	Documentation
AI Feeds Advisor (Feed Lifecycle)	DSM / DHM threshold tuning, monitoring state, adaptive delay, variable-delay schedules, impact-score weights	*AI Feeds Advisor*
AI ML Advisor	ML outlier model inspection and remediation (period exclusions, retrains, density-threshold tuning, false-positive clears)	*AI ML Advisor*
AI FQM Advisor	Field-quality thresholds and data-dictionary management (per-field validation, CIM compliance, regex testing); wizard-time dictionary generation	*AI FQM Advisor*
AI Component Health Advisor	WLK (workload scheduled-search health) + MHM (metric-host lag) thresholds	*AI Component Health Advisor*
AI FLX Threshold Advisor	FLX dynamic thresholds (add / update / delete / variable-threshold slots), entity state and priority	*(below)*
AI Concierge	Long-tail operational requests grounded in the live REST API catalog	*The AI Concierge*

Note

The advisors are **opt-in**: they require at least one configured *AI provider account*, and each is **disabled by default** until enabled per tenant (see *AI Agents automation* for the automated path). On-demand interactive runs from the UI and the chat are available to authorised operators as soon as a provider is configured.

Important

The AI Advisors require Splunk 10.2.x and later (Python 3.13.x). The five specialised advisors are built on the Splunk Agent SDK, which needs Python 3.13.x — shipped only with Splunk 10.2.x (Splunk Enterprise or Splunk Cloud). TrackMe enforces this automatically: an advisor cannot be invoked on an incompatible release (a clear message explains the requirement), and the *AI Assistant* will not propose an advisor when it detects an incompatible system. The AI Assistant and the *AI Concierge* do **not** require Python 3.13.x — they run on Python 3.9.x and later, so they remain available on previous Splunk releases. See *Artificial Intelligence in TrackMe* (Requirements & compatibility) and the *compatibility page*.

3.4.2 Inspect or Act — the operational contract

Every specialised advisor honours the same two-phase contract:

Mode	Behaviour
inspect	Read-only — always safe to run. The agent analyses and surfaces structured recommendations, then stops. No KV writes, no REST mutations: the write tools are *literally absent* from the agent's toolbox. This is the default for unfamiliar entities and the first stage of the chat bridge.
act	Full remediation. The agent applies the recommended changes through write tools, then re-reads to confirm. A hard rule requires every write to be verified by a follow-up read, and every write is prefixed `[AI Agent] <reason>` in the entity audit trail.

The two phases give the chat bridge a natural two-step flow: the operator describes the problem → the assistant proposes an **inspect** run → the operator reviews the recommendations and clicks **Run Act** (optionally with extra instructions) to apply them.

Note

Hard runtime caps are enforced by SDK middleware and a shared watchdog, not by prompt text — an *inspect* run has a smaller token / step budget and a 10-minute wall-clock cap; an *act* run has a larger budget and a 15-minute cap. A misbehaving run hits a real wall and is recorded as an error.

3.4.3 Two ways to run an advisor

Standalone modal

Every entity overview exposes an advisor tab appropriate to its component (for example **AI Feeds Advisor** on DSM/DHM, **AI ML Advisor** on the Outliers tab, **AI FQM Advisor** on FQM, **AI WLK Advisor** on WLK). The modal lets you pick the provider and mode, add optional instructions, and launch the run — showing a live tool-call progress feed and then the structured result with status badge, recommendations, actions taken, and a collapsible reasoning trace.

The chat consent-card bridge

The *AI Assistant* knows about the advisors. When you describe a symptom an advisor can resolve, it proposes a typed **advisor invocation** rendered as an inline **consent card** — without leaving the conversation. The chat *never* calls a write tool directly; it always proposes an advisor (or Concierge) invocation, and your click is the structural safety property.

The example below walks the full bridge flow on a DSM entity. The operator opens the assistant on the entity and asks the assistant to *"analyse and design proper settings for this feed"*.





The assistant reasons over the entity's live context and recommends the **Feed Lifecycle Advisor**, launching it in **inspect** mode. A live tool-call feed shows each read the agent performs (`get_entity_lifecycle_context`, `get_entity_delay_latency_history`, `get_entity_variable_delay_schedule`, …).





The inspect result is rendered as structured recommendations (here a variable-delay schedule and a priority change) each with current value, recommended value and rationale, plus a collapsible reasoning trace. From the result, a **Run Act now** panel offers an optional *additional instructions* textbox before applying the changes.





When you run **act**, the advisor applies the changes through TrackMe's REST API and verifies each write. It also attaches a formatted **investigation & remediation report** as an entity note, prefixed `[AI Agent]` — documenting the root-cause analysis, the entity summary, and every change applied — so the full story is visible directly on the entity.





3.4.4 Structured output and guardrails

Every advisor run emits a **typed envelope** (validated by the SDK) rather than free-form prose:

- `entity_status` — an at-a-glance verdict (e.g. `healthy` / `needs_tuning` / `decommission_candidate`).
- `summary` — a short executive summary.
- `recommendations` — structured items, each with the field, current value, recommended value and a rationale that cites the tool calls that produced it.
- `actions_taken` — populated in *act* mode only, one entry per write that actually occurred.
- `reasoning_trace` — a step-by-step trace citing the read (and write) tool calls.

The guardrails that bound every run:

- **Ground every claim in data** — the agent's first calls are always reads; a run with zero read tool calls is a bug. Speculation is forbidden by the system prompts.
- **Tag-filtered write gate** — write tools are present only in *act* mode.
- **Decommission guard** — disabling an entity (`monitored_state=disabled`) is gated and disabled by default in the automated path.
- **RBAC at the REST boundary** — the agent inherits the operator's session; an unauthorised write fails cleanly with a 403.
- **Audit reason discipline** — every write requires a meaningful `reason` citing the field, the from/to values, and the operational trigger; trivial reasons are rejected.

3.4.5 The AI FLX Threshold Advisor

The **FLX Threshold Advisor** handles *FLX (Flex Objects)* — the "any SPL search as a tracked entity" component. FLX entities expose custom KPIs (a `metrics` JSON blob computed by the tracker's search), dynamic thresholds (per-metric value comparisons with configurable operators), and optional variable thresholds (time-slot-based values for day-of-week / hour-of-day variation). The advisor answers: *"are these thresholds tuned for this entity's actual behaviour, or did the wizard's defaults drift?"*

When to use it

- A FLX entity is RED and you want to know whether the threshold was wrong (over-tight) or the data is genuinely anomalous.
- A new FLX entity created via the wizard has been running for a week and you want to validate the applied defaults against observed behaviour.
- A metric's breach pattern matches business-hours / off-hours variation and a variable-threshold schedule would fit better than a static one.
- You want to delete obsolete thresholds or review a customised use-case template entity.

What it reads and writes

In **inspect** mode the advisor reads the full FLX entity context, the 24h + 7d metric value history, the threshold-breach history, the entity's use-case template definition, and peer-entity thresholds. In **act** mode it can add / update / delete dynamic thresholds, configure variable-threshold slots, and update the entity's monitoring state or priority (decommission-gated).

Note

The FLX Threshold Advisor is the most hardened advisor against the FLX `condition_true` flag, which inverts alert semantics. **Inverse-style** (`condition_true=true`) means "alert fires when the comparison is FALSE" — used for "healthy if X holds" tracking. **Direct-style** (`condition_true=false`) means "alert fires when the comparison is TRUE" — used for "alert if X holds" tracking. The advisor validates a proposed threshold's direction before recommending it.

Where to find it

The **AI Threshold Advisor** tab appears on FLX entity overview modals, and the chat bridge proposes it for FLX threshold intents. It participates in the nightly automated batch — see *AI Agents automation*.

Tip

Beyond tuning *existing* entities, this advisor can also **build a brand-new FLX tracker from a plain-language description** at wizard time — grounded on the use-case library and validated by a live dry-run. See *Natural-language FLX tracker generation*.

3.4.6 Related

- *AI Feeds Advisor*, *AI ML Advisor*, *AI FQM Advisor*, *AI Component Health Advisor* — the per-advisor walk-throughs.
- *The AI Concierge* — the catalog-driven generalist for everything outside the specialists.
- *AI Agents automation* — running the advisors unattended on a schedule.
- *Auditing AI activity* — auditing every advisor run.

AI Feeds Advisor

Overview

The **AI Feeds Advisor** (the *Feed Lifecycle Advisor*) is the most commonly used advisor in production. It handles **DSM (Data Sources)** and **DHM (Data Hosts)** entities — the bread-and-butter of TrackMe monitoring — and answers the recurring question *"is this entity's lifecycle configuration appropriate for its actual behaviour?"*

A "lifecycle" in TrackMe terms is an entity's runtime configuration story: how long after the last event is "still healthy", what counts as an excessive delay or lag, whether to auto-tune thresholds, whether to apply different thresholds at different times of day, and how to weight delay/latency breaches against other anomaly signals. The advisor reads

the entity's recent history, compares it against peer entities on the same datasource, and proposes lifecycle changes grounded in real telemetry.

See *The AI Advisors* for the shared inspect/act model, the consent-card bridge, and the structured output contract that all advisors share.

When to use it

- An entity is RED with a delay or latency threshold breach and you suspect the threshold is too tight (the most common trigger).
- A new entity was created via the wizard with default thresholds and you want them tuned after a week of observation.
- An entity's delay bursts in clear weekday/weekend or business-hours patterns and a variable-delay schedule may fit better than a static threshold.
- Adaptive delay was previously locked and you want to know if it is safe to unlock now that the pattern has stabilised (or vice versa).
- You want to adjust the entity's impact-score weights, or decide whether an entity should be decommissioned.

What it reads and writes

In **inspect** mode the advisor reads the full lifecycle context (configuration, thresholds, state, scoring, adaptive flags), the 7-day delay/latency history, the state-flip history, peer-entity comparison, the current variable-delay schedule, and the adaptive-delay change history.

In **act** mode it can additionally:

- update delay / lag thresholds (`data_max_delay_allowed`, `data_max_lag_allowed`),
- set the monitoring state (decommission-gated),
- lock or unlock adaptive delay,
- create or update a variable-delay schedule (per-slot day-of-week / hour-of-day thresholds),
- adjust priority, tags and impact-score weights,
- and attach labels and a formatted investigation note to the entity.

Walkthrough

Every DSM and DHM entity overview exposes an **AI Feeds Advisor** tab. It summarises what the advisor does for this entity and offers a **Launch Feed Lifecycle Advisor** button. In this example the entity `siem-cloud-apac:google:gcp:pubsub:message` is RED because of both a lag-threshold and a delay-threshold breach.





The advisor runs in **inspect** mode first, reading the entity's lifecycle context and history. It returns a structured verdict — here *Needs Tuning* — with an executive summary and a list of recommendations, each carrying the current value, the recommended value, and a rationale grounded in the observed ingestion behaviour and peer comparison.





The recommendations are specific and reviewable — for example raising `data_max_delay_allowed` from 3600s to 9000s to align with observed ingestion buffering and peer thresholds, designing a variable-delay schedule for the business-hours pattern, and re-weighting the impact score.





When you choose to apply the changes, the advisor switches to **act** mode and executes each write through TrackMe's REST API, re-reading to confirm the new state.





The **Actions Taken** panel lists every write the advisor performed — threshold updates, the adaptive delay lock, the priority/tags change, and the entity label and investigation note — each with its audit reason. A **reasoning trace** documents the analysis, and a link jumps to the advisor's audit events for this entity.





Note

Coexistence with legacy Adaptive Delay — When the AI Feeds Advisor covers DSM/DHM for a tenant, it takes precedence over the legacy mechanical *Adaptive Delay* and *Variable Delay auto-review* mechanisms, which are automatically disabled for those components so they cannot revert the advisor's per-entity decisions. Re-enabling the legacy mechanisms is a manual action from the Tenant Configuration modal.

Running it from the chat

In addition to the standalone modal, the AI Feeds Advisor can be proposed by the *AI Assistant* as a one-click consent card when you describe a feed problem in chat (e.g. *"this entity is RED, the delay threshold looks too tight"*). See *The AI Advisors* for the full bridge walkthrough.

Automated runs

The AI Feeds Advisor participates in the nightly automated batch, tuning eligible DSM/DHM entities within per-tenant policy (priority filter, freshness window, decommission guard). See *AI Agents automation*.

Related

- *The AI Advisors* — the advisor family overview (inspect/act, the bridge, guardrails).
- *Use Case Demo: This is All About Feeds* — the DSM/DHM feed components this advisor operates on.
- *Outlier detection* and *AI ML Advisor* — the ML layer the AI ML Advisor reasons about.
- *Auditing AI activity* — auditing advisor activity.

AI ML Advisor

Overview

The **AI ML Advisor** was the first advisor TrackMe shipped, and the model the others adopted. It inspects and remediates **native Machine Learning outlier-detection models** — the density-function fits, the boundary configurations, the scoring timelines, and the period exclusions. It is the advisor you reach for when you ask *"should I retrain?"*, *"is this outlier real?"*, or *"should I exclude this window from training?"* — questions that previously required reading several SPL queries and the KV Store by hand.

The ML Advisor reasons about the **model internals** — distinct from the component advisors that tune the entity *configuration* on top of the model. It applies to any component with ML outlier detection enabled (DSM, DHM, FLX, FQM, WLK, and — since 2.4.18 — VOL, where the outlier verdict *is* the entity state). The automated batch (*AI Agents automation*) reviews VOL models like the others.

See *The AI Advisors* for the shared inspect/act model, the consent-card bridge, and the structured output contract, and *Outlier detection* for the underlying ML outlier detection feature.

When to use it

- An entity is RED because of an outlier-score breach, but you suspect the model is over-sensitive or the training window included a known incident.
- A retrain just completed and the model now flags healthy traffic as anomalous (a false-positive cascade).
- The `isOutlier` flag is sticky — the score will not return to baseline even after the anomaly has clearly passed.
- A density function's boundaries look wrong compared to the entity's actual traffic distribution.
- The model is "low confidence" and you want to know whether more training data would help.

What it reads and writes

In **inspect** mode the advisor reads the full outlier context (identity, health, score breakdown, model rules, detection state and boundaries, confidence level), the raw metric history, the per-group fitted distributions and fit quality, the outlier-score timeline, the entity's state-transition history, and the full training-window render (metric values, decided boundaries, flagged points).

In **act** mode it can additionally:

- add a **period exclusion** to remove a contaminated window from the training set,
- **trigger a retrain** (live or simulation),
- **set a false positive** to neutralise the current outlier,
- **update model rules** (density-function bounds, sliding-window size, seasonality flags, …),
- **enable / disable / reset** outlier detection (the disable action is decommission-gated),
- and attach labels and an investigation note to the entity.

A distinguishing field of the ML Advisor's structured output is `anomaly_windows` — detected periods worth excluding from training, each with a start, end, and reason.

Example 1 — a misbehaving model

The ML Advisor is launched from an entity's **Outliers anomaly detection** tab via the **AI ML Advisor** button. In this example the model is misbehaving — producing a large number of lower-bound outliers that do not reflect a genuine data problem.









The advisor inspects the model, classifies the situation, and proposes remediation. In **act** mode an *Additional instructions* textbox lets you constrain the agent — for example *"save investigations and remediations as a markdown-formatted note to the entity, and assign label under-review".*


Actions for entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit (alias: cloud:gcp:pubsub:audit)
AI ML Advisor
Entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit [TLX]
Act mode will re-analyze the entity and automatically apply recommended changes (e.g., add period exclusions, update model configuration, retrain models).
Additional instructions (optional)
save investigations and remediations as a markdown formated note to the entity, and assign label under-review
Back to Results
▶ Run Act
Close
144
933
Outliers Anomaly Detection


Actions for entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit (alias: cloud:gcp:pubsub:audit)
AI ML Advisor
Actions Taken (5)
add_period_exclusion
trigger_model_retrain
set_false_positive
manage_entity_labels
manage_entity_note
Reasoning Trace
See AI ML Advisor events
Close
Outliers Anomaly Detection


Actions for entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit (alias: cloud:gcp:pubsub:audit)
AI assistant
Overview entity
Incidents
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Handlers activity
About Outliers Anomaly Detection
Last 30 days
AI ML Advisor
Manage outliers detection
Outliers Anomaly Detection
Anomalies: 0


Actions for entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit (alias: cloud:gcp:pubsub:audit)
AI assistant
Overview entity
Incidents
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Handlers activity
Last 15 minutes
Audit changes over time
Audit changes events
Time
Event
action: "success",
change_type: "assign labels",
object: "demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:cloud:gcp:pubsub:audit",
object_category: "splk-flx",
result: "Labels assigned successfully (3 label(s))",
comment: "[AI Agent] ML Advisor ACT run: entity placed under-review during behaviour-change remediation. Model retrain in progress after excluding pre-May-4 contaminated baseline.",
tenant_id: "demo_outliers",



Example 2 — a true positive with period exclusions

Not every anomaly is a false alarm. In this example the model has flagged a **genuine** anomaly (a true positive). The advisor inspects the score history and the training render, identifies the contaminated window, and — in act mode — applies **period exclusions** so the incident period does not poison future training, then retrains the model and documents its reasoning.




AI ML Advisor
ML Advisor is analyzing...
Cancel
Close


AI ML Advisor
Anomaly Detected
Summary
Anomaly Windows (1)
Recommendations (3)
high
Period Exclusion
Close


AI ML Advisor
Recommendations (3)
high
Period Exclusion
high
Model Retrain
medium
False Positive
Reasoning Trace
New Analysis
Act Now
Close


AI ML Advisor
Additional instructions (optional)
Back to Results
▶ Run Act
Close


AI ML Advisor
ML Advisor is analyzing...
Cancel
Close


AI ML Advisor
Actions Taken (4)
Reasoning Trace
New Analysis
Close


ML Outliers Anomaly Detection
Machine Learning Outliers Anomaly Detection - Periods exclusions per ML model
Use this screen to manage ML models periods exclusions
Periods Exclusions can be used to exclude periods of time from the ML models training, and avoid models from being impacted by behaviour anomalies:
These periods are not taken into account while training ML models
When performing simulation, these periods are fully excluded (and can therefore appear as gaps in the chart)
When an exclusion period is expired and is out of the ML model training time range, it is automatically deleted when processing the next ML model training
The UI applies addition and deletion of periods exclusions to all ML models for this object
Time is shown in your own local time zone settings.
Starts on
Ends on
Created at
Add a new period exclusion
Starts on:
Ends on:
+ Add a new period exclusion
Delete selected (0)
Back to outliers manage
Close


Actions for entity: demo:836b074f-9bb7-42f6-9ed5-0fc259c2e244:endpoint:XmlWinEventLog:Microsoft-Windows-Sysmon/Operational (alias: endpoint:XmlWinEventLog:Microsoft-Windows-Sysmon/Operational)
AI assistant
Status reason
Overview entity
Incidents
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Handlers activity
Last 15 minutes
Audit changes over time
Audit changes events
user:
result: "ML models were updated successfully",
comment: "[AI Agent] Extended period exclusion covering the full anomaly window on Jun 1 2026. splk.flx.events_count dropped sharply from ~9.28M to ~2.32M (-75%) at 16:40-16:50 UTC, sustained well below the lower bounds of 5,250,019 (group 17), 5,289,739 (group 18), and 5,253,792 (group 19) until recovery at ~20:30-20:50 UTC. A prior acc-mode run on Jun 1 21:23 UTC only excluded 19:50-20:50 UTC, leaving ~3 hours of severely breached data points (16:40-19:50 UTC) unprotected in the training window. This broader exclusion (16:00-21:00) covers the complete anomaly span so that hourly groups 16-20 are not contaminated on future retrains. No prior occurrence of this magnitude in the 90 day render history - not a normal business pattern.",
tenant_id: "demo-outliers",



Note

Retraining is asynchronous — `trigger_model_retrain` returns once the retrain is *queued*, not once it completes; the model is retrained on the next scheduled fit. **Period exclusions are additive** — adding one does not replace existing exclusions. The advisor's reasoning trace and the on-entity note make both behaviours explicit.

Running it from the chat

The ML Advisor can also be proposed by the *AI Assistant* as a consent card when you describe an ML symptom in chat (e.g. *"this outlier seems wrong"*, *"should I retrain?"*, *"exclude the last two hours from training"*). See *The AI Advisors*.

Automated runs

The ML Advisor participates in the nightly automated batch, reviewing eligible entities with active outliers within per-tenant policy. The model-disable action has its own dedicated guard (`ai_mladvisor_allow_model_disable`, default off). See *AI Agents automation*.

Related

- *The AI Advisors* — the advisor family overview.
- *Outlier detection* — the native ML outlier detection feature the advisor reasons about.
- *AI Feeds Advisor* — the companion advisor that tunes the entity configuration on top of the model.
- *Auditing AI activity* — auditing advisor activity.

AI FQM Advisor

Overview

The **AI FQM Advisor** handles **FQM (Field Quality Monitoring)** — TrackMe's component for tracking field parsing quality and CIM compliance. It is the most feature-rich advisor of the family: it manages dynamic thresholds *and* a per-tracker **data dictionary** that defines per-field validation logic (regex, datatype, required/optional, expected

values, CIM datamodel mapping). It also has a unique wizard-time **dictionary generation** capability that proposes a starter dictionary from sampled-field statistics — before the entity even exists.

See *The AI Advisors* for the shared inspect/act model and the structured output contract, and *Use Case Demo: Fields Quality (CIM and non-CIM)* for the FQM component itself.

What FQM tracks

FQM answers *"are the fields in this sourcetype being parsed correctly and conforming to CIM expectations?"* For each tracked sourcetype, you define a **data dictionary** mapping field names to validation rules. A scheduled collect-job samples real events and computes `percent_success` (events that pass field validation) and `percent_coverage` (events where the field was present at all); dynamic thresholds turn those numbers into alerts.

This gives the advisor three flavours of work:

1. **Threshold tuning** — "your `percent_success` threshold of 99% is unrealistic for this noisy sourcetype; the observed median is 95% — recommend 90%."
2. **Dictionary maintenance** — "your `src_ip` field has a 15% failure rate because the regex does not match IPv6 — update the regex"; or "you have no validation for `user_id` but it is consistently a 10-digit number — add a rule."
3. **Wizard-time dictionary generation** — propose a starter dictionary from observed field types, value distributions, and CIM datamodel hints.

When to use it

- An FQM entity is RED because of a field-quality threshold breach and you want to validate whether the threshold is realistic.
- A specific field has a high failure rate and you want the advisor to inspect failing samples and propose a regex fix.
- The data dictionary was set up months ago and the sourcetype's payload has drifted (new field, renamed field, format change) — a "review my dictionary" pass.
- You are creating a new FQM tracker via the wizard and want a starter dictionary proposal from sampled-field statistics.

What it reads and writes

In **inspect** mode the advisor reads the full FQM entity context, the data dictionary (and which trackers share it), individual field entries, the quality history, sampled failing events, the collect-job health, peer-entity thresholds, and the CIM datamodel diff. It can also **test a candidate regex** against real sampled values before recommending it.

In **act** mode it can add / update / delete dynamic thresholds, update individual field-dictionary entries, bulk-update the dictionary, and update the entity's monitoring state or priority. A separate **dictionary_generate** mode runs at wizard time with no tools — it reasons purely from the sampled field statistics to propose a dictionary.

Note

The data dictionary is **per-tracker**, not per-entity — a dictionary change affects every entity sharing the tracker. Bulk dictionary updates are atomic (all-or-nothing), and operator review of a generated dictionary is always required regardless of the confidence score.

Example 1 — a CIM tracker with a generated dictionary

The FQM Advisor's dictionary generation is integrated directly into the **Fields Quality Monitoring** tracker-creation wizard. When creating a collect job for a CIM datamodel, the wizard steps you through selecting the tracker type, defining the search, sampling fields, and previewing a generated data dictionary. The embedded **Wizard Assistant** offers to *"generate the data dictionary based on the sampled fields"* and explains the validation options (e.g. `allow_unknown` vs `allow_empty_or_missing`).


Fields Quality Monitoring - Create a new collect job for CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Actions
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Datamodel & search
4 CIM parsing & dictionary
5 Preview the data dictionary
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
Select the type of quality tracker you want to create:
Tracker type:
Create a new collect job for CIM (generate or re-use a dictionary)
About tracker types


Fields Quality Monitoring - Create a new collect job for CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Datamodel & search
4 CIM parsing & dictionary
5 Preview the data dictionary
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
Enter a name for the new object and define if the deployment is local or a remote account:
apache:accesskv-5933
New collect job identifier: (will be added as a prefix to entities, and used to group entities)
This tracker name is valid and does not exist yet in this tenant.
Splunk deployment, either local or a configured remote account:
local
Open the configuration interface to add or manage accounts
About the environment target


Fields Quality Monitoring - Create a new collect job for CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Datamodel & search
4 CIM parsing & dictionary
5 Preview the data dictionary
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
About defining the search and the breakby fields definition:
Search type:
_raw
Datamodel:
Web
NodeName:
Web
Events resulting from this search will be processed by the TrackMe quality management backend, and indexed in the trackme:fields_quality sourcetype
Enter the Splunk search
Edit
SPL Preview
index=webserver sourcetype=apache:access:kv
Comma separated list of fields used to define entities, you can add a custom field available in the data set as metadata.<fieldname>
Breakby fields definition:
metadata.datamodel,metadata.nodename,metadata.index,metadata.sourcetype,fieldname


Fields Quality Monitoring - Create a new collect job for CIM
7 Collect job strategy
8 Collect job main settings
9 Validate creation
Simulate the search to continue
All fields:
Total fields:
34
Fields Passed:
17
Fields Failed:
17
CIM % Compliance all fields:
50.00%
Recommended fields:
Total recommended fields:
16
Recommended fields passed:
9
Recommended fields failed:
7
CIM % Compliance recommended fields:
56.25%
Show recommended fields only?
True
Field Name
Recommended
Status
Coverage %
Success %
Total
Success
Failure
Distinct Values
Index
Sourcetype
Inspect
action failure 100.0 0.00 1000 0 1000 8 webserver apache:access:kv
app failure 0.0 0.00 1000 0 1000 0 webserver apache:access:kv
bytes_in success 100.0 100.00 1000 1000 0 500 webserver apache:access:kv
bytes_out success 100.0 100.00 1000 1000 0 477 webserver apache:access:kv
category failure 17 1.60 1000 16 984 5 webserver apache:access:kv
dest success 100.0 100.00 1000 1000 0 4 webserver apache:access:kv
dest_ip failure 0.0 0.00 1000 0 1000 0 webserver apache:access:kv
http_content_type failure 17 1.20 1000 12 988 8 webserver apache:access:kv
http_method success 100.0 98.20 1000 982 18 5 webserver apache:access:kv


Fields Quality Monitoring - Create a new collect job for CIM
Collect job strategy
Collect job main settings
Validate creation
Simulate the search to continue
All fields:
Total fields:
34
Fields Passed:
17
Fields Failed:
17
Recommended fields:
Total recommended fields:
16
Recommended fields passed:
9
Recommended fields failed:
7
Show recommended fields only?
True
Field Name
Recommended
Status
Coverage %
Success %
Total
Success
Failure
Distinct
action
failure
100.0
0.00
1000
0
1000
8
app
failure
0.0
0.00
1000
0
1000
0
bytes_in
success
100.0
100.00
1000
1000
0
509
bytes_out
success
100.0
100.00
1000
1000
0
477
category
failure
17
1.60
1000
16
984
5
dest
success
100.0
100.00
1000
1000
0
4
dest_ip
failure
0.0
0.00
1000
0
1000
0
http_content_type
failure
1.7
1.20
1000
12
988
5
http_method
success
100.0
98.20
1000
982
18
5
AI Assistant
claude_haiku (claude-haiku-4-5)
Context: Wizard Assistant
Suggested questions:
Can you generate the data dictionary based on the sampled fields?
What does allow_unknown vs allow_empty_or_missing mean?
I haven't sampled fields yet — what should I do first?
Ask a question... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM
Collect job strategy
Collect job main settings
Validate creation
Simulate the search to continue
All fields:
Total fields:
34
Fields Passed:
17
Fields Failed:
17
Recommended fields:
Total recommended fields:
16
Recommended fields passed:
9
Recommended fields failed:
7
Show recommended fields only?
True
Field Name
Recommended
Status
Coverage %
Success %
Total
Success
Failure
Distinct
action
failure
100.0
0.00
1000
0
1000
8
app
failure
0.0
0.00
1000
0
1000
0
bytes_in
success
100.0
100.00
1000
1000
0
509
bytes_out
success
100.0
100.00
1000
1000
0
477
category
failure
17
1.60
1000
16
984
5
dest
success
100.0
100.00
1000
1000
0
4
dest_ip
failure
0.0
0.00
1000
0
1000
0
http_content_type
failure
1.7
1.20
1000
12
988
5
http_method
success
100.0
98.20
1000
982
18
5
AI Assistant
claude_haiku (claude-haiku-4-5)
Context: Wizard Assistant
You
Can you generate the data dictionary based on the sampled fields?
AI Assistant
Absolutely! I'll run the FQM Advisor in dictionary_generate mode to propose a starter data dictionary based on the fields you've already sampled in the wizard.
This will analyze the field distributions (cardinality, value patterns, data types) and recommend appropriate validation rules for each field — things like:
regex patterns (for structured fields like IPs, timestamps, etc.)
allow_unknown (whether to accept values outside the sampled set)
allow_empty_or_missing (whether nulls/blanks are valid)
The wizard will inject your sampled-fields payload automatically, so you'll get recommendations tailored to your actual data.
FQM ADVISOR — PROPOSED ACTION
Generate a starter data dictionary with field validation rules based on your sampled field distributions.
Mode: dictionary_generate
Generate dictionary
Cancel
Ask a question... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM
Collect job strategy
Collect job main settings
Validate creation
Simulate the search to continue
All fields:
Total fields:
34
Fields Passed:
17
Fields Failed:
17
Recommended fields:
Total recommended fields:
16
Recommended fields passed:
9
Recommended fields failed:
7
Show recommended fields only?
True
Field Name
Recommended
Status
Coverage %
Success %
Total
Success
Failure
Distinct
action
app
bytes_in
bytes_out
category
dest
dest_ip
http_content_type
http_method
failure
success
AI Assistant
FQM ADVISOR — COMPLETE (DICTIONARY_GENERATE)
SUMMARY
PROPOSED DICTIONARY ENTRIES (33)
Ask a question ... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM
Collect job strategy
Collect job main settings
Validate creation
Simulate the search to continue
All fields:
Total fields:
34
Fields Passed:
17
Fields Failed:
17
Recommended fields:
Total recommended fields:
16
Recommended fields passed:
9
Recommended fields failed:
7
Show recommended fields only?
True
Field Name
Recommended
Status
Coverage %
Success %
Total
Success
Failure
Distinct
action
app
bytes_in
bytes_out
category
dest
dest_ip
http_content_type
http_method
failure
success
AI Assistant
CONFIDENCE NOTES
Reasoning trace
Apply to wizard form
Ask a question ... (Shift+Enter for new line, supports Markdown)
Send



Example 2 — a non-CIM tracker with a brand-new dictionary

The same flow applies to a **non-CIM** tracker, where there is no datamodel to align against. The advisor proposes a brand-new dictionary purely from the observed field types and value distributions — proposing presence-only validation where a field has no clear pattern rather than inventing a regex. The proposed dictionary is rendered as an editable table for operator review before it is applied.




Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Search & metadata
4 Dictionary configuration
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
Enter a name for the new object and define if the deployment is local or a remote account:
businessordersitemsections-3035
New collect job identifier: (will be added as a prefix to entities, and used to group entities)
This tracker name is valid and does not exist yet in this tenant.
Splunk deployment, either local or a configured remote account:
local
Open the configuration interface to add or manage accounts
About the environment target


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Search & metadata
4 Dictionary configuration
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
About defining the search and the breakby fields definition:
Pseudo datamodel name (metadata):
raw
This metadata will be stored and used in the backend to identify the pseudo datamodel.
Events resulting from this search will be processed by the TrackMe quality management backend, and indexed in the trackmefields_quality sourcetype:
Enter the Splunk search:
Edit
SPL Preview
index="business_data" sourcetype="business:orders:transactions"
Comma separated list of fields used to define entities, you can add a custom field available in the data set as metadata.<fieldname>
Breakby fields definition:
metadata.datamodel,metadata.index,metadata.sourcetype,fieldname
When using a custom breakby definition, you should also update the Virtual Tenant group by setting for splk-fqm, you can do so through the Configuration UI or by running the following command to update to your tenant Id and breakby fields
Note that the metadata field name is called with an _ instead of a dot
| trackme mode=post url="/services/trackme/v2/configuration/admin/update_vtenant_account" body="{'tenant_id': 'mytenant', 'parameters_to_update': { 'splk_fqm_tabulator_groupby': 'metadata_datamodel,metadata_myfield,metadata_sourcetype'}}"


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Search & metadata
4 Dictionary configuration
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
About CIM parsing preview and dictionary configuration:
Dictionary: (create new or use existing)
Create new
New dictionary name?
businessoperationstransactions
Allow unknown by default?
False
Allow missing/null by default?
False
Event limit:
1000
% success fields threshold (simulation only): 85%
0
100
Fields summary
Fields summary (showing data for 1000 events):
Load Fields Summary
field
count
distinct_count
is_exact
numeric_count
values
mean
min
max
stdev
amount
65
65
1
65


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
AI Assistant
claude_haiku (claude-haiku-4-5)
Context: Wizard Assistant
Suggested questions:
Can you generate the data dictionary based on the sampled fields?
What does allow_unknown vs allow_empty_or_missing mean?
I haven't sampled fields yet — what should I do first?
Ask a question... (Shift+Enter for new line, supports Markdown)
Send
Fields Quality Monitoring Workflow Diagram
About CIM parsing preview and dictionary configuration:
Fields summary
Fields summary (showing data for 1000 events):


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Search & metadata
4 Dictionary configuration
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
About CIM parsing preview and dictionary configuration:
Dictionary: (create new or use existing)
Create new
New dictionary name?
Allow unknown by default?
False
Allow missing/null by default?
False
Event limit:
1000
% success fields threshold (simulatio
0
Fields summary
Fields summary (showing data for 1000 events):
field
count
distinct_count
is_exact
numeric_count
values
amount
65
65
1
65
AI Assistant
claude_haiku (claude-haiku-4-5)
Context: Wizard Assistant
You
Can you generate the data dictionary based on the sampled fields?
AI Assistant
Absolutely! I'll run the FQM Advisor in dictionary_generate mode to propose a starter data dictionary based on the fields you've already sampled in the wizard.
This will analyze the field distributions (counts, distinct values, data types, sample values) and recommend appropriate rules for each field — things like regex patterns, whether to allow unknown values, and whether to permit empty or missing fields. You'll get a complete dictionary ready to review and refine before you save the tracker.
FQM ADVISOR — PROPOSED ACTION
Generate a starter data dictionary with field-level rules (regex, allow_unknown, allow_empty_or_missing) from the sampled fields.
Mode: dictionary_generate
Generate dictionary
Cancel
responded in 3.1s
Ask a question... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
Follow these guided steps to define, test and create a new Quality collect and monitor job, scroll down within the screen to navigate through the steps
Fields Quality Monitoring Workflow Diagram
1 Select tracker type
2 tracker name & deployment
3 Search & metadata
4 Dictionary configuration
6 Default thresholds
7 Collect job strategy
8 Collect job main settings
9 Validate creation
About CIM parsing preview and dictionary configuration:
Dictionary: (create new or use existing)
Create new
New dictionary name?
Allow unknown by default?
False
Allow missing/null by default?
False
Event limit:
1000
% success fields threshold (simulatio
0
Fields summary
Fields summary (showing data for 1000 events):
field
count
distinct_count
is_exact
numeric_count
values
amount
65
65
1
65
AI Assistant
claude_haiku (claude-haiku-4-5)
Context: Wizard Assistant
You
Can you generate the data dictionary based on the sampled fields?
AI Assistant
Absolutely! I'll run the FQM Advisor in dictionary_generate mode to propose a starter data dictionary based on the fields you've already sampled in the wizard.
This will analyze the field distributions (counts, distinct values, data types, sample values) and recommend appropriate rules for each field — things like regex patterns, whether to allow unknown values, and whether to permit empty or missing fields. You'll get a complete dictionary ready to review and refine before you save the tracker.
FQM ADVISOR — RUNNING (DICTIONARY_GENERATE)
Job a281250c... · 50s elapsed
running reasoning_step_2 ..
Cancel run
responded in 3.1s
Ask a question... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
AI Assistant
Fields Quality Monitoring Workflow Diagram
Select tracker type
tracker name & deployment
Search & metadata
Dictionary configuration
Default thresholds
Collect job strategy
Collect job main settings
Validate creation
About CIM parsing preview and dictionary configuration:
Fields summary
Ask a question ... (Shift+Enter for new line, supports Markdown)
Send


Fields Quality Monitoring - Create a new collect job for CIM/non-CIM
AI Assistant
Apply to wizard form
Ask a question ... (Shift+Enter for new line, supports Markdown)
Send



Threshold tuning and dictionary review

Beyond the wizard, the **AI FQM Advisor** tab on FQM entity overviews runs the standard inspect/act flow for existing trackers — validating thresholds against observed quality history, inspecting failing samples, testing candidate regexes, and proposing dictionary updates (which can also be proposed by the *AI Assistant* as a consent card). It participates in the nightly automated batch for threshold tuning and dictionary maintenance; dictionary *generation* is wizard-only by design. See *AI Agents automation*.

Related

- *The AI Advisors* — the advisor family overview.
- *Use Case Demo: Fields Quality (CIM and non-CIM)* — the FQM component this advisor operates on.
- *Auditing AI activity* — auditing advisor activity.

AI Component Health Advisor

Overview

The **AI Component Health Advisor** handles two components in a single advisor with dynamic per-component tool selection:

- **WLK (Workload Monitoring)** — the execution health of Splunk scheduled saved searches: skip rate, error count, execution delay, versioning drift.
- **MHM (Metric Host Monitoring)** — the lag of hosts reporting metric data (the metric-equivalent of DHM), including per-class lagging thresholds.

The same advisor code drives both because the threshold-management surface is structurally identical and the decision-making pattern is the same (read history → compare peers → recommend). At launch the advisor selects only the tools for the component in scope, so a WLK run never sees MHM tools and vice versa.

See *The AI Advisors* for the shared inspect/act model, the consent-card bridge, and the structured output contract.

When to use it

WLK

- A WLK entity is RED because the skip rate exceeded its threshold and you want to know if the threshold is realistic for this saved search's load profile.
- An app deployment introduced new scheduled searches with default thresholds and you want them validated after a week.
- A saved search's error count is bursting and you want to inspect peer thresholds across the app.

MHM

- A metric host is RED because of excessive lag and you want to check whether the lag is genuinely problematic or the threshold is too tight.
- The tenant's lagging classes (platinum / gold / silver) have stale threshold values and you want them reviewed.

What it reads and writes

For **WLK**, inspect reads the full entity context, the execution history (skip rate, error rate, execution time per day), the breach history, and peer thresholds in the same app. In act mode it can add / update / delete execution-metric thresholds and update the entity's monitoring state or priority (decommission-gated).

For **MHM**, inspect reads the entity context, the metric-lag history, the breach history, and the tenant's lagging-class definitions. In act mode it can update the per-entity metric-lag threshold and the entity's monitoring state or priority.

Note

The advisor can *recommend* a lagging-class-level change for MHM but cannot apply one — class-level changes are tenant-level and go through the admin UI or the *AI Concierge*. For WLK, if the error count is consistently non-zero with no deployment change, the advisor recommends investigating the saved search itself rather than raising the threshold to mask a broken search.

Example — a WLK schedule in error

The Component Health Advisor is launched from a WLK entity's **AI Health Advisor** tab via the **Launch Workload Health Advisor** button. In this example the saved search `DA-ESS-sandbox:...:Threat - UC003 - Disallowed accesses - Rule` is RED with `execution_errors_detected` — the schedule is failing, not merely slow.





The advisor inspects the entity, reading the execution context, the execution-error history, the execution history, and the breach history via its live tool feed.





It then returns a structured verdict and recommendations — here correctly diagnosing that the search itself is failing and that the fix is to investigate the SPL rather than relax the threshold.





Where to find it

The advisor appears as the **AI Health Advisor** (WLK) tab on WLK entity overviews and as the **AI MHM Advisor** tab on MHM entity overviews — both run the same backend with the appropriate component selected. It can also be proposed by the *AI Assistant* as a consent card, and it participates in the nightly automated batch (one batch per component). See *AI Agents automation*.

Related

- *The AI Advisors* — the advisor family overview.
- *WLK — Workload Knowledge* — the WLK Workload Monitoring component.
- *Auditing AI activity* — auditing advisor activity.

3.5 AI Agents for findings

3.5.1 Overview

TrackMe 2.4.19 introduces the **AI Agent**: an agentic operator for the dedicated findings tenants. Where the *AI Advisors* tune the configuration of monitored entities, the AI Agent works on **findings**. It triages them the way an operator would: it reads the evidence TrackMe holds, assesses each finding, recommends a decision and, when the tenant allows it, applies that decision itself, for example marking a Finding group investigating, dismissing it for a bounded time or resolving it once a recovery is proven. It keeps one current `[AI Agent]` note on every finding it reviewed and records every decision in a decision log.

Tenant type	AI Agent
User Activity Monitoring	**UAM AI Agent**. It replaces the UAM Findings Advisor of TrackMe 2.4.18; on upgrade, an enabled advisor's settings are carried over to the agent.
Splunk HEC	**Splunk HEC AI Agent**
Splunk S2S	**Splunk S2S AI Agent**
Splunk UF Fleet	**Splunk UF Fleet AI Agent**
Splunk Forwarder Collectors	**Splunk Forwarder Collectors AI Agent**
Cribl Stream	**Cribl Stream AI Agent**
Cribl Edge	**Cribl Edge AI Agent**

The seven agents run on **one shared framework**: the same identity, the same inspect / act contract, the same settings, the same scheduling, the same decision log and the same user interface. Each tenant type brings what is specific to it: its evidence, its assessments and the decisions it may take.

Note

The AI Agent is part of the *findings model* shared by the seven dedicated tenant types. The product guide covers it from the tenant's point of view in *Findings AI Agents*; this page is the reference from the AI side.

How the AI Agent differs from the other AI features

	AI Advisors and AI Agents automation	AI Agent
Works on	Entities of the monitoring components (DSM, DHM, FLX, FQM, WLK, MHM, VOL) and their configuration: models, thresholds, delays, dictionaries	Findings and Finding groups of a dedicated findings tenant: User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream or Cribl Edge
Decides	Configuration changes, with your consent or within the automation policy	Triage decisions on findings, within the decisions the tenant allows
Acts as	The user who launched it (interactive) or the tenant owner (scheduled)	Its own actor, **AI Agent**, shown in the finding history; the decision log also records who asked
Schedule	Nightly batch (*AI Agents automation*)	Hourly scheduled triage, per tenant
Where to follow it	The entity's audit changes, the AI Advisor audit dashboards	The **AI Agent** tab of the workspace, the finding history and notes, the same audit dashboards

3.5.2 How it works

Every review follows the same five steps, whether you ask the agent about one finding or the hourly triage reviews the open findings of the tenant.

1. **Select.** One finding per Finding group, the most severe first. A scheduled run skips the findings it reviewed recently, unless something changed since.
2. **Gather the evidence.** TrackMe computes the facts about the finding on the server, from what it already holds, and gives them to the agent: the finding, its episodes and its history, the policy, and the object it names. For UAM, that is the account, its class and where the class comes from, and its activity. For Cribl Stream, that is when the object was first seen, whether it is disabled in Cribl or gone from the Inventory, which Routes reference it, its traffic and its traffic ML model. For Cribl Edge, that is the node or the Fleet as the node directory last published it: connected and disconnected nodes, versions and configuration drift, the node's state, version, disk and the health of its Sources and Destinations, and whether the node or Fleet is still in the directory. For Splunk HEC, that is the endpoint (requests, refusals, back pressure and the bottleneck stage, queues, its output to the receivers, version and resources), the pool (expected size, members up and their shares) or the token, known by name only. For Splunk S2S, that is the receiver or intermediate forwarder (pipeline queues and bottleneck, output pauses and failing destinations, its senders, resources, churn) or the pool, its members and its indexer cluster. For Splunk UF Fleet, that is the agent (state and since when, identity, version, receivers, Deployment Server and server classes, output pauses, failing destinations, input errors, restarts), the Deployment Server against its hourly baseline, or the group and the fleet. For Splunk Forwarder Collectors, that is the collector (what it collected and forwarded, queues, output, resources, input errors), the input (its state, errors and last message, and whether it must stay enabled), the group (members reporting and healthy against its minimum) or the failing destination. For every tenant type, the evidence also says whether the object is still in the tenant's directory or inventory and how current that is.
3. **Assess.** The agent classifies the finding and gives its confidence (high, medium or low).
4. **Decide.** It recommends one decision. In act mode it applies it, and the server checks again, at that moment, that the decision is allowed and that its preconditions hold, whatever the model wrote. A decision that fails these checks is refused and recorded as such.
5. **Record.** It writes or refreshes its `[AI Agent]` note on the finding, the decision appears under the actor **AI Agent** in the history, and the review lands in the decision log and the audit trail.

Deterministic evidence first, the LLM second. The agent reasons on facts TrackMe checked, not on guesses, and it cannot write searches of its own: the Cribl Stream agent reads the newest certified interval, the traffic models and the incident's own record; the Cribl Edge agent reads the newest publication of the node directory; the Splunk HEC agent reads only what the tenant's collection published; the UAM agent reads bounded activity records (at most 50 events per read). The Splunk S2S, Splunk UF Fleet and Splunk Forwarder Collectors agents may also run their tenant's own read commands — log evidence, trends and the recent window of one object, each a bounded TrackMe search with a timeout — at most three reads per finding after the evidence, under the identity of the run. Proofs such as "this object has had no traffic for 7 days" are computed by TrackMe, never inferred by the model.

Some conclusions are not left to the model. On Splunk Forwarder Collectors, an input that must stay enabled and is still not enabled is always a true positive: whatever the model wrote, TrackMe records that review as a true positive to investigate, with high confidence.

Assessments

Assessment	Meaning
True positive	The condition is real and needs an operator.
Likely false positive	The evidence points to a detection that does not reflect a real problem.
Expected behaviour	The condition is real but expected, for example a seasonal or planned pattern.
Needs investigation	The evidence is not enough to conclude.
Object not in use *(Cribl Stream)*	The object is disabled, removed or never carried traffic.
Node or Fleet not in use *(Cribl Edge)*	The node or Fleet is retired: gone from the node directory, or a Fleet the tenant no longer collects.
Endpoint or pool not in use *(Splunk HEC)*	The endpoint is retired or replaced: a current directory no longer lists it and the finding has not been evaluated for an hour; or the pool is no longer defined. A token is never "not in use".
Host or pool not in use *(Splunk S2S)*	Neither the directory nor the live window lists the receiver or intermediate any more, or the pool is no longer listed. Indexer cluster findings never qualify.
Agent or Deployment Server not in use *(Splunk UF Fleet)*	The agent was forgotten or taken out of monitoring, or the current directory no longer holds it; or the Deployment Server is no longer listed. A group, the fleet and an *Expected agent missing* finding never qualify: the absence is the finding.
Collector, input or group not in use *(Splunk Forwarder Collectors)*	The collector or input left the inventory, or the group is no longer defined. A silent collector stays in the inventory for 30 days, so silence alone never qualifies; nor do a destination, the fleet, an expected collector missing or an input that must stay enabled.
Misclassified account *(UAM)*	The finding comes from an account classified as the wrong type (human or service).

3.5.3 Inspect and act

Mode	What it does
Inspect	Assesses and recommends. It writes one current `[AI Agent]` note per finding and changes nothing else. This is the default, for asked reviews and scheduled ones.
Act	Inspects, then applies the decisions the evidence supports **and** the tenant settings allow. Act must be chosen explicitly.

The decisions act may take, each with its own setting:

Decision	Default	UAM	Cribl Stream	Cribl Edge	Splunk HEC, S2S, UF Fleet, Forwarder Collectors
Mark investigating	On	Yes	Yes	Yes	Yes
Dismiss for a time (7 days at most by default, up to 90)	On	Never a critical or high finding still observed	A critical failing finding only with proof that the object is disabled, missing or inactive	A critical or high finding still failing only when its node or Fleet is gone from the node directory, and never across the nodes of a Finding group	A critical or high finding still failing only when its object is gone from the directory or the inventory, and never across the objects of a Finding group
Not an anomaly (a traffic outlier)	On	—	Yes	—	—
Resolve	Off	Only a finding its policy no longer observes	Only after a measured recovery	Only after a measured recovery	Only after a measured recovery, with no other member of its Finding group still failing
Classify an account (human or service)	Off	Only an account without a class set by an operator	—	—	—
Stop monitoring an object	Off	—	Only with proof of inactivity (7 days by default)	—	—

The agent **never** dismisses permanently, assigns or unassigns, changes a severity, or changes a policy, a model or a setting (on Cribl Edge, the node selection or the collected Fleets): these decisions do not exist in its vocabulary. On Cribl Stream it may **recommend** *Forget this object* for an object removed from Cribl, but only a TrackMe administrator of the tenant can do it. In the same way, the Splunk UF Fleet agent may recommend forgetting an agent or adding it to the expected agents, and the Splunk Forwarder Collectors agent forgetting a collector, adding it to the expected collectors or excluding one input from one policy: an operator decides. The Splunk agents never change anything on a Splunk instance, and the Splunk HEC agent never reads a token's value.

3.5.4 Asking the AI Agent

Ask the AI Agent… is in the finding and Finding group menus, the investigation view and the **AI Agent** tab of the workspace, with optional instructions for the review. The agent always inspects first: you read its assessment, its evidence and its rationale, then choose **Run its recommendations…** to let it act on that inspection, with optional instructions, within 30 minutes.

The **AI Assistant** of the workspace can also propose an AI Agent review in the chat, as a consent card you confirm, the same way it proposes an AI Advisor (see *AI Assistant*).


Ask the AI Agent
Cribl Stream AI Agent
Inspect first
Finding: splunk_hec:splunk-scde
Critical
SUMMARY
The finding is a critical reported health failure on the destination splunk_hec:splunk-scde in the default Worker Group. The destination is present and referenced by many routes, with recent healthy traffic activity and no evidence of being disabled or inactive. The finding is currently failing and critical, with no recovery observed. The evidence indicates a real problem requiring operator attention rather than a false positive or expected behaviour.
FINDINGS ASSESSED (1)
True Positive splunk_hec:splunk-scde Reported health High confidence
Decision: Mark investigating
Evidence: The destination 'splunk-scde' is present, not disabled, referenced by 18 routes, and carries significant recent traffic. The finding condition is failing with critical severity and no measured recovery. Traffic models are in learning phase, so no anomaly dismissal applies. The finding history and system records confirm a critical reported health failure.
The finding represents a critical health failure on a destination actively used and carrying traffic, with no evidence of false positive or inactivity. Resolving or dismissing is not allowed by preconditions. The best decision is to keep the finding under investigation to allow operators to assess and resolve the real issue.
Reasoning trace
1 [AI Agent] note written or updated in the finding notes.
Close
Run its recommendations...


Ask the AI Agent
UAM AI Agent
Inspect first
Finding group: account: jane.doe
Attention
SUMMARY
The finding regarding CSV lookups exceeding the size allowance for account jane.doe is medium severity and currently open. The account is classified as human based on interactive activity signals. The finding is observed with two oversized CSV files attributed to the account exceeding both individual file and cumulative size limits. The evidence supports that this is a true positive issue requiring operator review and action. Temporary dismissal or investigating status are allowed, but resolution or reclassification are not at this time.
FINDINGS ASSESSED (1)
True Positive account jane.doe, CSV lookups exceed size allowance CSV lookups attributed to an account exceed its size allowance High confidence
Decision: Mark investigating
Evidence: The account jane.doe is classified as human from detected interactive activity (web login and searches). The finding shows two large CSV files (approximately 661 MiB and 440 MiB) attributed to jane.doe that exceed the per-file size limit of 100 MiB and the cumulative account quota of 500 MiB. The evidence coverage is complete and the attribution is probable writer with fallback to recorded owner 'nobody'. The finding is observed and open at medium severity by policy.
This is a medium severity finding for jane.doe, a human account confirmed by detection. The two CSV files are oversized individually and cumulatively exceed the allowed quota, clearly consistent with the policy definition and evidence. The finding is still observed, so resolving is disallowed. Classification change is disallowed as the account class is correct and not from an operator override. Temporary dismissal is allowed but less appropriate than tracking with investigating to prompt further review by the operators. Recommend investigating to track and address the oversized CSV files according to policy remediation guidelines.
Reasoning trace
1 [AI Agent] note written or updated in the finding notes.
Close
Run its recommendations...

Who can ask:

- **User Activity Monitoring**: the TrackMe administrators of the tenant ask the agent and let it act; the tenant's power roles can read its settings and decision log.
- **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet**, **Splunk Forwarder Collectors**, **Cribl Stream** and **Cribl Edge**: the tenant's power and admin roles, the roles that can triage, ask the agent and let it act; the settings belong to a TrackMe administrator of the tenant.

Asking needs the AI Assistant to be enabled and at least one *AI provider* configured, on Splunk 10.2.x or later (the AI Agent is built on the Splunk Agent SDK, like the AI Advisors).

3.5.5 Scheduled triage

The AI Agent can review the open and investigating findings of its tenant **every hour** on its own. Scheduled triage is **off by default**. Its report only runs when all of the following hold:

- the AI Assistant is enabled and at least one AI provider is configured;
- the tenant is enabled and its AI Agent is enabled (workspace **Settings → AI Agent**);
- the tenant owner is a real account, not `nobody` or `splunk-system-user`. The Guardian check `ai_automation_owner_is_system_user` warns otherwise. On Cribl Stream, Cribl Edge and Splunk HEC the owner must also be a Splunk administrator (a member of `admin` or `sc_admin`), because the run records itself in a store only they can write; Splunk S2S, Splunk UF Fleet and Splunk Forwarder Collectors record their runs through TrackMe and need no such owner.

Saving the AI Agent settings schedules or unschedules the report at once; the tenant health tracker checks these conditions every five minutes and follows any other change, such as a provider added or removed.

Each run:

- reviews up to 10 findings (1 to 50), one per Finding group, the most severe first, within a time budget that fits the hourly cadence;
- skips a finding reviewed within the review interval (1 day by default), unless a new episode started or the Finding group changed;
- **never overrides a person**: a finding or Finding group that an operator decided on, or that has an assignee, is left to the operators;
- applies the shared filters of the automated AI reviews, with the finding severity as the priority: by default only **critical and high** findings are selected. The filter expression applies too, and so do the shared custom instructions of the tenant, which apply to scheduled runs only (see *AI Agents automation*); the agent's own custom instructions apply to every run (see *Custom instructions and Evaluate*).

A run never makes more than one decision per finding and Finding group, and every decision belongs to its run, its tenant and its scope.

3.5.6 Settings

In the workspace **Settings → AI Agent**. Every findings AI Agent also appears in **Manage AI Agents automation** (*AI Agents automation*), with its tenant's row: enabling, mode, provider, review interval, runtime and custom instructions; the allowed decisions stay in the workspace.

Setting	Default	Notes
Enabled	Off	Scheduled triage only; asking the agent needs only AI to be available.
Mode	Inspect	Inspect or act, for scheduled runs.
AI provider	First available	
Review interval (days)	1	A finding is not reviewed again within this interval unless it changed.
Maximum runtime (seconds)	3000	Fits the hourly cadence.
Findings per run	10	1 to 50.
Custom instructions	Empty	At most 4000 characters; see *Custom instructions and Evaluate*.
Allowed decisions and longest dismissal	See *Inspect and act*	Per decision; Cribl Stream adds the days of proven inactivity before stop monitoring.

Allowing more (enabling the agent, act mode, a decision, a longer dismissal) needs a licence that allows changes; restricting it never does.

3.5.7 Custom instructions and Evaluate

Custom instructions are plain English that the tenant's administrators write for their agent, in **Settings → AI Agent → Custom instructions** (also in Manage AI Agents automation and in the Configure tenant dialog). The agent follows them on **every run**, asked or scheduled: what to treat as expected, what to look at first, how to word its recommendations. For example: "The accounts named `svc_*` are service accounts." or "The nodes named `lab-*` are lab machines: their disk usage up to 80% is expected."

- At most 4000 characters; line breaks are kept.
- They **never widen what the agent may do**. The allowed decisions of the tenant and the checks TrackMe makes before every decision still decide: an instruction to dismiss a critical finding that is still failing is refused by TrackMe like any other.
- They come after the provider's custom prompt and, for scheduled runs, after the tenant's shared custom instructions of *AI Agents automation* (which apply to scheduled runs only): the agent's own instructions are the most specific.

Evaluate…, next to the field, tries a draft before you save it. It runs the agent in **inspect** mode with the draft in place of the saved instructions, on one open finding or on a review of the open findings (at most 10, the most severe first), and can run the same review **without** custom instructions side by side, so you see what the instructions change.

An evaluation **writes nothing**: no `[AI Agent]` note, no decision, no entry in the decision log. It is recorded in the audit with the status `evaluation`, and TrackMe refuses any decision or review from an evaluation run, whatever the model writes.



3.5.8 What the AI Agent did

Everything the agent does can be followed from four places:

- **The AI Agent tab** of the workspace: whether it can run and why, its totals (scheduled runs, findings reviewed, recommendations, actions taken, decisions TrackMe refused, inspect and act), **Run a review now**, and the decision log with its filters. Each row opens the investigation of that decision: the facts the agent read and its rationale.
- **The finding** itself: its history shows each decision under the actor **AI Agent**, and its notes hold the current `[AI Agent]` assessment.
- **The audit dashboards** of *Auditing AI activity*: *Review AI Advisor activity* and *Review Scheduled AI Advisor activity* include the seven agents under their labels, from **UAM AI Agent** to **Splunk Forwarder Collectors AI Agent**.
- **SPL**: every run is indexed to the tenant summary index with the sourcetype `trackme:ai_agent:<agent>:<inspect|act>`, where `<agent>` is `uam_agent`, `hec_agent`, `s2s_agent`, `uf_agent`, `fwc_agent`, `cribl_agent` or `cribl_edge_agent`, with `launched_by` (UI, AI Assistant or automation), `automated` and the chat session that triggered it, next to the per-tool events of `trackme:ai_agent:tool_call`.

```
index=trackme_summary sourcetype="trackme:ai_agent:*_agent:*"
| table _time, sourcetype, tenant_id, launched_by, automated, status
```





3.5.9 Data and privacy

The AI Agent follows the same rules as the rest of the AI layer (see *AI Compliance, Privacy and Data Governance*): nothing is sent to an LLM until an administrator configures a provider, and the agent sees only what its tools return, pinned to its tenant and its run.

- The evidence is computed by TrackMe from what the tenant already holds. No credentials, connection details or secrets are part of it.
- **Cribl Stream**: the agent cannot run searches and never calls Cribl; its facts come from the inventory, the traffic snapshots and the traffic models TrackMe already collected.
- **Cribl Edge**: the agent cannot run searches and never calls Cribl; its facts come from the node directory and the findings TrackMe already holds.
- **Splunk HEC**: the agent cannot run searches and never calls an endpoint; its facts come from what the tenant's collection published. Tokens are known by name, never by value.
- **Splunk S2S**, **Splunk UF Fleet** and **Splunk Forwarder Collectors**: besides the facts the tenant already holds, the agent can run only its tenant's own bounded read commands, never a search it writes, and it never calls a Splunk instance.

- **User Activity Monitoring**: the evidence never carries the text of a search. The agent may read a search text only when it needs it to conclude, and only for searches collected while the tenant's *Store SPL search* setting was on.
- The anonymisation of entity and index names in the AI settings does not apply to the AI Agent: account and object names are part of its evidence, so choose your AI provider accordingly.

3.6 AI Routines & Automation

Scheduled and agentic automation: user-authored **AI Routines**, the broader **AI Agents automation** surface, and **natural-language FLX** tracker generation.

3.6.1 AI Routines

Overview

AI Routines let you describe a monitoring task in plain English — and TrackMe watches for it on a schedule, then **notifies you or acts**, autonomously and under your explicit consent.

Where the *AI Assistant* answers when you ask, and the *AI Advisors* investigate when you (or the automation) dispatch them, an AI Routine is **your own standing instruction**: *"Tell me if 4 or more high priority feeds turn red within 15 minutes, correlate what they have in common, and add a note to each entity documenting the incident."* You write that sentence once; TrackMe compiles it into a structured instruction, runs it on a cron schedule of your choice, and delivers the outcome — an email or a Splunk event, and (if you granted it) real, audited actions.

Start from a template or from a blank intent — either way, TrackMe shows you exactly how your routine will run before anything is created:





Important

AI Routines share the AI Advisors' platform requirements: **Python 3.13.x** (Splunk 10.2.x and later) and at least one configured *AI provider*. The feature is **opt-in by construction** — it is completely inert until the AI Assistant is enabled and a provider is configured — and ships with its own kill switch (`enable_ai_routines`) and a creator-roles policy (administrators only, or administrators and power users) in the system settings.

The three evaluation strategies

Every routine picks how its condition gets evaluated — this is the main cost dial:



- **Agentic** — the LLM runs on every scheduled cycle and does everything: it reads the tenant, decides whether the condition is met, and acts. The most capable and flexible option (it reasons over live context each run), and the most token-intensive.
- **Hybrid** — a cheap, compiled **SPL trigger** runs first on each cycle; the LLM is only invoked when the trigger matches. When nothing is happening, a cycle costs **zero LLM tokens**. This is the recommended shape for tight schedules at scale.
- **Deterministic** — the trigger fully computes the condition; the LLM formats and delivers the outcome. The cheapest option for conditions that are precisely expressible in SPL.

The trigger is a **cost gate in front of the agent, never a replacement for it** — the LLM remains central to understanding, correlating, and explaining.

The strategy is chosen at creation time, and the form adapts to it — the inline explainer states the behaviour and the cost consequence, and the cadence hint under the cron field translates your schedule into runs per day so the cost implication is explicit before you save.

Agentic — the LLM evaluates every cycle



The LLM is the evaluation: on every cron tick it reads the live tenant context, reasons over it, and decides itself whether the condition is met. There is no trigger to author — the plain-English intent is the whole configuration. This is the right choice when the condition is fuzzy or judgement-based (*"anything unusual compared to the last runs"*), when it needs correlation that is hard to express in SPL, or while you are still shaping what "abnormal" means.

The cost consequence is structural: **every scheduled cycle is a full LLM run**, whether or not anything is happening. At `*/30 * * * *` that is 48 LLM runs a day — the form says so under the cron field. Tighten the cadence and the cost scales linearly with it.



An Evaluate dry-run makes the mode's behaviour concrete before you save: the *How this routine runs* pipeline shows the LLM in the evaluation seat, and the token callout extrapolates the measured per-run cost to your cadence (here ~4K tokens per run ≈ 196K tokens/day at every 30 minutes) — the honest verdict below (*Would not fire now*, with the reasoning) is the same judgement the model will make on schedule:



Hybrid — a trigger gates the agent



A fast, cheap Splunk search — the **compiled trigger SPL** — runs first on each cycle; the LLM is called *only* when that trigger returns rows. You keep the full agentic reasoning (once invoked, the agent investigates, correlates and narrates exactly as an agentic routine would), but you pay for it only when the condition is actually present. A quiet cycle costs **zero LLM tokens** — the run is recorded as a no-fire and the routine goes back to sleep.

Selecting hybrid reveals two extra fields:

- **Compiled trigger SPL** — authored by the Evaluate step from your intent (you can refine it); it fires when it returns rows, so the canonical shape ends in something like `| stats count | where count>=N`.
- **Trigger time window** — the default `-5m / now` is right for `| trackmegetcoll` (it reads current state, unaffected by time); widen `earliest` if your trigger is an index search on a longer cadence (e.g. `-24h`, or `@d` for since-midnight).

This is the cost-efficient choice at scale: a tight cadence (every 5 minutes, say) stays cheap because the vast majority of cycles are trigger-only Splunk searches.



On Evaluate, the pipeline gains the **Trigger search** stage in front of the LLM (*"runs first — skips when it finds nothing (no LLM cost)"*), and the token callout switches to the hybrid cost model: you pay the per-run cost **per matching cycle, not every tick**. In this dry-run the trigger matched, so the evaluation simulated the fire end-to-end (verdict *Would fire now*, with the would-be notification and act-mode writes described — nothing is actually sent or changed):



Deterministic — the trigger decides



Your trigger fully computes the condition; when it matches, the LLM does the minimum — it turns the trigger's result into the human-readable notification (and, in act mode, the consented follow-up actions) rather than re-deciding whether to fire. The same trigger SPL and time-window fields apply as in hybrid. The cheapest and most predictable strategy: firing behaviour is exactly your SPL, reviewable and testable like any Splunk search, with the LLM spend limited to the cycles that fire — best for a clear-cut, well-defined condition (*"any critical entity red for more than an hour"*).



The Evaluate dry-run shows the same gated pipeline — but here the trigger owns the decision outright, and the verdict follows your SPL, not the model's judgement (in this example the condition window had just lapsed, so the honest answer is *Would not fire now* even though the entities are still red — exactly what the configured 30-minute window dictates):



Choosing, in cost terms

At the same `*/30 * * * *` cadence (48 cycles/day), the three strategies diverge only in **how many of those cycles reach the LLM**: all 48 for agentic; only the matching ones for hybrid and deterministic (a quiet day costs zero tokens, a noisy day costs one LLM run per matching cycle — with deterministic runs staying individually smaller, since the agent formats an already-decided outcome instead of investigating). A practical path: start **agentic** while you shape the intent, then let Evaluate compile the trigger and switch to **hybrid** when the routine earns a tight schedule — and use **deterministic** when the condition is precise enough that you want SPL, not the model, to own the firing decision. See *Cost transparency* for how consumption is surfaced per routine and per run.

Notify or Act

A routine runs in one of two action modes:

- **Notify** (read-only) — the routine observes and delivers: an email, a structured Splunk event indexed to your summary index, or both at once. Recipients and targets are configured on the routine and delivered deterministically — they never depend on the LLM.
- **Act** — the routine may also **change things**: tune thresholds, update priorities, attach notes and labels, manage maintenance windows, retrain ML models, trigger Splunk alert actions (open incidents in your ITSM, notify Slack or PagerDuty), and more.

Act mode is governed by **explicit, granular consent**:

- You grant **permission families** — capability-oriented checkboxes such as *entities thresholds*, *entities priority*, *labels/notes/tags*, *ML model changes*, *trigger Splunk alert actions*, or *execute external MCP tools*. Each family exposes exactly its tools to the routine at fire time — nothing more.
- Consent is **pinned to what you approved**: the exact intent text, the exact set of granted families, and (where applicable) the exact alert-action and external-MCP contracts. Editing any of them invalidates the consent and the routine cannot act again until you re-affirm.
- Every write lands in the entity audit trail with an `[AI Agent]` stamp, exactly like the *AI Advisors*.

Switching the mode to **Act** reveals the consent checkbox and the write-permission families — sixteen capability-oriented grants, each exposing exactly its own tools. Tick only what the routine genuinely needs (least privilege), then affirm the consent:



Note

Two read capabilities are independent of the action mode and enabled by default: **read-only Splunk searches** (a bounded correlation search the routine can use to investigate) and **AI Advisor dispatch** (the routine can commission up to two inspect-only *AI Advisor* investigations per run when a deeper expert diagnosis is warranted). Both are reads — they never require consent, and both can be switched off per routine.

Where to find AI Routines

AI Routines live in two places, matching their scope:

- **Tenant Home** — routines scoped to that tenant (they read and act on this tenant only).
- **Virtual Tenants — global** routines that correlate **across tenants** of your choice.





Global configuration

The feature's deployment-wide switches live in *Configuration → General → Artificial Intelligence*:



- **Enable AI features** — the master switch for everything AI in TrackMe (AI Assistant, every AI Advisor, and AI Routines). AI Routines require it to be on.
- **Enable AI Routines** — switches the AI Routines feature specifically, independently of the other AI features. When disabled, the AI Routines menu entries are hidden from every TrackMe page, the create / evaluate / run-now REST endpoints reject requests, and any scheduled routine execution is refused. Default: Yes.
- **Who can create AI Routines** — controls the privilege level required to *create* routines. The default (**Admins and power users**) lets holders of the `trackmeadminoperations` or `trackmepoweroperations` capability create routines; **Admins only** restricts creation to `trackmeadminoperations`. Global (cross-tenant) routines always require an admin regardless of this setting. This applies to creation only — running, editing and deleting an existing routine follow the standard tenant RBAC.

- **AI Routines run history retention (days)** and **AI Routines memory depth (runs)** — the two memory levers, covered in *Memory between runs* below. Defaults: 7 days and 8 runs.

Creating a routine

Creation is a single form — with an important twist: **you cannot create a routine blind**. The **Evaluate** step is a mandatory dry-run gate for the LLM-driven strategies.

1. **Describe your intent** in plain English. Be as specific as you naturally would with a colleague: thresholds, time windows, priorities, what to do when it happens.
2. **Pick the evaluation strategy**, the cron schedule, the AI provider, and the notification channel (email or Splunk event).
3. For **act** routines: tick the permission families the routine may use, and affirm the consent.
4. **Evaluate** — TrackMe runs the routine against your live environment as a **read-only dry-run**: nothing is created, sent, or changed.





The Evaluate result is where the feature earns your trust before it earns its schedule:

- A **compiled instruction —** the structured spec the routine will run against (objective, watch condition, signals, notify policy, act policy, assumptions). This is what gets persisted and replayed on every run, so what you approve is exactly what executes.
- A **verdict —** would the routine fire *right now*, against your real data, and why.
- A **permission-coverage check —** if your intent implies an action the granted permissions cannot perform (say, the intent wants notes added but the notes family is not ticked), Evaluate tells you before you create it.
- A **cost preview —** the tokens this evaluation consumed, extrapolated to your chosen cadence, plus an expandable *Raw LLM input* panel showing byte-for-byte what is sent to the model.
- A **refine loop —** reply to the evaluation in plain English (*"use a threshold of 20 red entities; ignore orange"*) and re-evaluate until the compiled instruction says exactly what you mean.





Tip

Routines understand **movement**, not just current state. The tenant snapshot the agent reads carries each degraded entity's *time since last state change*, and a dedicated tool lets the routine list the exact **state transitions** (flips) since its previous run — including entities that turned red and recovered in between. Conditions like *"N entities turn red within M minutes"* are evaluated from the real flip history of your tenant, not guessed.

Email notifications

When a routine's notification channel is **Email** (or **Email & Splunk event**), every fire delivers a formatted email to the recipients configured on the routine. Two principles govern the delivery:

- **Recipients are configuration, not AI output.** The recipient list and the delivery account are set on the routine and used deterministically — the AI authors the *content*, never the audience.
- **Your email delivery accounts are honoured end to end** — the same accounts used by TrackMe's stateful alerting (transport and security settings, allowed recipient domains with out-of-domain recipients skipped, HTML vs text format preference, custom footer). The form pre-selects your first configured delivery account; the local MTA remains available as the default fallback.





The email arrives as **multipart**: a branded HTML rendering for modern clients and a plain-text fallback carrying the same content. The HTML shell shows the routine name with the tenant, scope and the AI's confidence as header rows, followed by the AI-authored summary and evidence rendered from Markdown — headings, bullet and numbered lists, tables, inline code, and section rules all render properly, and the LLM output is HTML-escaped before rendering as a security measure. The subject line is stable and filter-friendly: `[TrackMe AI Routine] <routine name> — condition met`.





In the example above, the mass-degradation routine fired and the email carries the full incident narrative — the affected feeds grouped by root-cause cluster, per-cluster status and likely root cause, and concrete check-lists for the operator — exactly as the agent wrote it.

Splunk event notifications

When a routine's notification channel is **Splunk event** (or **Email & Splunk event** — both channels at once, each delivered independently), every fire indexes **one structured JSON event** to the index and sourcetype you configure (defaults: `trackme_summary` / `trackme:ai:routines`), with the event `source` set to the routine id so per-routine searches are trivial.

The event always carries a **deterministic envelope** — `tenant_id`, `routine_id`, `routine_alias`, `scope`, plus the AI-authored `summary`, `evidence` and `confidence` — and the targets come from the routine's configuration, never from the AI. Your SPL, alerts and dashboards built on these events never break because a model phrased something differently.

Custom event fields

The optional **Customize event fields** panel extends that envelope under a strict contract — *you own the schema, the AI only supplies values*:

- **Static fields** — fixed values merged into every event (routing and enrichment constants such as `team=siem` or `env=prod`). The AI never sees them.
- **AI-filled fields** — you declare the field name, a one-line instruction, and optionally a list of allowed values; the AI fills the value when the routine fires. Only the declared names ever reach the event: anything else the model produces is dropped, values are length-capped, and a value outside your allowed list is blanked rather than passed through.





The panel shows a **live preview** of the event shape as you type (AI-filled values as placeholders), and the **Evaluate** dry-run renders the real thing — an *"Event it would index"* block built by the exact same code as production delivery, so what you approve is byte-identical to what a fire indexes.

In the example below, the operator declared an `anomalies` field with the instruction *"Add a dictionary of entities and their anomaly_reason value"* — the fired event carries a compact `entity: reason; entity: reason` digest alongside the envelope, ready for downstream parsing:



Tip

Keep AI-filled values **short and typed** — a severity, a region, a count, a compact top-5 list. For **bulk per-entity data** (say, every entity's full `status_message`), don't use an event field: join it at search time instead, deterministically and at zero AI cost:

```
index=trackme_summary sourcetype=trackme:ai:routines
| join type=left tenant_id [ | trackmegetcoll tenant_id=... component=... |␣
↪fields object, status_message ]
```

What runs, as who

Each routine is materialised as **its own Splunk scheduled search** running on the routine's cron — no shared orchestrator, full failure isolation, and everything is visible with standard Splunk tooling. Splunk RBAC applies to everything the routine reads and does: the execution identity decides which data the routine can search, which entities it can touch in act mode, and whether the routine survives personnel changes.

Ownership is an **explicit choice at creation time**. When the tenant has a real service-account owner assigned (Virtual Tenants → three-dot menu → RBAC), the wizard presents the identity as a two-option selector:

- **Run as the service account** (default, recommended for scheduled routines) — the routine is independent of any personal account: it keeps running if you leave, change roles, or your account is disabled.
- **Run as me** — the routine uses your own permissions; it stops working if your account is disabled.



When the tenant has **no service-account owner** (owned by `nobody` / a system user — good Splunk hygiene for knowledge objects, but a Splunk AI agent cannot execute as the system user), there is no choice to offer: the wizard states it plainly and the routine runs as its creator. Assigning a service account to the tenant later and re-saving the routine moves it over:



Global (cross-tenant) routines always run as their creator — there is no single tenant whose service account could span the scope.

The resolved identity is never hidden: the management view carries an **Owner** column per routine, so an admin can audit at a glance which identity each routine executes with:



Managing routines

The management view lists your routines with their strategy, mode, provider (switchable in place), schedule state, and last-run outcome. From the row menu you can edit, enable/disable, delete, review the run history, or **run now** — a fire on demand with live progress:





A run ends in one of two honest outcomes — **No fire** (the condition is not met; the routine stays quiet, no notification noise) or **Fired** (with the action taken and the notification status). The run history keeps both, along with per-run duration and token counts:





Note

The run history is a lightweight operational view retained for **7 days**. For long-range review, use the *Review AI Routines activity* dashboard below — the underlying events are indexed to your summary index and follow your own retention policy.

Memory between runs



Routines remember. Every LLM-evaluated cycle ends with the agent writing a **state fingerprint** — a compact, comparable snapshot it authors itself (for example *"DSM red=12/112; firewall cluster amer/apac; NOTIFIED"*). On the next cycle, the most recent fingerprints are replayed to the agent as the routine's memory. This is what powers the anti-noise behaviour you see in practice: *notify once when the condition is newly met, stay quiet while the same entities remain red, re-notify when it clears and recurs or when the affected set changes materially.*

Because the fingerprint is the agent's own scratchpad, your intent can lean on it: *"track the worst-offending entity across runs and tell me when it changes"* works — the agent encodes what it needs to remember into the fingerprint. For **deeper history** than the recent fingerprints (say, *"compare this week's incident count with last week's"*), phrase it

in the intent: every fire's full narrative is indexed, and the agent can search it with its read-only correlation search like any other Splunk data.

Per-entity memory. The fingerprint is a narrative; entity-level intents are *set* questions — and those get their own dedicated memory. When a fire concerns specific TrackMe entities, the agent also records their **exact entity names** with the run (`affected_objects`). On later cycles, the recent fires are folded into a single deduplicated *recently fired entities* set with per-entity ages (*"entity X fired 32m ago"*) and handed back to the agent. This makes intents like *"do not re-notify for the same entity within 4 hours"* or *"only alert on entities that were not part of the last fire"* deterministic — decided on exact names, never on fuzzy recall from the fingerprint text. It is also token-frugal by design: quiet cycles and use cases that don't deal with entities carry no per-entity memory at all.

Two storage layers, should you want to investigate manually:

- **Run history (short-lived operational index)** — the KV Store collection `kv_trackme_ai_routine_runs`: one row per evaluated run with the outcome, fingerprint, affected entities, tokens, duration and error. Retained for **7 days by default**. This also answers *"which runs touched entity X?"* directly from SPL:

```
| inputlookup trackme_ai_routine_runs | search routine_id="<routine_id>" |␣
↪sort - cycle_at
```

- **Indexed run events (long-range, follows your index retention)** — every fire's full summary and evidence in the tenant's summary index:

```
index=trackme_summary sourcetype=trackme:ai_agent:ai_routine:*
```

Both memory levers are **tunable** under *Configuration → General → Artificial Intelligence* for deployments whose intents lean intensively on memory: ai_routines_run_retention_days (default 7, 1–90) controls how long the run rows above are kept, and ai_routines_memory_depth (default 8, 0–24) controls how many recent fingerprints are replayed to the AI on each cycle — each remembered run costs roughly 60 extra prompt tokens on every cycle.

Note

The memory has honest bounds: it is **per-routine** (routines do not read each other's fingerprints), it replays the most recent runs only, and **hybrid quiet cycles leave no memory** — when the trigger does not match, no LLM run happens at all (that is the cost model working as intended), so the memory timeline is "every LLM-evaluated cycle", not "every cron tick".

Cost transparency

Token consumption is a first-class citizen throughout: the routine list shows max/avg/last tokens per routine, the run history carries per-run counts, Evaluate extrapolates a daily cost for your cadence, and the dedicated **cost view** breaks consumption down over time:



A few cost habits that pay off:

- Prefer **hybrid** for tight schedules — quiet cycles cost zero LLM tokens.
- Right-size the cadence: an agentic routine at `*/30` runs the LLM 48 times a day whether or not anything happened.
- Use the **Raw LLM input** panel (Evaluate) to see exactly what your routine sends — and keep the intent focused; it is replayed verbatim on every run.

Auditing routine activity

Everything a routine does is observable. The **Review AI Routines activity** dashboard (Artificial Intelligence menu) charts runs by routine, outcome, and cadence, and lets you drill into each run — alongside the structured events in the summary index and the per-routine execution logs:





See also *Auditing AI activity* for the AI-wide audit surfaces, and *AI Agents automation* for the scheduled advisor batches — AI Routines are user-authored and complementary to those product-defined automations.

3.6.2 AI Agents automation

Overview

Every specialised *AI Advisor* can run **unattended** on a nightly schedule, in addition to the interactive on-demand path. The automated batch selects eligible entities for a tenant, invokes the relevant advisor (typically in **act** mode) against each, and records the outcome — all scoped by per-tenant policy that you control.

Automation is configured from **Artificial Intelligence > Manage AI Agents automation** and observed from **Artificial Intelligence > Review Scheduled AI Advisor activity**.

Important

Automation is **disabled by default** for every tenant. Nothing runs unattended until you explicitly enable the ML Advisor and/or the Components Advisor for a tenant and pin them to a provider. The decommission guard (the ability to disable entities automatically) is **also off by default**.

Note

The *AI Concierge* has **no** automated path — it is interactive-only by design. Automation applies to the five specialised advisors only.

Note

The *AI Agents for findings* of the seven dedicated tenant types (User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) have their own **hourly scheduled triage**, configured in the workspace of each tenant (*Settings → AI Agent*). They also appear on the **Manage AI Agents automation** page: a findings tenant shows its own agent (the UAM AI Agent, the Splunk Forwarder Collectors AI Agent, and so on) in place of the ML and Components advisors, with its enable toggle and a tab for its core settings (mode, provider, review interval, runtime and the agent's own custom instructions); the decisions the agent may take stay in the workspace. The shared scope filters of this page (priority filter, filter expression)

apply to them too, with the finding severity as the priority; the shared **Custom Instructions** of this page apply to their scheduled runs only, while the agent's own custom instructions apply to every run.





Managing automation per tenant

The **Manage AI Agents automation** page lists every tenant with its automation state. Two families of automation are configured independently:

- **ML Advisor** — the model-inspection advisor, with its own enable toggle and pinned provider. It reviews the ML outlier models of every component that has them (DSM, DHM, VOL, FLX, FQM, WLK).
- **Components Advisor** — the unified configuration advisor covering **Feed Lifecycle** (DSM/DHM), **FLX Threshold** (FLX), **FQM Advisor** (FQM), and **Component Health** (WLK/MHM), with its own enable toggle and pinned provider.

Each row also shows the **Automated Filters** (the scope of eligible entities) and any **Custom Instructions** in effect.





Opening a tenant's **AI Agent settings** reveals the per-section configuration across tabs — **Filters**, **ML Advisor**, **Components Advisor**, **Custom Instructions**, and **Concierge**. The Components Advisor tab, for example, enables the automated review and lets you scope which components are eligible.


AI Agent settings — secops-v1-sta
Filters
ML Advisor
Components Advisor
Custom Instructions
Concierge
AUTOMATED ML MODEL INSPECTION
Enable Automated ML Model Inspection
Disabled
Cancel
Save changes


AI Agent settings — secops-v1-sta
Filters
ML Advisor
Components Advisor
Custom Instructions
Concierge
AUTOMATED COMPONENTS AI ADVISOR
Enable Automated Components AI Advisor
Disabled
Cancel
Save changes


AI Agent settings — secops-v1-sta
Filters
ML Advisor
Components Advisor
Custom Instructions
Concierge
CUSTOM INSTRUCTIONS
Tenant level Custom Instructions
Cancel
Save changes


AI Agent settings — demo-outliers
Filters
ML Advisor
Components Advisor
Custom Instructions
Concierge
CUSTOM INSTRUCTIONS
Tenant level Custom Instructions
always save investigations and actions taken as a note to the entity
Cancel
Save changes





Scoping and cost control

The batch honours a set of per-tenant gates — the same gates are the primary cost-control knobs for billed providers:

Gate	What it does
Master switches	`ai_mladvisor_enabled` (ML) and `ai_components_advisor_enabled` (Components). When off, the scheduled search scans and exits — no entities selected, no tokens consumed.
Eligible components list	`ai_components_advisor_list` — restrict the Components Advisor to specific components (dsm, dhm, mhm, flx, fqm, wlk).
Priority filter	`ai_automated_priority_filter` (default `critical,high`) — only entities in the listed priorities are reviewed. An optional `ai_automated_filter_expression` narrows further.
Freshness window	`*_min_days_between_reviews` (default 30) — an entity reviewed within the window is skipped, so the same entity is not re-processed every night.
Per-entity opt-out	`ai_mladvisor_disabled=1` on an entity excludes it from the batch (useful after manual tuning).
Decommission guard	`ai_components_advisor_allow_decommission` / `ai_mladvisor_allow_model_disable` (both off by default) — block automatic disabling of entities / models even when act mode is on.
Runtime cap	`*_max_runtime_sec` (default 14400s = 4h) — a per-batch wall-clock ceiling.

Tip

The priority filter and the freshness window compound multiplicatively. Tightening from `critical,high` + 30 days to `critical` + 60 days typically cuts the token bill by 6–10×. Schedules are staggered across advisors so a tenant does not fire every batch at once and exhaust provider rate limits.

Reviewing scheduled activity

The **Review Scheduled AI Advisor activity** dashboard answers one operational question: *is the automation actually firing for every tenant?* Every panel filters to scheduled (`automated=true`) runs only — interactive runs appear in the *Review AI Advisor activity* dashboard instead.





It surfaces KPI cards (total scheduled runs, success / error rate, total tokens, average run duration, active tenants), breakdowns by advisor / status / tenant / provider-model, runs-over-time charts, an MCP tool-call leaderboard, and the recent scheduled runs — so you can confirm, for example, that the ML Advisor's nightly cron is firing in the expected hours and that no tenant's automation has gone silent.





Each nightly batch produces a per-entity terminal audit event per advisor run, all the per-tool-call events, one `[AI Agent]` audit comment per write, and a batch summary — exactly the same audit surface as interactive runs, marked with `automated=true`. See *Auditing AI activity*.

Related

- *The AI Advisors* — the advisors that run unattended.
- *Auditing AI activity* — the interactive AI Advisor and AI Assistant audit dashboards.
- *Managing tenants* — managing virtual tenants.

3.6.3 Natural-language FLX tracker generation

Overview

Flex Objects (FLX) is the most powerful TrackMe component — it turns *any* Splunk search into a set of tracked entities with custom KPIs, thresholds and anomaly detection. It is also the one with the steepest learning curve, because you have to author SPL that satisfies the FLX *search contract*.

Natural-language FLX tracker generation collapses that curve. From the Flex Objects tracker-creation wizard you describe *what you want to monitor in plain language*, and the *AI FLX Threshold Advisor* drafts the tracker for you: it grounds the SPL on the closest of the *60+ Flex Objects use-case templates*, proposes scheduling and starter thresholds, **dry-runs the search against your live data**, and fills the wizard with a validated proposal you review before anything is created.

This is the showcase of TrackMe's AI layer: it does not just *detect* problems, it now helps you *build* the monitoring in the first place — grounded in real data, with the operator firmly in the loop.

Note

Trust by design. The AI never silently creates anything. SPL hallucination is structurally mitigated by two rules: every proposal is **grounded on the use-case template library** rather than written from scratch, and the candidate search is **dry-run against your live data before you ever see it** — you always review real entity counts and sample results first. The advisor only *proposes*; the tracker is created through the wizard's own create step, with your credentials, on your explicit click.

Requirements

- An **AI provider** must be configured — the feature (like the whole AI layer) is opt-in and inert until then. See *Configuring AI Provider accounts*. The **Generate with AI** button only appears once at least one provider is available.
- The advisor runs on the same **Splunk 10.2.x+ / Python 3.13.x** requirement as the rest of the *AI Advisors*. See *Artificial Intelligence in TrackMe* for compatibility.
- Your role must be allowed to create Flex Objects trackers in the tenant (the create step uses your own credentials, exactly as the manual flow does).

Where to find it

Open the **Flex Objects tracking — create a new tracker** wizard (Tenant Home → Flex Objects → create a new tracker) and go to step **2 — Use case & search logic**. Alongside the manual options (pre-load a use case from the library, or write the search by hand) you will find the **Generate with AI** button.



Walkthrough

1. Choose a provider and describe what to monitor

Clicking **Generate with AI** opens the generation panel as a side drawer, so the wizard stays visible underneath. At the top, pick the **AI provider / model** to use for this run — a deliberate choice, because drafting and dry-running SPL is a token-heavy operation and different models trade off cost against quality.

Then describe, in plain language, **what you want to monitor**: the data source, the metric(s) that matter, what "bad" looks like, and any silence/inactivity concern. The tracker settings shown as controls below the description (**account**, **earliest**, **latest**, **cron**, **name prefix**) are the starting constraints the dry-run will use — they are fully editable here, exactly as if you set them on the wizard's own steps.



In this example the operator asks to *monitor the number of SSH sessions* from a sourcetype, **per index**, and to **use outlier detection to flag an abnormal number of sessions**.

2. Generate

Click **Generate tracker**. The agent works in front of you: a live tool-call feed shows it pulling the closest use-case template (`get_flx_use_case_definition`) and then **dry-running** the candidate search (`run_flx_tracker_dry_run`). Each dry-run is capped and read-only — you can **Cancel run** at any time.



3. Review the validated proposal

When the agent finishes you get a complete proposal whose evidence comes from the **last successful dry-run of the exact SPL being proposed** — never a proposal without live results. It includes:

- a **summary** of what the tracker monitors and which template it was grounded on;
- the **dry-run evidence** — the number of entities found and a sample of them (object / group / status / metrics), so you can confirm the search behaves as intended *before* creating it;
- the **proposed tracker search** (the SPL), and grounding/schedule chips (template used, account, cron, earliest, latest, name).



Scrolling down, the proposal also surfaces **starter threshold recommendations** and **confidence notes** (what was assumed, what could not be verified — e.g. a field name that may need adjusting), and a collapsible **reasoning trace**.





Note

The **proposed thresholds are recommendations to apply after creation —** the Flex Objects create step does not take thresholds directly. They give you a calibrated starting point you (or the *AI FLX Threshold Advisor* later) can apply on the new entities.

4. Refine (optional, repeatable)

If the proposal is close but not quite right, you do not start over. Use **Not quite right?** at the bottom of the result: type what should change (for example *"use a different field for the distinct count"* or *"aggregate over 15 minutes instead of 5"*) and click **Re-generate with feedback**. The agent **iterates on the previous proposal —** keeping everything you did not contest, applying your feedback, and re-validating with a fresh dry-run. Repeat until you are happy.

5. Apply to the wizard

When the proposal is right, click **Apply to wizard form**. The drawer closes and the wizard jumps to **step 4 — Test and Configure**, pre-filled with the proposed search, metrics, default metric and schedule, and it **re-runs the wizard's own simulation** so you immediately see the live entities in the wizard's own panel.



From here it is the **standard Flex Objects creation flow** — review the metrics and thresholds, adjust anything you want, set the cron schedule, and create the tracker. Nothing about creation is AI-driven; you are simply finishing the wizard you would have used manually, with the hard part already done. See *Creating a Flex Tracker* for the manual steps.

6. The tracker is live

Once created, the new Flex Objects tracker behaves like any other — its entities appear on Tenant Home, are scored and alertable, and participate in the full TrackMe feature set.



Because the description in this example asked for **outlier detection**, the generated tracker is configured for it: the entity overview shows the live KPI, and the **Outliers anomaly detection** tab shows the native ML model already tracking the per-index session counts.





Good to know

- **The AI proposes, you create.** The advisor never calls the tracker-create endpoint. Creation always happens through the wizard's own create step, with your credentials — so license limits, name sanitisation and the audit trail all flow through the proven path.
- **Always dry-run-validated.** A proposal's entity counts and samples always come from a real dry-run of the exact SPL being proposed. If the data is sparse or a field name is uncertain, the agent says so in the confidence notes rather than hiding it.
- **Editable settings.** Account, time window, cron and name prefix are editable in the panel and mirror straight into the wizard form. The AI may still adjust the schedule/window to fit the chosen use case; whatever the final

proposal returns is what **Apply** writes.

- **Provider choice matters.** Generation is token-heavy and iterative — pick the model that fits your cost/quality preference at the top of the panel.
- **Interactive only.** Tracker generation is a wizard-time, human-in-the-loop capability — there is no unattended/automated generation path. (The *AI Advisors automation* schedule covers inspect/act remediation, not creation.)
- **Audited.** Like every AI action, generation runs are recorded. In the **Review AI Advisor activity** dashboard they appear under their own **FLX Generation** category (distinct from FLX threshold tuning). See *Auditing AI activity*.

Related

- *The AI FLX Threshold Advisor* — the AI FLX Threshold Advisor that powers generation and tunes thresholds on existing entities.
- *Flex Objects - Adapt TrackMe to any monitoring use case (splk-flx)* — the Flex Objects component itself.
- *FLX (Flex Objects) Use Case Template Catalog* — the use-case template library the generation grounds on.
- *Creating a Flex Tracker* — the manual tracker-creation flow.
- *AI Assistant* — the AI Assistant, which can also propose advisor runs in chat.
- *Auditing AI activity* — auditing AI activity.

3.7 Governance

How to audit what TrackMe's AI did, the privacy, data-governance and compliance posture of the AI features, and how TrackMe positions against the EU AI Act.

3.7.1 Auditing AI activity

Overview

Auditability and traceability are first-class principles of the TrackMe AI layer. Every AI Assistant chat, every AI Advisor run, and every Concierge action is indexed to the tenant summary index and surfaced in dedicated audit dashboards — so you can always answer *who launched what, which provider and model were used, what was proposed or changed, and how long it took*.

There are four audit dashboards under the **Artificial Intelligence** menu:

- **Review AI Advisor activity** — interactive (UI- or chat-launched) advisor runs, and the runs of the seven *AI Agents for findings*, one per dedicated tenant type (UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge AI Agents).
- **Review Scheduled AI Advisor activity** — the automated nightly batch (covered in *AI Agents automation*) and the hourly scheduled triage of the findings AI Agents.
- **Review AI Routines activity** — user-authored scheduled routines, interactive and scheduled.
- **Review AI Assistant activity** — chat conversations and stateful-alert AI status reports.

Note

In addition to the dashboards, every advisor or Concierge **write** is stamped with an `[AI Agent] <reason>` prefix in the affected entity's *Audit changes* panel, so the per-entity history shows exactly what the AI changed and why — independently of the dashboards. The findings AI Agents record their decisions in the finding history under the actor **AI Agent** and in the **AI Agent** tab of the workspace (see *AI Agents for findings*).

Review AI Advisor activity

This dashboard gives end-to-end visibility into AI Advisor activity across every tenant — who launched what, which providers and tools were used, and how long each call took. It covers the **interactive** path; scheduled runs live in the *Review Scheduled AI Advisor activity* dashboard.

The KPI strip shows total runs, success / error rate, total tokens, average run duration, and active tenants. Donut charts break activity down by advisor, by status, by mode (inspect / act / dictionary_generate), and by top provider-model. A tool-call leaderboard and a recent-runs table complete the picture.





The **MCP tool-call leaderboard** shows every tool the advisors invoked, with call counts, average and p95 durations, and success / error counts — useful for spotting a slow or failing tool. The **recent advisor runs** table lists each run with its advisor, mode, status, tenant, component, object, user (or `automated`), duration, tokens, and the number of recommendations and actions taken.



Tip

Every panel has a 3-dot menu with **Open in Search** that deep-links into the Splunk Search app with the panel's SPL pre-loaded. The underlying events use the sourcetype pattern `trackme:ai_agent:*advisor*` — and `trackme:ai_agent:*_agent:*` for the findings AI Agents (plus `trackme:ai_agent:tool_call` for per-tool events), so you can pivot AI activity in plain SPL.

Review AI Routines activity

This dashboard gives end-to-end visibility into *AI Routines* — the user-authored, scheduled natural-language tasks that watch a condition and then notify or act. It combines **interactive** runs (Evaluate dry-runs and Run-now) with the **scheduled** fires, so you can see how often each routine ran, whether it fired, what it cost, and how it was launched.

The KPI strip shows total runs, success / error rate, total tokens, average run duration, and active tenants. Donut charts

break activity down by routine, by status (fired / no-fire / error), by top tenants, and by top provider-model. Runs-over-time column charts (by routine and by status) and an hour-of-day cadence histogram make it easy to confirm a routine's schedule is actually firing in its window, and a per-tenant × routine last-run matrix highlights any routine that has gone silent.





Each routine fire is indexed to the tenant summary index with the sourcetype pattern `trackme:ai_agent:ai_routine:*` (carrying `advisor="ai_routine"`), alongside the same per-tool `trackme:ai_agent:tool_call` events the advisors emit — so the routine's reads and writes are auditable in plain SPL, and each act-mode write is additionally stamped `[AI Agent] <reason>` in the affected entity's *Audit changes* panel. See *AI Routines* for the routine model itself and its per-routine execution logs.

Review AI Assistant activity

This dashboard covers the conversational AI Assistant — both interactive chats and the AI status reports embedded in stateful-alert emails. It records who launched each chat, which provider and model answered, the context it ran in, and how long it took.

The KPI strip shows total runs, success / error rate, skipped status reports, total tokens, average duration, and active tenants. Donut charts break activity down by kind (interactive chat vs status report), by status, by context (entity / Tenant Home / wizard / feature pages), and by top provider-model. A status-report skip-reasons panel explains any skipped email reports.





The **recent runs** table lists each chat or status report with its kind, context, status, tenant, object, provider-model, user, duration and token usage; a **recent errors** table isolates failures. The underlying events use the sourcetype pattern `trackme:ai_assistant:*`.

Note

Cancelled chats are deliberately **not** indexed (the user walked away — counting them would distort the error rate). Non-entity chat contexts (e.g. Virtual Tenants) are bucketed under a synthetic `system` tenant so the by-tenant breakdown stays stable.

The audit envelope

All AI audit events share a common envelope so the dashboards can aggregate without parsing JSON. Useful fields include:

Field	Meaning
`tenant_id`	The tenant the event belongs to (`system` for non-entity contexts).
`advisor` / `kind`	The advisor name (Advisor events), `ai_routine` (Routine events), or `chat` / `status_report` (Assistant events).
`mode`	`inspect` / `act` / `dictionary_generate` for advisors; `inspect` (notify) / `act` for routines.
`status`	`success` / `error` / `skipped`.
`launched_by`	`ui` / `ai_assistant` / `automation` — who triggered the run (routine fires on a schedule report `automation`).
`automated`	`true` for scheduled batch runs, `false` for interactive.
`provider` / `model`	The resolved provider account and model.
`duration_ms` / `*_tokens`	Wall-clock time and token usage.
`chat_session_id`	Correlates a bridge-launched advisor run with the chat conversation that triggered it.

Tip

To find which conversation triggered a given advisor run:

```
index=trackme_summary sourcetype="trackme:ai_agent:*advisor*"
  launched_by="ai_assistant" chat_session_id="chat-XXX-yyy"
```

Related

- *The AI Advisors* — the advisor runs recorded here.
- *AI Agents automation* — the scheduled-activity dashboard for unattended runs.
- *AI Routines* — the user-authored scheduled routines audited here.
- *AI Assistant* — the AI Assistant whose chats are audited here.

3.7.2 AI Compliance, Privacy and Data Governance

Overview

This page is written for **compliance, security, privacy and risk teams** evaluating the use of TrackMe's AI capabilities in regulated or data-sensitive environments. It answers, in one place, the questions that matter for an AI risk assessment:

- *Where does the AI processing happen, and who is in the data path?*
- *Exactly what data is exchanged with the LLM provider — and what is never sent?*
- *What can the AI actually do — which tools does it have, and which are read-only?*
- *How are write actions controlled, audited and reversed?*

- *How do we restrict, monitor, or completely disable the AI layer?*

Every statement on this page reflects how the product is implemented. For the functional walkthroughs of each AI capability, see *Artificial Intelligence in TrackMe* and the dedicated pages of this section; for provider configuration, see *Configuring AI Provider accounts*. For the mapping of these capabilities to the **EU AI Act**'s roles, risk categories and Article 50 transparency obligations, see *EU AI Act compliance*.

Important

The short version — TrackMe's AI layer is **opt-in and inert by default**: no LLM call is ever made until an administrator explicitly configures an AI provider account. When enabled, LLM calls go **directly from your Splunk search head to the provider you chose —** including fully air-gapped self-hosted models — using **your own credentials**. TrackMe Limited (the vendor) operates **no AI cloud service, no proxy, and no telemetry collection** in the AI data path and never receives your data. The AI works on **monitoring metadata and telemetry**, not on your raw indexed events, and every interaction is **recorded in an audit trail you control**.

Compliance at a glance

Question	Answer
Is AI enabled by default?	**No.** The AI layer is completely inert until an administrator creates at least one AI provider account. Until then, no data leaves Splunk and no LLM call is made.
Is the vendor (TrackMe Limited) in the data path?	**No.** TrackMe is a Splunk application running entirely inside your Splunk deployment. LLM requests are sent directly from your search head to the provider endpoint you configured. There is no intermediate TrackMe-operated service, and TrackMe Limited never receives any of your data, prompts, or responses.
Can we keep all AI processing on-premises?	**Yes.** TrackMe supports fully self-hosted inference via Ollama and any private OpenAI-compatible endpoint (e.g. NVIDIA NIM, LM Studio, corporate LLM gateways) — zero external API calls. On Splunk Cloud, the **Splunk Hosted (SLIM API)** provider keeps processing inside the Splunk platform. See *Configuring AI Provider accounts*.
Is our data used to train models?	TrackMe sends requests to **API endpoints** using **your own API keys**, under the data-use terms of your agreement with the provider. Major providers (OpenAI, Anthropic, Azure OpenAI, Google, Mistral) contractually do not train on API traffic by default. TrackMe itself never stores or reuses your prompts for any training purpose. See *Provider choice, data residency and model training*.
Are raw Splunk events sent to the LLM?	**No, with one narrow, documented exception.** AI context is built from TrackMe's own monitoring metadata and telemetry (entity state, thresholds, metric histories, audit records). The only feature that can expose event samples is the FQM Advisor's sampled-failure inspection, scoped to field-quality diagnosis. See *What is sent to the LLM — and what is not*.
Are credentials or secrets ever sent to the LLM?	**No.** API keys, Splunk credentials, session tokens and stored passwords are never part of any prompt or context payload. Provider API keys are stored encrypted in Splunk's `storage_passwords` and are write-only once saved.
Can the LLM run arbitrary searches in our environment?	**No.** The LLM never generates raw SPL for execution. All data access goes through a fixed registry of typed tools whose SPL and REST calls are pre-built and parameter-validated by TrackMe. See *The complete tool inventory*.
Can the AI change things without a human?	**Not unless you explicitly opt in.** Interactive advisor/Concierge writes always require an explicit user consent click (executed under the user's own credentials and RBAC). Unattended advisor automation is **disabled by default**, per-tenant, and protected by additional guards (e.g. the agent can never decommission an entity or disable a model unless you enable a dedicated flag). **AI Routines** — user-authored scheduled tasks — are notify-only unless the author selects *act* mode and grants explicit capability-scoped permissions under a **hash-pinned consent** captured at creation; editing the intent or the grants invalidates the consent and the routine cannot act again until re-affirmed. See *AI Routines: user-authored automation, explicitly consented*.
Is AI activity audited?	**Yes, comprehensively.** Every chat, every advisor run, and every individual tool call is indexed to your Splunk environment with user, provider, model, token usage, duration and outcome. Every write is additionally stamped in the per-entity audit trail with an `[AI Agent] <reason>` justification. See *Auditing AI activity*.
Can we restrict who uses AI?	**Yes.** AI endpoints are gated by TrackMe's standard role model (read / power / admin capabilities, per-tenant role scoping), enforced at the REST boundary. See *Access control (RBAC)*.
Who controls retention of AI audit data?	**You do.** AI audit events are indexed to your tenant summary index and follow your Splunk index retention policies; no AI data is retained outside your Splunk deployment by TrackMe.

Architecture and data flow

Where the AI runs

TrackMe's AI layer runs **entirely inside your Splunk deployment**:

1. A user (or a scheduled automation you enabled) triggers an AI interaction in TrackMe.
2. The TrackMe backend, running on your **search head**, builds the request: a system prompt, the relevant monitoring context, and — for agentic features — the typed tool registry.
3. The search head makes a **direct HTTPS call** to the AI provider endpoint configured in *Configuring AI Provider accounts*, authenticated with **your** API key (or no key at all for self-hosted Ollama and Splunk Hosted).
4. The response is rendered to the user, and the interaction is **audited** to your tenant summary index.

There is **no third-party relay**: the only parties in the data path are your Splunk platform and the AI provider **you** selected. If you select a self-hosted provider, no data leaves your network at all.

Note

TrackMe Limited never receives your data. The vendor operates no AI backend, collects no AI telemetry, and has no technical ability to observe your prompts, your monitoring data, or the provider responses. The AI provider relationship — endpoint, credentials, region, data-use terms — is entirely between your organisation and the provider you configure.

Agentic execution model

The conversational **AI Assistant** is a single-call interface: TrackMe builds a context payload (see below), sends one request, and renders the answer. The assistant **executes no tools and performs no writes** — when an action would help, it can only *propose* one via a consent card that the user must explicitly approve.

The **AI Advisors** and the **AI Concierge** are agentic: they run a bounded reasoning loop (built on the Splunk Agent SDK, `splunklib.ai`) in which the LLM may call tools from a **fixed, tag-filtered registry**. The tools — not the LLM — perform all data access, through TrackMe's own REST API and pre-built searches, under Splunk authentication. Every tool invocation is individually audited.

The **AI Routines** are agentic on the same SDK and tool registry, but are **user-authored and scheduled** rather than product-defined: an operator describes a monitoring task in plain English and TrackMe runs it on a cron schedule. They are the one surface where a write can occur **on a schedule without a per-action click** — so their governance is described in full below.

AI Routines: user-authored automation, explicitly consented

AI Routines let an operator schedule a natural-language task that watches a condition and then **notifies or acts**. Because a routine can act autonomously on its schedule, it is governed by a distinct, stricter consent model than the interactive advisors:

- **Notify by default, act only on opt-in.** A routine in **notify** mode has **no write tools** — it can only deliver a message. Writing requires the author to switch it to **act** mode and grant specific permissions; a notify routine can never change configuration.
- **Capability-scoped permission families.** Act permissions are granted as **16 capability-oriented families** (e.g. entity thresholds, priority, labels/notes, ML models, alert actions). Ticking a family exposes **exactly** its tools at fire time and nothing more — there are no implied grants.
- **Hash-pinned, perishable consent.** At creation the author affirms an explicit consent that is **cryptographically pinned** to the verbatim intent, the exact set of granted families, and — where applicable — the exact alert-action contract and the external-MCP tool-name list. **Any edit to those invalidates the consent**, and the routine cannot act again until the author re-affirms. The consent is checked on every fire.

- **Deterministic, non-LLM delivery.** Recipients (email) and event targets (index/sourcetype) are configuration, never model output — the LLM authors the *content*, never the *audience*. Alerts and SPL built on routine events cannot be redirected by a model phrasing something differently.
- **Off-box side effects are gated.** Two families reach outside TrackMe's own KV Store — *trigger Splunk alert actions* (open incidents / notify ITSM, Slack, PagerDuty) and *execute external MCP tools* (the Splunk MCP Server). Both are act-only, part of the pinned consent, and enforced at the Splunk boundary: the MCP token is minted for the routine's **owner** under Splunk RBAC, exposing only the explicitly allow-listed tool names.
- **Identities and tenant scope are never model-authored.** A routine runs as a resolved **owner** (the tenant's service account, or the creator) recorded on every event; the tenant it may touch is fixed from the routine's own scope, not the LLM — a tool call targeting any other tenant is refused before it executes.
- **Who may create routines is policy-controlled.** A deployment-wide creator-roles setting restricts routine creation to administrators, or to administrators and power users; global (cross-tenant) routines always require an administrator. The whole feature has its own kill switch (`enable_ai_routines`) and is inert until the AI layer is enabled and a provider configured.

Every routine fire — reads, writes and the fire/no-fire decision — is audited exactly like an advisor run (see *Auditing AI activity*), with act-mode writes additionally stamped `[AI Agent] <reason>` in the per-entity trail.

What is sent to the LLM — and what is not

Data sent, per feature

All AI context is built server-side by TrackMe from its own collections and summary telemetry. The table below describes the **maximum** content each feature can include in a request to the LLM:

Feature	Data included in LLM requests
AI Assistant (chat)	The TrackMe system prompt for the current page context; the user's typed message and the conversation history of the current chat; and a structured context payload built from TrackMe's *describe* endpoints — entity identity and configuration (names, identifiers, state, thresholds, schedules), monitoring telemetry (lag/latency/volume KPIs, outlier scores), active alerts and Guardian configuration findings, tenant/component summaries, and license state on the relevant pages. Context is rebuilt fresh on every request.
AI status reports (stateful alert emails)	The same class of entity context as the AI Assistant, for the alerting entity, used to generate the short AI investigation summary embedded in the alert email.
AI Advisors (ML, Feed Lifecycle, FLX Threshold, FQM, Component Health)	Entity configuration and state; metric and quality **histories** (e.g. 7–30 day delay, latency, outlier-score or quality-percentage series, aggregated via `mstats`/summary searches); peer-entity threshold comparisons; ML model training metadata (distributions, boundaries, confidence); per-entity audit/state-transition history; and the optional free-text instruction the operator provides when launching the run.
AI Concierge	The TrackMe REST API **catalog metadata** (endpoint paths, schemas, descriptions, danger levels — product documentation, not customer data); the user's request; and the responses of read-only catalog-validated endpoint calls the agent performs to verify state.
AI Agents automation (scheduled)	The same per-entity context as the corresponding advisor, for the entities selected by your per-tenant scope filters (priority filter and optional filter expression).
AI Agents for findings (every findings tenant: UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge)	The evidence TrackMe computes for the finding under review: the finding, its episodes and history, the policy, and the object it names — for UAM the account (its name, class and open findings) and bounded activity records, for Cribl Stream the inventory, traffic and traffic-model facts of the object, for Cribl Edge the node or Fleet as the node directory published it, for Splunk HEC, Splunk S2S, Splunk UF Fleet and Splunk Forwarder Collectors the facts the tenant's collection published about the endpoint, pool, agent, collector or input. Every run adds the agent's own custom instructions (written by the tenant administrators); interactive runs add the operator's optional instructions; scheduled runs add the tenant's shared custom instructions. See *AI Agents for findings*.
AI Routines	The operator's plain-English intent and its compiled instruction; a slimmed per-tenant entity summary (state, priority, recency of state changes) for the routine's scope; the trigger-match rows for hybrid/deterministic routines; and the routine's own **memory** — compact fingerprints of its recent runs and, for entity-level intents, the exact entity names it acted on last (see *AI Routines*). No credentials, no user directory data, no raw indexed events. Anonymisation toggles apply to routine context as they do elsewhere.

What is never sent

The following are structurally excluded from every AI request:

- **Raw indexed events** from your Splunk indexes — with the two exceptions documented below. TrackMe's AI reasons over monitoring metadata and aggregated telemetry, not over your log data.
- **Credentials and secrets** — Splunk passwords, session keys, API keys (including the AI provider keys themselves), and tokens are never included in prompts, context payloads, or tool results.
- **User directory data** — usernames and email addresses of your Splunk users are not part of context payloads (the *operator's* identity is recorded in the local audit trail, but not sent to the LLM). The exception is the account a User Activity Monitoring finding is about, documented below; email addresses are never sent.
- **Data from other configured providers** — provider accounts are isolated; no data is ever shared between providers.

Important

The FQM exception — the *AI FQM Advisor* includes a `get_fqm_sampled_failures` tool that lets the agent inspect **samples of failing events** already collected by the FQM collect job, strictly to diagnose field-quality issues (e.g. test a regex against real values). If your FQM-monitored sourcetypes contain sensitive payloads and your policy prohibits sending event samples to your chosen provider, either run the FQM Advisor against a self-hosted provider, or do not enable the FQM Advisor. All other advisors and the assistant operate on metadata and aggregated telemetry only.

Important

The User Activity Monitoring exception — the subject of a User Activity Monitoring finding is a Splunk account, so the *UAM AI Agent* sees account names, their class, their roles and bounded records of their activity (at most 50 per read). It sees the text of a search only when it asks for it, and only for searches collected while the tenant's *Store SPL search* setting was on (off by default). The name anonymisation options do not apply to the findings AI Agents. If your policy prohibits sending account names to your provider, use a self-hosted provider or do not use the UAM AI Agent.

Anonymisation options

For organisations whose *infrastructure metadata itself* is sensitive (hostnames, IPs, index names), TrackMe provides global anonymisation toggles (**Configuration > General > Artificial Intelligence**, see *Configuring AI Provider accounts*):

Setting	Effect
Anonymize entity names for AI	Entity names (object and alias fields) are replaced by SHA256 hashes in every payload sent to the AI provider. The AI is guided to reason on opaque `object_id` references instead; when an action is approved, the real identifiers are resolved locally by TrackMe — the LLM never constructs primary keys.
Anonymize index names for AI	Splunk index names are replaced by SHA256 hashes in entity information, investigation searches and metric details sent to the provider.

Both options trade some answer readability for confidentiality and can be combined with a self-hosted provider for defence in depth.

Provider choice, data residency and model training

You decide where data is processed

TrackMe is **provider-agnostic** — the same AI code runs against every supported backend, so the choice of jurisdiction, deployment model and data-use terms stays with you:

Requirement	Option
Air-gapped / fully on-premises	Ollama (self-hosted open-source models) or any internal OpenAI-compatible endpoint (NVIDIA NIM, LM Studio, corporate gateways). Zero external calls.
Keep data inside the Splunk platform (Splunk Cloud)	**Splunk Hosted (SLIM API)** — models operated by Splunk within the Splunk Cloud platform; no external API key, short-lived platform tokens, no third-party egress.
Specific geography / sovereign cloud	**Azure OpenAI** deployed in your chosen region, or region-pinned deployments from other providers via the Custom provider type.
Managed frontier models	OpenAI, Anthropic (Claude), Google Gemini, Mistral AI, xAI (Grok) — via their public API endpoints with your own account.

See *Configuring AI Provider accounts* for the full configuration reference of each provider type.

Model training and provider-side retention

Three facts frame the training question:

1. **TrackMe never trains on, stores, or reuses your AI traffic.** Prompts and responses transit the search head and are rendered to the user; what TrackMe persists is the audit metadata and the structured advisor results, in **your** Splunk indexes.
2. **All cloud providers are consumed through their API tier**, authenticated with **your** organisation's API key. The applicable data-use and retention terms are those of *your* agreement with the provider — not consumer-product terms. As of this writing, the major supported providers (OpenAI, Azure OpenAI, Anthropic, Google, Mistral) state that API/business traffic is **not used to train their models by default**, and several offer zero-data-retention or regional-processing agreements on request.
3. **Self-hosted options remove the question entirely** — with Ollama or a private endpoint, no third party ever sees the data.

Note

Provider terms evolve. For a formal assessment, verify the current data-usage commitments directly with your provider — e.g. OpenAI's *API data usage policies*, Anthropic's *commercial terms*, Microsoft's *Data, privacy, and security for Azure OpenAI*, Google's *Gemini API terms*, Mistral's *privacy policy* — and attach them to your AI risk file alongside this page.

Credential management

- AI provider API keys are entered once in **Artificial Intelligence > Manage AI Provider accounts** and stored **encrypted** in Splunk's secure credential store (`storage_passwords`). They are write-only: the UI and REST API never return a stored key.
- Decryption is performed server-side, only at the moment an LLM request is made, by TrackMe's backend under Splunk's access controls. Keys are never logged, never included in audit events, and never part of any prompt.
- Self-hosted (Ollama, custom internal endpoints) and Splunk Hosted providers require no stored API key at all.

The complete tool inventory

The compliance-critical property of TrackMe's agentic features is that the LLM can only interact with your environment through a **fixed registry of typed tools**. There is no generic "run a search" or "call an API" capability exposed to the model:

- **The LLM never writes SPL.** Every search a tool performs is pre-built by TrackMe (`mstats` on TrackMe metric indexes, bounded searches on TrackMe summary/audit indexes); the model only supplies validated, typed parameters.
- **Tool sets are mode-filtered at runtime.** In **inspect** mode, write tools are *absent from the registry* — not merely discouraged. **Act** mode adds the write tools and is gated by RBAC.
- **Every write tool requires a ``reason``** that is stamped into the per-entity audit trail as `[AI Agent] <reason>`, and every tool invocation (read or write) emits its own audit event.

The tables below enumerate every tool, per advisor.

AI Assistant (chat)

The assistant has **no tools and no write path**. It receives context built server-side and can only *propose* advisor or Concierge invocations as consent cards; nothing executes without the user's explicit click, which runs under the user's own credentials.

AI Concierge

Catalog-driven generalist with **six read tools and zero write tools by design** — mutations are proposed as action contracts and executed only by the user's consent click, through the user's own RBAC-checked credentials. Destructive actions additionally require the user to **type the action name** to confirm.

Tool	Purpose / data accessed	Access
`discover_endpoints`	Search the TrackMe REST API catalog by intent (endpoint metadata only).	Read
`list_resource_groups`	List the curated resource groups of the API catalog.	Read
`list_endpoints_in_group`	Enumerate the endpoints of one group with their schemas.	Read
`describe_endpoint`	Full self-documentation of a single endpoint (parameters, examples, danger level, RBAC).	Read
`read_via_endpoint`	Execute a **read-classified** endpoint to verify live state. Refuses any path not present in the catalog or not marked read-only.	Read
`propose_action`	Validate and record a proposed action contract — **does not execute**. The consent card click is what fires the actual REST call.	Read (proposal only)

AI ML Advisor

Scope: ML outlier model integrity. See *AI ML Advisor*.

Tool	Purpose / data accessed	Access
`get_entity_outlier_conte`	Entity description and ML model state (config, boundaries, confidence).	Read
`get_entity_metric_histor`	Metric time series from TrackMe metric indexes (`mstats`).	Read
`get_model_training_detai`	Fitted distributions, boundaries, fit-quality indicators.	Read
`get_outlier_score_histor`	Historical outlier scoring timeline.	Read
`get_entity_alert_history`	Entity state-transition / alert history from the audit trail.	Read
`get_model_render_history`	Training-window rendering with boundaries and outlier flags.	Read
`add_period_exclusion`	Exclude an anomalous window from model training. Reversible.	Write
`trigger_model_retrain`	Retrain the model (live or simulation).	Write
`set_false_positive`	Neutralise the current outlier score as a false positive.	Write
`update_model_rules`	Tune model parameters; refuses empty/no-op updates.	Write
`manage_outlier_detection`	Enable / disable / reset outlier detection. Disabling in automated mode is blocked unless the dedicated tenant flag is set.	Write (guarded)

AI Feeds Advisor (Feed Lifecycle — DSM/DHM)

Scope: data source / data host thresholds and lifecycle. See *AI Feeds Advisor*.

Tool	Purpose / data accessed	Access
get_entity_lifecycle_con	Full entity lifecycle context (config, thresholds, state, alerts).	Read
get_entity_delay_latency	30-day delay/latency history from the summary index.	Read
get_entity_alert_flip_hi	State-transition (flip) history from the audit index.	Read
get_entity_peer_comparis	Threshold comparison against peer entities of the same tenant.	Read
get_entity_variable_dela	Current variable-delay schedule, if any.	Read
get_entity_adaptive_dela	Adaptive-delay change history from the audit trail.	Read
update_entity_thresholds	Update delay / latency thresholds.	Write
update_entity_monitoring_s	Enable or disable monitoring of the entity. Disabling in automated mode is blocked unless the decommission flag is set.	Write (guarded)
update_entity_adaptive_del	Lock or unlock adaptive delay.	Write
update_entity_variable_del	Create or update a variable-delay schedule.	Write
update_entity_priority_and	Update entity priority and/or manual tags.	Write
update_entity_impact_score	Adjust per-entity impact-score weights.	Write

AI FQM Advisor

Scope: field-quality thresholds and the data dictionary. See *AI FQM Advisor*.

Tool	Purpose / data accessed	Access
get_fqm_entity_context	Full FQM entity context.	Read
get_fqm_dictionary	Data dictionary and the trackers using it.	Read
get_fqm_field_dictionary	A single field's dictionary entry.	Read
get_fqm_quality_history	Success / coverage percentage history with day-of-week breakdown.	Read
get_fqm_sampled_failures	**Sampled failing events** collected by the FQM collect job (see the FQM exception above).	Read
get_fqm_collect_job_cont	Health of the collect-job and monitor saved searches.	Read
get_fqm_peer_entity_thre	Peer FQM entities' thresholds on the same tracker.	Read
get_fqm_datamodel_contex	CIM datamodel context and diff versus expected fields.	Read
test_fqm_regex	Test a candidate regex against real sampled values.	Read
add_fqm_threshold / update_fqm_threshold / delete_fqm_threshold	Manage field-quality thresholds.	Write
update_fqm_field_dictionar / update_fqm_dictionary_bulk	Update data-dictionary validation configuration (single field or bulk).	Write
update_fqm_entity_state_pr	Update monitoring state or priority.	Write (guarded)

AI FLX Threshold Advisor

Scope: Flex Object dynamic thresholds. See *The AI Advisors*.

Tool	Purpose / data accessed	Access
`get_flx_entity_context`	Full FLX entity context.	Read
`get_flx_metric_history`	30-day metric value history (`mstats`).	Read
`get_flx_threshold_breach`	State-transition and threshold-breach history.	Read
`get_flx_use_case_definit`	The use-case definition of the entity's tracker.	Read
`get_flx_peer_entity_thre`	Peer FLX entities' thresholds on the same tracker.	Read
`add_flx_threshold` / `update_flx_threshold` / `delete_flx_threshold`	Manage dynamic thresholds.	Write
`set_flx_variable_thresho`	Configure time-slot-based variable threshold values.	Write
`update_flx_entity_state_pr`	Update monitoring state or priority.	Write (guarded)

AI Component Health Advisor (WLK / MHM)

Scope: workload (WLK) and metric-host (MHM) health. See *AI Component Health Advisor*. The tool registry is component-filtered at runtime — a WLK run only sees WLK tools, an MHM run only MHM tools.

Tool	Purpose / data accessed	Access
`get_wlk_entity_context`	Full WLK entity context.	Read
`get_wlk_execution_histor`	Execution history with day-of-week breakdown.	Read
`get_wlk_breach_history`	State-transition and threshold-breach history.	Read
`get_wlk_peer_thresholds`	Peer WLK entities in the same Splunk app.	Read
`add_wlk_threshold` / `update_wlk_threshold` / `delete_wlk_threshold`	Manage execution-metric thresholds.	Write
`update_wlk_entity_state_pr`	Update monitoring state or priority.	Write (guarded)
`get_mhm_entity_context`	Full MHM entity context.	Read
`get_mhm_metric_lag_histo`	Metric-lag history with day-of-week breakdown.	Read
`get_mhm_breach_history`	State-transition and lag-breach history.	Read
`get_mhm_lagging_classes`	Lagging-class definitions for the tenant.	Read
`update_mhm_metric_max_lag`	Update the metric-lag threshold.	Write
`update_mhm_entity_state_pr`	Update monitoring state or priority.	Write (guarded)

Shared metadata tools (all specialised advisors, act mode only)

Tool	Purpose / data accessed	Access
`manage_entity_labels`	Assign or remove labels from the tenant's curated catalog. Does not auto-create labels.	Write
`manage_entity_note`	Create or delete free-form entity notes (append-only; notes are immutable once created).	Write

AI Routines (act mode)

A routine draws from the **same fixed, typed tool registry** as the advisors — there is no routine-specific "run anything" capability. Read tools (the tenant/scope summary, a bounded read-only correlation search, inspect-only advisor dispatch, and the flip-history reader) are always available; write tools are exposed **only** for the capability families the author consented to, one coherent capability per family (the 16 families are enumerated in *AI Routines*). Two families reach off-box under the pinned consent and Splunk RBAC — *trigger Splunk alert actions* and *execute external MCP tools* — as described in *AI Routines: user-authored automation, explicitly consented*. Notify-mode routines receive **no write tools at all**.

AI Agents for findings

Each findings AI Agent has its own small registry, pinned to the tenant and the run that started it: a tool can neither reach another tenant nor decide outside its run. Write tools are absent in inspect mode. The write tool reads the finding again and the server re-checks every precondition before applying anything; the model never supplies the version fences.

Tool	Purpose / data accessed	Access
`list_<type>_findings` (`<type>` is `uam`, `cribl`, `cribl_edge`, `hec`, `s2s`, `uf` or `fwc`, the same for the tools below)	The open findings of the tenant, one row per Finding group (at most 50), with the settings and the decisions the tenant allows.	Read
`get_<type>_finding_evide`	The evidence of one finding, computed by TrackMe, and the preconditions of each decision.	Read
The cached views of the Splunk S2S, UF Fleet and Forwarder Collectors agents (`get_s2s_overview`, `get_s2s_host`, `get_uf_agent`, `get_fwc_inputs` and the like)	What the tenant's collection already published about the fleet, a host, a pool, an agent, a Deployment Server, a group or the inputs, and the traffic models.	Read
`read_<type>_log_evidence` `read_<type>_trend`, `read_<type>_recent` (`s2s`, `uf`, `fwc`)	The tenant's own bounded read commands: log evidence, trends and the recent window of one object, each a TrackMe search with a timeout, at most three reads per finding after the evidence. Never a search the model writes.	Read
`get_uam_policy_evaluatio`	How a UAM policy is configured and evaluated in the tenant.	Read
`get_uam_account`	A UAM account: its class, where the class comes from, and its findings.	Read
`get_uam_activity`	Bounded activity records of an account (at most 50 events); search text only on request and only when the tenant stored it.	Read
`apply_<type>_finding_decis`	Apply one decision the tenant allows, with its rationale, under the actor **AI Agent**; refused when a precondition does not hold.	Write

Guardrails and safety controls

Defence in depth, by layer:

Layer	Control
Structural	The chat assistant has no tools and no write path. Write tools are physically absent from the registry in inspect mode. The Concierge has zero static write tools. The LLM never constructs primary keys (tenant and entity identifiers come from session state, not model output) and never emits raw SPL.
Human-in-the-loop	Every interactive mutation goes through a **consent card**: the user reviews the exact proposed action (and can edit the request body for Concierge actions) before clicking. The click executes through the **user's own credentials**. Destructive Concierge actions require typed confirmation of the action name.
RBAC	Enforced at the REST boundary, not in the prompt: inspect requires TrackMe read capability, act requires power-level capability, provider administration requires admin capability — all additionally scoped by per-tenant roles. An unauthorized write returns HTTP 403 regardless of what the agent attempts. See *Access control (RBAC)*.
Automation gates	Unattended runs are **disabled by default** per tenant and per advisor family. Even when enabled, dedicated flags (both **disabled by default**) are required before an automated agent may decommission an entity (`allow_decommission`) or disable an ML model (`allow_model_disable`). Entity selection is scoped by priority filters and an optional filter expression; an unparseable expression **fails closed** (no entity processed).
Runtime budgets	Hard token, step and wall-clock budgets per run, enforced by SDK middleware plus an independent watchdog thread; per-search-head concurrency caps; per-tenant Concierge rate limits and per-proposal action caps. A misbehaving run hits a real wall, not a soft prompt.
Validation	Advisor outputs are typed (Pydantic schemas) — invalid completions are retried or rejected. Concierge action paths are validated byte-for-byte against the live REST catalog at proposal time and re-validated at execution time; write tools refuse no-op updates.
Fail-safe semantics	AI failures never block operations: a failed chat degrades to an error message, a failed AI status report is replaced by "AI investigation unavailable" while the alert email still sends, and an unavailable provider simply disables the feature with a clear message.

Access control (RBAC)

AI features inherit TrackMe's standard three-tier role model (see *Roles & access control*), enforced at every REST endpoint:

Operation	Required capability	Notes
Use the AI Assistant; launch advisors in **inspect** mode; view audit dashboards	`trackmeuseroperations`	Read-only surface. Audit views and advisor context are additionally filtered to the tenants visible to the caller.
Launch advisors in **act** mode; approve consent-card actions	`trackmepoweroperations`	Writes execute under the approving user's credentials and are tenant-scope checked.
Ask a findings AI Agent and let it act	UAM: a TrackMe administrator of the tenant; every other dedicated tenant type (Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge): the tenant's power and admin roles	The settings of the AI Agent belong to a TrackMe administrator of the tenant. Decisions are recorded under the actor **AI Agent**, with the requesting user in the decision log.
Manage AI provider accounts and global AI settings	`trackmeadminoperations`	Provider CRUD, connectivity tests, anonymisation and feature toggles.

Per-tenant role instances (`trackme_user|power|admin_<tenant_id>`) apply on top of the global capabilities — holding a global capability is not sufficient without the per-tenant role. Scheduled automation runs under the Splunk system user within the per-tenant gates described above.

Audit and traceability

Every AI interaction leaves a queryable trace in **your** Splunk environment — see *Auditing AI activity* for the dashboards. Three layers of records are emitted:

Record	Content
Per tool call (`trackme:ai_agent:tool_c`	One event per tool invocation: timestamp, advisor, tool name, status, duration, tenant, component, entity, mode (inspect/act), interactive vs automated, job correlation ID, result size, error details.
Per run (`trackme:ai_agent:<advis` `trackme:ai_agent:ai_rout` `trackme:ai_assistant:cha` `trackme:ai_assistant:sta`	One terminal event per advisor run, routine fire or chat: user (or resolved owner) identity, provider and model used, token usage, steps taken, duration, status, how it was launched (UI / chat / automation), the chat session correlation ID, and — for advisors and routines — the full structured result (recommendations and actions taken).
Per finding decision (findings AI Agents)	Every decision is recorded in the finding history under the actor **AI Agent**, in the tenant's decision log (with the requesting user, the provider and whether the run was scheduled), and in the current `[AI Agent]` note of the finding. Refused decisions are recorded too.
Per entity write (per-entity audit trail)	Every write is stamped `[AI Agent] <reason>` in the affected entity's *Audit changes* panel, with the agent required to cite the field changed, the before/after values and the operational trigger.

Properties relevant to compliance review:

- **Attribution** — every event carries the operator's identity (or `automated` for scheduled runs), the tenant, the provider account and the exact model. The `chat_session_id` field lets you answer "which conversation triggered this change?" with one SPL query.
- **Chat content is not indexed** — audit events for chats record metadata and token counts, not the conversation text. Chat exchanges transit a KV Store collection only for asynchronous delivery to the browser and are automatically purged within minutes; they are never written to a Splunk index. Advisor events do persist the structured result, which is the operational record of what was recommended and done.
- **Retention is yours** — all audit events land in your tenant summary index and follow your Splunk retention policies; nothing is retained outside your deployment.
- **Cost observability** — token usage is recorded per run and aggregated in the audit dashboards, per provider and model.

Enablement controls and defaults

The complete switchboard, from most global to most granular:

Control	Default	Effect
AI provider account configured	**None**	The root opt-in. With no provider account, the entire AI layer is inert — no UI surface, no LLM call, no egress.
Enable AI assistant (global, Configuration > General)	Enabled	Master feature switch. When off, every AI endpoint returns HTTP 403 and all AI UI is hidden. Enabled by default for discoverability, but functionally inert until a provider exists.
Anonymize entity / index names for AI (global)	Off	SHA256-hash entity and/or index names in all payloads sent to providers.
Max concurrent AI chats (global)	10	Concurrency cap; excess requests are rejected with HTTP 429.
ML Advisor automation (per tenant)	**Disabled**	Opt-in scheduled runs; mode, provider, review-frequency and runtime caps per tenant.
Components Advisor automation (per tenant)	**Disabled**	Opt-in scheduled runs for Feed Lifecycle / FLX / FQM / Component Health; component list, mode and caps per tenant.
Enable AI Routines (global, Configuration > General)	Enabled	Feature switch for AI Routines specifically (on top of *Enable AI assistant*). When off, routine menus are hidden, the create / evaluate / run-now endpoints return 403, and scheduled routine execution is refused.
Who can create AI Routines (global)	Admins & power users	Restricts routine creation to administrators, or administrators and power users; global (cross-tenant) routines always require an administrator.
Routine act mode + permission families (per routine)	**Notify / none**	A routine is notify-only unless the author selects *act* and grants specific capability families, under a hash-pinned consent that any later edit invalidates.
Automation scope filters (per tenant)	`critical, high`	Priority filter plus optional filter expression restricting which entities automation may touch. Invalid expressions fail closed.
Allow decommission / allow model disable (per tenant)	**Disabled**	Without these flags, automated agents can never disable an entity or an ML model.
AI Agent scheduled triage (per findings tenant)	**Disabled**	Opt-in hourly triage of findings; inspect mode by default. Act mode applies only the decisions allowed one by one in the tenant settings, never over an operator's decision.
Per-entity opt-out	—	Individual entities can be flagged to be skipped by automated advisor runs.
RBAC	—	Standard TrackMe roles gate every AI surface per user and per tenant (see above).

To **completely disable AI**: do not configure any provider account (or delete existing ones), and set **Enable AI assistant** to off. Both conditions are independently sufficient to stop all LLM calls.

Network egress and transport security

All LLM traffic is **outbound HTTPS from the search head(s)** to the single endpoint of each configured provider. For firewall/egress allow-listing:

Provider	Egress destination (TCP 443)
OpenAI	`api.openai.com`
Azure OpenAI	`<your-resource>.openai.azure.com` (your region)
Anthropic	`api.anthropic.com`
Google Gemini	`generativelanguage.googleapis.com`
Mistral AI	`api.mistral.ai`
xAI (Grok)	`api.x.ai`
Ollama / Custom (OpenAI-compatible)	Your own endpoint — typically internal; **no Internet egress required**.
Splunk Hosted (SLIM API)	Splunk Cloud platform-internal (auto-discovered); no customer-managed egress.

Transport properties:

- TLS certificate verification uses the system CA trust store and is **strict for all cloud providers**. For Ollama only, hostname/certificate verification is relaxed **exclusively for loopback addresses** (`127.0.0.1` / `localhost`); a remote Ollama or custom endpoint is verified strictly.
- No inbound connectivity is required by any AI feature; TrackMe never opens listening ports for AI.
- Requests are bounded by the per-provider request timeout (see *Configuring AI Provider accounts*).

Frequently asked compliance questions

Does TrackMe Limited (the vendor) see our prompts, data, or AI responses?
No. TrackMe is a Splunk app executing in your environment; there is no vendor-operated AI service, proxy, or telemetry in the data path. The only external party — if you choose a cloud provider — is that provider, under your own account and terms.

Is anything sent to an LLM before we configure a provider?
No. Without a provider account the AI layer is inert: no call, no egress, no background AI processing of any kind.

Which data crosses the boundary to the provider, in one sentence?
The system prompt, the user's question or operator instruction, and a TrackMe-built context of monitoring metadata and aggregated telemetry for the entity/tenant in scope — never credentials, never user directory data, and never raw indexed events (FQM sampled failures being the single documented, avoidable exception).

Can the AI exfiltrate data by running its own searches?
No. The LLM cannot emit SPL or arbitrary REST calls. It can only invoke the typed tools listed on this page, whose queries are pre-built, scoped to TrackMe's own collections and indexes, bounded in time range, and individually audited. The Concierge's only generic read path refuses any endpoint not present in the catalog and not classified read-only.

Can the AI delete data, entities, or tenants on its own?
No. Interactive writes require an explicit user consent click (RBAC-checked, user credentials); destructive Concierge actions additionally require typed confirmation. Automated runs are opt-in, disabled by default, and structurally prevented from decommissioning entities or disabling models unless you enable dedicated flags that are also disabled by default.

Under whose identity do AI actions run?
Interactive reads/writes execute under the launching/approving user's Splunk session and roles. Scheduled advisor automation runs under the Splunk system user, inside the per-tenant gates and filters you configured. An **AI Routine** runs as a resolved **owner** — the tenant's service account, or the routine's creator — recorded on every event. All paths are fully attributed in the audit trail.

Can an AI Routine act on its own, and how is that controlled?
Only if its author explicitly made it an *act* routine and granted specific capability families under a hash-pinned consent. A routine is notify-only by default; in act mode it may apply just the consented capabilities on its schedule, with no per-fire click — but any edit to the intent or the grants invalidates the consent and stops it acting until re-affirmed, delivery targets are fixed configuration (not model output), the tenant it can touch is

fixed from its scope, and every fire (including each write, stamped `[AI Agent] <reason>`) is audited. The whole feature has a global kill switch and a policy for who may create routines. See *AI Routines: user-authored automation, explicitly consented*.

Is the conversation history stored?
Chat conversations are ephemeral. Message exchanges pass through a KV Store collection solely for asynchronous delivery and are automatically purged within minutes; chat bodies are never indexed. Audit events record metadata only (user, model, tokens, duration, status). Advisor runs persist their structured results as the operational record.

How would we investigate an AI-made change six months later?
The per-entity *Audit changes* panel shows the `[AI Agent] <reason>` entry; the summary index holds the per-run and per-tool events with the user, provider, model, and `chat_session_id` correlation back to the conversation that proposed it — all queryable in plain SPL, retained per your index policies.

Can we run a different provider per sensitivity tier?
Yes. Multiple provider accounts can coexist (e.g. self-hosted Ollama for sensitive tenants, a cloud provider elsewhere), and the provider is selected per chat / per advisor run. No data is shared between providers.

Does using TrackMe AI change our Splunk data's exposure to Splunk (the company)?
No, with one explicit exception you control: selecting the **Splunk Hosted (SLIM API)** provider on Splunk Cloud sends AI requests to models operated by Splunk inside the Splunk Cloud platform. All other providers bypass Splunk entirely; self-hosted providers bypass everyone.

What should our DPIA / AI risk assessment reference?
This page (data flows, tool inventory, controls), *Configuring AI Provider accounts* (provider terms, residency), *Auditing AI activity* (evidence and monitoring), your chosen provider's API data-use terms, and your Splunk index retention configuration for the audit trail.

Related

- *Artificial Intelligence in TrackMe* — the AI layer at a glance and its three design pillars.
- *Configuring AI Provider accounts* — provider accounts, global AI settings, anonymisation toggles.
- *The AI Advisors* — the advisor family, inspect/act and consent cards.
- *AI Agents automation* — unattended automation and its per-tenant policy gates.
- *AI Routines* — user-authored scheduled routines and their consent model.
- *Auditing AI activity* — the audit dashboards and the audit event envelope.
- *Roles & access control* — TrackMe's role-based access-control model.

3.7.3 EU AI Act compliance

Overview

This page describes how TrackMe's AI capabilities position against **Regulation (EU) 2024/1689 — the EU Artificial Intelligence Act** ("AI Act"). It is written for compliance, risk and legal teams that need to answer one question quickly — *"can we use TrackMe's AI features in the EU, and what does the AI Act require from us if we do?"* — and for TrackMe administrators who want to understand what the product already does for them.

It complements *AI Compliance, Privacy and Data Governance* (privacy, data flows, tool inventory, guardrails) and *Auditing AI activity* (audit trail): those pages describe *how the product behaves*; this page maps that behaviour to the *AI Act's roles, risk categories and obligations*.

Note

This page reflects TrackMe Limited's good-faith self-assessment of the product as implemented, and is provided for information only — it is **not legal advice**. The AI Act's application always depends on *your* deployment context and use; organisations subject to the Act should perform their own assessment with qualified counsel.

Important

The short version — TrackMe Limited assesses TrackMe's AI layer, for its intended purpose, as performing none of the Act's prohibited practices and as falling outside both high-risk routes (the Annex III use-case list and the Annex I product-safety route). The operative obligations are therefore the **Article 50 transparency** duties — a formal AI Act category of its own, not an exemption: users must know when they interact with AI, and AI-generated content must be identifiable as such, by machines as well as humans. TrackMe implements both: every AI surface is explicitly labelled, and artefacts embedding LLM-generated content carry a **machine-readable marking** (see *Article 50 transparency — how TrackMe complies*).

Roles under the AI Act

The AI Act assigns obligations by **role**. A TrackMe deployment involves three:

AI Act role	Who	What it means
GPAI model provider	The entity that developed the model you configure and placed it on the market under its own name — OpenAI, Anthropic, Microsoft (Azure OpenAI), Google, Mistral, Splunk, or the upstream developer of a self-hosted model served via Ollama or a private OpenAI-compatible endpoint	The general-purpose AI model obligations of the Act (Articles 53/55 — model documentation, copyright policy, systemic-risk measures) sit with the **model provider**, not with TrackMe and not with you as a TrackMe user. Simply configuring — or self-hosting — a model does **not** make your organisation the model provider: you remain a deployer unless you develop or substantially modify a model, or place one on the market under your own name or trademark. Your agreement with the provider governs data use; see *AI Compliance, Privacy and Data Governance*.
AI system provider	**TrackMe Limited**	TrackMe integrates general-purpose models into an AI system with a defined intended purpose (Splunk data-monitoring assistance) and places it on the market. TrackMe Limited therefore treats itself as the *provider* of that AI system and owns the system-level transparency design described on this page.
Deployer	**Your organisation**, when you enable the AI layer	Deployers use an AI system under their own authority. Your obligations are limited in TrackMe's risk category — see *What you should do as a deployer* below.

Risk classification

The AI Act is risk-tiered. TrackMe's AI capabilities assess as follows:

AI Act category	Applies?	Rationale
Prohibited practices (Article 5)	**No**	TrackMe's AI performs none of the banned practices: no manipulation or exploitation of persons, no social scoring, no biometric identification or categorisation, no emotion recognition. Its intended purpose is Splunk data-monitoring assistance; the exact data exchanged with the LLM per feature — including user-typed chat messages and, for FQM, sampled event excerpts — is documented feature-by-feature in *AI Compliance, Privacy and Data Governance*.
High-risk systems (Article 6 — Annex I and Annex III routes)	**No**	Neither of the Act's two high-risk routes applies. **Annex I route (Article 6(1))**: TrackMe is not a product — nor a safety component of a product — covered by the Annex I Union product-safety legislation (machinery, medical devices, vehicles, etc.). **Annex III route (Article 6(2))**: IT observability is not an Annex III domain. The closest category — AI used as a *safety component in the management and operation of critical digital infrastructure* — does not apply: TrackMe **observes and alerts**, it does not operate or control infrastructure, and the AI layer is an *advisory* surface on top of TrackMe's deterministic monitoring engine (detection, scoring and alerting are conventional code and remain fully functional with AI disabled). AI write actions target TrackMe's own monitoring configuration and are RBAC-checked, consented and audited; AI Routines can additionally trigger operator-configured Splunk alert actions and external MCP tools — each individually and explicitly consented at routine creation — see *AI Compliance, Privacy and Data Governance*.
Transparency-risk systems (Article 50)	**Yes**	TrackMe's AI Assistant interacts directly with users, and several features generate synthetic text (status reports, advisor summaries, routine notifications). The Article 50 transparency obligations are the operative requirements — fully addressed below.
General-purpose AI models (Chapter V)	**Not TrackMe**	TrackMe does not provide, train, or fine-tune any model. Model-level obligations rest with the model provider you select.

Article 50 transparency — how TrackMe complies

Article 50 (applicable since **2 August 2026**) contains the two obligations relevant to TrackMe.

Disclosure of AI interaction — Article 50(1)

Persons interacting with an AI system must be informed they are interacting with AI, unless obvious from context. In TrackMe:

- Every conversational surface is explicitly and persistently branded **AI Assistant**; assistant messages are labelled as such in the chat transcript.
- Every agentic surface (AI Advisors, AI Concierge, AI Routines, the AI Agents for findings) is presented under an explicit **AI** name, launched from clearly-labelled entry points, and — for any write action — gated by an explicit consent flow (see *AI Compliance, Privacy and Data Governance*). The AI Agents for findings act under their own **AI Agent** identity, inspect by default, are off by default for scheduled triage, and apply only the decisions the tenant allows (see *AI Agents for findings*).
- The entire AI layer is **opt-in**: nothing AI-related is shown to end users until an administrator configures a provider, so no user can encounter AI output unknowingly.

- Disclosure is effective from the **first interaction**, as Article 50(5) requires: the AI Assistant panel is branded before any message is exchanged, the labelling is persistent (not a dismissible one-time notice), and it is rendered as clear, distinguishable text within the standard Splunk UI (theme-aware, keyboard-accessible) rather than hidden in tooltips or fine print.

Machine-readable marking of AI-generated content — Article 50(2)

Providers of AI systems generating synthetic content must ensure outputs are marked as artificially generated **in a machine-readable format**, in addition to any human-readable labelling. TrackMe applies **machine-readable marking** to artefacts that leave the product with embedded LLM content (the email surfaces below), and **provenance labelling** to AI result records stored in your Splunk indexes:

AI output surface	Marking
Stateful alert emails with an AI status report (`ai_status_report=1`)	The message carries an `X-TrackMe-AI-Generated: true` MIME header; the HTML section holding the report carries a `data-ai-generated="true"` attribute; the plain-text alternative labels the section `--- AI Status Report (AI-generated content) ---` (TrackMe 2.4.13+). Emails without an AI report carry **no** marking — the signal is meaningful precisely because it is not blanket-applied.
AI Routine notification emails	The notification body *is* the LLM-generated run summary, so every routine email carries the same `X-TrackMe-AI-Generated: true` header and `data-ai-generated="true"` HTML attribute (TrackMe 2.4.13+).
AI Routine Splunk event notifications	Routine `splunk_event` deliveries index the AI-authored `summary` / `evidence` under the dedicated `trackme:ai:routines` sourcetype (event `source` = the routine id), so these records are identifiable as AI output in plain SPL — provenance labelling, like the advisor results below. The email variant of the same notification carries the full email marking above.
Advisor / agent results indexed in Splunk	Every automated or interactive agent run is indexed with a dedicated `trackme:ai_agent:<advisor>:<mode>` sourcetype, so stored agent results are identifiable — and filterable — as AI output in plain SPL. These sourcetypes are provenance labelling of the stored result records (they double as the audit trail, see *Auditing AI activity*); the artefact-level Article 50(2) marking for content that leaves the product is the email marking above.
Interactive chat responses	Rendered exclusively inside the persistently-labelled AI Assistant panel — an Article 50(1) interaction surface where the AI origin of every response is disclosed by construction. The product does not itself embed chat output into distributable artefacts; where TrackMe embeds LLM output into an artefact that leaves the product (the email surfaces above), it applies the machine-readable marking. TrackMe tracks the Commission's transparency guidelines and the Code of Practice on marking of AI-generated content as they mature, and will extend marking mechanisms accordingly.
Deterministic notifications (Configuration Guardian, topology alerts, connectivity test emails, standard alert emails without an AI report)	**Never marked** — these contain no LLM-generated content. TrackMe deliberately excludes them so the marking remains a reliable discriminator for downstream mail rules, DLP and compliance tooling.

Note

Article 50(2) expects markings to be *effective, interoperable, robust and reliable as far as technically feasible*, reflecting the state of the art. The `X-TrackMe-AI-Generated` header and `data-ai-generated` attribute are TrackMe's **current product marking**, aligned with Article 50(2)'s direction for email artefacts; as the Commission's guidelines and the Code of Practice on marking of AI-generated content converge on common mechanisms, TrackMe will evolve its markings accordingly (the same commitment stated for the chat surface above). Deployers should not assume the proprietary header alone survives every mail transformation — see the transparency-chain

guidance in *What you should do as a deployer*.

Alignment with the Act's broader principles

Beyond its formal obligations, TrackMe's AI design already implements the practices the AI Act promotes for all AI systems:

AI Act principle	TrackMe implementation
Human oversight	Read-only *inspect* mode versus consented *act* mode on every advisor; explicit per-run consent cards; per-tenant enablement flags; per-model opt-outs; a global administrator kill switch (`enable_ai_assistant`). See *AI Compliance, Privacy and Data Governance*.
Transparency of operation	Structured results with reasoning traces; live tool-call progress feeds; every tool an agent may use is publicly documented in the complete tool inventory of *AI Compliance, Privacy and Data Governance*.
Traceability / record-keeping	Every AI interaction and every agent action is recorded in the per-entity audit trail and in dedicated audit dashboards, with structured audit events in your own Splunk indexes. See *Auditing AI activity*.
Robustness	The AI layer is fail-open by design: an AI failure never blocks monitoring, alerting or email delivery. LLM output is treated as untrusted input (sanitised and escaped before rendering).
Privacy and data governance	Direct search-head-to-provider data path with no vendor intermediary, self-hosted / air-gapped options, optional anonymisation of entity and index names, no training on your data by TrackMe. See *AI Compliance, Privacy and Data Governance*.

Applicability timeline

Date	Milestone
2 February 2025	The Act's original core prohibited-practices ban (Article 5) and the AI-literacy obligation (Article 4) apply. *(No impact: TrackMe performs no prohibited practice.)*
2 August 2025	General-purpose AI model obligations apply — to model providers, not TrackMe.
27 July 2026	The "digital omnibus on AI" amendment — Regulation (EU) 2026/1744 — enters into force: it adds a new Article 5 prohibition (non-consensual intimate imagery / CSAM generation, applicable 2 December 2026) *(not relevant to TrackMe)*, defers Annex III high-risk obligations to 2 December 2027 and Annex I embedded systems to 2 August 2028 *(TrackMe is neither)*, and keeps the 2 August 2026 Article 50 application date — while adding, via the new Article 111(4), the four-month transitional period for the Article 50(2) machine-readable marking on systems already on the market (next-but-one row).
2 August 2026	**Article 50 transparency obligations apply** — the operative requirements for TrackMe, addressed as described on this page.
2 December 2026	End of the transitional grace period for the machine-readable marking obligation for AI systems placed on the market before 2 August 2026. TrackMe ships the marking from 2.4.13, ahead of this deadline.

What you should do as a deployer

For a system in TrackMe's risk category, deployer obligations are narrower than for high-risk systems, but they are real. The following covers what applies (and what is good practice):

- **AI literacy (Article 4)** — ensure staff operating TrackMe's AI features have a sufficient understanding of what the AI does and its limits. The *Artificial Intelligence in TrackMe* and *AI Compliance, Privacy and Data Governance* pages are written to support exactly this.

- **Article 50(4) disclosure triggers —** deployers must clearly disclose (i) **deep fakes** (AI-generated or manipulated image, audio or video presented as authentic) and (ii) **AI-generated or manipulated text published to inform the public on matters of public interest** (unless the text underwent human review and a person holds editorial responsibility). TrackMe produces neither: its AI output is operational monitoring text delivered to your own teams. The obligation would only be triggered by *your* downstream use — for example, publishing an AI-generated TrackMe report verbatim as public communication — in which case disclose accordingly.
- **Keep the transparency chain intact —** if you forward or re-publish AI-generated content (e.g. pasting an AI status report into a ticket or report), preserve its AI-generated labelling. Note that preservation is **conditional on the path**: forwarding as an attachment preserves the `X-TrackMe-AI-Generated` header, but inline forwards typically re-compose the message and may drop custom headers (the in-body `data-ai-generated` attribute and the plain-text section label are more likely to survive). When a republication path strips the marking, re-apply an equivalent AI-generated label yourself.
- **Review your model provider's terms —** data-use, residency and retention are governed by your agreement with the provider you configure; see *AI Compliance, Privacy and Data Governance* for provider-specific notes and self-hosted options.
- **Use the audit surfaces —** the audit dashboards and `trackme:ai_agent:*` events give you the record-keeping evidence an AI governance programme typically asks for; see *Auditing AI activity*.
- **Run your own assessment —** if your organisation uses TrackMe in an unusual context (for example, as part of a system that *does* fall under Annex III), assess that composed system yourself; TrackMe's classification above covers the product as shipped.

Version applicability

The Article 50(2) machine-readable email marking ships in the companion **TrackMe 2.4.13** application release; it will appear in the Version 2.4.13 entry of the *release notes* when that release is published. All other mechanisms referenced on this page (labelled AI surfaces, consent flows, audit trail, indexed `trackme:ai_agent:*` and `trackme:ai:routines` events, kill switch, anonymisation) are available from TrackMe 2.4.x as described in *AI Compliance, Privacy and Data Governance*.

CHAPTER

FOUR

LICENSE & SUPPORT:

4.1 EULA License

Download the EULA License as a text file

4.1.1 TrackMe Limited (TrackMe) EULA

End-User License Agreement ("Agreement")

Our EULA was last updated on 10 February 2025.

We provide the Application, also known as TrackMe. In this Agreement, you means the person or entity using the Application. If you are agreeing to this Agreement not as an individual but on behalf of your company, government, or other entity for which you are acting (for example, as an employee or governmental official), then you means your entity and you are binding your entity to this Agreement.

Please read this End-User License Agreement carefully before clicking the "I Agree" button, downloading or using TrackMe, Data Tracking system for Splunk.

Interpretation and Definitions

Interpretation

The words of which the initial letter is capitalized have meanings defined under the following conditions. The following definitions shall have the same meaning regardless of whether they appear in singular or in plural.

Definitions

For the purposes of this End-User License Agreement:

- "Agreement" means this End-User License Agreement that forms the entire agreement between you and the Company regarding the use of the Application.
- "Application" means the software program provided by the Company downloaded by You through an Application Store's account to a device, named TrackMe, Data Tracking system for Splunk.
- "Application Store" or "Splunk Base" means the digital distribution service operated and developed by Splunk Inc. ("Splunk") by which the Application can be downloaded to your Splunk Enterprise or Splunk Cloud environment.
- "Company" (referred to as either "the Company", "We", "Us" or "Our" in this Agreement) refers to TrackMe Limited, U.K.
- "Consequential Loss" includes any consequential loss, indirect loss, real or anticipated loss of profit, loss of benefit, loss of revenue, loss of business, loss of goodwill, loss of opportunity, loss of savings, loss of reputation, loss of use and/or loss or corruption of data, whether under statute, contract, equity, tort (including negligence), indemnity or otherwise.
- "Content" refers to content such as text, images, or other information that can be posted, uploaded, linked to or otherwise made available by You, regardless of the form of that content.
- "Country" refers to: United Kingdom.

- "Third-Party Services" means any services or content (including data, information, applications and other products services) provided by a third-party that may be displayed, included or made available by the Application.

Acknowledgment

By clicking on "Agree to Download" button in SplunkBase Website, or downloading or using the Application, you are agreeing to be bound by the terms and conditions of this Agreement. If you do not agree to the terms of this Agreement, do not click on the "Agree to Download" button in SplunkBase Website, do not download or do not use the Application.

You must be at least 16 years old to use the Application.

This Agreement is a legal document between you and the Company and it governs your use of the Application made available to you by the Company.

This Agreement is between you and the Company only and not with the Application Store. Therefore, the Company is solely responsible for the Application and its content.

The Application is licensed, not sold, to you by the Company for use strictly in accordance with the terms of this Agreement.

We may amend this Agreement at any time, by providing written notice to you, unless otherwise stated in this Agreement. By clicking "I accept" or continuing to use the Application after the notice or 30 days after notification (whichever date is earlier), you agree to the amended Agreement. If you do not agree to the amendment, you may cease accessing the Application.

License

Scope of License

Subject to your compliance with this Agreement, the Company grants you a revocable, non-exclusive, non-transferable, royalty free, limited license to download, install and use the Application strictly in accordance with the terms of this Agreement. All other uses are prohibited without our prior written consent.

You may only use the Application on a device that you own or control and as permitted by the Application Store's terms and conditions.

License Restrictions

When using the Application, you agree not to, and you will not permit others to:

- do or attempt to do anything that is unlawful or inappropriate;
- anything that would constitute a breach of an individual's privacy (including uploading private or personal information without an individual's consent) or any other legal rights;
- use the Application to defame, harass, threaten, menace or offend any person, including using the Application to send unsolicited electronic messages;
- tamper with or modifying the Application (including by transmitting viruses and using trojan horses);
- use data mining, robots, screen scraping or similar data gathering and extraction tools on the Application;
- License, sell, rent, lease, assign, distribute, transmit, host, outsource, disclose or otherwise commercially exploit the Application or make the Application available to any third party; or
- Remove, alter or obscure any proprietary notice (including any notice of copyright or trademark) of the Company or its affiliates, partners, suppliers or the licensors of the Application.

Intellectual Property

The Application, including without limitation all copyrights, patents, trademarks, trade secrets and other intellectual property rights (**Our Intellectual Property**) are, and shall remain, the sole and exclusive property of the Company.

We authorise you to use Our Intellectual Property solely for your limited commercial use. You must not exploit Our Intellectual Property for any other purpose, nor allow, aid or facilitate such use by any third party.

You must not, without our prior written consent:

- *(a)* copy, in whole or in part, any of Our Intellectual Property;

- *(b)* reproduce, retransmit, distribute, disseminate, sell, publish, broadcast or circulate any of Our Intellectual Property to any third party; or
- *(c)* breach any intellectual property rights connected with the Application, including (without limitation) altering or modifying any of Our Intellectual Property, causing any of Our Intellectual Property to be framed or embedded in another website, or creating derivative works from any of Our Intellectual Property.

Modifications to the Application

The Company reserves the right to modify, suspend or discontinue, temporarily or permanently, the Application or any service to which it connects, with notice and without liability to you.

Updates to the Application

The Company may from time to time provide enhancements or improvements to the features/functionality of the Application, which may include patches, bug fixes, updates, upgrades and other modifications.

Updates may modify or delete certain features and/or functionalities of the Application. You agree that the Company has no obligation to (i) continue to provide or enable any particular features and/or functionalities of the Application to you.

You further agree that all updates or any other modifications will be (i) deemed to constitute an integral part of the Application, and (ii) subject to the terms and conditions of this Agreement.

Maintenance and Support

The Company provides maintenance and support for licensed customers, customers using the Foundation Edition trial are not entitled to any maintenance or support. To the extent that any maintenance or support is required by applicable law, the Company shall be obligated to furnish any such maintenance or support.

Third-Party Services

The Application may display, include or make available third-party content (including data, information, applications and other products services) or provide links to third-party websites or services.

You acknowledge and agree that the Company shall not be responsible for any Third-party Services. To the maximum extent permitted by law, we shall have no liability for any Third-Party Services, or any unavailability of the Application due to a failure of the Third-Party Services.

Our solution use a number of third party components, which are documented at the following address: https://docs.trackme-solutions.com/latest/third_party_credits.html

By using our product, you also agree to the terms and conditions of these third party components.

Indemnification

Neither party may benefit from the limitations and exclusions set out in this clause in respect of any liability arising from its deliberate default.

The restrictions on liability in this Indemnification clause apply to every liability arising under or in connection with this Agreement including liability in statute, contract, equity, tort (including negligence), misrepresentation, restitution, indemnity or otherwise.

Nothing in this Agreement limits any liability which cannot legally be limited, including liability for:

- *(a)* death or personal injury caused by negligence;
- *(b)* fraud or fraudulent misrepresentation;
- *(c)* breach of the terms implied by section 2 of the Supply of Goods and Services Act 1982 (title and quiet possession); and
- *(d)* defective products under the Consumer Protection Act 1987.

To the maximum extent permitted by law:

- *(a)* you agree to indemnify us for any liability we incur due to your breach of the Acknowledgement and/or Licence clauses and the Intellectual Property clause of this Agreement;
- *(b)* neither party will be liable for Consequential Loss;

- *(c)* each party's liability for any liability under this Agreement will be reduced proportionately to the extent the relevant liability was caused or contributed to by the acts or omissions of the other party or any of that party's personnel, including any failure by that Party to mitigate its losses; and
- *(d)* our aggregate liability for any liability arising from or in connection with this Agreement will be limited to us resupplying the Application to you.

No Warranties

2.1 You represent, warrant and agree that:

- *(a)* you will not use our Application, including Our Intellectual Property, in any way that competes with our business;
- *(b)* if you are agreeing to this Agreement not as an individual but on behalf of your company, government, or other entity for which you are acting (for example, as an employee or governmental official), then you access and use the Application on behalf of that entity;
- *(c)* there are no legal restrictions preventing you from entering into this Agreement;
- *(d)* all information and documentation that you provide to us in connection with this Agreement is true, correct and complete; and
- *(e)* you have not relied on any representations or warranties made by us in relation to the Application (including as to whether the Application is or will be fit or suitable for your particular purposes), unless expressly stipulated in this Agreement.

Severability

If any provision of this Agreement is held to be unenforceable or invalid, such provision will be changed and interpreted to accomplish the objectives of such provision to the greatest extent possible under applicable law and the remaining provisions will continue in full force and effect.

Waiver

Except as provided herein, the failure to exercise a right or to require performance of an obligation under this Agreement shall not affect a party's ability to exercise such right or require such performance at any time thereafter nor shall the waiver of a breach constitute a waiver of any subsequent breach.

Access, Term and Termination

This Agreement shall remain in effect until terminated by you or the Company. The Company may, in its sole discretion, at any time and for any or no reason, suspend or terminate this Agreement with or without prior notice.

We may revoke access to the Application at any time by giving 30 days' written notice to you.

Should we suspect that you are in breach of this Agreement, we may suspend your access to the Application while we investigate the suspected breach. Should we determine that you are in breach of this Agreement, your access to the Application will be terminated immediately.

You may also terminate this Agreement by deleting the Application and all copies thereof from your device or from your computer.

Upon termination of this Agreement, you shall cease all use of the Application and delete all copies of the Application from your device.

Termination of this Agreement will not limit any of the Company's rights or remedies at law or in equity in case of breach by you (during the term of this Agreement) of any of your obligations under the present Agreement

Termination and Exit Strategy

TrackMe Limited provides customers with clear termination rights and an exit strategy to ensure a seamless transition in case of service discontinuation.

Termination Rights:

- Customers may terminate their agreement with TrackMe by providing a written notice of 30 days, to be sent by email to contact@trackme-solutions.com.

- TrackMe reserves the right to terminate the agreement if the customer breaches its terms, fails to comply with applicable regulations, or engages in unauthorized use of the services.

Exit Strategy and Data Retrieval:

- Upon termination, customers will have 30 days to download or request a copy of their data before deletion.
- TrackMe will provide reasonable assistance in transitioning data to another service provider if requested.
- After the grace period, all customer data will be securely deleted, except where legal or regulatory retention requirements apply.

These provisions ensure that customers maintain control over their data and can transition away from TrackMe services with minimal disruption.

Cooperation with Regulatory Authorities

TrackMe Limited is committed to compliance with applicable regulatory frameworks, including the Digital Operational Resilience Act (DORA). As part of this commitment, TrackMe agrees to:

- **Regulatory Requests**: Cooperate with financial regulators, supervisory authorities, and other relevant bodies by providing necessary information related to operational resilience, security, and risk management.
- **Inspections and Assessments**: Allow authorized regulators to conduct inspections and assessments of TrackMe's systems and processes related to ICT risk management.
- **Incident Reporting Assistance**: Provide timely support to financial entities in fulfilling their regulatory obligations, including the reporting of ICT-related incidents.
- **Compliance Documentation**: Maintain detailed records of security, performance, and business continuity measures, making them available to regulators upon request.

These commitments ensure transparency and regulatory compliance while upholding the security and confidentiality of TrackMe's services.

Incident Reporting and Notification

TrackMe Limited prioritizes transparency and accountability in managing ICT-related incidents. In accordance with regulatory requirements, including the Digital Operational Resilience Act (DORA), TrackMe follows a structured incident reporting framework.

Incident Notification Commitments:

- **Significant Incidents**: TrackMe will notify affected customers within 24 hours of detecting a security breach, data compromise, or major service disruption.
- **Regulatory Assistance**: TrackMe will provide customers with relevant details needed to fulfill their regulatory incident reporting obligations.
- **Communication Channels**: Incident notifications will be sent via TrackMe's support helpdesk.
- **Remediation Updates**: TrackMe will provide ongoing updates about mitigation efforts and expected resolution timelines for critical incidents.

TrackMe ensures that all incidents are logged, investigated, and remediated promptly, with continuous improvement to prevent recurrence.

Information Security and Compliance

TrackMe Limited is committed to maintaining high security standards to protect customer data and ensure operational resilience. TrackMe adheres to industry-recognized security frameworks and best practices, including:

- **ISO 27001 Compliance**: TrackMe follows the principles of ISO 27001 for information security management.
- **SOC 2 Controls**: TrackMe implements security, availability, and confidentiality controls aligned with SOC 2 standards.
- **Regular Security Audits**: TrackMe conducts periodic internal and external security audits to validate compliance with regulatory and security requirements.

- **Secure Development Practices**: TrackMe employs secure coding practices, vulnerability testing, and continuous monitoring to safeguard its applications.
- **Customer Assurance**: Upon request, TrackMe can provide security documentation detailing compliance measures and risk management processes.

By adhering to these standards, TrackMe ensures the security, integrity, and confidentiality of its services, in compliance with DORA and other applicable regulations.

Access and Audit Rights

TrackMe Limited acknowledges the importance of regulatory compliance and operational resilience. To ensure adherence to applicable financial regulations, including the Digital Operational Resilience Act (DORA), TrackMe grants authorized financial entities and regulatory authorities the right to conduct audits and assessments of TrackMe's services related to operational resilience, cybersecurity, and incident response.

Such audits shall be conducted with reasonable notice and during normal business hours. TrackMe shall provide all necessary documentation, access to relevant systems, and support to facilitate these audits, ensuring transparency and compliance with applicable laws.

Financial entities using TrackMe's services may also conduct internal assessments, subject to confidentiality and security agreements.

Service Performance Standards

TrackMe Limited is committed to maintaining a high standard of service performance to support operational resilience and regulatory compliance. As part of this commitment, TrackMe ensures the following service levels:

- **Availability**: TrackMe aims to maintain an uptime of 99.5% measured on a monthly basis, excluding planned maintenance periods.
- **Response Time**: TrackMe's support team will respond to critical service issues within twenty fours (24) business hours.
- **Maintenance and Updates**: Regular maintenance and security updates will be communicated in advance, with minimal disruption to service.
- **Data Processing Performance**: TrackMe ensures that all data processing activities meet industry standards for accuracy and efficiency.

Any deviations from these service levels will be investigated and remediated promptly. TrackMe Limited will also provide customers with performance reports upon request.

Service Locations and Data Processing

TrackMe Limited operates its services in secure data centers located in the United Kingdom and EU. Customer data may be processed and stored in these locations based on applicable regulatory and compliance requirements.

- **Primary Hosting Locations**: TrackMe Limited uses cloud service providers and data centers located in [List countries or regions].
- **Data Transfers**: Any data transfers between regions comply with applicable data protection laws, including the General Data Protection Regulation (GDPR) where applicable.
- **Subcontractors**: TrackMe may engage third-party cloud providers or partners for infrastructure services. A list of such subcontractors is available upon request.
- **Customer Control**: Customers may request information on where their specific instance of TrackMe services is hosted.

TrackMe Limited ensures that all service locations meet high security and operational resilience standards.

Data Protection and Confidentiality

TrackMe Limited is committed to protecting the confidentiality, integrity, and security of customer data. All data collected, processed, and stored within TrackMe services is handled in compliance with applicable data protection laws, including but not limited to the General Data Protection Regulation (GDPR).

Key Data Protection Measures: - **Data Encryption**: Customer data is encrypted at rest and in transit using industry-standard encryption protocols. - **Access Controls**: Strict access controls ensure that only authorized personnel can access sensitive data. - **Data Minimization**: TrackMe collects and processes only the minimum necessary customer data required for service operation. - **Confidentiality Commitments**: TrackMe employees, contractors, and third-party service providers are bound by confidentiality obligations. - **Customer Rights**: Customers have the right to request information on how their data is processed and may request data deletion or correction.

TrackMe Limited continuously evaluates and improves its security measures to maintain compliance with evolving regulatory requirements.

Business Continuity and Disaster Recovery

TrackMe Limited is committed to ensuring the resilience and availability of its services, even in the event of disruptions. To maintain operational stability, TrackMe has implemented comprehensive business continuity and disaster recovery plans.

Key Resilience Measures: - **Redundant Infrastructure**: TrackMe operates on a highly available infrastructure with redundant components to minimize downtime. - **Regular Backups**: Customer data is backed up at regular intervals to prevent data loss and ensure quick recovery in case of incidents. - **Incident Response Procedures**: TrackMe maintains a structured incident response framework to detect, investigate, and mitigate potential threats. - **Disaster Recovery Plan**: A formal disaster recovery plan is in place, ensuring timely restoration of services in the event of system failures. - **Testing and Reviews**: Continuity plans and recovery strategies are regularly tested and updated to adapt to evolving risks.

In case of a major disruption, TrackMe will promptly inform customers and provide updates on recovery efforts.

Governing Law

The laws of the Country, excluding its conflicts of law rules, shall govern this Agreement and your use of the Application. Your use of the Application may also be subject to other local, state, national, or international laws.

Entire Agreement

The Agreement constitutes the entire agreement between you and the Company regarding your use of the Application and supersedes all prior and contemporaneous written or oral agreements between you and the Company.

You may be subject to additional terms and conditions that apply when you use or purchase other Company's services, which the Company will provide to you at the time of such use or purchase.

Assignment

You must not assign or deal with the whole or any part of your rights or obligations under this Agreement without our prior written consent.

Contracts (Rights of Third Parties) Act 1999

Notwithstanding any other provision of this Agreement, nothing in this Agreement confers or is intended to confer any right to enforce any of its terms on any person who is not a party to it.

Contact Us

If you have any questions about this Agreement, you can contact Us:

- By visiting this page on our website: www.trackme-solutions.com
- By sending us an email: contact@trackme-solutions.com

4.2 Terms & Conditions

`Download the Terms & Conditions as a text file`

Our Terms & Conditions were last updated on 25th of January 2026.

4.2.1 TrackMe Limited IT Software Terms and Conditions

In these terms and conditions (Terms), when we say you or your, we mean both you and, if applicable, any entity you are authorised to represent (such as your employer). When we say we, us, or our, we mean TrackMe Limited, a company registered in England and Wales with company number 13817409.

Please read these Terms carefully before using our downloadable platform ("TrackMe") for Splunk Enterprise, Splunk Cloud software or before accessing TrackMe through our direct website.

These Terms form our contract with you, and set out our obligations as a service provider and your obligations as a customer. You cannot use TrackMe unless you agree to these Terms. By using our software, you agree to be bound by these Terms.

1. License Types & Restrictions

You and Each Authorised User must be at least 16 years old to use TrackMe. If you are using TrackMe on behalf of your employer or a business entity, you, in your individual capacity, represent and warrant that you are authorised to act on behalf of your employer or the business entity and to bind the entity and the entity's personnel to these Terms.

You represent, warrant and agree that:

- *(a)* you will not use TrackMe, including Our Intellectual Property, in any way that competes with our business;
- *(b)* there are no legal restrictions preventing you from entering into these Terms;
- *(c)* all information and documentation that you provide to us in connection with these Terms is true, correct and complete; and
- *(d)* you have not relied on any representations or warranties made by us in relation to TrackMe (including as to whether TrackMe is or will be fit or suitable for your particular purposes), unless expressly stipulated in these Terms.

You must not:

- *(a)* access or use TrackMe in any way that is improper or breaches any laws, infringes any person's rights (for example, intellectual property rights and privacy rights), or gives rise to any civil or criminal liability;
- *(b)* interfere with or interrupt the supply of TrackMe, or any other person's access to or use of TrackMe;
- *(c)* introduce any viruses or other malicious software code into TrackMe;
- *(d)* use any unauthorised or modified version of TrackMe, including but not limited to for the purpose of building similar or competitive software or for the purpose of obtaining unauthorised access to TrackMe;
- *(e)* attempt to access any data or log into any server or account that you are not expressly authorised to access;
- *(f)* use TrackMe in any way that involves service bureau use, outsourcing, renting, reselling, sublicensing, concurrent use of a single user login, or time-sharing;
- *(g)* circumvent user authentication or security of any of our networks, accounts or hosts or those of any third party; or
- *(h)* access or use TrackMe to transmit, publish or communicate material that is, defamatory, offensive, abusive, indecent, menacing, harassing or unwanted.

License Types:

While you have an account with Splunk or with us (Account), we grant you a right to use TrackMe (which may be suspended or revoked in accordance with these Terms). This right cannot be passed on or transferred to any other person. The types of licences that we offer will be set out on our website or on the Splunk Enterprise or Splunk Cloud software (as applicable).

2. Support

We offer support for features and behaviors which relate directly to the usage of TrackMe for Splunk Cloud and Splunk Enterprise. Issues related to Splunk products or any other third-party applications are not covered by our support agreement. Prioritized support is included for customers with a valid TrackMe Enterprise or TrackMe Unlimited. Support is available by email. We will use reasonable endeavours to respond to all email enquiries within 24 hours.

Contacting TrackMe support:

- Customers with a valid TrackMe Enterprise or TrackMe Unlimited can contact us at support@trackme-solutions.com
- If the support case requires it, we will propose a virtual support meeting through our Zoom meeting services, or your own meeting tools depending on your preferences

Prioritized developments:

- If you have specific needs or requirements that are not covered by our product, we will prioritize developments if you are a customer under active subscription.
- The company remains the sole owner of the developments and the intellectual property of the developments made for the product.
- We also reserve the right not to implement a feature if it is not in line with our product roadmap or if it is not technically feasible, although we will always do our best to meet your requirements.

Unless we agree otherwise, the support we may agree to provide cannot be used to support any other products or services, and does not include training, installation of software or hardware, software development or the modification, deletion or recovery of data or any on-site services.

Additional Services: If you require additional services, we may, in our sole discretion, provide such additional services (to be scoped and priced in a separate contract provided by us).

Third Party Products or Services: Where you engage third parties to operate alongside our services (for example, any third-party software systems you wish to integrate with TrackMe), those third parties are independent of us and you are responsible for (meaning we will not be liable for) the goods or services they provide, unless we expressly agree otherwise.

3. License Subscription Renewal and Expiration

If you wish not to renew your TrackMe Foundation, TrackMe Enterprise or TrackMe Unlimited license, TrackMe will automatically transition to the Foundation Edition trial mode (read-only after expiry), and components out of the scope of the Foundation Edition will automatically be disabled. If you renew your subscription, your license will be extended accordingly. You are responsible for paying any levies or taxes associated with your use of TrackMe, for example sales taxes, value-added taxes or withholding taxes (unless we are required by law to collect these on your behalf.

4. Changes to TrackMe

While we strive to always make TrackMe available to you, we do not make any guarantees that it will be available 100% of the time. Our services may be disrupted during certain periods, including, for example, as a result of scheduled or emergency maintenance. Our services (including TrackMe) may interact with, or be reliant on, products or services provided by third parties, such as cloud hosting service providers. To the maximum extent permitted by law, we are not liable for disruptions or downtime caused or contributed to by these third parties. We will try to provide you with reasonable notice, where possible, of any disruptions to your access to TrackMe.

Minor changes to TrackMe: We may change TrackMe:

- *(a)* to reflect changes in relevant laws and regulatory requirements; and
- *(b)* to implement minor technical adjustments, improvements, and to keep up-to-date with technological advancements. These changes will not substantially affect your use of TrackMe.

More significant changes to TrackMe: We will try to avoid making any significant changes to TrackMe which are likely to materially disadvantage your use of TrackMe. However, where we intend to make a change to TrackMe which may materially disadvantage your use of TrackMe, we will notify you in advance of making any changes, and you may then contact us to terminate these Terms and receive a full refund for any unused services (if applicable) before the changes take effect.

5. Changes to the Terms

We reserve the right to modify, amend, or update these Terms at any time and for any reason without prior notice. Such modifications, amendments, or updates will be effective immediately upon posting on our website.

To ensure transparency and keep our customers fully informed of any changes that may affect their use of the TrackMe software for Splunk Enterprise and Splunk Cloud, we commit to sending an automated notification of any such

changes to the designated point of contact for each customer. This notification will be sent to the email address on file with us for such communications.

Customers are encouraged to periodically review the most current version of the Terms, which will always be available on our website. Continued use of the TrackMe software after any changes to the Terms signifies acceptance of those changes. It is the responsibility of the customer to update their designated point of contact and corresponding email address as necessary to ensure that notifications of changes are received in a timely manner.

If you disagree with the revised Terms, you have the right to terminate your use of the TrackMe software and we agree to refund you for any prepaid unused Fees on a pro-rata basis.

6. Payments

The license can be activated in your Splunk environment at any time, with or without online verification through Cryptolens Licensing API.

For more details and technical information about the Cryptolens API service and our registration process, please consult the following documentation: https://docs.trackme-solutions.com/latest/license_registration.html

We provide a license key which requires online verification through Cryptolens Licensing API, and a license file which allows offline registration for Splunk environments with no outside communications capabilities.

Once you have chosen the licence that you require, you agree to pay the fees set out on TrackMe (Fees) by the date specified on TrackMe to access certain features of TrackMe. You agree that the Fees will be payable in advance of you receiving access to TrackMe and your chosen licences (if licence Fees are payable). We accept payments by bank transfer once we issue a tax invoice (U.K. national and international IBAN, payments can be processed in USD, GBP AND EUR currencies).

7. Delivery

After you agree to our Terms and accept our quotation, we will issue an invoice through our billing system. Once the invoice is generated and the Fees are paid as set out above, we will provide you with the TrackMe license key and license file via our licensing services.

The TrackMe Enterprise or TrackMe Unlimited license key and license key file which allows offline registration is provided by electronic means, we will also deliver the license through our DocuSign service to electronically sign the delivery.

8. Authorised Users

You may be permitted to invite a number of users to TrackMe, who will be permitted to download and use TrackMe under your license (**Authorised Users**).

You must ensure that each Authorised User complies with these Terms and our End User Licence Agreement. You are responsible and liable for the acts or omissions of your Authorised Users.

Each Authorised User must agree to our End User Licence Agreement (as available at [insert URL]) in order to download and use TrackMe.

9. Legal Compliance and Jurisdiction

We are committed to ensuring that any information we store is secure and required for the purposes of providing our services to you.

As a Software which runs on your environment, whenever you are a Splunk Enterprise or Splunk Cloud customer, we do not store any personal data, and we do not have access to your data.

Information stored by us is restricted to the minimal required information for support and billing purposes notably the end customer name and contact details and the TrackMe Enterprise or TrackMe Unlimited license details.

10. Termination and Refund Policy

This clause defines the conditions under which subscriptions may be terminated and the applicable refund policy for TrackMe licenses purchased through our standard ordering channels.

Refund Policy (30-day window):

Customers may request a full refund within **thirty (30) days** from the date of purchase. Refund requests made within this period will be processed in full. After the 30-day refund window has elapsed, all fees paid are non-refundable.

Refunds are handled through the original payment method and processed by our authorized payment provider.

Subscription Termination by the Customer:

TrackMe subscriptions are non-renewing by default. Customers may choose not to renew their subscription without formal notice. If a customer wishes to discontinue use of TrackMe before the end of the subscription term, access will remain available until the subscription expires. Early termination does not entitle the customer to a refund beyond the defined 30-day refund window.

Termination by TrackMe:

We reserve the right to terminate or suspend a subscription in cases of material breach of these Terms, misuse of the software, or failure to comply with payment obligations. In such cases, no refund will be provided except where required by applicable law.

Subscription Expiration:

Upon subscription expiration, access to paid features will be suspended and the software may revert to a limited or free edition where applicable.

Policy Changes:

We reserve the right to update this policy from time to time. Material changes will be communicated via our website or direct notification where appropriate.

If you have any questions regarding subscription termination or refunds, please contact us at contact@trackme-solutions.com.

11. Intellectual Property

You acknowledge and agree that any Intellectual Property or content (including copyright and trademarks) available on TrackMe, TrackMe itself (including how it looks and functions), any algorithms or machine learning models used on TrackMe, as well as our copyrighted works, trademarks, inventions, designs and other intellectual property (**Our Intellectual Property**) will at all times vest, or remain vested, in us.

We authorise you to use and download Our Intellectual Property solely for your limited commercial use. You must not exploit Our Intellectual Property for any other purpose, nor allow, aid or facilitate such use by any third party. Use must be limited to Authorised Users on devices that are controlled or approved by you.

You must not, without our prior written consent:

- *(a)* copy, in whole or in part, any of Our Intellectual Property;
- *(b)* reproduce, retransmit, distribute, disseminate, sell, publish, broadcast or circulate any of Our Intellectual Property to any third party; or
- *(c)* breach any intellectual property rights connected with TrackMe, including (without limitation) altering or modifying any of Our Intellectual Property, causing any of Our Intellectual Property to be framed or embedded in another website, or creating derivative works from any of Our Intellectual Property.

This Clause 11 will survive the termination or expiry of these Terms.

12. Liability

Neither party may benefit from the limitations and exclusions set out in this clause in respect of any liability arising from its deliberate default.

The restrictions on liability in this 'Liability' clause apply to every liability arising under or in connection with these Terms including liability in statute, contract, equity, tort (including negligence), misrepresentation, restitution, indemnity or otherwise.

Nothing in these Terms limits any Liability which cannot legally be limited, including Liability for:

- *(a)* death or personal injury caused by negligence;
- *(b)* fraud or fraudulent misrepresentation; and
- *(c)* defective products under the Consumer Protection Act 1987.

To the maximum extent permitted by law, we shall have no Liability for any third party products or services, or any unavailability of TrackMe due to a failure of the third party products or services.

To the maximum extent permitted by law:

- *(a)* you agree to indemnify us for any Liability we incur due to your breach of these Terms;
- *(b)* neither party will be liable for any Consequential Loss; and
- *(c)* a party's liability for any Liability under these Terms will be reduced proportionately to the extent the relevant liability was caused or contributed to by the acts or omissions of the other party, including any failure by that other party to mitigate its loss; and
- *(d)* our aggregate liability for any Liability arising from or in connection with these Terms will be limited to us resupplying the services to you or, in our sole discretion, to us repaying you the amount of the fees paid by you to us in respect of the supply of the relevant services to which the Liability relates.

The terms implied by sections 3, 4 and 5 of the Supply of Goods and Services Act 1982 are, to the maximum extent permitted by law, excluded from these Terms.

This 'Liability' clause will survive the termination or expiry of these Terms.

13. General

Assignment: Subject to the below clause, a party must not assign or deal with the whole or any part of its rights or obligations under these Terms without the prior written consent of the other party (such consent is not to be unreasonably withheld).

Assignment of Debt: You agree that we may assign or transfer any debt owed by you to us, arising under or in connection with these Terms, to a debt collector, debt collection agency, or other third party.

Contracts (Rights of Third Parties) Act 1999: Notwithstanding any other provision of these Terms, nothing in these Terms confers or is intended to confer any right to enforce any of its terms on any person who is not a party to it.

Disputes: Neither we or you may commence court proceedings relating to any dispute, controversy or claim arising from, or in connection with, these Terms (including any question regarding its existence, validity or termination) (Dispute) unless we and you first meet (in good faith) to resolve the Dispute. Nothing in this clause will operate to prevent us or you from seeking urgent injunctive or equitable relief from a court of appropriate jurisdiction. If the Dispute is not resolved at that initial meeting:

- *(a)* where you are resident or incorporated in England and Wales, refer the matter to mediation, administered by The Centre for Effective Dispute Resolution; or
- *(b)* where you are not resident or incorporated in England and Wales, refer the matter to arbitration administered by the London Court of International Arbitration (LCIA), with such arbitration to be conducted in London, before one arbitrator, in English and in accordance with the LCIA Arbitration Rules.

Force Majeure: To the maximum extent permitted by law, we shall have no Liability for any event or circumstance outside of our reasonable control.

Marketing: You agree that we may send you electronic communications about our products and services. You may opt-out at any time by using the unsubscribe function in our electronic communications

Governing law: These Terms are governed by the laws of England and Wales. Each party irrevocably and unconditionally submits to the exclusive jurisdiction of the courts operating in England and Wales and any courts entitled to hear appeals from those courts and waives any right to object to proceedings being brought in those courts. TrackMe may be accessed in the UK and overseas. We make no representation that TrackMe complies with the laws (including intellectual property laws) of any country outside of the UK. If you access TrackMe from outside the UK, you do so at your own risk and are responsible for complying with the laws in the place you access TrackMe. The United Nations Convention of Contracts for the International Sale of Goods is expressly excluded from these Terms.

Privacy: All personal data you and your Authorised Users provide to us will be treated in accordance with our privacy policy. You can find our privacy policy at https://trackme-solutions.com/privacy-policy/.

Severance: If a provision of these Terms is held to be void, invalid, illegal or unenforceable, that provision is to be read down as narrowly as necessary to allow it to be valid or enforceable, failing which, that provision (or that part of

that provision) will be severed from these Terms without affecting the validity or enforceability of the remainder of that provision or the other provisions in these Terms.

Third party sites: TrackMe may contain links to websites operated by third parties. Unless we tell you otherwise, we do not control, endorse or approve, and are not responsible for, the content on those websites. We recommend that you make your own investigations with respect to the suitability of those websites. If you purchase goods or services from a third party website linked from TrackMe, such third party provides the goods and services to you, not us. We may receive a benefit (which may include a referral fee or a commission) should you visit certain third-party websites via a link on TrackMe (Affiliate Link) or for featuring certain products or services on TrackMe. We will make it clear by notice to you which (if any) products or services we receive a benefit to feature on TrackMe, or which (if any) third party links are Affiliate Links.

14. Definitions

Consequential Loss includes any consequential loss, indirect loss, real or anticipated loss of profit, loss of benefit, loss of revenue, loss of business, loss of goodwill, loss of opportunity, loss of savings, loss of reputation, loss of use and/or loss or corruption of data, whether under statute, contract, equity, tort (including negligence), indemnity or otherwise.

Intellectual Property means any copyright, registered or unregistered designs, patents or trade marks, business names, get-up, goodwill, domain names, know-how, inventions, processes, trade secrets or confidential information, circuit layouts, software, computer programs, databases or source codes, including any application for registration of, and any improvements, enhancements or modifications of, the foregoing, and any right to apply for and be granted, renewals or extensions of, and rights to claim priority from, such rights and all similar or equivalent rights or forms of protection which subsist or will subsist now or in the future, including in respect of the foregoing.

Liability means any expense, cost, liability, loss, damage, claim, notice, entitlement, investigation, demand, proceeding or judgment (whether under statute, contract, equity, tort (including negligence), misrepresentation, restitution, indemnity or otherwise), howsoever arising, whether direct or indirect and/or whether present, unascertained, future or contingent and whether involving a third party or a party to these Terms or otherwise.

4.3 Support

4.3.1 Contacting Support

All support requests must be directed to our support team through the following channels:

- **Support Portal:** https://support.trackme-solutions.com
- **Email:** support@trackme-solutions.com

For licensing, sales, or general inquiries, please contact:

- **Email:** contact@trackme-solutions.com

> **Note**
>
> Support requests submitted through any channel other than the support portal or the support email address above will not be processed.

4.3.2 Support Portal Access

As a registered TrackMe customer, one or more points of contact within your organisation will be identified, and personal access credentials will be provided for our support portal.

Once connected to the support portal, you can:

- Open a new support request
- Review and manage the status of your existing support requests

If your organisation does not yet have a registered account, please send an email to support@trackme-solutions.com **and we will create one for you.**

Once your request has been received and processed, an activation email will be sent with your personal access credentials.

You can also submit a support case by emailing support@trackme-solutions.com **directly; a support request will be created automatically and associated with your user account.**

4.3.3 Minimum Version Requirement

Important

To ensure efficient support delivery, we require customers to be running **TrackMe version 2.3.x or later**. If you are running an older version, upgrading to the latest release will very likely be requested as a first step before investigating any reported issue.

4.3.4 SLA, Response Time and Support Coverage

- We provide support for features and behaviours that relate directly to the usage of TrackMe for Splunk Cloud and Splunk Enterprise.
- Issues related to Splunk products or any other third-party applications are not covered by our support agreement.
- Prioritised support is included for all customers with a valid subscription.
- Support is available by email with a Service Level Agreement (SLA) of 24 hours for a support request to be acknowledged, Monday to Friday, 8 am to 8 pm UK time.
- If a support case requires it, we will propose a virtual meeting through our Zoom meeting services or your own meeting tools if preferred.
- If you have specific needs or requirements that are not covered by our product, we will prioritise developments for customers under an active subscription.

4.3.5 Support Diagnostic Archive

When investigating a customer-reported issue, our support team will frequently ask for a **Support Diagnostic Archive**. The TrackMe app ships with a self-service page that generates this archive directly from your Splunk environment, so you no longer need to run ad-hoc searches or export KV Store collections by hand.

Where to find it

The page is available in the TrackMe app navigation under both:

- **API & tooling → Support - generate diags**
- **Audit & troubleshoot → Support - generate diags**

It requires the `trackme_power` or `trackme_admin` role.

What it does

The page offers two diagnostic modes:

- **Entity diag** — for when the question is about one (or a small number of) specific entities in a given tenant. Pick the tenant, the component, then up to 20 entities. The archive will include each entity's record, the last 7 days of metrics, and the most recent summary / notable events for those entities.
- **Global diag** — for when the question is about the system as a whole (degraded scheduler, slow trackers, generic skipping). The archive will include the central KV Store collections, knowledge objects for every enabled tenant, the last 24 hours of scheduler skipping and performance reports, the last 24 hours of TrackMe runtime metrics, and the last 10 000 ERROR events across the TrackMe REST API, custom commands, and alert actions over the past 7 days.

In both cases the result is a single timestamped `.tgz` archive that you can attach to the support ticket on the portal or send by email.

Note

Nothing is collected or transmitted automatically. The page only runs the selected collection on your Splunk instance, packages the result locally, and streams the archive to your browser. No data leaves the instance until you choose to share the file with TrackMe Support.

Optional anonymisation

If your environment uses tenant names that are sensitive or non-shareable, tick the **Anonymise tenant names** checkbox before clicking *Generate diagnostic archive*. The worker will rewrite every occurrence of every tenant id and tenant alias to a stable random alias (for example `tenant_anon_0003_ab12cd`) across every file in the archive — including filenames — and present the original-to-anonymised mapping to you on screen once the archive is ready. The mapping is **never written into the archive itself**; you keep a local copy and share it with TrackMe Support separately if and when needed.

Global diag



Entity diag



Tip

When you open a support case, our team may ask you to attach a fresh diagnostic archive — generated as close as possible to the time the issue was last observed. Running the page yourself before contacting support, and attaching the archive to your initial ticket, will significantly accelerate triage.

4.3.6 Reporting Bugs and Enhancement Requests

To report a bug or submit a request for enhancement, please contact us through:

- **Support Portal:** https://support.trackme-solutions.com
- **Email:** support@trackme-solutions.com

Our team will triage your request and keep you informed of progress through the support portal.

4.3.7 Community Resources

Splunk Community Slack:

- https://splunk-usergroups.slack.com

Splunk Community:

- https://community.splunk.com

Note

Community channels are provided for general Splunk discussions and peer-to-peer assistance. For TrackMe-specific support, please use the official support channels listed above.

CHAPTER

FIVE

COMPATIBILITY AND DOWNLOAD:

5.1 Compatibility

Important

We strongly recommend running the latest release of TrackMe at all times.

Keeping TrackMe up to date ensures you benefit from the latest features, bug fixes, security patches, and performance improvements. If a support case is opened and the running version is identified as significantly outdated, **an upgrade to a current release may be required before further investigation can proceed**.

5.1.1 Splunk core compatibility

Splunk core

- TrackMe is compatible with Splunk 9.1 and later.
- TrackMe is developed, tested and qualified against latest generation of Splunk Enterprise and Splunk Cloud Victoria in priority.

5.1.2 Splunk Enterprise and Splunk Cloud compatibility

Splunk Enterprise compatibility

- TrackMe is compatible with Splunk Enterprise from Splunk 9.2.x
- TrackMe fully supports Python 3.9.x/3.13.x (which are both used by currently support main Splunk versions) and later.

Operating system compatibility for Splunk Enterprise customers

OS compatibility for Splunk Enterprise customers

- TrackMe only officially supports **Linux** as the Operating System for Splunk Enterprise customers.
- On Windows and MacOS, TrackMe might partially work, but advanced features such as Machine Learning Outliers detection will not be available.

Splunk Cloud compatibility

- TrackMe is compatible with Splunk Cloud Victoria and Splunk Cloud Classic
- On Splunk Cloud Classic, TrackMe Version 2.0.60 and later is required to address some issues due to Splunk SLIM partitioning

Appinspect and vetting

We carefully comply with Splunk development best practices to ensure that we provide the highest level of application quality.

Therefore, TrackMe releases are reviewed and validated with Splunk Appinspect:

- https://dev.splunk.com/enterprise/reference/appinspect

When a new release of TrackMe is published, our automation also processes the release through Appinspect automation and the Appinspect HTML report is available in the release directory of our download Website:

- https://downloads.trackme-solutions.com/

TrackMe is compatible with Splunk Enterprise and Splunk Cloud products.

5.1.3 Python compatibility

Python 3 compatibility

- TrackMe supports **Python 3** only.
- TrackMe is fully compatible with **Python 3.9.x/3.13.x** (which are both used by currently support main Splunk versions) and later.

5.1.4 Artificial Intelligence features compatibility

TrackMe AI features — Splunk & Python requirements

- **TrackMe remains fully compatible with all supported Splunk releases.** The AI features are additive and opt-in; the Python/Splunk requirements below do not affect TrackMe's core data-monitoring capabilities.
- The **AI Assistant** (conversational chat) and the **AI Concierge** are compatible with **Python 3.9.x and later** — i.e. they run on the same Splunk releases as the rest of TrackMe.
- The **AI Advisors** (ML, Feed Lifecycle, FLX Threshold, FQM, and Component Health) are "agentic" features built on the Splunk Agent SDK and require **Python 3.13.x**, which ships only with **Splunk 10.2.x and later** (Splunk Enterprise or Splunk Cloud). These advanced AI features are therefore only available on Splunk 10.2.x and later.
- **AI Routines** and the **AI Agents for findings** (one per dedicated tenant type: UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) are built on the same SDK and have the same requirement: Splunk 10.2.x and later.
- **Automatic safeguards:** TrackMe automatically prevents an Advisor from being invoked when the Python version requirement is not met, returning a clear message about the incompatibility with the running Splunk release. In addition, the AI Assistant will not propose an AI Advisor when it detects that the system is not running a compatible release.

In summary: TrackMe remains fully compatible with previous Splunk releases (Python 3.9.x), including the AI Assistant and the AI Concierge. The advanced AI Advisors require Splunk 10.2.x and later (Python 3.13.x). See *Artificial Intelligence in TrackMe* for the full AI feature set.

5.1.5 Dedicated tenant types compatibility

User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge

- These tenant types require the **Enterprise** or **Unlimited** edition (trial and Developer licences included); they are not available on the Foundation edition.
- **User Activity Monitoring** reads `_audit`, `_introspection` and the search completion records of the indexers. Splunk Enterprise 9.4, 10.4 and 10.6 and Splunk Cloud 10.5 are qualified; other releases run as well and only add a soft diagnostic in Collection health. See *User Activity Monitoring*.
- **Cribl Stream** and **Cribl Edge** monitor a Cribl.Cloud workspace or a self-managed Leader, through a read-only API account, and read Cribl's internal metrics from a Splunk metric index. See *Cribl Stream*, *Cribl Edge* and *Cribl connections*.
- **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet** and **Splunk Forwarder Collectors** read the telemetry Splunk already writes (`_internal`, `_introspection`, `_metrics` for S2S when used, the Deployment Server indexes for UF Fleet, `_configtracker` for Forwarder Collectors, which needs Splunk 9.0 or later on the collectors), on this search tier or through a TrackMe remote account, Splunk Cloud included. Nothing is installed on the instances they watch. See *Splunk HEC*, *Splunk S2S*, *Splunk UF Fleet* and *Splunk Forwarder Collectors*.

5.1.6 TrackMe Broker compatibility

The TrackMe Broker add-on

- The add-on runs on Splunk Enterprise 9.4.x and later, on a Heavy Forwarder on your side. It refuses to start on a Splunk Cloud instance.
- It needs add-on version 1.1.1 or later on the forwarder, and TrackMe 2.4.24 or later on the stack.
- On the stack, it works on Splunk Cloud, including on a search head cluster.
- It uses the `cryptography` library from the Python of the Splunk instance it runs on, and does not package its own.
- It is compatible with Splunk FIPS mode, on Splunk 9.4.x and later. On a forwarder, the algorithms run under Splunk's FIPS module. On the stack, they run in the library TrackMe packages: see *the security page*.
- See *TrackMe Broker* for what it needs and how to set it up, and *Release notes: trackme-broker* for each release.

5.1.7 FIPS Mode compatibility

TrackMe and FIPS mode

- **Since TrackMe version 2.0.99, TrackMe is officially compatible with Splunk FIPS mode, and continuously validated on FIPS.**
- About FIPS: https://docs.splunk.com/Documentation/Splunk/latest/Security/SecuringSplunkEnterprisewithFIPs
- In Splunk 9.3.0, several changes were made and some aspects of FIPS are correctly applied, this requires TrackMe 2.0.99 and evolutions we have made to rely on sha256 instead of md5 crypto in various search and Python logics.
- For the TrackMe Broker add-on, see *TrackMe Broker compatibility*.

5.1.8 Web Browser compatibility

TrackMe is supported on Splunk supported Web Browsers:

See: https://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements

Supported Web Browsers:

- Firefox (latest)
- Safari (latest)
- Chrome (latest)
- Microsoft Edge (latest)

5.2 Download

5.2.1 Product brochure

A nine-page overview of TrackMe for Splunk: what it watches, the two ways of watching (entities and findings), Topology Studio, the AI layer, operating at scale and the editions.

- `TrackMe for Splunk, product brochure (PDF)`

5.2.2 TrackMe for Splunk Enterprise & Splunk Cloud

TrackMe for Splunk Enterprise & Splunk Cloud can be downloaded from Splunk Base, as well as from our release Website:

- https://splunkbase.splunk.com/app/4621
- https://downloads.trackme-solutions.com

5.2.3 TrackMe App on SOAR

We provide a TrackMe application for Splunk SOAR (Phantom), this application for SOAR allows powerful actions to automate and interact with TrackMe straight from Splunk SOAR:

- https://downloads.trackme-solutions.com/trackme-app-on-soar

You can also access the source code and the releases from our GitHub repository:

- https://github.com/trackme-limited/trackme-app-on-soar

5.2.4 TrackMe Broker

The TrackMe Broker add-on runs on a Heavy Forwarder on your side and carries TrackMe's API calls from Splunk Cloud to destinations on your network. It can be downloaded from Splunk Base, as well as from our release Website, where each release sits next to its checksum:

For more information, see: *TrackMe Broker*, and *Release notes: trackme-broker* for what each release changed.

- https://splunkbase.splunk.com/app/10323
- https://downloads.trackme-solutions.com/trackme-broker/

5.2.5 TrackMe Configuration Manager (TCM)

The TrackMe Configuration Manager (TCM) is a Splunk Application which can be used for CI/CD and configuration management purposes of TrackMe, it acts as a utility which can receive and replay configuration transactions, to promote changes between environments and also to restore configuration steps by replaying these, it can be downloaded from our release Website.

For more information about the TCM, see: *Configuration Manager (CI/CD)*

- https://downloads.trackme-solutions.com/TA-trackme-conf-manager

You can also access the source code and the releases from our GitHub repository:

- https://github.com/trackme-limited/TA-trackme-conf-manager

5.2.6 TA-trackme-cribl

This Splunk application provides API integration between Splunk and Cribl API, it acts as an SPL interface to the Cribl API so you can even more interactions between the platforms, leverage real time requests from Splunk dashboard and also leverage it in TrackMe such as custom Flex Object trackers, it can be downloaded from:

- https://downloads.trackme-solutions.com/TA-trackme-cribl

You can also access the source code and the releases from our GitHub repository:

- https://github.com/trackme-limited/TA-trackme-cribl

CHAPTER

SIX

REQUIREMENTS:

6.1 Requirements for TrackMe

6.1.1 Target for deployments

For Splunk Enterprise customers

For Splunk on-premise customers, we recommend to deploy TrackMe on:

- A dedicated Search Head or Search Head Cluster, especially if you intend to monitor a large number of environments or if you are a large scale Splunk customer
- Alternatively, you can deploy TrackMe on the instance used for the purposes of the Splunk Monitoring Console

Splunk premium applications

- We do not support colocating TrackMe with any of the Splunk premium applications (Splunk Enterprise Security, Splunk ITSI)
- The reason is that both TrackMe and premium applications can have heavy requirements in term of workload, and TrackMe could therefore slightly impact these products

For Splunk Cloud customers

From Splunk Cloud Victoria, applications are deployed through self-services equally on all Search Heads, including Premium applications Search Head:

- When deploying TrackMe on Splunk Cloud, the application will be deployed automatically on all Search Heads, including your Premium application Search Heads, if any
- However, a virgin deployment of TrackMe has no workload as long as there are no Virtual Tenants created
- TrackMe should be configured and used from the first Search Head tier of your Splunk Cloud stack, commonly called the Ad-hoc Search Head tier

Splunk premium applications

- Similarly to on-premise deployments and while TrackMe is deployed on the premium applications Search Heads, we do not support running TrackMe on these Search Heads
- The reason is that both TrackMe and premium applications can have heavy requirements in term of workload, and TrackMe could therefore slightly impact these products

6.1.2 System requirements for the use of TrackMe

Supported Operating Systems

Warning

Linux is the only fully supported Operating System for TrackMe.

All other environments (Windows, MacOS) are provided on a **best effort** basis only. We develop, test, and qualify TrackMe exclusively on Linux and Splunk Cloud Victoria. If you encounter issues on a non-Linux platform, we will do our best to assist, but resolution cannot be guaranteed.

- **Linux** – fully supported (strongly recommended, and used by the vast majority of customers)
- **MacOS** – best effort support only (suitable for development and testing purposes)
- **Windows** – best effort support only (not recommended; we do not expect good performance from Splunk on Windows in general)

Containerized computing platforms

Similarly to Splunk Enterprise, running TrackMe on containerized environment is supported:

- https://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements

Resources requirement

The requirements for TrackMe in terms of CPU slightly relies on its usage and configuration, the main factors are:

- If the Splunk instance hosting TrackMe is itself more than a Search Head (if the instance is as well an indexer for instance in the case of all in one Splunk instance, which is not recommended)
- The number of enabled TrackMe Virtual Tenants
- The number of enabled Hybrid, Flex and Elastic Trackers
- The number of entities monitored, which influence factors like the number of Machine Learning models to be created and maintained for instance
- The repartitions between local tracking and Splunk Remote Searches tracking, which influence how much the load is shared between the local instance and the remote Splunk deployments, if any
- The number of active users, the more users are actively using TrackMe, the less search slots are available for the back-end purposes

Splunk typical requirements:

- TrackMe relies on Splunk and shares its typical requirements
- We recommend to honour the minimal specifications for Splunk: https://docs.splunk.com/Documentation/Splunk/latest/Installation/Systemrequirements

Minimal CPU requirements:

- We recommend a minimal setup of **16 vCPUs** for running TrackMe in a Production context.

- This highly depends on the number of trackers and use cases that are implemented, the nature of the use cases, as well as the number and activity of users in TrackMe.

Recommended requirements:

- **32 vCPUs** (or more) for running TrackMe in a Production context with a large scale deployment, and a large number of use cases.
- **32 GB** of memory is recommended for running TrackMe in a Production context.

Applications dependencies

Hint

TrackMe 2.2.0 and later:

- Since **TrackMe 2.2.0**, the Splunk_SA_CIM application is no longer a required dependency, it is now optional and only required for CIM tracking purposes.

Important

TrackMe 2.4.1 and later — no Machine Learning add-on dependency:

Starting with **TrackMe 2.4.1**, TrackMe **no longer requires** the Splunk AI Toolkit (formerly Splunk Machine Learning Toolkit) or the Python for Scientific Computing packages.

Every Machine Learning driven feature now runs on TrackMe's **built-in native density function engine** (`TrackMeNativeDensityFunction`), which uses scipy directly and stores its models in per-tenant KVstore collections:

- **ML Outliers detection** — native by default since TrackMe 2.3.15.
- **Adaptive delay thresholding** — migrated to the native engine in TrackMe 2.4.1. This was the last feature relying on the Splunk AI Toolkit's `fit DensityFunction` command; it now performs the same statistical (density) calculation inline through the native engine, with no model retained.

You can therefore deploy and run TrackMe without installing either add-on.

Optional applications:

The following applications are **optional** and are not required for any default TrackMe feature:

- Splunk AI Toolkit (formerly Splunk Machine Learning Toolkit): https://splunkbase.splunk.com/app/2890
- Python for Scientific Computing: https://splunkbase.splunk.com/app/2882

Note

These packages are only needed if you explicitly select the Splunk AI Toolkit `DensityFunction` algorithm for an ML Outliers model (see *Outlier detection*). The default native engine (`TrackMeNativeDensityFunction`) requires neither add-on.

CHAPTER

SEVEN

INSTALLATION:

7.1 Installation of TrackMe

7.1.1 Installation Target

TrackMe is a Search Head component only; therefore, it must be deployed only on a target Search Head layer:

Splunk roles	Required
Search head	yes (*)
Monitoring console	yes (*)
Indexer tiers	no
Heavy Forwarders	no

- You can deploy TrackMe on a dedicated Search Head or Search Head Cluster, or alternatively on the machine hosting the Splunk Monitoring Console.
- If you choose to install TrackMe on your Search Head layer in a Search Head Cluster (SHC), TrackMe must be deployed through the SHC deployer node.

7.1.2 Indexes Definition

Default Indexes

TrackMe requires at least 4 indexes to be defined. In a distributed context, this usually means defining indexes in your manager node:

Default index name	Purpose	Minimal recommended retention
trackme_summary	TrackMe entities activity such as the state, flipping events, and Smart Status	3 months
trackme_audit	TrackMe modifications and application audit activity	1 year or more
trackme_notable	Notable events are generated by the TrackMe notable alert action	3 months
trackme_metrics	Various metrics are generated depending on the components	1 year or more (notably to allow long term outliers detection and metrics analytics)

Indexes are defined by default in the application package:

```
trackme/default/indexes.conf
```

Typically:

- If TrackMe is running on an "all-in-one" instance, you do not need to define the indexes as the application defines these already (mostly for testing and development purposes).
- It is good practice to use Splunk volumes rather than the default SPLUNK_DB variable to define the location of the buckets (which is required when using SmartStore).

Therefore, a typical definition would be:

```
[trackme_notable]
coldPath = volume:primary/trackme_notable/colddb
homePath = volume:primary/trackme_notable/db
thawedPath = $SPLUNK_DB/trackme_notable/thaweddb

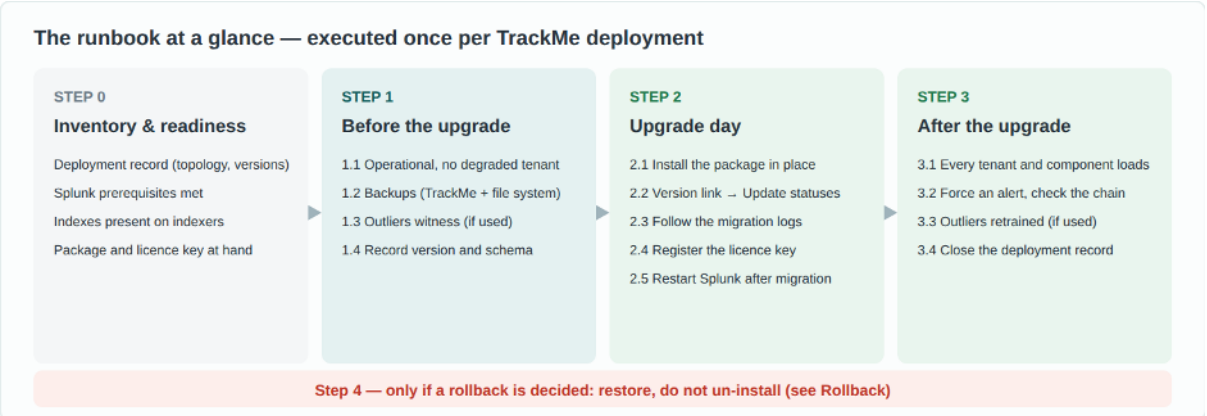
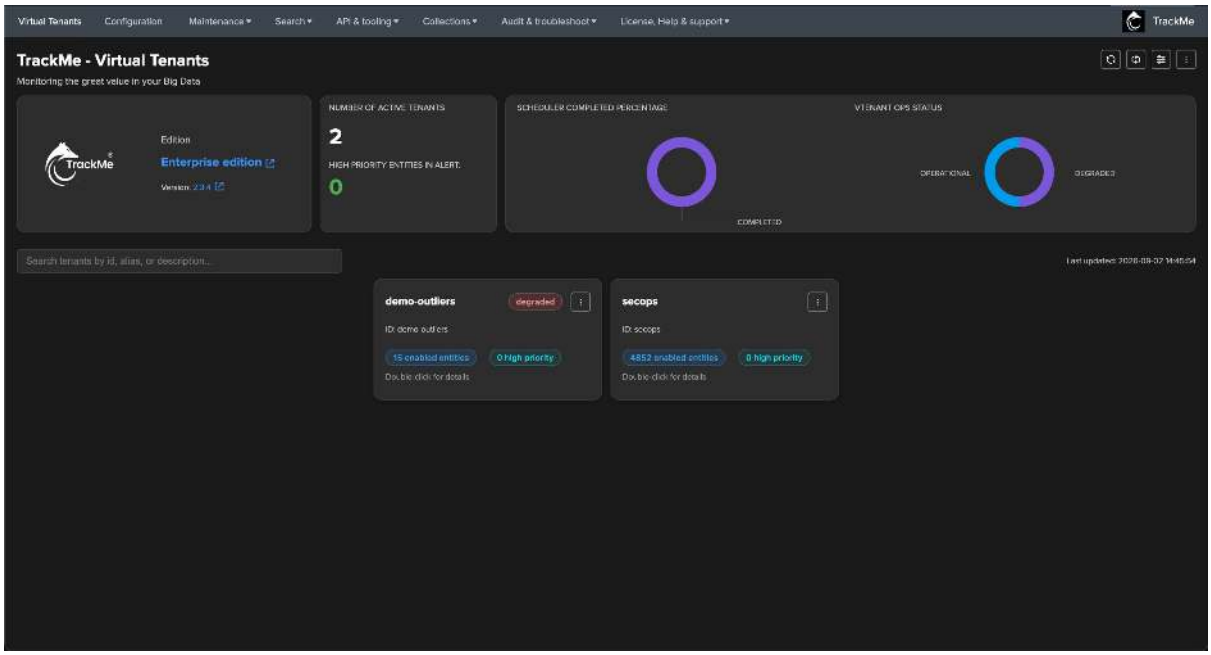
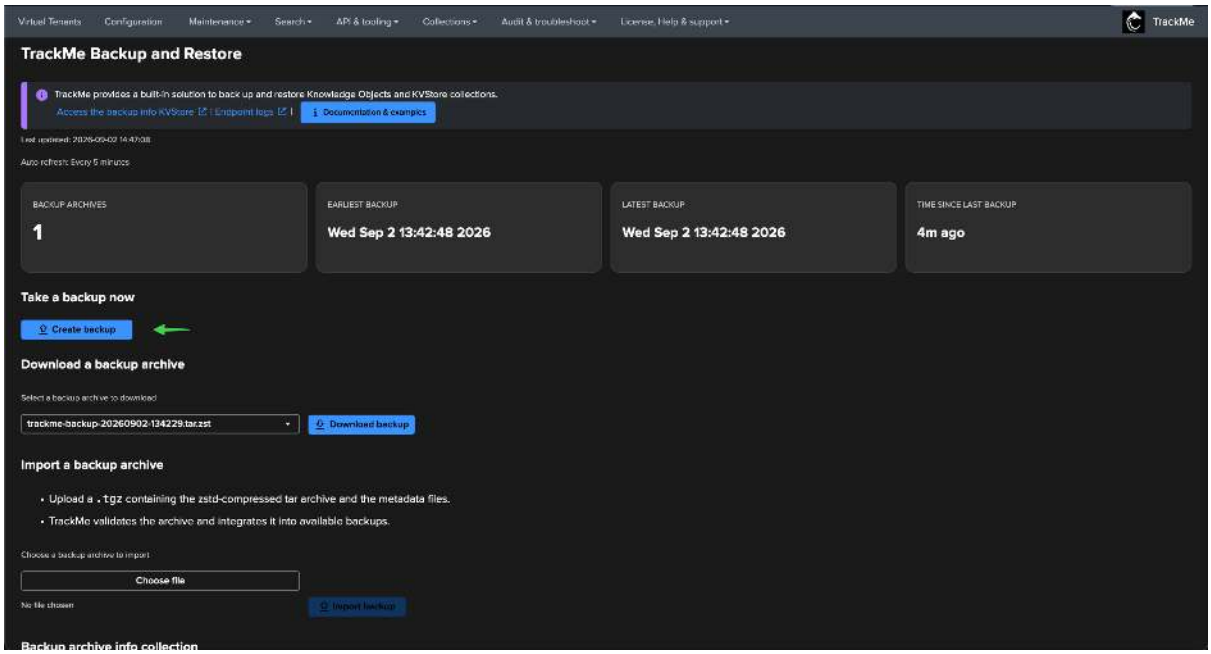
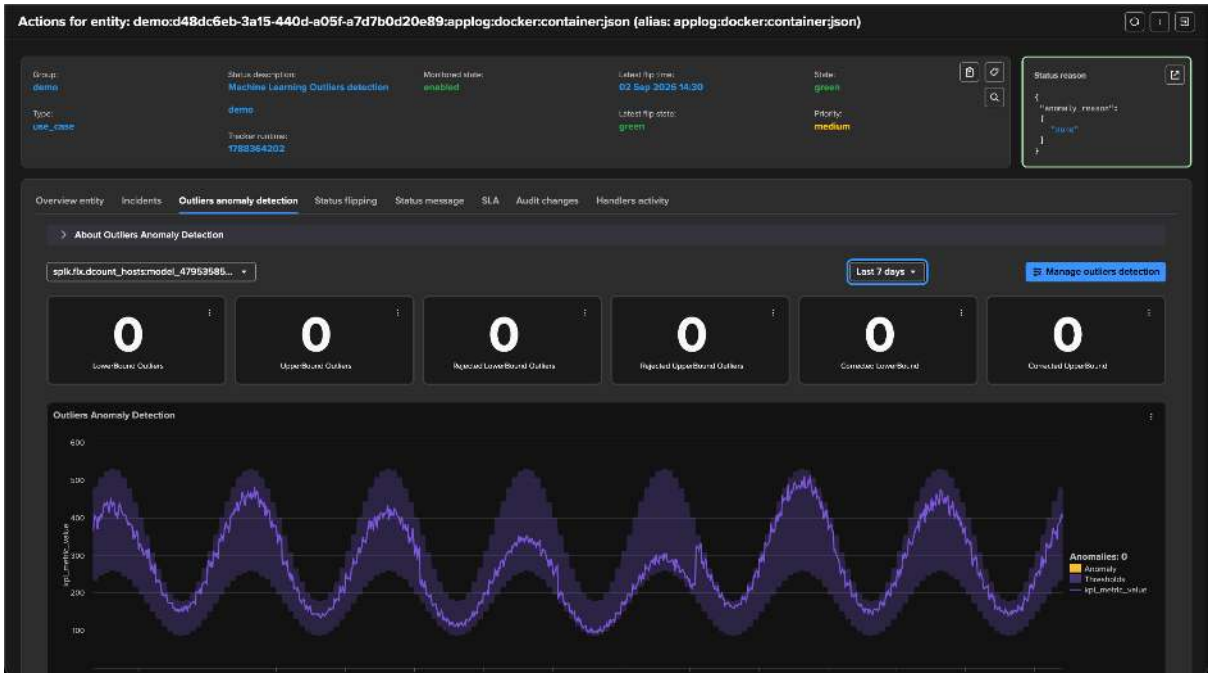
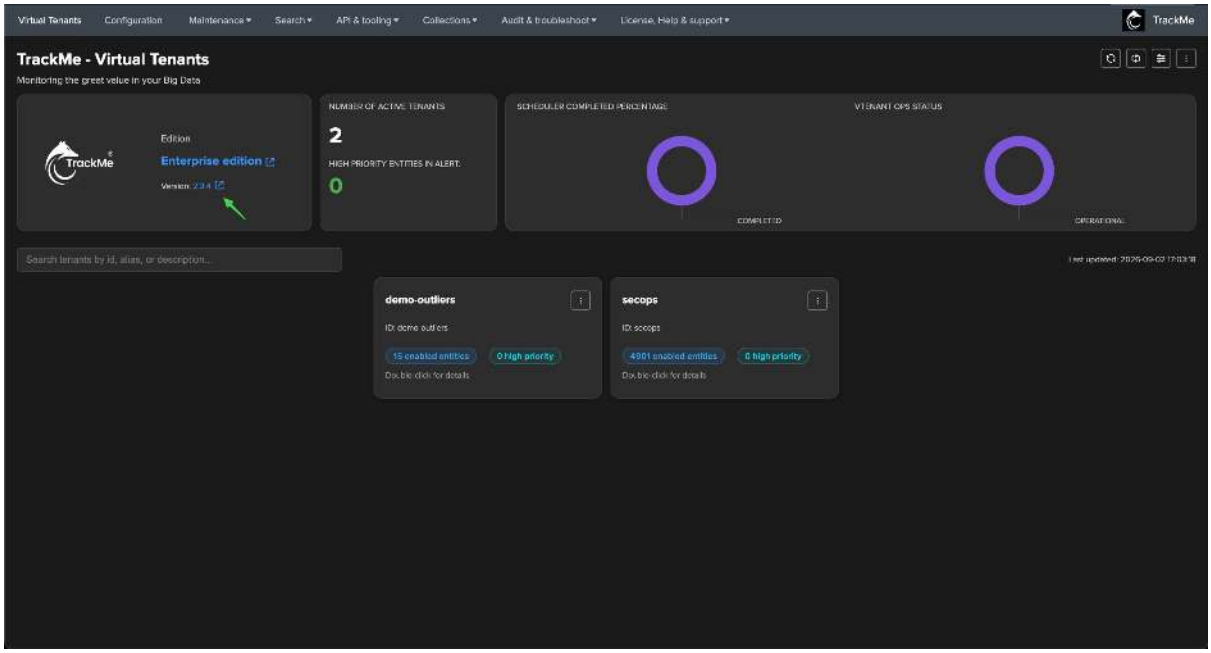
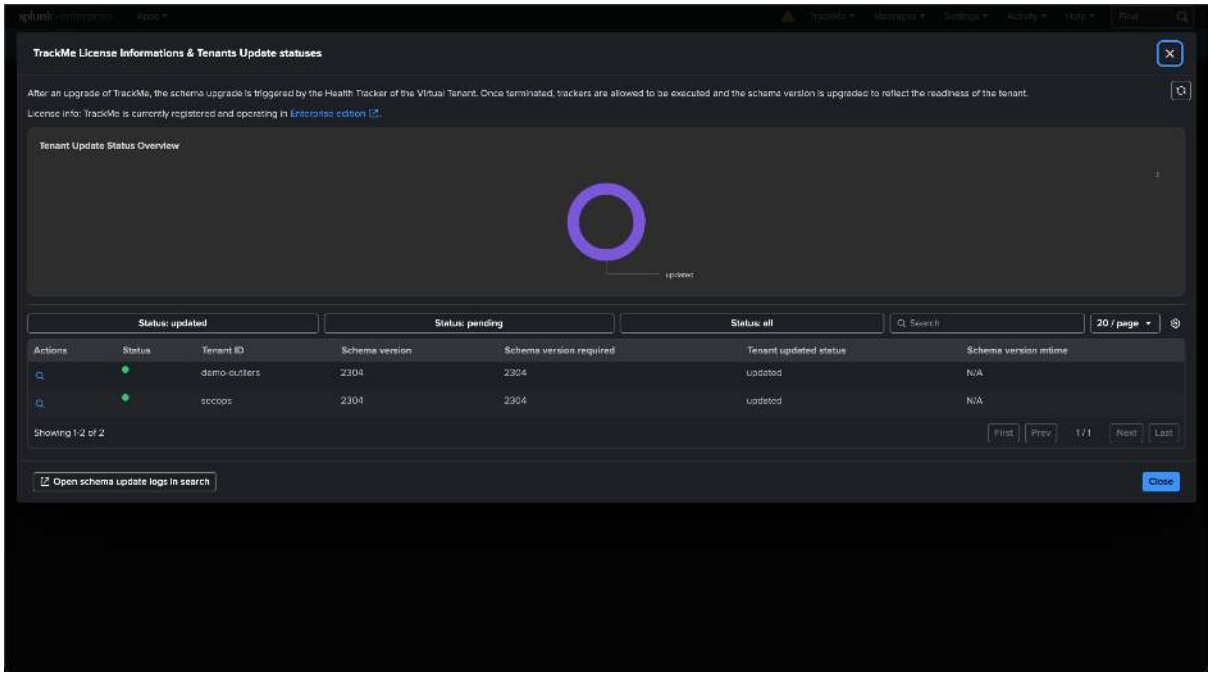
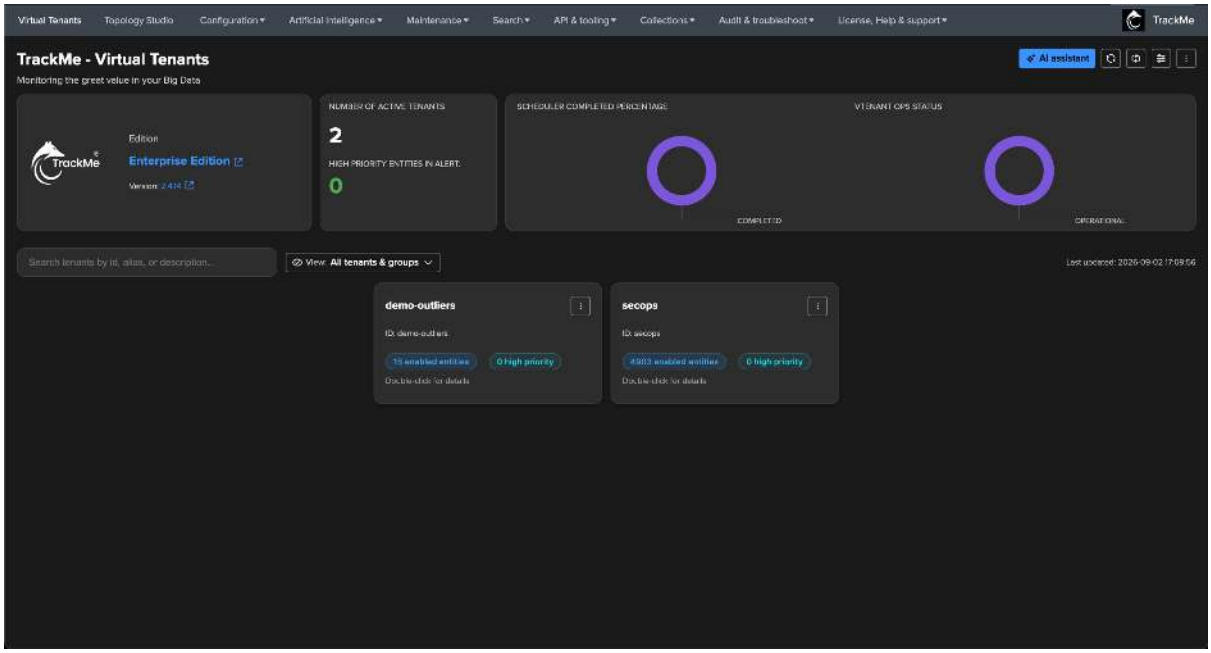
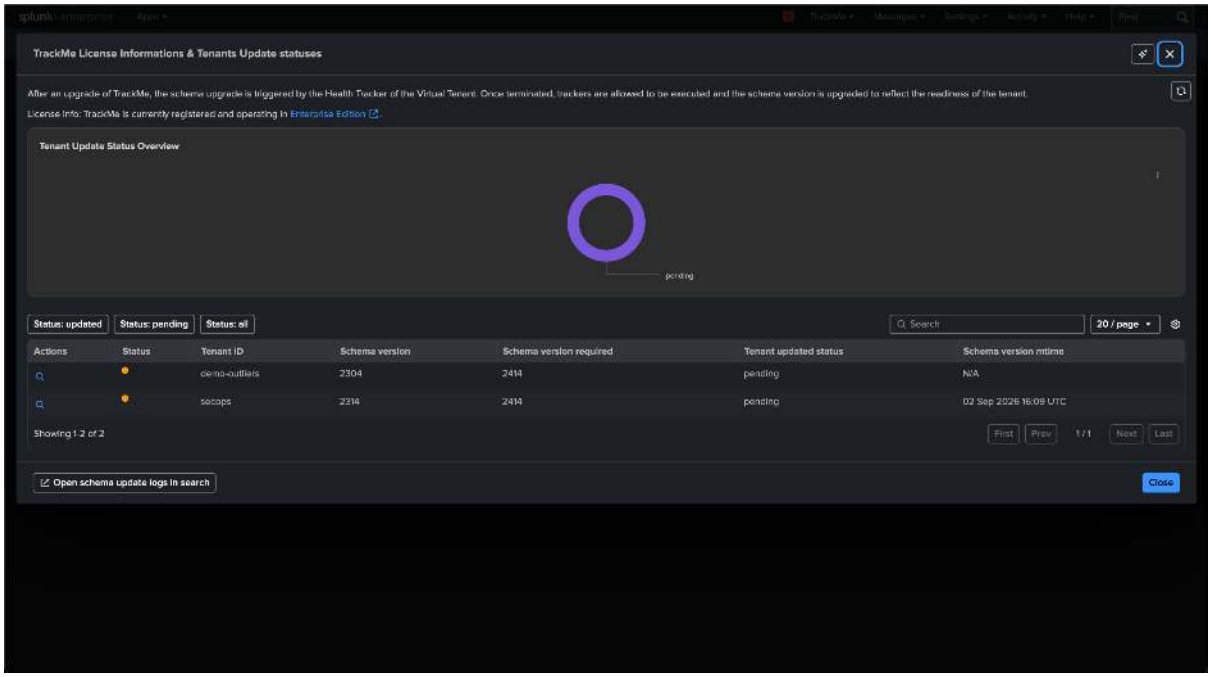
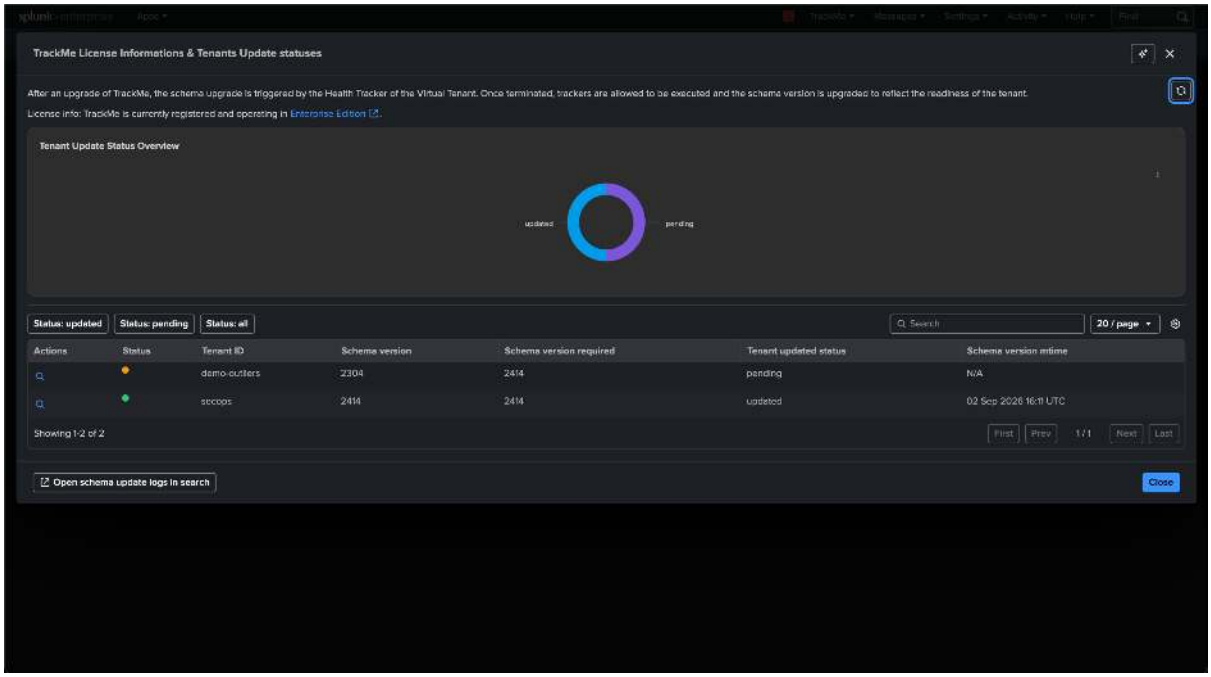
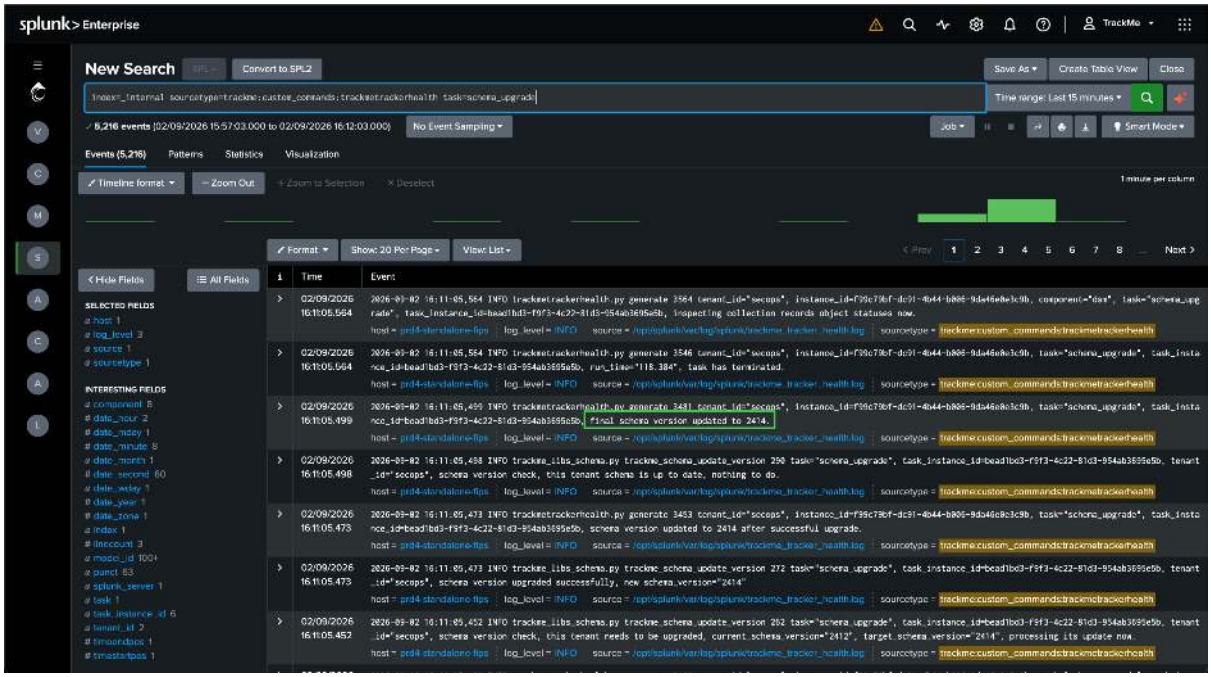
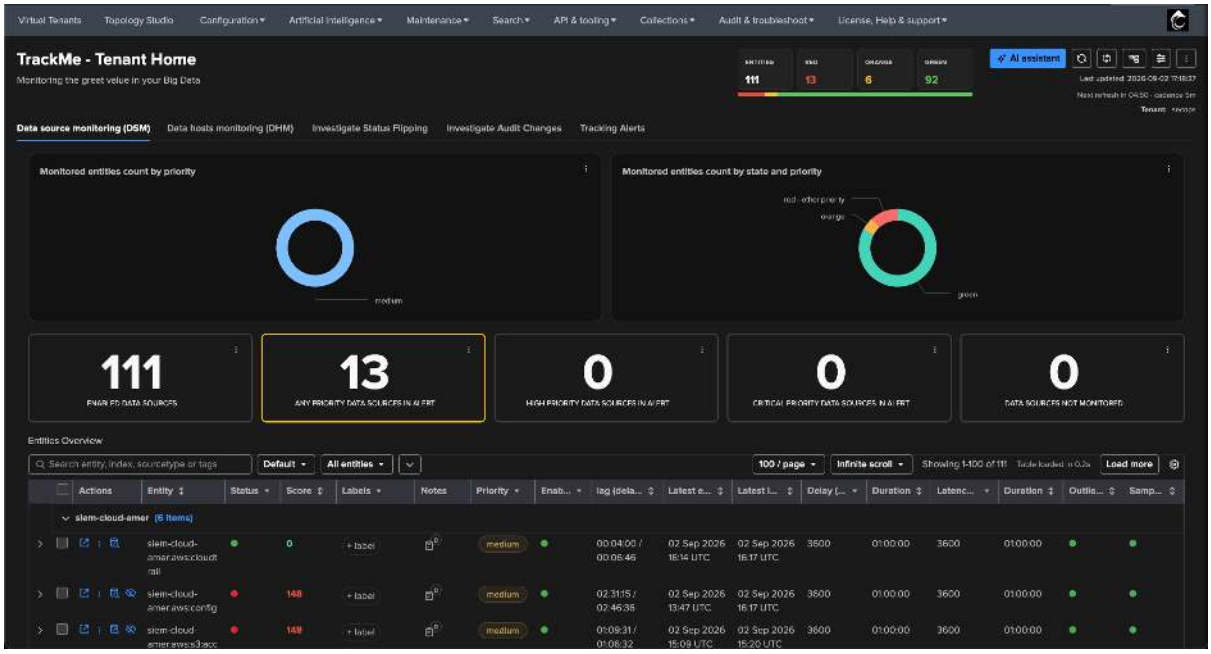
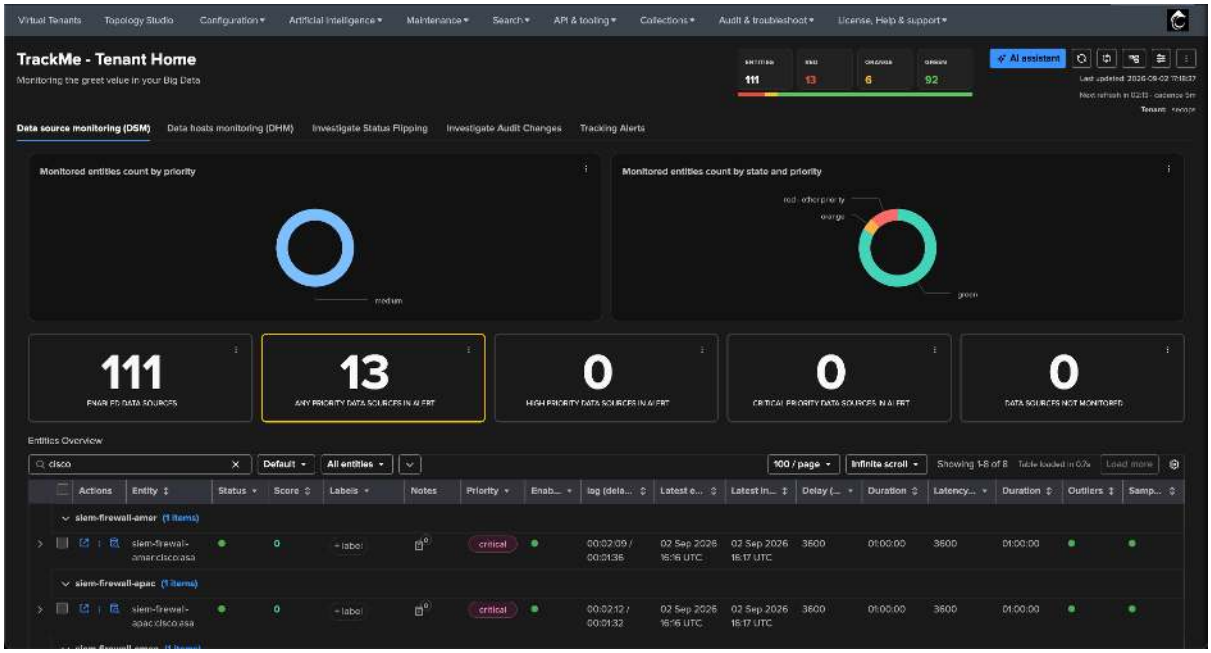
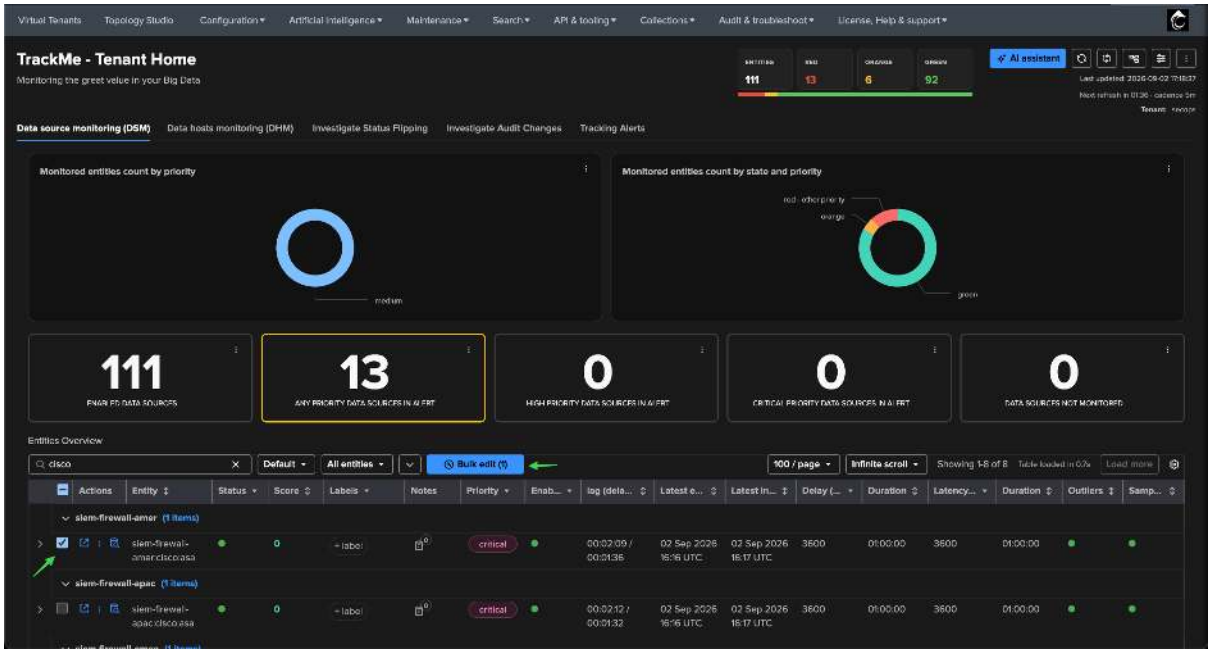
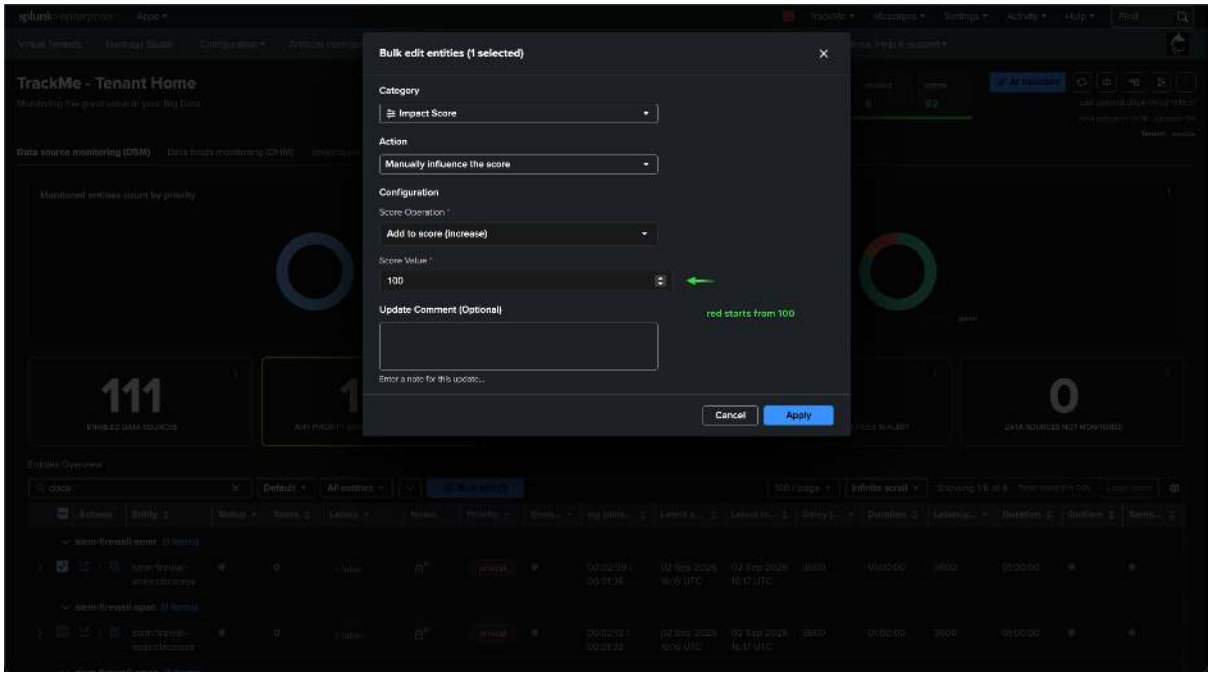
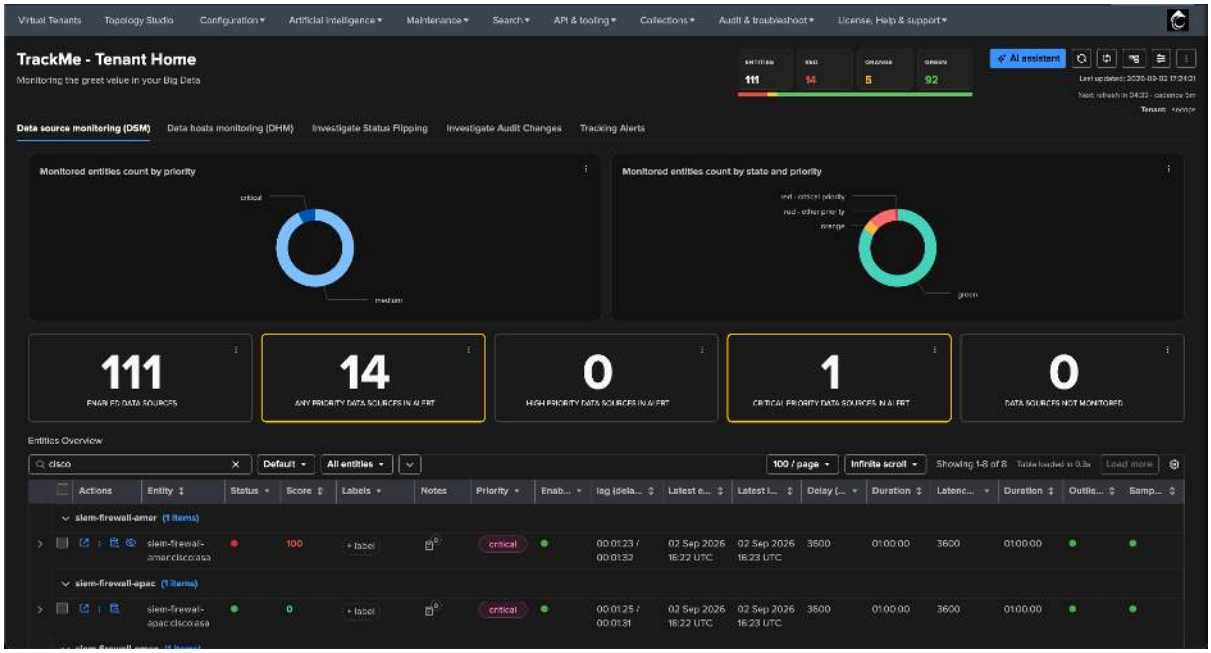
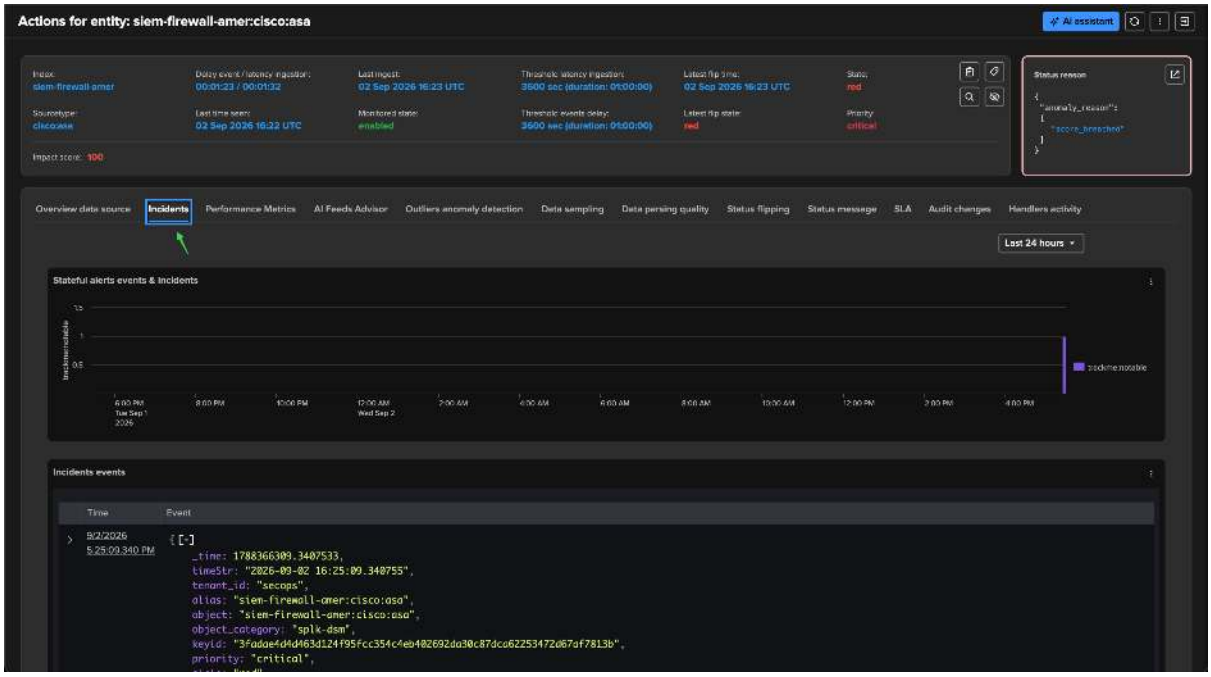
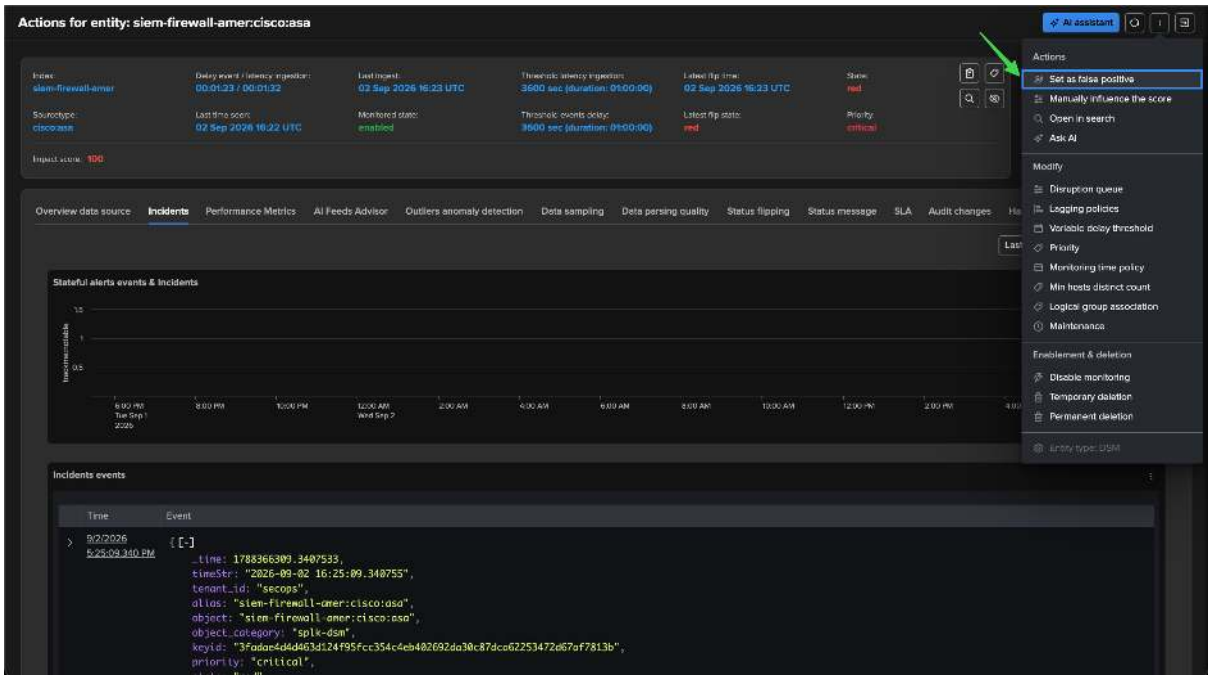
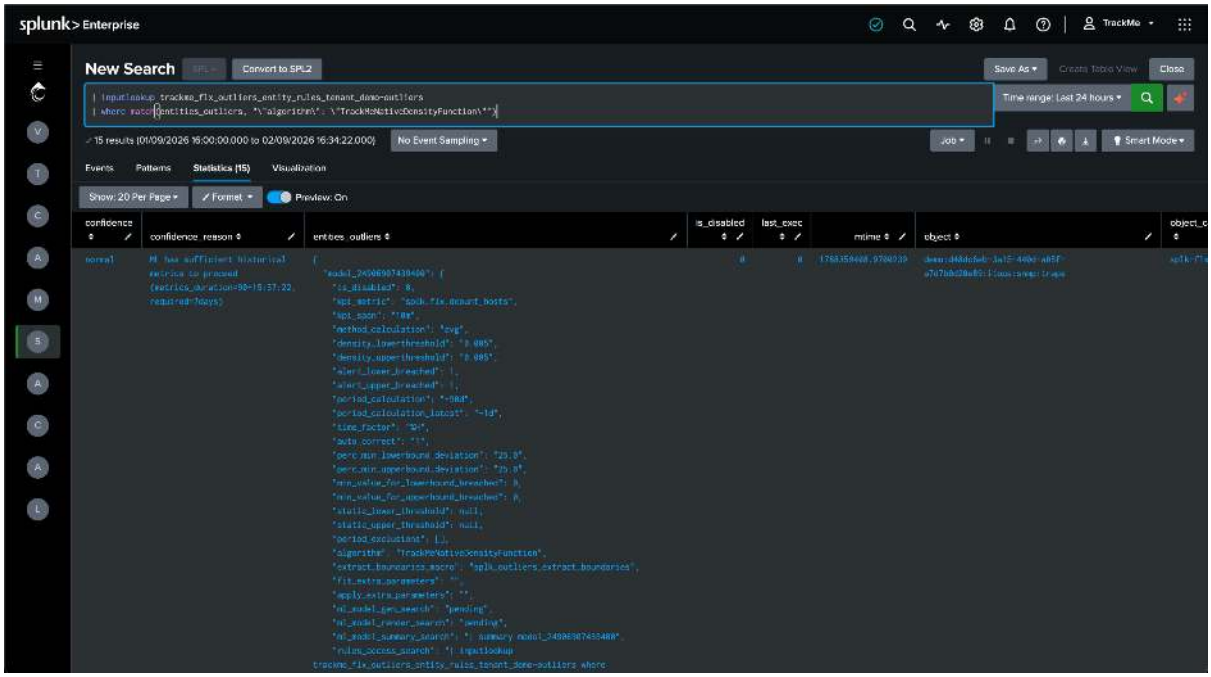
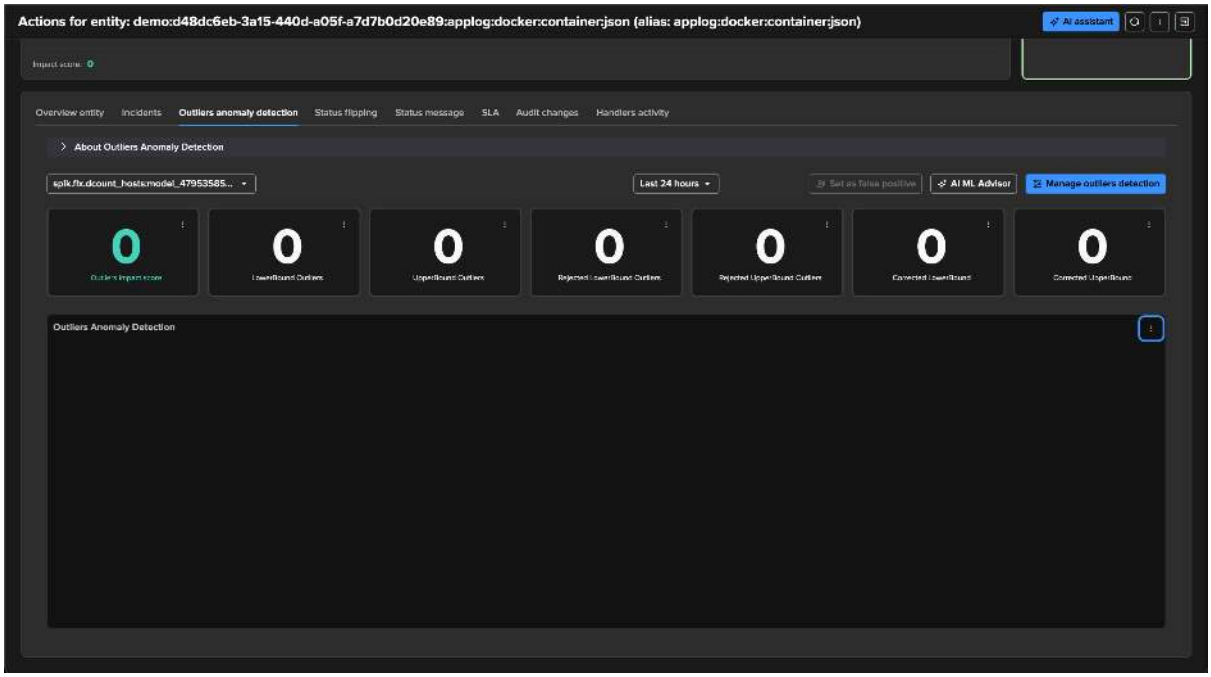
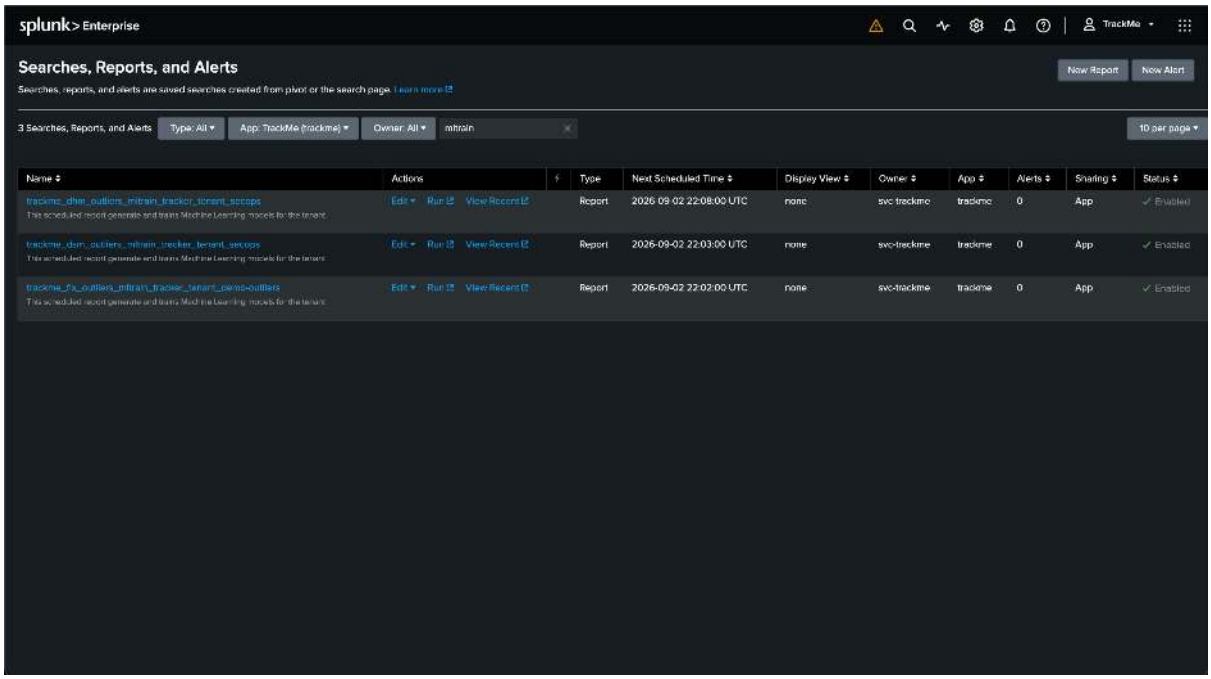
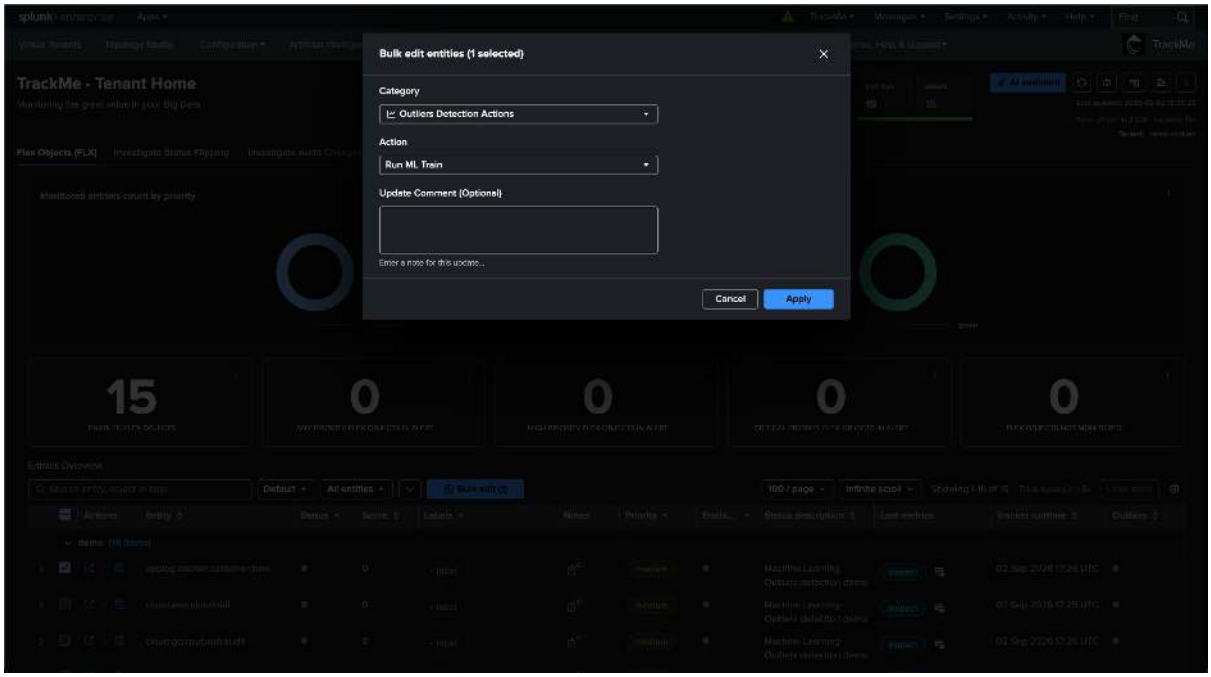
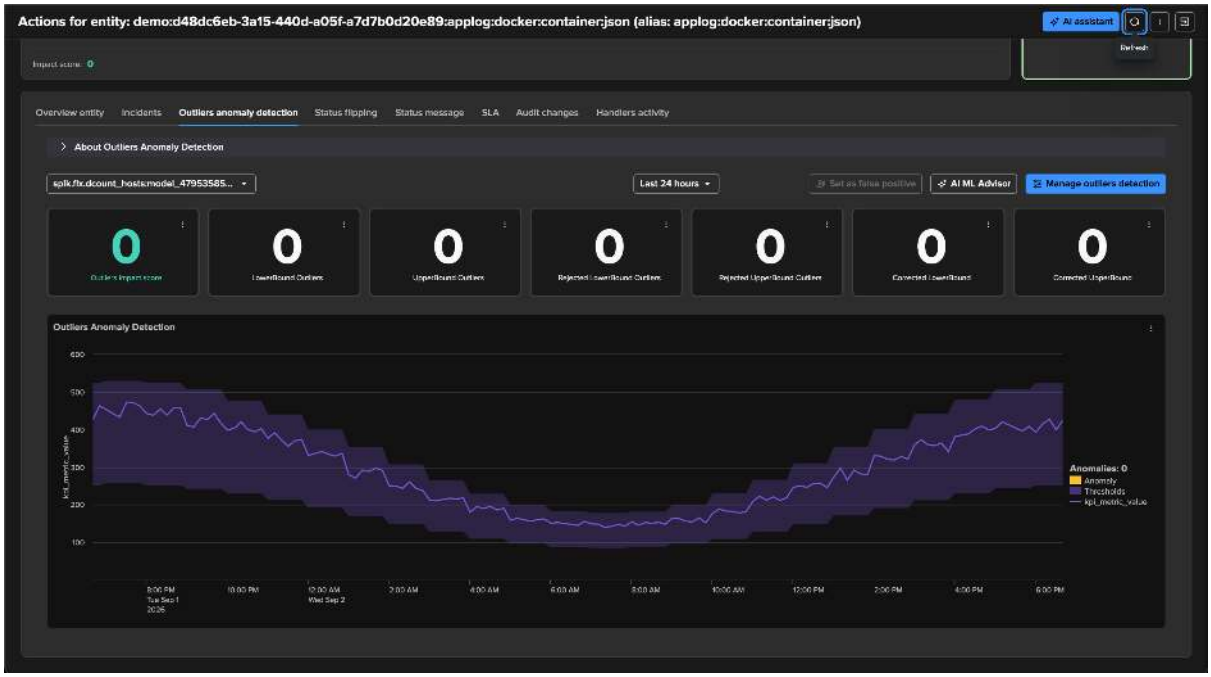
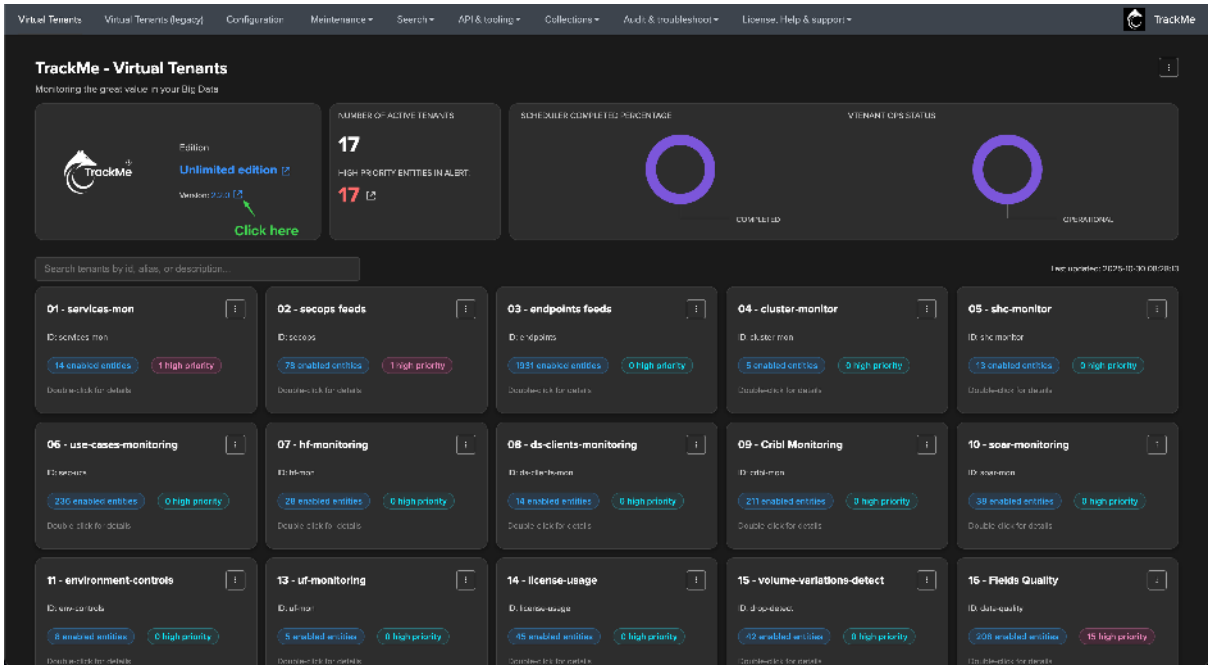
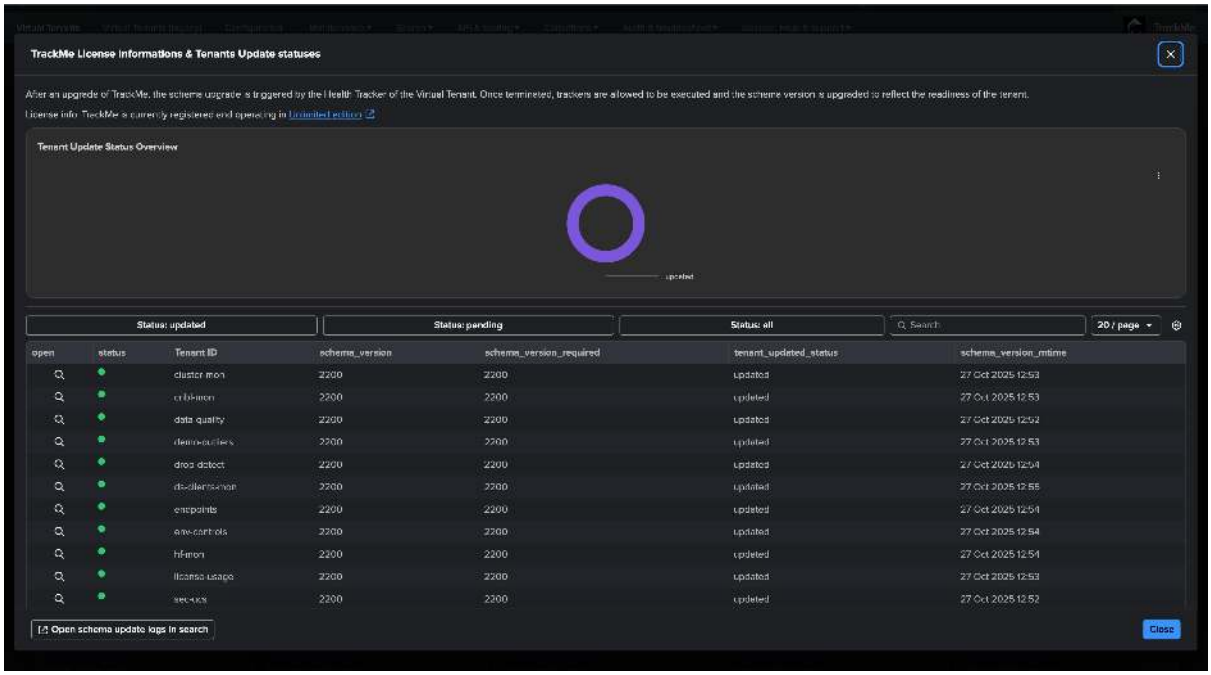
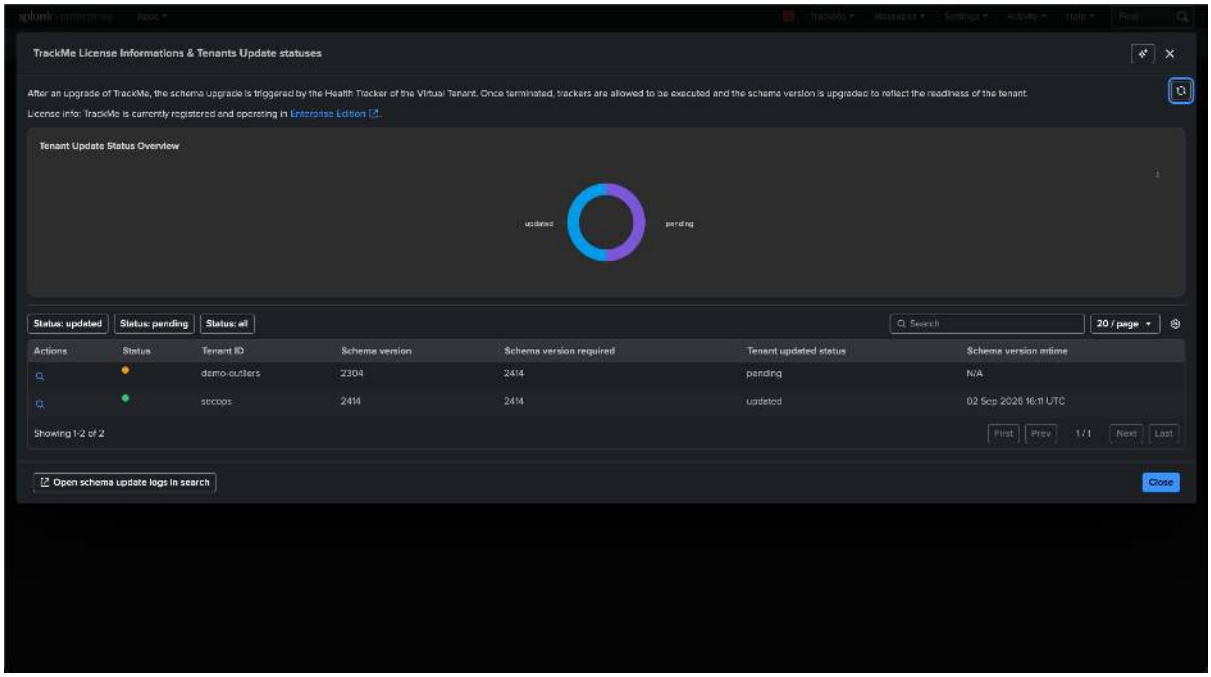
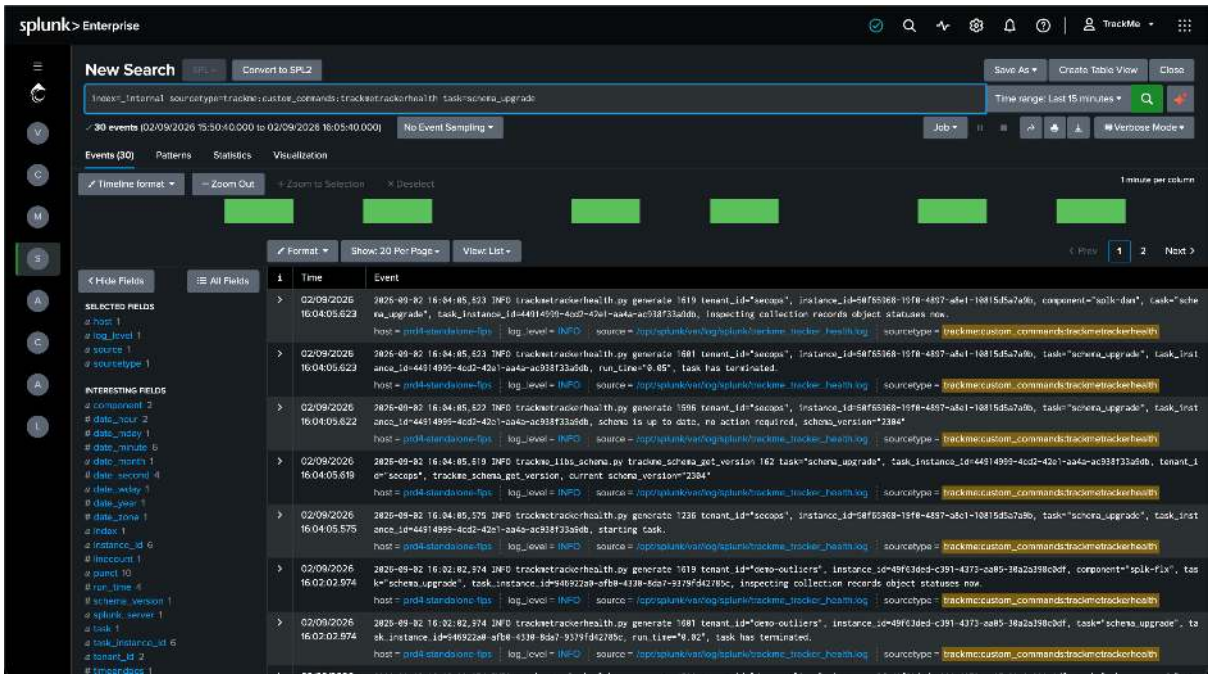
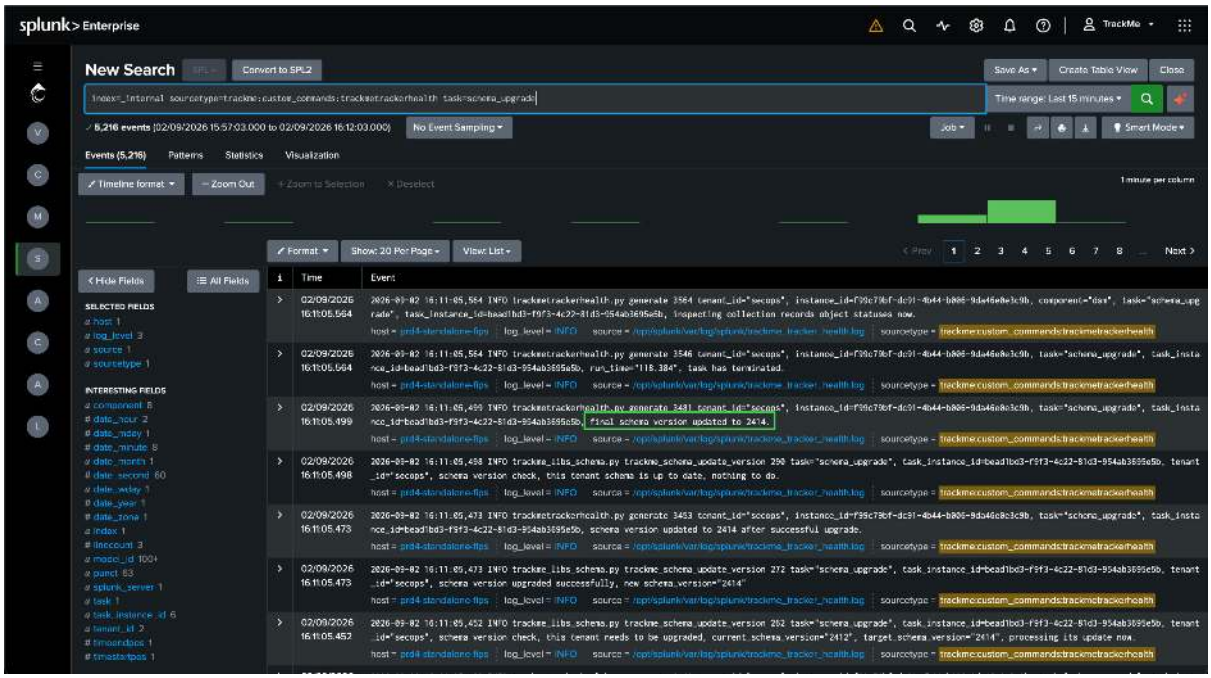
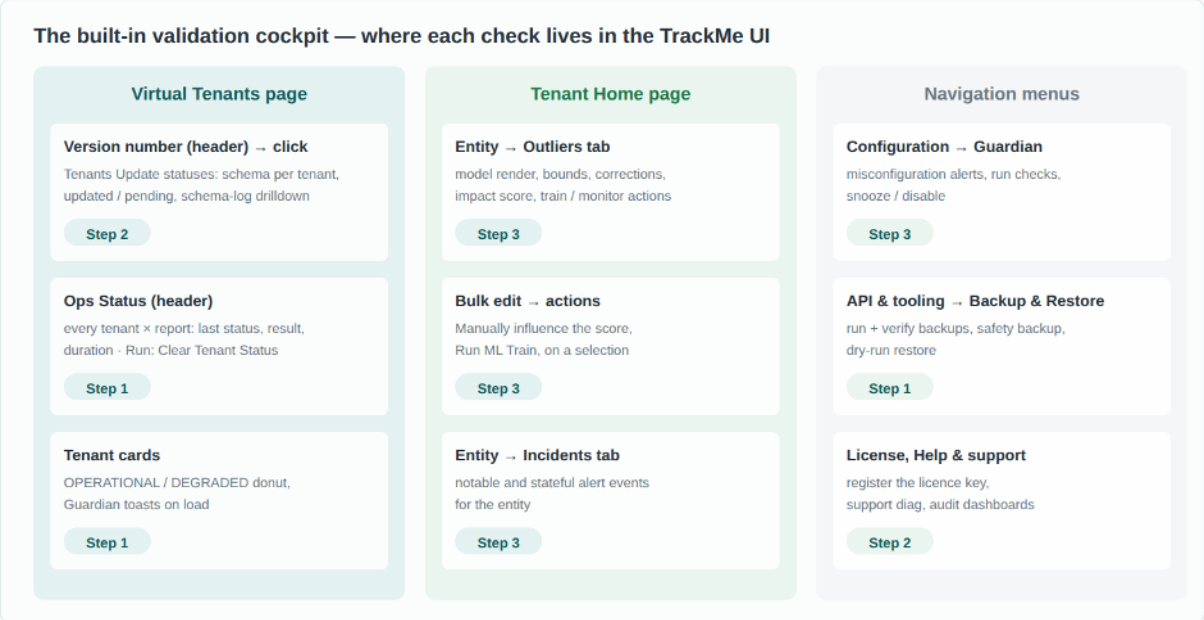
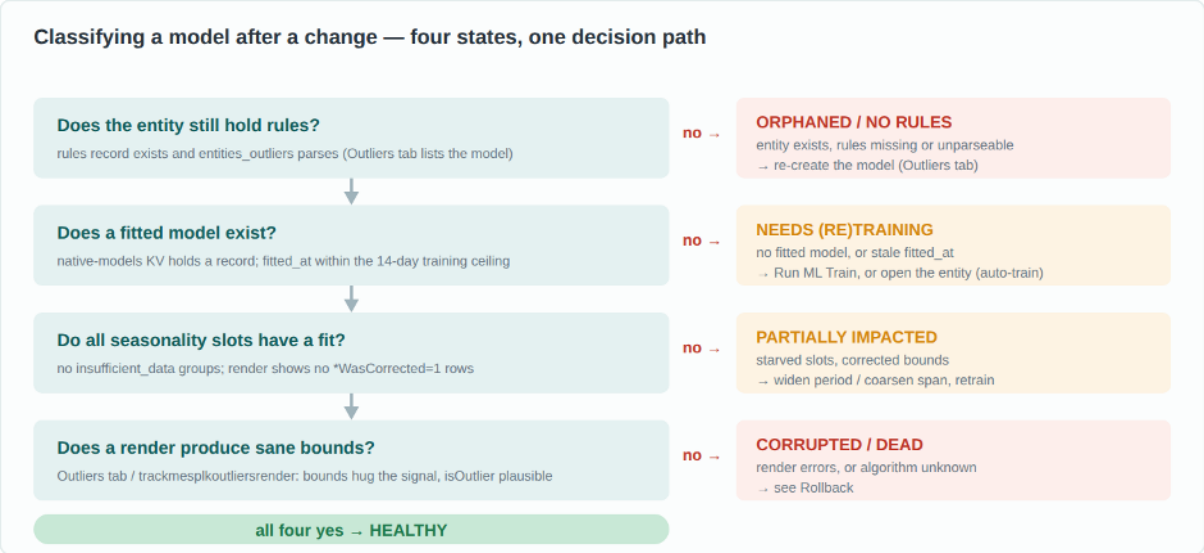
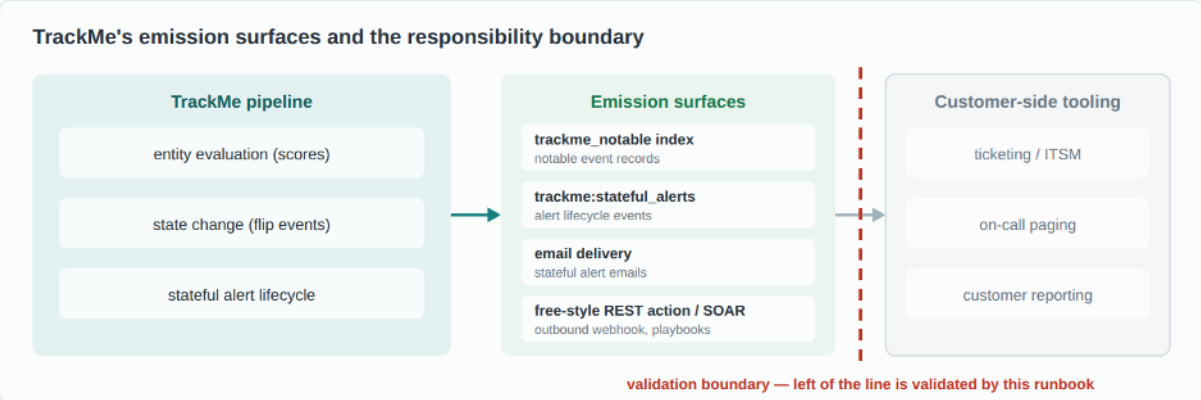
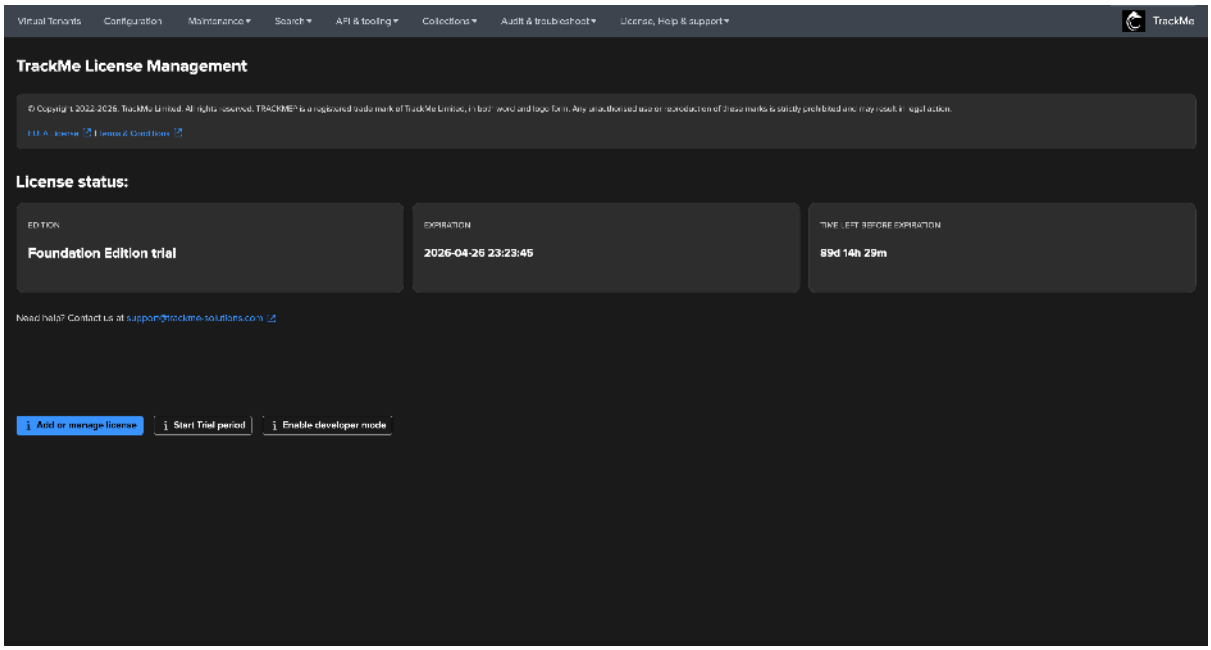
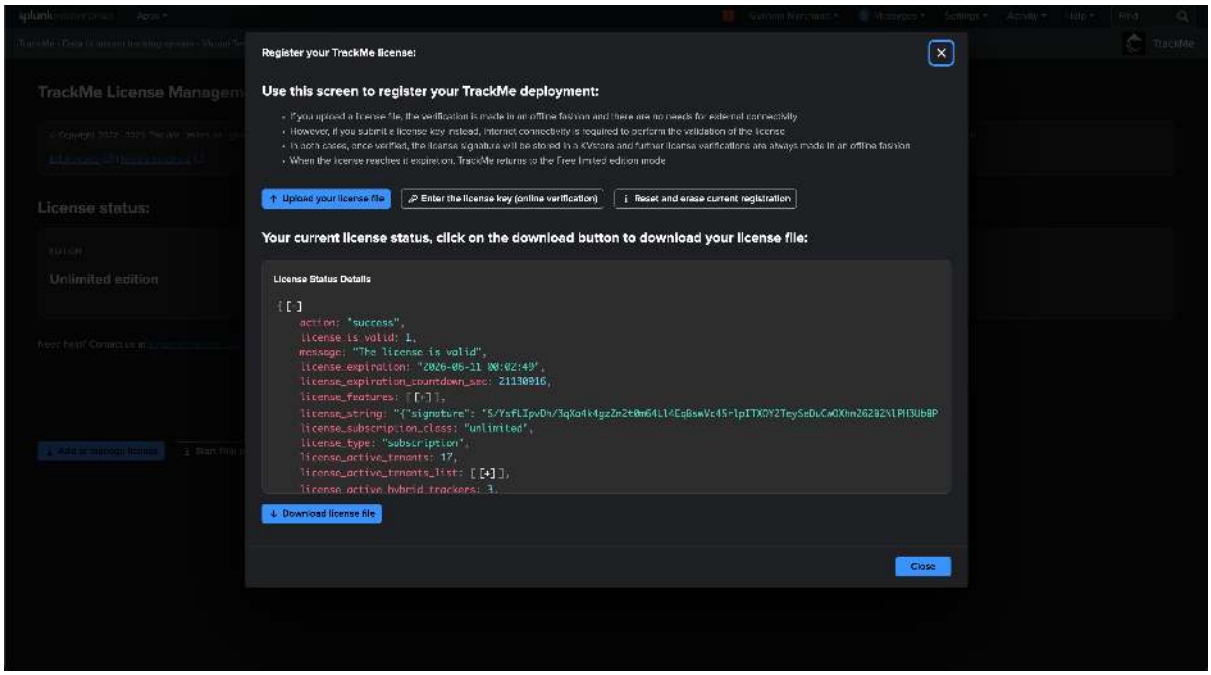
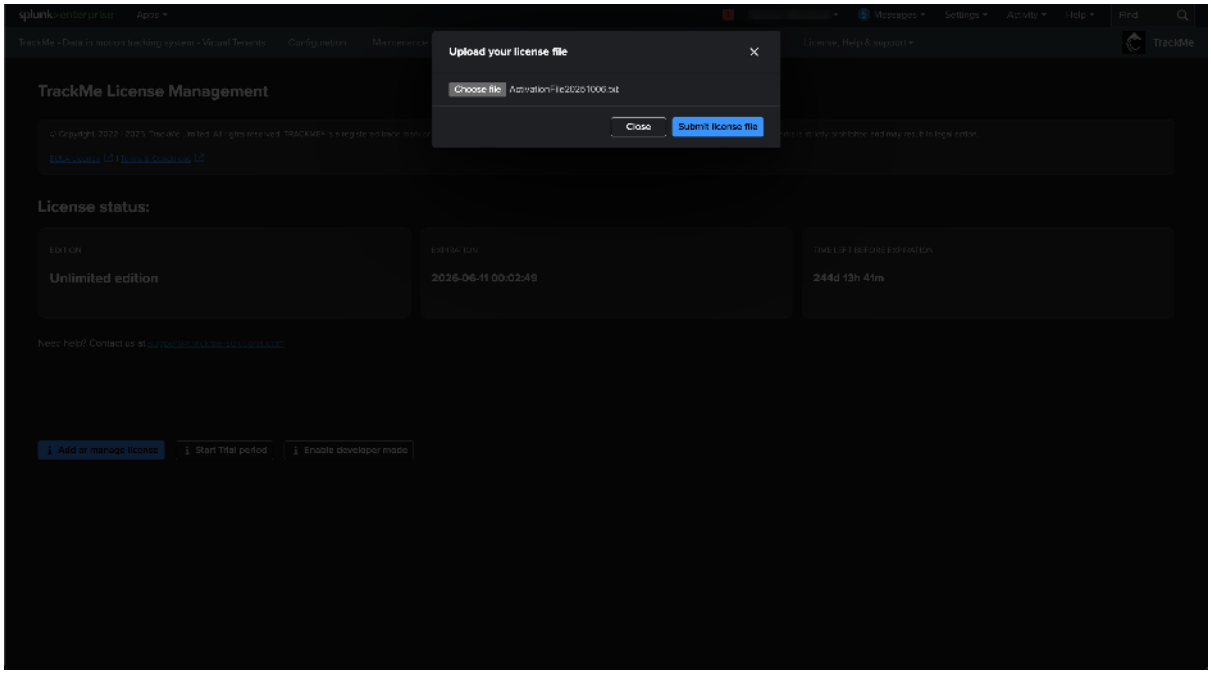
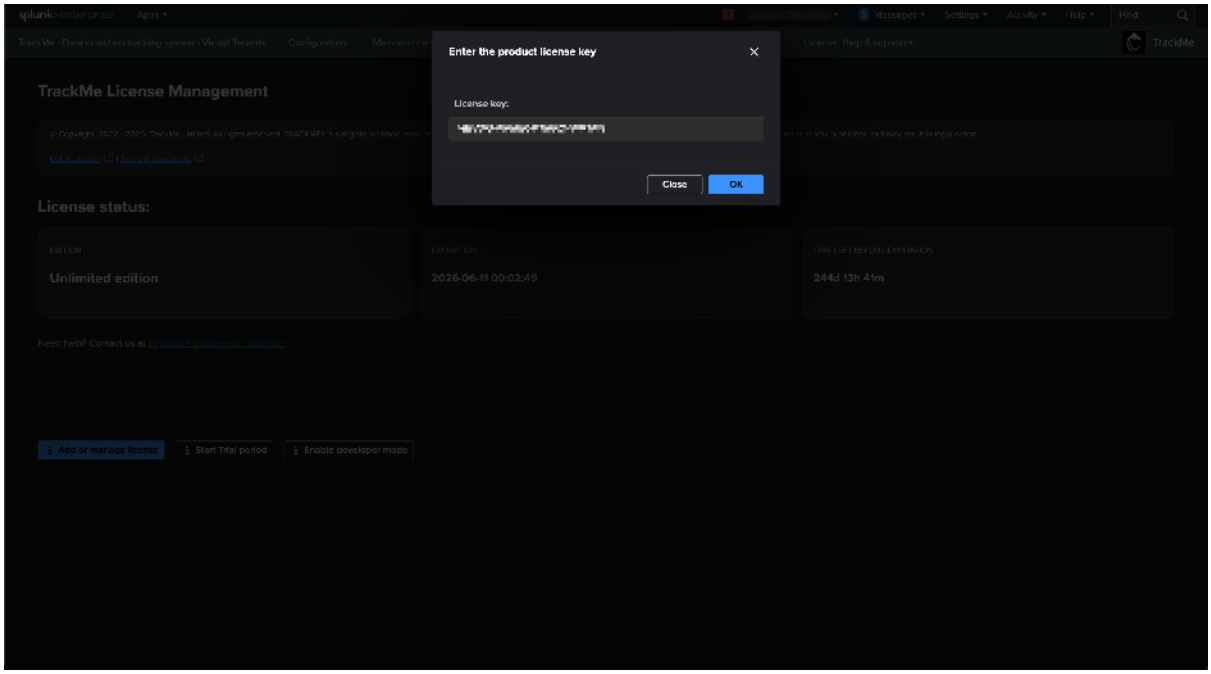
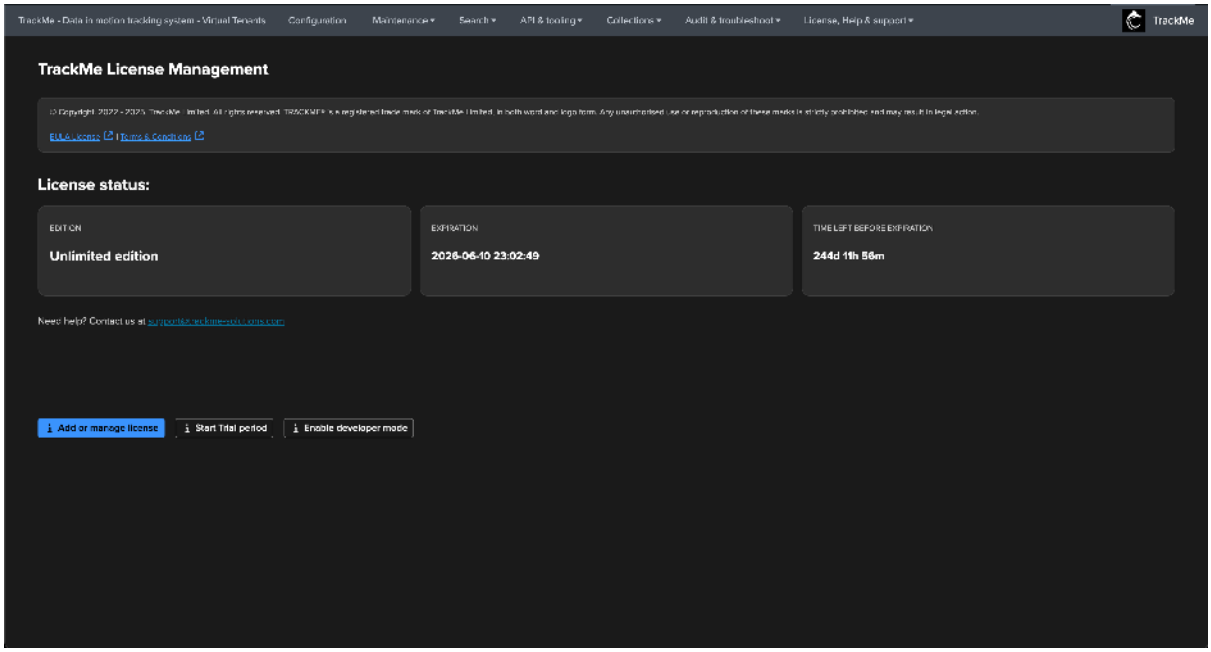
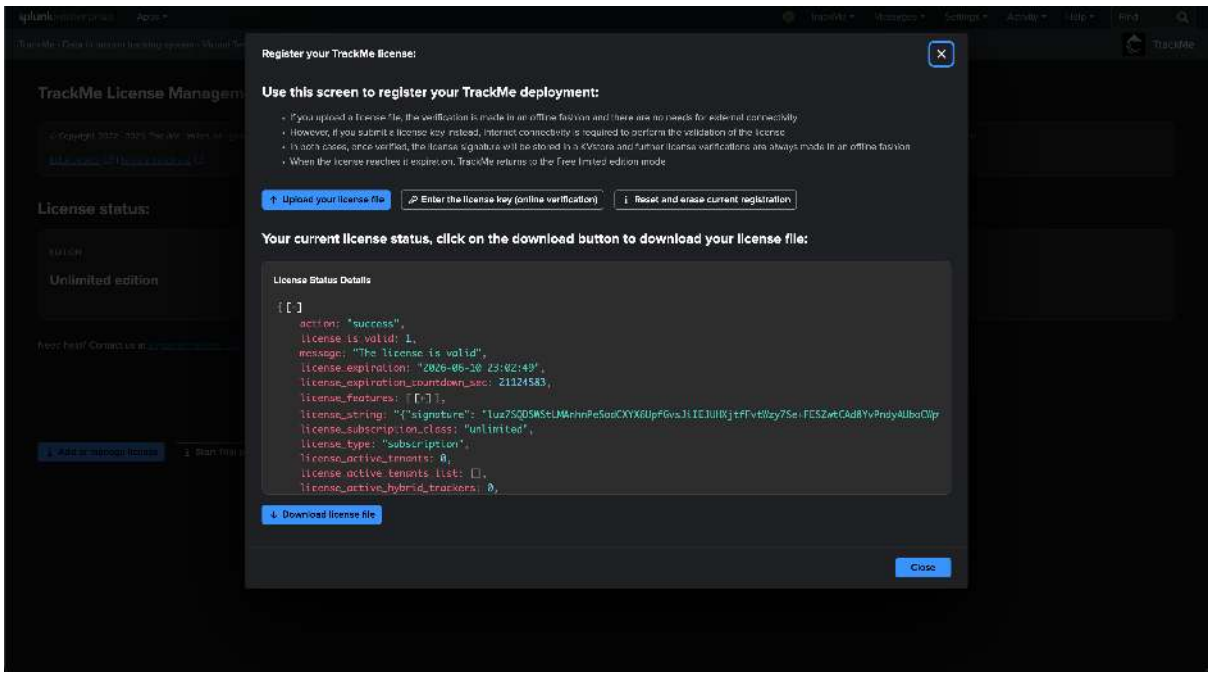
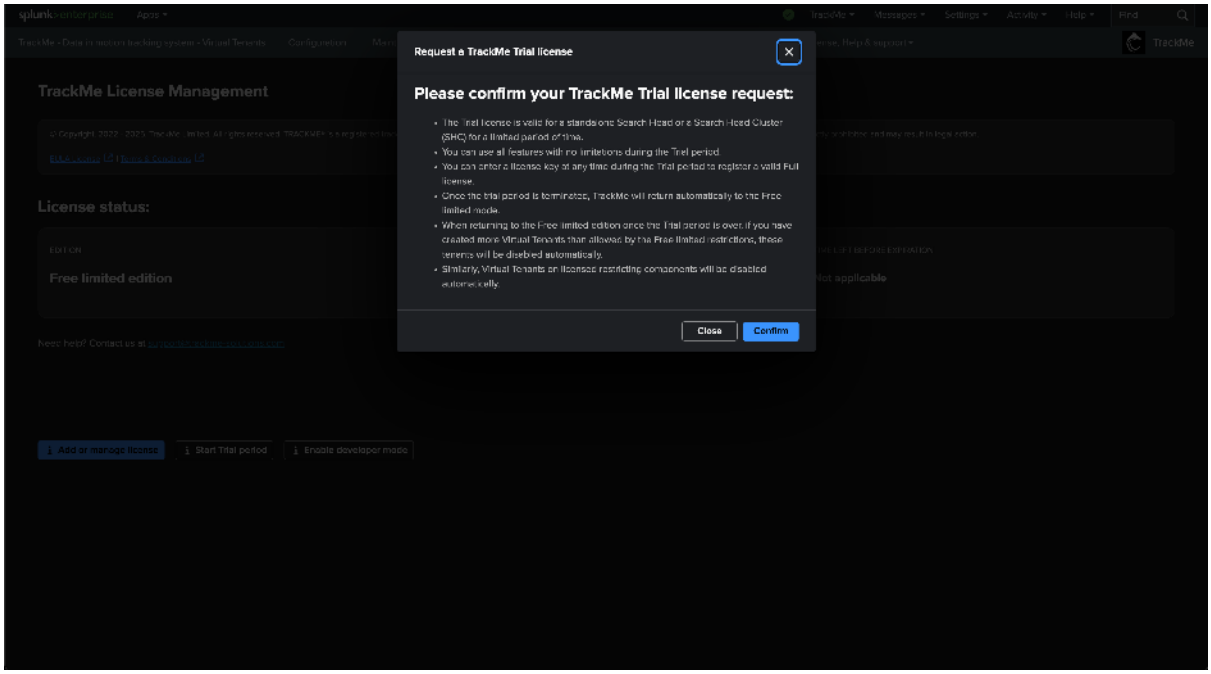
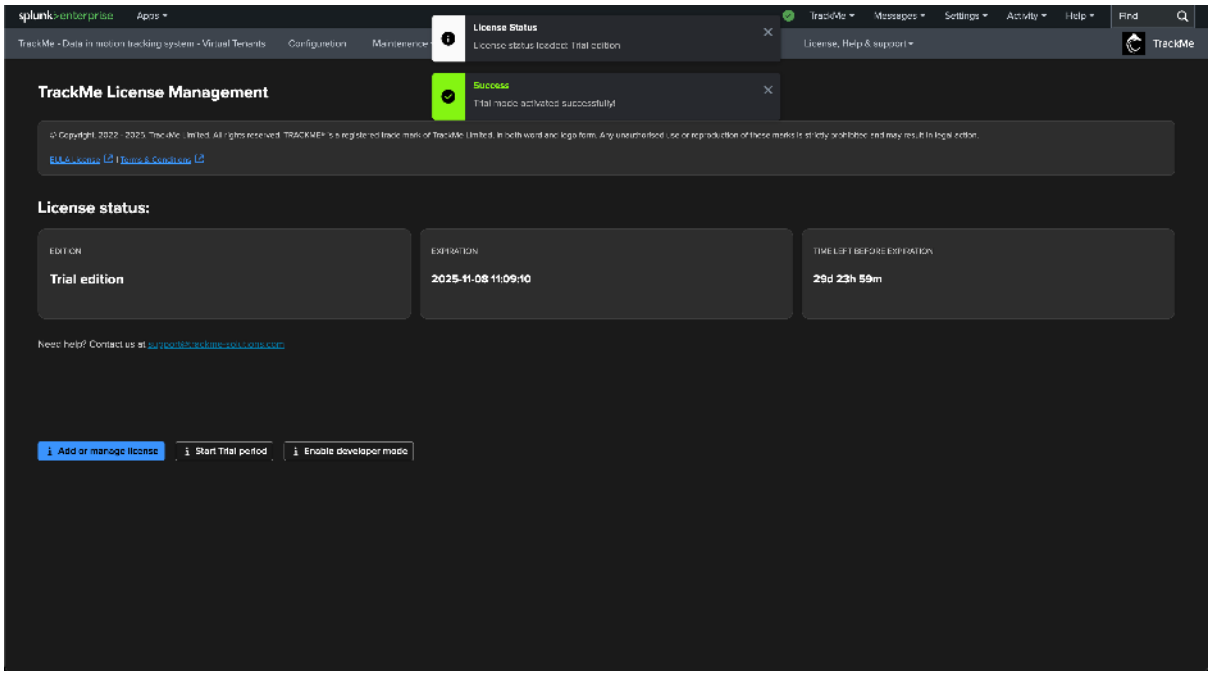
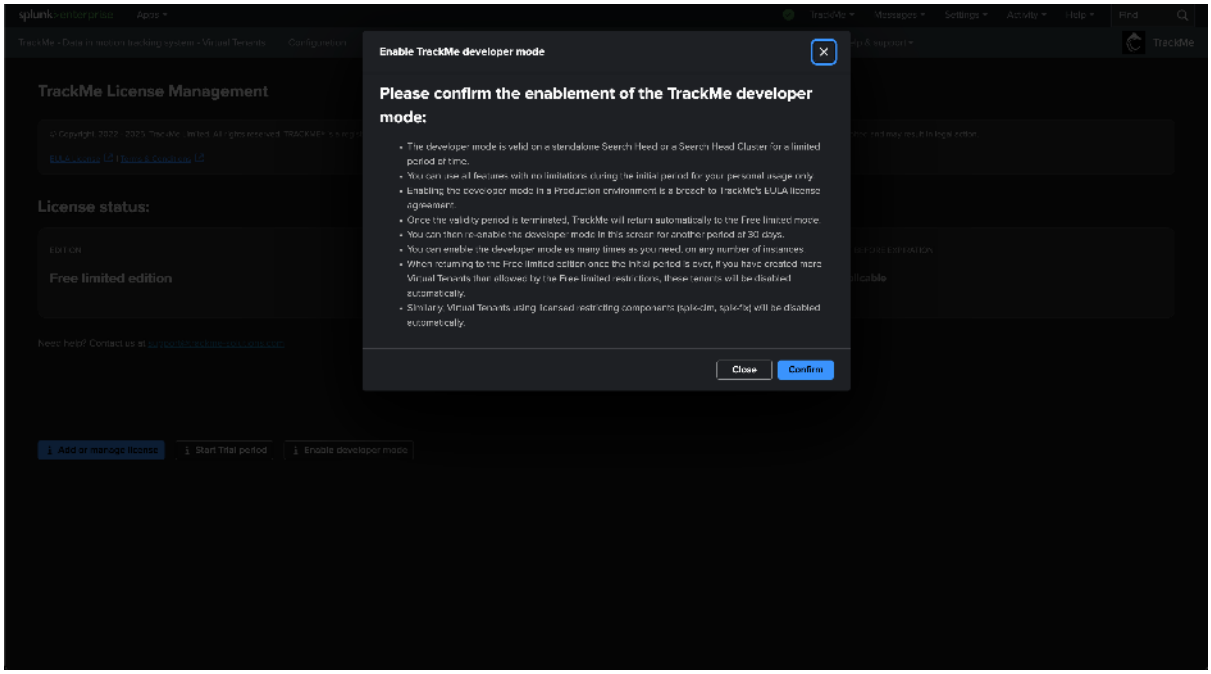
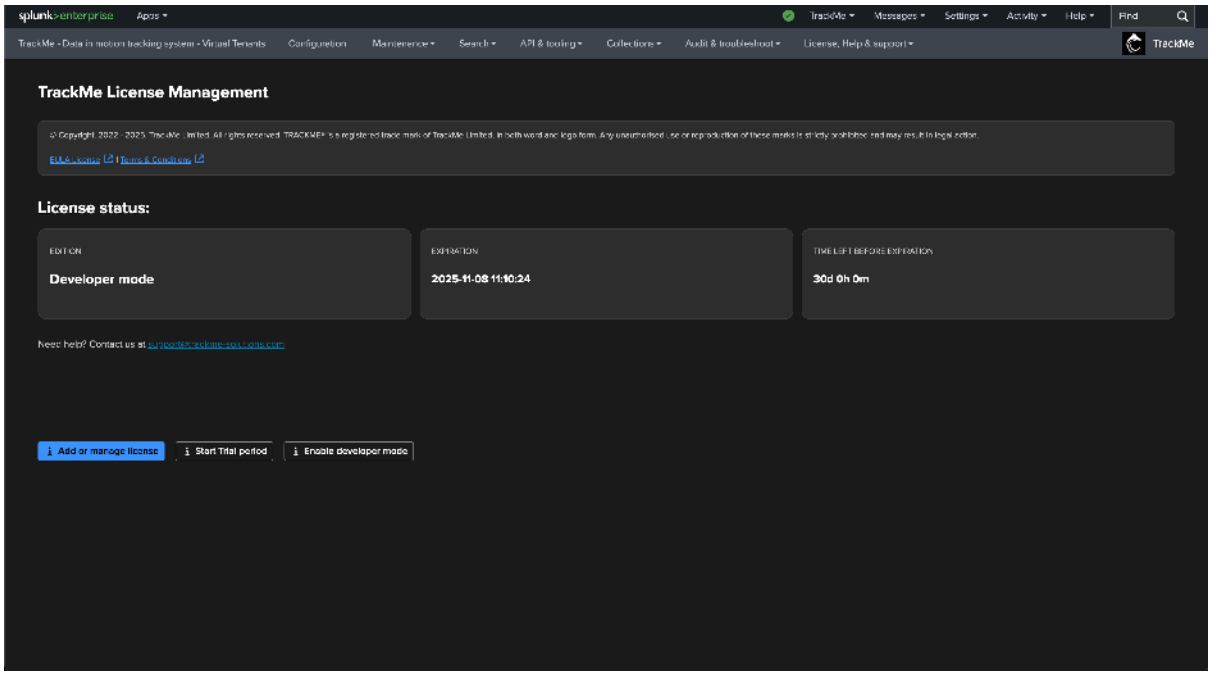
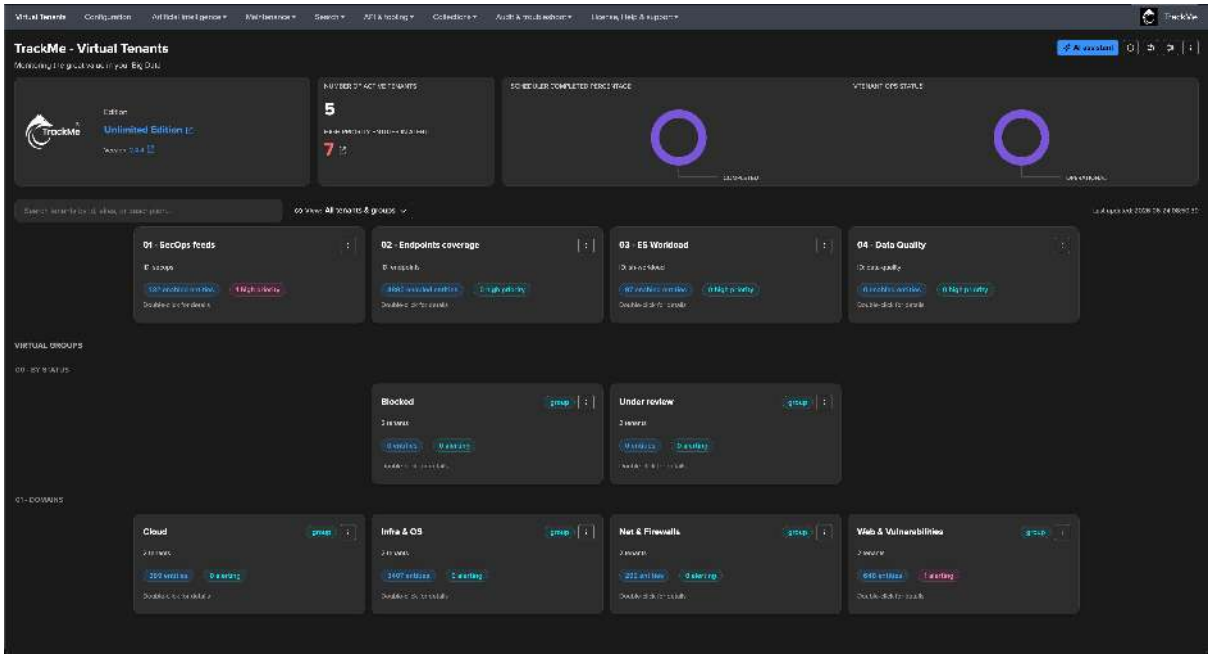
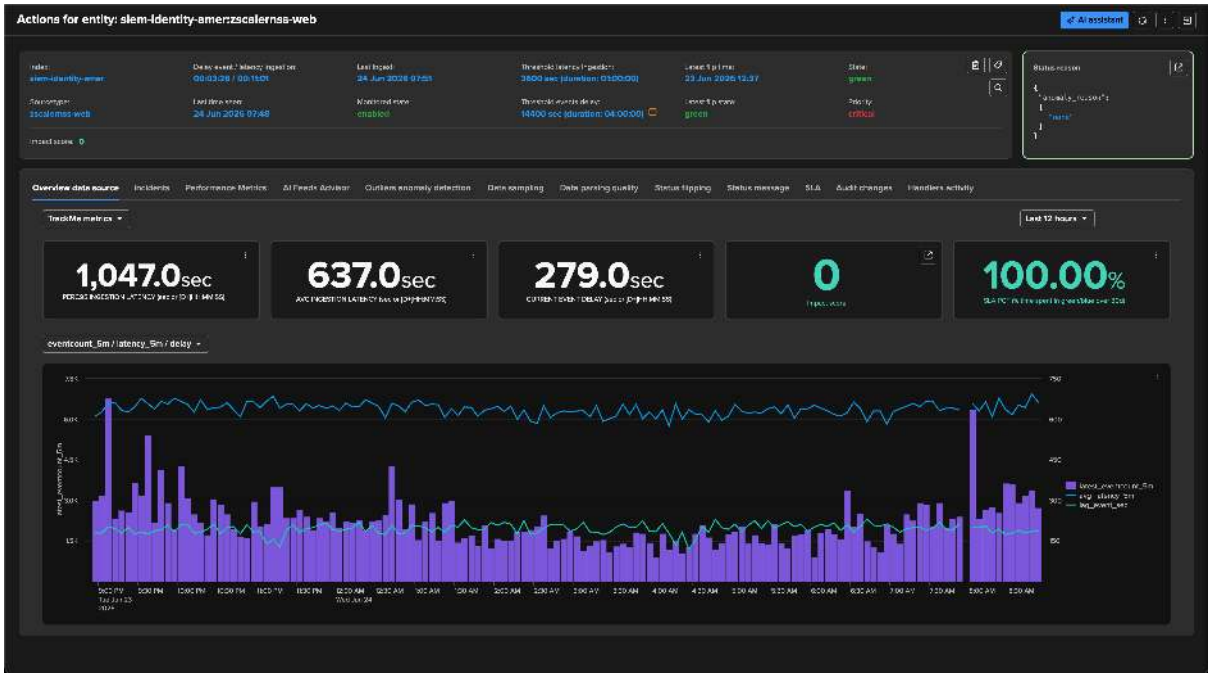
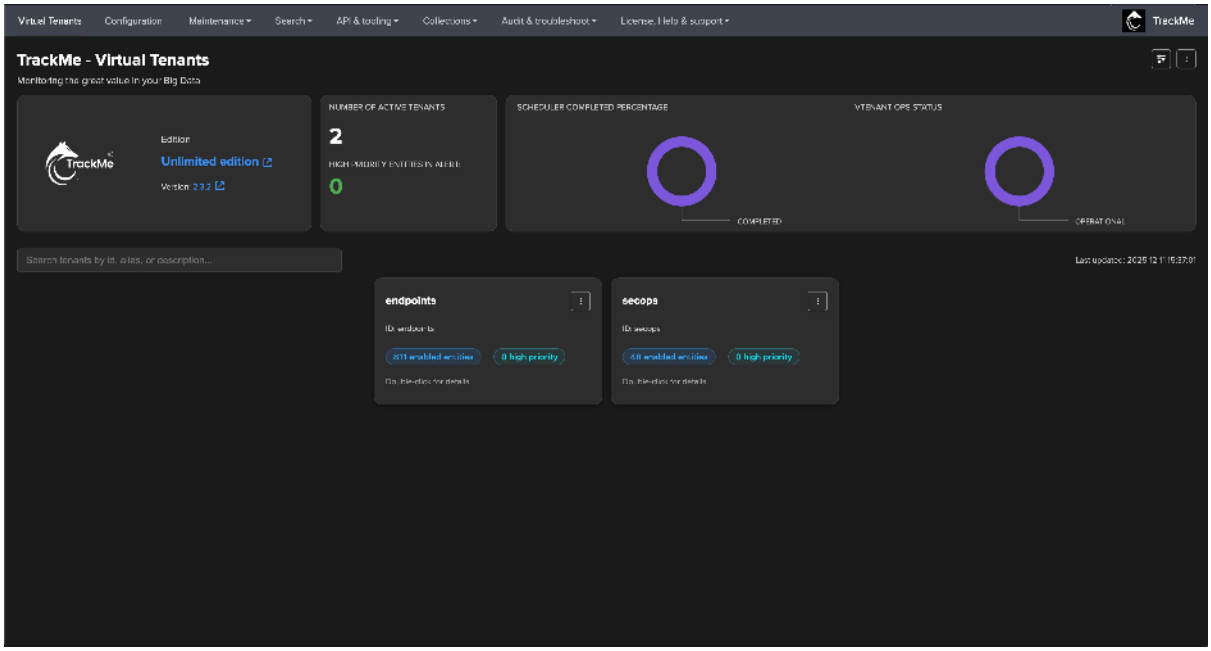
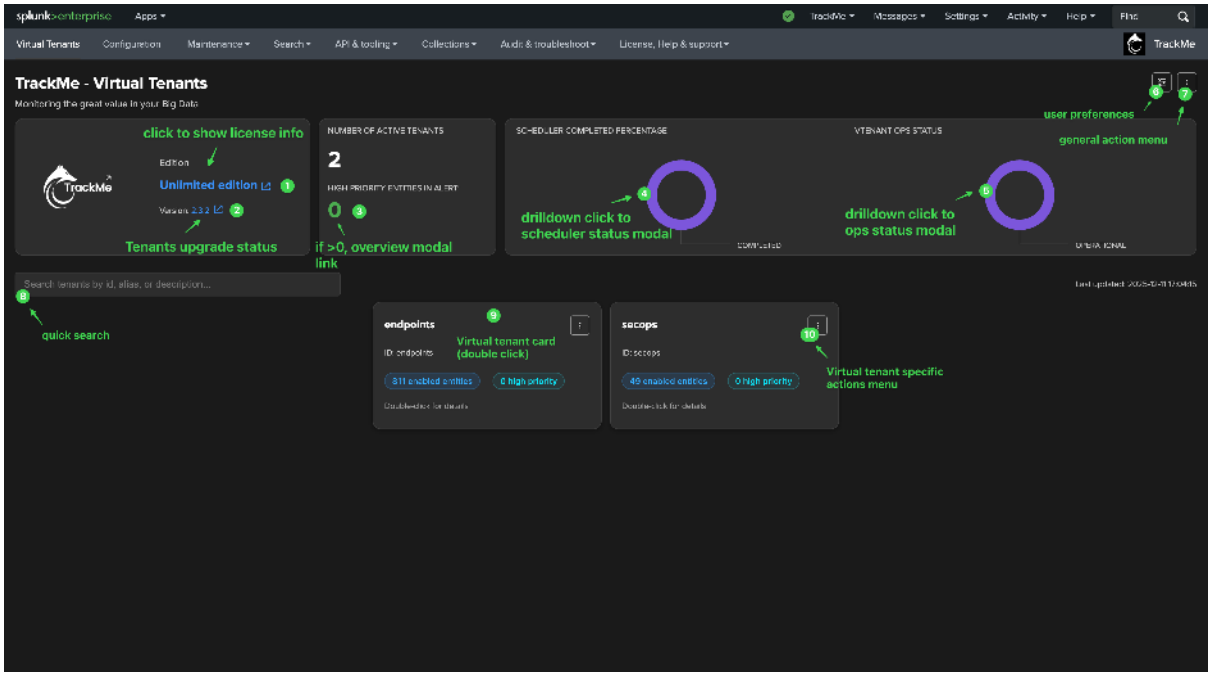
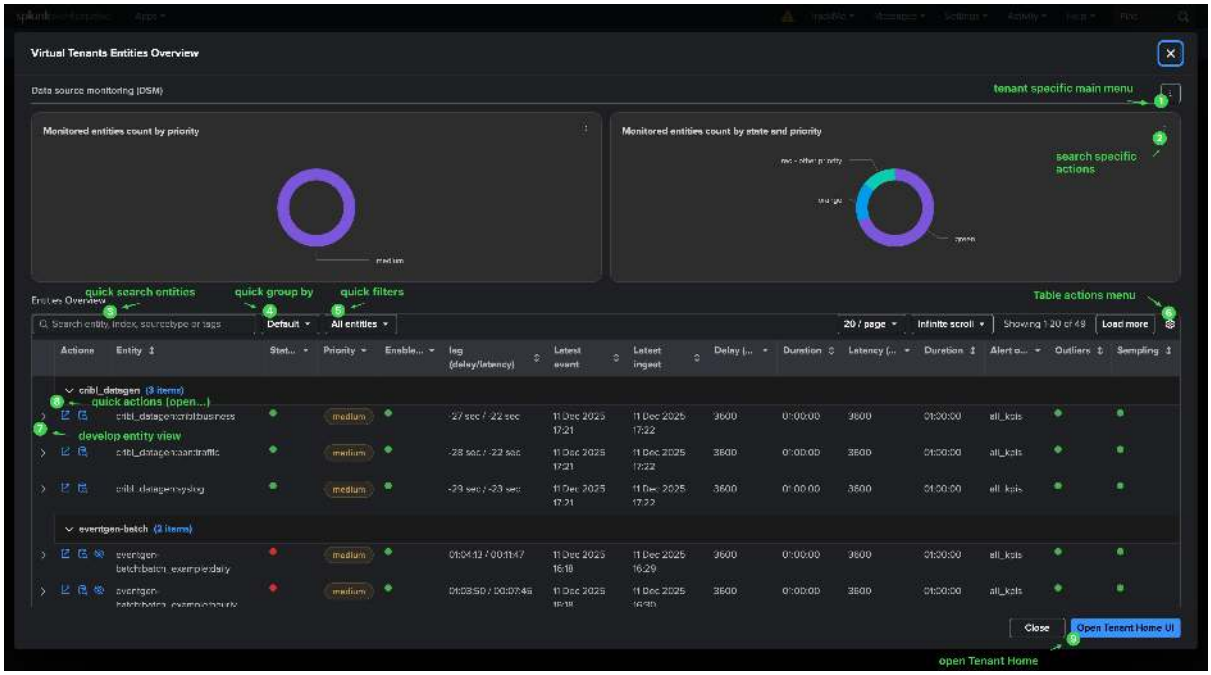
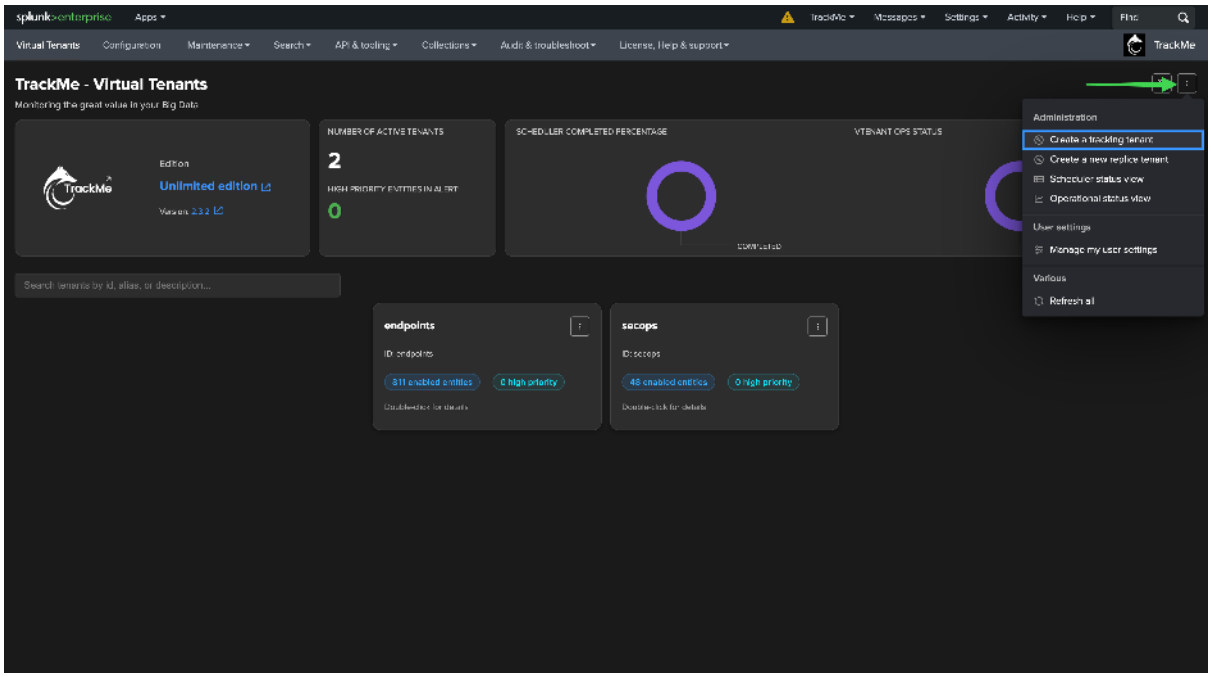
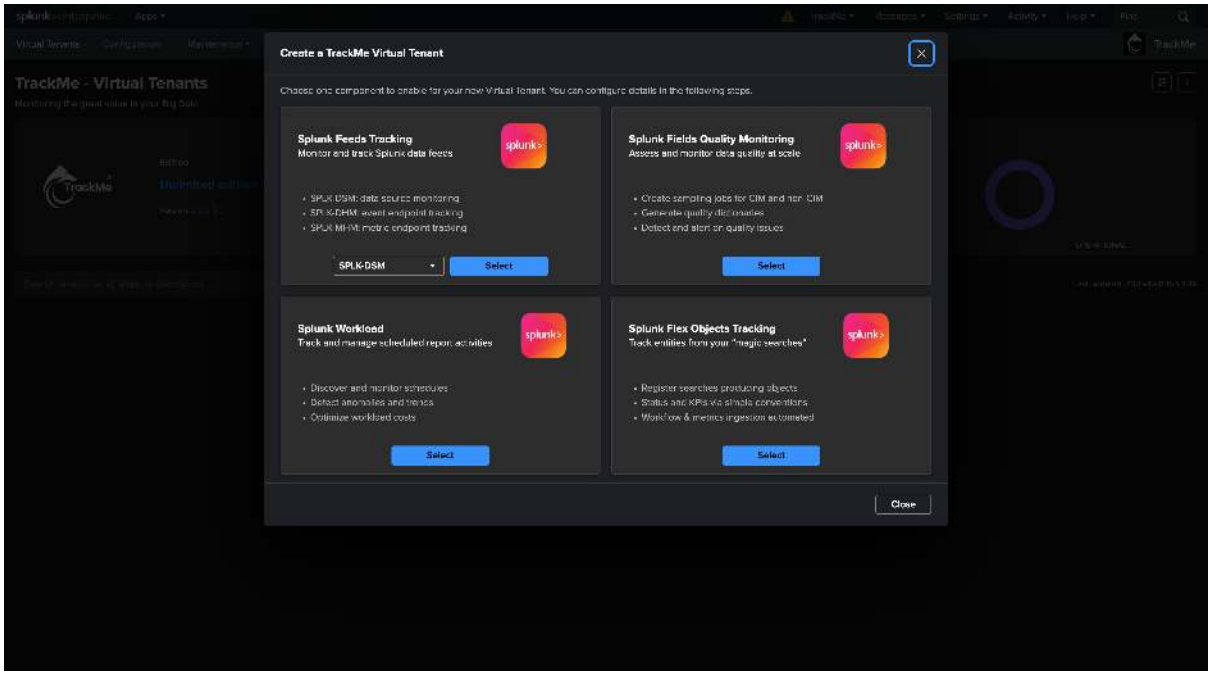
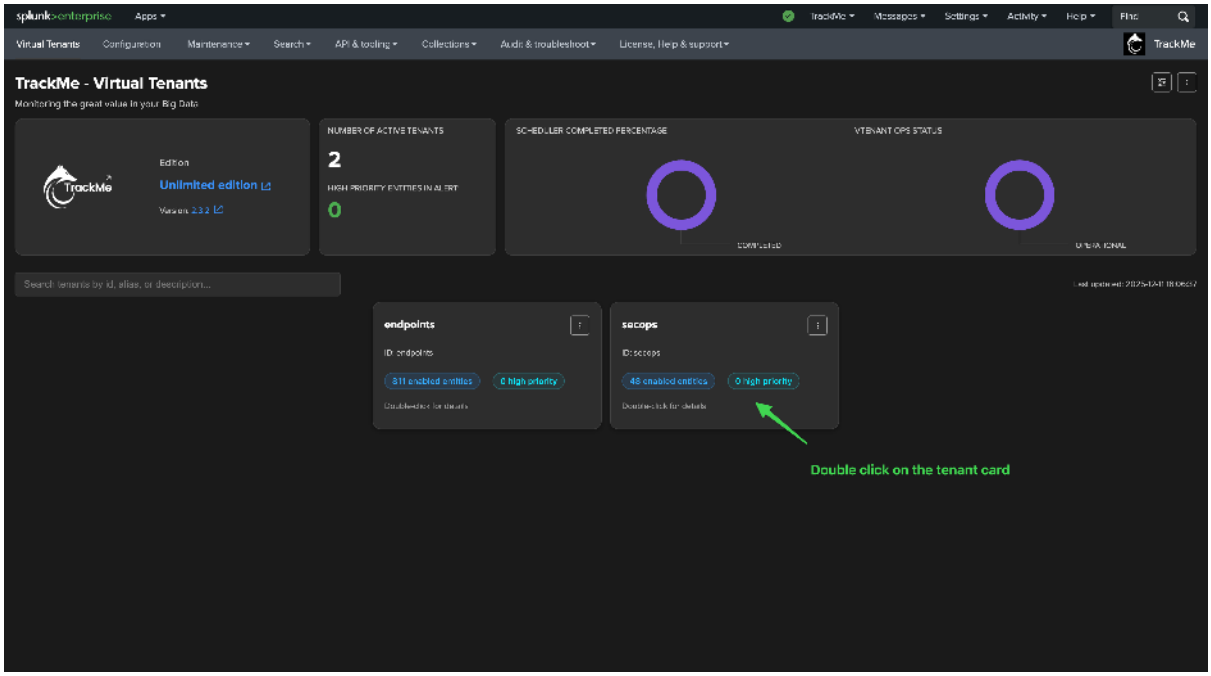
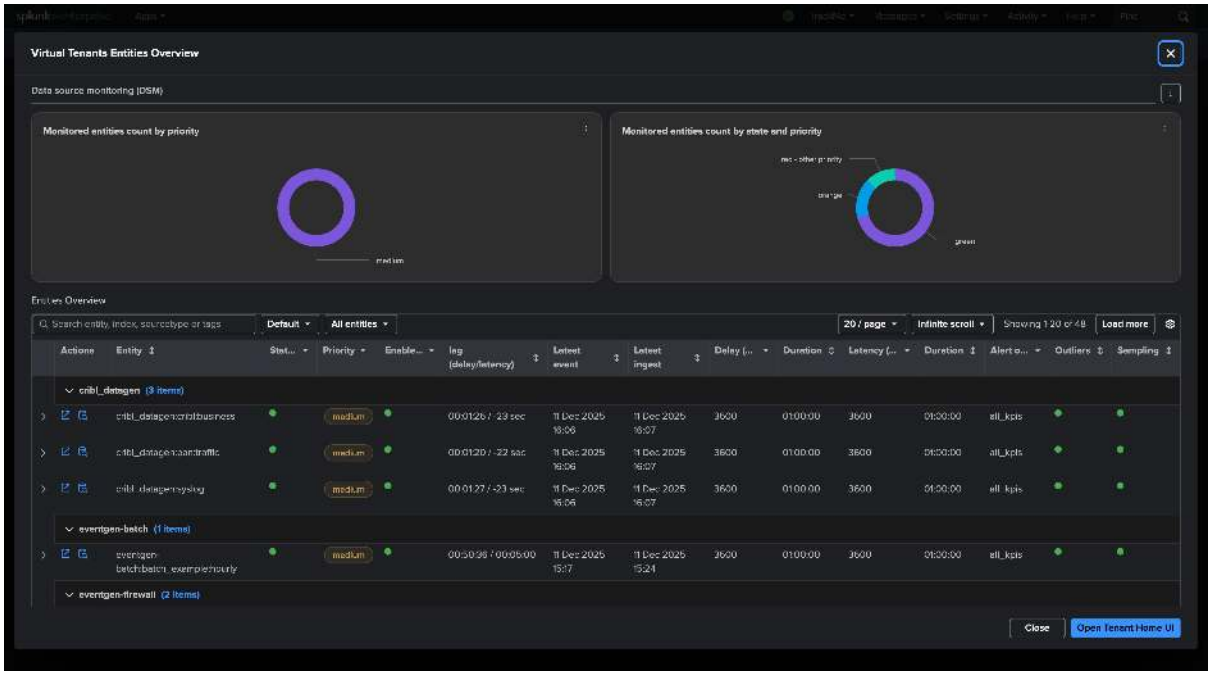
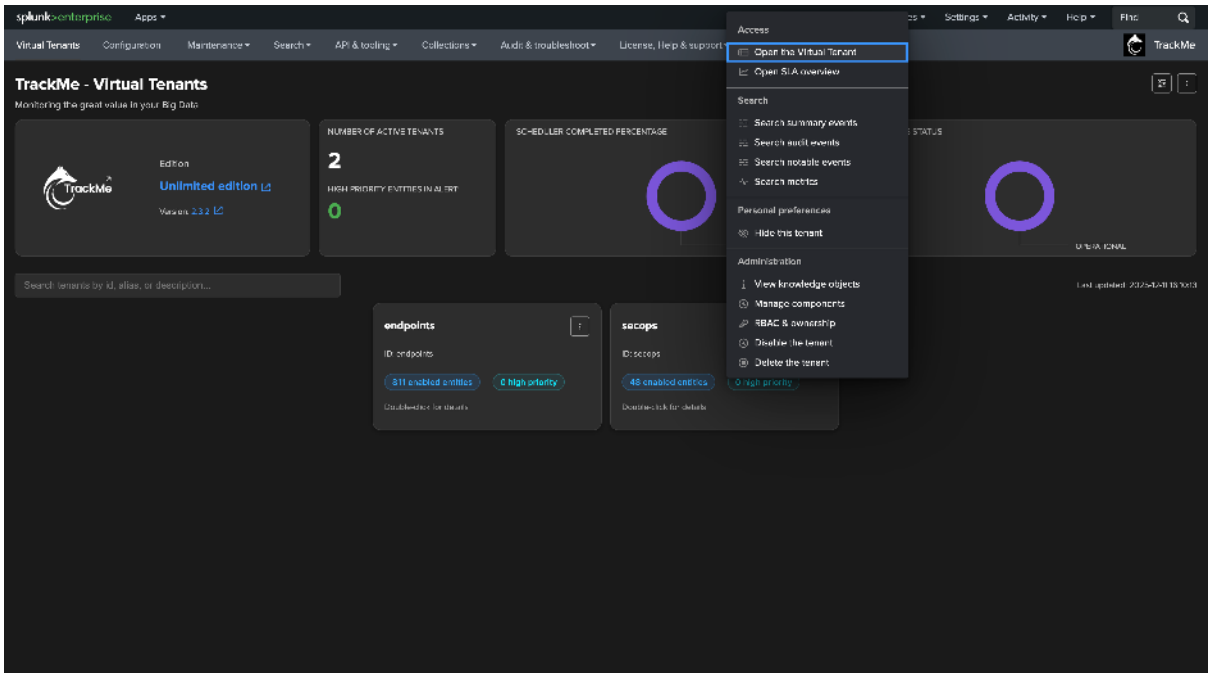
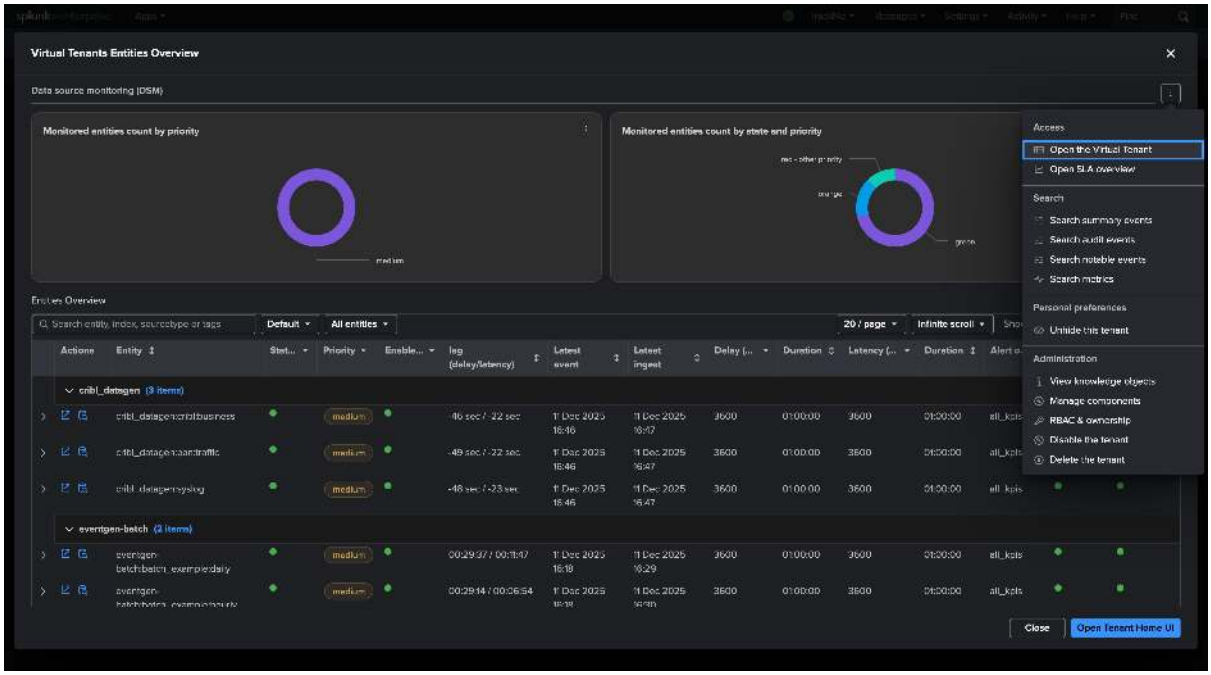
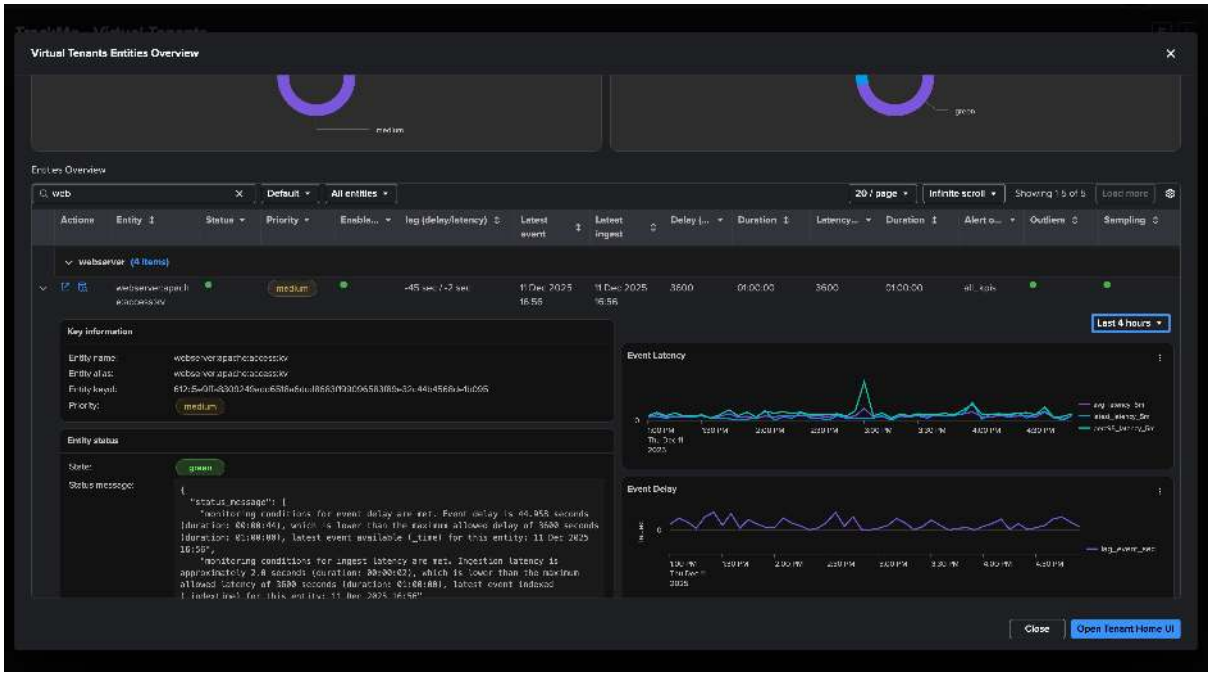
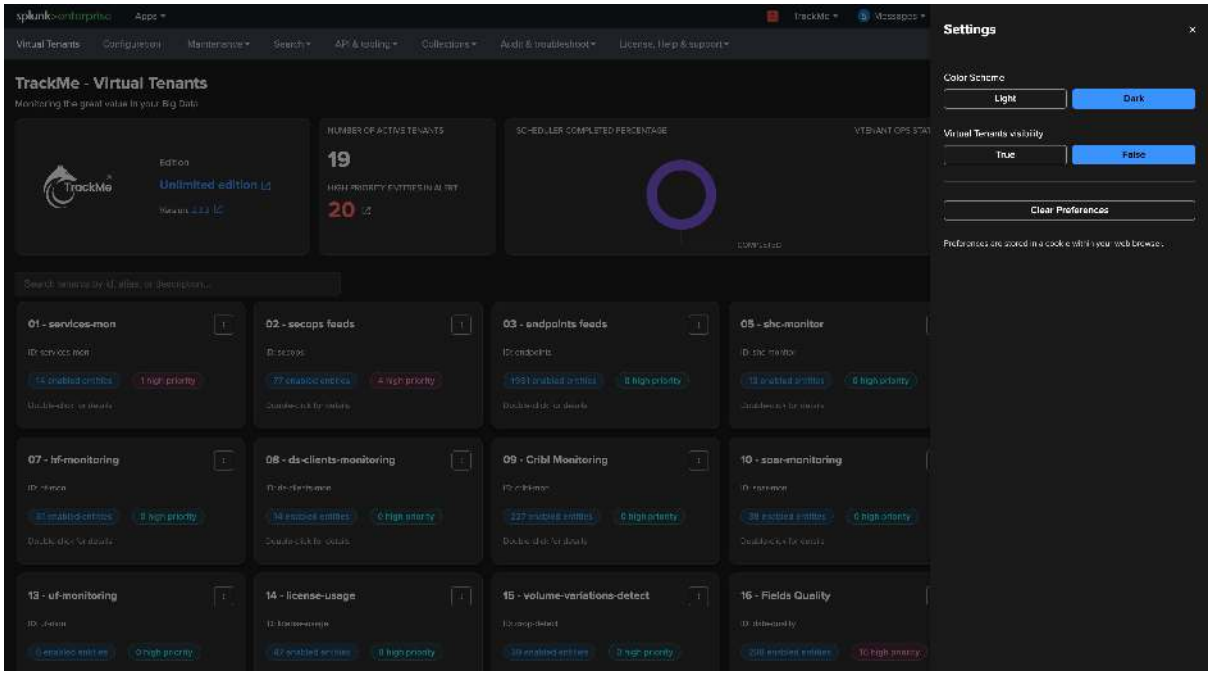
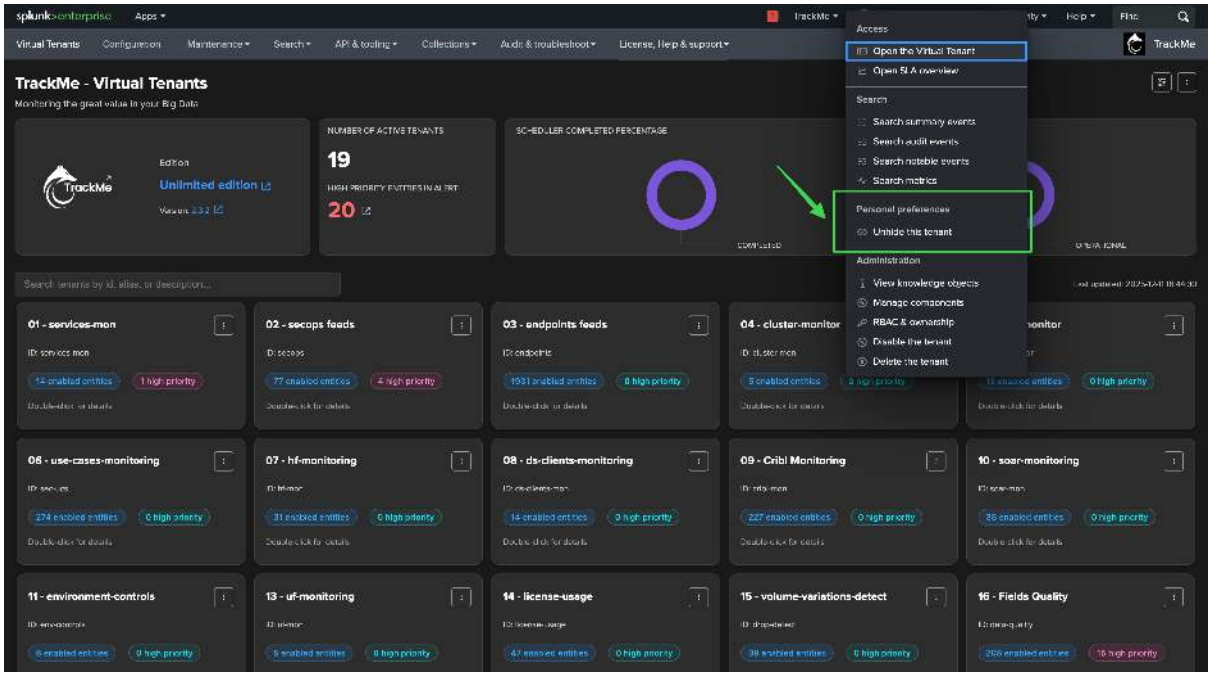
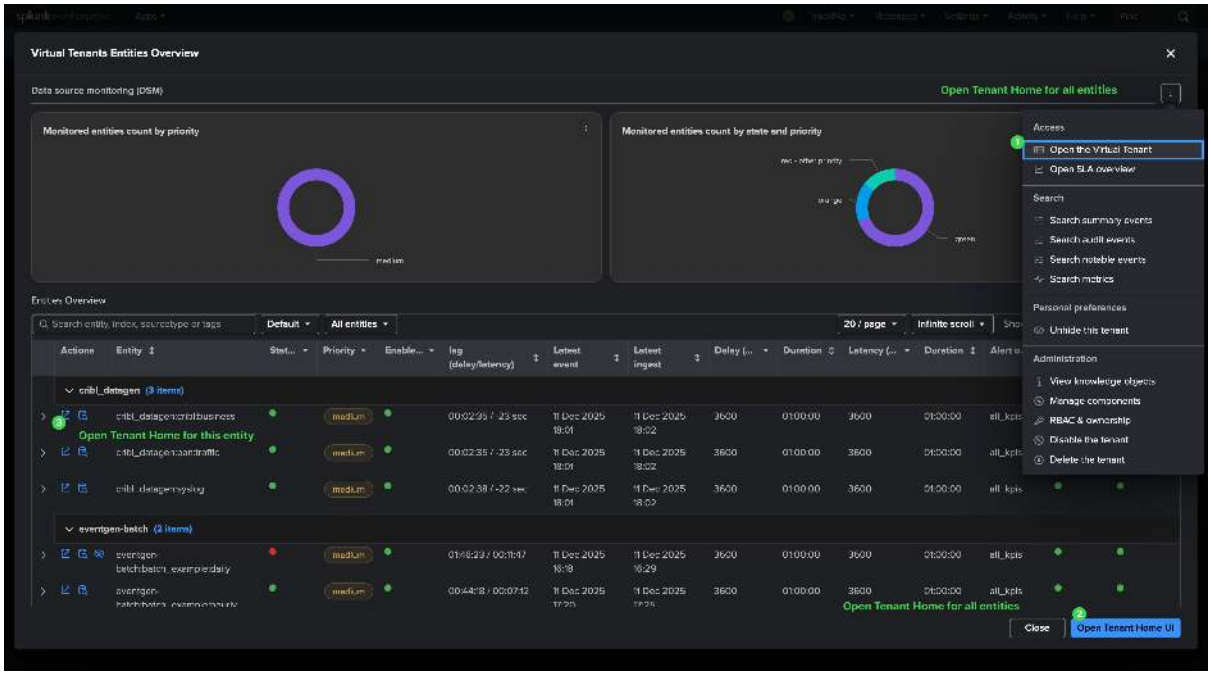
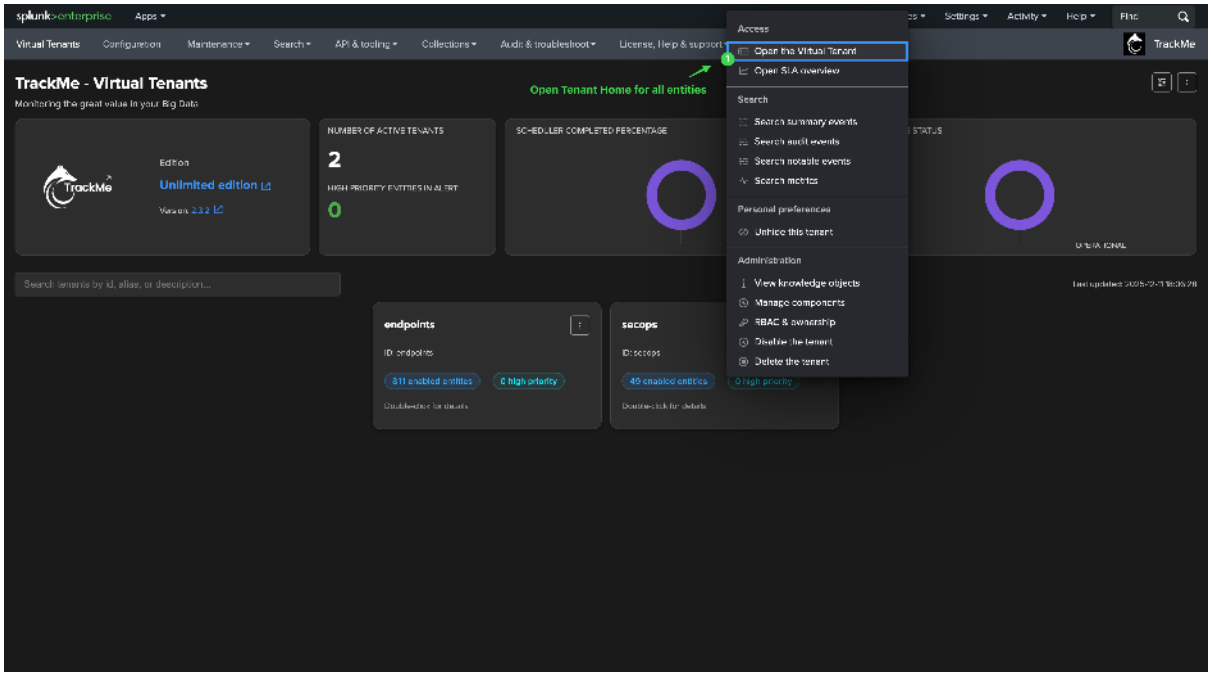
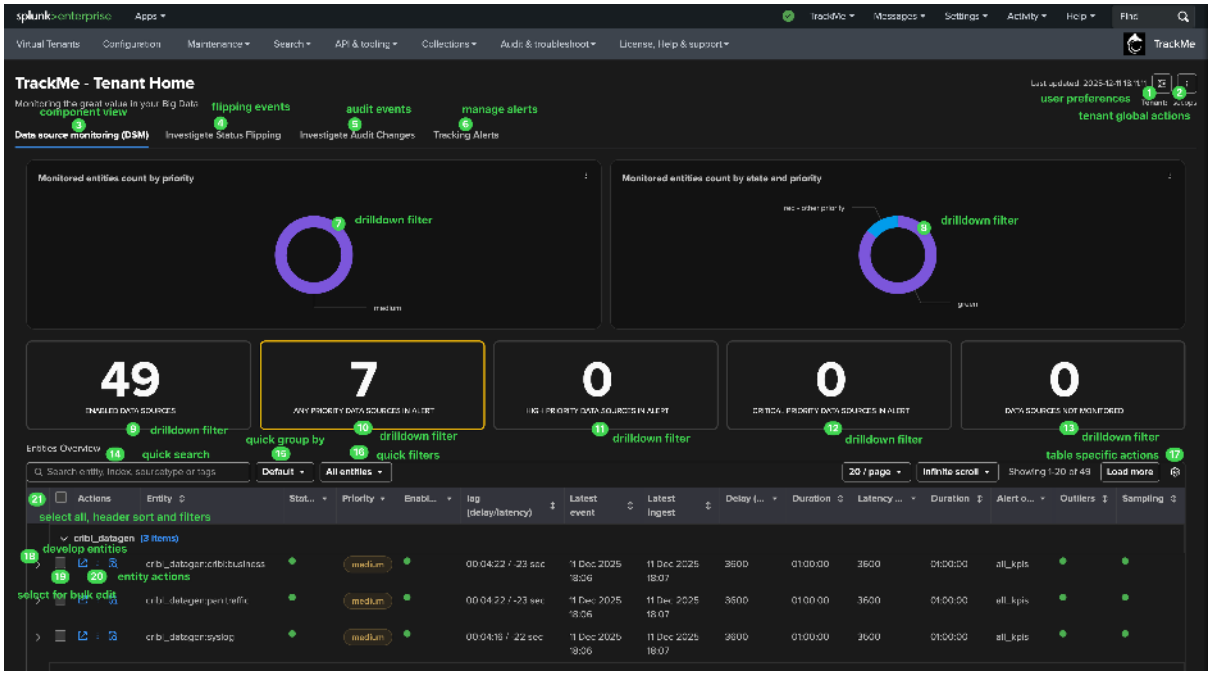
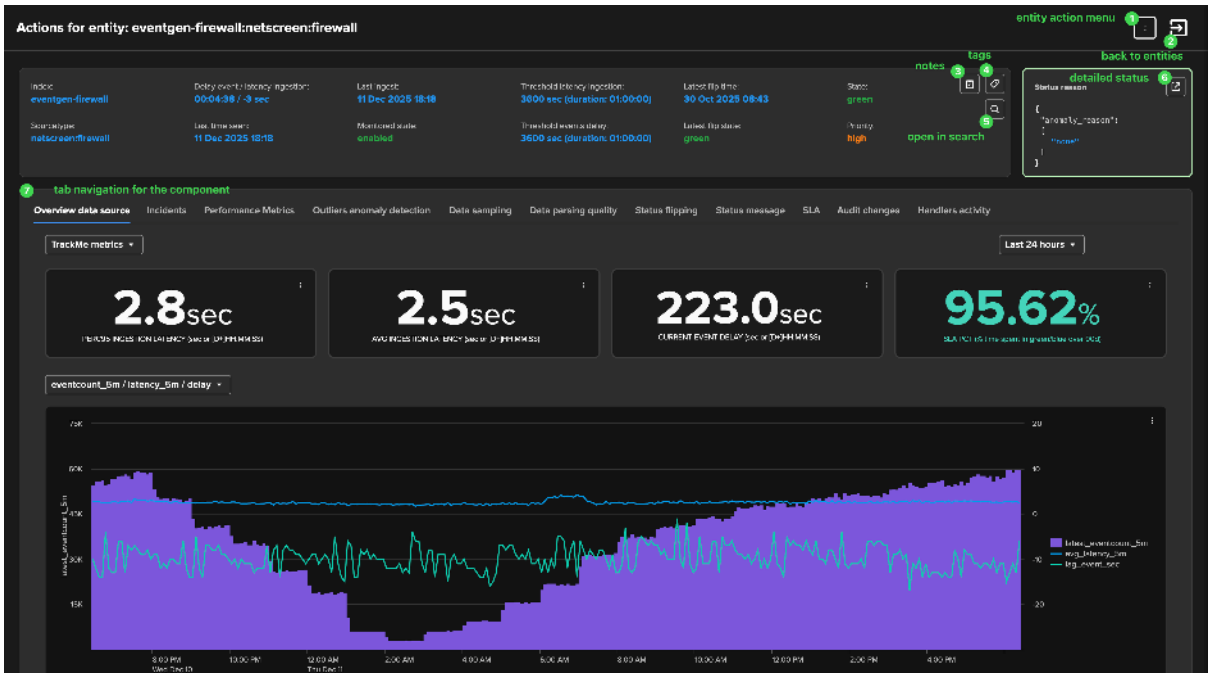
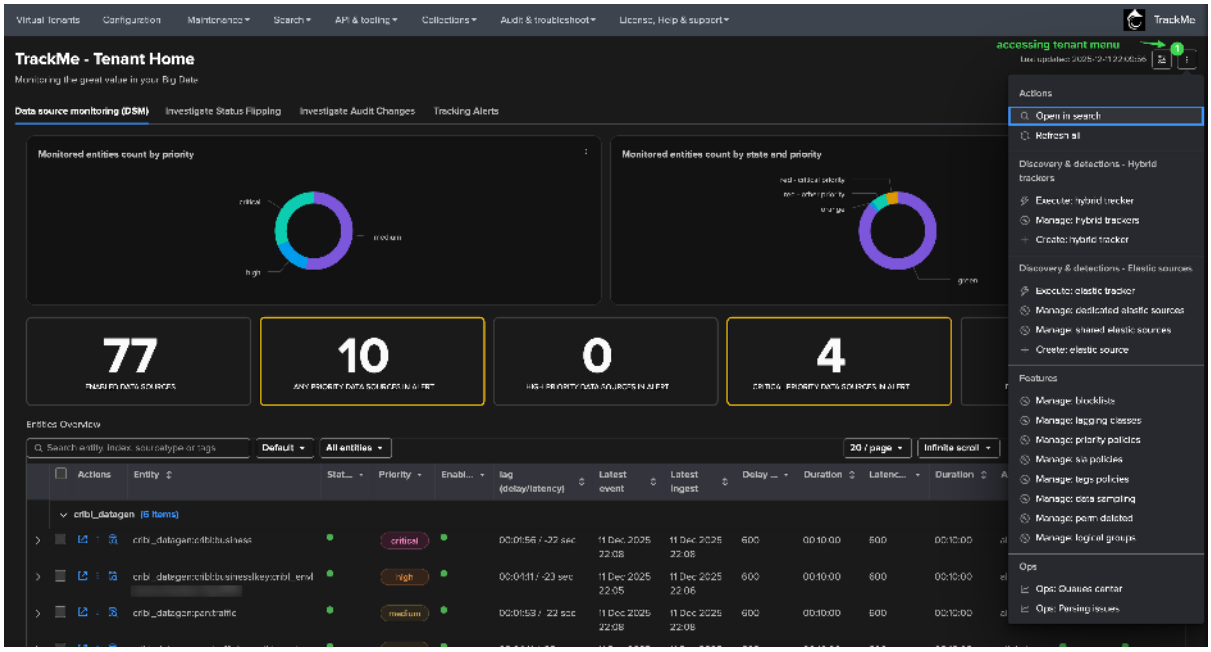
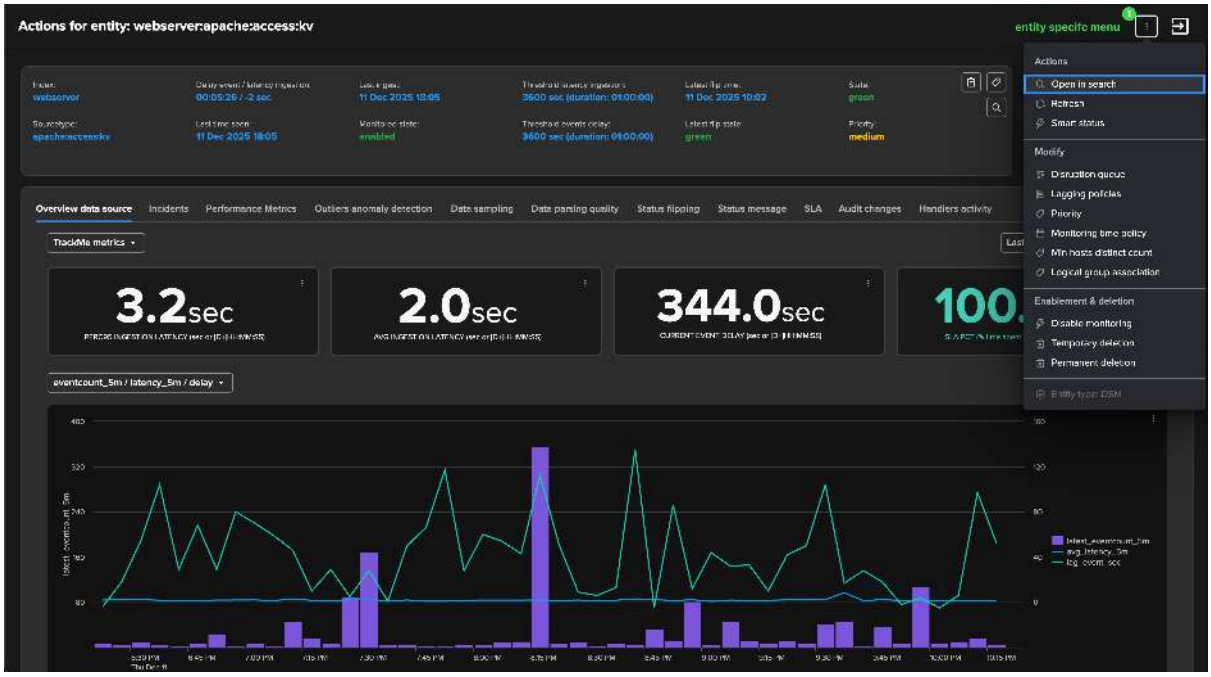
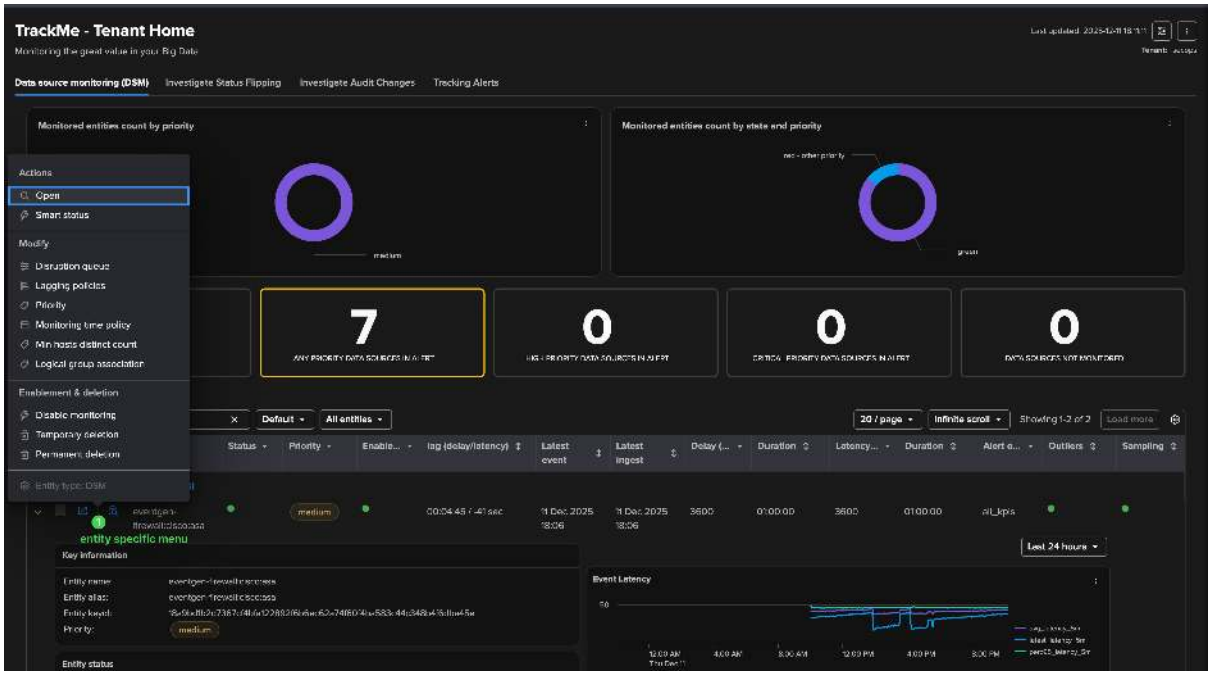
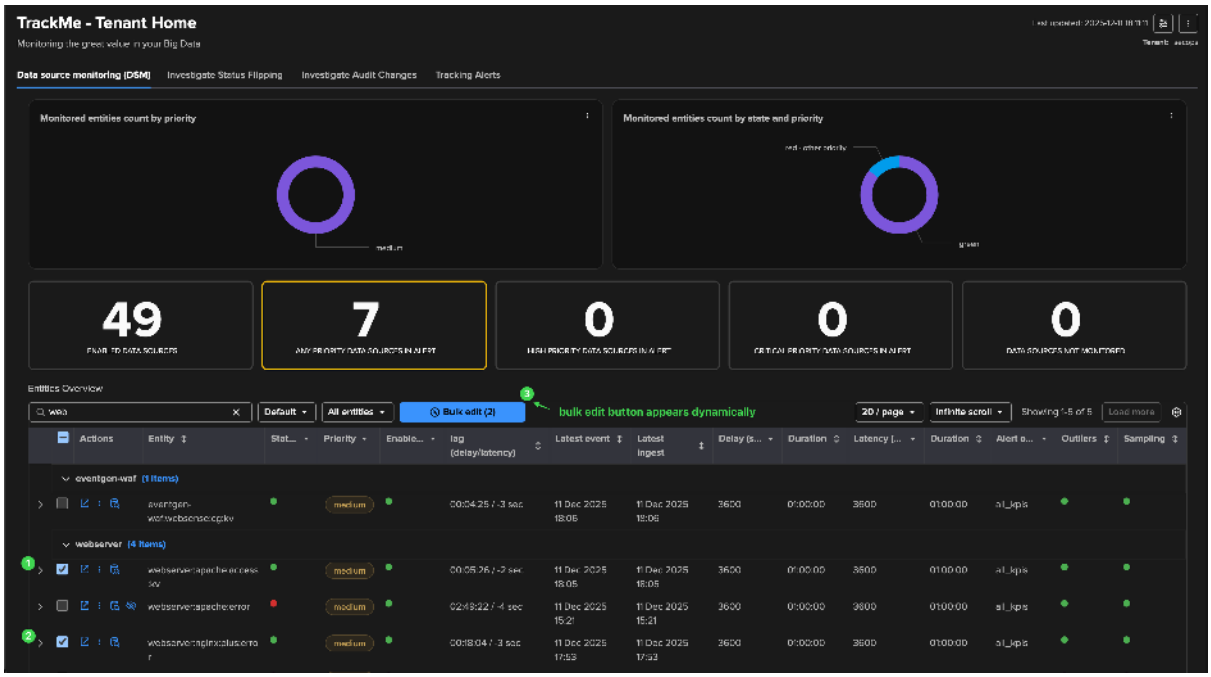
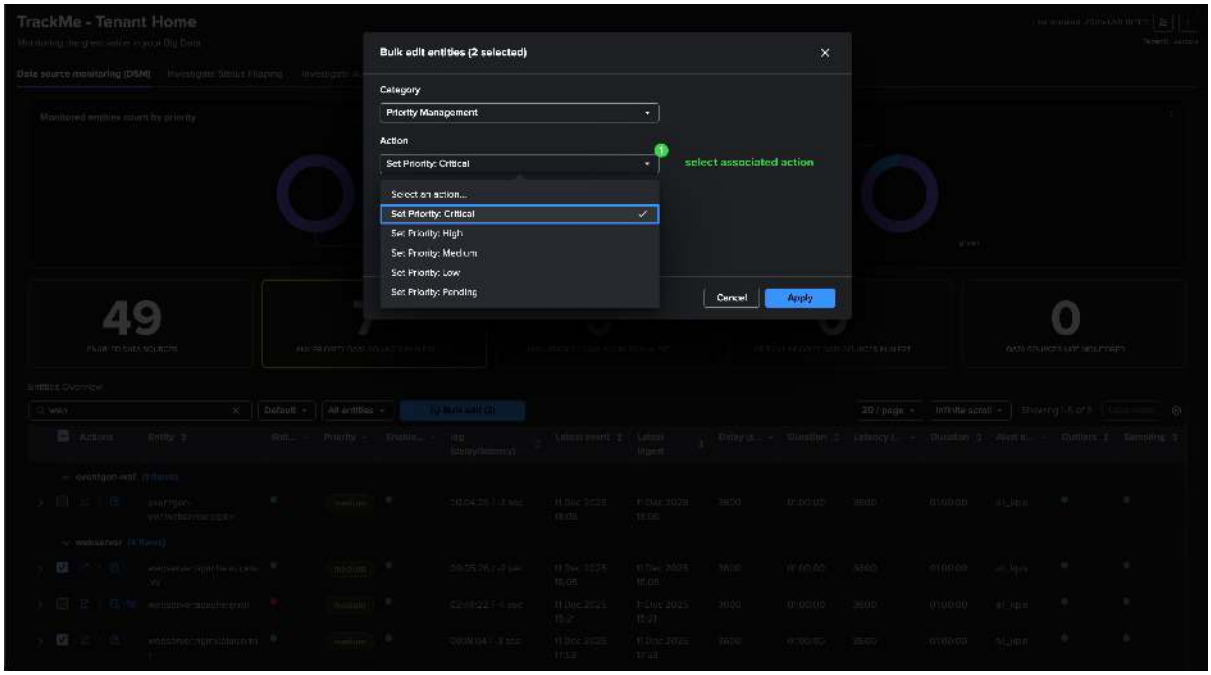
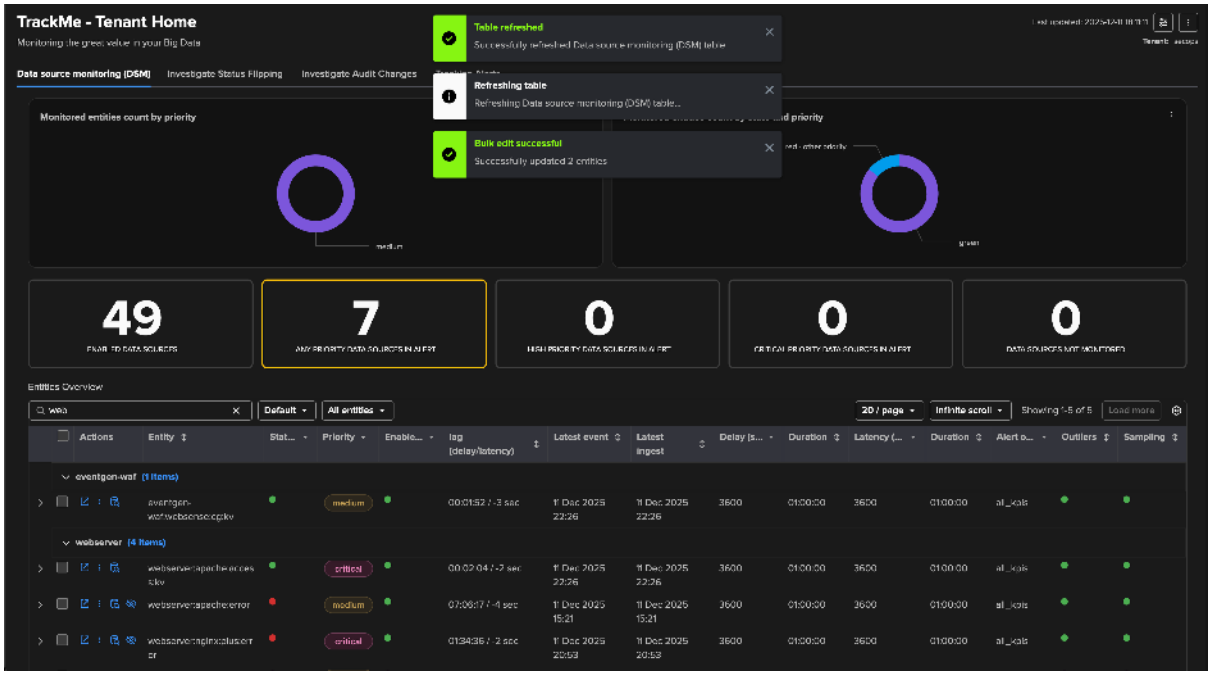
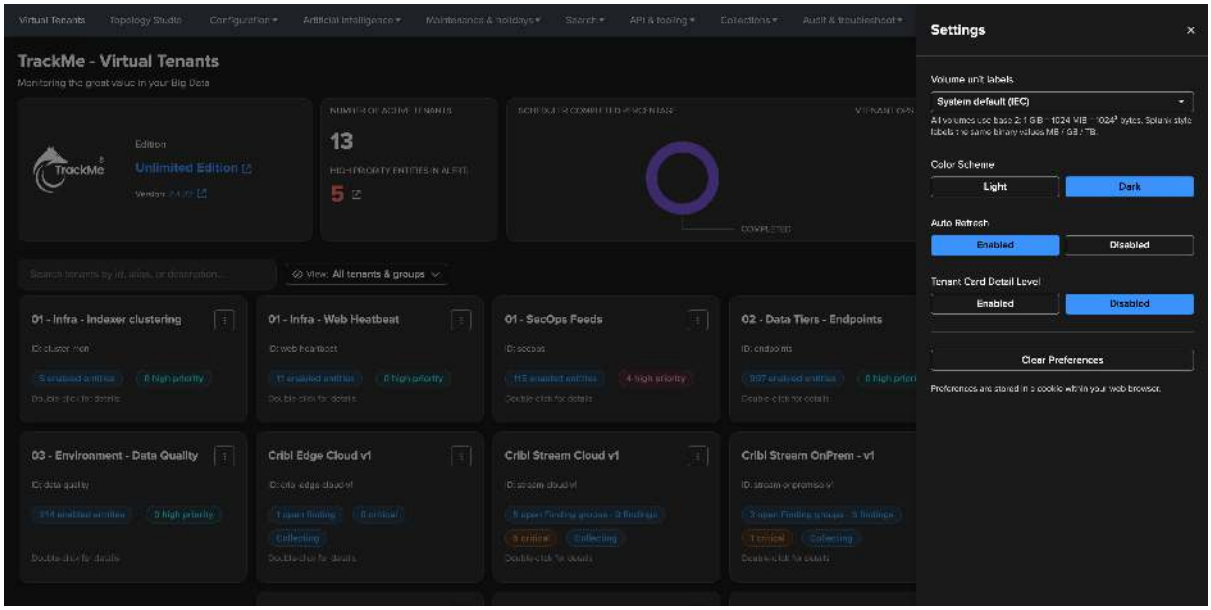
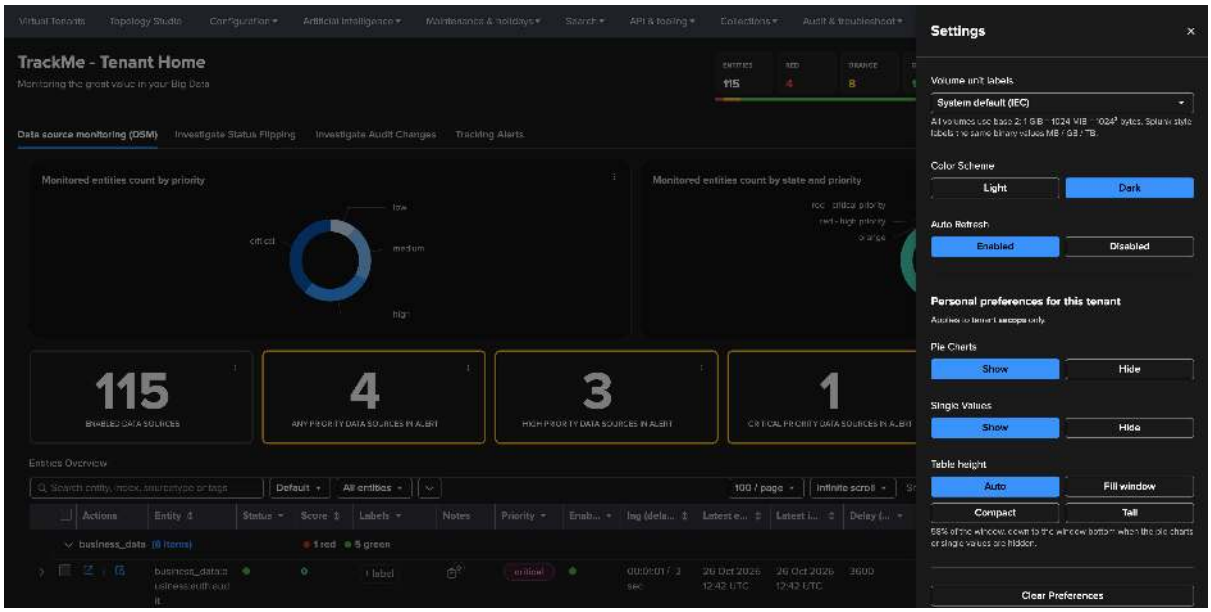
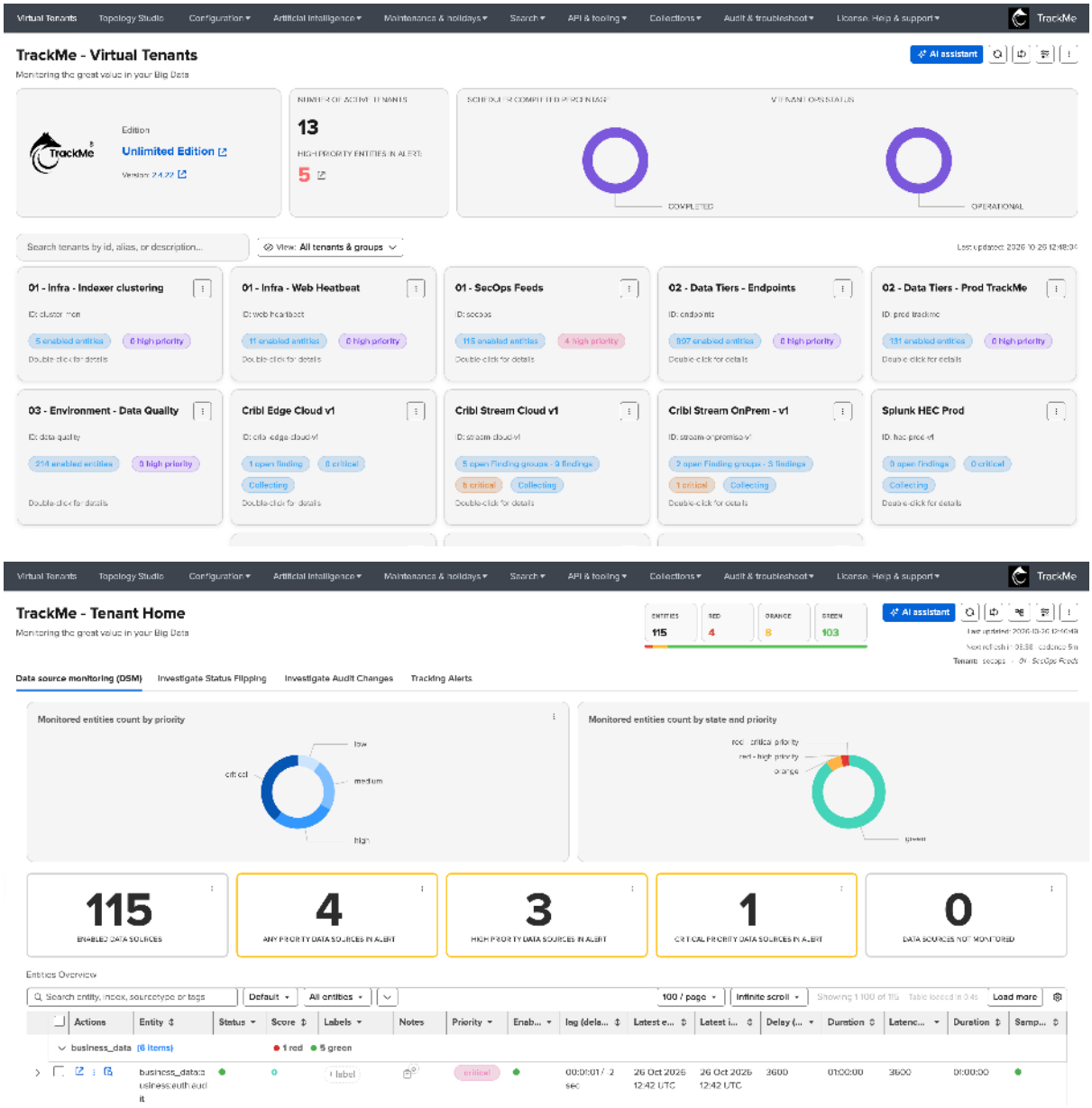
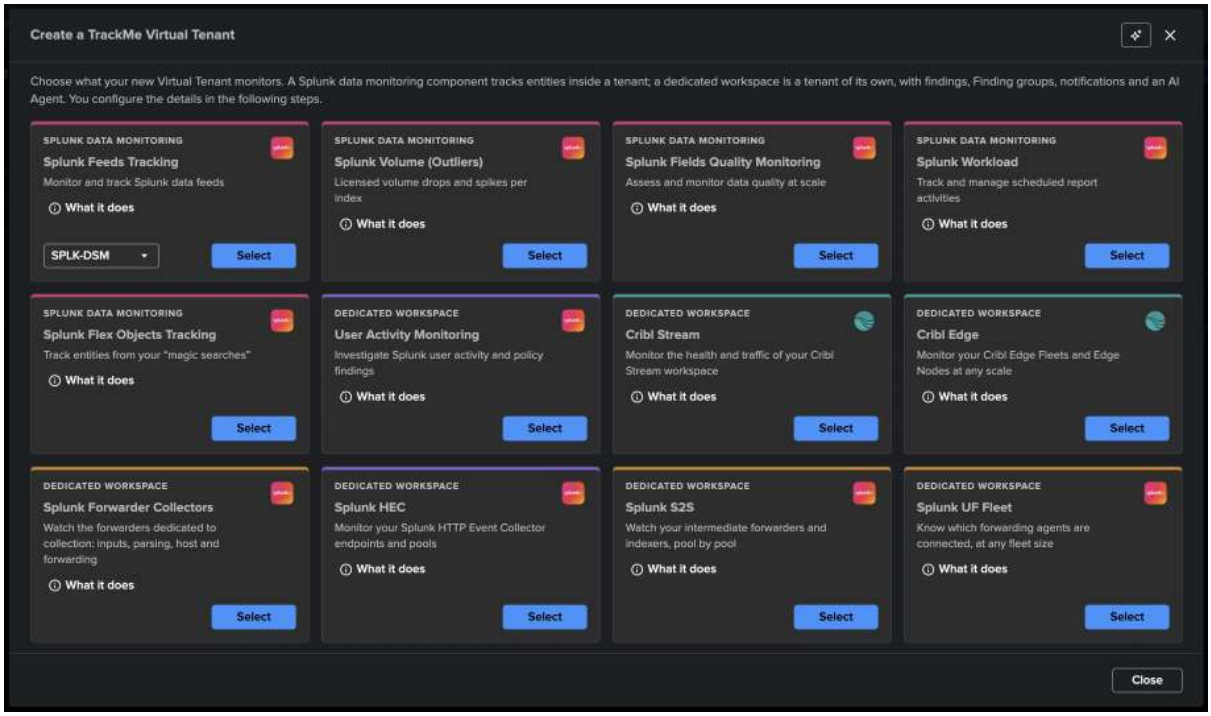
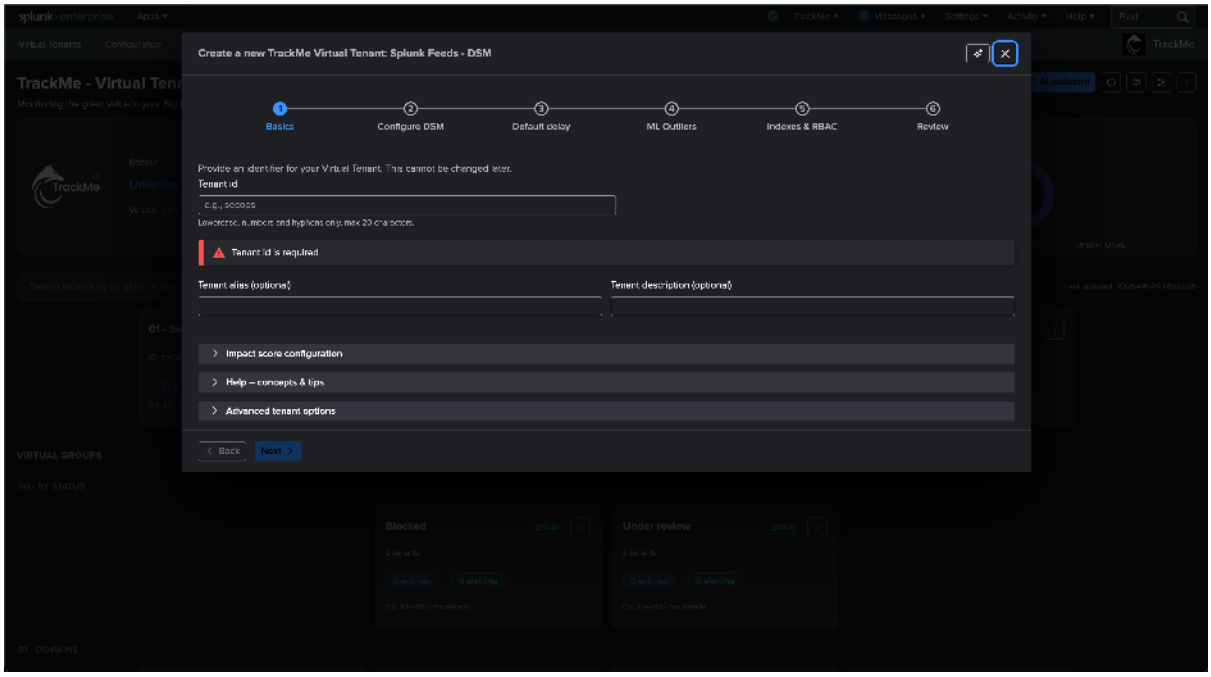
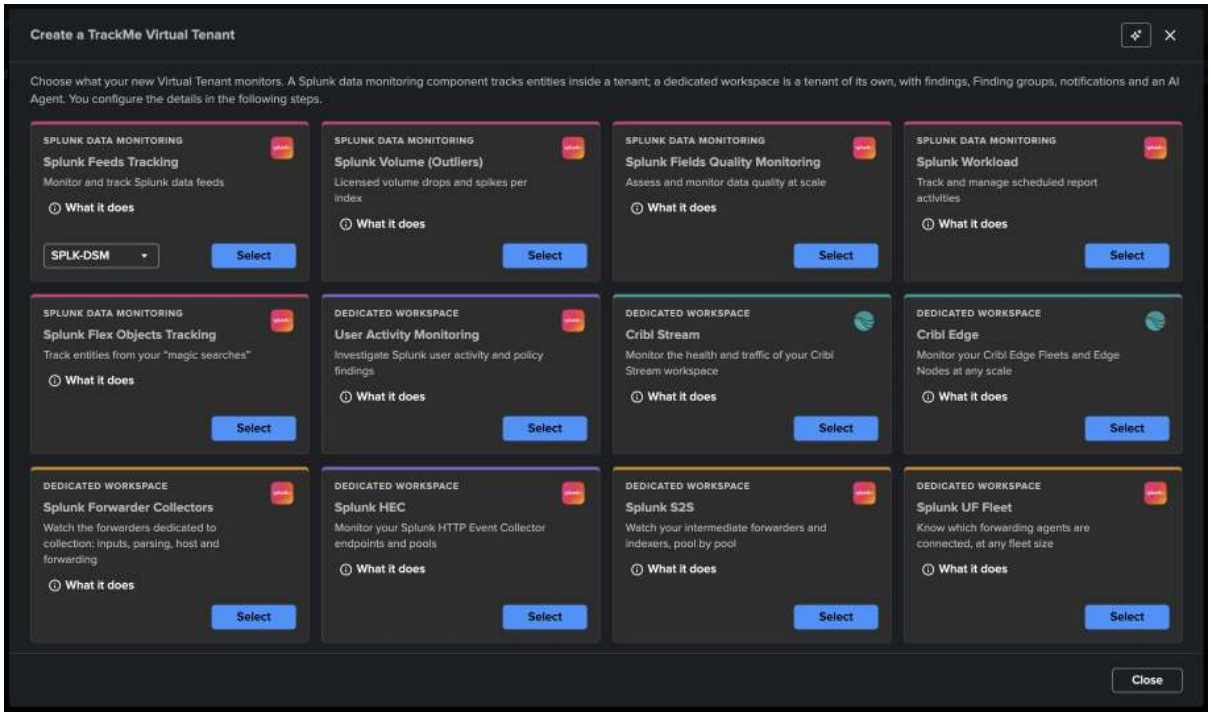
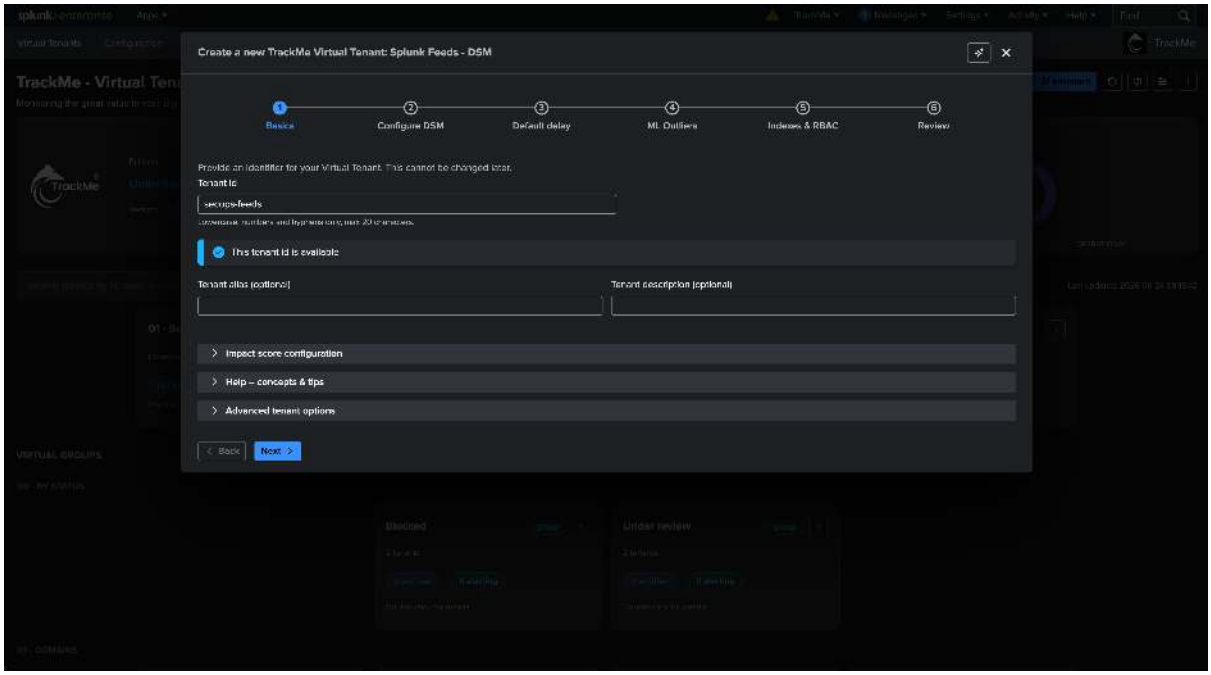
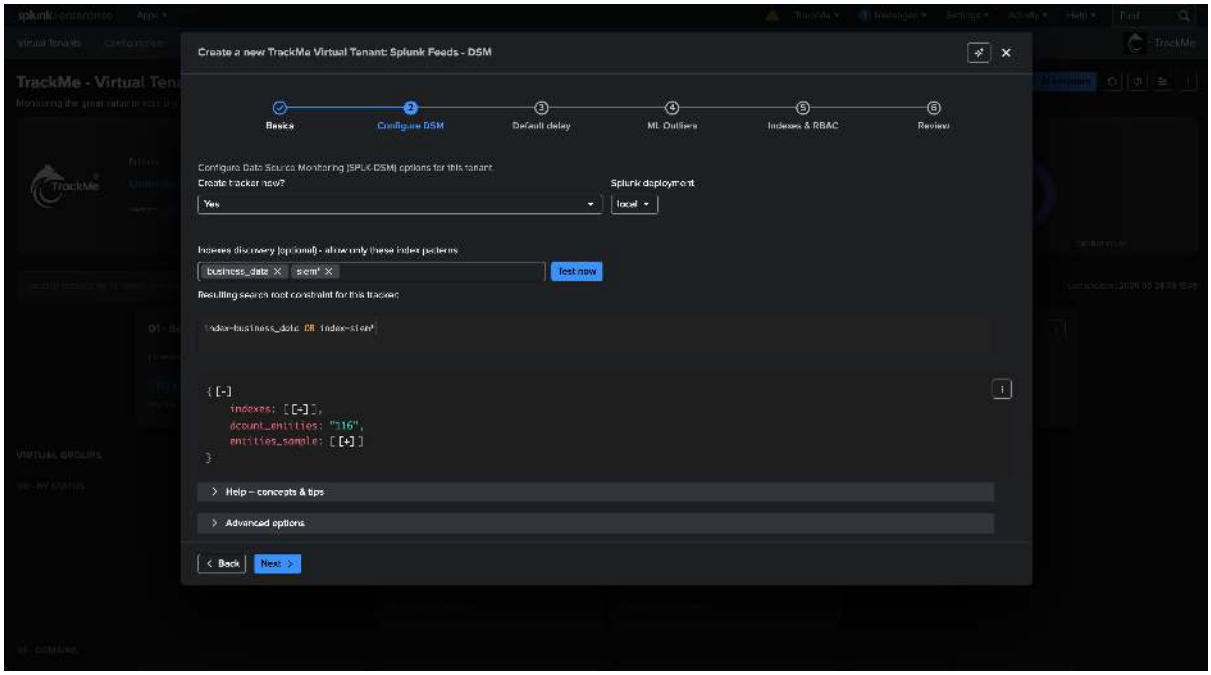
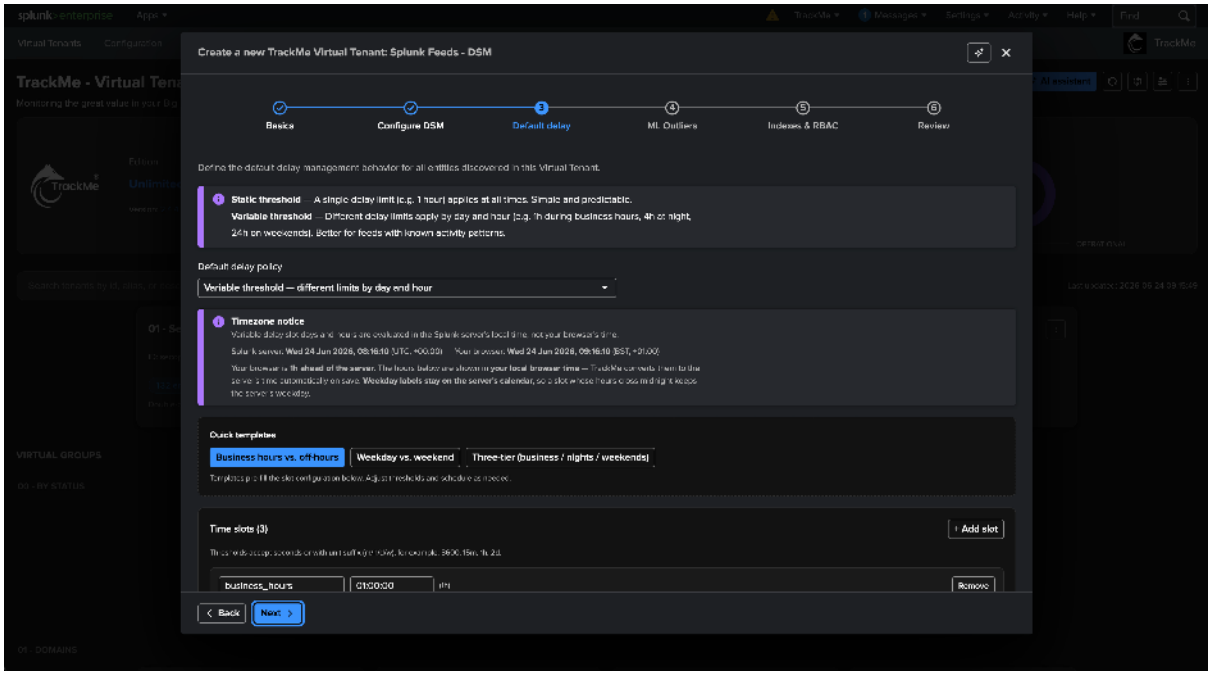
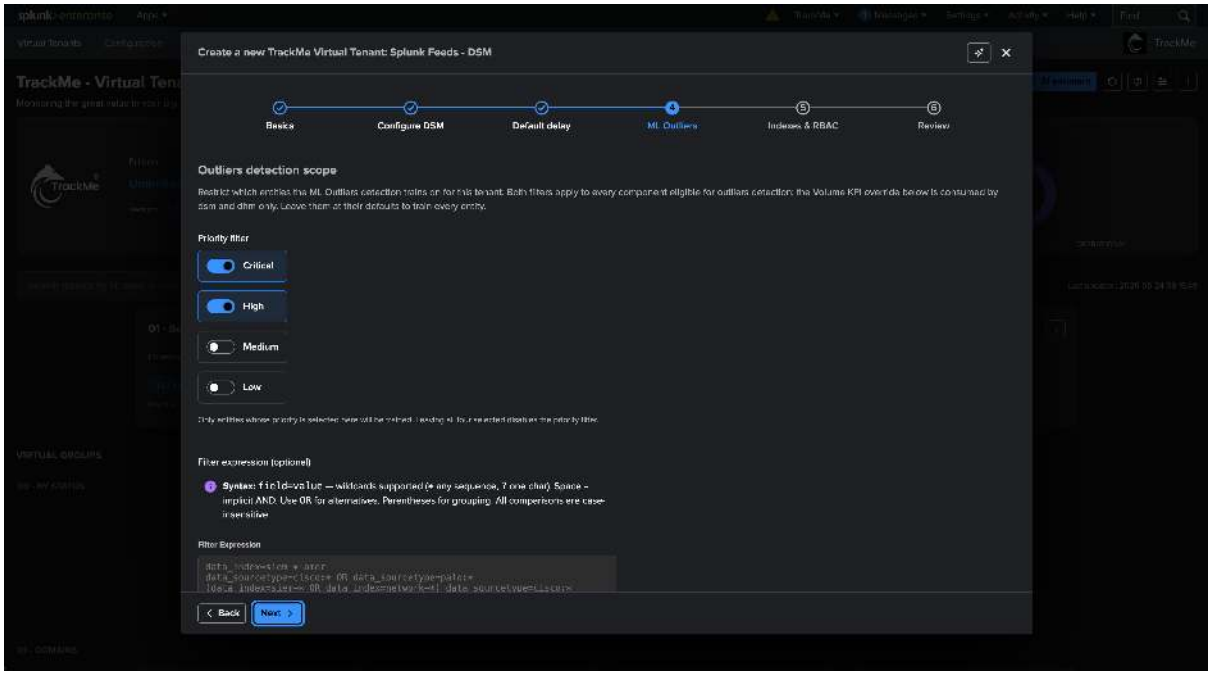
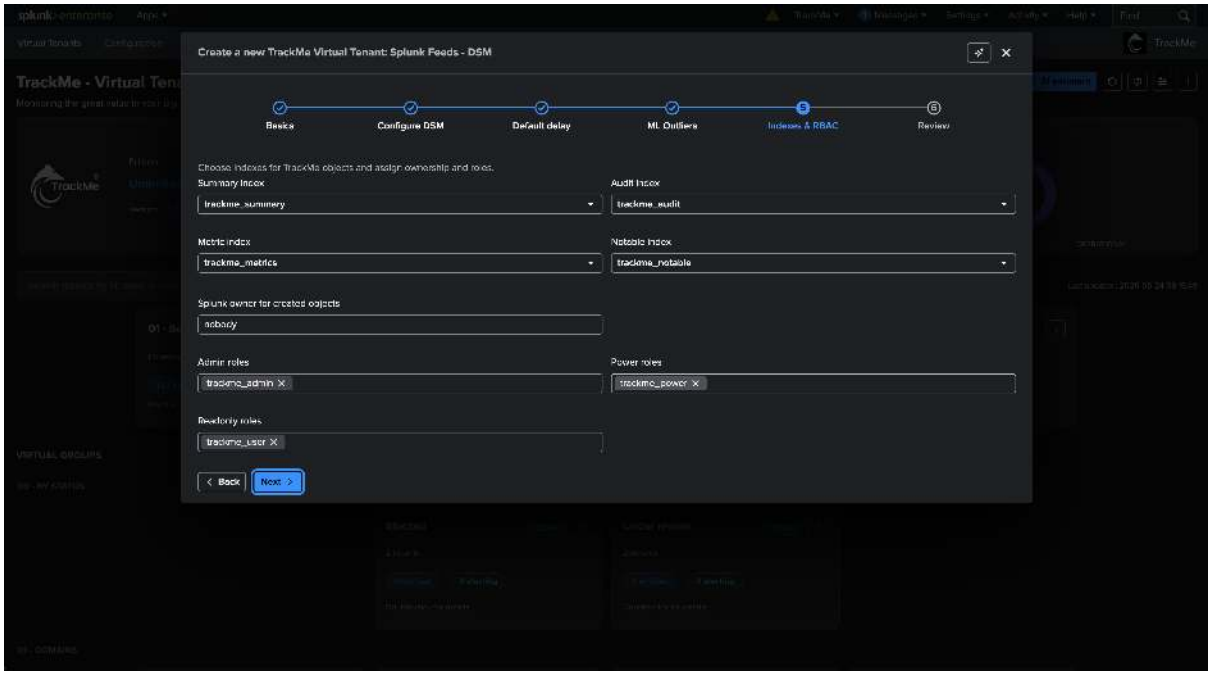
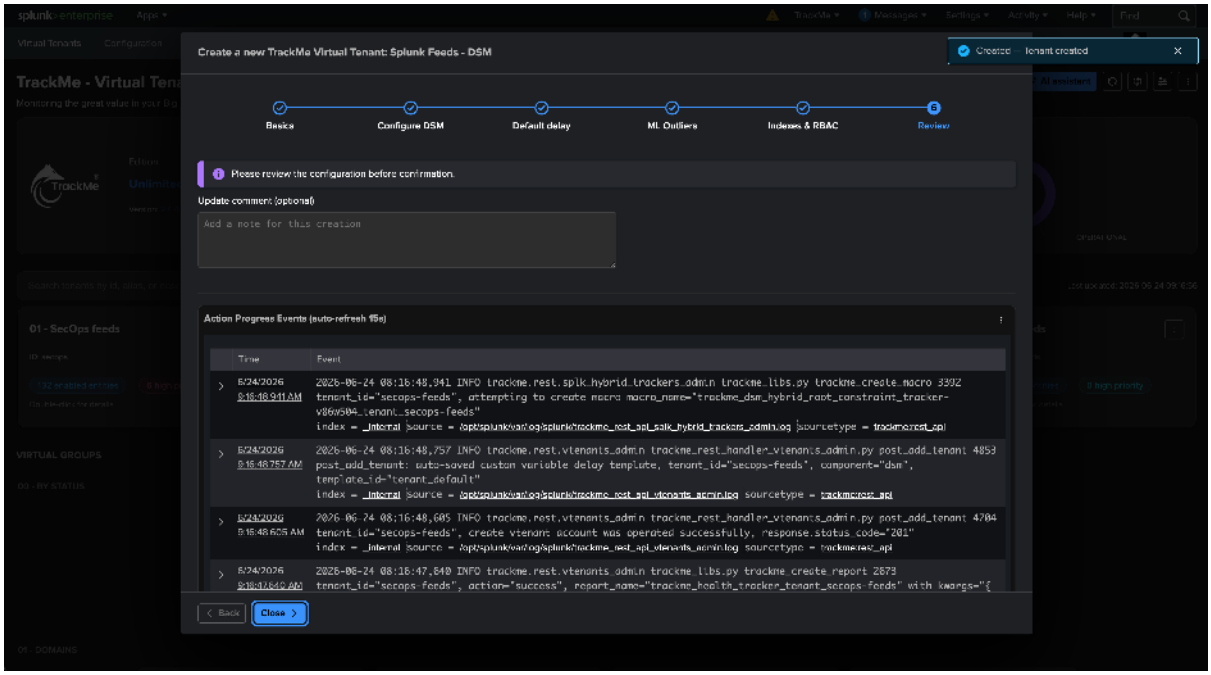
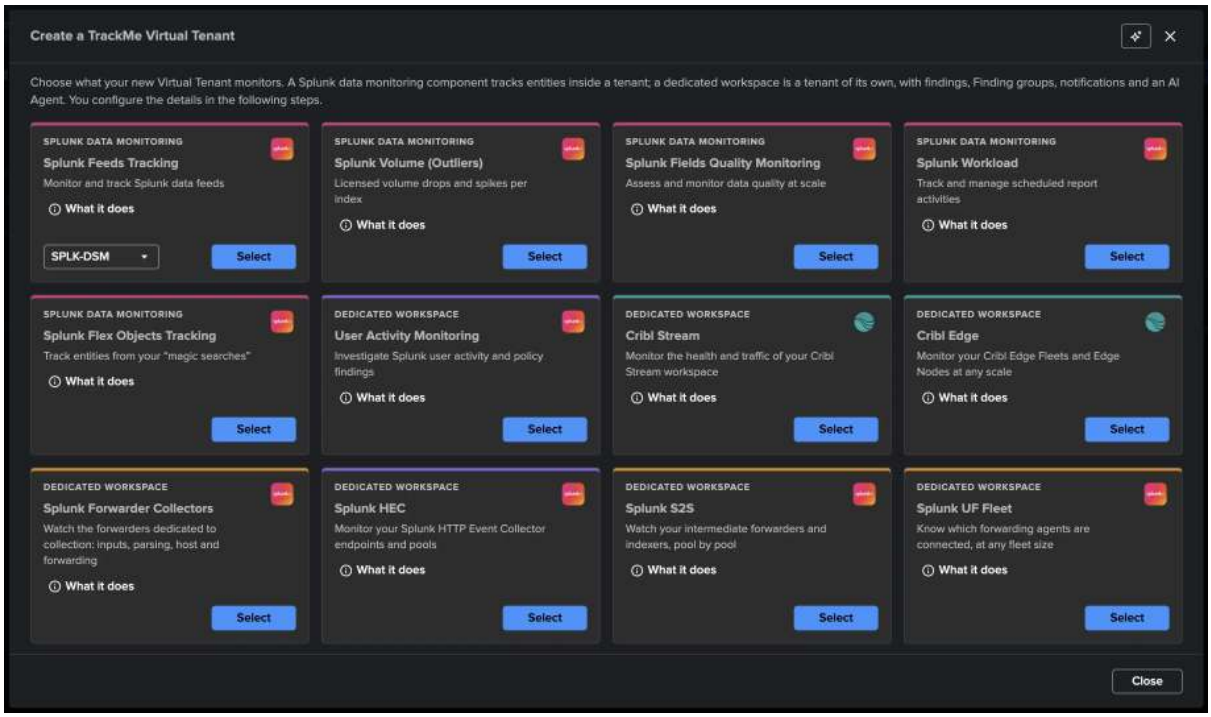
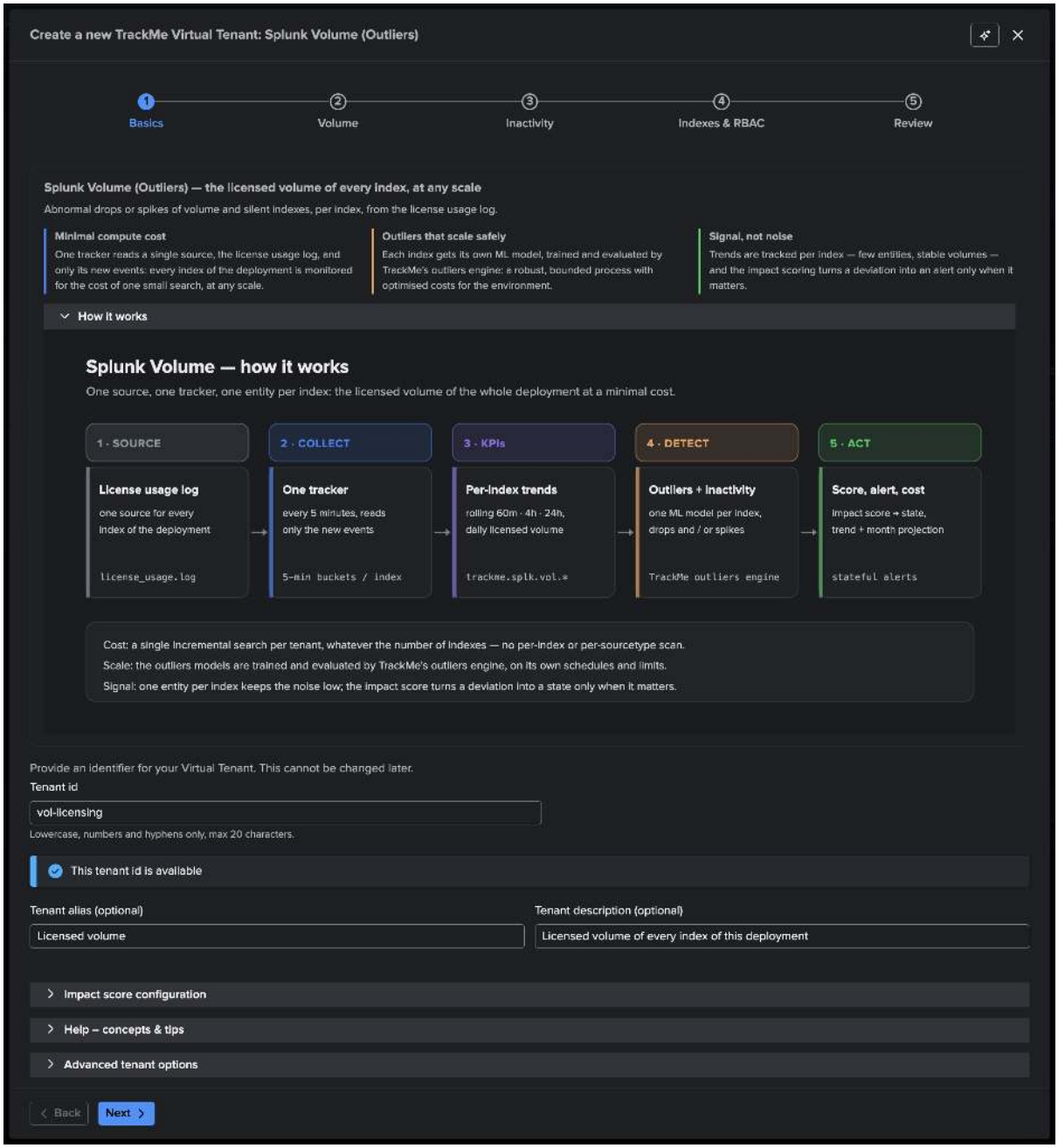
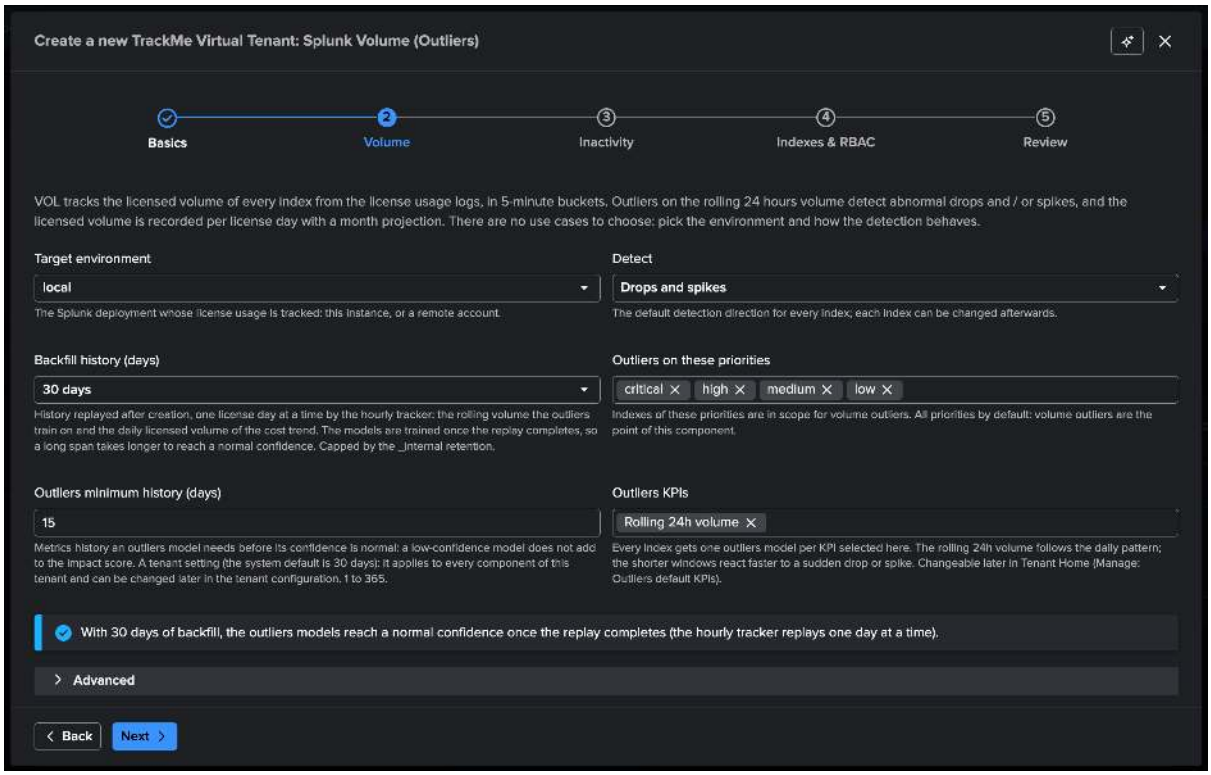
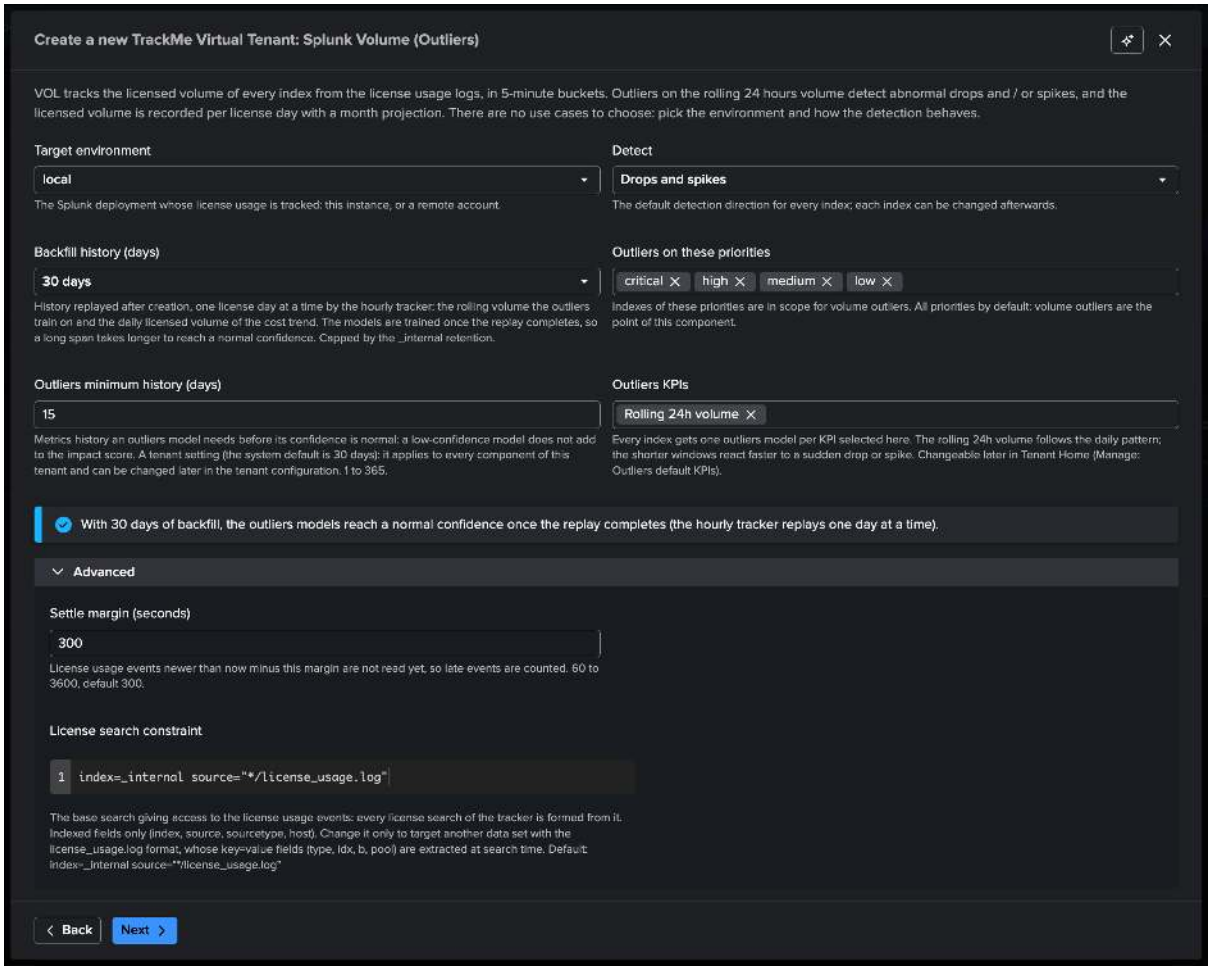
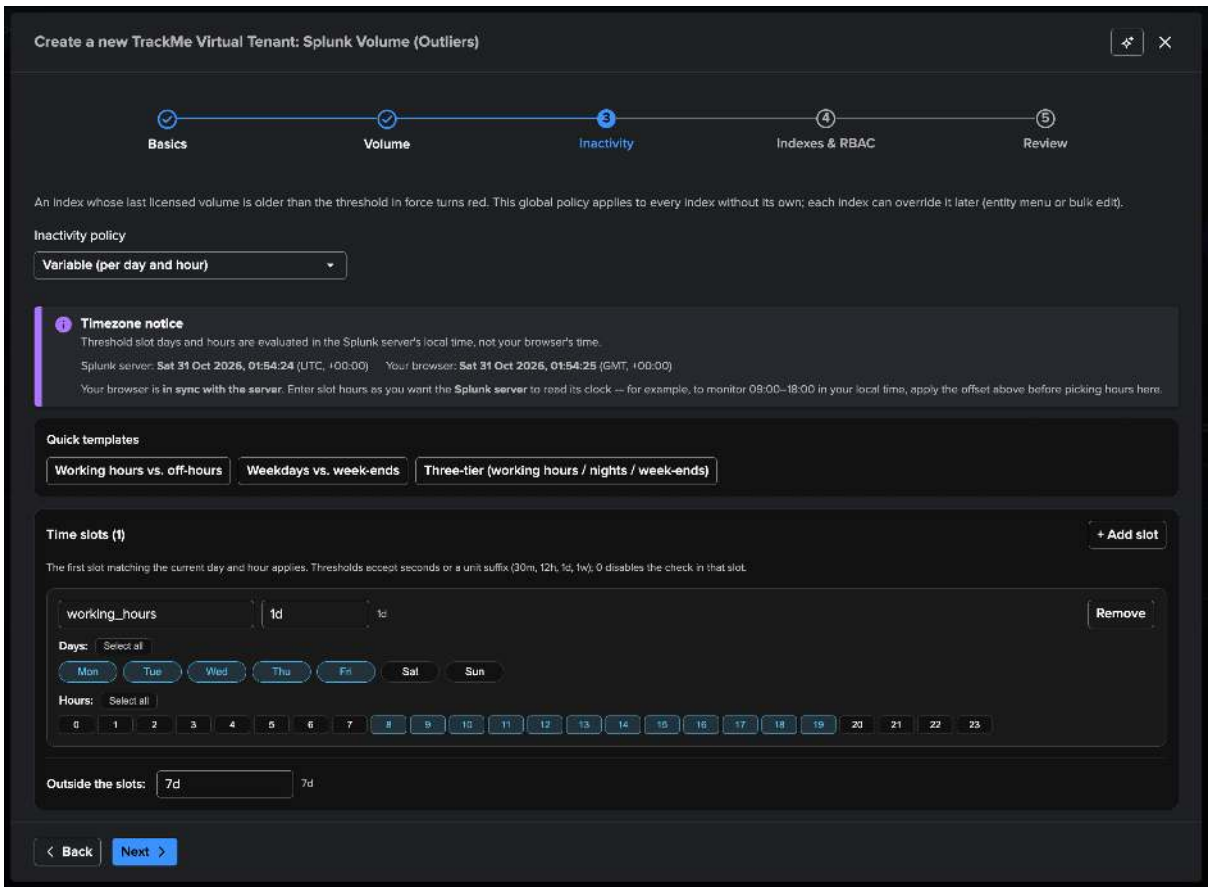
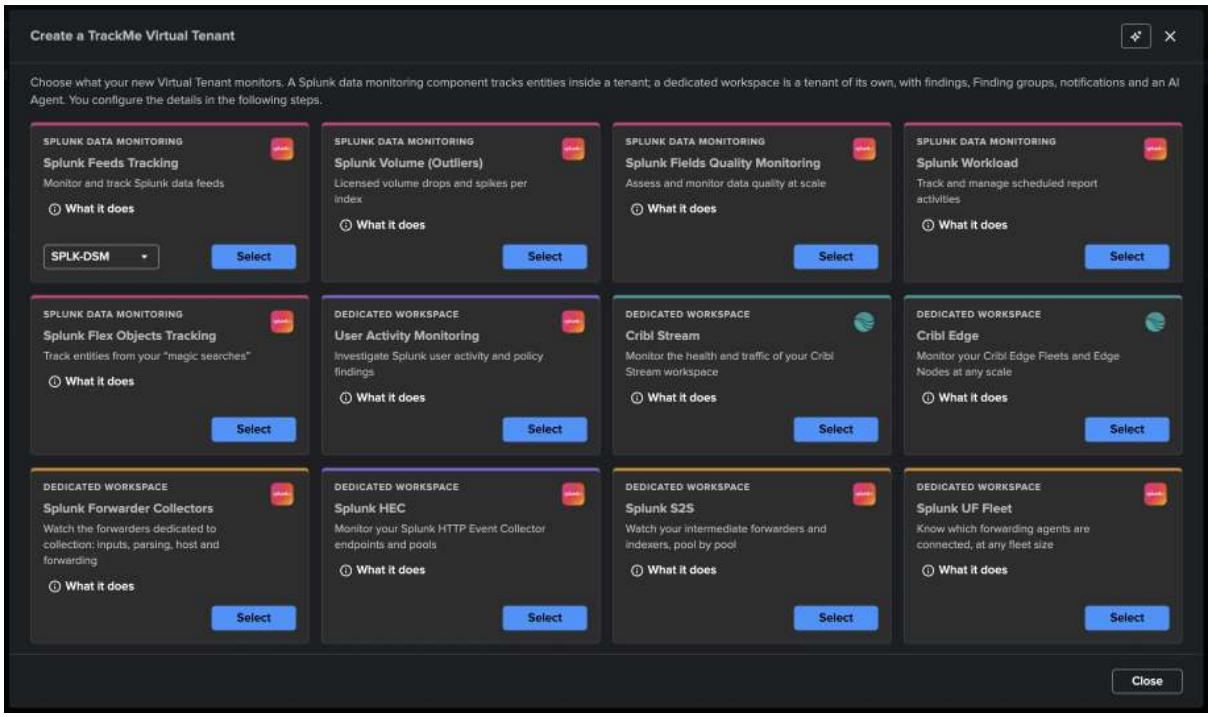
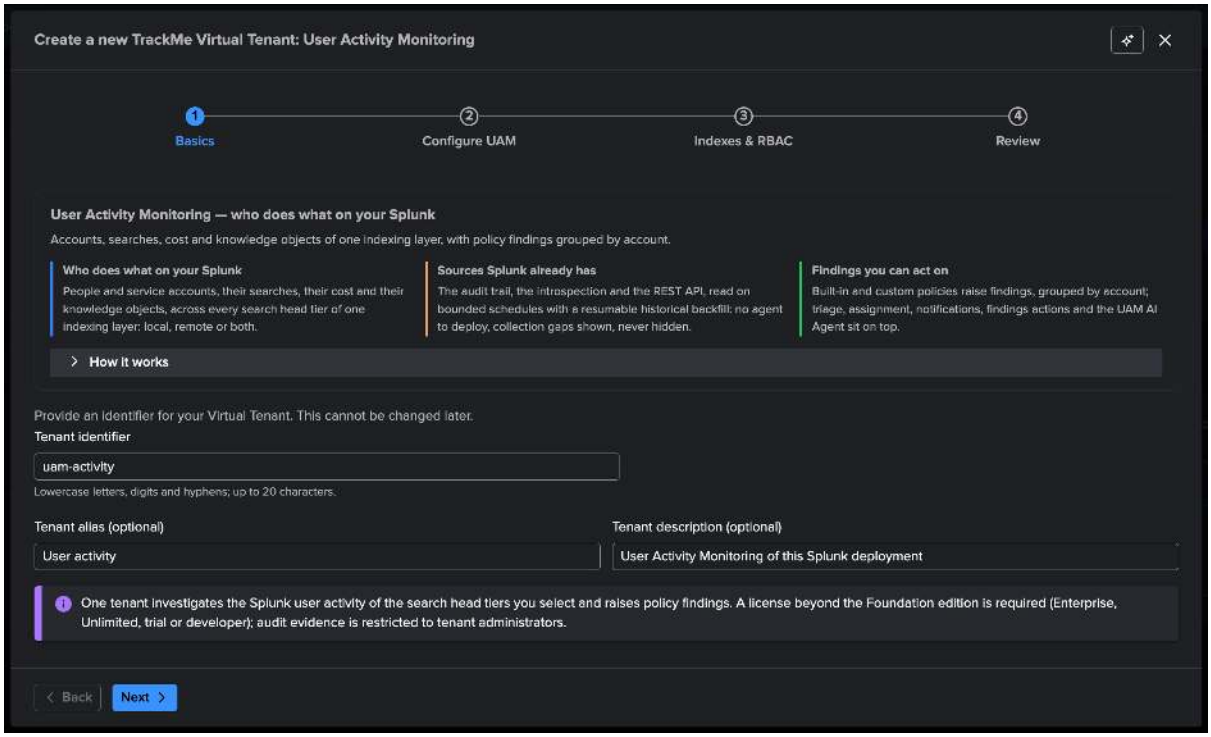
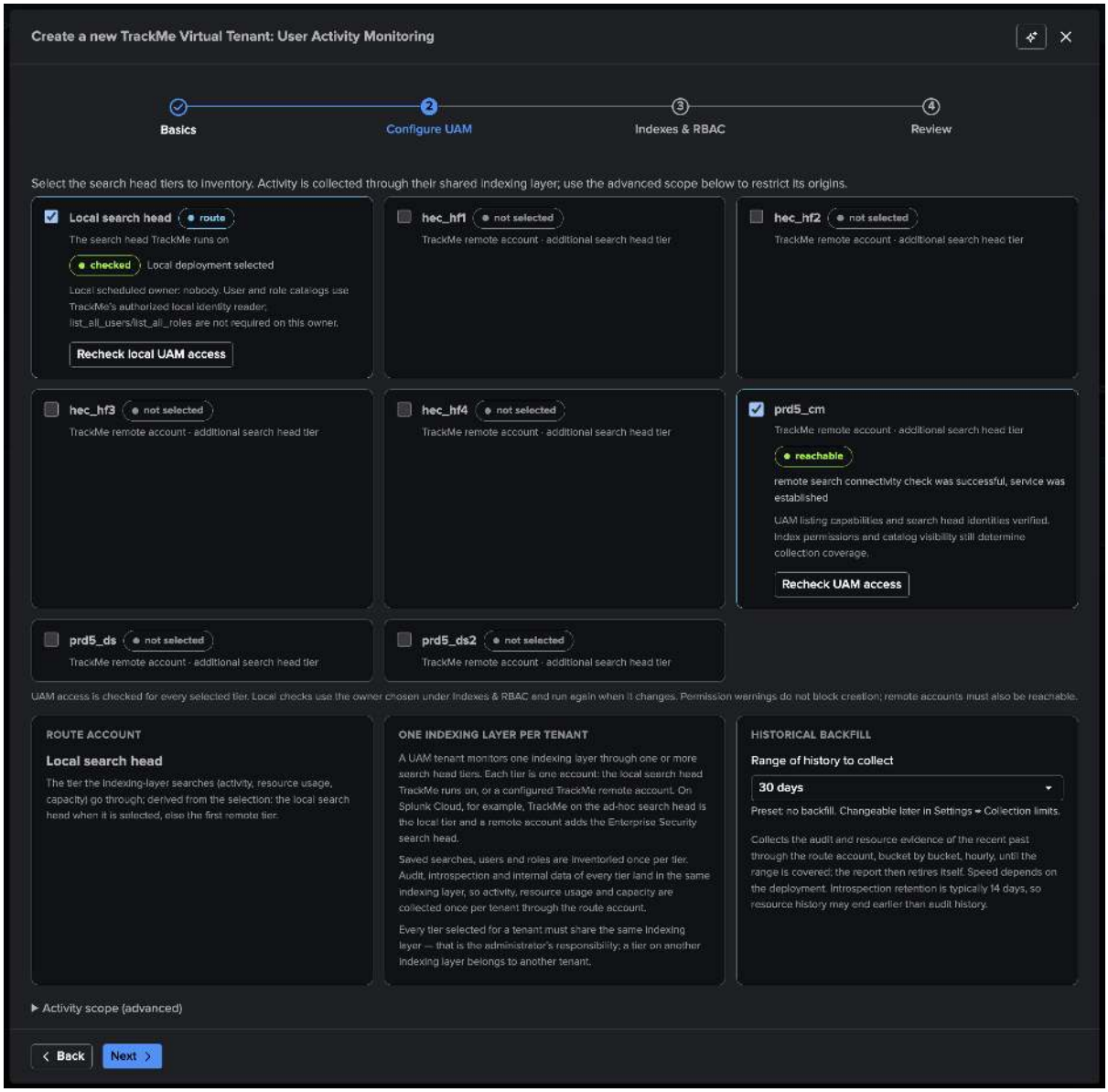
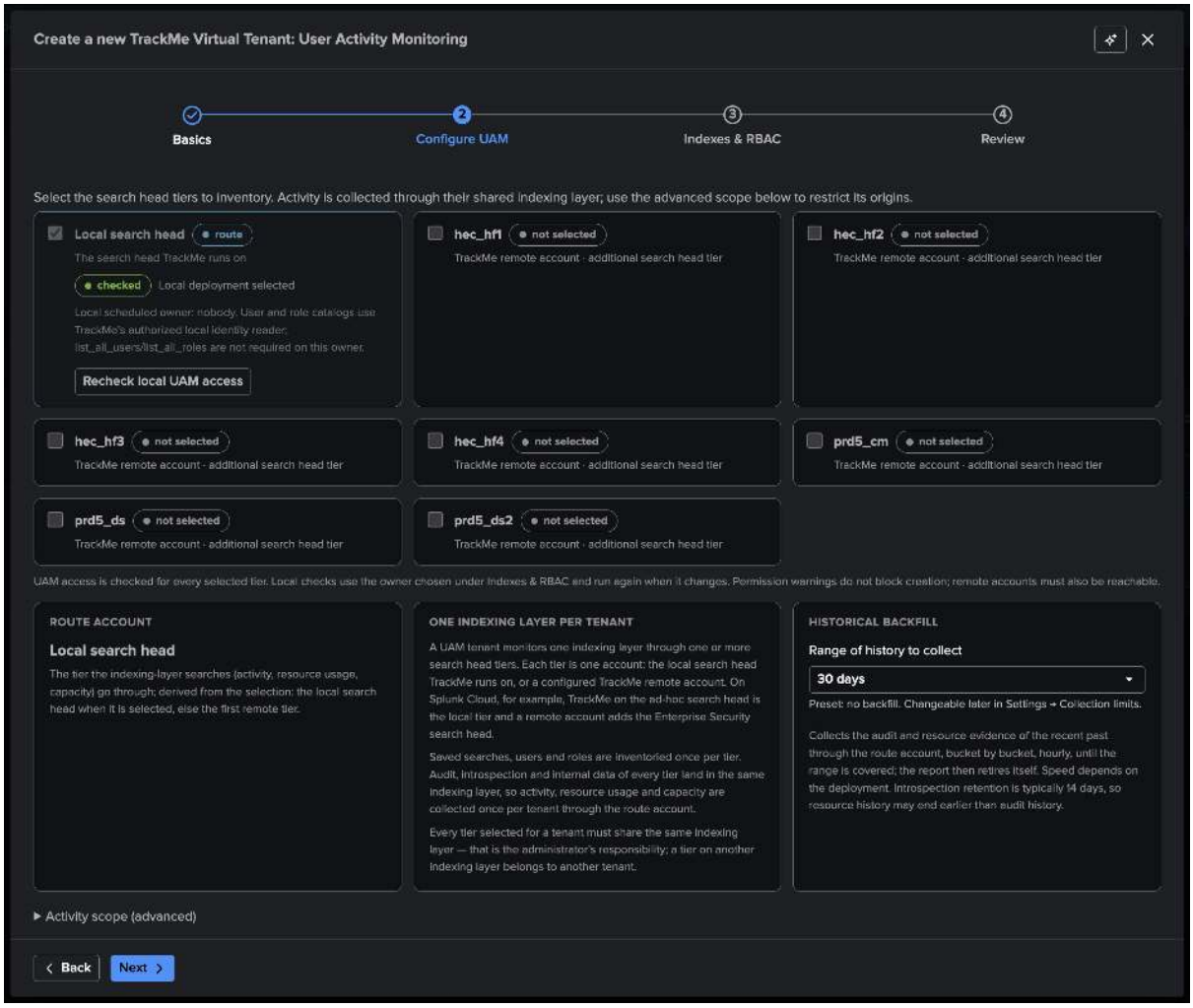
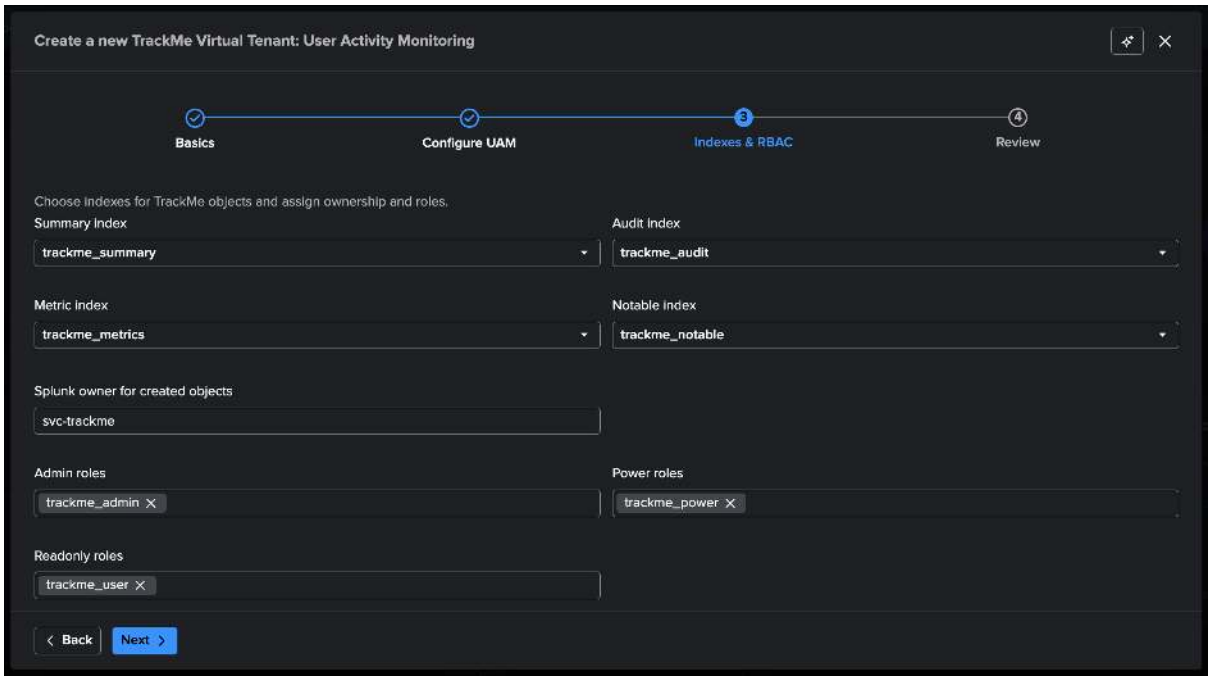
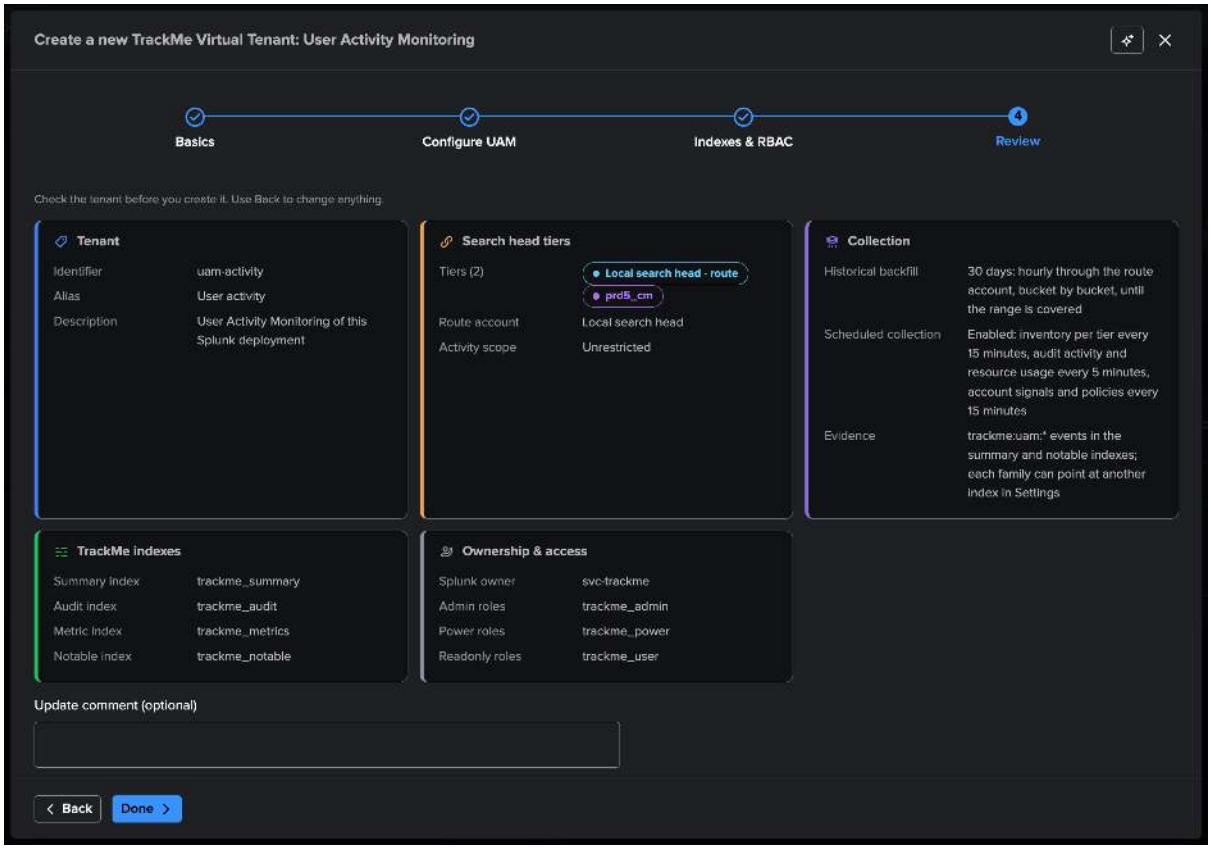
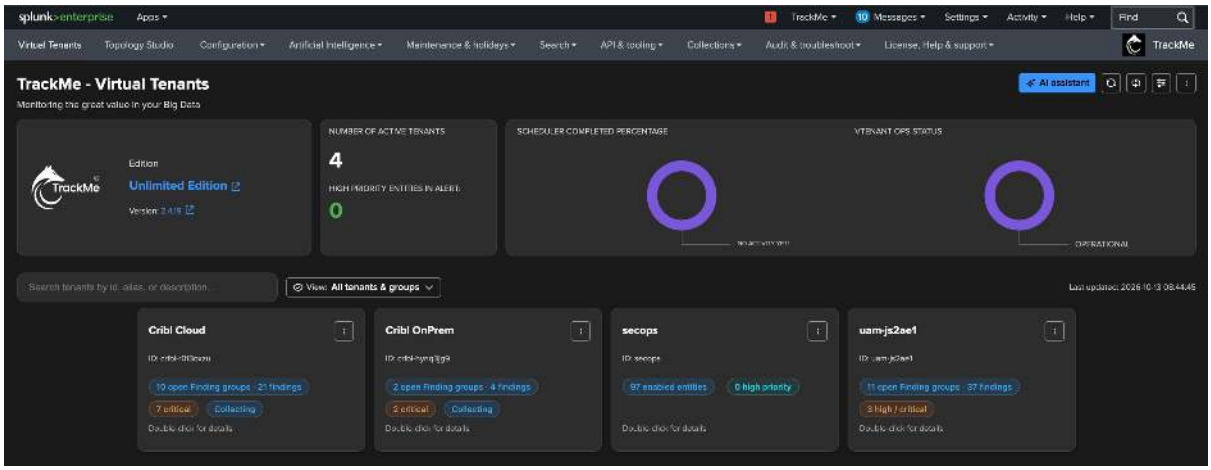
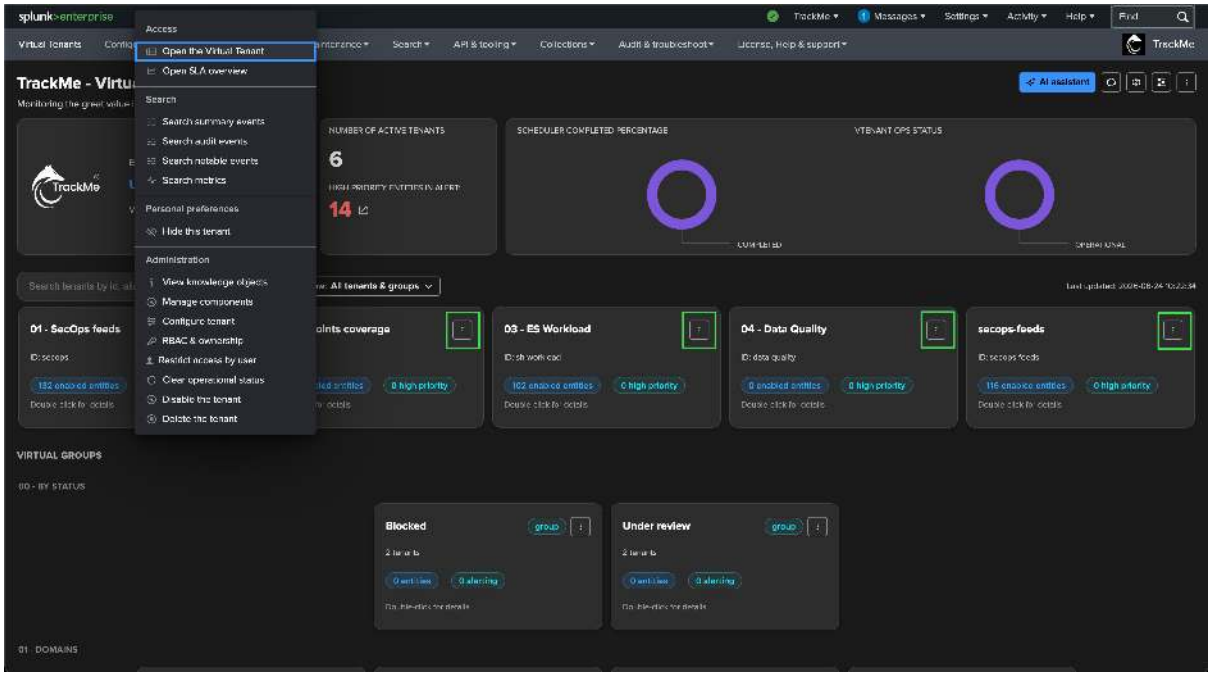
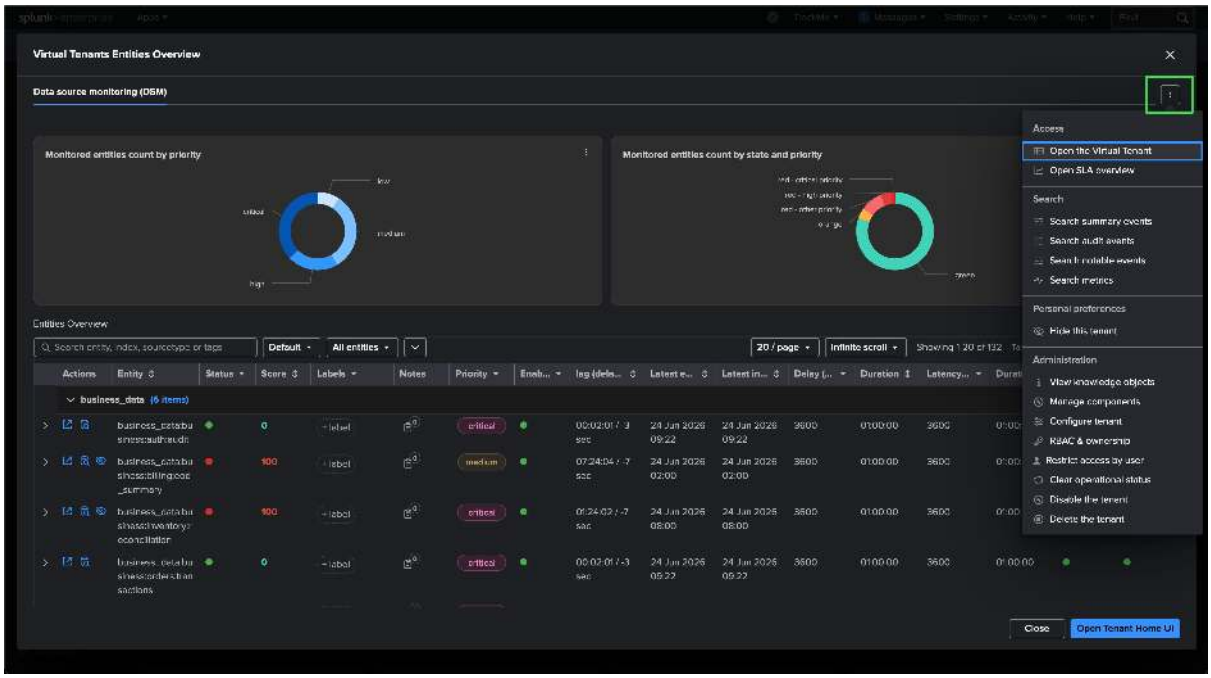
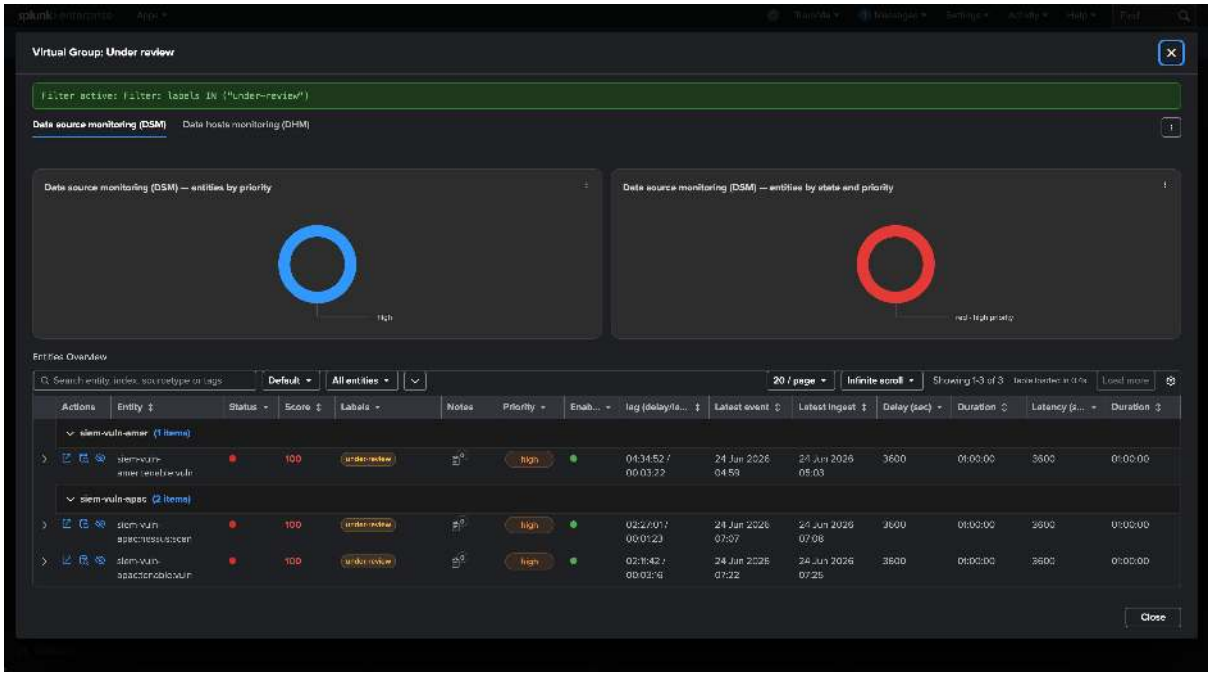
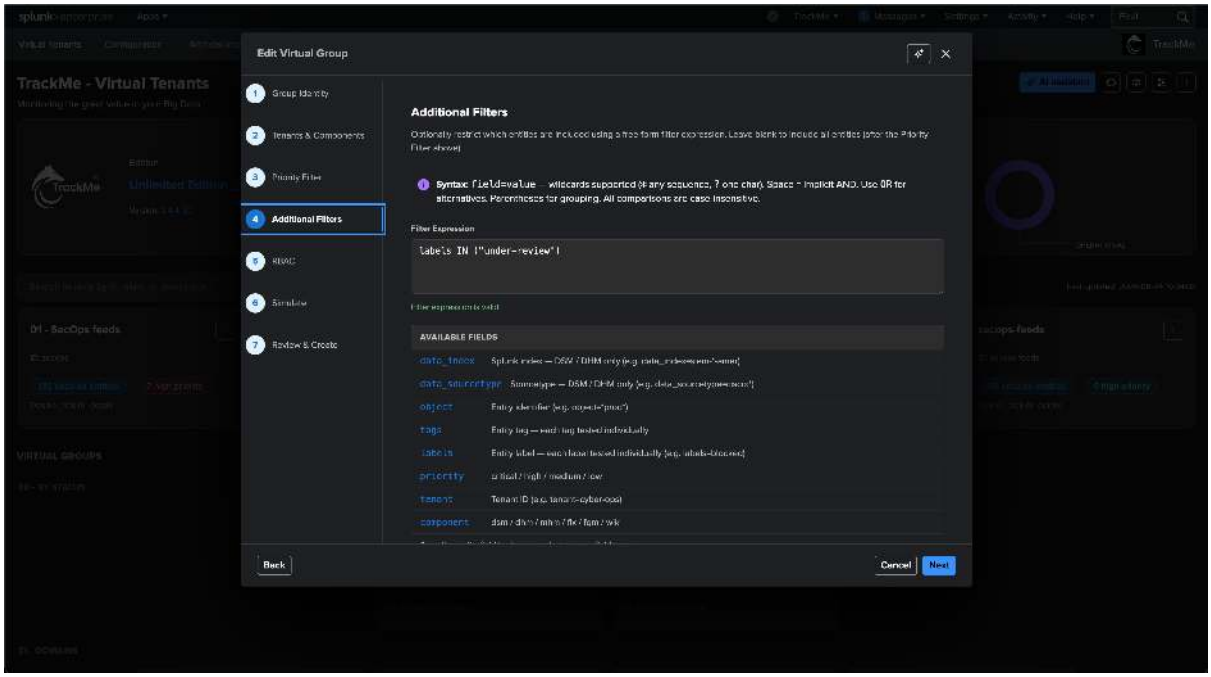
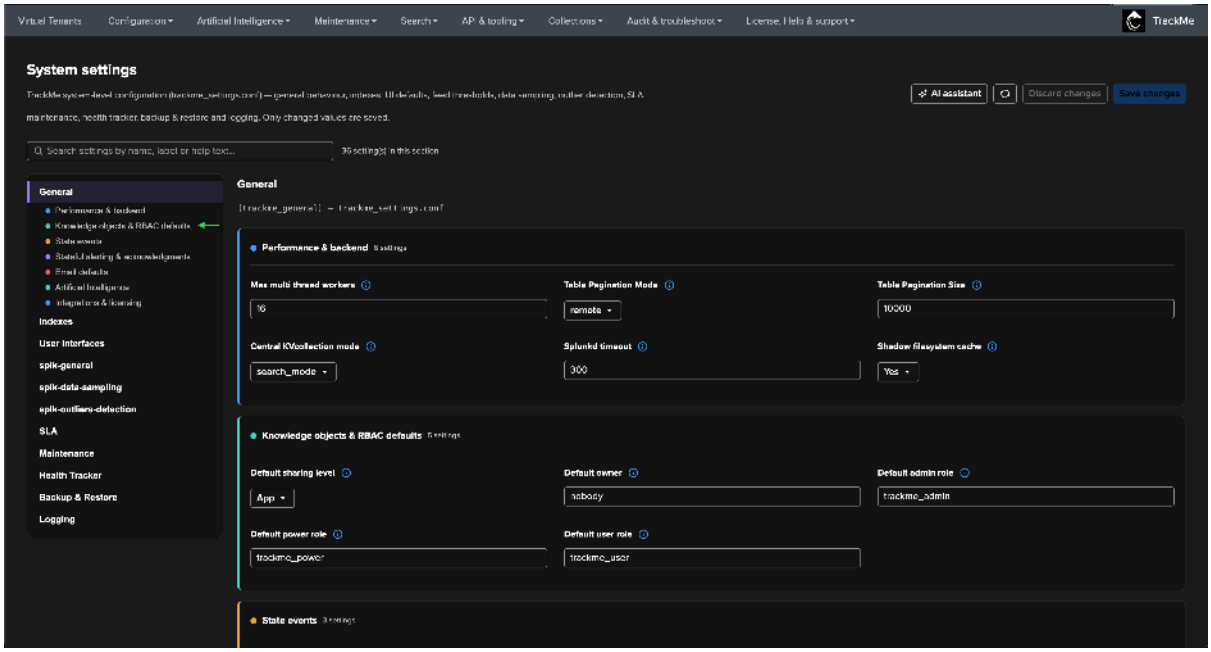
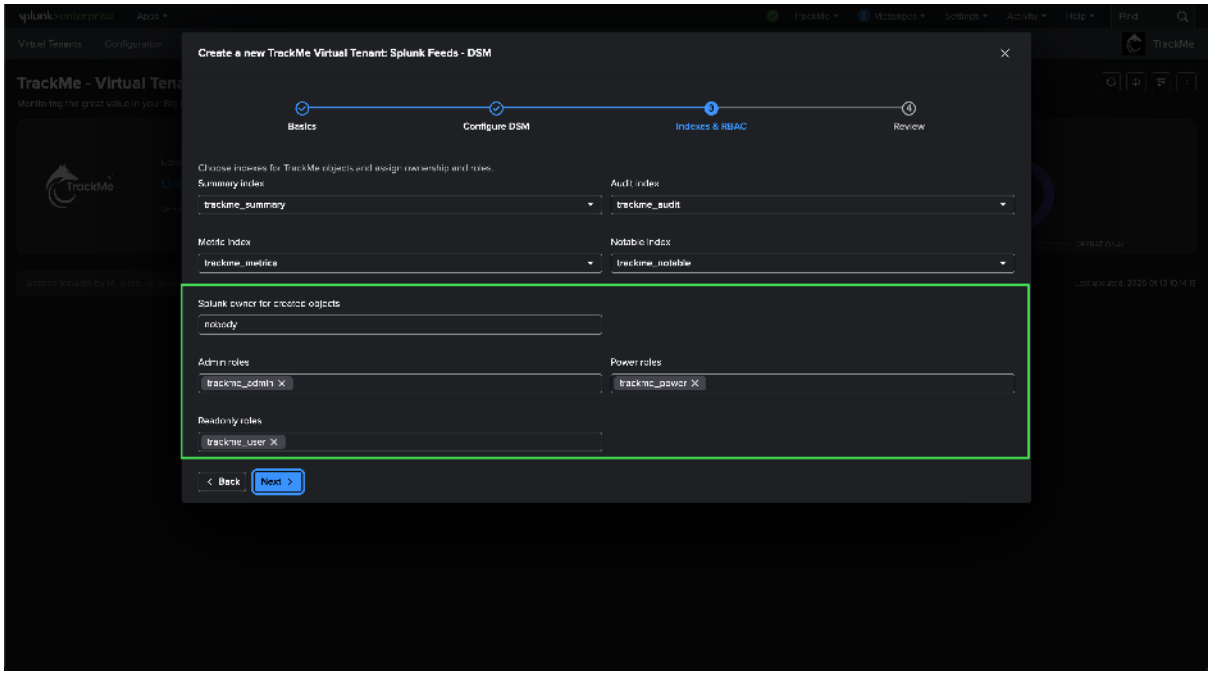
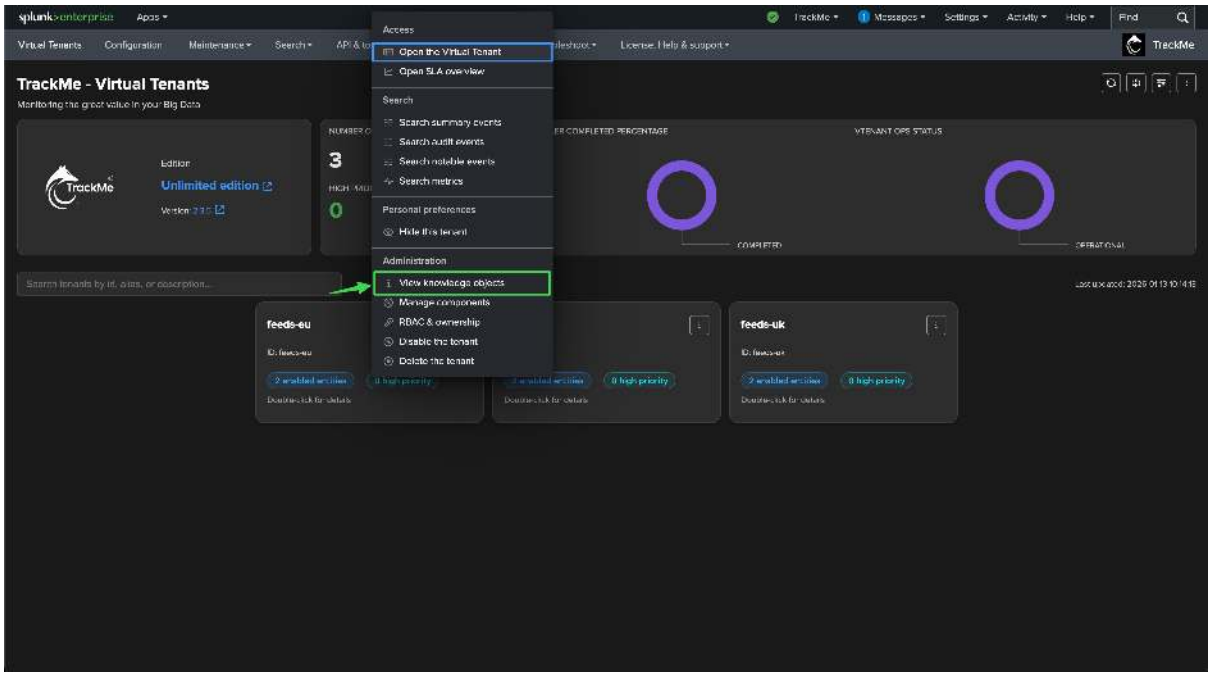
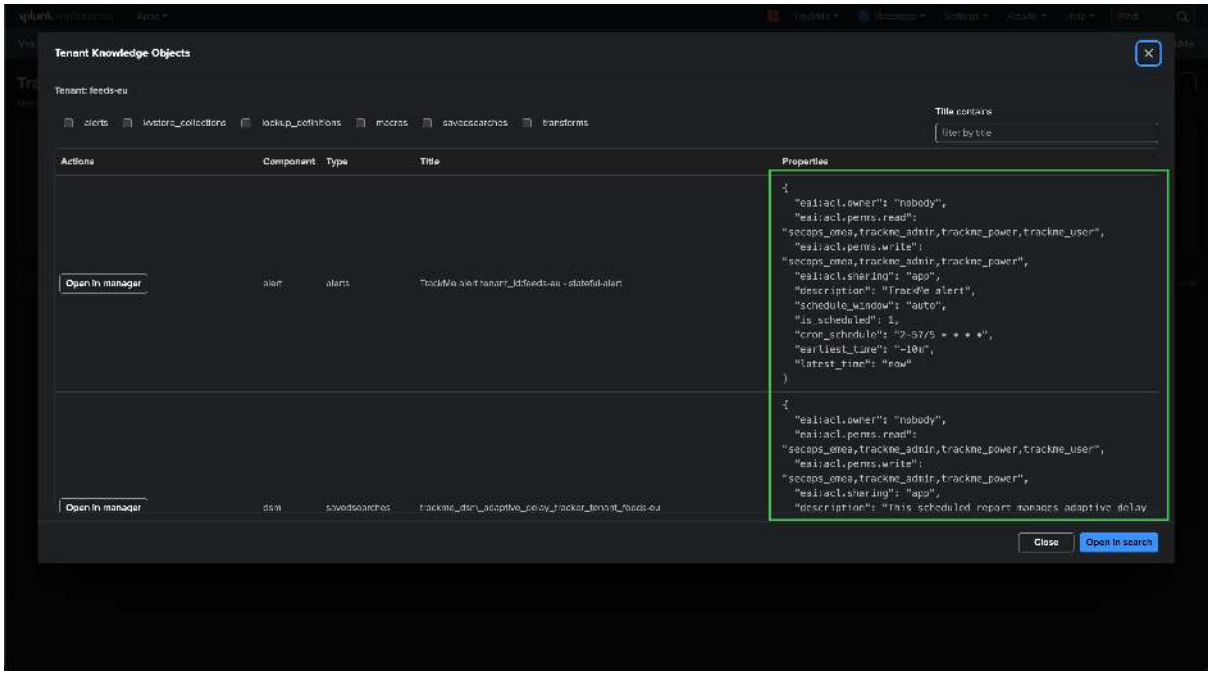
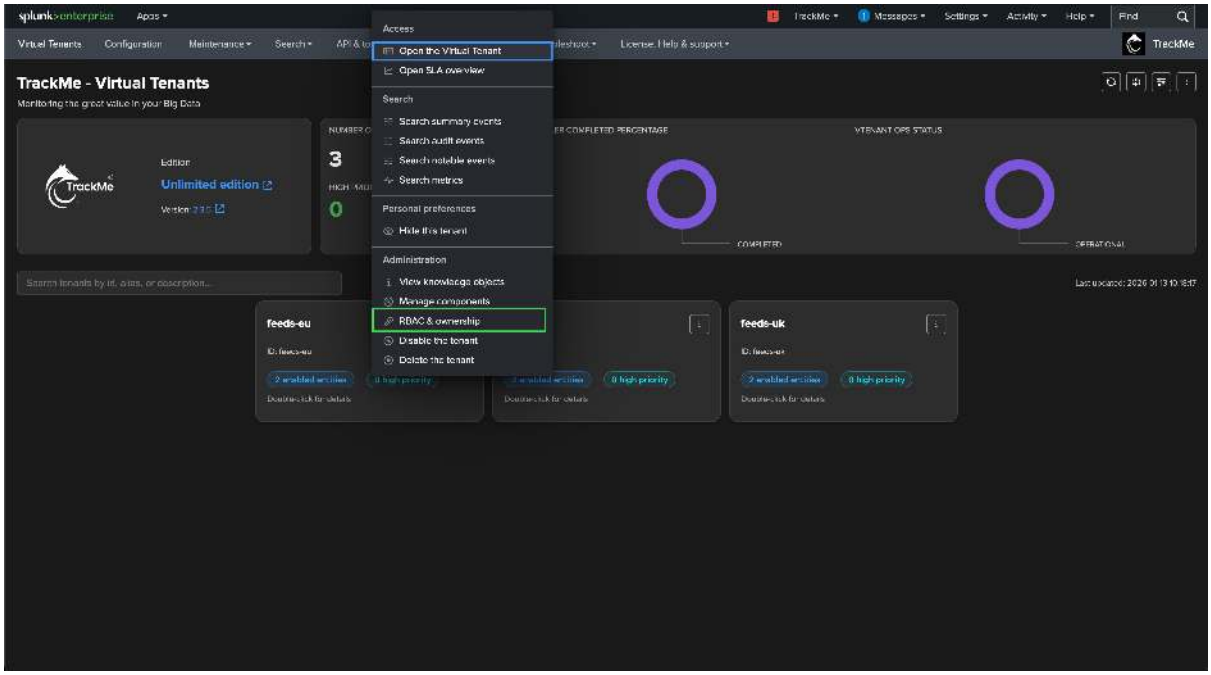
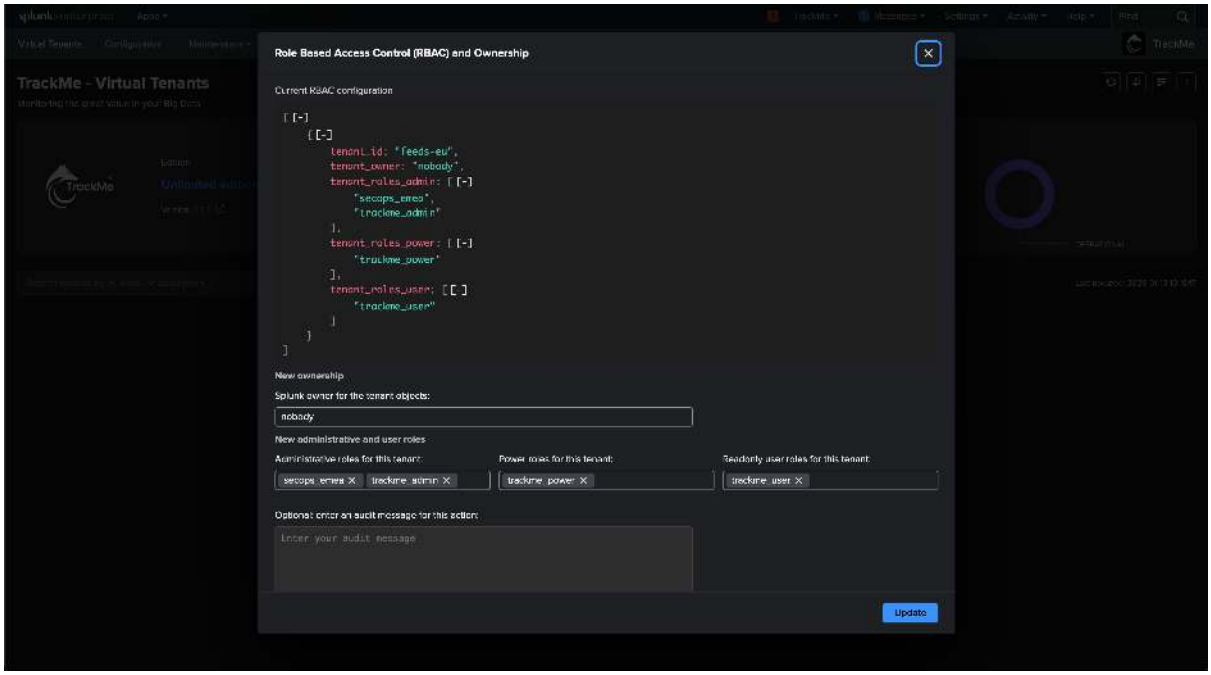
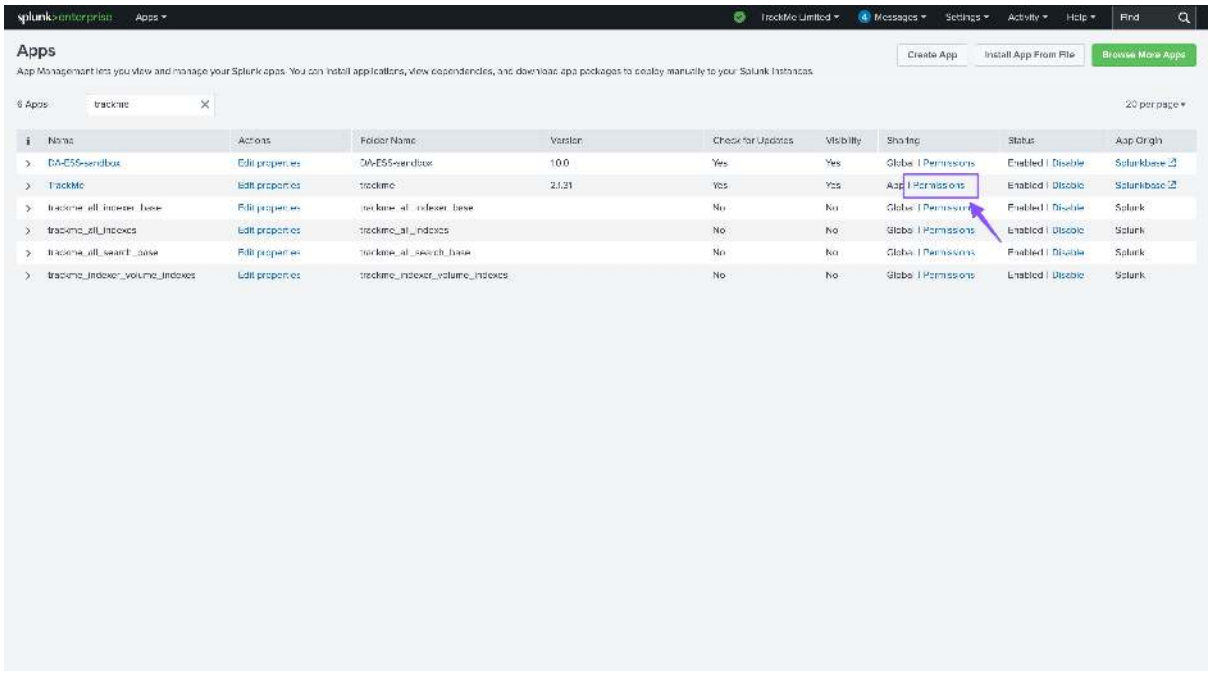
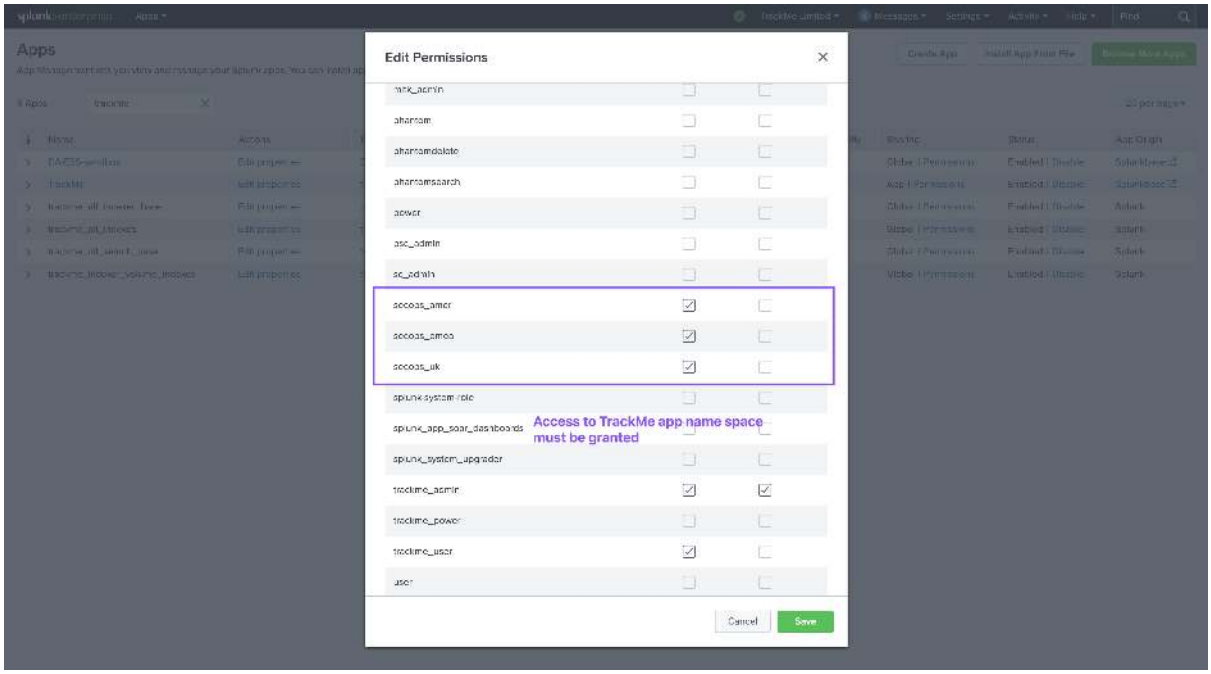
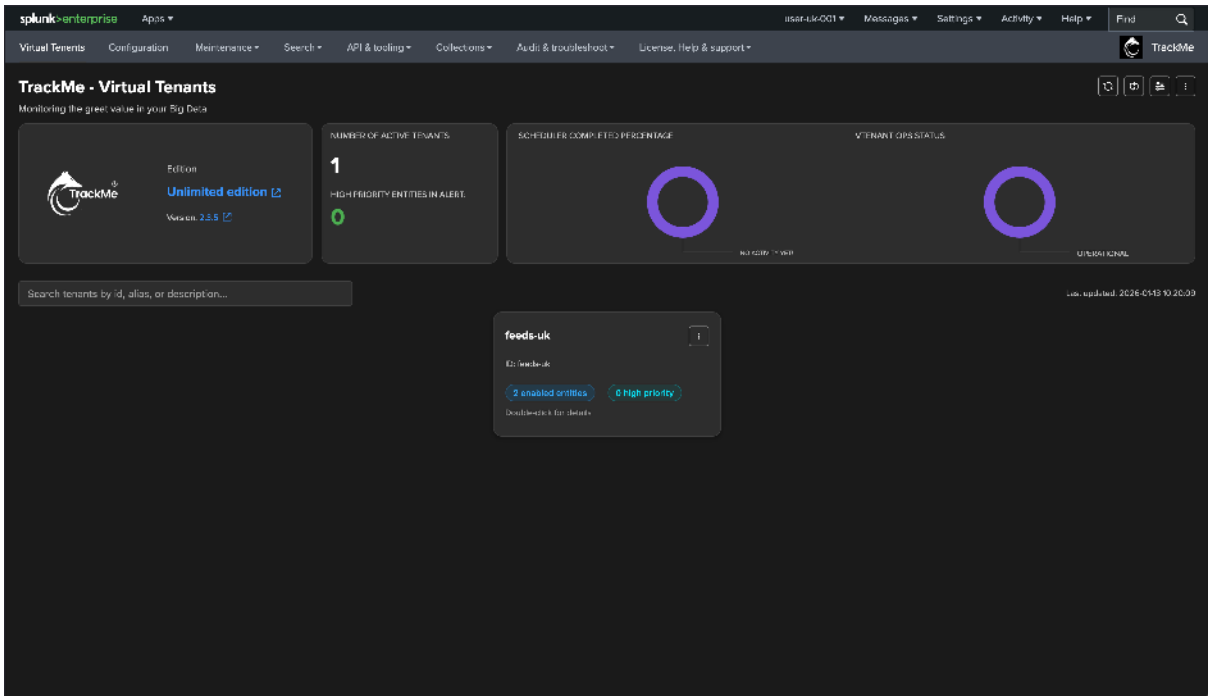
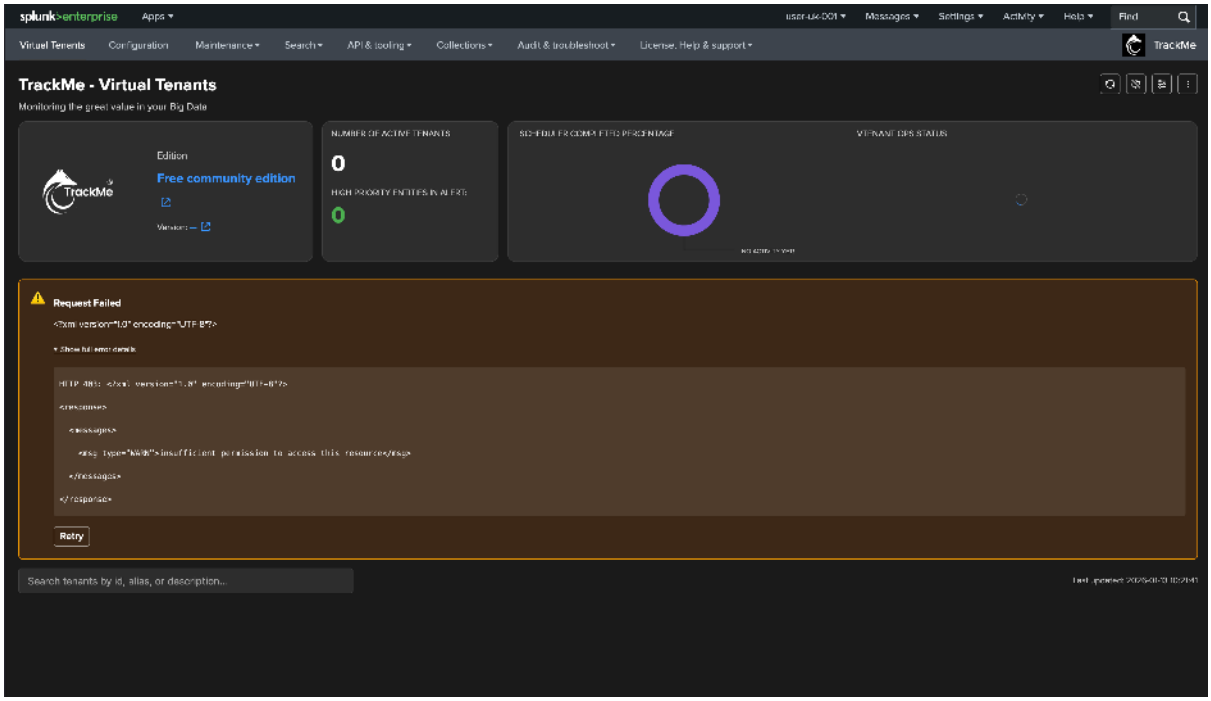
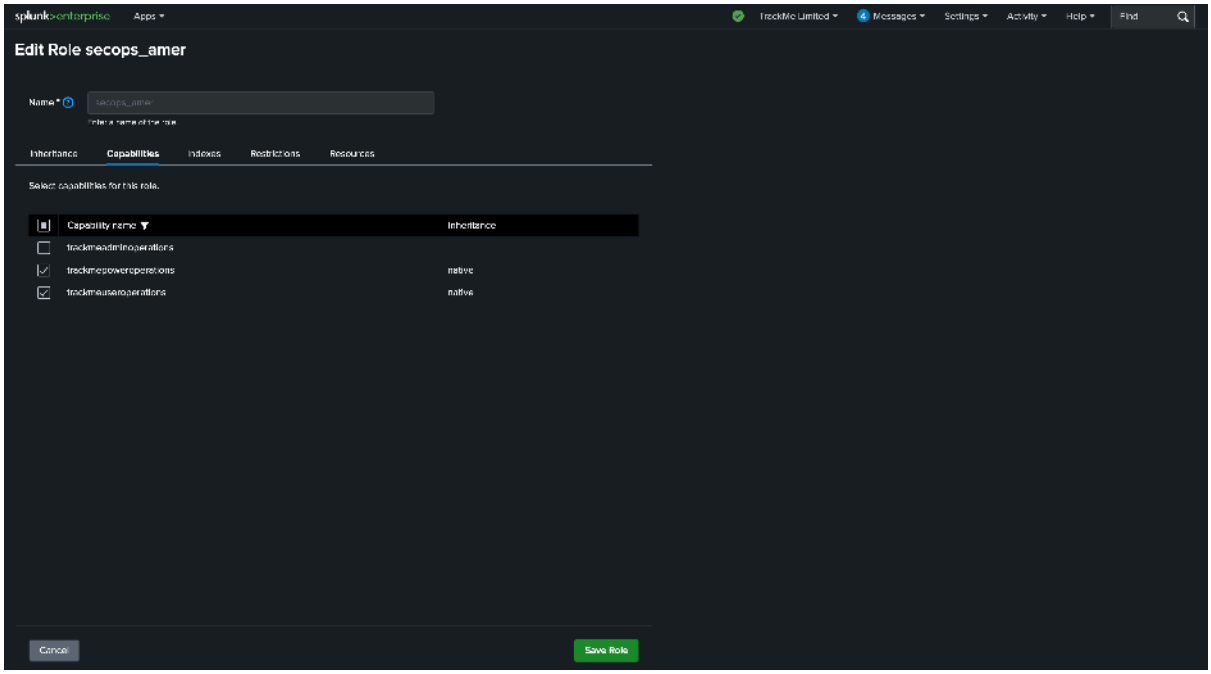
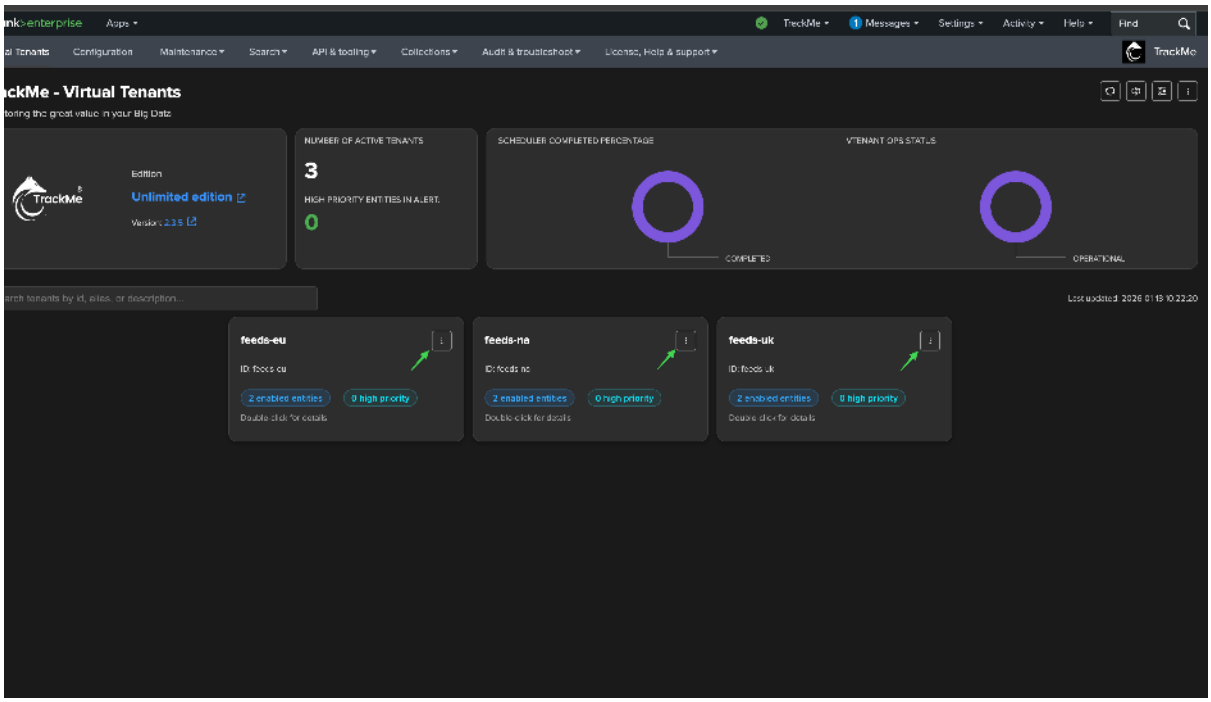
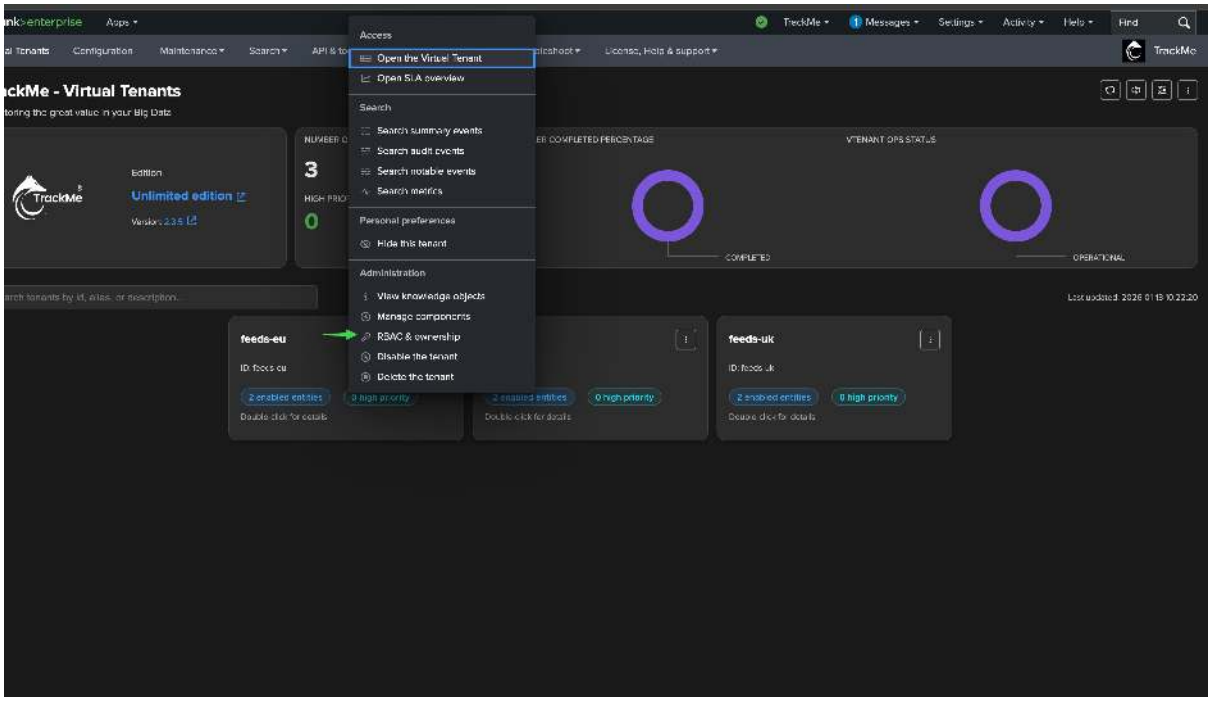
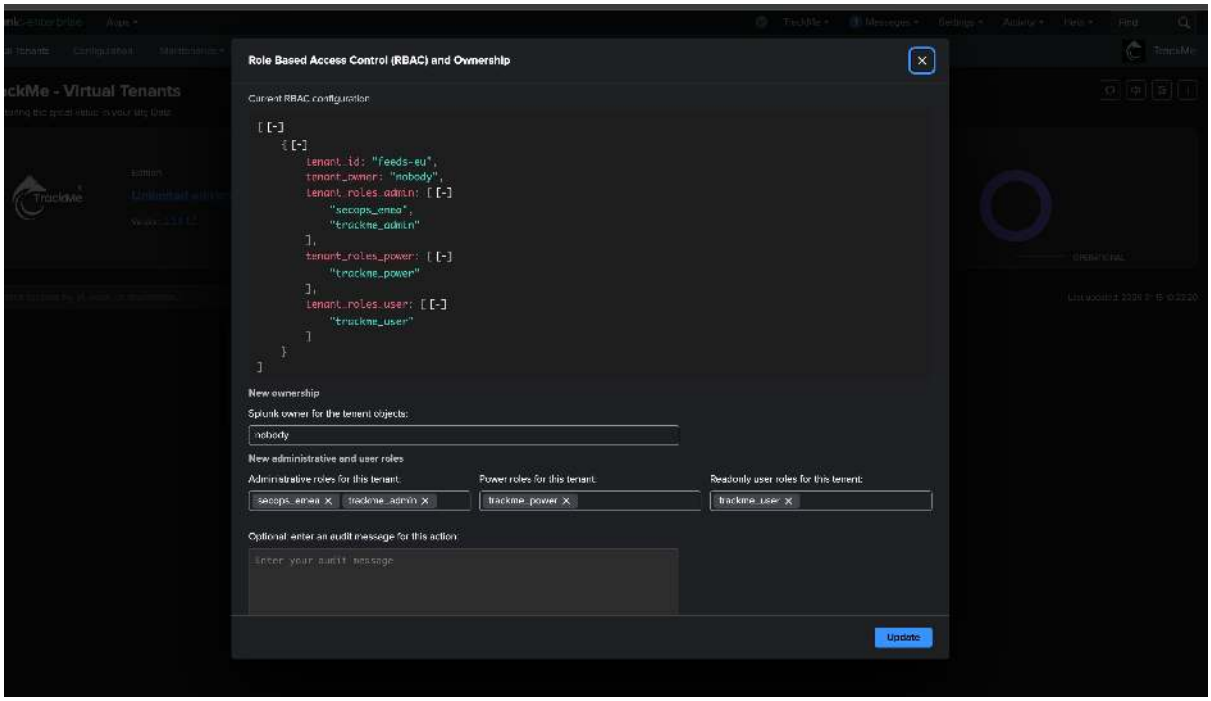
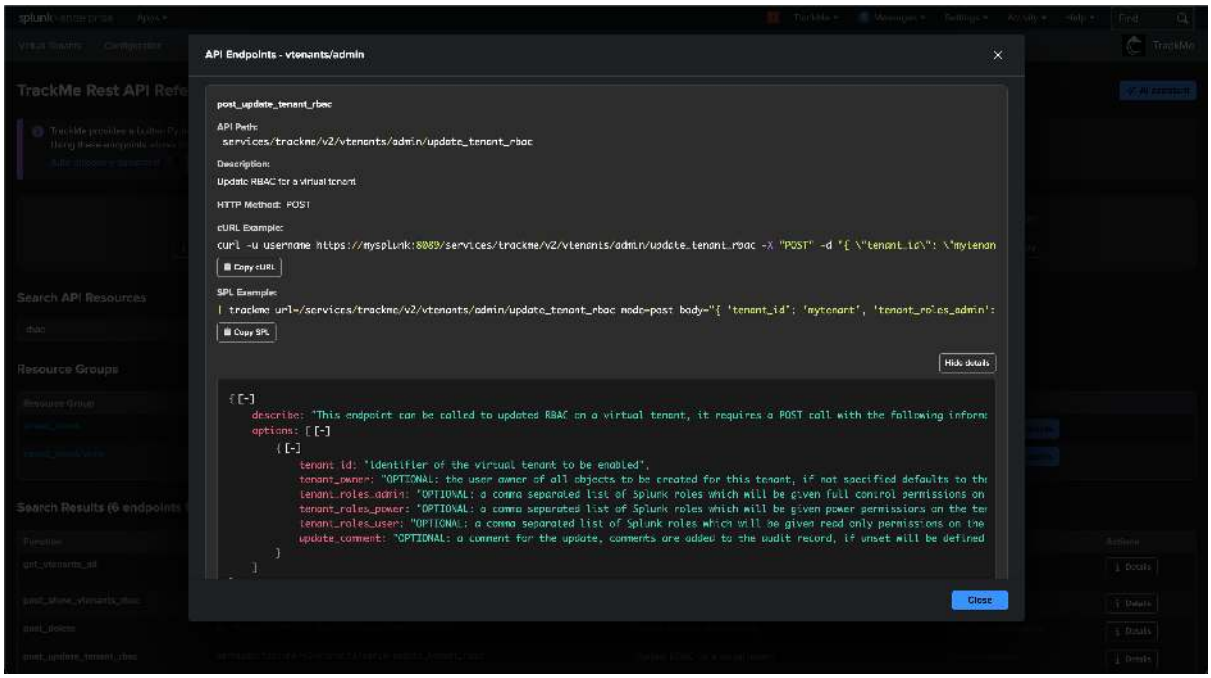
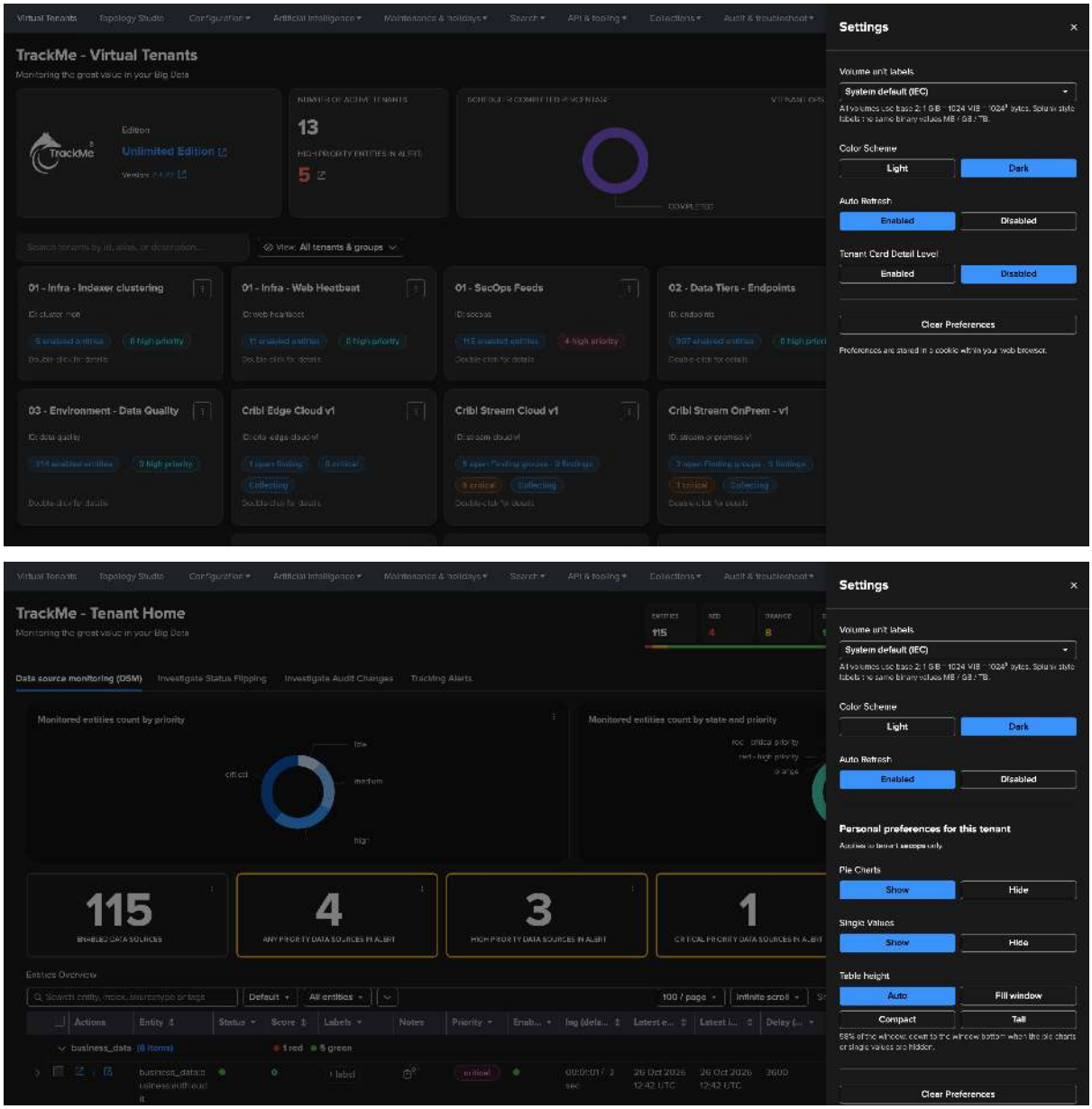
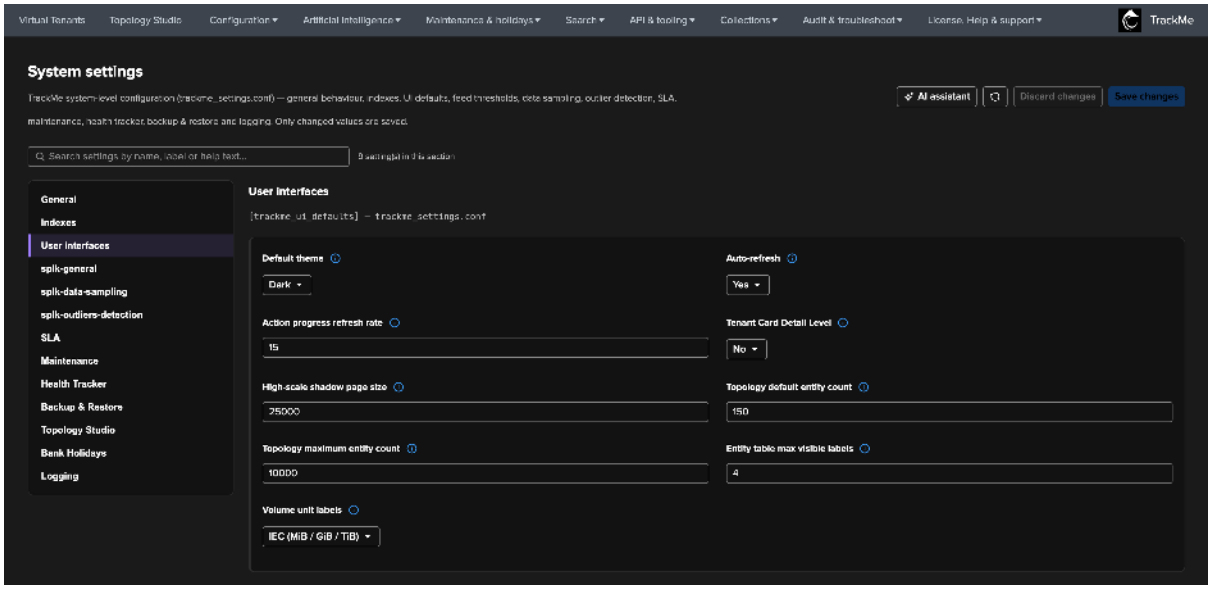
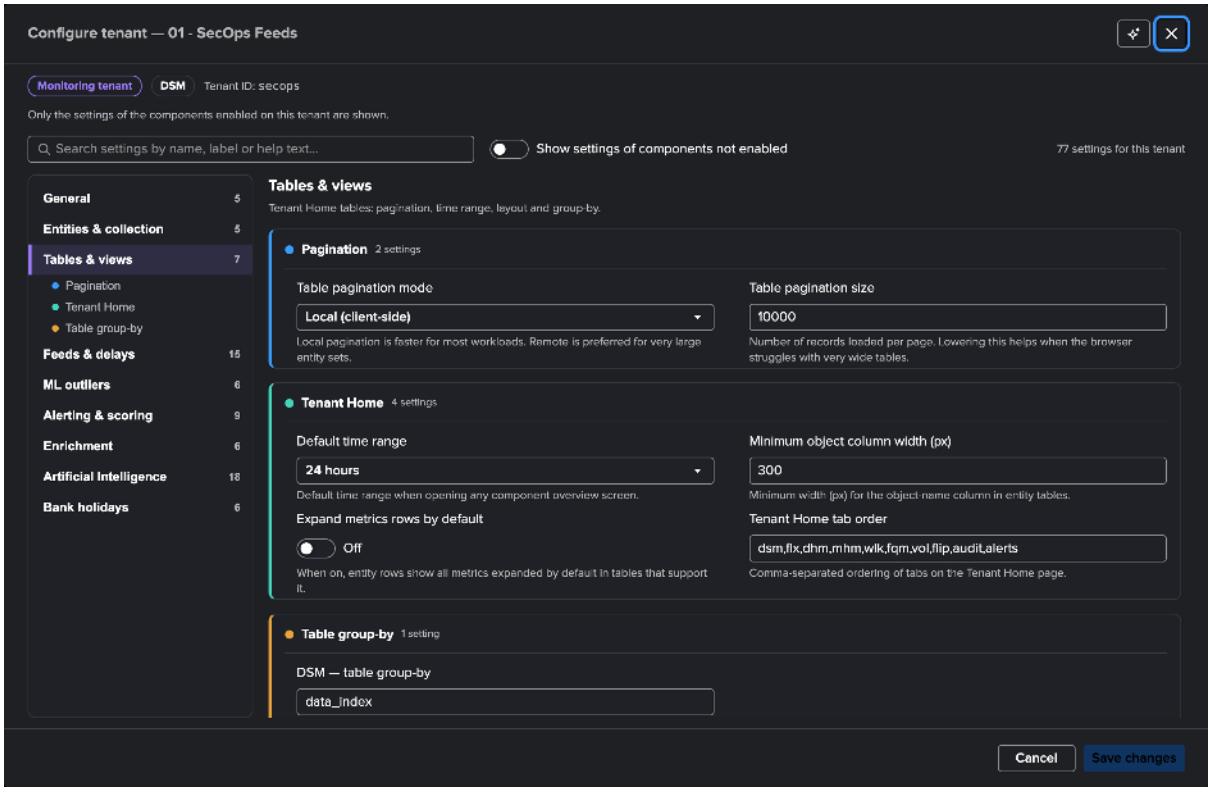
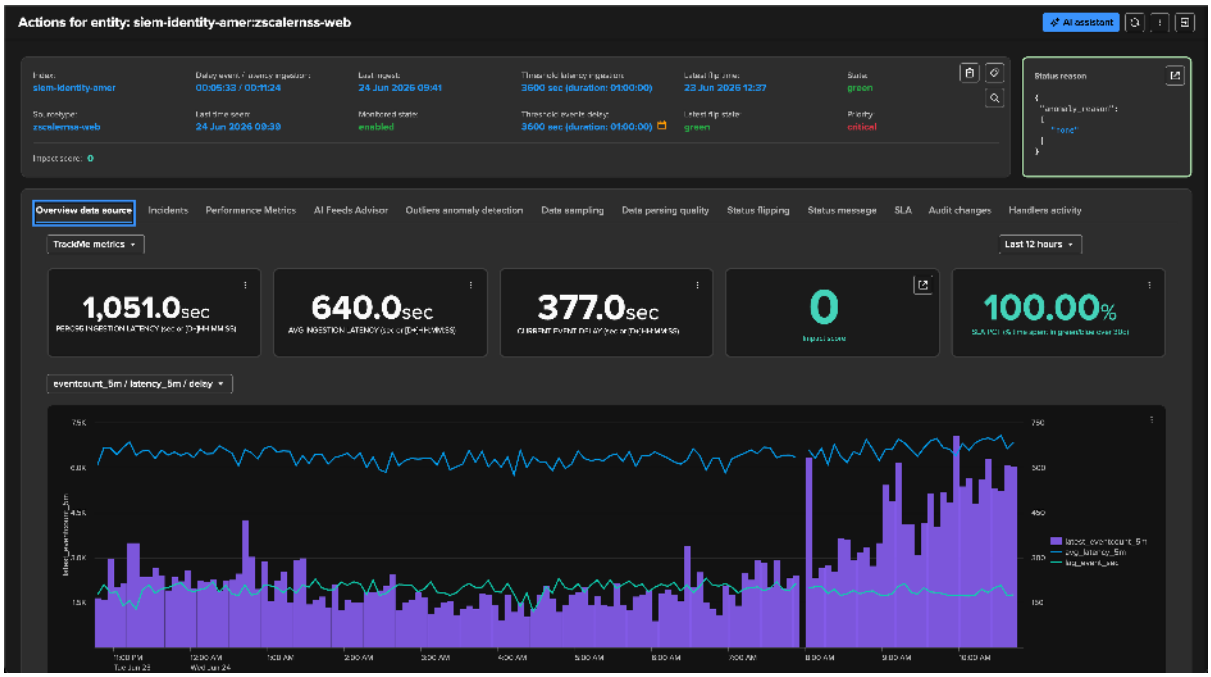
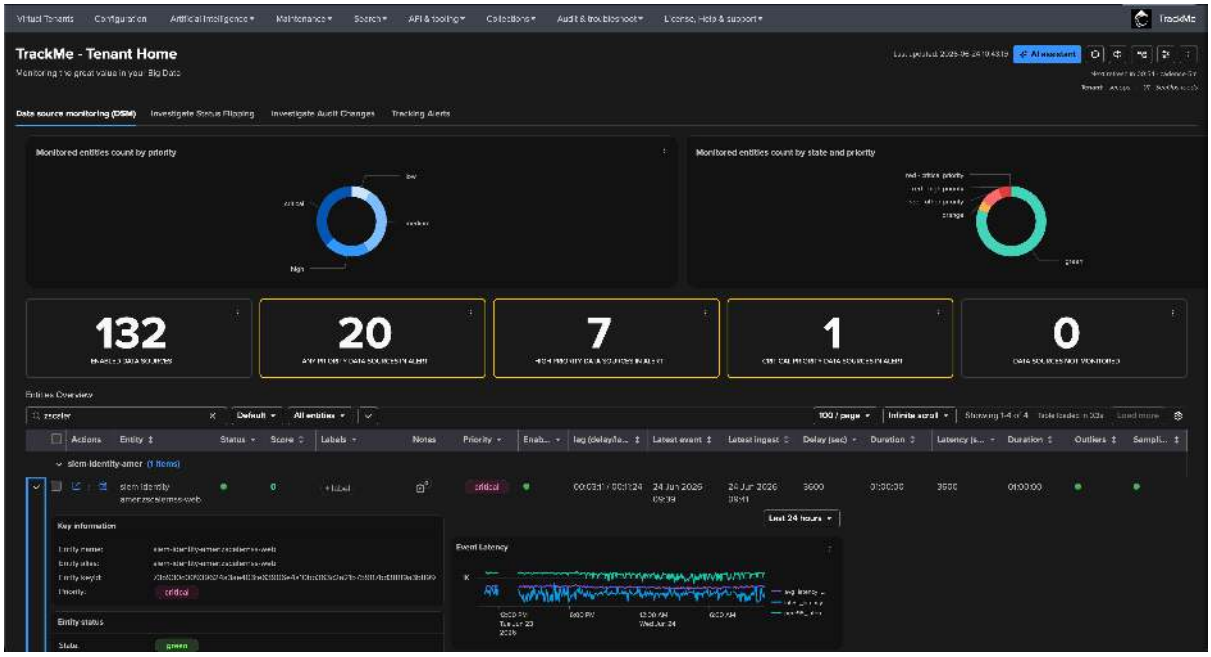
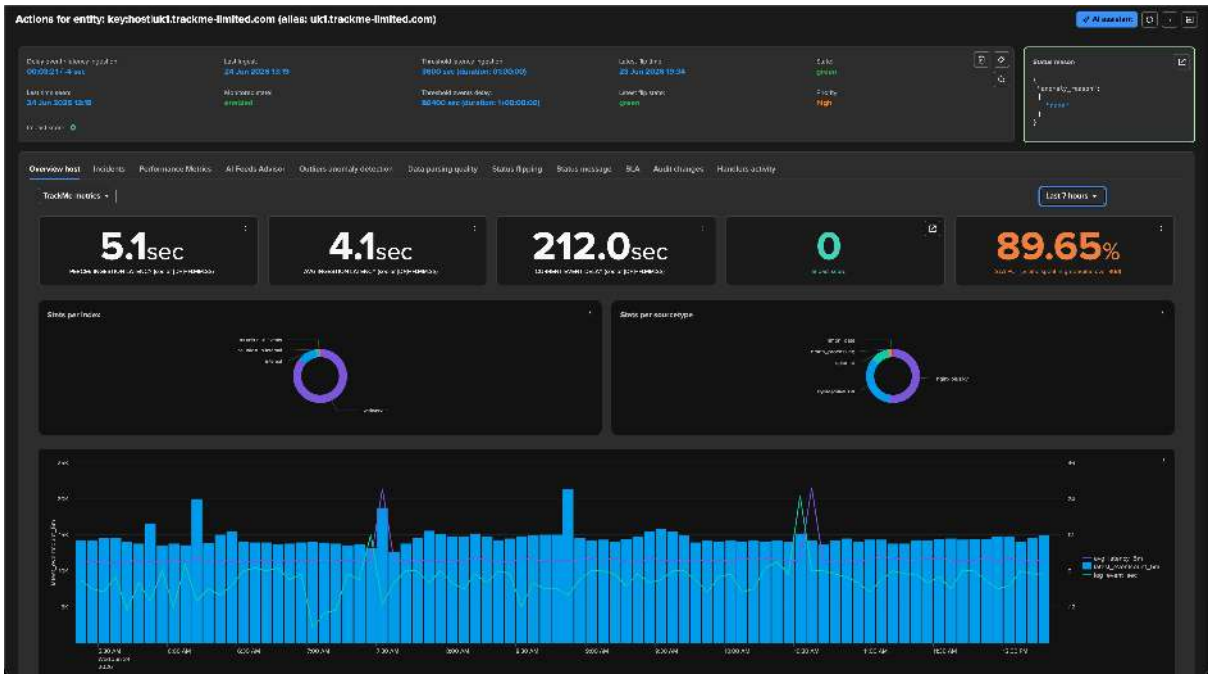
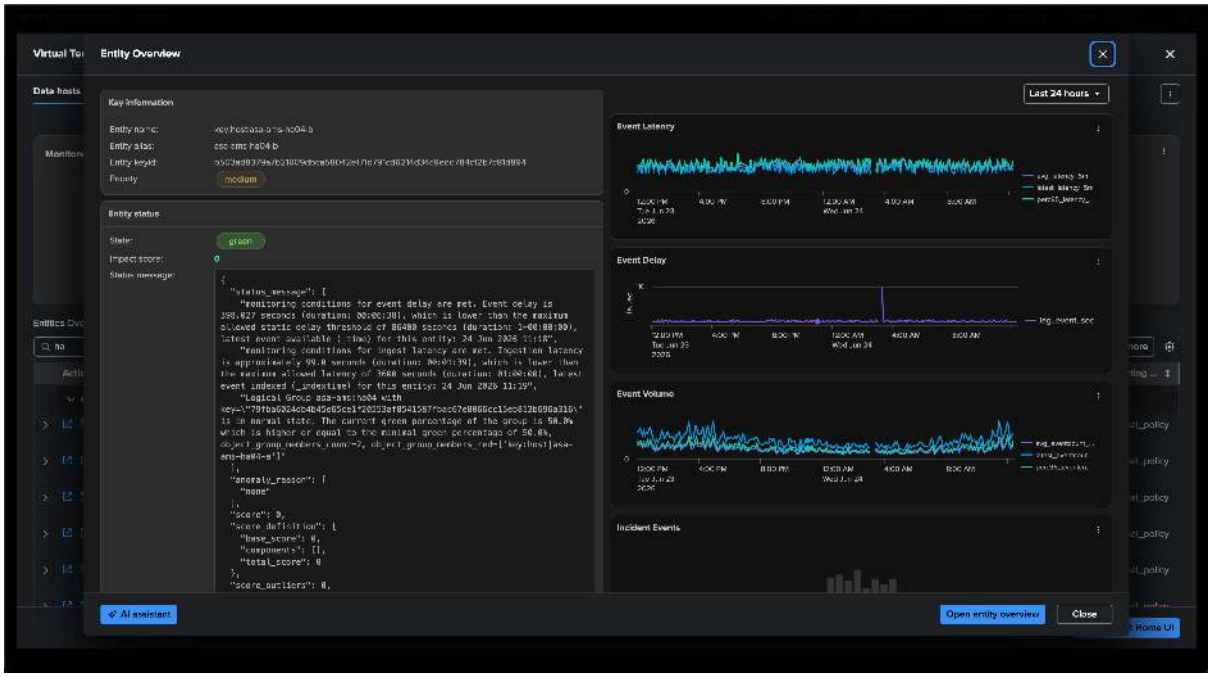
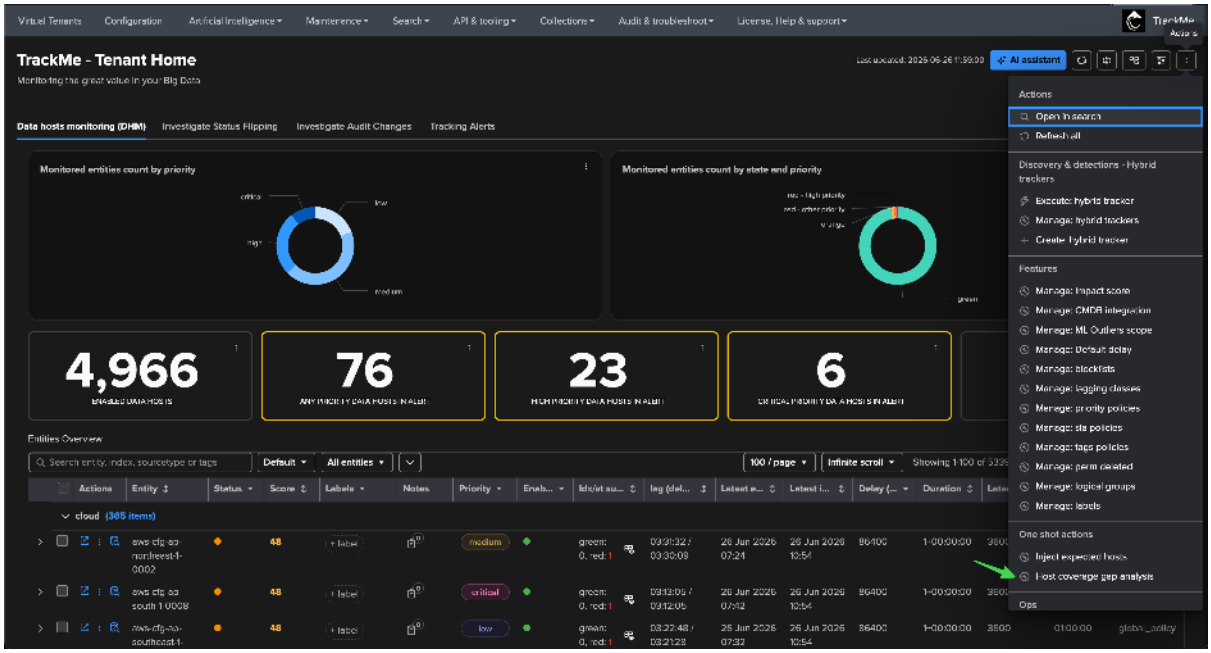
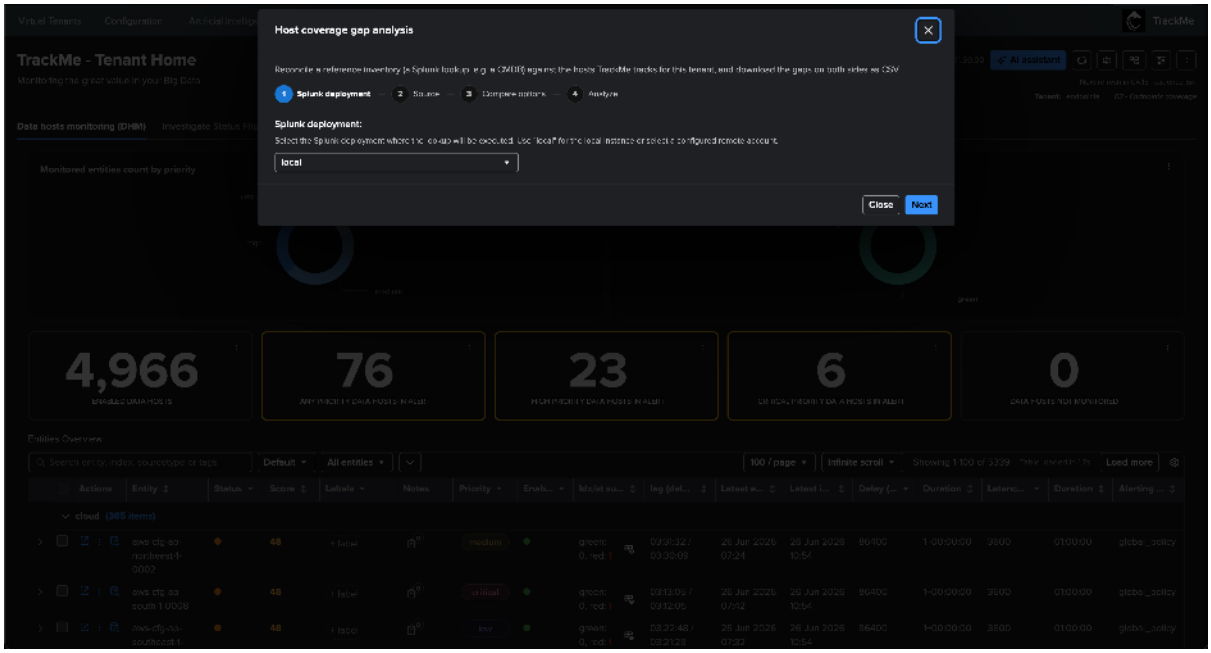
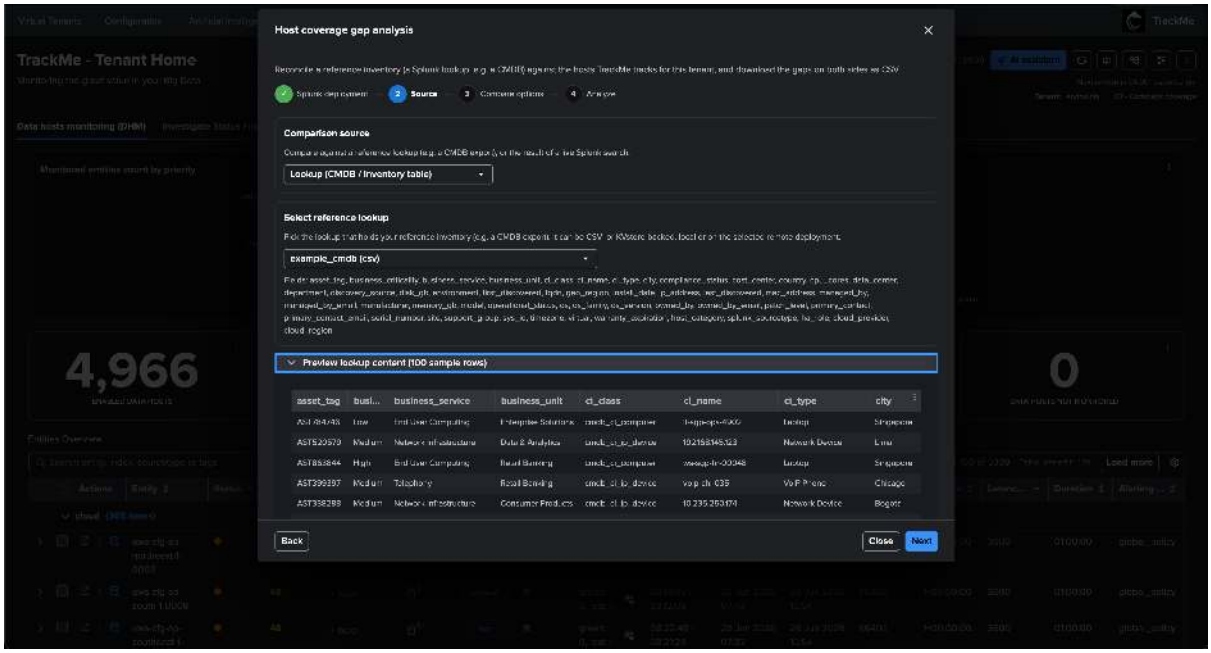
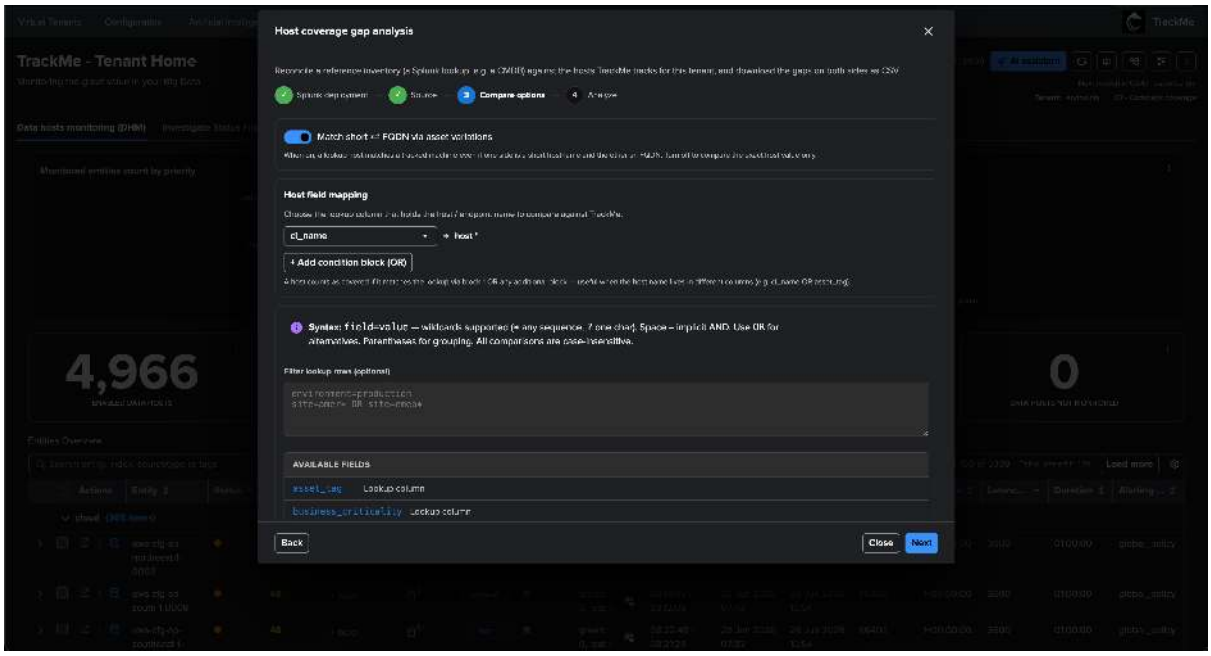
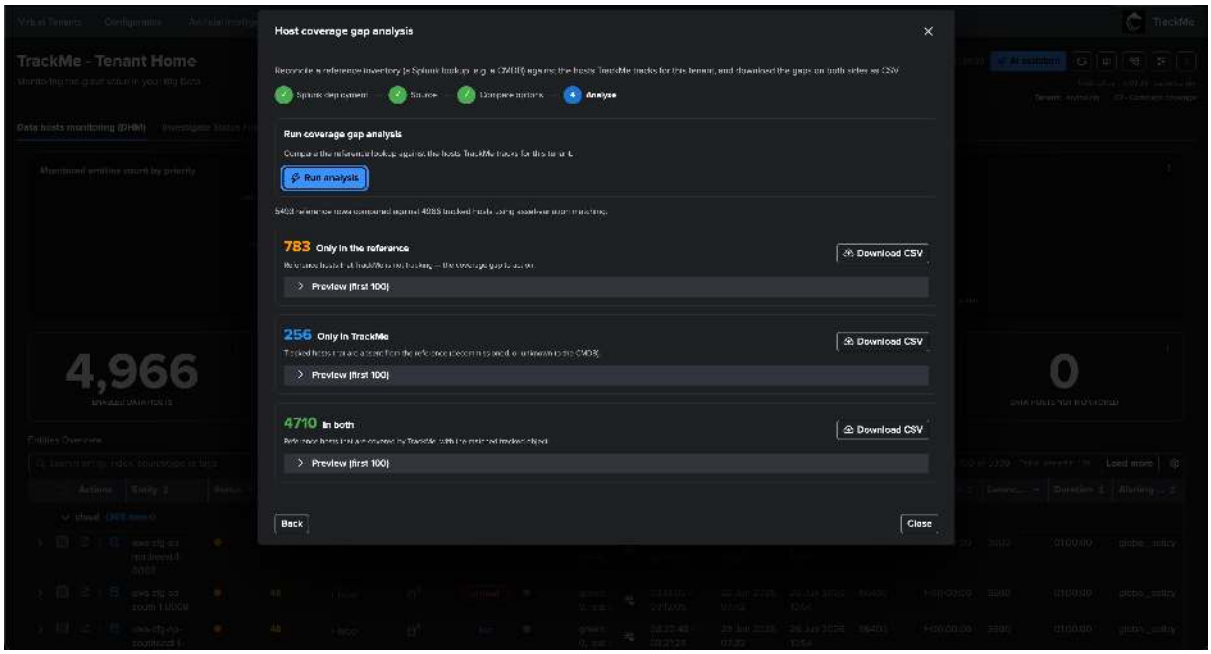
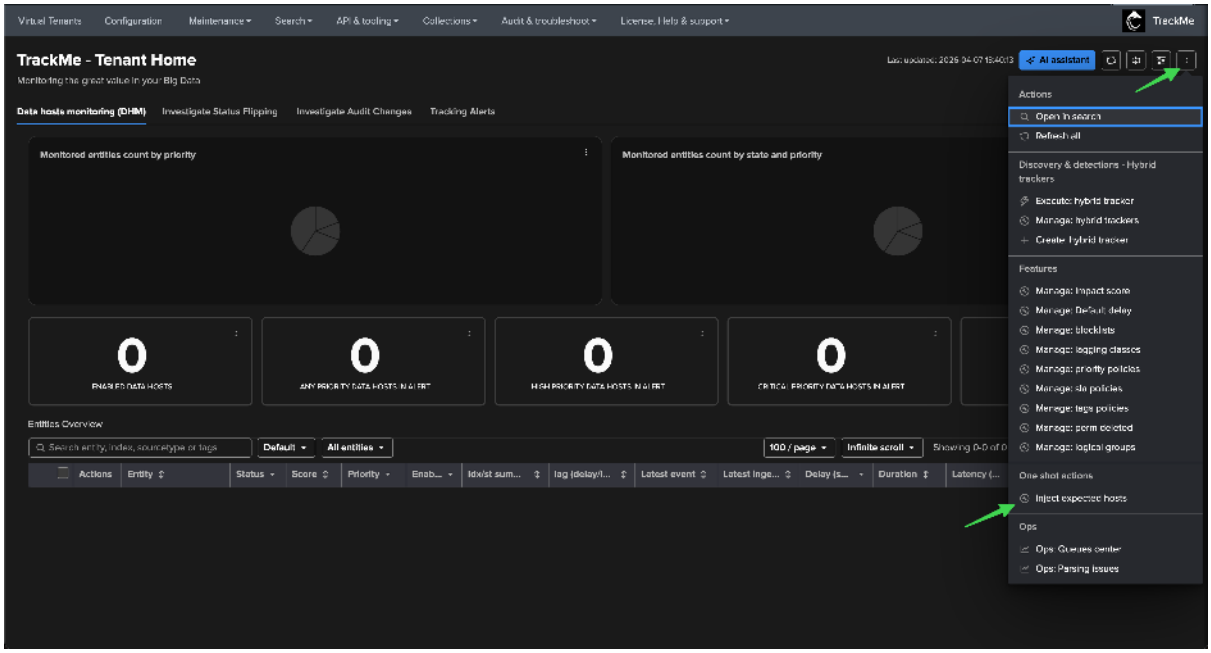
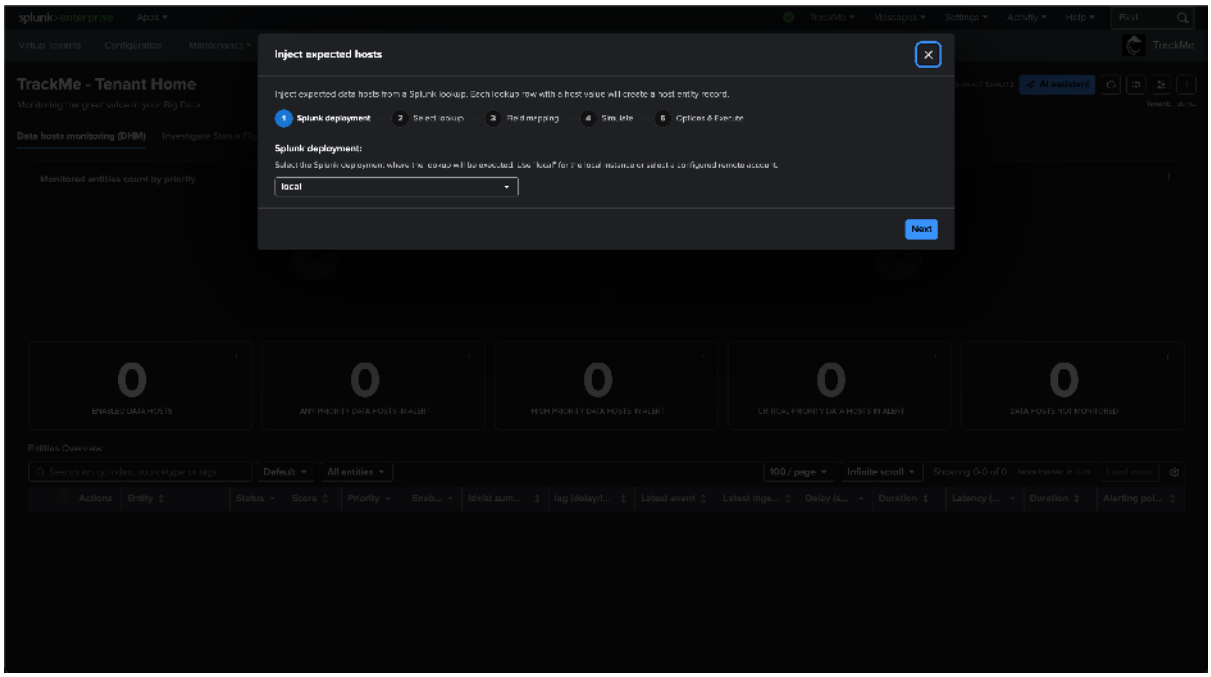
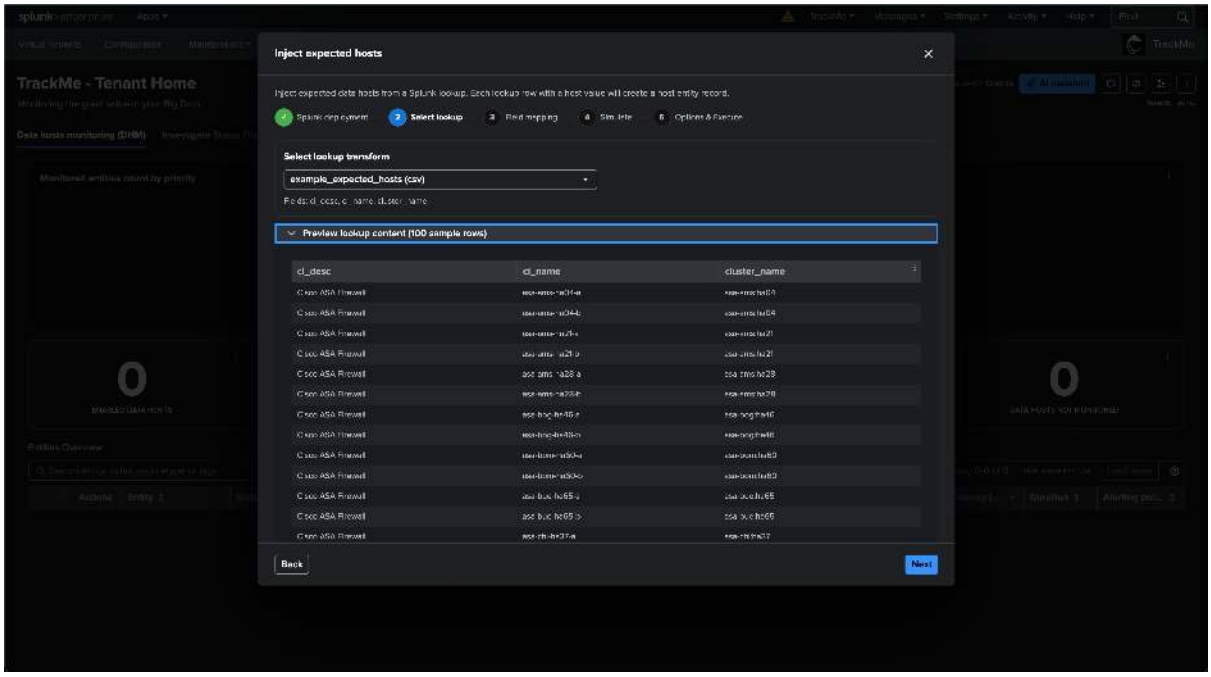
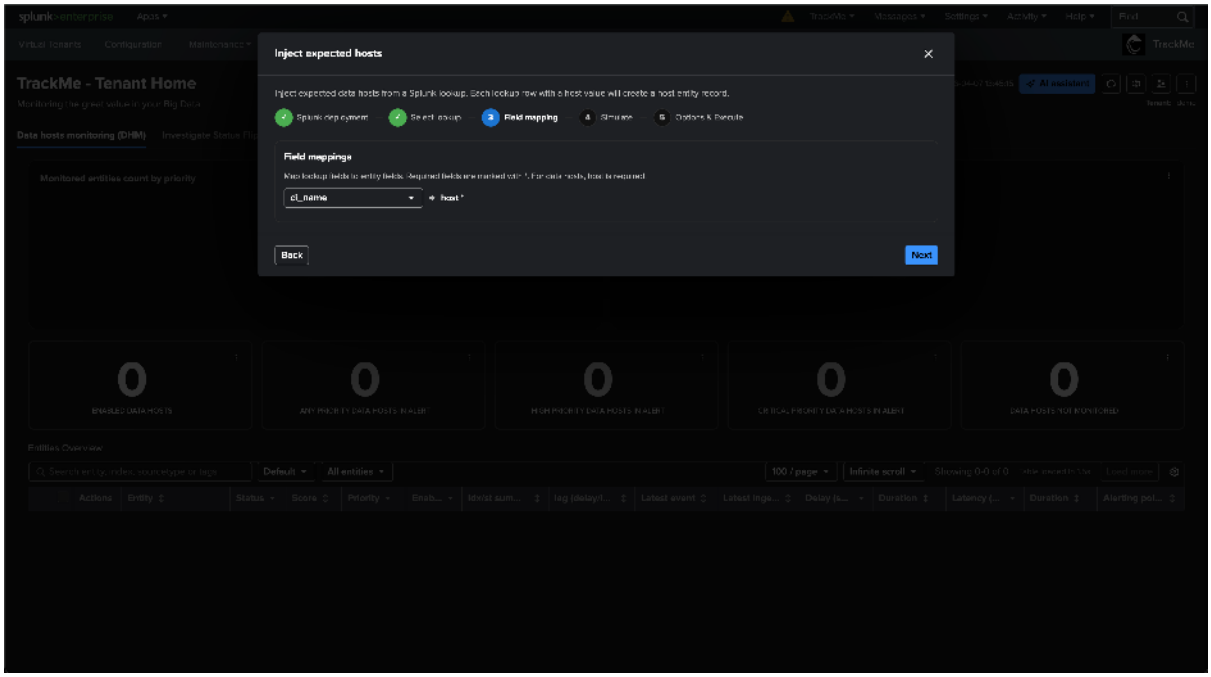
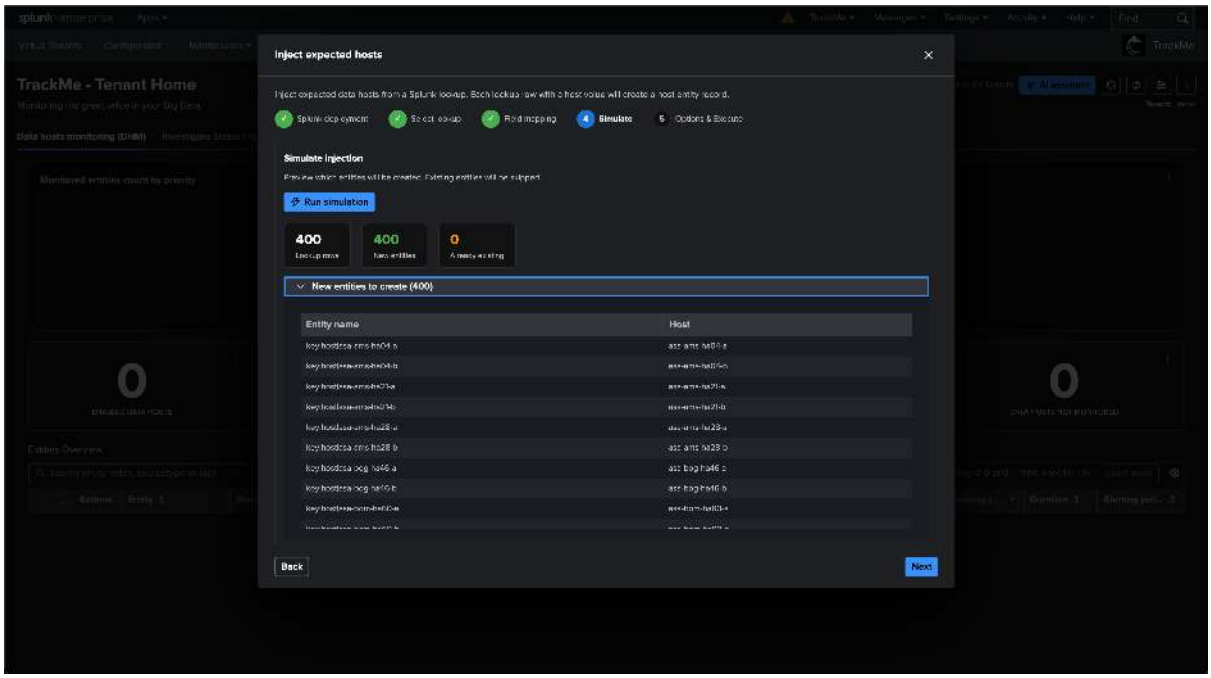
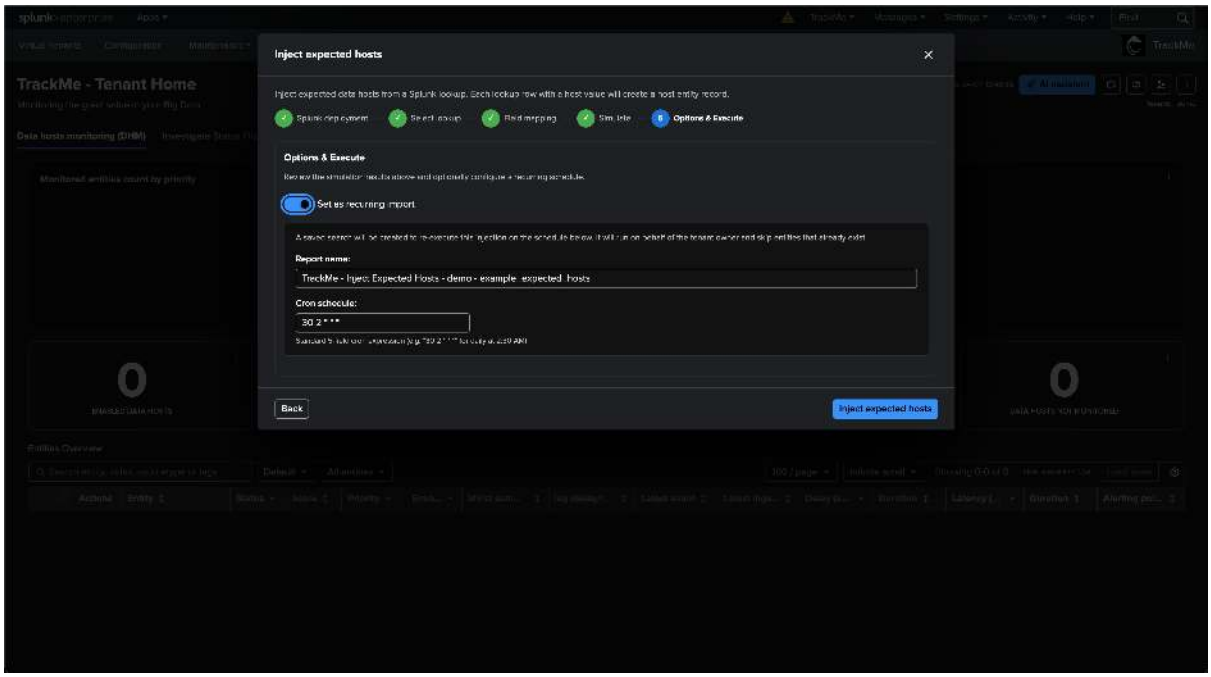
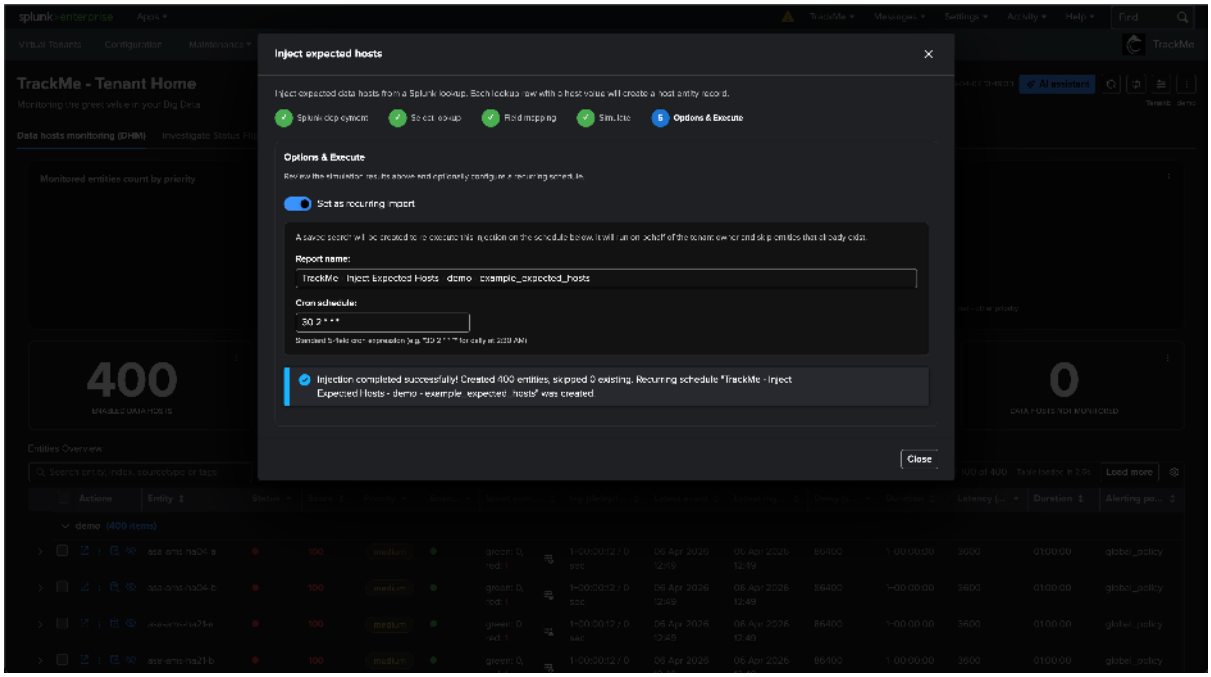
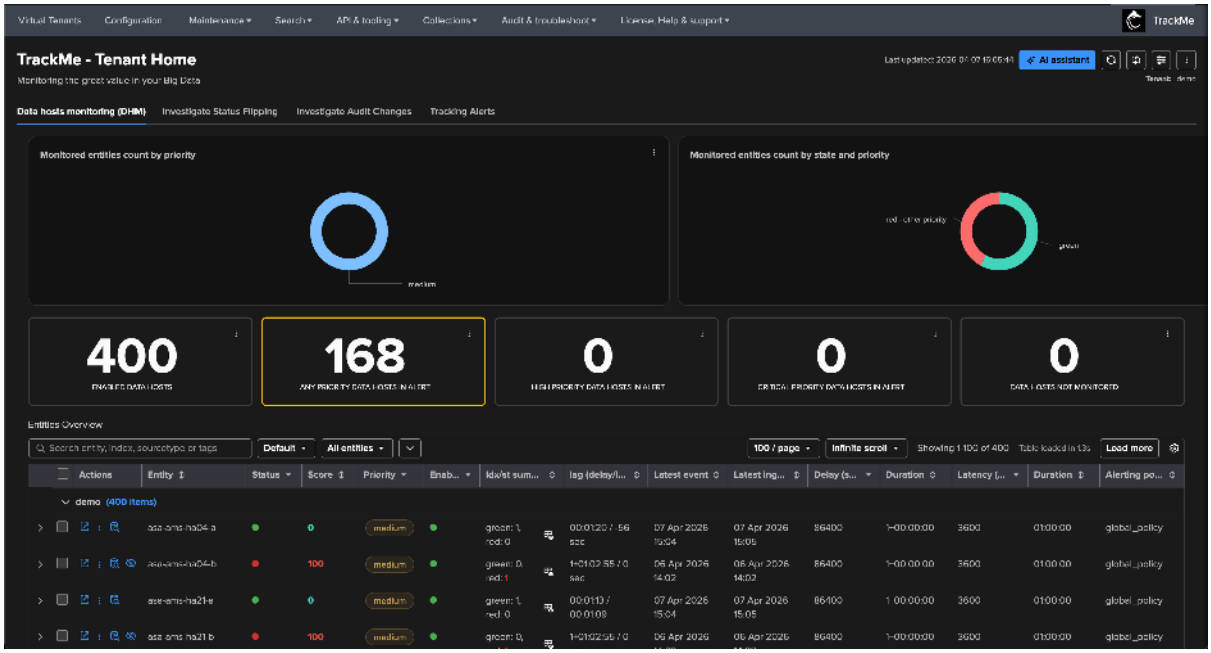
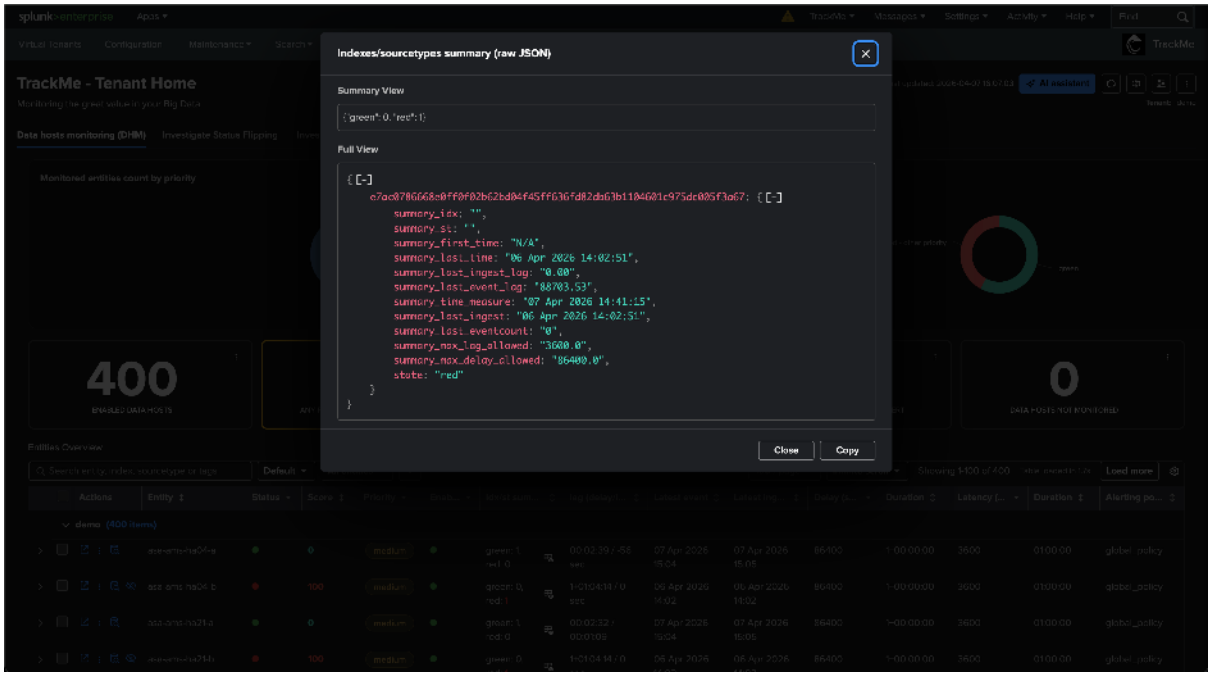
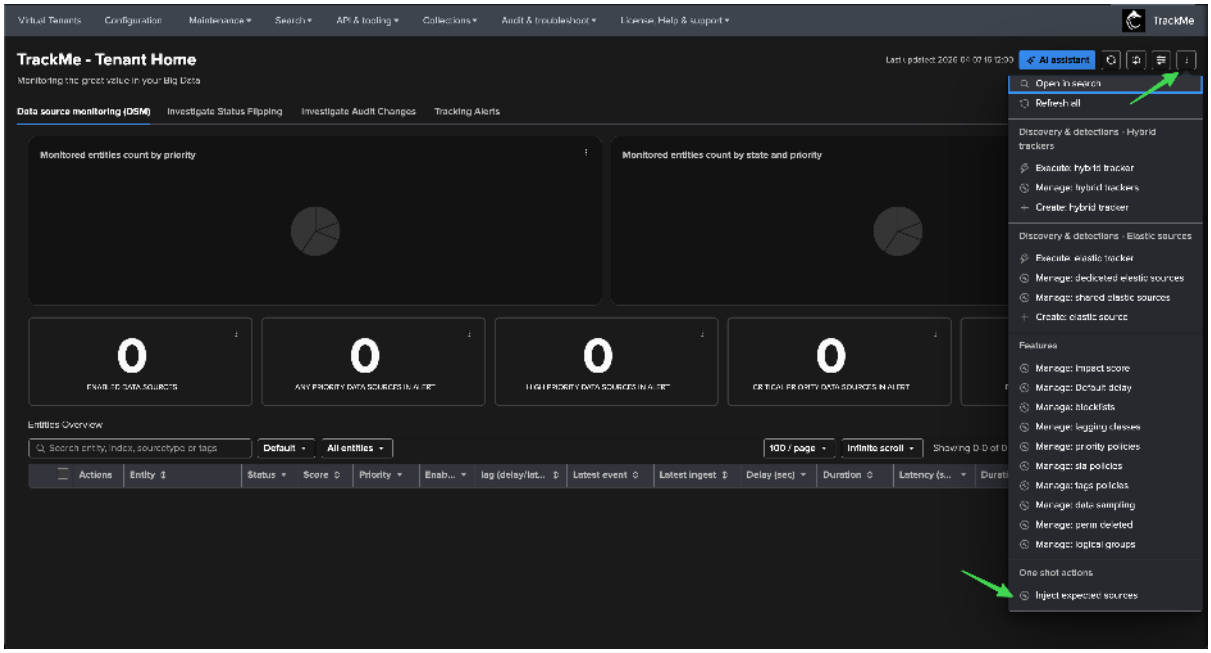
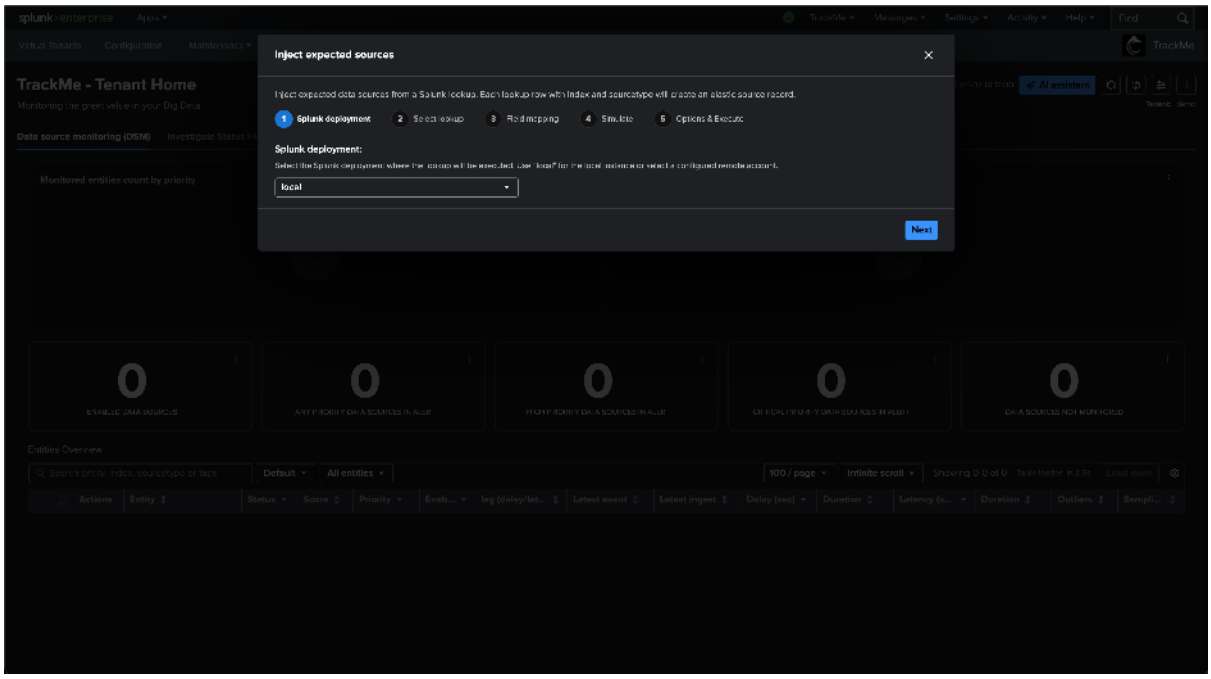
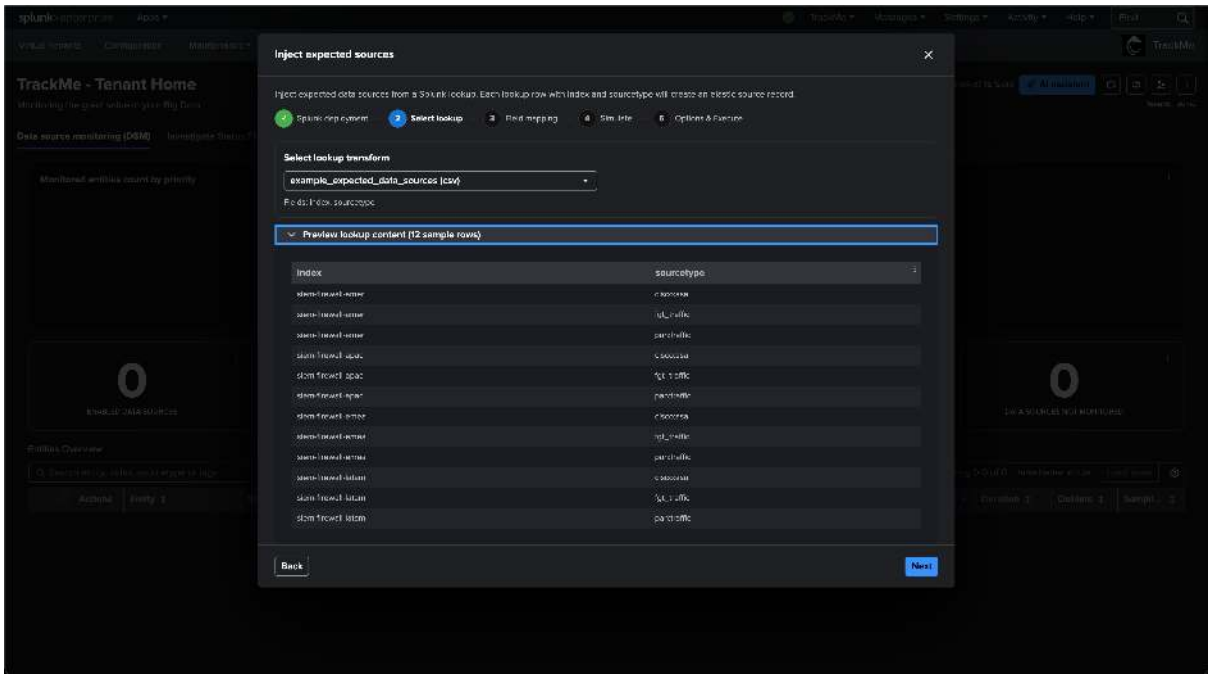
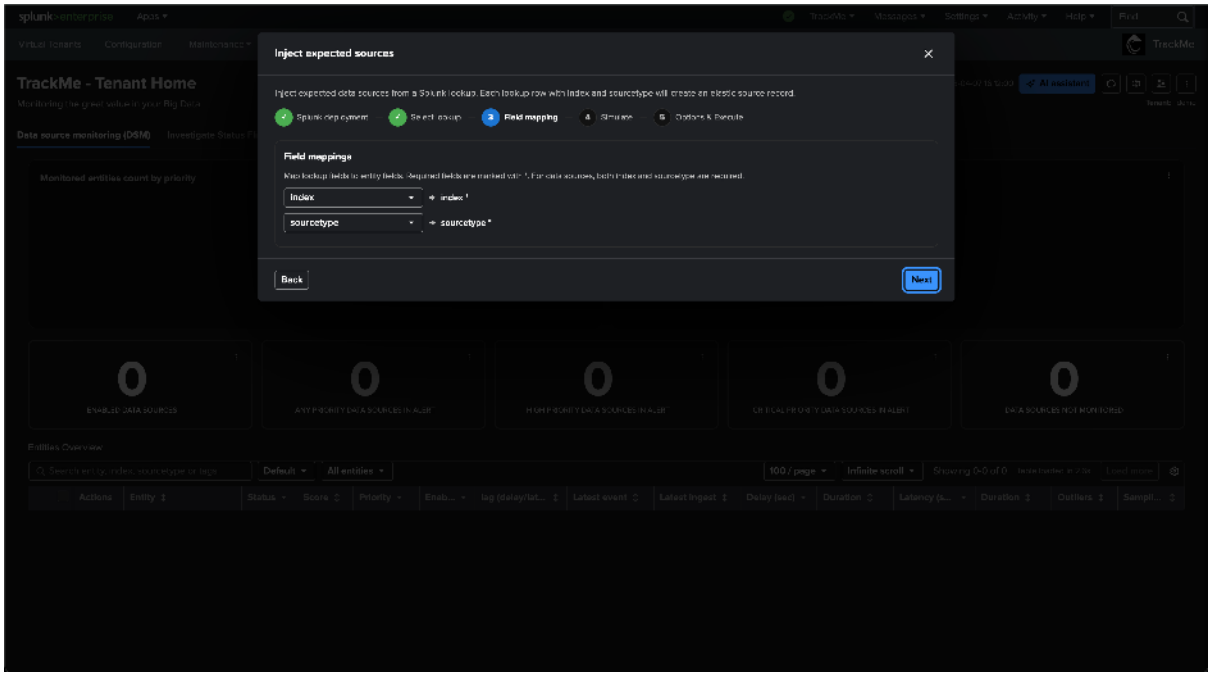
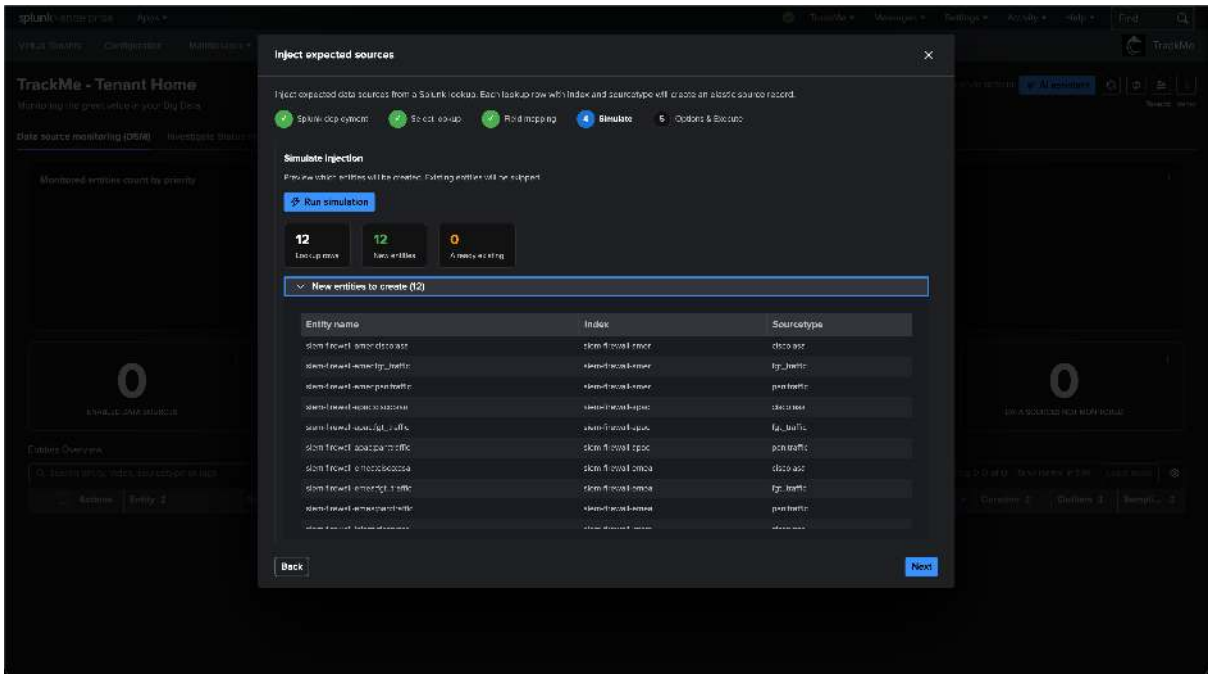
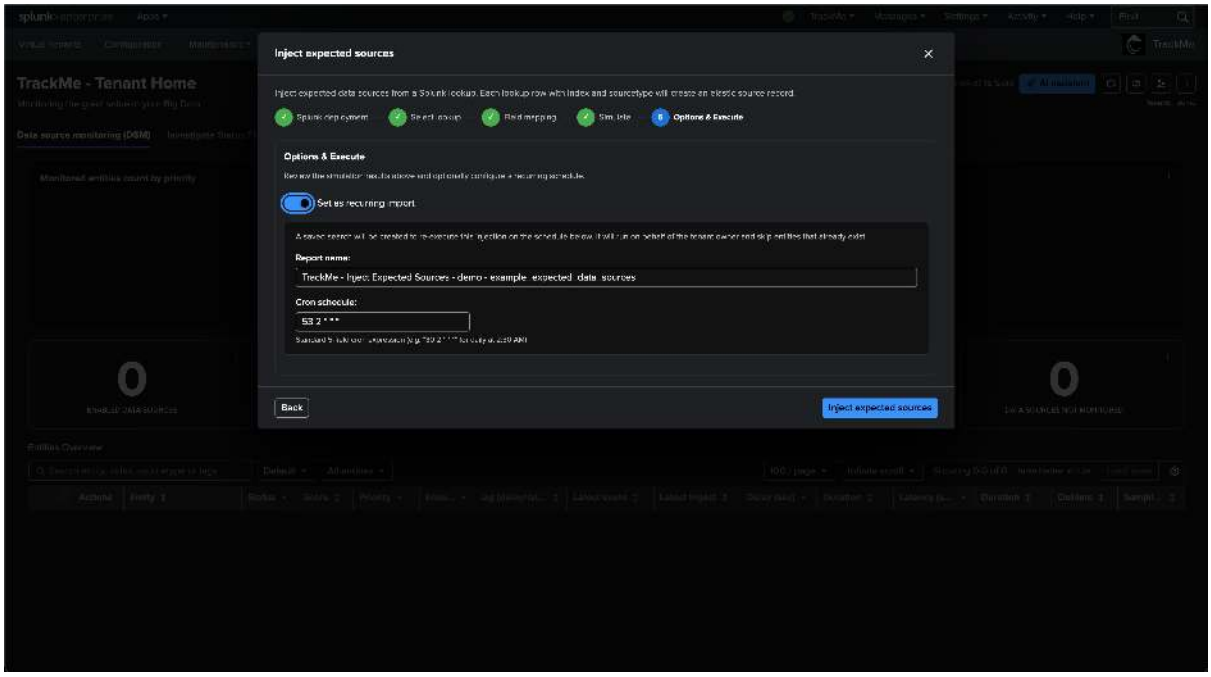
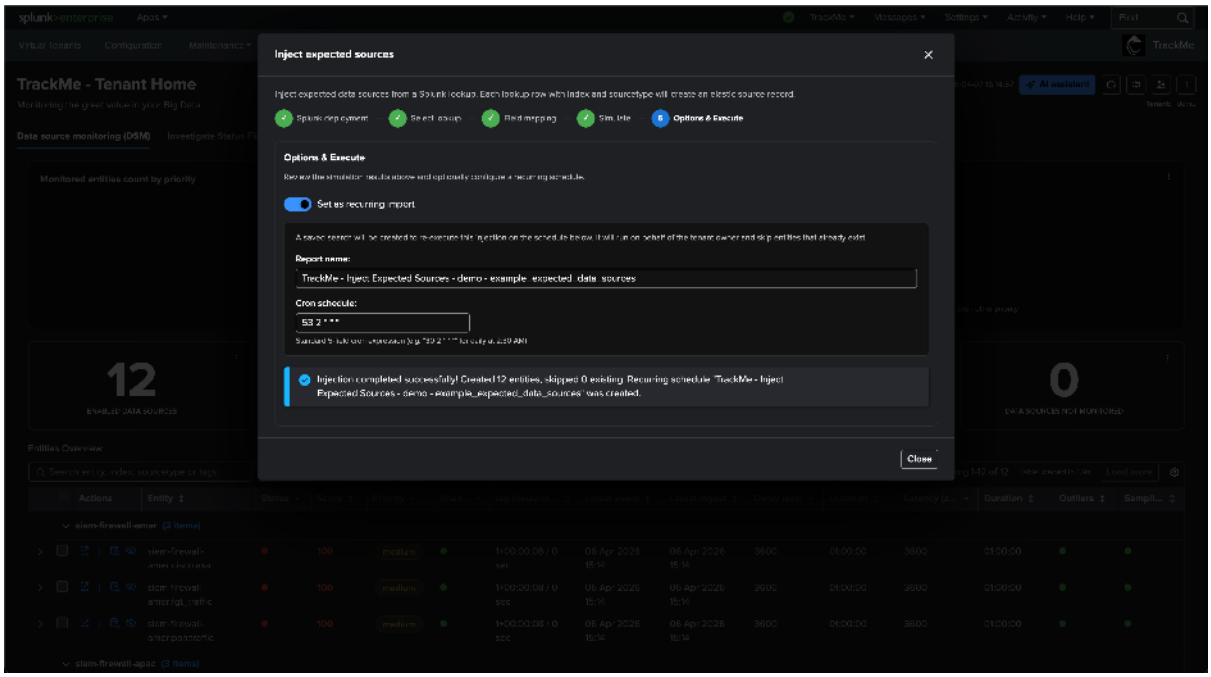
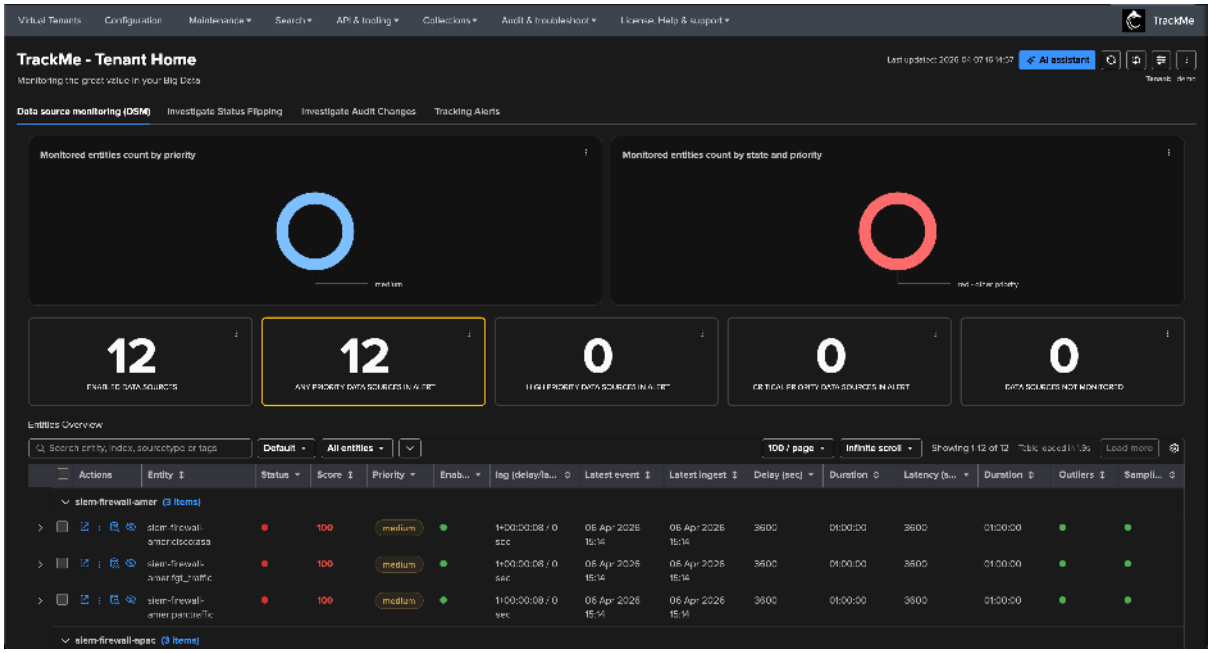
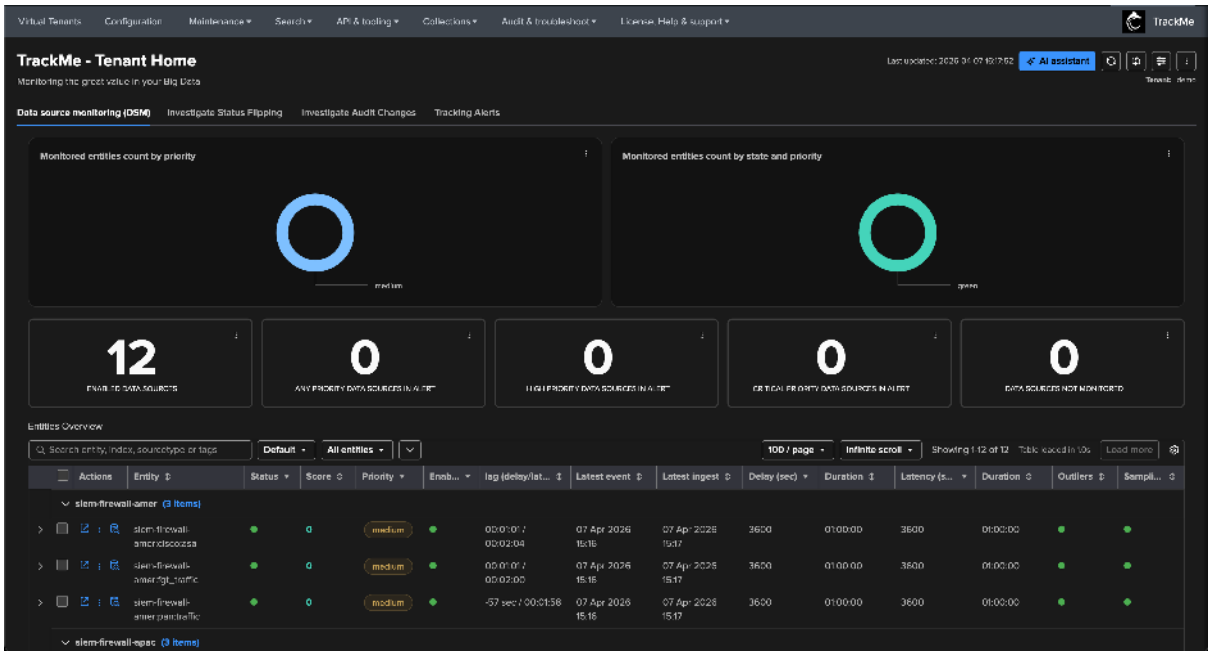
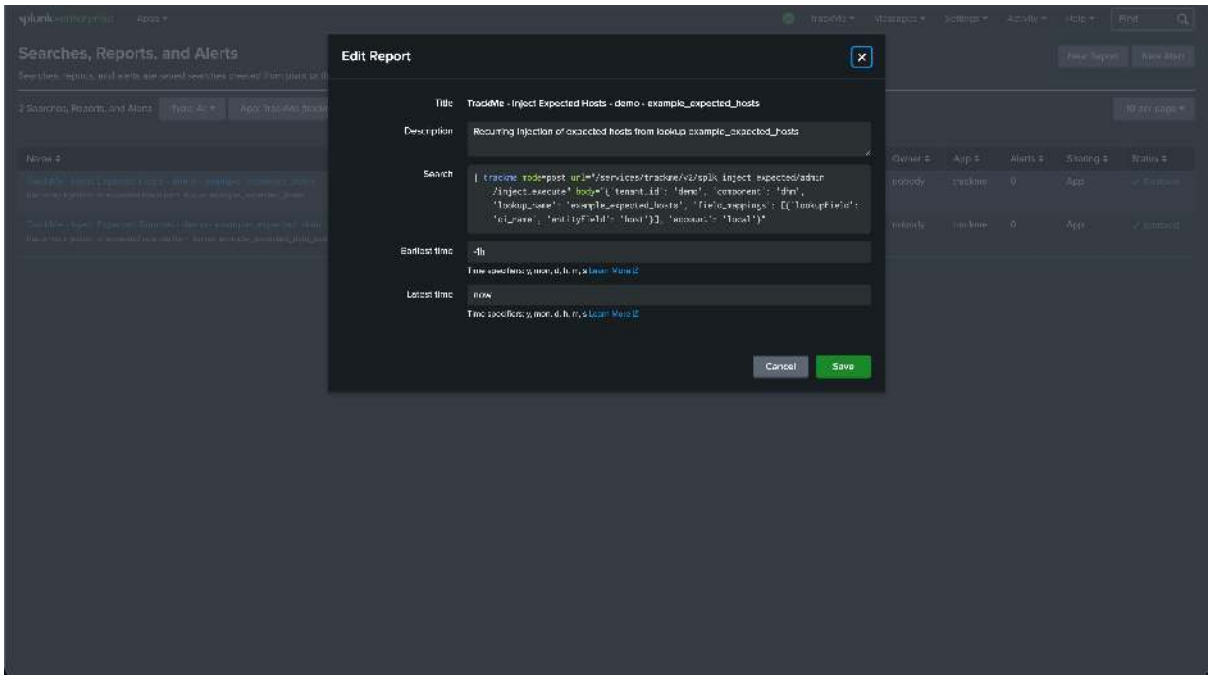
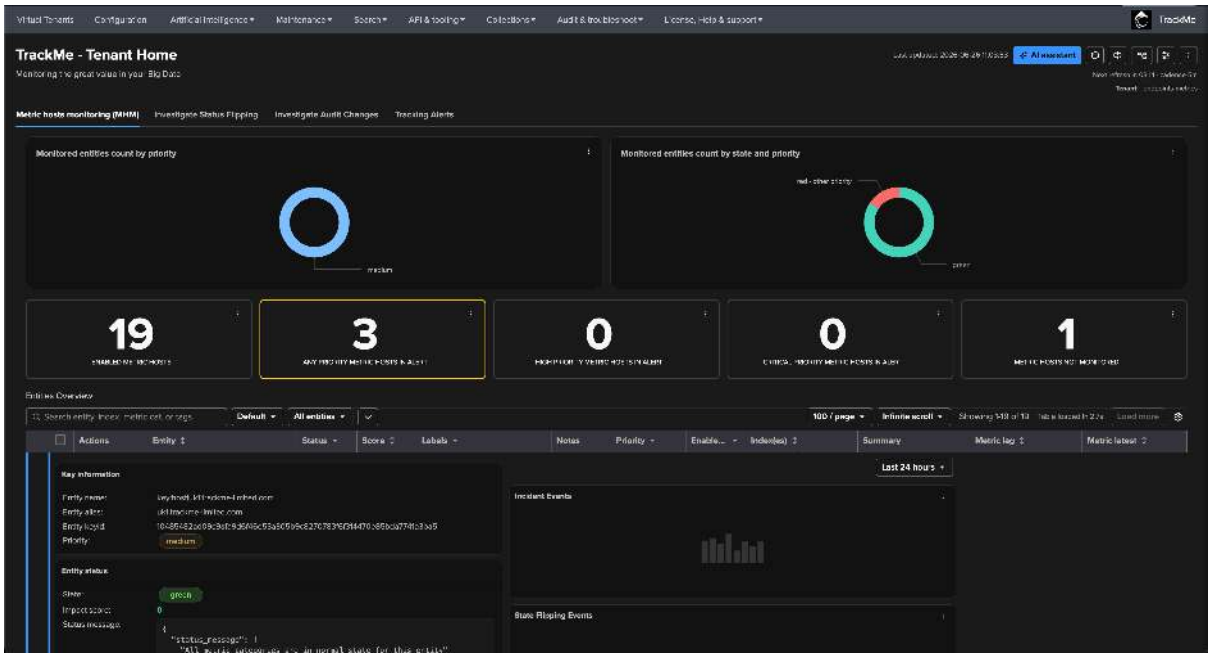
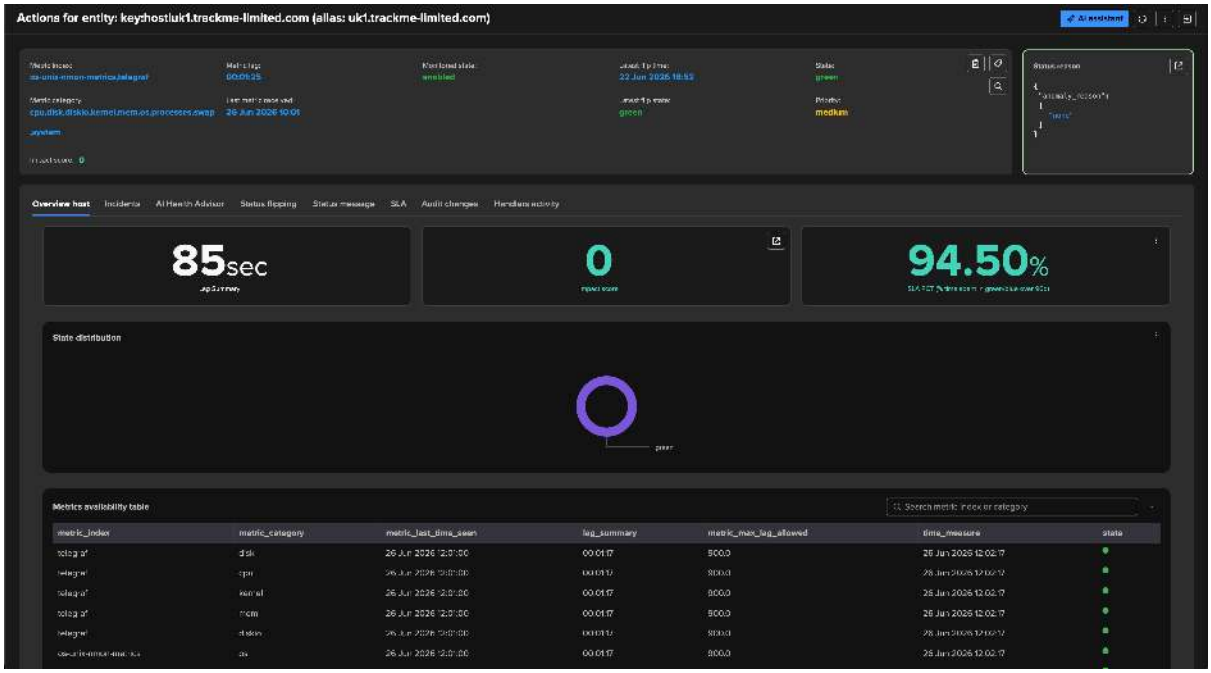
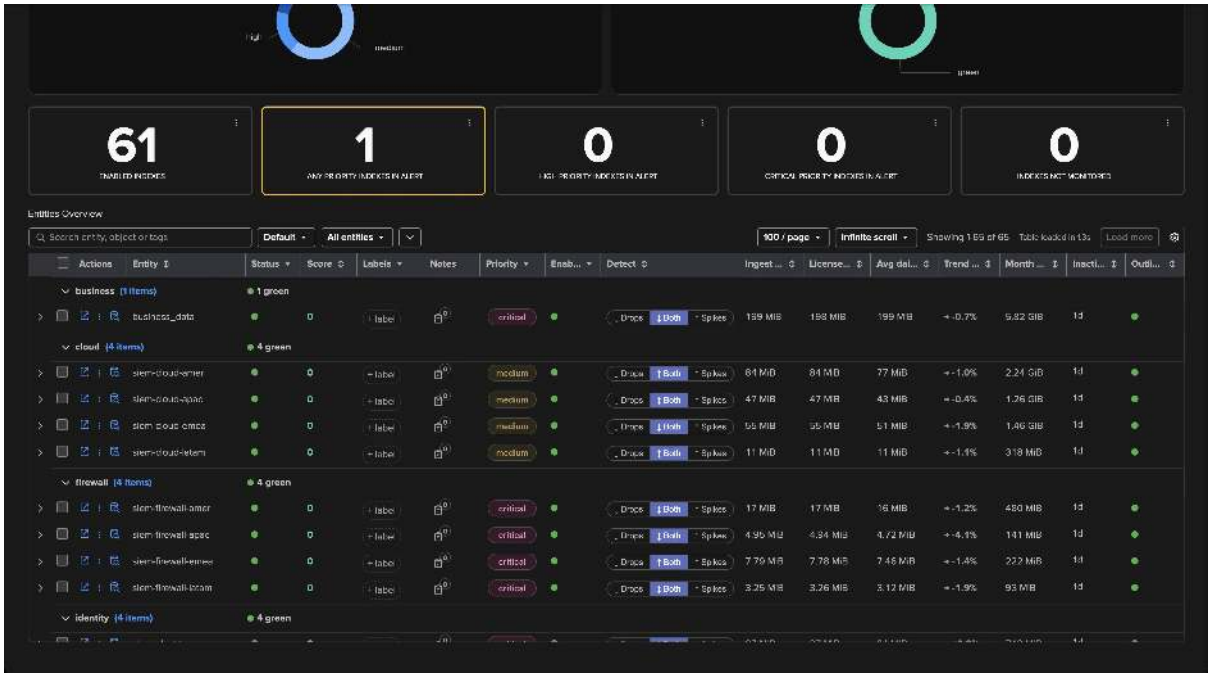
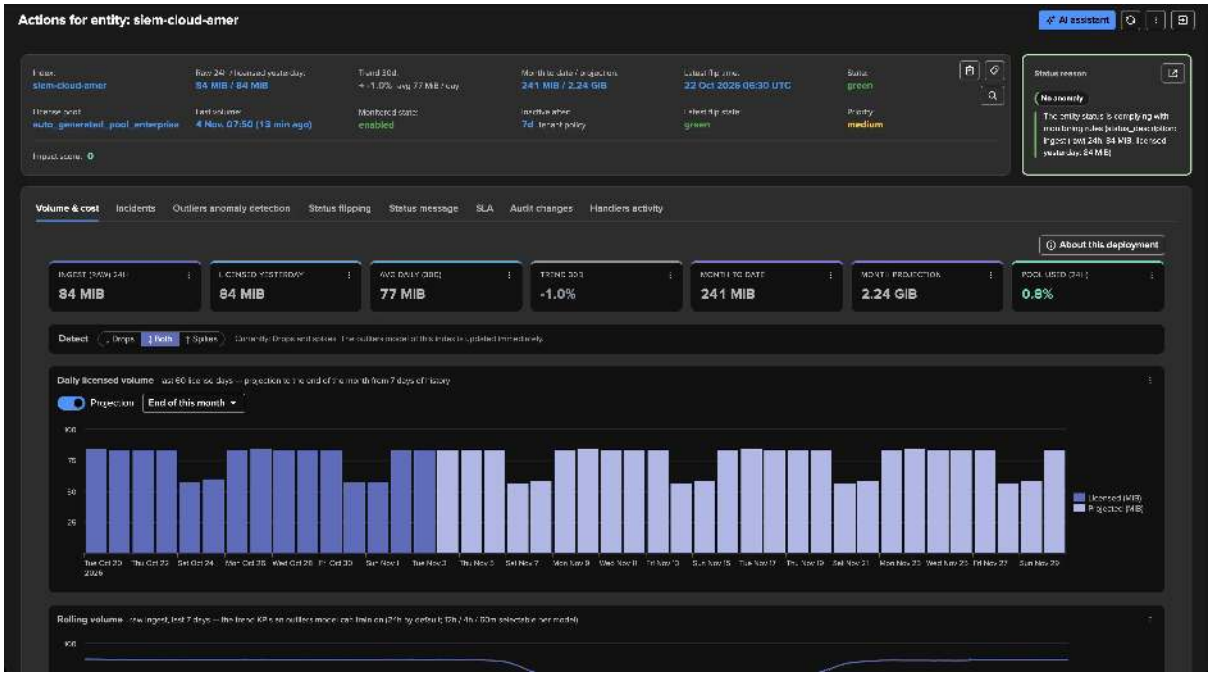
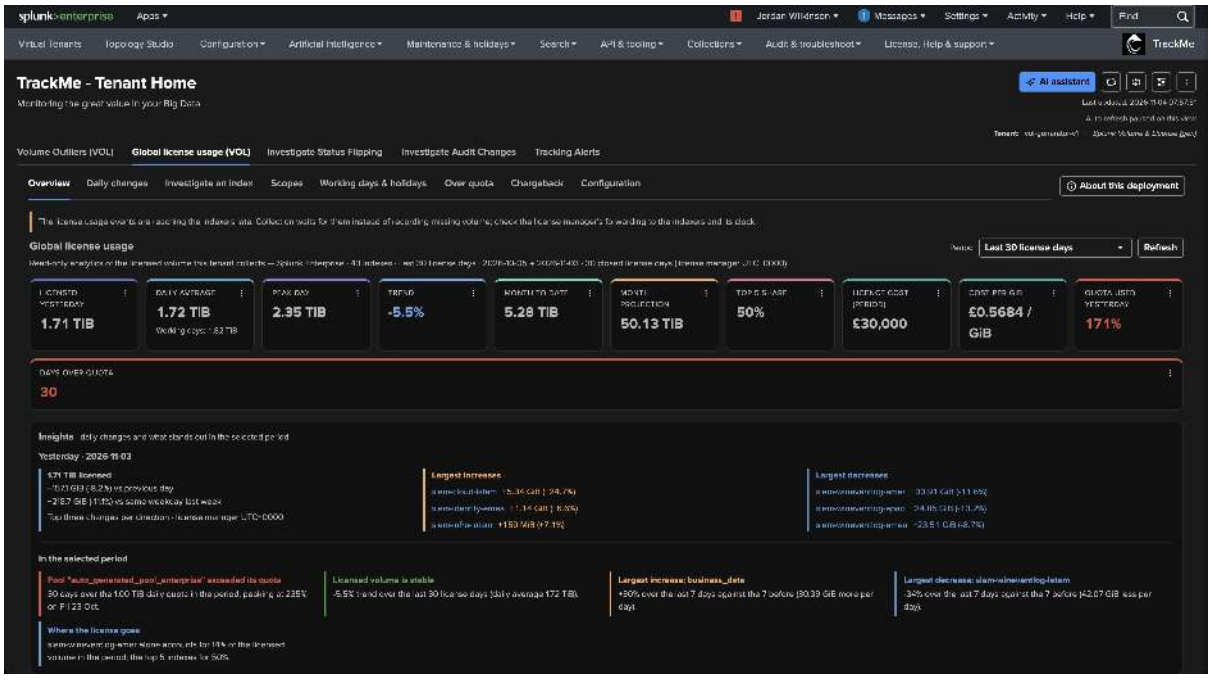
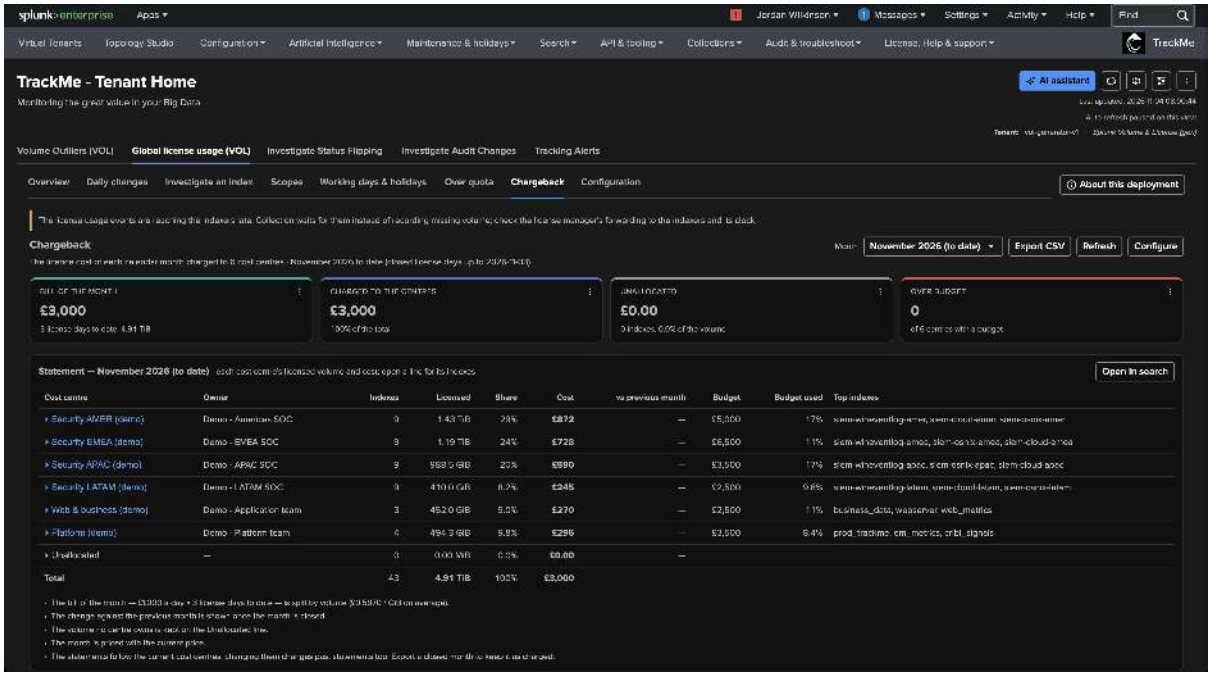
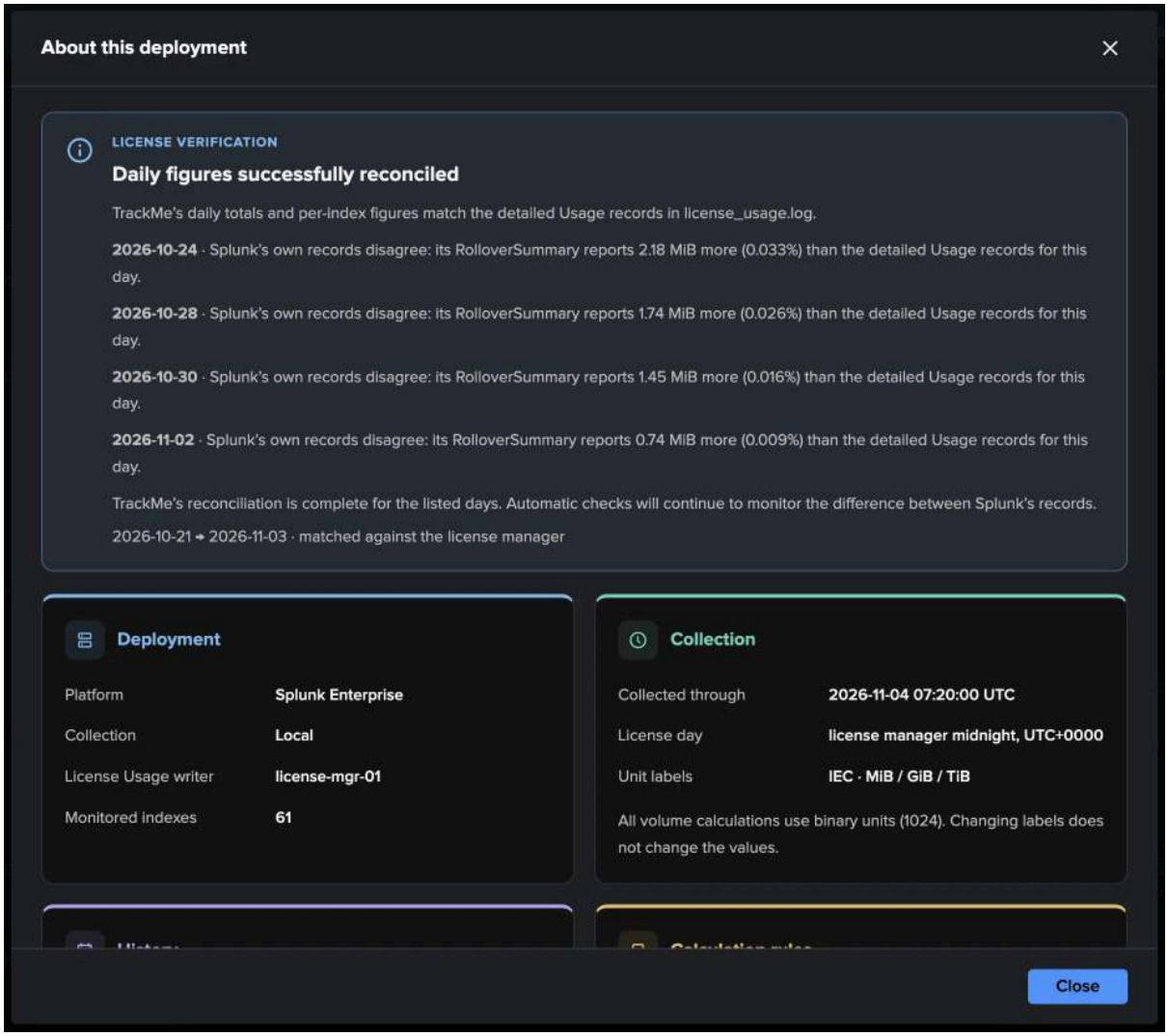
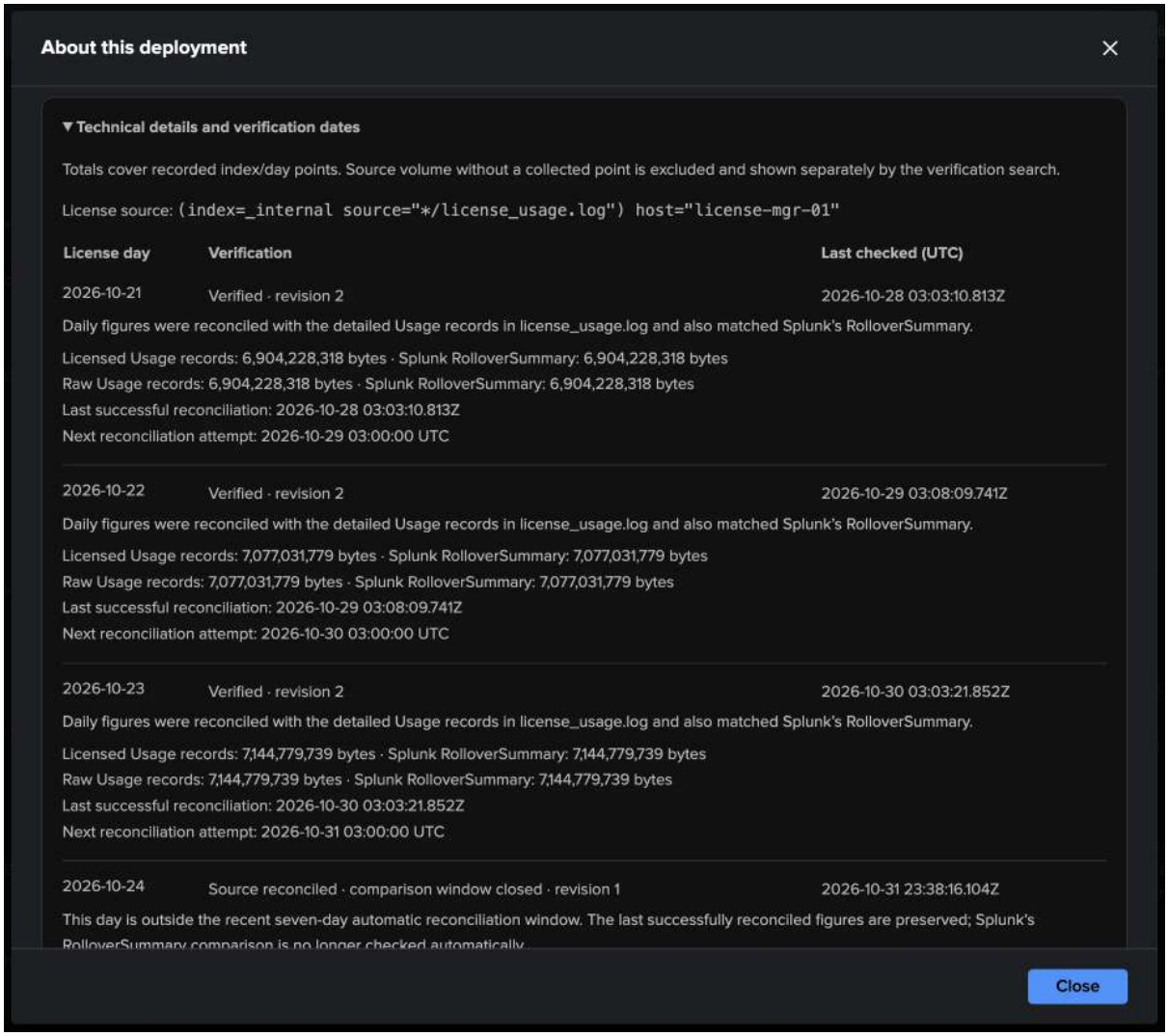
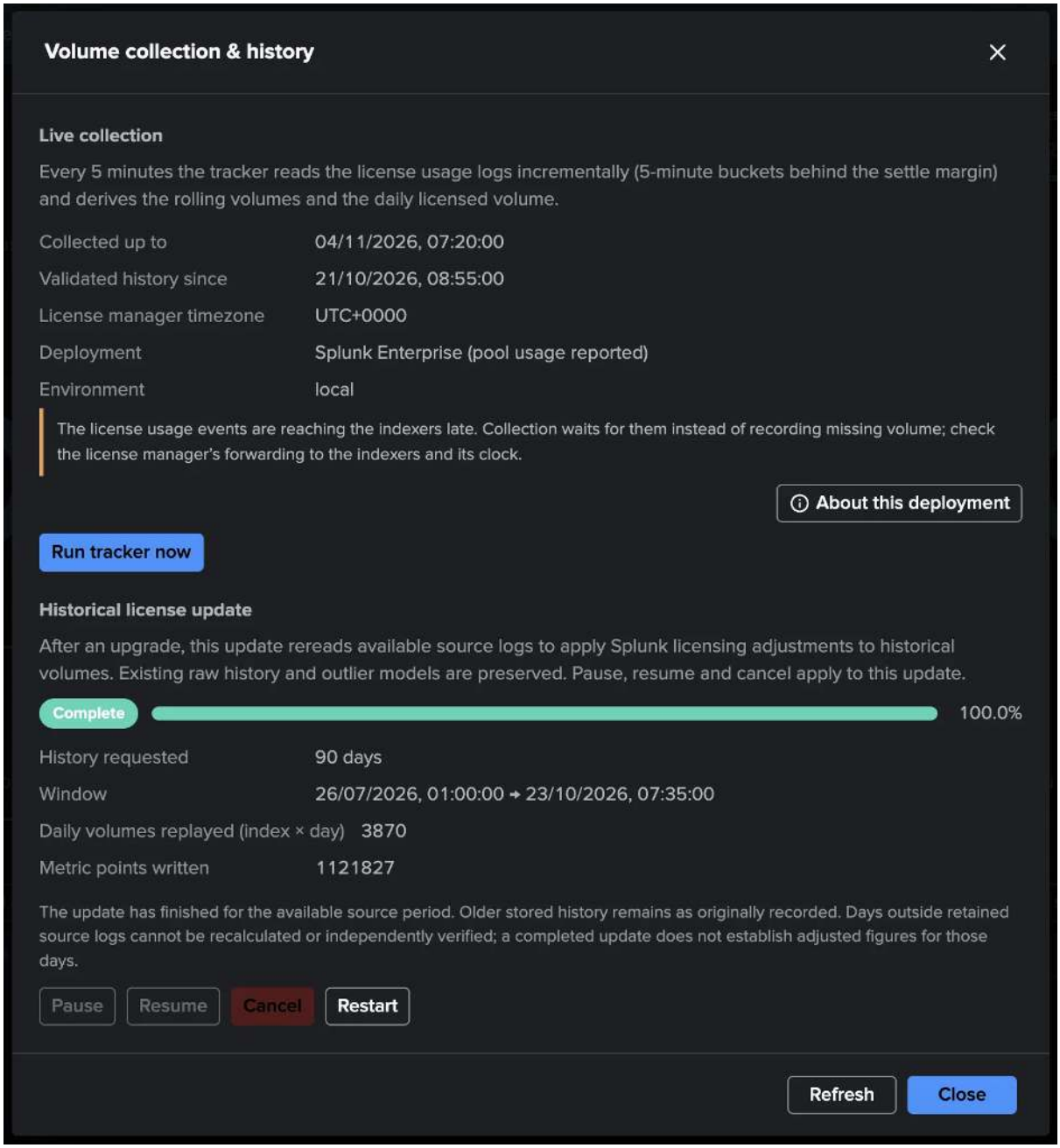
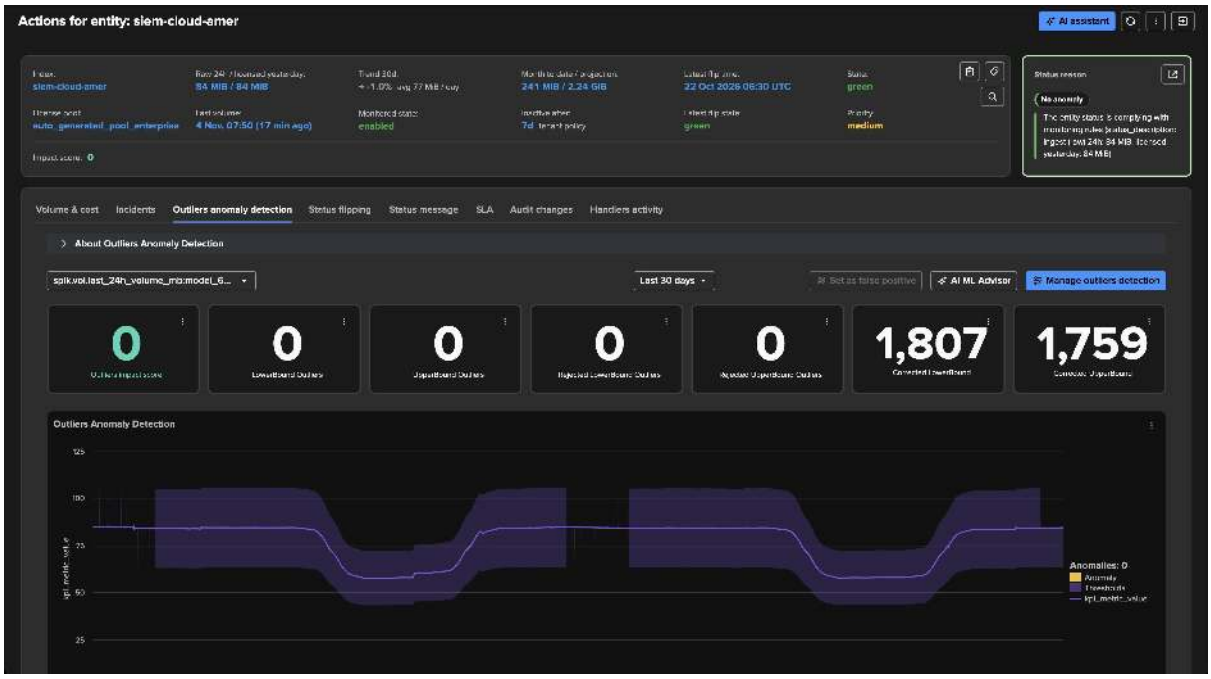
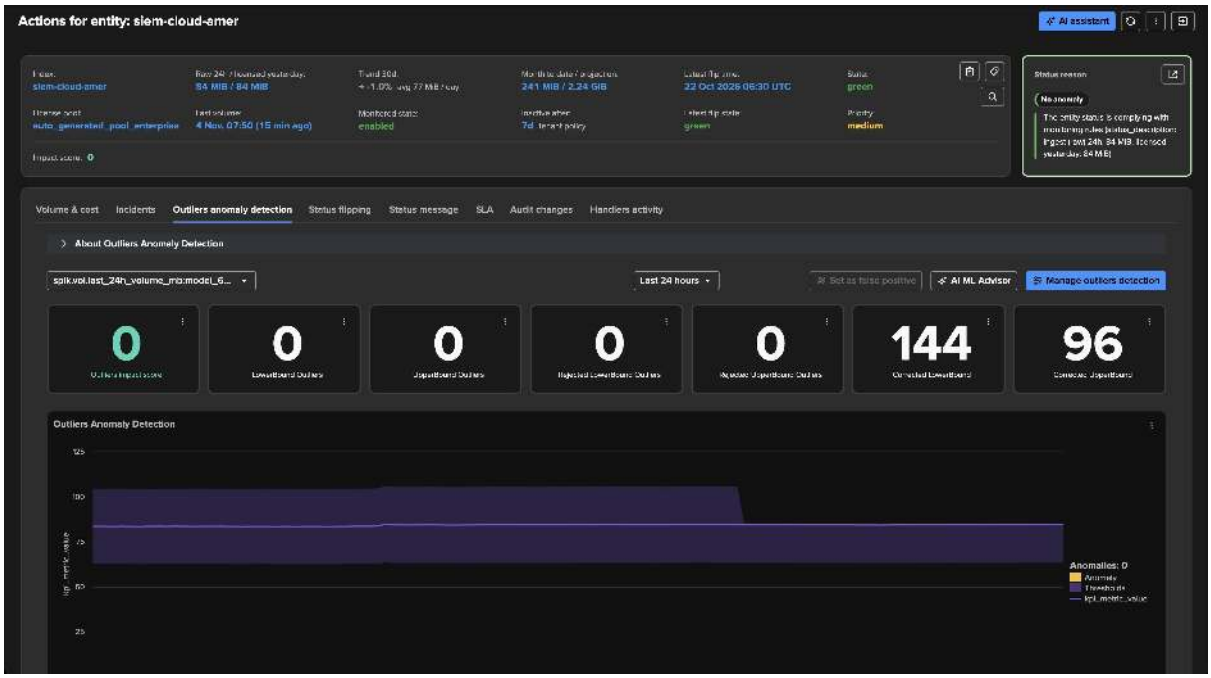
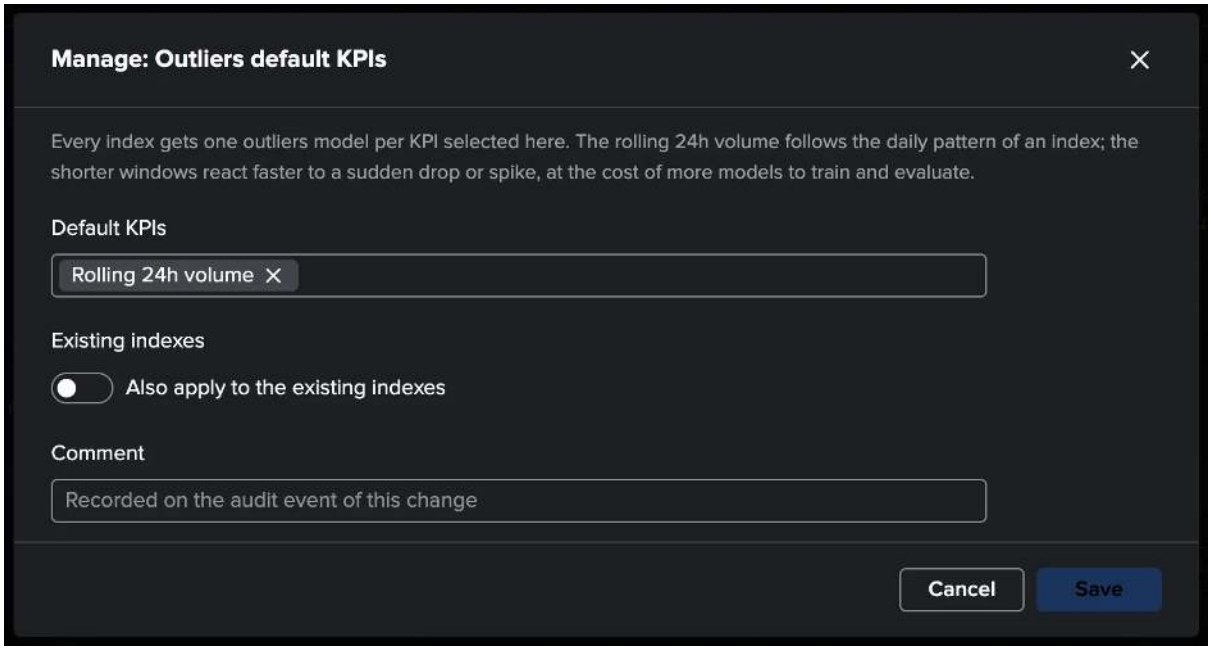
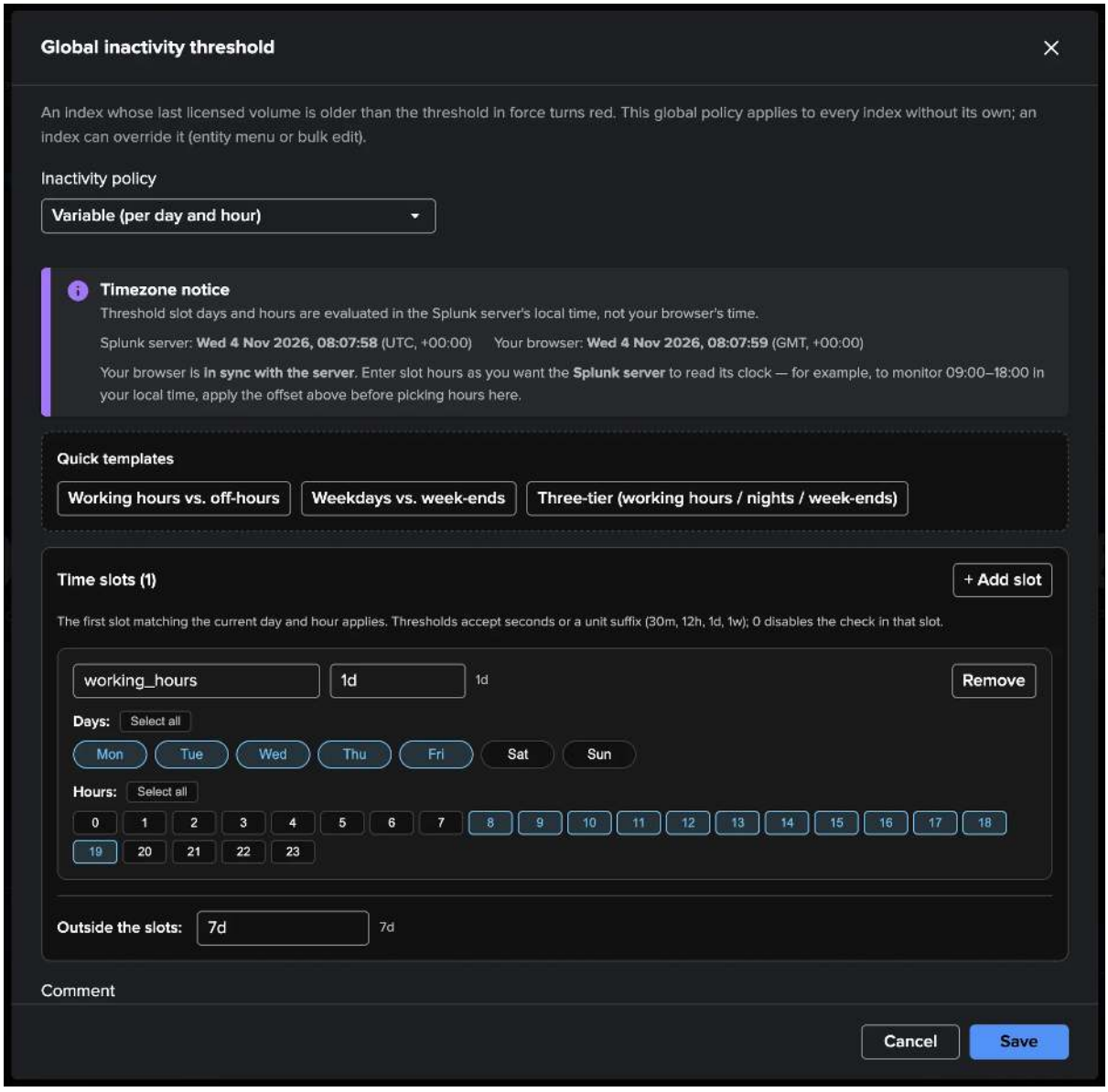
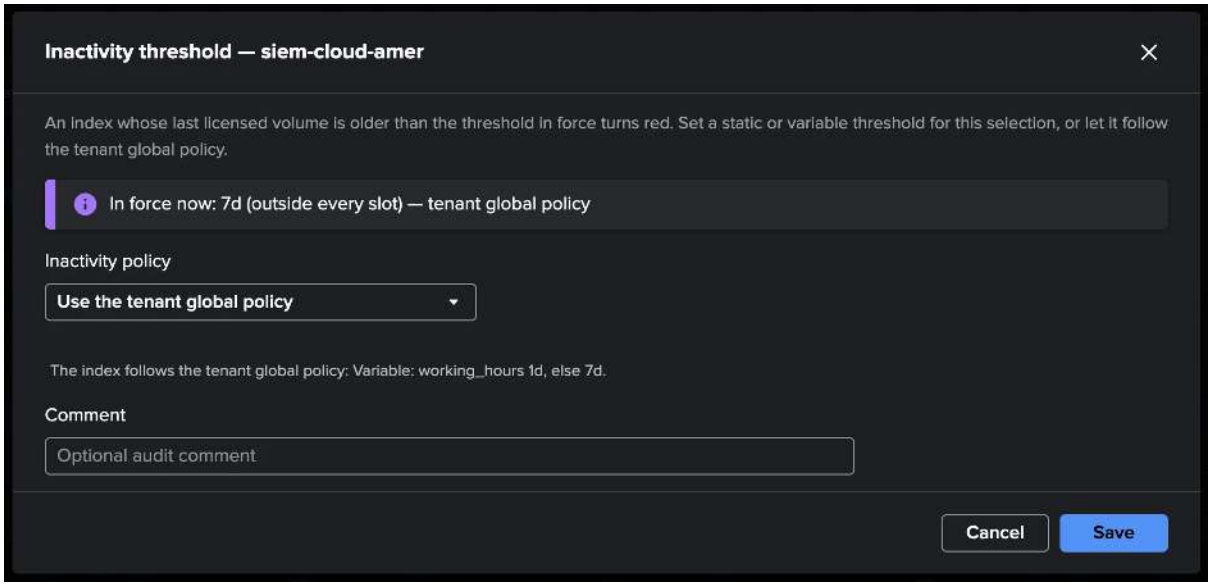
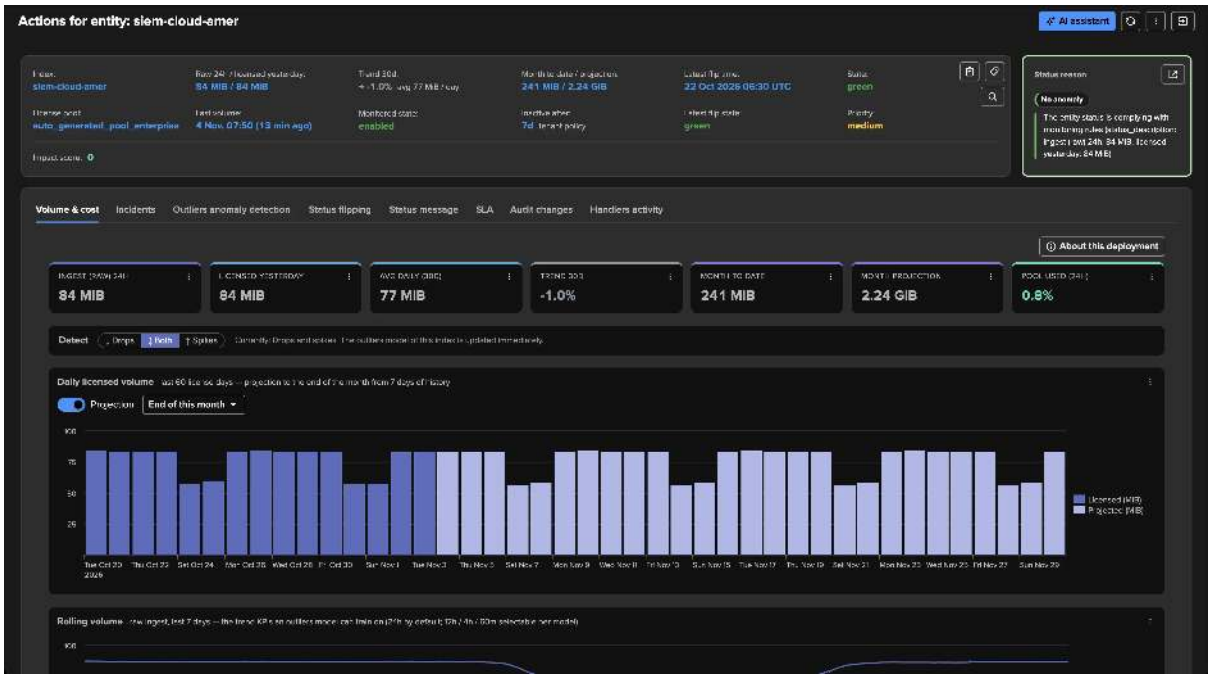
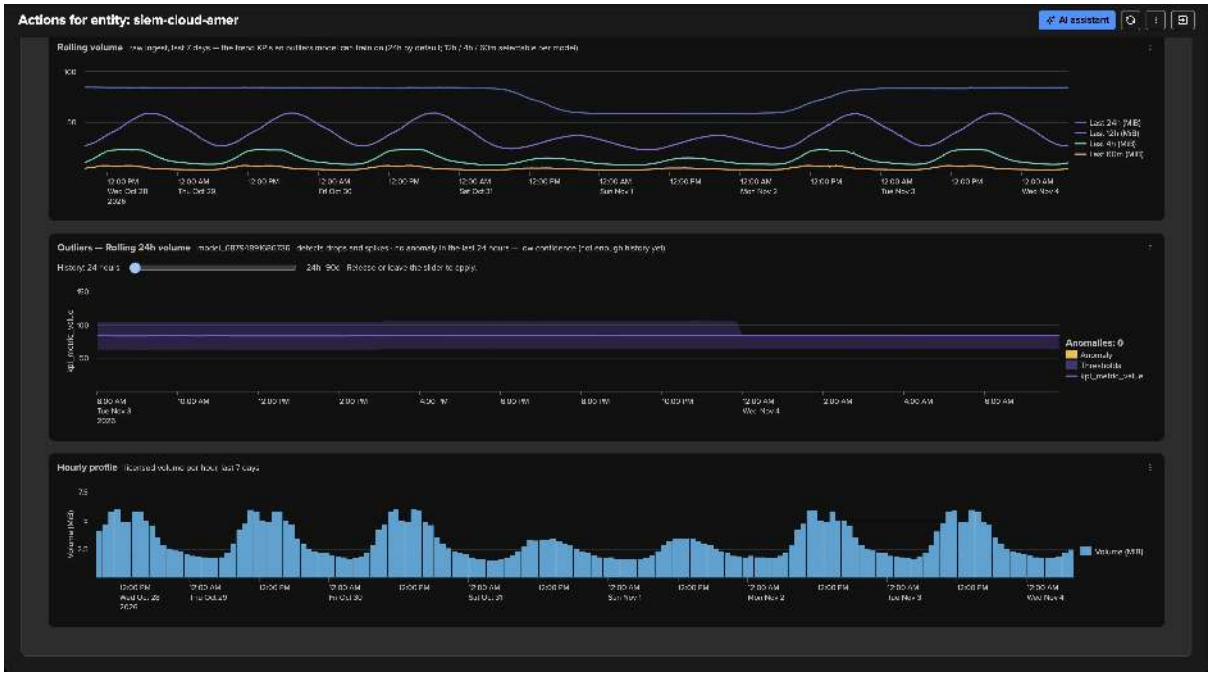
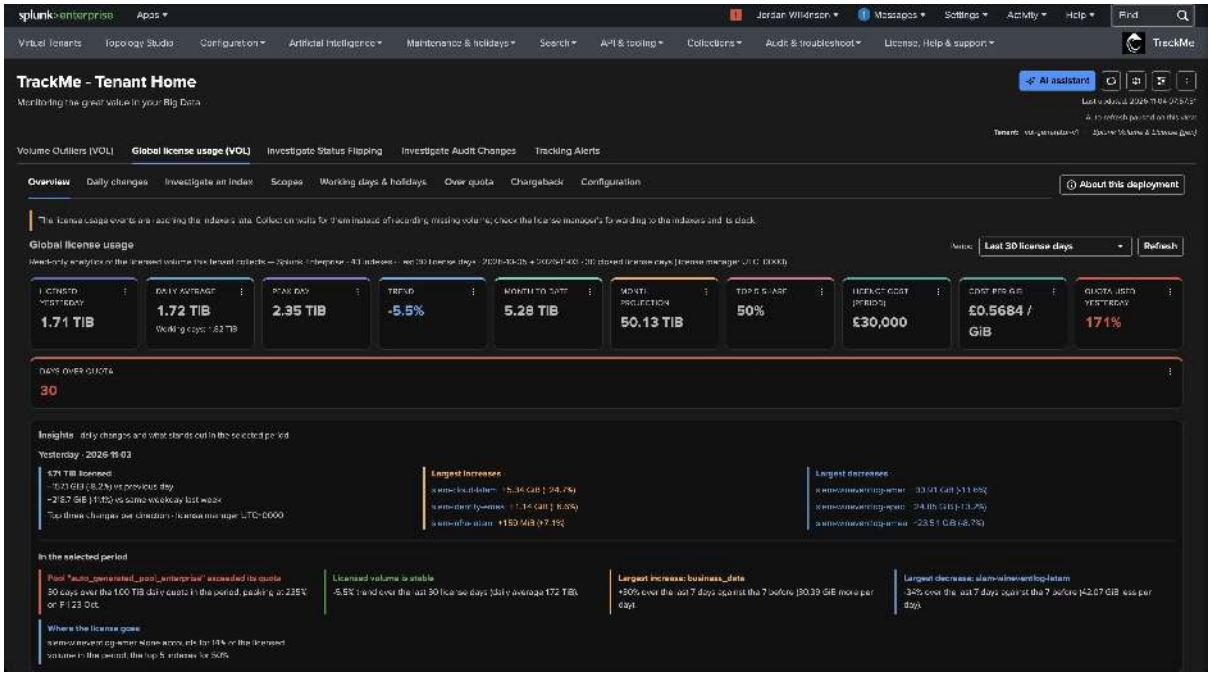
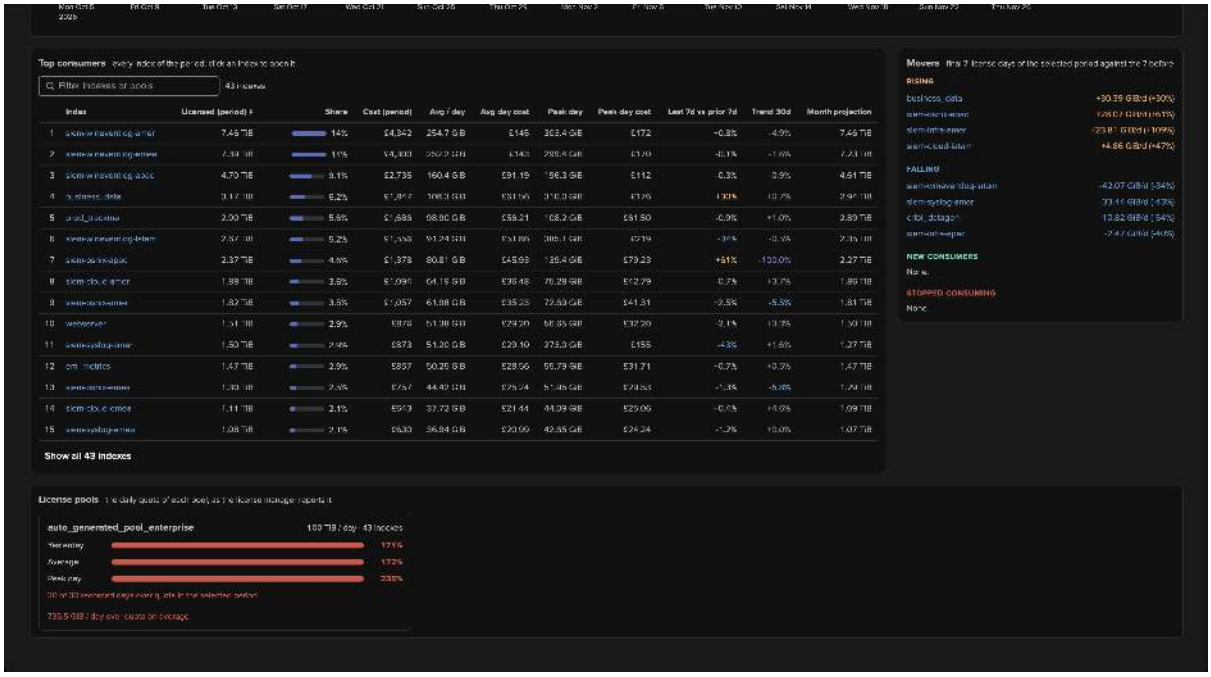
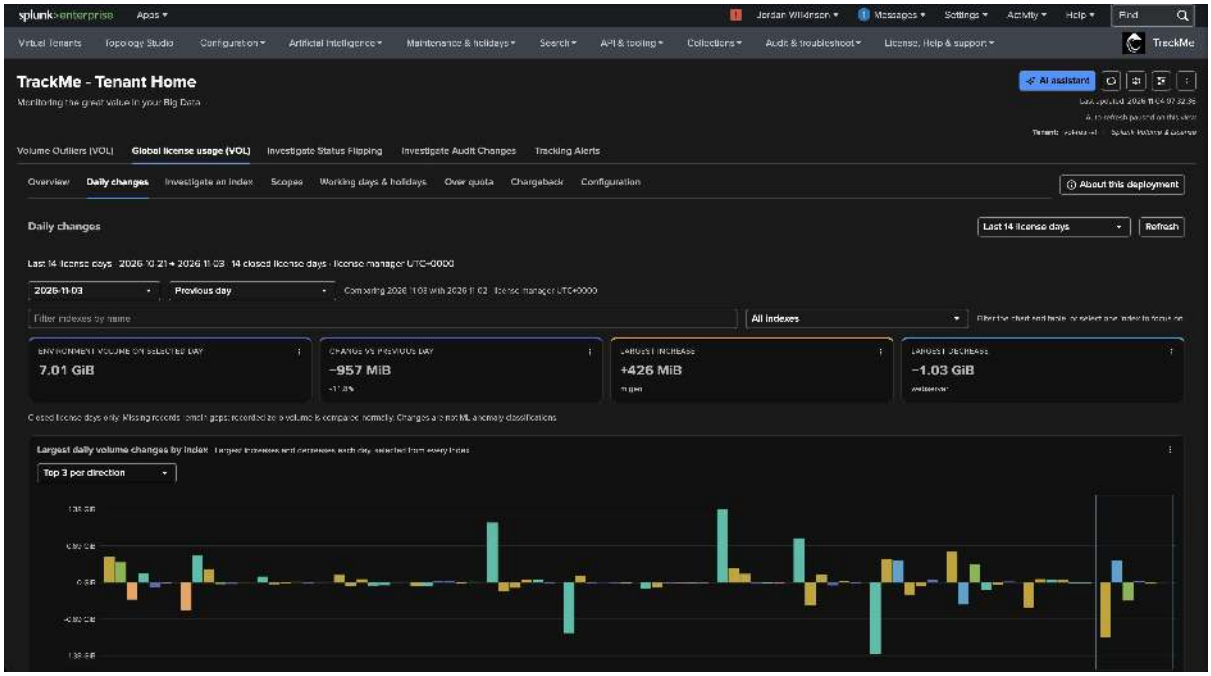
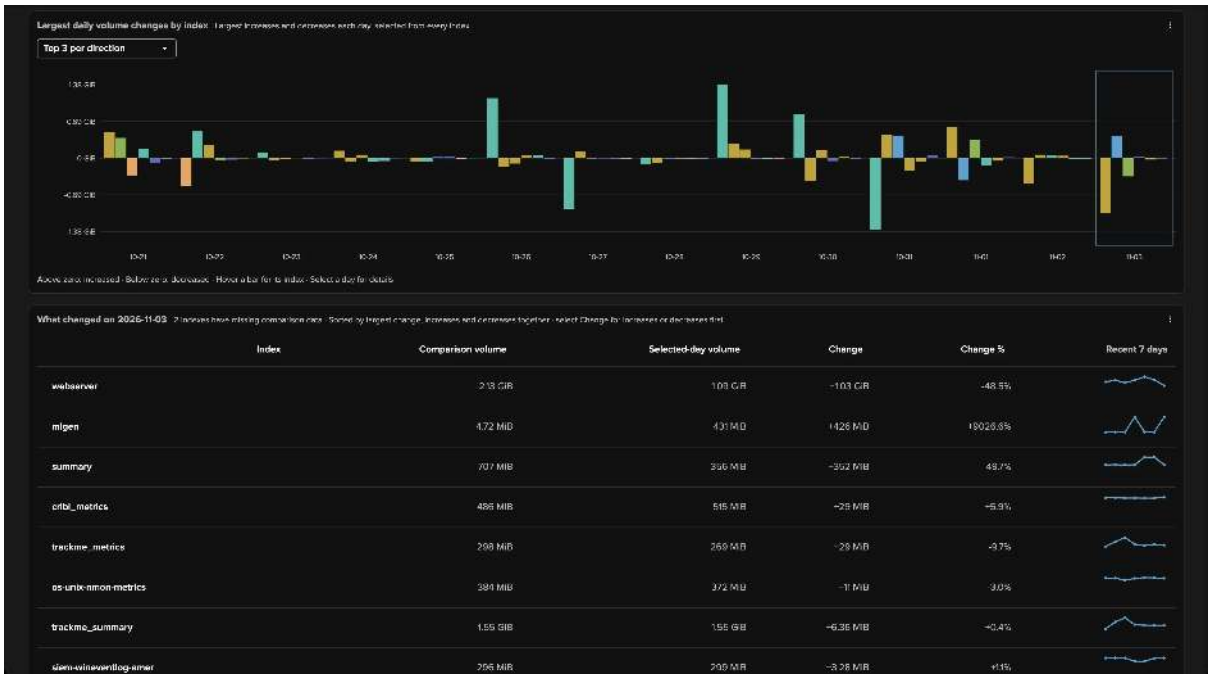
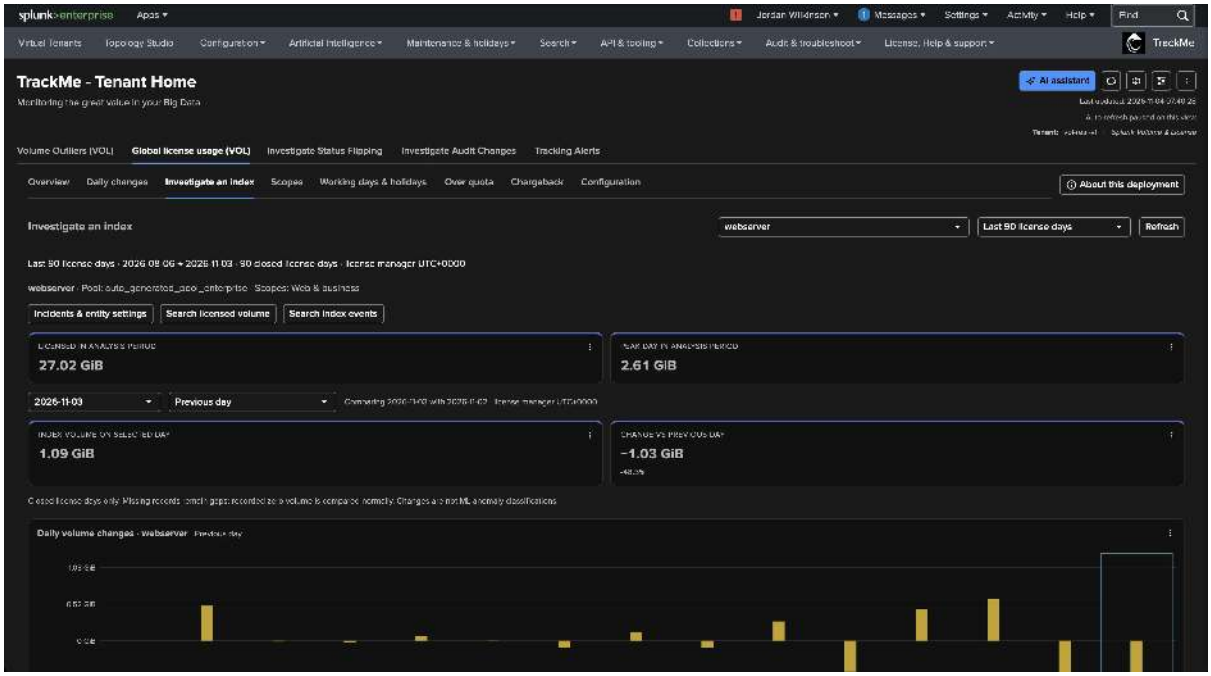
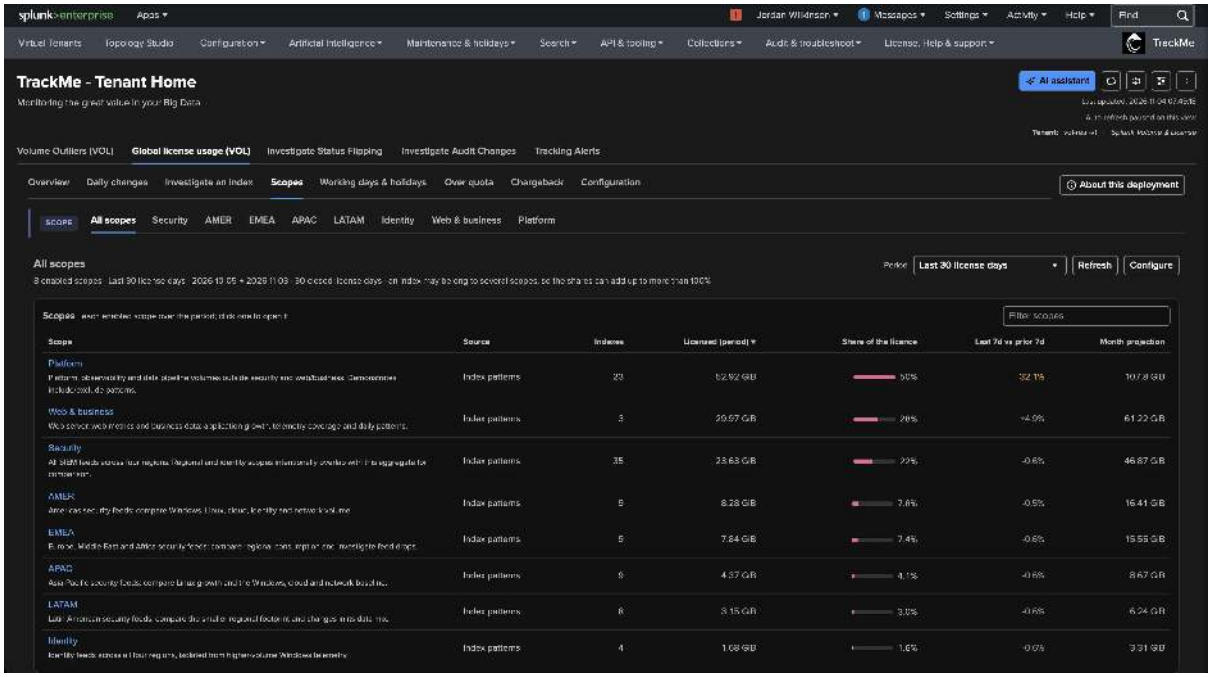
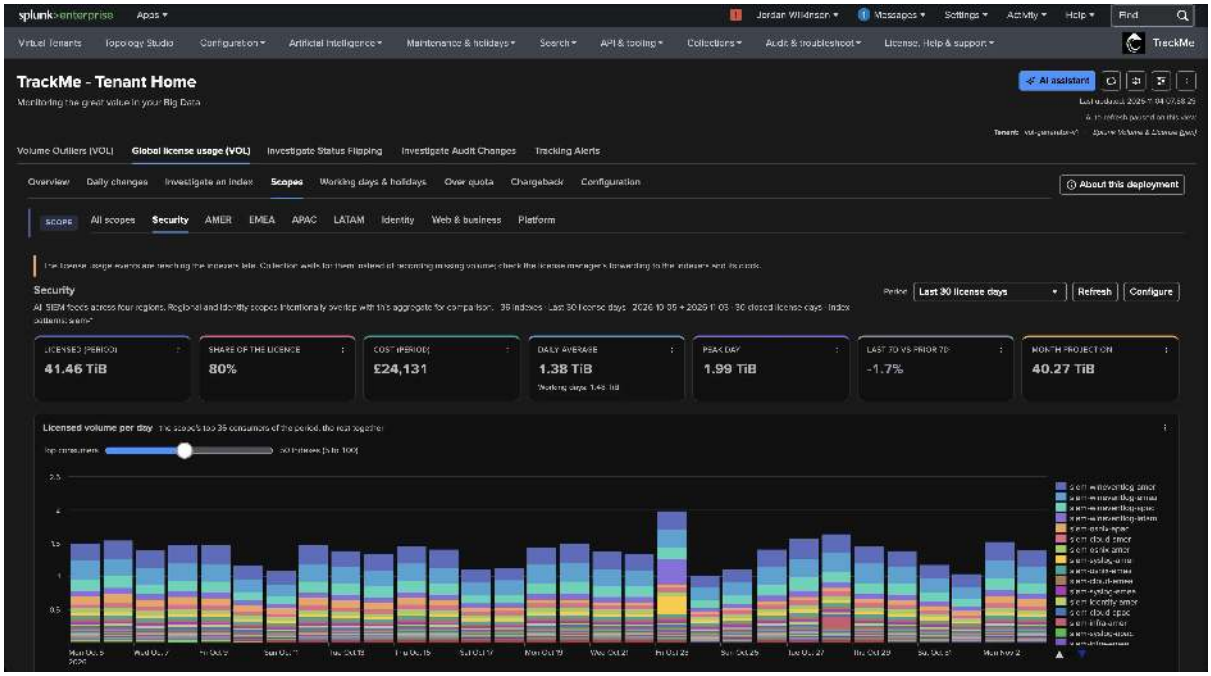
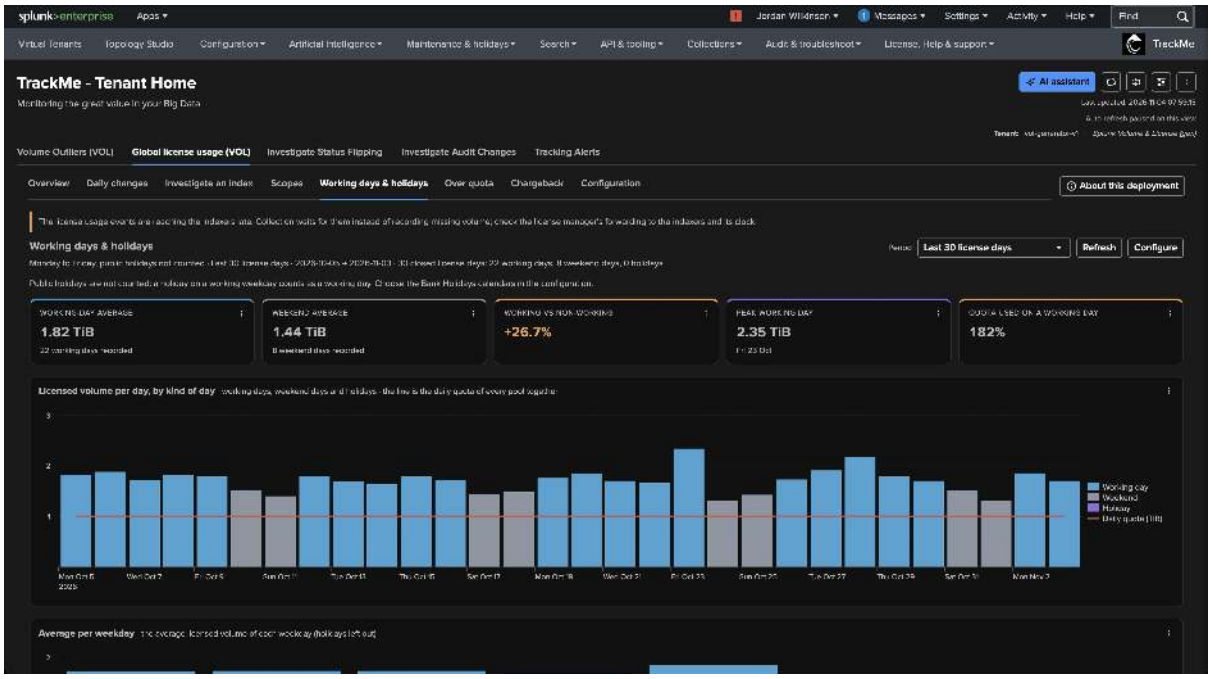
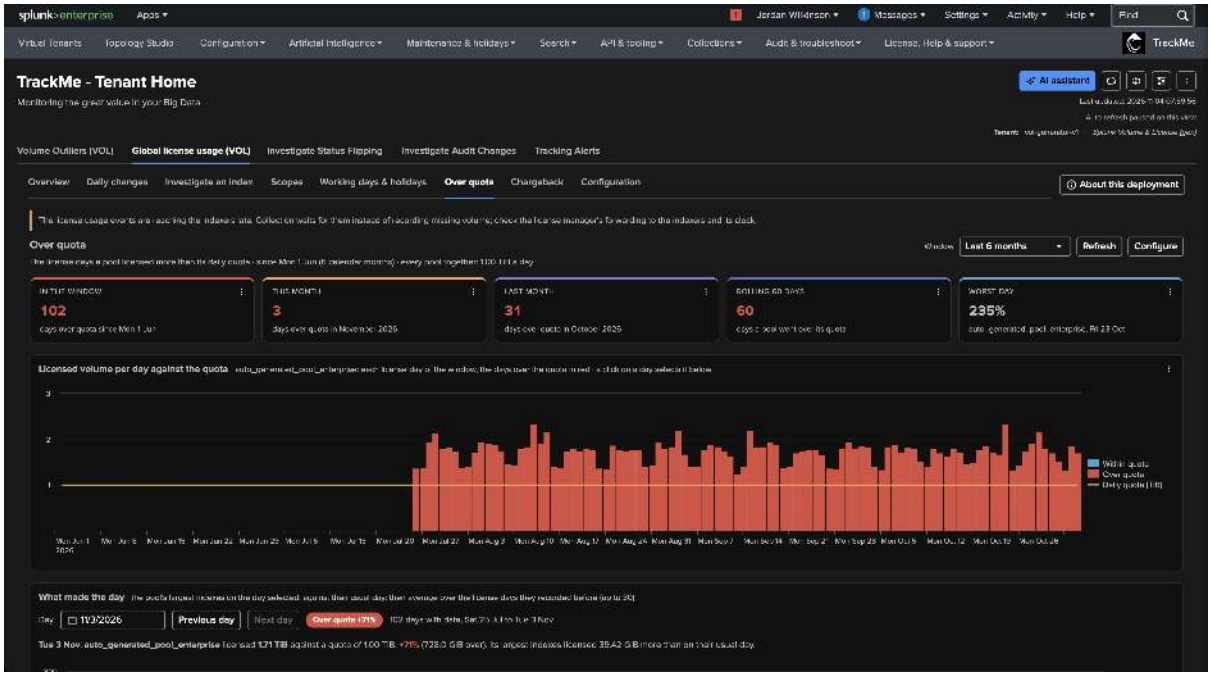
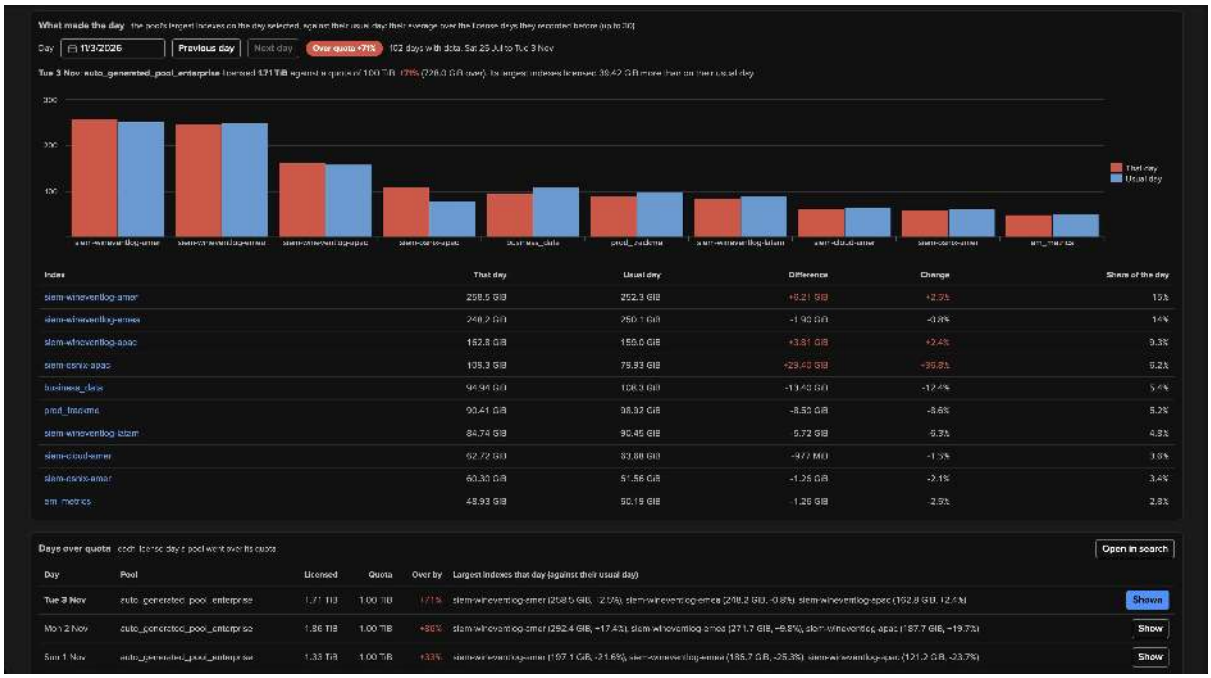
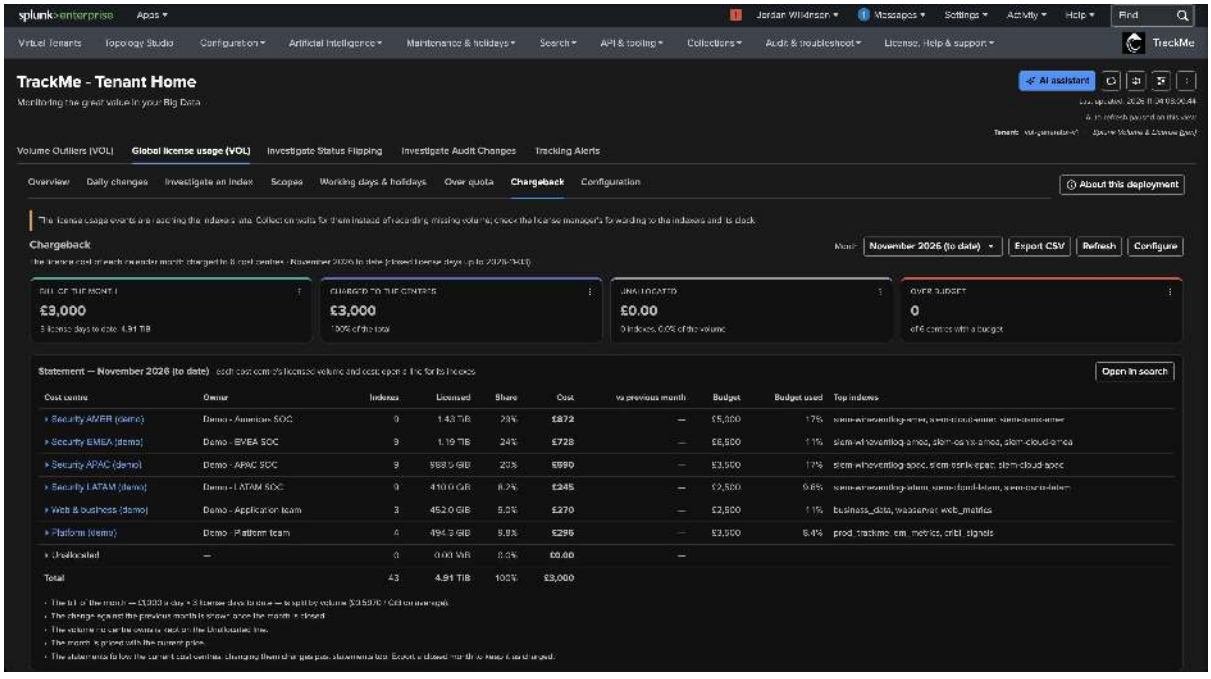
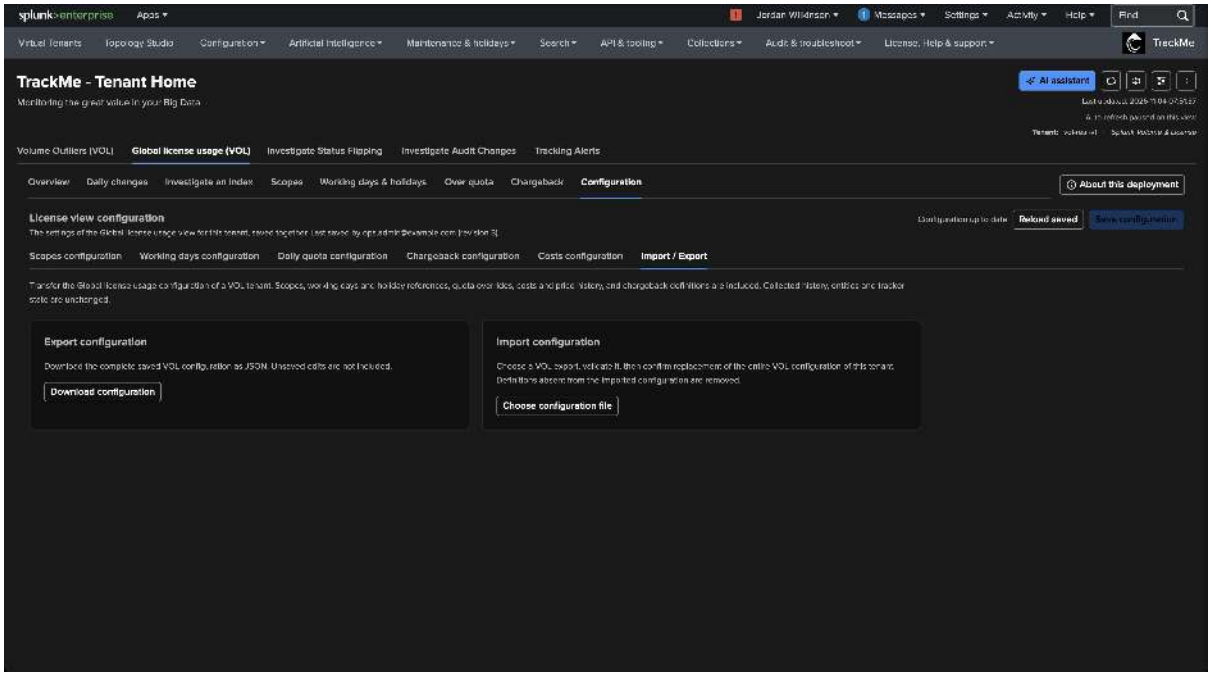
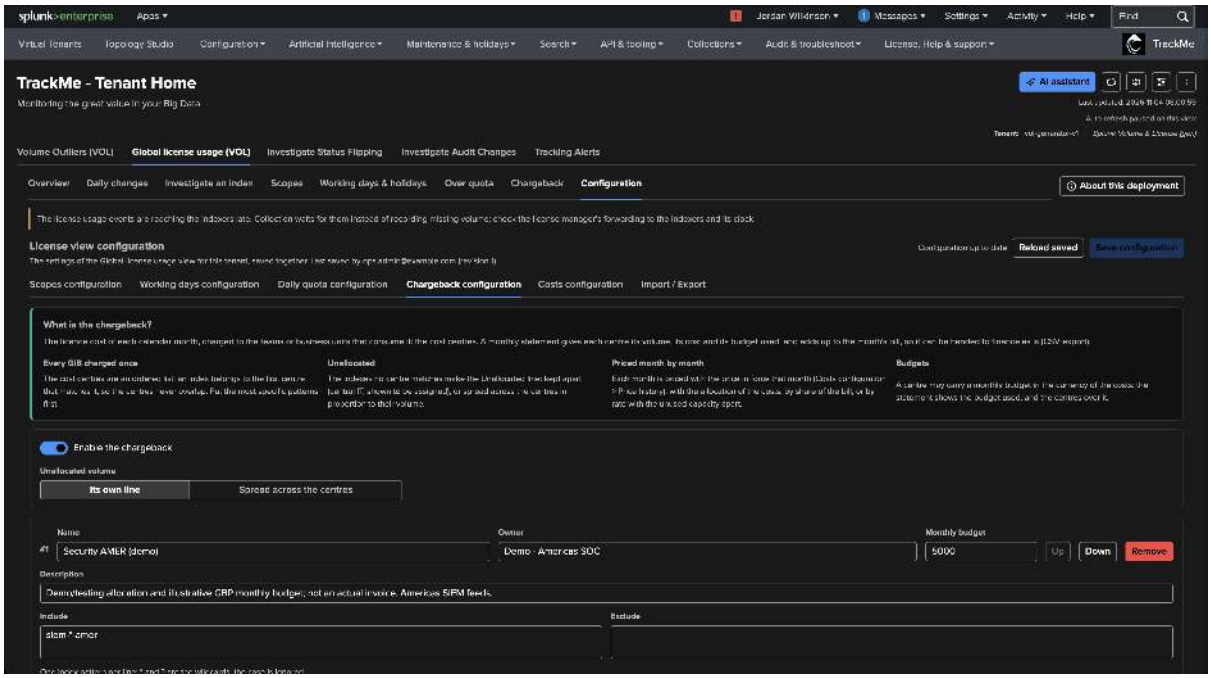
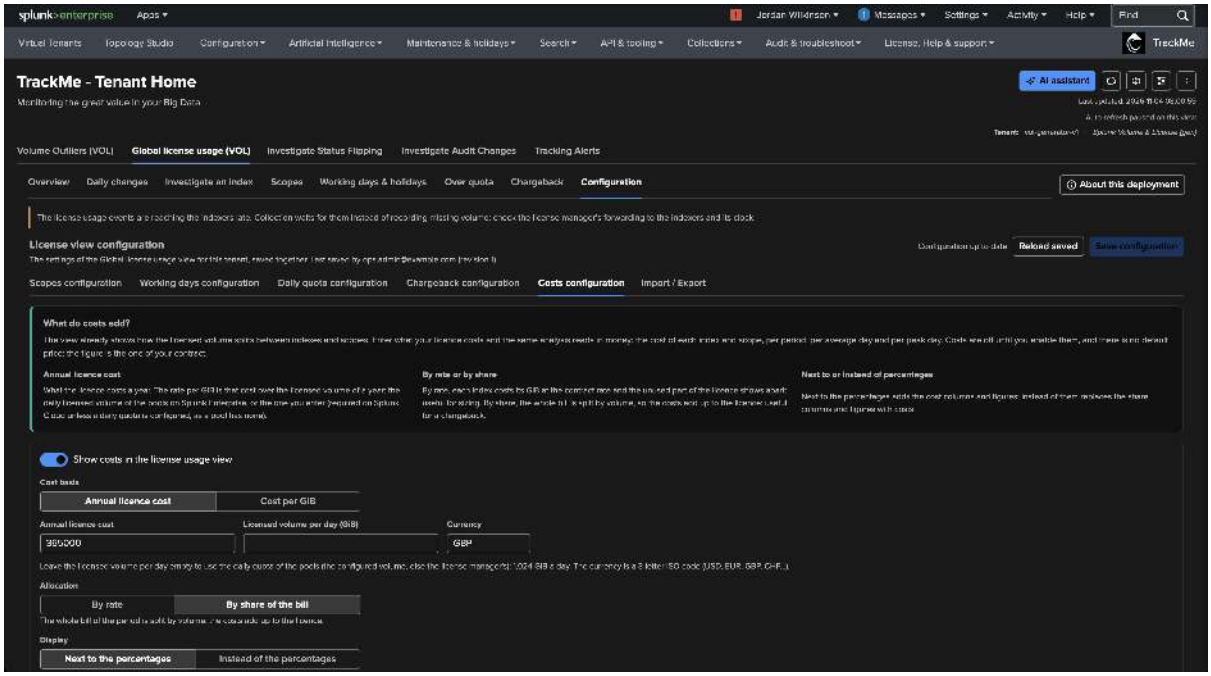
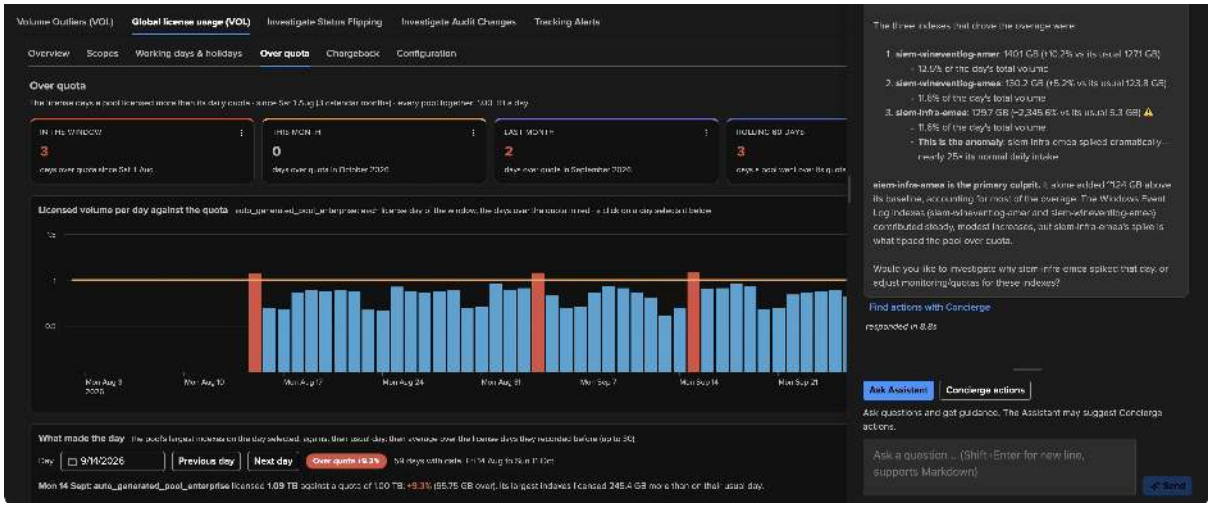
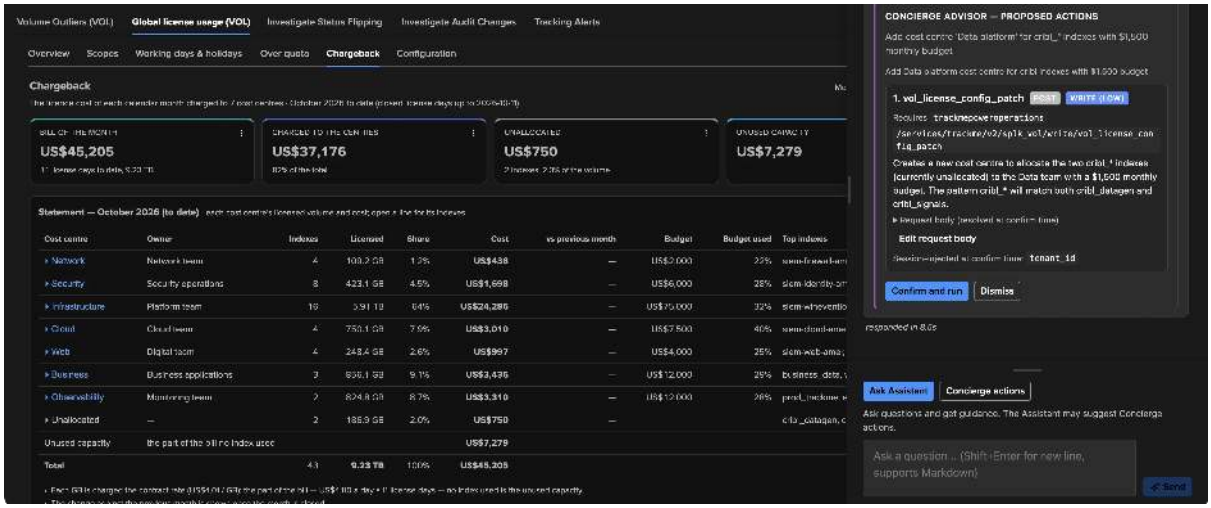
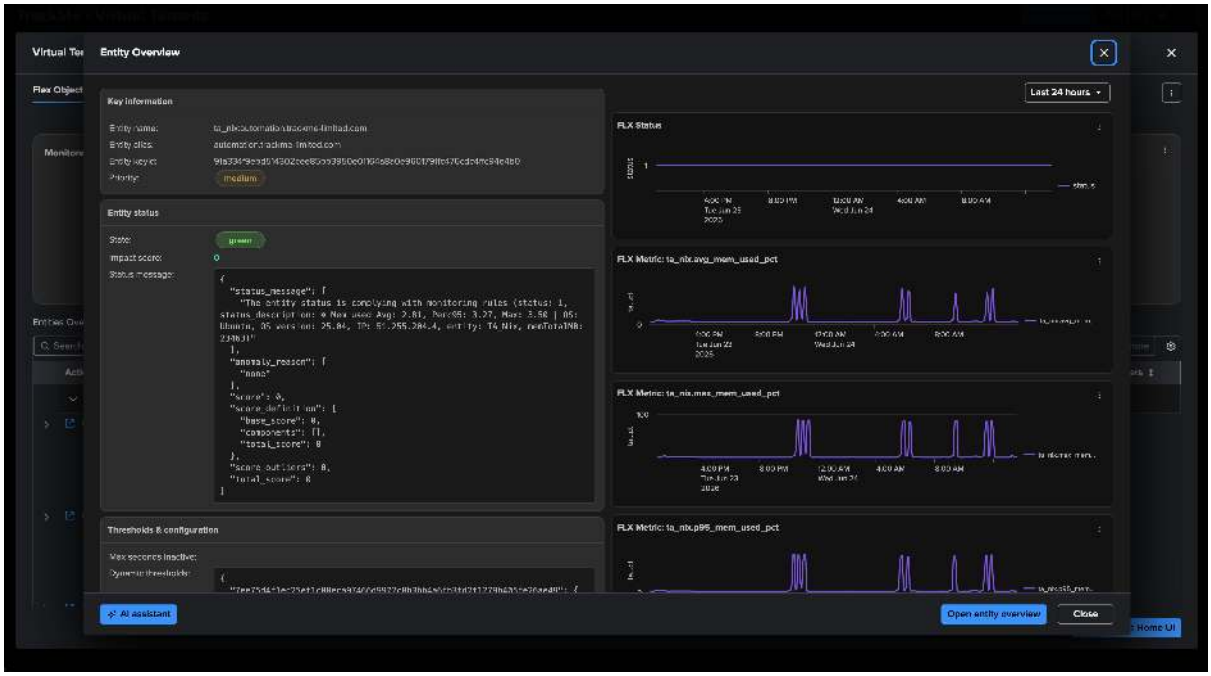
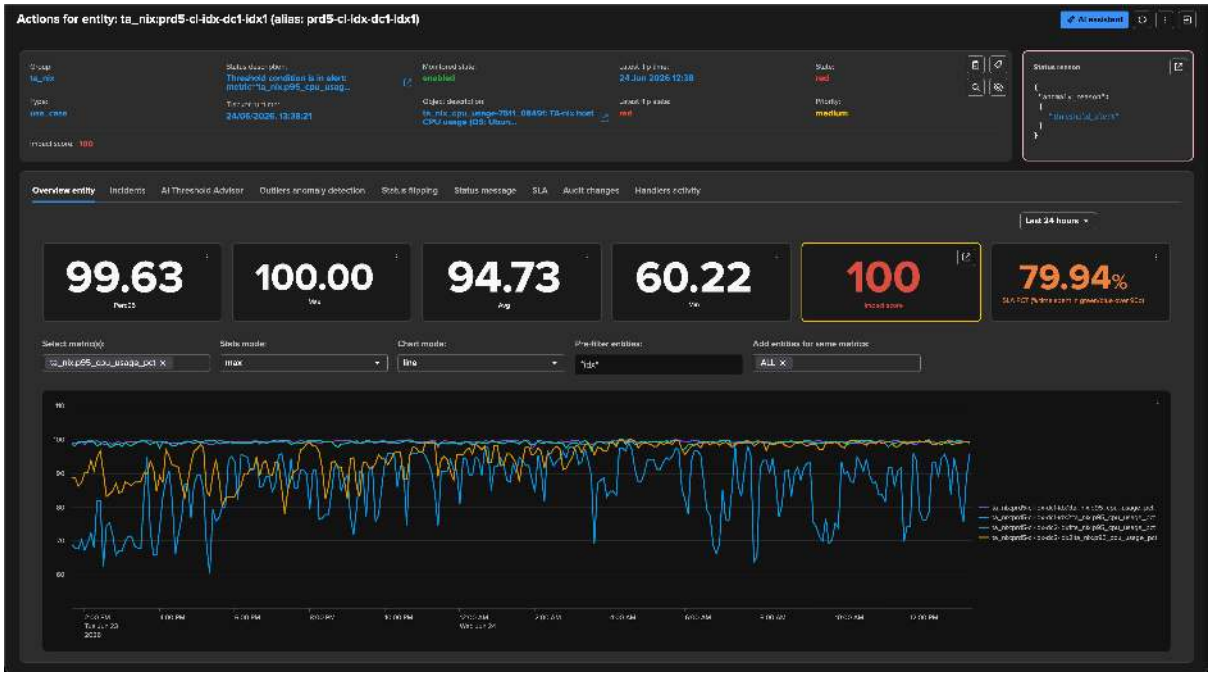
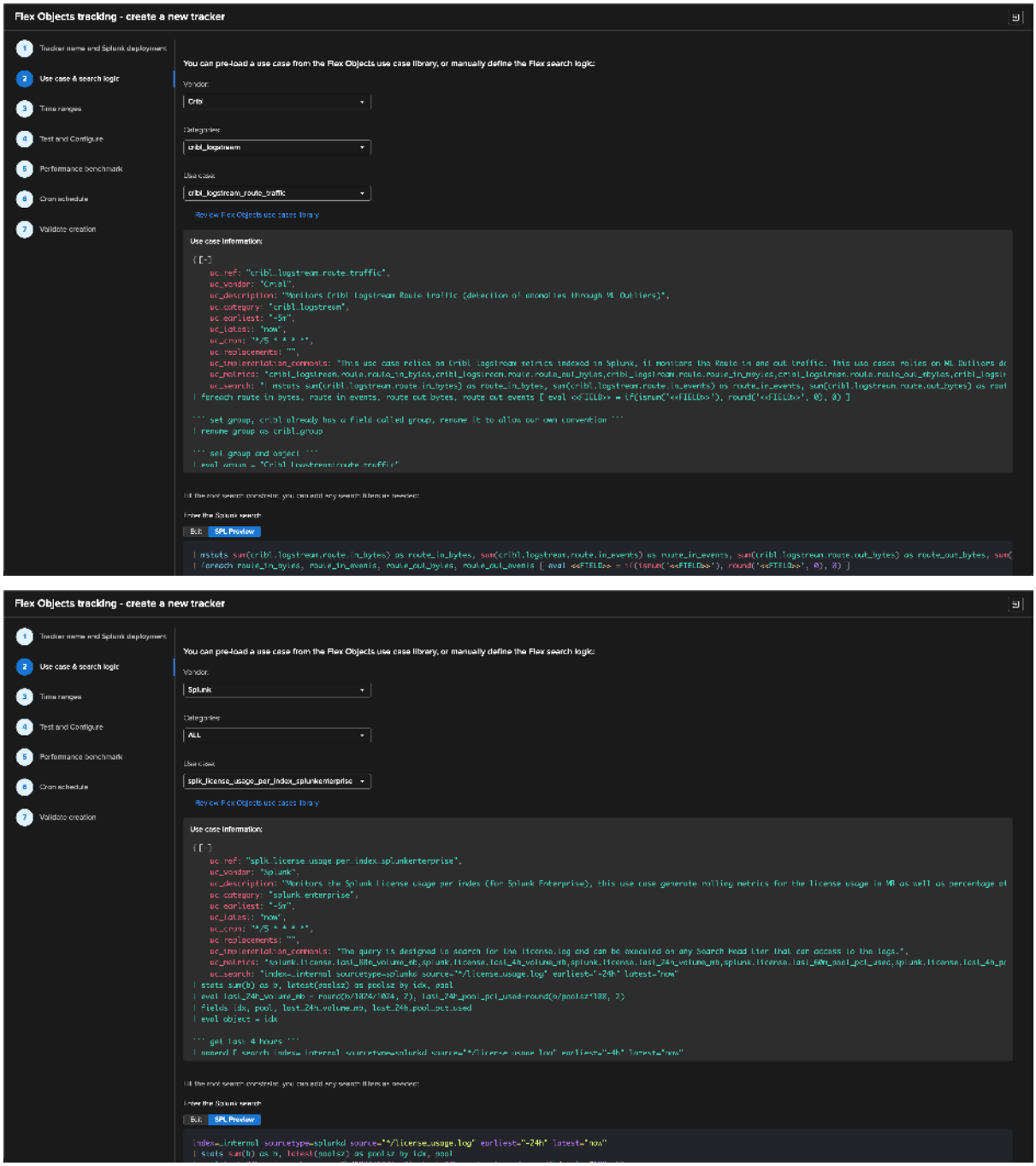
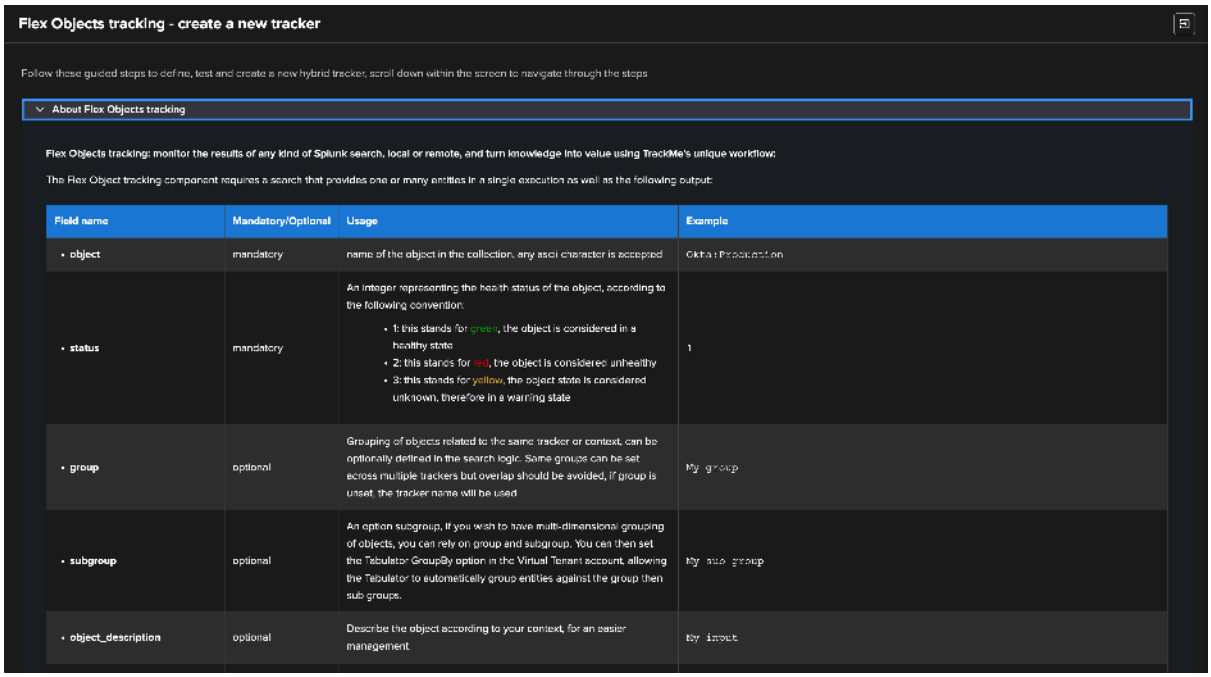
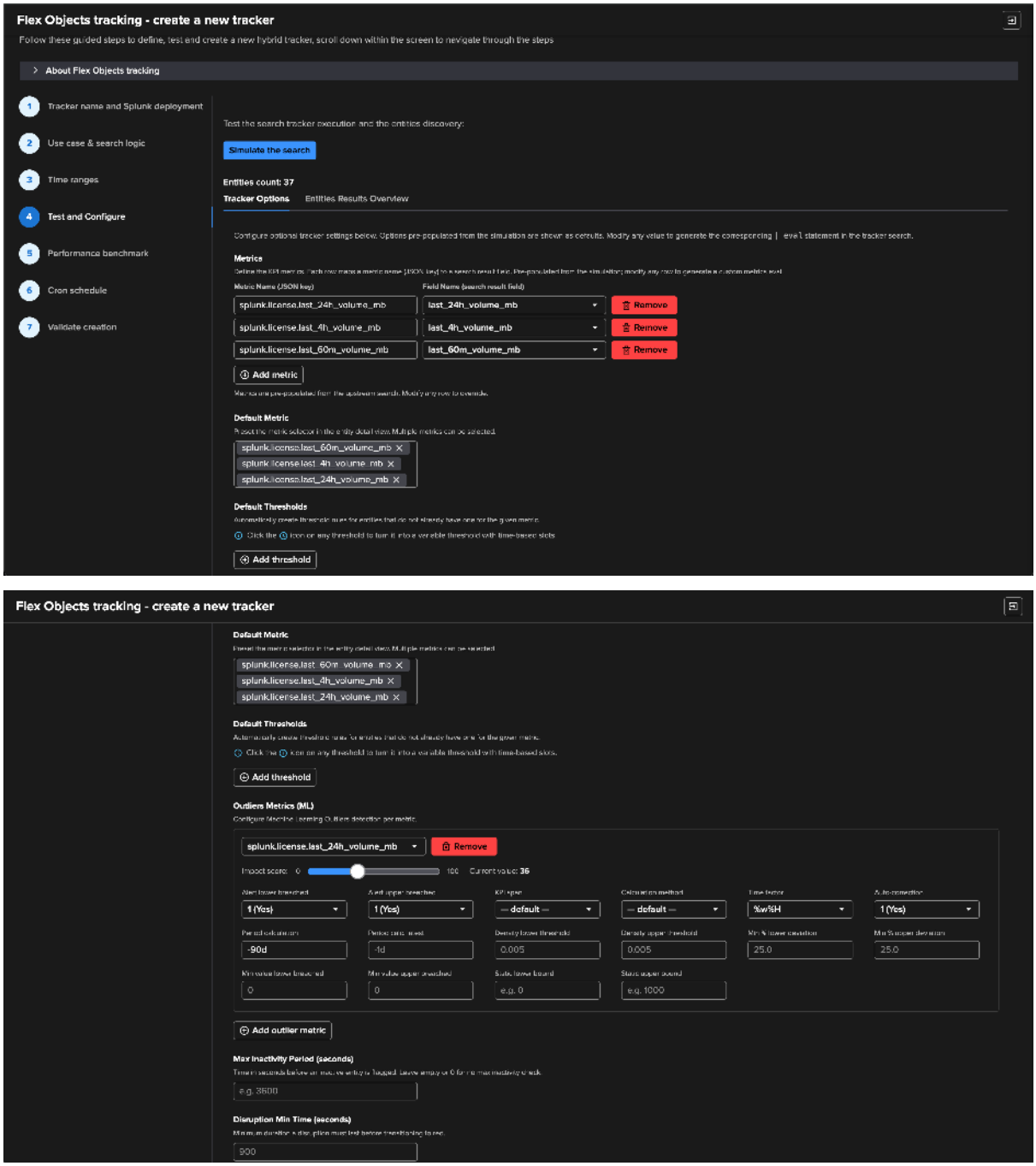
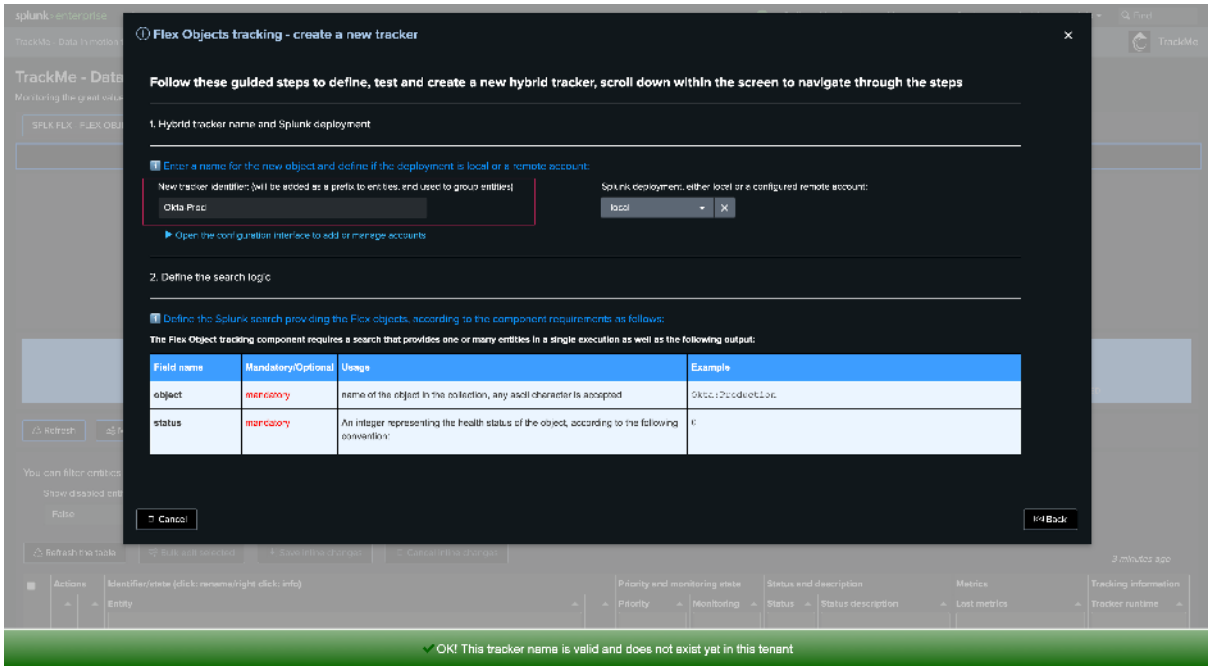
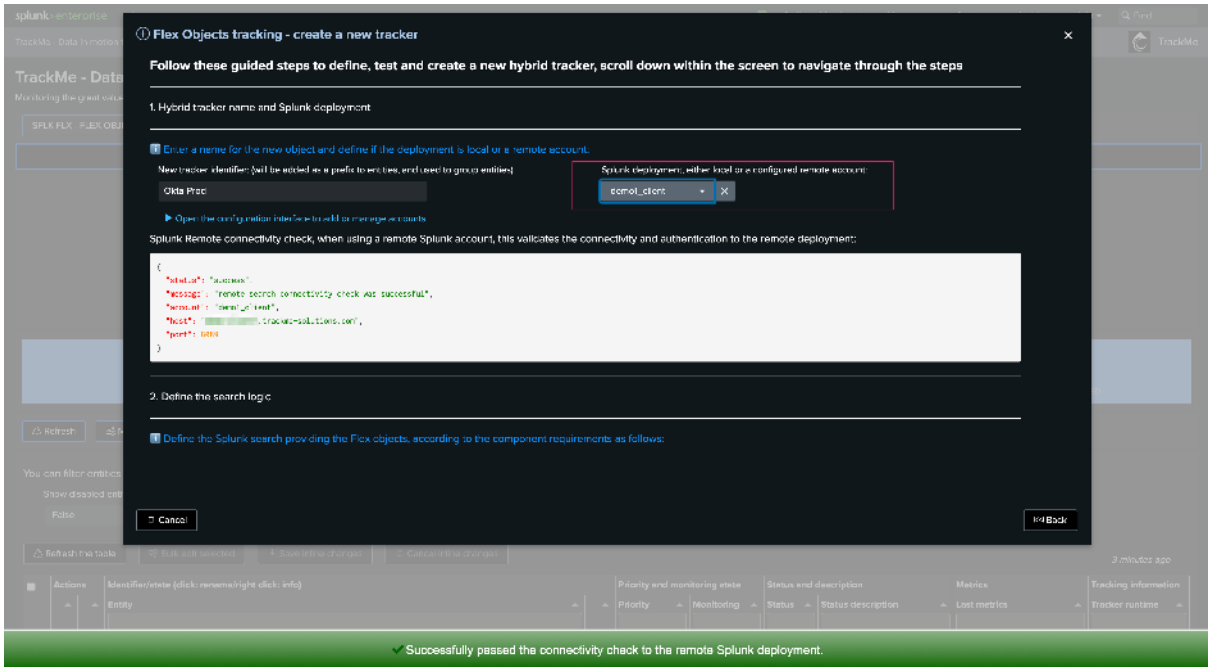
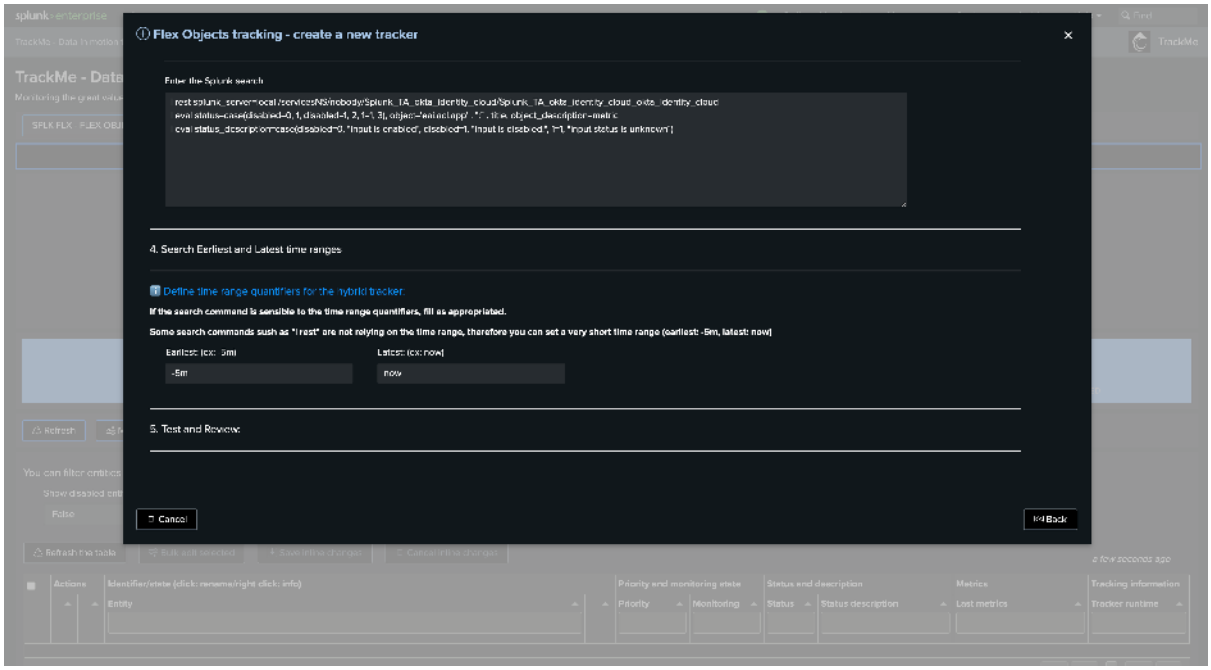
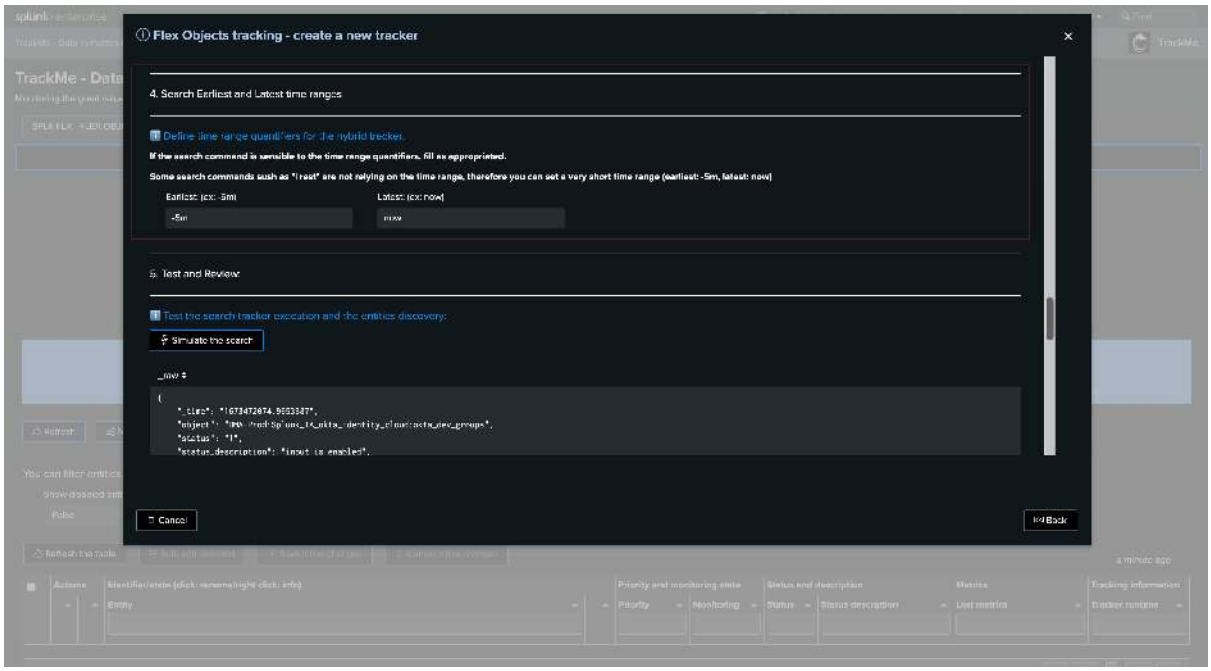
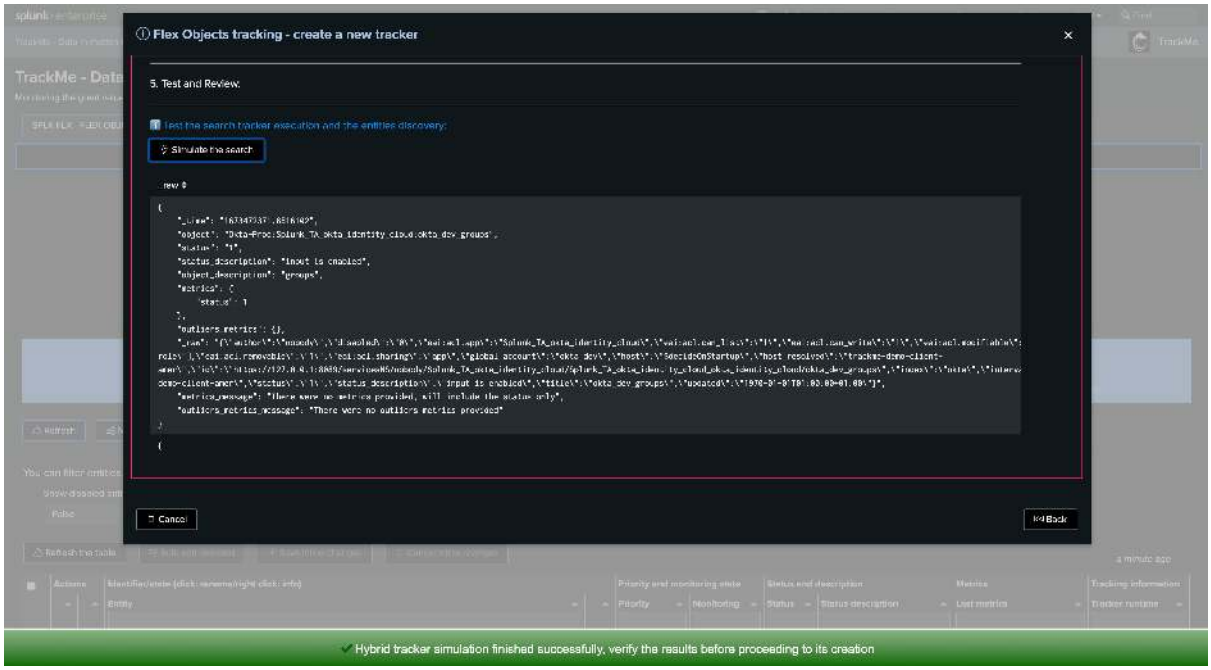
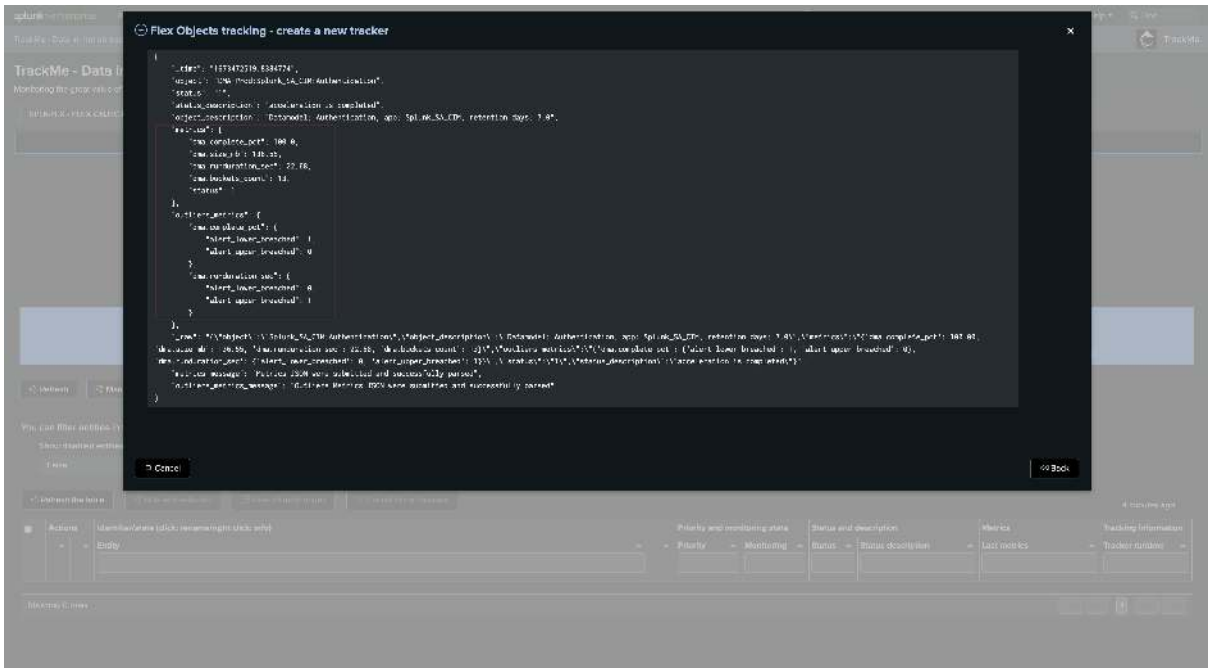
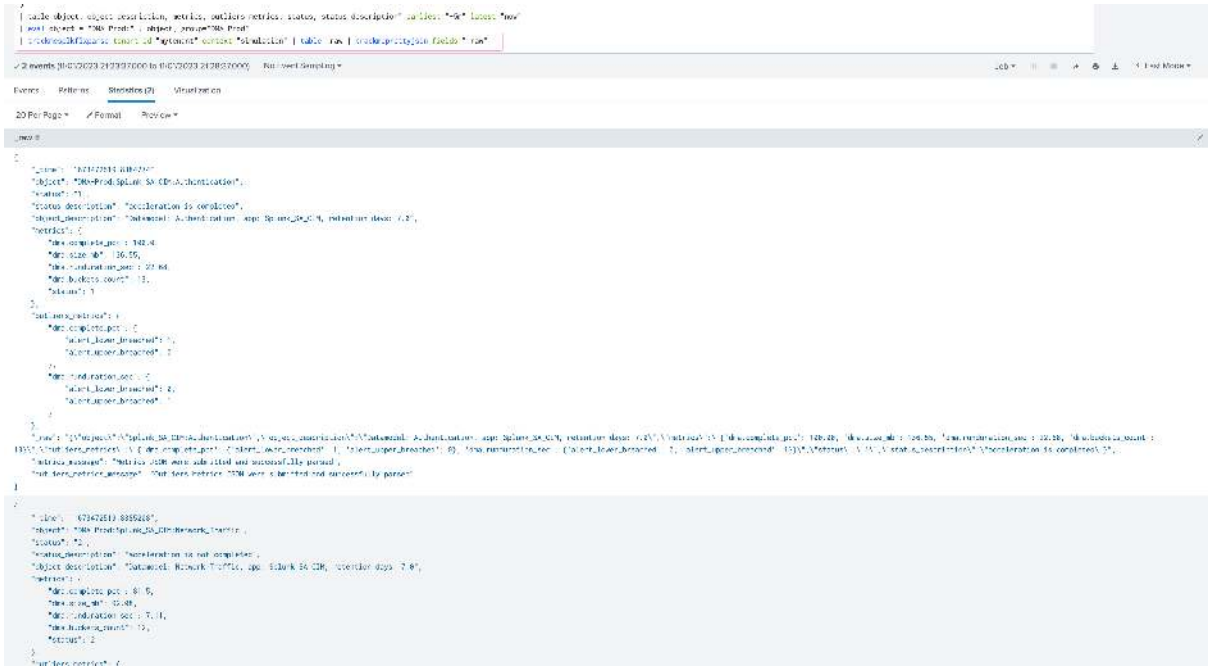
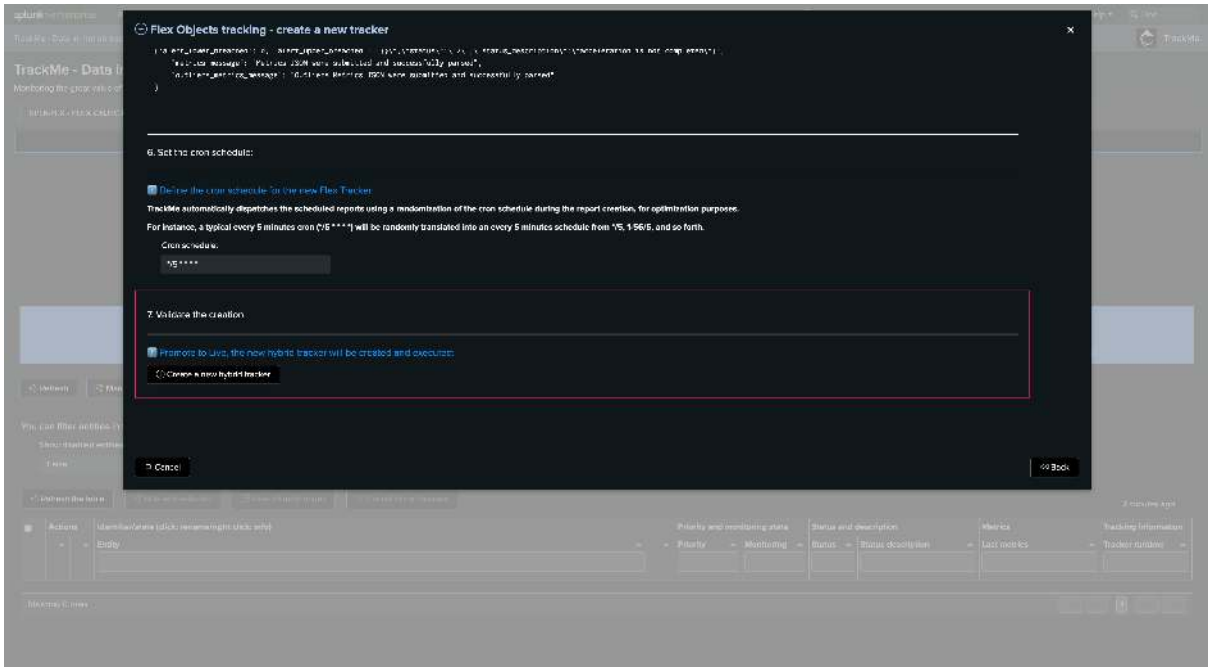
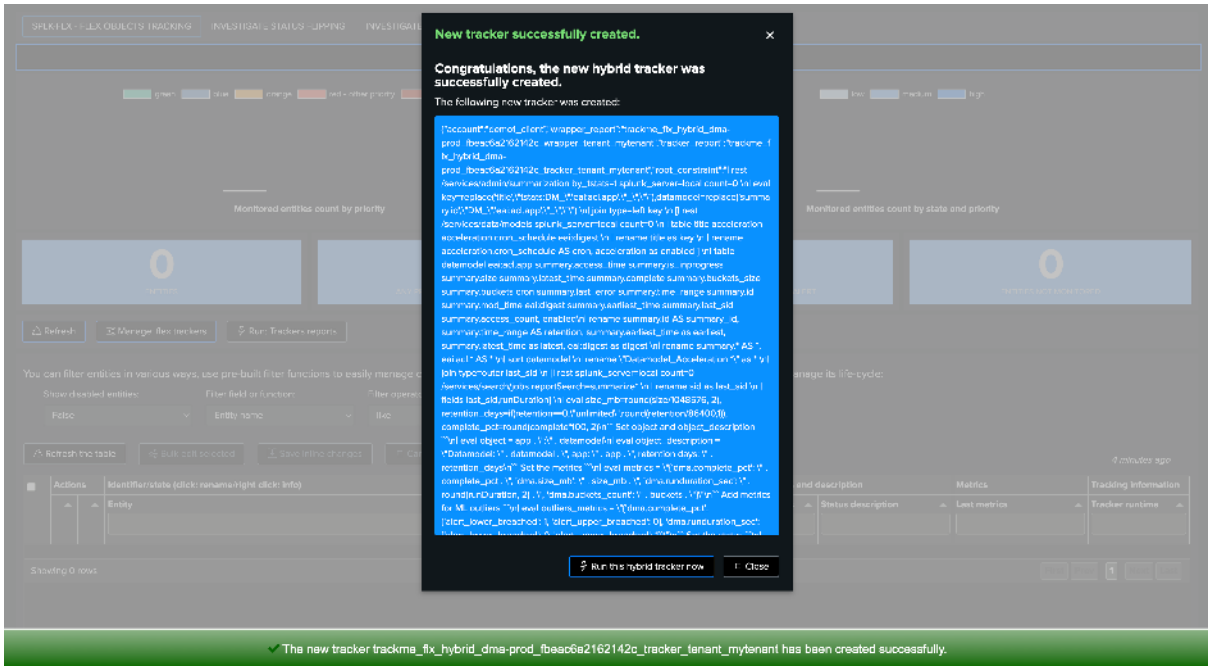
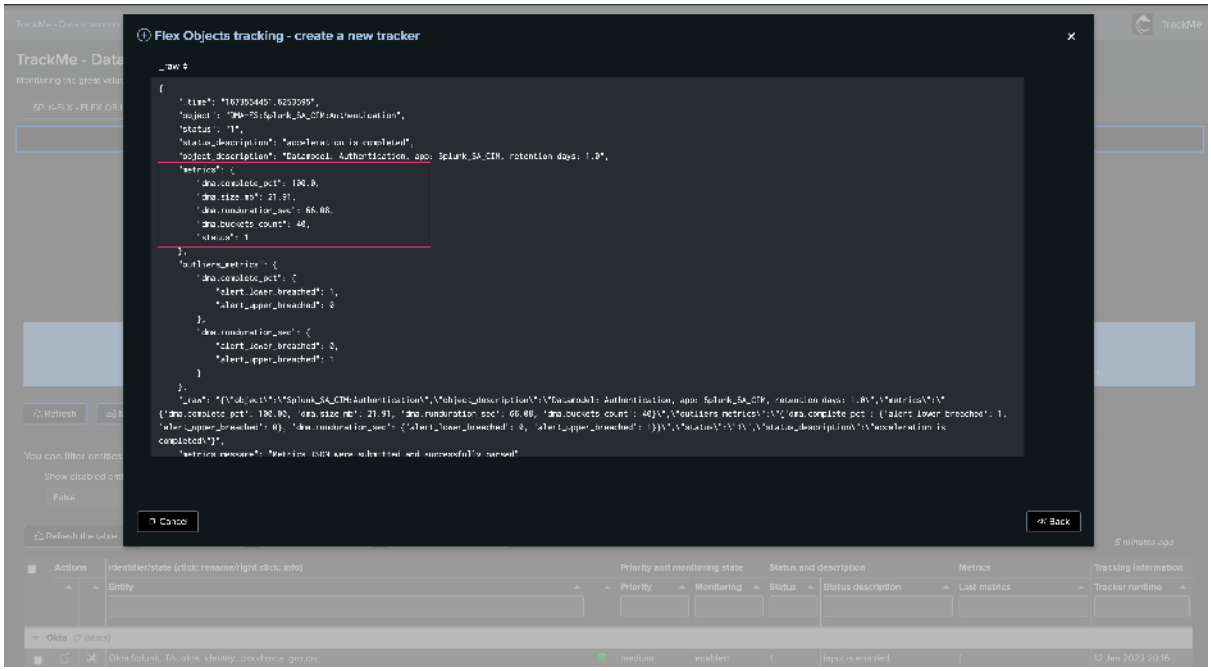
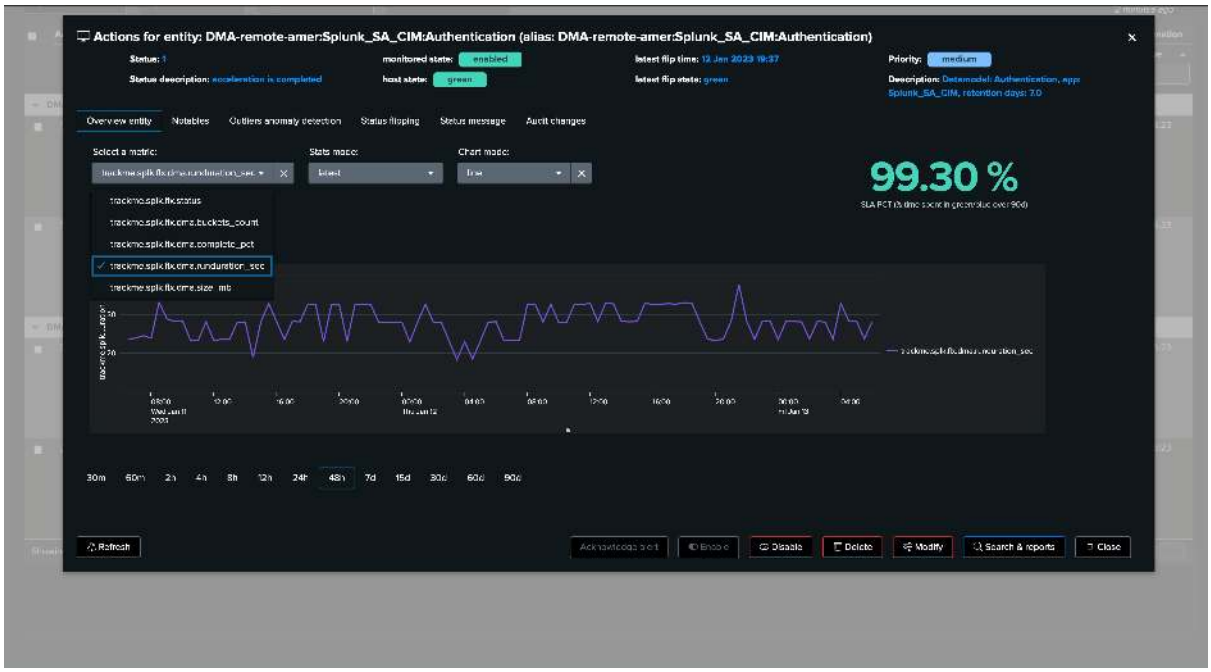
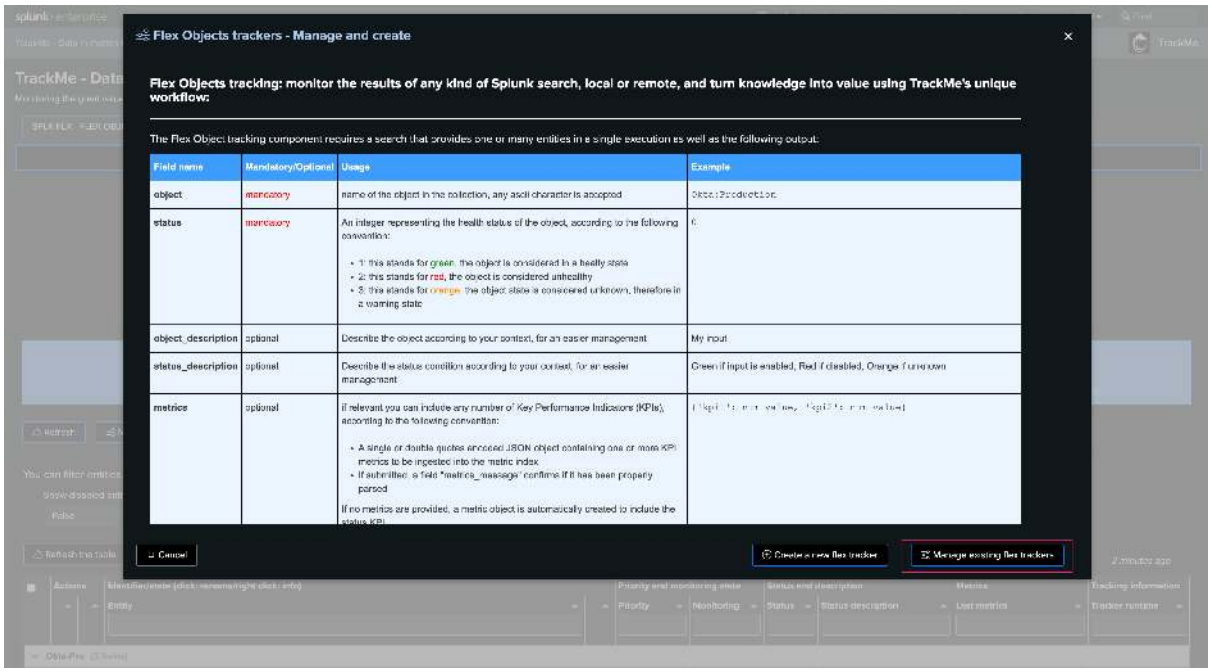
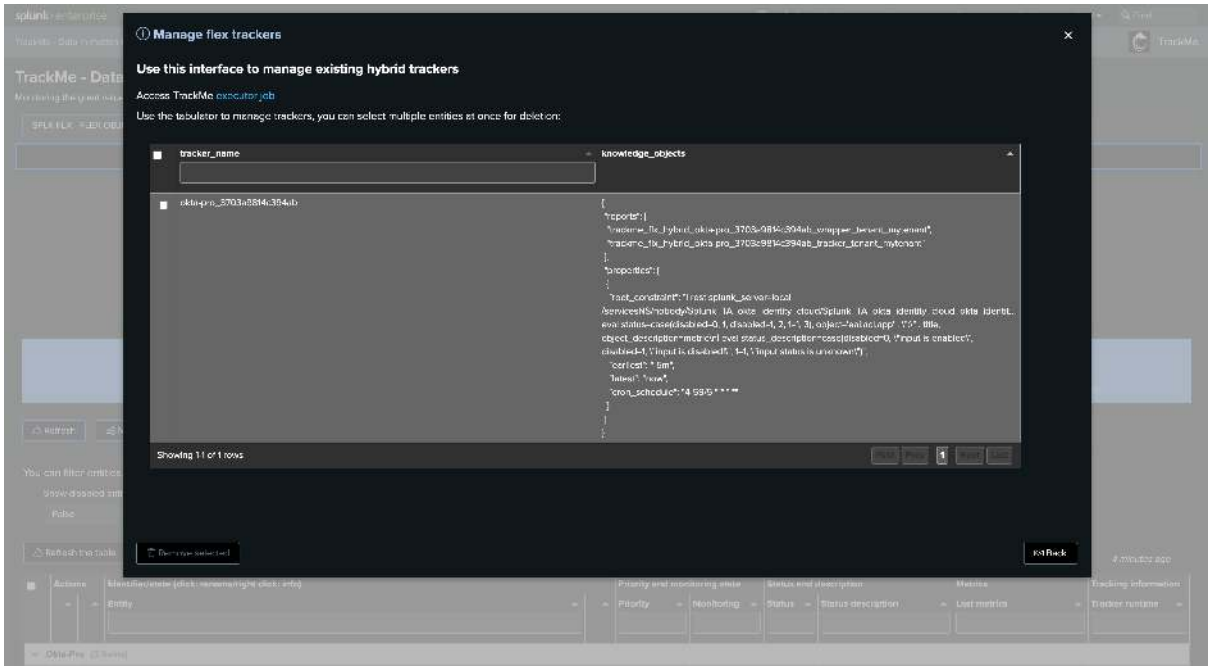
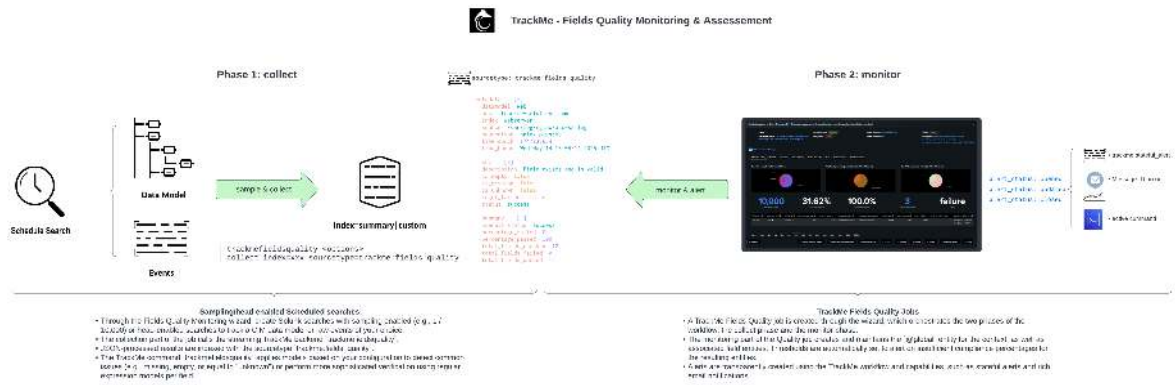
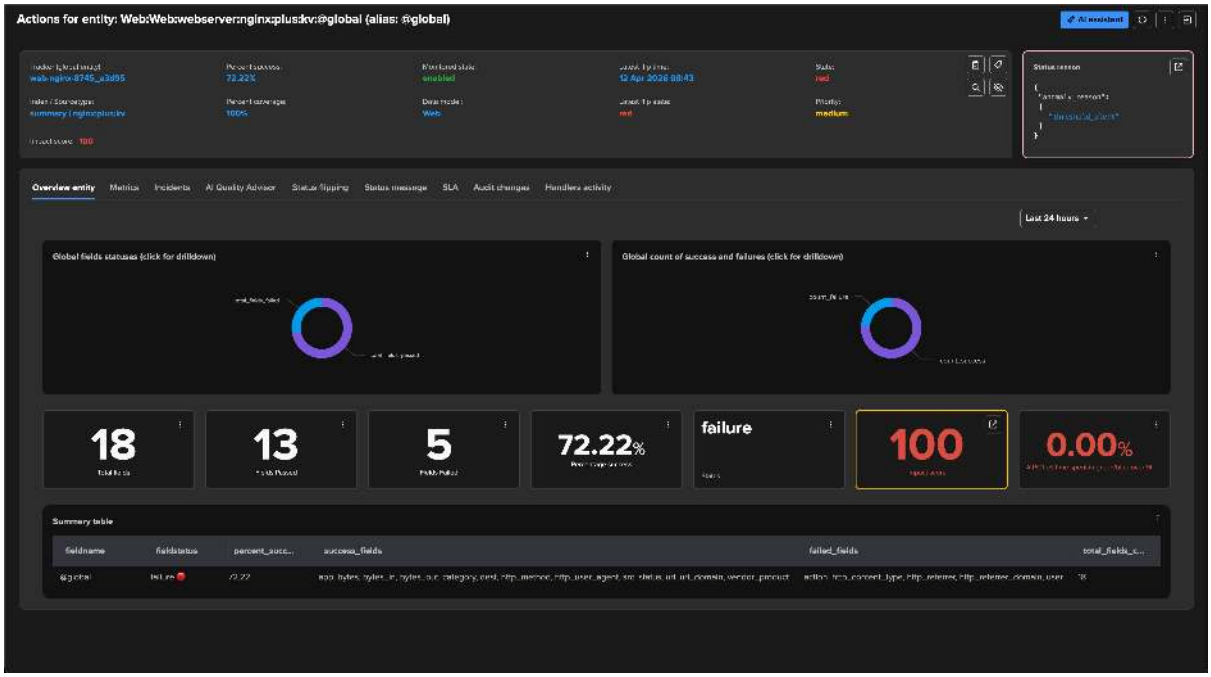
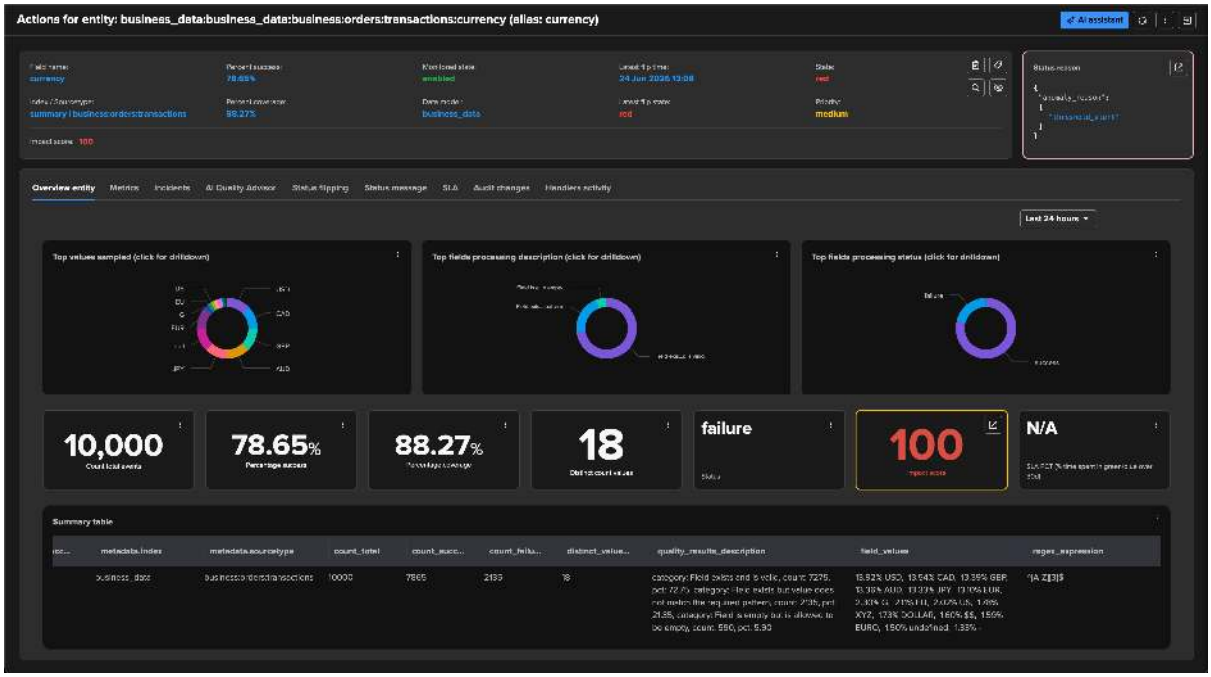
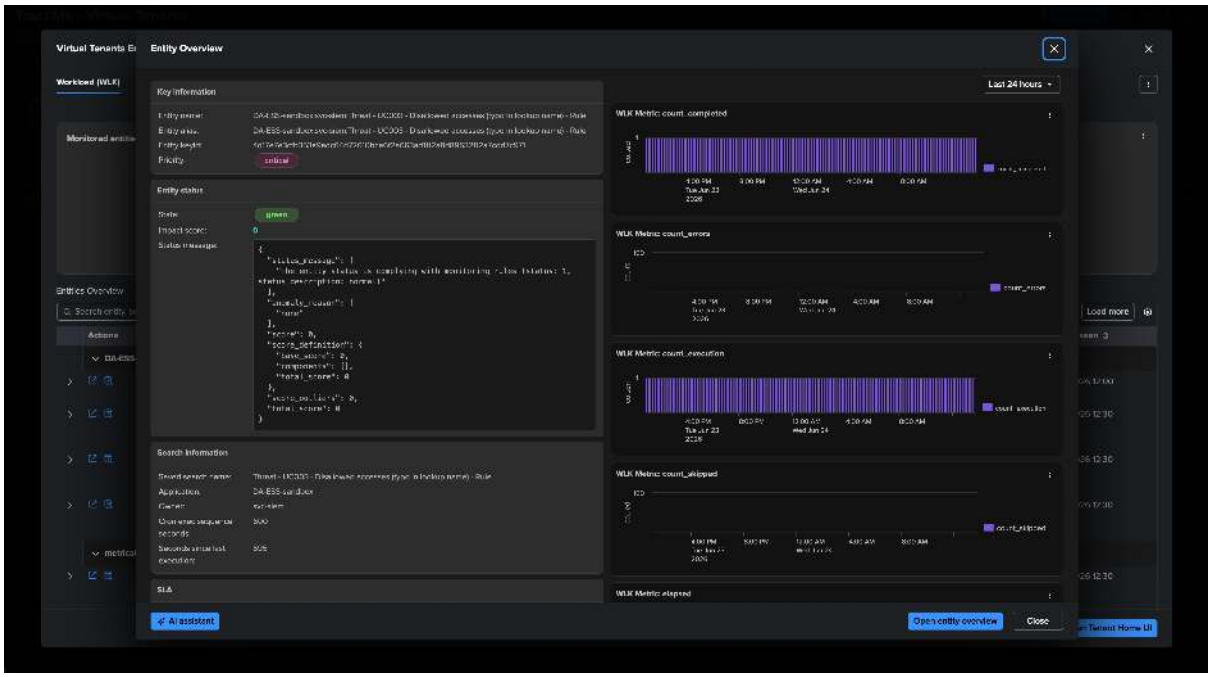
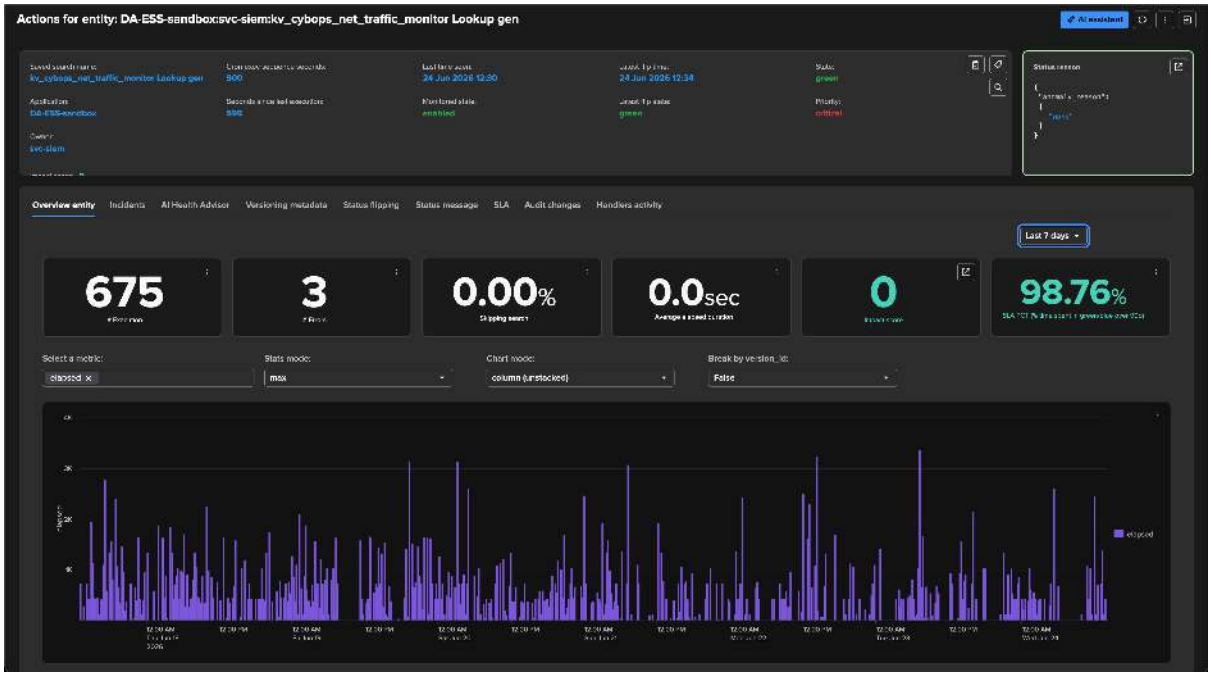
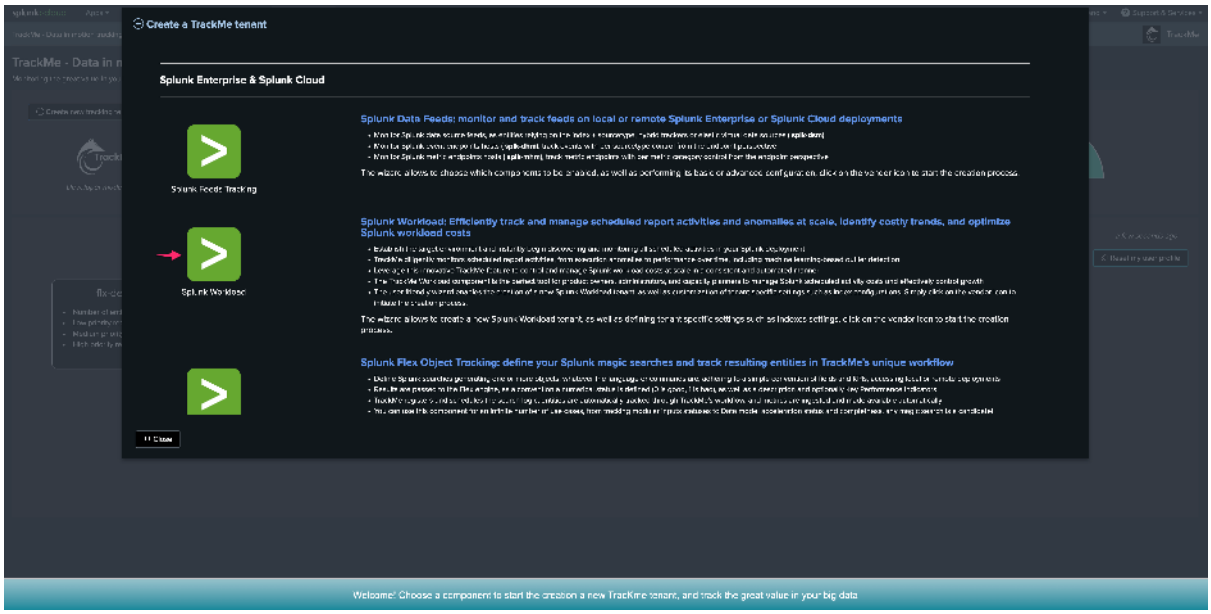
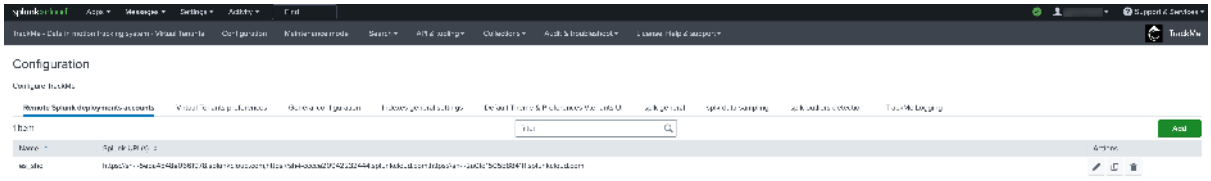
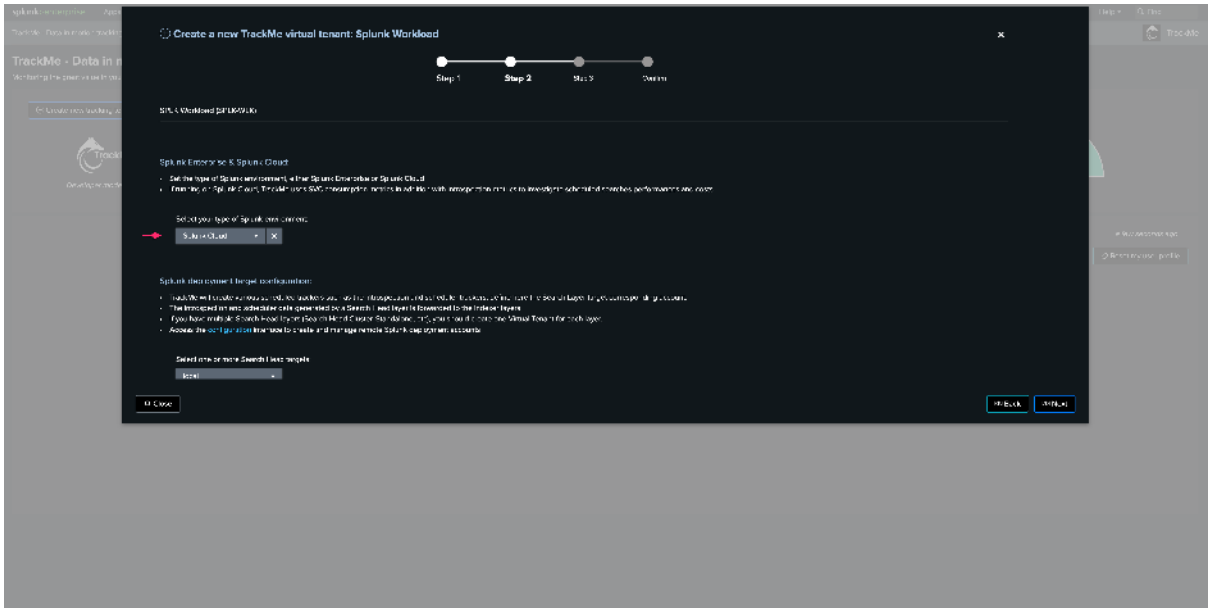
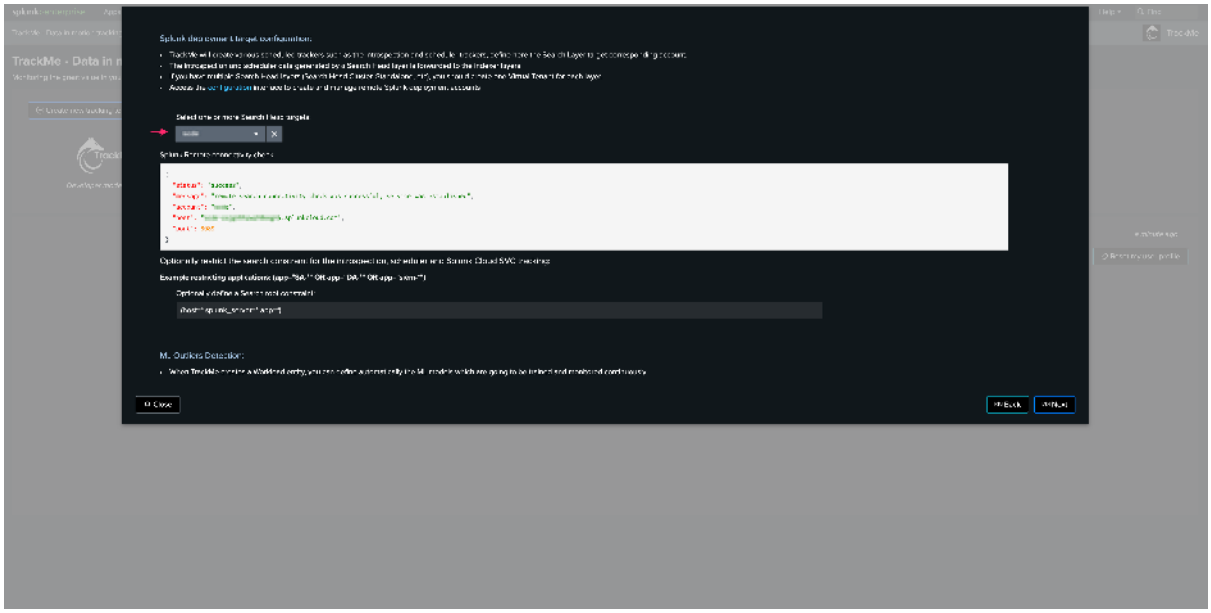
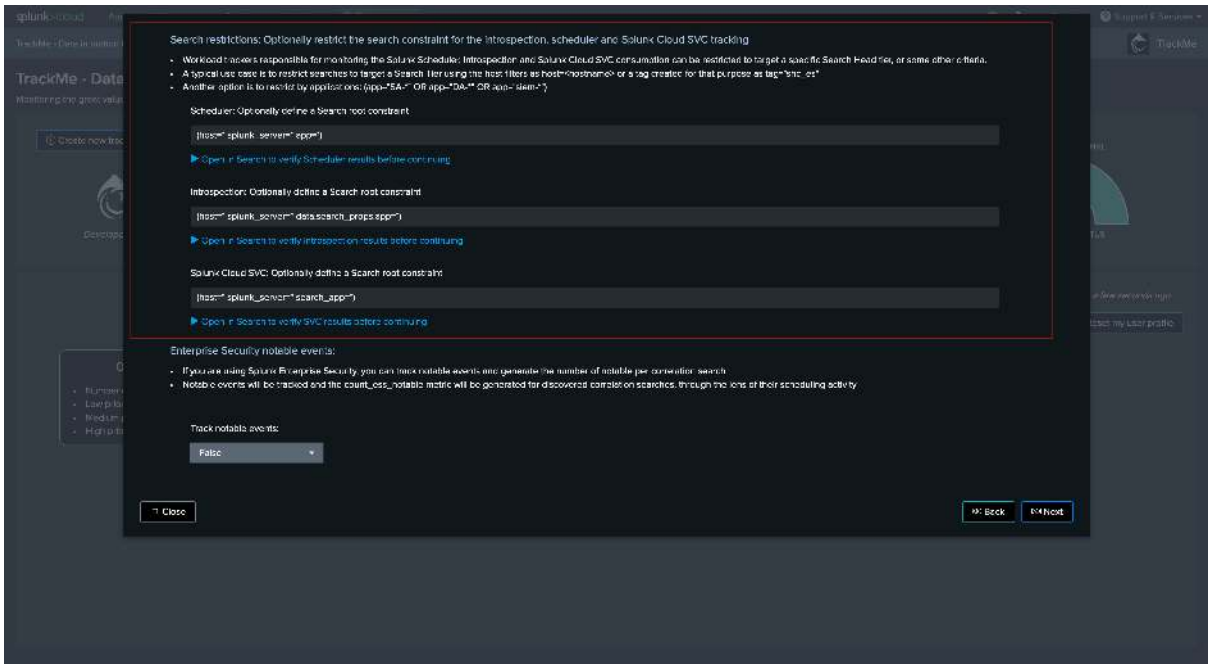
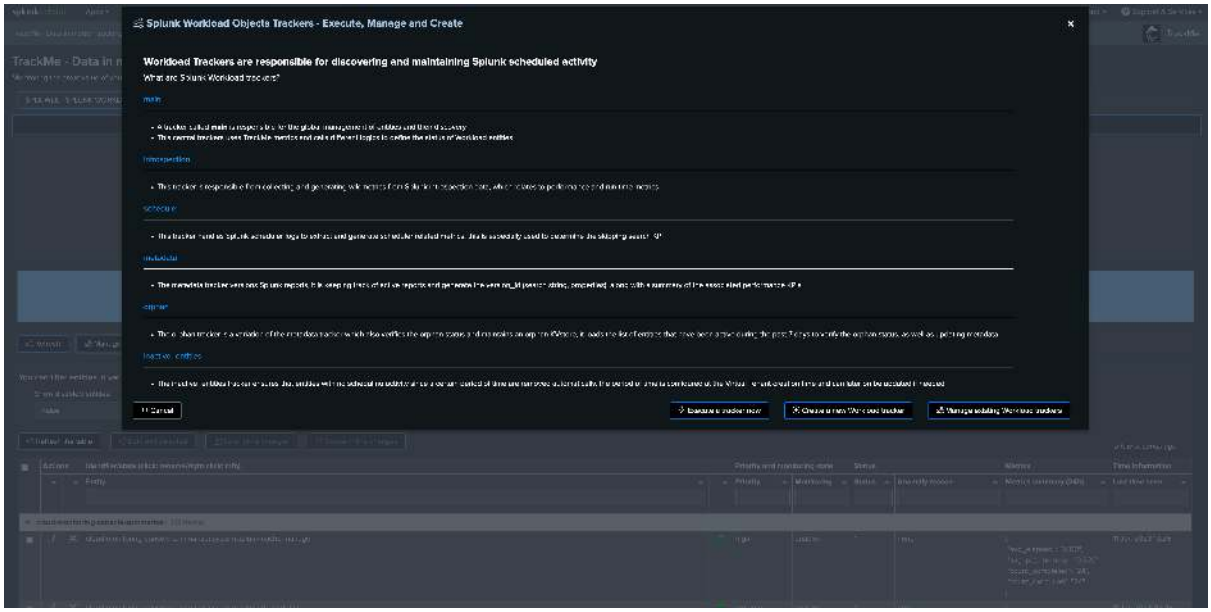
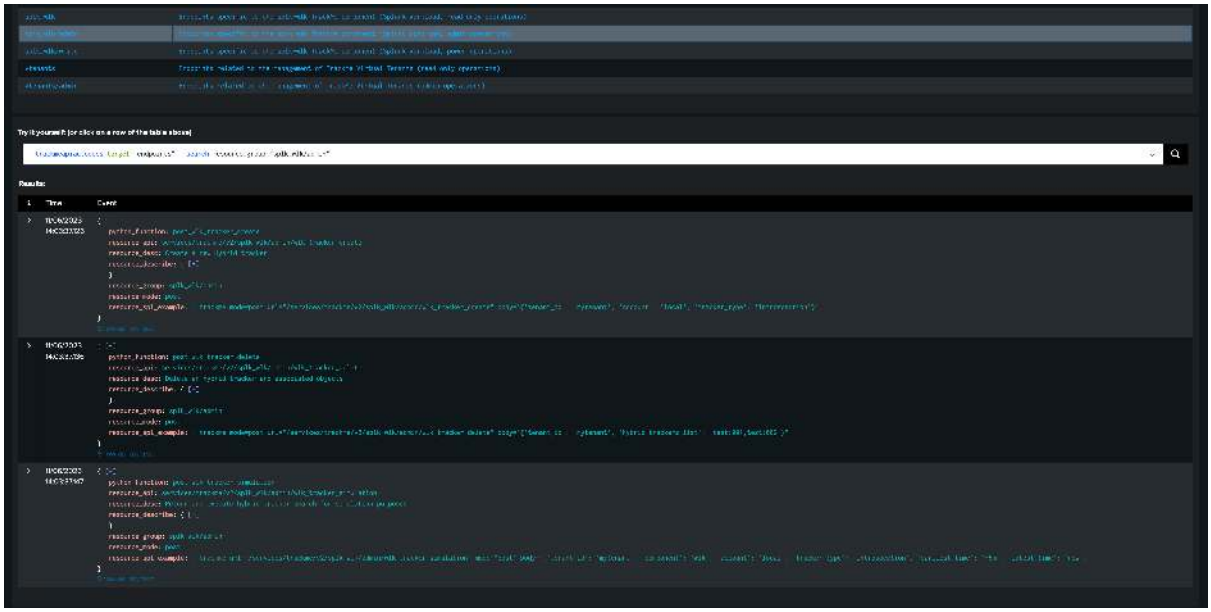
[trackme_summary]
coldPath = volume:primary/trackme_summary/colddb
homePath = volume:primary/trackme_summary/db
thawedPath = $SPLUNK_DB/trackme_summary/thaweddb

[trackme_audit]
coldPath = volume:primary/trackme_audit/colddb
homePath = volume:primary/trackme_audit/db
thawedPath = $SPLUNK_DB/trackme_audit/thaweddb

[trackme_metrics]
coldPath = volume:primary/trackme_metrics/colddb
homePath = volume:primary/trackme_metrics/db
thawedPath = $SPLUNK_DB/trackme_metrics/thaweddb
datatype = metric
```

Adapt with the volume definitions in your context.

Virtual Tenants Specific Indexes

A core concept of TrackMe is called Virtual Tenants, which provides many powerful features. Part of this concept provides the capability to define a per-tenant specific set of indexes.

Therefore, you can choose to define specific indexes for one or more tenants, and a different set of indexes for other tenants.

This allows TrackMe to comply with any Role-Based Access Control (RBAC) requirements, for instance, to allow a population of users to access specific tenants while another population can access others.

Example: Virtual Tenant indexes definition configuration screen:



7.1.3 Installing TrackMe

Splunk Enterprise

Installing TrackMe on Splunk Enterprise on-premises deployments depends on the type of deployment.

- For standalone instances, refer to: https://docs.splunk.com/Documentation/AddOns/released/Overview/Singleserverinstall
- For distributed deployments, refer to: https://docs.splunk.com/Documentation/AddOns/released/Overview/Distributedinstall

Notes for Older Generations of Splunk (prior to Splunk 9.x)

TrackMe is developed for the latest generation of Splunk software; therefore, some built-in parameters are primarily targeting Splunk 9.x compatible configurations.

If you are running TrackMe on a version prior to Splunk 9.x, you should apply the following customizations:

Create a local/distsearch.conf:

```
# distsearch.conf

# Avoid the replication to the indexers of the KVstore backup tarball compressed␣
↪files
[replicationBlacklist]
trackme_backup_tgz = apps/trackme/backup/*.tgz
trackme_backup_dirs = apps/trackme/backup/...

# These lookups do not need to be replicated
trackme_cim_regex = apps/trackme/lookups/trackme_cim_regex.csv

# Machine Learning models: Anomaly detection will generate various ML models␣
↪files, these are not needed on the indexer layers
trackme_mlmodels = apps/trackme/lookups/__mlspl_*.mlmodel
```

Notes:

- In Splunk 9.x, biased language has been addressed; this stanza became replicationDenylist.

- In TrackMe, this stanza prevents ML models files from being unnecessarily replicated to the indexers and being part of the knowledge bundle.

Splunk Cloud

Installing TrackMe on Splunk Cloud relies on Cloud self-services. Refer to:

- https://docs.splunk.com/Documentation/SplunkCloud/latest/Admin/SelfServiceAppInstall

7.1.4 Upgrading TrackMe

In summary, upgrading TrackMe follows the same process as installation:

- In Splunk Enterprise, you will download the updated release, extract the new version, restart the instance if in a standalone Search Head, or apply the SHC bundle if running in an SHC.
- In Splunk Cloud, when a new release has been published and vetted, the new version release number appears as upgradable through the application management interface; you will therefore follow the self-services process.

TrackMe implements an automated concept to perform required application-level upgrade procedures, called `schema version`. The complete upgrade guide, including a step-by-step procedure with checkpoints, the post-upgrade validation and the rollback rules, is in *Upgrading TrackMe*.

7.2 Upgrading TrackMe

Upgrading TrackMe is a standard Splunk application upgrade followed by an automated, per-tenant **schema migration** that TrackMe runs on its own. The package replaces the shipped code; TrackMe then reshapes each Virtual Tenant's knowledge objects and KVstore collections to the new release, tenant by tenant, and reports its progress in the user interface.

This section is the upgrade guide. It is written to be executed, not only read: the procedure page is a step-by-step runbook with the screen to look at, the search to run and the expected result at every step, and the other pages hold the reference material behind it.

Upgrade procedure The step-by-step runbook: before the upgrade, upgrade day, after the upgrade, with checkpoints.

Upgrade procedure Schema migrations How the automated per-tenant migration works, how to follow it, what runs between releases.

Schema migrations Validating after an upgrade The built-in validation screens, the extended checklist, the behaviour changes to brief operators on, the evidence searches.

Validating after an upgrade Rollback Why a rollback is a restore and never a re-install of the older package, and how to do it.

Rollback



The upgrade procedure at a glance. Step 0 is done once per estate; steps 1 to 3 run on every TrackMe deployment.

7.2.1 Upgrade procedure

This page is the runbook. It is organised in four steps: **step 0** is done once per estate (or once, for a single deployment), **steps 1 to 3** run on every TrackMe deployment, and **step 4** applies only if a rollback is decided. Each step ends with a checkpoint.

In every search of this page, replace `<component>` with the component acronym (`dsm`, `dhm`, `mhm`, `flx`, `fqm`, `wlk`) and `<tenant_id>` with the Virtual Tenant identifier. Screenshots come from a two-tenant lab; replace tenant and entity names with your own.

Step 0 — Inventory and readiness

Deployment inventory

One record per TrackMe deployment. It is the input of the planning and the header of the validation record (see *Validating after an upgrade*).

Field	Where it comes from / notes
Deployment name	Your internal reference
Splunk topology	Standalone search head, distributed (search head + indexers), search head cluster (with deployer), or Splunk Cloud
Splunk version	From the search head; see *Compatibility* for the supported range
TrackMe version and schema	Virtual Tenants header, and the **Tenants Update statuses** screen behind the version number (step 1.4)
Tenants and components	Number of Virtual Tenants; components enabled per tenant
ML Outliers in use?	Yes / no per tenant (the search of step 1.3 returns rows). Decides whether steps 1.3 and 3.3 apply
Alert actions and integrations	Stateful alerts, notables, email delivery accounts, free-style REST actions, SOAR: what your downstream tooling consumes (the target of the step 3.2 test)
Custom knowledge objects	Your own saved searches, dashboards or lookups built on TrackMe data
Change window and contact	Agreed window; the person who confirms the downstream side of the step 3.2 test

Platform readiness

- Splunk version within the supported range for the target release; a splunkd restart is possible in the change window (Splunk Enterprise).
- KVstore healthy on the search head, or on every search head cluster member: `| rest /services/kvstore/status` reports *ready*.
- The four TrackMe indexes exist on the indexers: `trackme_summary`, `trackme_metrics` (metric index), `trackme_audit`, `trackme_notable` (see *Installation of TrackMe*).
- Free disk on the search head for the application (2.4.x packages bundle Python libraries and are larger than 2.1.x) and for the KVstore backups.
- The target package downloaded from https://downloads.trackme-solutions.com/ and checksummed; the same file is used for every deployment of a programme.
- The licence key at hand for step 2.4, if the deployment is not registered yet.

Hint

Search head clusters

In a search head cluster the application is upgraded through the deployer (step 2.1). The members' own `local/` configuration is preserved by the bundle push; do not remove the application from the deployer's `shcluster/`

`apps` and re-add it, that would reset it. Migrations run once per tenant, on whichever member runs the tenant's health tracker; the KVstore is replicated, so every member sees the result.

Step 1 — Before the upgrade

Per deployment. Prerequisites, backups, baseline. About 30 minutes.

1.1 Confirm TrackMe is operational and no tenant is degraded

Open **Virtual Tenants**. Every tenant card must be free of the **degraded** badge and the tenant operations status must be fully OPERATIONAL. A degraded tenant means at least one of its scheduled reports failed on its last run.



The same check from a search, for the record:

```
| savedsearch trackme_ops_status
```

Warning

If a tenant is degraded

Identify the failing report (header → **Ops Status**, filter on failure) and address the root cause before the upgrade. Migrating a tenant whose trackers already fail makes the post-upgrade picture impossible to read: you will not know whether a failure is old or new.

1.2 Take the backups

TrackMe backup. Open **API & tooling → Backup & Restore** and click **Create backup**. The archive (knowledge objects and KVstore collections) is listed on the same page and can be restored from there. See *Backup & restore*.



File-system and KVstore backup (Splunk Enterprise). On-premises, also take a copy of the application directory and a KVstore backup; together they are the fastest restore path if anything goes wrong:

```
tar -czf /tmp/trackme-app-$(date +%Y%m%d).tgz -C $SPLUNK_HOME/etc/apps trackme
$SPLUNK_HOME/bin/splunk backup kvstore -archiveName trackme-pre-upgrade-$(date␣
↪+%Y%m%d)
```

Run both on the search head (on every search head cluster member for the application directory; the KVstore backup once, on the captain). The KVstore archive lands in `$SPLUNK_HOME/var/lib/splunk/kvstorebackup/`. Keep both copies outside the search head.

Note

On Splunk Cloud the file system is not accessible; the TrackMe backup is the restore path, and Splunk Cloud's own platform backups are a last resort to be requested from Splunk.

1.3 ML Outliers baseline — only where ML Outliers are in use

When upgrading from a release earlier than 2.3.15, every Outliers model refers to the Splunk AI Toolkit algorithm `DensityFunction`. The upgrade migrates the models to TrackMe's native engine (`TrackMeNativeDensityFunction`) and they are retrained automatically afterwards (step 3.3). Record the pre-migration state per tenant and component:

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| where match(entities_outliers, "\"algorithm\": \"DensityFunction\"")
| stats count
```

Then pick one well-behaved entity per component as the witness (trained model, normal confidence, no current outlier, green state), open it in **Tenant Home** and screenshot its **Outliers anomaly detection** tab:

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id> where␣
↪confidence="normal"
| lookup trackme_<component>_outliers_entity_data_tenant_<tenant_id> object OUTPUT␣
↪models_summary, isOutlier
| where isOutlier=0
| lookup trackme_<component>_tenant_<tenant_id> object OUTPUT object_state
| where object_state="green"
```



1.4 Record the running version and schema state

On **Virtual Tenants**, note the version in the header card, then click the version number. The **Tenants Update statuses** screen must show every tenant with *Schema version* equal to *Schema version required*, status **updated**. Screenshot it; you will compare it with the same screen in step 2.


Virtual Tenants
Configuration
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
TrackMe - Virtual Tenants
Monitoring the great value in your Big Data
Edition
Enterprise edition
NUMBER OF ACTIVE TENANTS
2
HIGH PRIORITY ENTITIES IN ALERT:
0
SCHEDULER COMPLETED PERCENTAGE
TENANT OPS STATUS
COMPLETED
OPERATIONAL
Search tenants by id, alias, or description...
demo-outliers
ID: demo-outliers
15 enabled entities
0 high priority
Double click for details
secops
ID: secops
4901 enabled entities
0 high priority
Double click for details


TrackMe License Informations & Tenants Update statuses
After an upgrade of TrackMe, the schema upgrade is triggered by the Health Tracker of the Virtual Tenant. Once terminated, trackers are allowed to be executed and the schema version is upgraded to reflect the readiness of the tenant.
License info: TrackMe is currently registered and operating in Enterprise edition.
Tenant Update Status Overview
updated
Status: updated
Status: pending
Status: all
Search
20 / page
Actions
Status
Tenant ID
Schema version
Schema version required
Tenant updated status
Schema version mtime
demo-outliers
2304
2304
updated
N/A
secops
2304
2304
updated
N/A
Showing 1-2 of 2
First
Prev
1 / 1
Next
Last
Open schema update logs in search
Close

Step 1 checkpoint — go / no-go

- All tenants OPERATIONAL, none degraded (1.1)
- TrackMe backup taken minutes before the change window, status success (1.2)
- On Splunk Enterprise: application directory and KVstore backups taken and stored off the search head (1.2)
- Where Outliers are used: model count exported and witness entity screenshot captured (1.3)
- Update-status screen captured: every tenant updated (1.4)
- Target release confirmed (never 2.3.22 to 2.4.1; one release for the whole estate)

Step 2 — Upgrade day

Install in place, follow the migration, register the licence, restart.

2.1 Install the package in place

The upgrade is a standard Splunk application upgrade: the new package replaces `default/`, `bin/` and `lib/`, and leaves `local/`, the KVstore and the indexes untouched. **Never remove the application directory first.**

Topology	Procedure
Standalone or distributed search head	Splunk Web → **Apps → Manage Apps → Install app from file**, choose the package, tick **Upgrade app**, install. Or from the CLI: `splunk install app trackme_<version>_<build>.tgz -update 1`. Restart splunkd when prompted (new custom commands and Python libraries require it).
Search head cluster	On the deployer, replace the application under `$SPLUNK_HOME/etc/shcluster/apps/trackme` with the extracted package (overwrite in place; keep any deployer-side `local/`), then `splunk apply shcluster-bundle -target https://<captain>:8089`. Members restart in a rolling fashion. Do not install the package directly on members.
Splunk Cloud	When the release has been vetted, it appears as upgradable in the application management interface; follow the self-service process. There is usually a delay between the request and the installation, so by the time you connect to TrackMe the upgrade has often already happened.
Indexers	Nothing to do beyond the index definitions already in place (step 0).

2.2 Read the new version and open the update-status screen

Once splunkd is back, reload **Virtual Tenants**. The header shows the new version. Click it.



The screen explains what happens next: *the schema upgrade is triggered by the Health Tracker of the Virtual Tenant; once terminated, trackers are allowed to run and the schema version is updated to reflect the readiness of the tenant.* Immediately after the installation, every tenant is **pending** with the new required schema.



2.3 Follow the migration

Each tenant's health tracker runs the migration chain on its next cycle (every 5 minutes): every migration from the tenant's stored schema up to the required one, in order, after an automatic safety backup. Refresh the screen with the icon at the top right; enabled tenants move from *pending* to *updated* one by one. A large version jump means a long chain; allow two to three cycles per tenant, and more on large tenants, since some migrations touch every entity record.

Note

Disabled Virtual Tenants stay pending. A disabled tenant's health tracker does not run, so the tenant is not migrated and keeps showing *pending* until it is re-enabled (its migration then runs within the next 5 minutes). Either re-enable such tenants during the change window so that they migrate with the others, or record them explicitly as excluded from the completion criteria; the checkpoint below counts enabled tenants only.



The migration log is one click away, **Open schema update logs in search**, or per tenant through the magnifier in the Actions column. Manually:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪task=schema_upgrade
```

Each migration logs its start and its outcome; the tenant's chain ends with **final schema version updated to <required>**. See *Schema migrations* for the meaning of every line.



Then filter on errors:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪task=schema_upgrade log_level=error
```

Migrations are fail-soft: a failing item is logged and skipped, the chain continues, and the schema version still advances. TrackMe also runs reconciliation logic on regular cycles (the longest every 24 hours) that repairs the vast majority of such conditions on its own. An error here is therefore rarely a concern for the upgrade itself. Do share the errors with TrackMe Support (see *Support*) so that each one can be confirmed. Do not judge anything before the update-status donut is fully *updated*: that screen is the authoritative progress view.

2.4 Register the licence key

If the deployment is not registered yet (for example a former Community Edition install), open **License, Help & support → License**, register the licence key and confirm the page shows the edition registered with its expiry date. See *License registration*. Licensed deployments keep their licence through the upgrade; verify the page anyway and record the edition and expiry in the deployment record.

2.5 Restart Splunk once every tenant is migrated

On a **standalone or distributed search head**, for a large version jump we recommend one more restart of splunkd **after** the update-status screen shows every tenant *updated* and the licence is registered. The migration chain refreshes many per-tenant knowledge objects (transforms, lookup definitions, saved searches) in `local/`; a restart guarantees that every search process and scheduled report starts from the final, fully reloaded configuration. Run `splunk restart` in the change window.

This step is **exempt** on the other topologies:

- Search head cluster: no restart after the migration. The members already restarted during the bundle push (2.1), and the knowledge objects the migration refreshes are written through the REST API, replicated and reloaded on every member without a restart.
- Splunk Cloud: not applicable.

Where the restart applies, wait until the health trackers have run once more (about 5 minutes) before starting step 3.

Step 2 checkpoint

- Header shows the target version (2.2)
- Update-status screen: every enabled tenant at the required schema, status *updated* (2.3)
- Migration log shows *final schema version updated* for every tenant; errors captured and sent to Support (2.3)
- Licence page shows the edition registered, with its expiry date (2.4)
- Standalone or distributed search head only: splunkd restarted after the migration; health trackers have run once since (2.5). Search head cluster and Splunk Cloud: recorded as exempt

Step 3 — After the upgrade

Platform, alerting chain, ML Outliers, hand-over.

3.1 Every tenant and every component loads without error

Wait about 15 minutes after the post-migration restart (or, where step 2.5 is exempt, after the last tenant reached *updated*): the health tracker also reschedules trackers and reconciles feature gates during the first cycles. Then open **Virtual Tenants**: the page loads, no error toast, every tenant OPERATIONAL. Open each tenant (**Tenant Home**) and click through its enabled component tabs: entity tables load, counters are populated, no error toast. Finally open **Configuration → Configuration Guardian**: it lists misconfigurations with remediation steps; anything raised must have a documented cause (see *Configuration Guardian*).



3.2 Validate the full alerting chain with a manual score influence

An entity's colour is driven by an impact score (0 = green, below 100 = orange, 100 and above = red; see *Impact scoring & thresholds*). The score can be influenced manually, which gives a clean, reversible way to force a healthy entity into alert and exercise everything downstream: notable creation, stateful alerts and emails, and your own tooling (ticketing, paging).

a. Pick a healthy, high-priority entity. In Tenant Home, search for a critical or high-priority entity that is green with a score of 0.



b. Select it and open Bulk edit. Tick the entity's checkbox; the **Bulk edit (1)** button appears above the table.



c. Add 100 to the score. Category **Impact Score**, action **Manually influence the score**, operation **Add to score (increase)**, value **100** (red starts from 100). Enter a comment such as "post-upgrade alerting chain test" and apply. The same action exists on a single entity from its **Actions** menu.



d. Watch the entity turn red. On the next evaluation the entity shows a score of 100 and a red state, with the anomaly reason `score_breached`. This can take a few minutes.



e. Confirm the incident. Alerts run on their own schedule, so the incident typically appears 5 to 10 minutes later. Open the entity's **Incidents** tab: the `trackme:notable` event is listed with the entity's fields. This is the payload your downstream tooling consumes; verify that the ticket, page or email arrived.



f. Clear the test. From the entity's **Actions** menu choose **Set as false positive** and enter a comment. This records a negative score that cancels the manual influence; the entity returns to green on the next evaluation and the stateful alert, if any, closes on its own.



Hint

What this proves

Score evaluation, state flip, notable emission and your downstream chain all work on the new release, with one reversible action and no synthetic data. Repeat it once per class of tenants if they feed different downstream pipelines.

3.3 ML Outliers — only where they are in use

When the upgrade crosses the native ML engine migration (release 2.3.15), every model definition is rewritten to `TrackMeNativeDensityFunction` and reset for retraining (`last_exec=0`). The migration does **not** train the new models: until a model is trained its Outliers chart is empty and it produces no detection. Training runs automatically through the per-tenant, per-component report `trackme_<component>_outliers_mltrain_tracker_tenant_<tenant_id>`, which processes the backlog in batches over its scheduled runs. Three checks:

a. Definitions migrated. The step 1.3 count search now returns 0; the positive form shows the migrated records: `algorithm` is `TrackMeNativeDensityFunction`, the stored searches read `pending` and the record's `last_exec` is 0. That reset is the instruction to the training job.

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| where match(entities_outliers, "\"algorithm\": \"TrackMeNativeDensityFunction\"")
```



Until retraining, the witness entity's Outliers tab shows the model in the selector, the counters at zero and an empty chart. This is expected at this point, not a fault.



b. Backlog draining. The following search drains to 0 rows as training progresses:

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| where is_disabled=0 AND last_exec=0
```

To accelerate, run the mltrain report from Splunk's **Searches, Reports, and Alerts** page (application TrackMe, filter `mltrain`) with the **Run** action, once per tenant and component, repeated until the backlog is empty; or select entities in Tenant Home and use **Bulk edit → Outliers Detection Actions → Run ML Train**. Opening an entity whose native model is missing also triggers an automatic first training (expect a slow first render rather than an error).





c. Witness entity renders again. Reopen the step 1.3 witness: once trained, its Outliers tab shows the chart again on the native engine. The band may differ slightly from the step 1.3 screenshot, the model having been refitted on the current rolling window. See *Outlier detection* for the model tuning reference.



3.4 Close the deployment record and hand over

Complete the validation record for the deployment (template in *Validating after an upgrade*), archive the step 1 screenshots and exports next to it, and inform the operators that the upgrade is complete and that the alerting test has been cleared. Brief them on the behaviour changes of the new release (see *Validating after an upgrade*).

Step 3 checkpoint

- Virtual Tenants, every tenant / component and the Configuration Guardian page load with no unexplained item (3.1)

- Forced alert turned the test entity red; notable created; downstream incident confirmed; test cleared as false positive (3.2)
- Where Outliers are used: no legacy-engine model left; backlog drained; witness entity renders again (3.3)
- Validation record completed and archived; operators informed (3.4)

Step 4 — Only if a rollback is decided

Danger

A rollback is a restore, not an un-install.

Schema migrations are forward-only: reinstalling the previous package reverts the code and none of the migrated data, and the update-status screen would even report *updated*. The reliable rollback is to restore the step 1.2 backups on the reinstalled package. When the driver is alerting behaviour (impact-score weights, outlier score, policy cadence, threshold locks), tune first: those are configurable on the current release. Changes that are not configurable (the native ML engine, the schema shape, the rebuilt interface) are the only ones a rollback reverts; if the decision stands, follow *Rollback*.

The runbook in four lines

- **Upgrade path:** in place, one release for the whole estate.
- **Progress view:** click the version number.
- **Proof of health:** force one alert and clear it.
- **Rollback model:** restore the backups.

7.2.2 Schema migrations

Once TrackMe has been upgraded through the standard Splunk process (Splunk Enterprise or Splunk Cloud, search head cluster or standalone), TrackMe automatically verifies and performs the additional upgrade procedures each Virtual Tenant needs. This mechanism is the **schema version**: an automated, per-tenant, forward-only migration chain that creates, updates or deletes TrackMe knowledge objects and KVstore collections depending on the origin version and the state of the tenant. This page describes how it works, how to follow it, and what runs between releases.

The schema marker

The schema version is versioning information stored in the KVstore record of each Virtual Tenant. It corresponds to the TrackMe version without the dots, with a two-digit patch number: version 2.1.0 is schema `2100`, version 2.3.4 is `2304`, version 2.4.14 is `2414`.

```
| inputlookup trackme_virtual_tenants | eval keyid=_key
| fields keyid, tenant_id, schema_version, schema_version_mtime
```





The required schema version derives from the running application version. The `trackmetenantstatus` command returns both, together with the derived *updated* / *pending* status, for every tenant the user can access:

```
| trackmetenantstatus output="tenants"
| table tenant_id, tenant_status, schema_version, schema_version_required,␣
↪tenant_updated_status, schema_version_mtime
```

The Tenants Update statuses screen

On the **Virtual Tenants** page, the version number in the header is a link. It opens the **Tenants Update statuses** screen, the authoritative progress view of a migration:

- a donut of tenants *updated* vs *pending*;
- one row per tenant with **Schema version**, **Schema version required**, **Tenant updated status** and **Schema version mtime**;

- a per-row action and a global **Open schema update logs in search** button, both landing on the migration log of the tenant.



Who runs the migration

The schema verification is handled by the **Health tracker** of the Virtual Tenant. When a tenant is created, TrackMe creates a dedicated health tracker for it, responsible for administration tasks such as detecting issues in the tenant's trackers, and for running the schema verification:

- `trackme_health_tracker_tenant_<tenant_id>`

The health tracker runs every 5 minutes. When it detects that the stored schema version is behind the required one, it runs every migration between the two, in order, and advances the marker after each one. Migrations therefore run on the health-tracker cycle, **not at package installation time**: allow one to two cycles per tenant before judging anything, and more on large tenants, since some migrations touch every entity record.

Disabled Virtual Tenants

- Disabled Virtual Tenants are not migrated as long as they are disabled (their health tracker does not run).
- If a tenant is re-enabled, its migration is processed automatically within the next 5 minutes.

Search head clusters

The migration of a tenant runs on whichever member runs the tenant's health tracker. The KVstore is replicated, so the result is visible on every member.

The automatic safety backup

Before the first migration of a chain runs, TrackMe executes a backup job automatically. The run appears in **API & tooling → Backup & Restore** with the comment *Backup initiated for schema migration from version X to Y* (on releases with the multi-archive backup format, one archive per enabled tenant plus one global archive). The backup is deduplicated per required version and day, so several tenants migrating in the same window share one safety backup. See *Backup & restore*.

```
| inputlookup trackme_backup_archives_info
| table backup_run_id, archive_scope, tenant_id, status, mtime, comment
| sort - mtime
```

Reading the migration log

The logs of the health tracker and of every migration are available through the navigation bar shortcut **Audit & troubleshoot → Logs - TrackMe custom commands** and match:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth task=
↪"schema_upgrade"
```

Every task carries a `task_instance_id` so that one run can be followed end to end. The run time of every health-tracker task can be tracked with:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪instance_id=* task_instance_id=* task=* run_time=* tenant_id=*
| table _time tenant_id instance_id task task_instance_id run_time _raw
| sort 0 - _time
```

Before an upgrade, the health tracker only logs routine verifications:



When the tracker runs, it verifies the schema version:

```
INFO trackmetrackerhealth.py generate tenant_id="feeds-tracking",␣
↪instance_id=879a5021-..., task="schema_upgrade", task_instance_id=80a9366d-...,␣
↪starting task.
INFO trackme_libs_schema.py trackme_schema_get_version task="schema_upgrade",␣
↪task_instance_id=80a9366d-..., tenant_id="feeds-tracking",␣
↪trackme_schema_get_version, current schema_version="2097"
```

If an upgrade is required, the following message is logged:

```
INFO trackmetrackerhealth.py generate tenant_id="feeds-tracking", ..., task=
↪"schema_upgrade", ..., detected migration required for schema version 2098,␣
↪schema_version="2097", schema_version_required="2100", processing now.
```

The safety backup is executed:

```
INFO trackmetrackerhealth.py generate tenant_id="feeds-tracking", ..., task=
↪"schema_upgrade", ..., backup post call executed successfully
```

Each migration then runs in order and logs its actions:

```
INFO trackme_libs_schema.py trackme_schema_upgrade_2098 task="schema_upgrade", ...,
↪ Starting function trackme_schema_upgrade_2098, tenant_id="feeds-tracking"
INFO trackme_libs_schema.py trackme_schema_upgrade_2098 task="schema_upgrade", ...,
↪ schema migration 2098, tenant_id="feeds-tracking", successfully deleted␣
↪transform definition, transform="trackme_dsm_tenant_feeds-tracking"
```

Once every migration has completed, the marker is updated and the task terminates:

```
INFO trackme_libs_schema.py trackme_schema_update_version task="schema_upgrade", ..
↪., tenant_id="feeds-tracking", schema version upgraded successfully, new␣
↪schema_version="2100"
INFO trackmetrackerhealth.py generate tenant_id="feeds-tracking", ..., task=
↪"schema_upgrade", ..., final schema version updated to 2100.
INFO trackmetrackerhealth.py generate tenant_id="feeds-tracking", ..., task=
↪"schema_upgrade", ..., run_time="26.2", task has terminated.
```



Errors and the fail-soft rule

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪task=schema_upgrade log_level=error
```

Migrations are written to be **fail-soft**: per-item work (a record backfill, a transform refresh, a saved search update) is caught, logged and skipped, the chain continues, and the schema version still advances. A single bad record or knowledge object cannot wedge the chain. The bootstrap of a migration (the connection to splunkd and the initial KVstore read) is deliberately *not* guarded, so that a transient outage propagates and the tracker simply retries on its next cycle without advancing the marker.

Important

updated means the chain ran, not that every item succeeded.

Because of the fail-soft rule, a tenant can report *updated* while one record or knowledge object was skipped with an error. The update-status screen is therefore the progress view, and the error search above is part of the completion criteria: review every error logged during the migration window before signing off the upgrade checkpoint.

TrackMe also runs reconciliation logic on regular cycles (the longest every 24 hours) that repairs the vast majority of skipped items on its own, and the Configuration Guardian surfaces persistent inconsistencies with remediation steps (see *Configuration Guardian*). Handle each logged error as follows:

- **Self-healed**: the same item is processed successfully by a later cycle, or the Guardian raises no alert for it after 24 hours. Record the error and its resolution; nothing else to do.
- **Persistent**: the error repeats, a Guardian alert names the item, or a tenant stays DEGRADED because of it. Apply the Guardian's remediation steps or, for a record that cannot be reconciled, re-run the migration's effect through the corresponding management action (for example a model retrain, a policy re-apply, a tracker re-creation from the tenant's management screen).
- **Unclear**: share the error lines with TrackMe Support (see *Support*) and keep the checkpoint open until they are confirmed as benign or resolved.

What runs between releases

Every release may ship at most one migration, numbered with its schema version. A migration is never modified after it ships; robustness fixes and new fields go into a later one. A chain from an old release therefore runs every migration between the two versions, in order. The condensed catalogue below covers the 2.1 line onward; the full detail of each release is in the *release notes*.

Schema	Effect (condensed)
2121 to 2132	2.1 line: tenant configuration reconciliation (new tenant-level preferences with defaults), remote-account preferences, transform refreshes; the FQM component initialised disabled on existing tenants; FLX drilldown-search collections; WLK workload trackers moved to the new index. Mechanical, no behaviour change.
2300 to 2314	2.3 line: main-collection transform refreshes; stateful alerts rewritten to use the `trackmestateful` command (2304); lookup refreshes; variable delay collections (2312); per-component lagging classes (2313); policy transforms (2314).
2315	**Native ML engine migration.** Creates the per-tenant native-models KVstore collection and transform; rewrites every Outliers model's `algorithm` from `DensityFunction` to `TrackMeNativeDensityFunction`; sets `model_storage=kvstore`; resets the stored model searches to `pending` and `last_exec` to 0 so that the next training cycles regenerate them. Models retrain automatically on the following mltrain runs (step 3.3 of the *Upgrade procedure*). This is the one migration a rollback must compensate for (see *Rollback*).
2316 to 2322	Shadow read cache and score cache collections; health-tracker state; labels; policy tracker cadence reduced from every 15 minutes to twice daily; outlier result transforms; tenant-level ML settings backfill.
2400 to 2406	Tenant schema reconciliation; AI advisor saved searches created (seeded unscheduled, inert unless AI features are enabled); Smart Status decommission; threshold intent-lock collections; labels backfill; WLK execution-errors collection.
2408 to 2414	DSM `min_records` transform refresh; native-models transform refresh (incremental-training diagnostics); SLA manual-override contract repair (clears the flag that SLA *policies* had stamped on entities, so those entities return to policy evaluation; genuinely manual overrides are preserved).

Two timing consequences for validation

- Migrations run on the health-tracker cycle, not at installation time. The update-status donut moves from *pending* to *updated* as tenants complete; do not judge anything before it is fully updated.
- Migration 2315 resets model searches to `pending` and `last_exec` to 0: that is the migration working, not a failure. Models refit on the following mltrain cycles, and a missing native model is also trained automatically the first time the entity is opened.

What the marker does on a downgrade



Migrations are forward-only. If an older package is reinstalled over migrated data, no migration runs, nothing is reverted, and the health tracker silently **realigns the marker down** to the running version. The Tenants Update statuses screen then shows every tenant as *updated* while the data keeps its migrated shape. This is why a rollback is a restore of the pre-upgrade backups and never a re-install of the older package, and why post-downgrade validation must inspect the data itself: see *Rollback*. On a later re-upgrade the migrations run again; they are idempotent, so that is safe.

7.2.3 Validating after an upgrade

The *Upgrade procedure* covers what matters on the day. This page is the reference behind it: the built-in screens that answer most checks, the behaviour changes to brief operators on, the exhaustive checklist for formal validation records, the evidence searches, and a record template.

The built-in validation cockpit

Most of the validation is executed from TrackMe's own screens. Use them first; capture screenshots as evidence; fall back to the searches at the end of this page when you need an export or a before/after diff.



Surface	What it tells you	Use it for
Tenants Update statuses (Virtual Tenants → click the version number)	Donut of tenants *updated* vs *pending*; one row per tenant with schema version, required version, status and mtime; per-row and global links to the migration log.	Steps 1.4, 2.2, 2.3
Ops Status (Virtual Tenants → header)	Every tenant × report (tracker) with status, last result, last execution and duration; success / failure filters. **Run: Clear Tenant Status** resets the execution summary so a failure recorded *before* the upgrade does not keep a tenant DEGRADED afterwards.	Steps 1.1, 3.1
Tenant cards (Virtual Tenants grid)	OPERATIONAL / DEGRADED per tenant; Configuration Guardian alerts raised as toasts on page load.	Steps 1.1, 3.1
Configuration Guardian (Configuration menu)	Persistent misconfiguration alerts with remediation steps; per-check enable / snooze; on-demand run. Upgrade-relevant checks include `health_tracker_not_executing`, `tenant_operations_degraded`, `backup_run_incomplete` and `insufficient_tenant_owner_capabilities`. See *Configuration Guardian*.	Step 3.1
Backup & Restore (API & tooling menu)	Run and verify backups; the automatic pre-migration safety backup appears with a *schema migration* comment; restore with a mandatory dry-run preview. See *Backup & restore*.	Step 1.2, rollback
License (License, Help & support menu)	Edition in force, registration, expiry.	Step 2.4
Entity → Outliers tab (Tenant Home)	Per model: rendered bounds over the KPI, corrected-bounds counters, outliers counters, impact score; false-positive and management actions.	Steps 1.3, 3.3
Bulk edit (Tenant Home entity table)	Category **Impact Score → Manually influence the score**; category **Outliers Detection Actions → Run ML Train**, on any selection of entities.	Steps 3.2, 3.3
Audit & troubleshoot (navigation menu)	**Audit - Operational Statuses**, **Audit - Trackers Performance DeepDive**, **Audit - KVstore Collections**, **Audit - Logs Inspector interface** (ERROR / WARN across TrackMe components), **Support - generate diags** (see *Support diagnostics*).	Whenever a check fails

Behaviour changes to brief operators on

None of the changes below modifies stored entities or models destructively, but several change **what alerts fire and when**, which is exactly what operators should be told before an upgrade rather than discover after it. The 2.1 and 2.2 lines, and the 2.3 line up to 2.3.4, bring mechanical changes and user-interface modernisation; the alerting behaviour changes cluster from 2.3.5 onward. The *release notes* carry the full detail per release.

Since	Change	What operators observe
2.2.0	Splunk 9.1.1 or later required	No compatibility with Splunk 9.0.x and earlier from the 2.1 line onward; see *Compatibility*.
2.3.0	User interfaces rebuilt on Splunk UI native React	Every main interface changes layout; the information is the same, its place on screen is not. Brief users before the upgrade.
2.3.5	**Impact scoring introduced**	The single biggest change. Entity colour is no longer a direct rule outcome but a weighted score: 0 = green, below 100 = orange, 100 and above = red. Alert volumes and severity distribution shift; tuning weights replaces tuning booleans. See *Impact scoring & thresholds*.
2.3.5	Score-based outliers	An outlier no longer turns an entity red on its own; it contributes its model's score (seeded at 36 by default). Isolated blips stop alerting; sustained anomalies still escalate. Low-confidence models emit no score at all.
2.3.15	Native ML engine (migration 2315)	Models move from MLTK `.mlmodel` files to the KVstore; the MLTK dependency is no longer needed for outliers; models retrain once after the migration. Minimum 10 samples per seasonality slot enforced.
2.3.16	Shadow cache and score cache	Faster UI at scale; false-positive and manual score edits take effect on the next evaluation.
2.3.19	Policy tracker cadence	SLA, tags and priority policy re-evaluation drops from every 15 minutes to twice daily; policy changes apply with hours, not minutes, of latency.
2.3.22 to 2.4.1	**ML training regression (fixed 2.4.2)**	In these builds model training silently stops for every component. Never land on them.
2.4.0	Per-entity maintenance mode	Entities can be forced BLUE for a window, taking precedence over acknowledgements and other protections; alerting suppressed until expiry.
2.4.0	AI advisor scheduled searches	New per-tenant saved searches appear, **created unscheduled**; they only activate if AI features are enabled. Their presence is expected, not a fault.
2.4.1	Threshold intent lock	The per-entity adaptive-delay and lagging-class override controls are replaced by a single **Threshold lock**; locked entities show a lock marker and adaptive mechanisms respect it.
2.4.1	Adaptive delay on the native engine	No MLTK dependency; a historic bug silently flooring bounds to zero is fixed, so adaptive thresholds may legitimately differ from earlier values.
2.4.7	Configuration Guardian management UI	Guardian alerts gain a dedicated page, snooze and disable controls and email notifications; the DEGRADED signal has a persistent, tunable surface.
2.4.9	Search-driven diagnostics and backups	Long-running diagnostics and backups run in search processes; backup archives can be downloaded and imported from the UI.
2.4.10	Alert enrichment	Optional per-tenant SPL enrichment merges custom fields into stateful alerts and notables, a supported carrier for downstream ticket references.
2.4.14	Policy apply at scale; SLA contract repair	Priority, tags and SLA policies are applied inside the tracker process (no cut-off on very large tenants). Migration 2414 returns entities that SLA *policies* had frozen to policy evaluation; their SLA class may legitimately change on the next policy cycle.

Warning

Expect the alert-volume profile to change, by design.

Comparing raw incident counts before and after an upgrade that crosses 2.3.5 compares two different alerting models (boolean rules against impact scoring). Use the baselines below to *understand* the delta, not to demand

that it be zero: fewer one-off blips and unchanged-or-better detection of sustained anomalies is the expected signature.

Extended validation checklist

Organised in phases for formal validation records. Each table gives the built-in screen first and the evidence search (E-numbered, at the end of this page) second.

P0 — Platform

Check	Where to look / method	Expected
Running version is the target	Virtual Tenants header; E1.1	The target release
Schema migration completed	Tenants Update statuses; E1.2	Donut 100% *updated*; every enabled tenant at schema version = required version
Migration log clean	*Open schema update logs in search*; E1.3	*final schema version updated* per tenant; errors reviewed and sent to Support
Safety backup was taken	Backup & Restore archives list; E1.4	A backup run with a *schema migration* comment, status success
Licence in force	License page	Edition registered, expiry recorded

P1 — Tenant operations

Check	Where to look / method	Expected
Every tenant OPERATIONAL	Tenant cards; Ops Status with the failure filter; E4.1	All reports `success`. A failure dated *before* the upgrade window: **Run: Clear Tenant Status** and recheck on the next cycle. A failure dated after: open the report's last result and the Logs Inspector
Health tracker executing per tenant	E4.2	Successful cycle within the last 30 minutes for every enabled tenant
Guardian is clean	Configuration Guardian page (run checks on demand)	No `health_tracker_not_executing`, no `tenant_operations_degraded`; anything raised has a documented cause
No new errors in TrackMe components	Audit - Logs Inspector interface, level ERROR, last 4 hours	No recurring error tied to a tracker or REST handler; one-off warnings during the migration window are acceptable

P2 — Scheduled searches and data

Check	Where to look / method	Expected
Tracker inventory matches the pre-change export	E3.1 re-run, diff	Same set per tenant, plus the expected additions of the new release (for example AI advisor searches, unscheduled); no tracker lost its schedule
Trackers actually firing	Audit - Trackers Performance DeepDive; E3.2 over the first 24 hours	Each enabled tracker shows recent scheduled runs; no persistent skips attributable to TrackMe
KVstore collections intact	Audit - KVstore Collections; E6	Record counts per main entity collection unchanged (plus or minus normal discovery churn); new collections present and small
Custom knowledge objects still work	Open your own TrackMe-based dashboards and reports	They render; a field renamed by a migration is the usual cause of a broken panel

P3 — ML Outliers (where in use)



Check	Where to look / method	Expected
Definitions migrated	E2.1 (algorithm column)	Every model `TrackMeNative-DensityFunction`; no `Density-Function` left
Retraining backlog drained	E2.3	No enabled rules record with `last_exec=0` after the mltrain report has run enough batches
Fitted models present	E2.2	One record per model with a recent `fitted_at`; `confidence=normal` in E2.1 for established entities
Charts render	Outliers tab on the witness and on priority entities	Bounds hug the signal; the *Corrected* counters are zero or rare (a cluster of corrected bounds means models starved of history: lengthen the calculation period and retrain; see *Outlier detection*)
Training history preserved	E2.4	Per-entity training events before and after the window; `fitted_at` *advancing* after the upgrade is correct behaviour (models retrain on a rolling window), not history loss

P4 — Alerting pipeline and the downstream hand-off



Check	Where to look / method	Expected
Notables emitted	Entity **Incidents** tab (step 3.2); E5.2 over the test window	Notable records carry tenant, object, state, score, anomaly reasons, priority and drilldown fields. See *Notable events*
Stateful alert lifecycle	Step 3.2 open → close on the test entity; E5.3	Lifecycle events indexed (`trackme:stateful_alerts`); emails and commands fire per the alert's delivery configuration; close follows the false-positive clean-up. See *Alerting*
Alert actions ran without error	Navigation → *Logs - TrackMe alert actions*	No ERROR from `trackme_stateful_alert`, `trackme_notable`, `trackme_free_style_rest_call`
Email delivery accounts still valid	Configuration → email accounts → test; credentials in `local/` are preserved by the upgrade	Test email received
Downstream hand-off	The step 3.2 forced alert	Your ticket, page or email is produced from the notable or stateful alert, confirmed by its owner

TrackMe's responsibility ends at its emission surfaces: the notable index, the stateful-alert events, email delivery, the free-style REST action and the SOAR integration. Downstream ticketing and paging consume those surfaces; the validation contract at the boundary is the emitted payload, which the forced-alert test of step 3.2 produces on demand. From 2.4.10, alert enrichment can merge custom fields into notables and stateful alerts; a stable correlation key (tenant, entity, alert identifier) as an enriched field gives the downstream pipeline a deterministic join key.

Evidence searches

Each block is one standalone search: run it on its own and export the result to CSV. Running the same searches before and after the change produces the comparison. Index names are the defaults; replace them if the tenant uses custom indexes.

E1 — version, schema, migration log, safety backup

E1.1 Running application version

```
| rest /services/apps/local/trackme | table label version
```

E1.2 Schema state, all tenants in one search (the data behind the Tenants Update statuses screen)

```
| trackmetenantstatus output="tenants"
| table tenant_id, tenant_status, schema_version, schema_version_required,␣
↪tenant_updated_status, schema_version_mtime
```

E1.3 Schema migration log

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪task=schema_upgrade tenant_id="<tenant_id>"
| table _time, log_level, _raw
```

E1.4 Backup archives, including the automatic pre-migration safety backup

```
| inputlookup trackme_backup_archives_info
| table backup_run_id, archive_scope, tenant_id, status, mtime, comment
| sort - mtime
```

E2 — ML Outliers inventory (where in use)

E2.1 Model definitions per entity (per component)

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| rex max_match=0 field=entities_outliers "\"(?<model_id>model_\d+)\":"
| rex max_match=0 field=entities_outliers "\"algorithm\": \"(?<algorithm>[^\"]+)\""
| table object, model_id, algorithm, confidence, confidence_reason, last_exec,␣
↪is_disabled
```

E2.2 Fitted native models (per tenant, 2.3.15 and later)

```
| inputlookup trackme_native_ml_models_tenant_<tenant_id>
| table _key, feature_name, fitted_at, group_count, training_mode, mtime
```

E2.3 Retraining backlog

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| where is_disabled=0 AND last_exec=0
```

E2.4 Training history, one event per entity per training run (audit index)

```
index=trackme_audit sourcetype=trackme:handler tenant_id="<tenant_id>"
    handler="trackme_<component>_outliers_mltrain_tracker_tenant_<tenant_id>"␣
↪earliest=-30d
| stats count as training_runs, min(_time) as first_trained, max(_time) as␣
↪last_trained by object
| convert ctime(first_trained), ctime(last_trained)
```

E3 — scheduled searches

E3.1 Tracker inventory

```
| rest /servicesNS/-/trackme/saved/searches search="trackme_*_tenant_<tenant_id>"
| table title, is_scheduled, disabled, cron_schedule, dispatch.earliest_time,␣
↪dispatch.latest_time
```

E3.2 Scheduler activity, last 24 hours

```
index=_internal sourcetype=scheduler app="trackme" savedsearch_name=
↪"trackme_*_tenant_<tenant_id>" earliest=-24h
| stats count by savedsearch_name, status
```

E4 — tenant operations

E4.1 Per-report execution status (the data behind Ops Status)

```
| trackmetenantstatus tenant_id="<tenant_id>"
```

E4.2 Health-tracker task state

```
| inputlookup trackme_health_tracker_state where tenant_id="<tenant_id>"
| table task_name, execution_count, last_execution_status, last_execution_time,␣
↪last_execution_duration
```

E5 — alert-flow baselines (7 days)

E5.1 State-change (flip) volume, summary index

```
index=trackme_summary sourcetype=trackme:flip tenant_id="<tenant_id>" earliest=-7d
| timechart span=1d count by object_category
```

E5.2 Notable volume

```
index=trackme_notable sourcetype=trackme:notable tenant_id="<tenant_id>" earliest=-
↪7d | timechart span=1d count
```

E5.3 Stateful alert lifecycle events

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="<tenant_id>"␣
↪earliest=-7d
| timechart span=1d count by object_state
```

E6 — KVstore record counts (repeat per collection of interest)

```
| inputlookup trackme_<component>_tenant_<tenant_id> | stats count as entities
```

Validation record template

One record per deployment: the header from the step 0 inventory, then one row per check with the evidence reference. This is the artefact to retain as validation evidence.

Step	Check	Result	Evidence ref	Notes
0	Platform readiness (Splunk version, KVstore, indexes, package, key)		checklist	
1.1	All tenants operational, none degraded		screenshot / trackme_ops_status	
1.2	TrackMe backup (and on Splunk Enterprise: application directory and KVstore backups) taken		archive names	
1.3	Outliers: model count exported; witness screenshot (if used)		CSV + screenshot	
1.4	Update statuses: all updated at the source schema		screenshot	
2.1	Package installed in place; splunkd restarted		package name	
2.2	Version = target		screenshot	
2.3	All tenants updated at the required schema; final log line present; errors sent to Support		screenshot / E1.3	
2.4	Licence registered or verified, expiry recorded		screenshot	
2.5	splunkd restarted after the migration completed		time of restart	
3.1	Every tenant / component loads; Guardian reviewed		screenshots	
3.2	Forced alert → notable → downstream incident → cleared		Incidents tab + ticket reference	
3.3	Outliers: no legacy model left; backlog drained; witness renders (if used)		E2.3 + screenshot	
3.4	Record archived; operators informed			
P1	Health tracker current; no new errors (Logs Inspector)		E4.2, screenshot	
P2	Tracker inventory matches; trackers firing; KVstore counts stable; custom knowledge objects work		E3.1 diff, E3.2, E6	
P4	Alert actions clean; email delivery test		logs, test email	

7.2.4 Rollback

Danger

A rollback is a restore, not an un-install.

Schema migrations are forward-only. Reinstalling an older package reverts the code and none of the migrated data, over collections and knowledge objects that the migrations already reshaped, and the schema marker is silently realigned so that the Tenants Update statuses screen reports *updated*. The reliable rollback is to restore the pre-upgrade backups on the reinstalled package.

Before deciding, separate the two kinds of change listed in the behaviour changes table of *Validating after an upgrade*. Alerting behaviour (impact-score weights, the score contributed by an outlier, policy cadence, threshold locks, per-entity maintenance) is configurable: if the driver is alert-volume surprises or a scoring or policy outcome, tune on the current release first, a rollback is not needed. Changes that are not configurable (the native ML engine, the schema shape, the rebuilt user interface) are the only ones a rollback actually reverts; weigh them against its cost. A rollback across the native ML engine migration (2.3.15) additionally requires a full retraining of every Outliers model, so the models are inactive until it completes.

Procedure

1. **Decide against fresh evidence.** Capture the post-upgrade evidence first (update-status screenshot, model inventory, Ops Status) so that the comparison stays possible, and confirm the driver is not addressable by configuration on the current release.
2. **Freeze changes and take a final backup.** A TrackMe backup at the current release, plus, on Splunk Enterprise, copies of the application directory and of the KVstore, before any package operation.
3. **Reinstall the previous release in place.** Same procedure as step 2.1 of the *Upgrade procedure* with the previous package. Never remove the application directory: `local/` holds every tenant, tracker and credential.
4. **Restore the pre-upgrade state.** Restore the pre-upgrade KVstore backup (`splunk restore kvstore`) and the pre-upgrade `local/` from the application-directory copy taken in step 1.2 of the procedure. This returns every collection, transform and tracker to their pre-upgrade shape. On Splunk Cloud, or when those copies are not available, restore the pre-upgrade TrackMe backup from **Backup & Restore** (see *Backup & restore*) as a **whole run**, so that every tenant archive and the global archive are restored together (restoring a single archive leaves tenant and global state inconsistent; releases before the multi-archive format produce one archive per run, which is restored as such). Run the restore **dry run** first and review its preview before the real restore. Then apply the manual compensation below.
5. **Fallback only: compensate the native ML engine migration by hand.** If the rollback crosses release 2.3.15 and the KVstore was not restored, rewrite every model's algorithm back to the MLTK engine (see below) and retrain all models. Until retraining completes, Outliers are inactive.
6. **Re-validate.** Steps 1.1, 3.1 and 3.2 of the procedure, with the old behaviour expected (boolean alerting and no score events when rolling back below 2.3.5). The update-status screen will read *updated*; it cannot see a downgrade. Validate the data.
7. **Report to TrackMe Support.** A rollback must be understood before the next attempt; send the validation record and the migration logs (see *Support*).

The native ML engine compensation (fallback only)

Migration 2315 rewrote every Outliers model to `algorithm=TrackMeNativeDensityFunction` and set `model_storage=kvstore`. Code earlier than 2.3.15 has no such engine and does not validate the value: the training search fails at search time and **models stay silently inactive until reverted**. Detect, per tenant and component:

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
| rex max_match=0 field=entities_outliers "\"algorithm\": \"(?<algorithm>[^\"]+)\""
| stats count by algorithm
```

Rewrite both fields the migration changed, `algorithm` back to `DensityFunction` and `model_storage` back to `file` (back up the collection first; the key-preserving update keeps the records in place):

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id> | eval␣
↪keyid=_key
| rex field=entities_outliers mode=sed "s/\"algorithm\": \
↪"TrackMeNativeDensityFunction\"/\"algorithm\": \"DensityFunction\"/g"
| rex field=entities_outliers mode=sed "s/\"model_storage\": \"kvstore\"/\
↪"model_storage\": \"file\"/g"
| outputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>␣
↪append=t key_field=keyid
```

The stored model searches that the migration reset to `pending` are regenerated by the training run that follows; the fitted native models in `kv_trackme_native_ml_models_tenant_<tenant_id>` are simply ignored by the older release.

Then run **ML Train** across the affected models (Tenant Home → Bulk edit → Outliers Detection Actions, or the mltrain reports) and confirm the Outliers charts render again.

What a rollback does not undo without the KVstore restore

Migrated state	Behaviour on the older release
New KVstore collections (native models, score cache, shadow cache, intent locks, labels, maintenance and others)	Ignored: the older code never reads them. Dormant, not harmful. They re-activate on a later re-upgrade.
New fields on existing records; refreshed transforms	Carried along untouched; the older code reads the fields it knows.
Rewritten TrackMe-managed alert SPL and tracker definitions touched by migrations	Remain in their migrated form in `local/`. This is the reason the pre-upgrade `local/` copy matters.
Model `algorithm` and `model_storage` values (migration 2315)	**Actively incompatible below 2.3.15: the one compensation that must be performed by hand if the KVstore is not restored.**
Schema marker	Silently realigned to the running version; the update-status screen reports *updated* (see *Schema migrations*).

Hint

Why retraining is part of every rollback across 2.3.15

The two releases fit models with different engines in different stores, so a retrain on the rolling window is unavoidable, and freshly retrained bounds legitimately differ because the window has moved. The pre-upgrade evidence (step 1.3 of the procedure) makes this explainable instead of surprising.

7.2.5 What a version change actually touches

Every question about *what survives an upgrade* reduces to which of four layers the state lives in.



- **Application code and defaults** (`etc/apps/trackme`: `default/`, `bin/`, `lib/`) are replaced by the package. This is the only layer an upgrade replaces.
- **Local configuration** (`local/`) holds every Virtual Tenant definition, every per-tenant tracker and saved search, every per-tenant KVstore transform, the settings overrides and the account credentials. The package replacement leaves it entirely untouched. The schema migration that follows may then refresh the TrackMe-managed objects it contains (per-tenant transforms, lookup definitions, saved searches) to the new release's shape; user-owned configuration (tenants, settings overrides, credentials, your own knowledge objects) is retained.
- **KVstore data** (`kv_trackme_*` collections: entities, rules, fitted ML models, policies, state) is untouched by the package. Only the forward-only schema migration reshapes it: it creates collections and fields, rewrites values (for example the Outliers model engine) and resets state. This is what the pre-upgrade backups exist to undo.
- **Indexes** (`trackme_summary`, `trackme_metrics`, `trackme_audit`, `trackme_notable`) are never touched.

Danger

Remove and reinstall is not an upgrade path.

Deleting the application directory destroys `local/`: every tenant, every tracker, every per-tenant KVstore transform and every credential. The KVstore *data* survives, but becomes unreachable from SPL until the transforms are rebuilt. Always upgrade **in place**. If a reinstall is ever unavoidable, restore a TrackMe backup immediately afterwards (see *Backup & restore*).

7.2.6 Principles

Upgrade in place, from a backup. Take a TrackMe backup immediately before the change window (and, on Splunk Enterprise, a copy of the application directory plus a KVstore backup). TrackMe also takes an automatic safety backup before the first migration runs, but the explicit one is the one you will want to restore from.

Follow the migration from the screen behind the version number. On the Virtual Tenants page the version number in the header is a link: it opens the **Tenants Update statuses** screen, which shows every tenant's schema version against the required one, its *updated* / *pending* status, and a one-click drilldown to the migration logs. It is the authoritative progress view; do not judge anything before it reads fully updated.

One release for an estate. When several TrackMe deployments are upgraded in a programme (managed service providers, multi-instance customers), validate one release on a representative test deployment and deploy that same release everywhere. Move the whole estate to a newer release later, in one deliberate wave.

Land on a current release. We recommend running the latest release at all times (see *Compatibility*). Never land on the 2.3.22 to 2.4.1 builds: they carry a regression where ML outliers model training silently stops for every component (fixed in 2.4.2).

A rollback is a restore, not an un-install. Schema migrations are forward-only. Reinstalling an older package reverts the code and none of the migrated data; the reliable rollback is to restore the pre-upgrade backups. See *Rollback*.

Validate the data, not the version marker. After a downgrade, the schema marker is silently realigned to the running version, so the update-status screen reports *updated* even though the data kept its migrated shape. Post-downgrade validation must inspect actual behaviour and actual records.

Licensing after an upgrade

Deployments that ran the discontinued Free Community Edition start a 90-day Foundation trial automatically after upgrading to 2.4.x; nothing stops, but the trial must be converted by registering a licence key. Licensed deployments keep their licence. Registering a key is step 2.4 of the *Upgrade procedure*; see also *License registration*.

7.2.7 Where to go next

- Read the *Upgrade procedure* end to end once, then execute it with its checkpoints.
- Keep *Schema migrations* open on upgrade day: it explains every line of the migration log.
- Use *Validating after an upgrade* for formal validation records and to brief operators on the behaviour changes between releases.
- Read *Rollback before* the change window, not after.

7.3 Migrating and Cloning TrackMe

Since TrackMe version 2.1.5, you can easily migrate TrackMe from a deployment to another, or cloning a deployment, using its powerful built-in backup and restore capabilities:

- Consult the white paper: *Backing up and Restoring TrackMe*.

7.4 License registration

7.4.1 TrackMe licensing modes

TrackMe has different licensing modes available:

- Foundation Edition trial mode (default)
- Unlimited edition / Enterprise Edition / Trial registered edition
- Developer mode
- Named Developer license

Once TrackMe has been installed, the **Foundation Edition trial mode** is enabled.

To register and manage the license, go to:

- Navigation bar / License, Help & support / License Management



TrackMe Licensing API services

TrackMe for the management of licensing relies on the services of Cryptolens:

- https://help.cryptolens.io

Under the following circumstances, an external call to the Cryptolens API service is made:

- Registering for a Trial license within TrackMe
- Registering the license key for Enterprise and Unlimited customers (online activation)

An external call to the Cryptolens API service is made to verify the validity of the license or generate a new Trial license key if requesting a Trial:

- https://help.cryptolens.io/security/api-ip

The traffic is outgoing:

- https://app.cryptolens.io:443

If you do not want this external connection, or if your Search Head cannot access this service, you can use the offline registration process using the license file which we will have provided.

Register your TrackMe license

If you are a licensed customer, you will have received your licensing information:

- A unique license key
- A license file containing the unique signature corresponding to your license

To register your license, click on the button “Add or manage license”:



You can choose to:

- Upload your license file containing the license signature, which does not require connectivity to our licensing API services
- Submit your license key, TrackMe will contact our licensing API services, verify and validate the license, and if successful will cache the license signature

Once successfully submitted, the license signature is cached in a KVstore, and all further license verifications are systematically made in an offline fashion without the need to contact our licensing API services.

Uploading your license file activation:



Submitting the license key for verification:



Once registered, the user interface shows the status and the expiration date, you can click on the “Add or manage license” button to review and download the license signature:





Start a Trial

You can request and start a Trial period immediately in TrackMe. A Trial period allows you to test all features with no restrictions for a limited period of time (90 days).

The Trial license can be generated directly from the application, although it requires that the Search Head has Internet connectivity to our licensing API.

If you need a Trial activation file for an environment with no connectivity, or wish to have a longer Trial period, please contact us at: support@trackme-solutions.com





Trial additional information:

- The Trial license is valid for a standalone Search Head or a Search Head Cluster (SHC) for a limited period of time.
- You can use all features with no limitations during the Trial period.
- You can enter a license key at any time during the Trial period to register a valid Full license.
- Once the trial period is terminated, TrackMe will automatically return to the Foundation Edition trial mode (read-only after expiry).
- When returning to the Foundation Edition trial once the Trial period is over, if you have created more Virtual Tenants than allowed by the Foundation Edition restrictions, these tenants will be automatically disabled.
- Similarly, Virtual Tenants using licensed restricting components (splk-flx, splk-fqm / splk-cim, splk-wlk) will be automatically disabled. The dedicated tenant types (User Activity Monitoring, splk-uam, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) are not available

on the Foundation Edition either: their collection and their workspace are refused until a license beyond the Foundation Edition is registered. The Volume Outliers component (splk-vol) is part of the Foundation Edition and is not affected.

Enable the developer mode

You can enable the developer mode in non-Production contexts, which allows you to use all features with no restrictions during a period of 30 days, up to a maximum of 3 activations per instance:

- The developer mode is valid on a standalone Search Head or a Search Head Cluster for a period of 30 days.
- You can use all features with no limitations during this period for your development, testing, and qualification usage only.
- Enabling the developer mode in a Production environment is a breach of TrackMe's EULA license agreement.
- Once the validity period is terminated, TrackMe will automatically return to the Foundation Edition trial mode (read-only after expiry).
- You can enable the developer mode up to 3 times per instance. Each enablement counts toward this limit, even if the current period is still active.
- When returning to the Foundation Edition trial once the initial period is over, if you have created more Virtual Tenants than allowed by the Foundation Edition restrictions, these tenants will be automatically disabled.
- Similarly, Virtual Tenants using licensed restricting components (splk-flx, splk-fqm / splk-cim, splk-wlk) will be automatically disabled. The dedicated tenant types (User Activity Monitoring, splk-uam, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) are not available on the Foundation Edition either: their collection and their workspace are refused until a license beyond the Foundation Edition is registered. The Volume Outliers component (splk-vol) is part of the Foundation Edition and is not affected.
- After the 3rd activation expires, contact TrackMe at support@trackme-solutions.com to obtain a **named developer license** for continued developer access with no restrictions.

Named Developer license

If you need ongoing developer access beyond the 3 self-service activations, TrackMe can issue a **named developer license**. This is a time-limited subscription license that provides full access to all TrackMe features with no restrictions for its validity period.

To request a named developer license, contact TrackMe at: support@trackme-solutions.com

Once issued, you register the named developer license using the same process as any subscription license (license key or file upload).





Foundation Edition trial mode

TrackMe defaults to the Foundation Edition trial mode for new or unlicensed deployments. The trial is time limited and becomes read-only after expiry. The Foundation Edition covers the DSM, DHM, MHM and VOL (Volume Outliers) components; FLX, FQM and WLK require the Enterprise or Unlimited Edition, and so do the dedicated tenant types (User Activity Monitoring, splk-uam, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge), which are also available on the Unlimited trial and on the developer license but not on the Foundation Edition or its trial. There are also restrictions in terms of features and the number of hybrid trackers:

Consult our pricing page on TrackMe's Website for more details about the restrictions and plans:

- https://trackme-solutions.com/pricing

If you need more information or if you have any questions, contact us at:

- contact@trackme-solutions.com

To return to the Foundation Edition trial mode from a paid subscription license, use the reset registration option available in the "Add or manage license" screen.

Note

The reset registration option is only available for subscription-based licenses. It is not available for Foundation Edition trial or developer mode license types.

CHAPTER

EIGHT

PRODUCT GUIDE:

8.1 Introduction

This part of the Product Guide gives you the mental model you need before configuring anything. Read it once, top to bottom — it is short — and the rest of the guide will make sense the first time.

- **Overview** — what TrackMe is, who it is for, and what it does for your Splunk platform.
- **Core concepts** — the vocabulary used everywhere else in this guide: entities, components, virtual tenants, state and score, policies, trackers.
- **How TrackMe works** — the monitoring cycle that turns raw `index=…` activity into per-entity states, scores, and alerts.
- **Navigating the UI** — an in-depth visual tour of the TrackMe interface.

Tip

New to TrackMe and want to get something running first? Start with the *Quickstart*, then come back here when you want to understand *why* it works the way it does.

8.1.1 What is TrackMe?

TrackMe is a Splunk app that **discovers, maintains, and monitors the availability and quality of your data**, and of the platform around it — across one Splunk deployment or many, at any scale.

Think of it as a control plane that sits *on top of* Splunk. It continuously reads what Splunk already holds and turns it into something you can actually operate on: per-entity health states for your data, **findings** for the platform around it — the people using Splunk, the forwarders, receivers and collectors carrying the data, the HEC endpoints receiving it and the Cribl pipelines in front of it — service-level objectives, stateful alerts, live topology maps and, optionally, AI-assisted investigation and triage.





What problem does it solve?

In a large Splunk environment, data silently breaks. A forwarder stops, a sourcetype starts arriving late, a field stops parsing, a scheduled search begins skipping. Splunk itself will not tell you — the searches simply return less, or nothing, and nobody notices until a detection misses or a dashboard goes blank.

The platform around the data fails just as quietly: an index doubles its licensed volume, an account runs all-time searches on a schedule nobody owns, two hundred forwarders drop behind a receiver tier, an intermediate forwarder pauses and holds every sender behind it, a database input stops collecting on a heavy forwarder that still reports, a HEC token goes silent behind a load balancer that still answers, a Cribl Destination backs up in front of Splunk.

TrackMe answers the questions that classic Splunk monitoring leaves open:

- **Is my data still flowing?** Every data source, host, and metric feed is tracked for delay, latency, and volume.
- **Is my data still good?** Field parsing quality and CIM compliance are monitored, not assumed.

- **Are my searches still healthy?** Scheduled-search execution, skipping, and errors are tracked as first-class entities.
- **Is my license volume normal?** Every index's licensed volume is tracked for drops, spikes, inactivity and trend.
- **Who is doing what on Splunk?** Accounts, their searches, their cost, their privileges and the knowledge objects they own.
- **Is my fleet forwarding?** Every forwarding agent, connected or not, at any fleet size; the intermediate forwarders and the receivers, pool by pool.
- **Is every input collecting?** The heavy forwarders and syslog collectors that run your add-on, scripted and file inputs, input by input.
- **Is the pipeline in front healthy?** HEC endpoints, pools and tokens; Cribl Stream and Cribl Edge, from their own metrics and Leader.
- **Who do I tell, and how urgently?** Priority, SLA, and impact scoring decide what is worth an alert and what is just noise.

Who is it for?

- **Splunk administrators** who need to prove the platform is healthy and catch silent failures before users do.
- **SOC and detection engineers** who depend on complete, timely, well-parsed data for their correlation searches.
- **Platform owners** who answer for the licence, the usage and the forwarding tier.
- **Service owners** who report availability and SLA attainment upward.
- **Managed service providers** who operate many isolated customer environments from one place.

What can you monitor?

Everything tracked in TrackMe is an **entity**, and every entity belongs to one of seven **monitoring components**. The component decides what is measured. Seven **dedicated tenant types** — User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge — raise **findings** instead of tracking entities (see *Findings, Finding groups and findings actions*).

Code	Component	What it tracks
DSM	Data Source Monitoring	`(index, sourcetype)` data feeds — delay, latency, volume, format quality.
DHM	Data Host Monitoring	Host × sourcetype activity — delay, latency, host count.
MHM	Metric Host Monitoring	Hosts sending `metric` data — metric lag per category.
VOL	Volume Outliers	The licensed volume of every index, read from the license usage logs — ML outliers (drops / spikes), inactivity, daily volume, trend and month projection.
FLX	Flex Objects	*Any* SPL you write, turned into a tracked entity with custom KPIs and dynamic thresholds.
FQM	Field Quality Monitoring	Field parsing quality and CIM compliance — coverage and success per field.
WLK	Workload Knowledge	Scheduled-search execution health — skip ratio, errors, execution delay.
UAM	User Activity Monitoring	Who is doing what on the platform — the accounts, their scheduled searches, their search activity and resource usage, read from the audit trail and the introspection and raised as **findings**. A dedicated tenant type rather than an entity component: see *User Activity Monitoring*.
Splunk HEC	Splunk HEC	The HTTP Event Collector endpoints of a Splunk deployment, their pools and tokens — availability, refused requests, back pressure, load balancing, rejected data, silent tokens — read from the HEC telemetry Splunk already writes and raised as **findings**. A dedicated tenant type: see *Splunk HEC*.
Splunk S2S	Splunk S2S	The receiving side of forwarding: the receivers, the intermediate forwarders and the pools they form — accepting or silent, back pressure and its origin, imbalance, churn, TLS and acknowledgement posture — read from the metrics and logs Splunk already writes and raised as **findings**. A dedicated tenant type: see *Splunk S2S*.
Splunk UF Fleet	Splunk UF Fleet	Every forwarding agent and the Deployment Servers that manage them, at any fleet size — connected or not, output blocked, destinations failing, restart loops, version drift, expected agents missing — read from the receivers and the agents' own logs and raised as **findings**. A dedicated tenant type: see *Splunk UF Fleet*.
Splunk Forwarder Collectors	Splunk Forwarder Collectors	The forwarders used as collection points — heavy forwarders running add-ons, scripted and modular inputs, and the forwarders reading syslog files — and every input they run: inputs that stop, fail, go quiet or get disabled, blocked outputs, groups below their minimum. Raised as **findings**. A dedicated tenant type: see *Splunk Forwarder Collectors*.
Cribl Stream	Cribl Stream	The health and traffic of a Cribl Stream deployment — Sources, Destinations, Pipelines, Routes and Workers — read from the Cribl metrics in Splunk and the Cribl Leader API and raised as **findings**. A dedicated tenant type: see *Cribl Stream and Cribl Edge*.
Cribl Edge	Cribl Edge	The Fleets of a Cribl Edge Leader, their Edge Nodes and the Sources and Destinations they run — connectivity, rollouts and version drift, health, disk — read from the Leader API at any fleet size and raised as **findings**. A dedicated tenant type: see *Cribl Edge*.

The short version: **DSM, DHM, and MHM** answer "is the data arriving?", **FQM** answers "is it well-formed?", **WLK** answers "are my searches healthy?", **VOL** answers "is my license volume normal?", **FLX** lets you monitor anything else by writing the SPL yourself. Seven dedicated findings tenants complete the picture: **UAM** answers "who is doing what?", a **Splunk HEC** tenant "are my HEC endpoints receiving, and who is refused?", a **Splunk**

S2S tenant "is my forwarding path healthy, pool by pool?", a **Splunk UF Fleet** tenant "is every forwarding agent connected?", a **Splunk Forwarder Collectors** tenant "is every input collecting?", a **Cribl Stream** tenant "is my Cribl pipeline healthy?" and a **Cribl Edge** tenant "is every Edge Node where it should be?".

Each component is covered in *Part 3 — Monitoring Components*.

How TrackMe is built

TrackMe is organised around three layers. You do not need to know the internals to use it, but the shape explains why the product behaves the way it does.

Layer	What it does
Data plane	Scheduled searches ("trackers") run on a cycle and a shared *decision maker* computes each entity's state and score. This is where "is it healthy?" is decided. Findings tenants run their own collection and policy jobs instead.
Control plane	A REST API and a per-tenant KV Store hold all configuration and current state — the single source of truth.
Presentation	A modern React interface (Virtual Tenants, Tenant Home, the findings workspaces, Topology Studio) and an optional AI Assistant.

The mechanics of the cycle are explained in *How TrackMe works*.

Editions and licensing

TrackMe ships in several editions (Foundation, Enterprise, Unlimited, and trial / developer variants). A new install automatically starts a **90-day Foundation Trial** with full functionality. When a license expires, TrackMe enters **read-only mode**: monitoring keeps running and all data, dashboards, alerts, and history stay visible, but creating or editing configuration is disabled until a license is applied.

See the *License* page for the full edition matrix and the *License registration* page to apply one.

See also

- *Core concepts and vocabulary* — the vocabulary used throughout this guide.
- *How TrackMe works* — the monitoring cycle in detail.
- *Quickstart* — get a tenant running in minutes.

8.1.2 Core concepts and vocabulary

TrackMe has a small, consistent vocabulary. These terms appear on every page of this guide and in the interface itself. Learn them once here.

The mental model in one sentence

> TrackMe tracks **entities** (grouped into **components**, isolated inside **virtual tenants**); on every cycle a **decision maker** gives each entity a **state** and a **score**, **policies** classify it, and **alerting** turns critical states into notifications.

Dedicated tenant types add a second, shorter sentence:

> A **dedicated tenant** watches the platform around the data; its **policies** raise a **finding** for each condition they detect on one object, and an object's findings are decided together as one **Finding group**.

Everything below expands one word of those sentences.

Entity

An **entity** is the fundamental unit TrackMe monitors — one data source, one host, one metric feed, one scheduled search, or one custom object you defined. Each entity has an `object_id` (its identity, e.g. an `index|sourcetype` pair), a current state, a score, and the metadata attached to it (priority, tags, labels, SLA, and so on).

There is no global list of entities — every entity lives inside exactly one virtual tenant.

Component

Every entity belongs to one of seven **components**, which decide *what* is measured: **DSM**, **DHM**, **MHM**, **VOL**, **FLX**, **FQM**, **WLK**. The component determines the metric axes (delay, latency, volume, licensed volume, quality, execution health…) and the tracking logic applied. See *What is TrackMe?* for the one-line description of each, and Part 3 for the deep dives.

The products that watch the platform around the data are not more components: they are dedicated tenant types, described next.

Dedicated tenant type

A **dedicated tenant type** is a tenant that has no entities and no state colour. It watches one part of the platform around your data and raises **findings** instead. There are seven:

- **User Activity Monitoring (UAM)** — the audit trail and the introspection of a deployment turned into accounts, scheduled searches, activity, resource usage and privileges. See *User Activity Monitoring*.
- **Splunk HEC** — the HTTP Event Collector endpoints, pools and tokens of one Splunk deployment, from the telemetry Splunk already writes. See *Splunk HEC*.
- **Splunk S2S** — the receiving side of forwarding: the receivers, the intermediate forwarders and the pools they form. See *Splunk S2S*.
- **Splunk UF Fleet** — every forwarding agent and the Deployment Servers that manage them, at any fleet size. See *Splunk UF Fleet*.
- **Splunk Forwarder Collectors** — the forwarders used as collection points and every input they run: add-ons, scripted and modular inputs, syslog files. See *Splunk Forwarder Collectors*.
- **Cribl Stream** — the Cribl metrics already in Splunk and the Cribl Leader API turned into the inventory, the traffic, the Workers and the findings of one Cribl Stream deployment. See *Cribl Stream and Cribl Edge*.
- **Cribl Edge** — the Fleets of a Cribl Edge Leader and their Edge Nodes, read in bulk from the Leader's node directory. See *Cribl Edge*.

A tenant is either an entity tenant or one dedicated tenant type, never both. The seven share one findings model: policies, findings, Finding groups, assignment, notifications, findings actions and an AI Agent. See *Findings, Finding groups and findings actions*.

Finding

A **finding** is raised by a policy of a dedicated tenant about one object — an account, a HEC endpoint, a receiver pool, a forwarding agent, a collector's input, a Cribl Destination — with its evidence and a severity from Informational to Critical. The same condition on the same object is always the same finding: it keeps its notes and its history across runs, and recovers by itself when the condition clears. A healthy object has no finding at all. See *A finding and its lifecycle*.

Finding group

A **Finding group** gathers every finding about one object: one status, one assignee, one severity and one notification thread. It is the unit you triage — investigate, resolve, dismiss, assign or snooze — so an object with three problems is still one decision. See *Finding groups*.

Virtual Tenant

A **virtual tenant** is an isolated monitoring domain — typically one customer, one environment, or one team. Each tenant has its own entities, its own configuration and policies, its own set of Splunk roles, and optionally its own remote Splunk deployment to monitor.

Tenants are the unit of isolation and the unit of scale: you create, scope, experiment with, and destroy them independently. Everything else in TrackMe is scoped *to* a tenant.

Virtual Group

A **virtual group** is a read-only card that aggregates entities from several tenants into a single cross-tenant view — useful for a NOC screen that spans many customers. A virtual group is *not* a tenant: it does not run monitoring of its own, it only summarises what the underlying tenants already computed.

Entity state

At every cycle each entity is assigned an **object state** — one of four colours:

State	Meaning
green	Healthy — no active anomaly.
orange	Warning — a non-critical anomaly is active (score above zero, below critical).
red	Critical — the entity has crossed the critical threshold.
blue	Suppressed — the entity is *protected* from alerting (logical-group protection, a disruption grace period, or maintenance), even though an anomaly may be present.

The state machine and the blue "protected" state are covered in Part 4.

Impact score

The state is derived from a numeric **impact score**. TrackMe adds up the contributions of everything that is currently wrong with an entity — delay breach, latency breach, ML outliers, breached dynamic thresholds — into a single `total_score`:

- `total_score == 0` → green
- `0 < total_score < 100` → orange
- `total_score ≥ 100` → red

The weight of each contribution is configurable per tenant and can be overridden per entity, so you decide whether (say) a latency breach matters more than a volume drop. Impact scoring is covered in Part 4.

Tracker

A **tracker** is a scheduled search that runs on a cycle and feeds the decision maker. Trackers are what make TrackMe *active* rather than a static dashboard — they periodically re-evaluate every entity. Most are managed for you; advanced users can add **hybrid trackers** (Part 7) to inject extra signals.

Decision maker

The **decision maker** is the shared engine that, each cycle, reads an entity's metrics and metadata and computes its state and score. It is the single place where "is this entity healthy?" is answered, so the answer is consistent across the UI, alerts, and the API.

Policy

Manually setting priority or SLA on thousands of entities does not scale. A **policy** assigns metadata automatically using a rule — a regex, a Splunk lookup, or an SPL search. Three policy families exist: **priority**, **tags**, and **SLA**. Policies *suggest*; a manual override always wins, and removing a policy auto-cleans the assignments it made. Policies are covered in Part 5.

On a dedicated tenant, a **policy** is something else: the rule that evaluates one condition and raises a finding. Each one is built in or written in SPL, enabled or not, with a default severity and its own settings.

Priority, tags, labels, SLA

These are the four kinds of classification metadata you attach to entities:

- **Priority** — `critical / high / medium / low / pending`. How much an entity matters.
- **Tags** — free-form string metadata for filtering and grouping.
- **Labels** — a curated, colour-coded catalogue of chips (distinct from tags: different storage, different UI, and assignable by rules).
- **SLA class** — `platinum / gold / silver` with associated thresholds.

Note

Tags and labels are not the same feature. Both are chip-style metadata, but tags are free-form strings while labels come from a managed, coloured catalogue and can be assigned automatically by rules. Part 5 shows them side by side.

Logical group

A **logical group** clusters N entities and judges them *collectively*. If the percentage of green members stays above a threshold, struggling members are demoted from red to blue ("group protection") — so one flaky member of a healthy cluster does not page you.

Disruption queue

The **disruption queue** is a grace period. When an anomaly first appears, the entity is held in blue until the anomaly has persisted for a configured minimum time — absorbing transient blips so only sustained problems escalate to red.

Acknowledgment (ACK)

An **acknowledgment** suppresses further notifications for an entity you already know about. *Sticky* ack stays until you clear it; *unsticky* ack expires automatically after a set period. Ack can be revoked automatically if the underlying anomaly reason changes.

Notable event

A **notable event** is the traditional Splunk alerting output — an event written for Enterprise Security or a custom correlation workflow. It is one of several delivery channels TrackMe's stateful alerting can drive; see Part 6.

See also

- *How TrackMe works* — how these pieces interact on every cycle.
- *What is TrackMe?* — what TrackMe is, the seven components and the seven dedicated tenant types.
- *Findings, Finding groups and findings actions* — findings, Finding groups and findings actions.

8.1.3 How TrackMe works

This page traces the path from raw Splunk activity to an alert in your inbox, or to a finding in your work queue. It is the "why does it behave like that?" companion to the vocabulary in *Core concepts and vocabulary*.

TrackMe watches in two ways. **Entity tenants** (DSM, DHM, MHM, VOL, FLX, FQM, WLK) give every entity a state on a monitoring cycle, described first. **Dedicated tenants** (User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge) run the same idea as a findings cycle, described in *The findings cycle — dedicated tenants*.

The monitoring cycle

TrackMe is not a live dashboard that queries Splunk when you open it. It is a set of **trackers** (scheduled searches) that run on a cycle and persist their results. Opening the UI just reads the last computed state. This is what lets TrackMe scale to thousands of entities and alert without anyone watching a screen.

Each cycle, for every entity tenant, the following happens:

```
1. DISCOVER     trackers scan the tenant's indexes and find entities
                (new (index,sourcetype) pairs, hosts, searches, …)
                     |
2. MEASURE      each entity's metrics are refreshed
                (delay, latency, volume, parsing quality, execution health)
                     |
3. DECIDE       the decision maker computes, per entity:
                  • impact score  = Σ (delay, latency, outliers, thresholds…)
                  • object state  = green / orange / red   (from the score)
                     |
4. CLASSIFY     policies assign priority, tags, SLA; labels are evaluated
                     |
5. PROTECT      logical groups, disruption grace, maintenance, ack
                may demote an entity to blue (suppress alerting)
                     |
6. ALERT        stateful alerting opens / updates / closes alert records
                and delivers email, notable events, ingest, commands
```

Steps 3–6 are the heart of the product. The rest of this guide is, essentially, a detailed tour of each step.

Step 3 — how state is decided

The decision maker computes a single `total_score` by summing every active problem:

```
total_score =  base_score              (from the metrics index)
            +  impact_score_delay       (if the delay threshold is breached)
            +  impact_score_latency     (if the latency threshold is breached)
            +  impact_score_outliers    (from ML outlier detection)
            +  threshold_scores         (sum of breached dynamic thresholds)
            +  …
```

The score is then mapped to a colour: `0` is green, `< 100` is orange, `≥ 100` is red. Because every contribution is just a number with a configurable weight, you control exactly what makes an entity "critical" — see Part 4 for tuning impact scores and thresholds.

Step 5 — why an entity might *not* alert

A red score does not always mean a page. Four mechanisms can hold an entity in **blue** (suppressed) instead:

- **Logical group protection** — the entity's group is mostly green, so this member is carried.
- **Disruption grace** — the anomaly is too new; TrackMe waits to see if it persists.
- **Maintenance mode** — the entity, tenant, or whole instance is in a maintenance window.
- **Acknowledgment** — someone already accepted this problem.

This separation — *what state* (decision maker) versus *whether to tell you* (protection and alerting) — is deliberate, and it is why TrackMe is quiet in practice despite tracking so much.

Step 6 — how you get told

The **stateful alerting** system tracks each anomaly as it opens, persists, and closes:

- It opens a stateful record when an entity first turns red (or orange, if you opt in), updates it each cycle the problem persists, and closes it when the entity goes green.
- On those transitions it can deliver an **email** (with embedded 24-hour charts and, optionally, an AI-written incident summary), write a **notable event**, ingest an **event**, and/or run a **saved-search command**.

Alerting is covered in Part 6.

The findings cycle — dedicated tenants

A dedicated tenant has no entities and no state colour. Its scheduled jobs collect what it watches, and its policies raise **findings** about the objects that need a look; a healthy object has none. Each run, for every dedicated tenant:

```
1. COLLECT      collection jobs read what the tenant watches from what Splunk
                (or the Cribl Leader) already holds: inventory, activity, metrics
                     |
2. EVALUATE     the policies, built in or your own SPL, evaluate their condition
                against what was collected, object by object
                     |
3. RAISE        each condition detected on one object is a finding, with its
                evidence and a severity; a condition that clears recovers by itself
                     |
4. GROUP        an object's findings form one Finding group: one status, one
                assignee, one severity, one notification thread
                     |
5. ROUTE        assignment rules route the group to a team; snoozes and bank
                holidays hold what should wait
                     |
6. NOTIFY       an email digest or one threaded email per Finding group, webhooks,
                indexed finding events; the AI Agent can review and triage
```

The same separation holds as for entities: a policy decides *what is wrong*, and a person — or, within what the tenant allows, the AI Agent — decides what to do about it. Nothing is installed on the platforms being watched. Findings, Finding groups and everything around them are covered in *Findings, Finding groups and findings actions*.

Multi-tenancy and remote deployments

Everything above happens **per tenant**. A tenant can monitor the local Splunk search head, or point at a **remote Splunk deployment** (Enterprise or Cloud) over an authenticated, token-rotated connection with automatic failover. From the user's point of view the experience is identical — only the data source differs. Remote deployments are covered in Part 8.

Where the AI fits

TrackMe's AI features sit *alongside* this cycle, never in the critical path:

- The **AI Assistant** is a context-aware chat drawer on every page, off by default until you configure a provider.
- **AI Advisors** are on-demand agents that can inspect TrackMe's state and propose (or, with consent, apply) tuning.
- **AI Routines** are scheduled agents you configure to evaluate TrackMe on a cadence and then notify (email or Splunk event) or, with granted scope, act on what they find.
- **AI status reports** are the LLM summaries optionally embedded in alert emails.
- The **AI Agent** of each dedicated tenant type reviews findings on demand or on an hourly schedule, recommends a decision and, within what the tenant allows, applies it.

All of this is opt-in and documented in the dedicated *Artificial Intelligence* section.

See also

- *Core concepts and vocabulary* — definitions for every term used above.
- *Findings, Finding groups and findings actions* — findings, Finding groups and findings actions.
- *Artificial Intelligence* — the AI Assistant, Advisors, Routines and the AI Agents for findings.

8.1.4 Navigating the TrackMe UI — in depth

Tip

This is an **in-depth visual tour** of the TrackMe user interface. For the conceptual introduction, start with *Overview* and *How it works*. This page walks the Virtual Tenant interface, the Tenant Home, the three-dot menus, bulk editing, theme preferences, and hiding tenants.

Note

TrackMe Design

- This page describes the essential concepts and design principles behind TrackMe's user interfaces.
- TrackMe 2.3.0 introduced a complete rewrite of the user interfaces, built on Splunk's native React UI.
- It covers the key things to know to navigate the interface and use its features.

TrackMe has **two primary interfaces**: the **Virtual Tenant interface** — the landing page where you see and manage all your tenants — and the **Tenant Home** — the rich per-tenant screen where you work with entities. A dedicated findings tenant (User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge) opens its own **findings workspace** in Tenant Home instead of an entity table; its conventions are described in *The shared workspace conventions*. A consistent **three-dot menu** exposes actions everywhere, and a per-user **theme** (dark or light) applies across both. This page tours each in turn.

The Virtual Tenant interface

When you open TrackMe, the default page is the **Virtual Tenant interface**. It gives an overview of your Virtual Tenants and the actions you can perform on them — such as creating, managing, or entering a specific tenant.



The home page at a glance

The key interactions of the Virtual Tenant interface:

main screen:



tenant overview:



The three-dot menu

The three vertical dots in the top right of the screen are the **main selector menu**. They give quick access to the main actions, such as creating a new Virtual Tenant:





As a core design principle, **every action in TrackMe** — whether on a Virtual Tenant, an entity, a chart, and so on — is available through this same three-dot menu, placed on the relevant element.

Opening a Virtual Tenant overview

Double-click a Virtual Tenant card to open its **overview** — a modal that summarises the tenant and offers actions such as managing or entering it.





The tenant actions menu

Click the tenant's three-dot icon to open the menu specific to that tenant. The available actions depend on your privileges and the tenant's status.



Reviewing entities (read-only)

From the Virtual Tenant overview you can browse the tenant's entities in **read-only** mode. Many actions are available — for instance, expand an entity to review detailed information and mini-charts.





Hiding a Virtual Tenant

You can **hide** a specific Virtual Tenant from the interface (tenant menu → *Personal preferences* → **Hide this tenant**); the choice is kept in your browser with your other personal preferences and affects no one else. In a large environment this is useful when a user has access to a tenant (through *RBAC*) but does not need to see it in the UI. A hidden tenant's menu offers **Unhide this tenant**, and once at least one tenant is hidden the Settings panel shows a **Virtual Tenants visibility** switch to display them all again.





The Tenant Home interface

The **Tenant Home** is the main interface for managing a tenant's entities — a rich screen whose interactions vary by component type. You can reach it several ways from the Virtual Tenant interface:





Key interactions

The key interactions in the Tenant Home (the exact view varies by component type):



key interactions from the entity view:



Entity and tenant menus

As everywhere in TrackMe, actions are reached through three-dot menus (the exact items depend on the component type):

the tenant menu:



the entity menu from the top selector:



the entity menu from the entity table:



Bulk editing

The Tenant Home supports **dynamic bulk editing**: select one or more entities and the bulk action button appears automatically.







Themes & personal preferences

TrackMe has two colour schemes — **Dark** and **Light** — applied globally across all interfaces. Each user's choice is kept in a browser cookie with their other personal preferences; until they choose, the administrators' default applies (dark after installation).

Setting your theme

From the Virtual Tenant interface, a Tenant Home or Topology Studio, click the **settings** icon (the sliders button) at the top right to open the **Settings** panel, then pick **Light** or **Dark** under **Color Scheme**. The same panel holds the other personal preferences — auto-refresh, volume unit labels, the tenant card detail level, and on a Tenant Home the layout of that tenant's page:





TrackMe in Light mode:



Every preference, the administrators' defaults and how to reset them are described in *Personal preferences*.

See also

- *What is TrackMe?* and *How TrackMe works* — the conceptual introduction.
- *Virtual Tenants & Access* — creating, managing, and securing Virtual Tenants.
- *Monitoring Components* — the per-component views the Tenant Home adapts to.

8.2 Virtual Tenants & Access

Everything in TrackMe lives inside a **virtual tenant** — an isolated monitoring domain with its own data, configuration, trackers, and access control. This part covers the full tenant lifecycle and the role model that governs who can see and change what.

- **Overview** — what a tenant is, why you create them, and a tenant's anatomy.
- **Creating a tenant** — the creation wizard and the choices it asks you to make.

- **Managing tenants** — editing, enabling/disabling, deleting, operational status, and scheduling health.
- **Virtual groups** — read-only cards that aggregate entities across several tenants.
- **Roles & access control** — the three-role model and how to scope access cleanly.
- **Personal preferences** — per-user theme, refresh and layout settings, and the defaults administrators set for them.

Note

A fresh TrackMe install has **no tenants yet** — creating your first tenant is the first thing you do. If you haven't, start with the *Quickstart*.

8.2.1 Virtual Tenants explained

A **virtual tenant** is TrackMe's unit of multi-tenancy: a self-contained, isolated instance of TrackMe living inside your Splunk app. It owns its data, its configuration, its trackers, its access control, and — optionally — its own remote Splunk deployment to monitor. You create, scope, operate, and destroy tenants independently of one another.





Why tenants exist

Manually, a single flat monitoring space does not survive contact with a real Splunk estate. Tenants let you carve that estate into domains that match how your organisation actually works. Common reasons to create separate tenants:

- **Segmentation** — by company, country, business unit, or team.
- **By technology** — keep firewall data, performance data, and application logs in their own spaces with their own thresholds.
- **By scope** — one tenant per Splunk cluster, per index family, or per search head tier.
- **Access control** — give different teams ownership of different data spaces without exposing each other's entities (see *Roles & access control*).

Everything else in TrackMe — entities or findings, policies, thresholds, alerts — is scoped *to* a tenant. There is no global entity list.

Anatomy of a tenant

From a user's point of view, a tenant is made of the following:

Element	What it is
Tenant identifier (`tenant_id`)	The immutable internal name (lowercase letters, digits, underscores; up to 40 characters). It is stamped as an indexed field on every event and metric TrackMe generates for the tenant. **Choose it carefully — it cannot be changed later.**
Tenant alias	A human-readable display name shown in the UI and used to order tenants. Editable at any time.
Description	Free text describing the tenant's purpose.
Status	`enabled` or `disabled` — the source of truth for whether the tenant's trackers run and its entities are maintained.
Owner	The Splunk user that owns the tenant's knowledge objects and on whose behalf its trackers execute.
Enabled components	Which monitoring components the tenant runs, or the one dedicated tenant type it is (see below).
Indexes	The Splunk indexes the tenant uses (summary, audit, metric, notable).
Default settings	Per-tenant defaults — time range, monitoring policy, default priority, ML outliers configuration — that cascade down to entities and can be overridden per entity.

Components and tenant types

A tenant is either an **entity tenant** or a **dedicated tenant**. An entity tenant runs one or more **monitoring components**: when you create it you choose which to enable, and each can be turned on or off later. A dedicated tenant is exactly one of the seven **dedicated tenant types**, which have no entities and raise findings instead (see *Findings, Finding groups and findings actions*); it never mixes with entity components or with another dedicated type.

Tenant type	Components it brings
splk-feeds	The data-flow family: **DSM** (Data Source Monitoring), **DHM** (Data Host Monitoring), and **MHM** (Metric Host Monitoring). Each is enabled independently.
splk-vol	**VOL** — Volume Outliers: the licensed volume of every index, read from the license usage logs. One tenant covers a whole Splunk deployment, local or remote.
splk-wlk	**WLK** — Workload Knowledge: scheduled-search execution health and SVC consumption.
splk-flx	**FLX** — Flex Objects: custom trackers you define with your own SPL.
splk-fqm	**FQM** — Fields Quality Monitoring.
splk-cim	Common Information Model compliance tracking.
splk-uam	**UAM** — User Activity Monitoring: a *dedicated* tenant type with no entity component — the accounts, their scheduled searches, activity and resource usage on one indexing layer, through one or more search head tiers, turned into findings. A tenant is either an entity tenant or a UAM tenant.
cribl-stream	**Cribl Stream**: a *dedicated* tenant type with no entity component — one Cribl Stream connection, its Sources, Destinations, Pipelines, Routes, Packs and Workers, read from the Cribl metrics in a Splunk metric index and from the Cribl Leader API, with findings from policies and traffic ML models. Requires the Enterprise or Unlimited edition.
cribl-edge	**Cribl Edge**: a *dedicated* tenant type with no entity component — the Fleets of one Cribl Edge Leader and their Edge Nodes, Sources and Destinations, read in bulk from the Leader's node directory at any fleet size, with findings from policies. Requires the Enterprise or Unlimited edition. A tenant is either Cribl Stream or Cribl Edge.
Splunk HEC	A *dedicated* tenant type with no entity component — the HTTP Event Collector endpoints, pools and tokens of one Splunk deployment, local or remote, read from the HEC telemetry Splunk already writes, with findings from policies and traffic ML models. See *Splunk HEC*.
Splunk S2S	A *dedicated* tenant type with no entity component — the receivers, the intermediate forwarders and the pools they form on one Splunk deployment, the receiving side of forwarding, with findings from policies and traffic ML models. See *Splunk S2S*.
Splunk UF Fleet	A *dedicated* tenant type with no entity component — every forwarding agent of one Splunk deployment and the Deployment Servers that manage them, at any fleet size, with findings from policies. See *Splunk UF Fleet*.
Splunk Forwarder Collectors	A *dedicated* tenant type with no entity component — the forwarders one Splunk deployment uses as collection points and every input they run (add-ons, scripted and modular inputs, syslog files), with findings from policies and traffic ML models. See *Splunk Forwarder Collectors*.

The four Splunk dedicated tenant types are not available with the Foundation edition or the Foundation trial.

Each component is covered in *Part 3*.

Saved Views

On a large estate the Virtual Tenants grid can hold dozens of tenant and Virtual Group cards. A **Saved View** is a named, curated subset of that grid — a hand-picked set of tenant cards (optionally narrowed to specific components) plus Virtual Group cards — so a team lands on just the tenants relevant to them instead of the whole estate. It is a pure **display** convenience: Saved Views never touch monitoring, scoring, or alerting.

A power or admin user creates one from the Virtual Tenants page (**Create a saved view**) via a short wizard — identity, the tenants and components to include, the Virtual Groups to include, and the **roles** allowed to use it. Each user then picks a view from the selector in the search-bar row (the default is *All tenants & groups*); the choice is

remembered per browser and silently falls back to *All* if the view is later removed or access is revoked.

Note

Saved Views are **RBAC-scoped** — a view lists the Splunk roles allowed to see it (administrators always can). Creating and managing views requires the power capability (`trackmepoweroperations`); simply using one needs only read access.

License limits

Some license editions **cap the number of tenants** you can create. If you reach the limit, creation is refused until you remove a tenant or apply a higher edition. See the *License* page for the edition matrix.

See also

- *Creating a tenant* — create your first tenant.
- *Roles & access control* — control who can see and manage a tenant.
- *Virtual Groups* — aggregate entities across several tenants.

8.2.2 Creating a tenant

Tenants are created from a guided wizard in the Virtual Tenants UI (and, for automation, through the REST API). The wizard adapts to the component type you choose, but every path ends with the same access-control step.




Create a new TrackMe Virtual Tenant: Splunk Feeds - DSM
Basics
Configure DSM
DeFault delay
ML Outliers
Indexes & RBAC
Review
Provide an identifier for your Virtual Tenant. This cannot be changed later.
Tenant Id:
secops-feeds
This tenant id is available
Tenant alias (optional)
Tenant description (optional)
Impact score configuration
Help – concepts & tips
Advanced tenant options
Back
Next


Create a new TrackMe Virtual Tenant: Splunk Feeds - DSM
Basics
Configure DSM
DeFault delay
ML Outliers
Indexes & RBAC
Review
Configure Data Source Monitoring (SPLK-DSM) options for this tenant
Create tracker now?
Yes
Splunk deployment
local
Indexes discovery (optional) - allowonly these index patterns
business_data
siem*
Test now
Resulting search root constraint for this tracker:
index=business_data OR index=siem*
{ [-]
indexes: [[+]],
dcount_entities: "116",
entities_sample: [[+]]
}
Help – concepts & tips
Advanced options
Back
Next


Create a new TrackMe Virtual Tenant: Splunk Feeds - DSM
Basics
Configure DSM
Default delay
ML Outliers
Indexes & RBAC
Review
Define the default delay management behavior for all entities discovered in this Virtual Tenant.
Static threshold — A single delay limit (e.g. 1 hour) applies at all times. Simple and predictable.
Variable threshold — Different delay limits apply by day and hour (e.g. 1h during business hours, 4h at night, 24h on weekends). Better for feeds with known activity patterns.
Default delay policy
Variable threshold — different limits by day and hour
Timezone notice
Quick templates
Business hours vs. off-hours
Weekday vs. weekend
Three-tier (business / nights / weekends)
Time slots (3)
Add slot
business_hours
Remove
Back
Next


Create a new TrackMe Virtual Tenant: Splunk Feeds - DSM
Basics
Configure DSM
Default delay
ML Outliers
Indexes & RBAC
Review
Outliers detection scope
Priority filter
Critical
High
Medium
Low
Filter expression (optional)
Filter Expression
Back
Next





Important

The Splunk **indexes** a tenant uses (summary, audit, metric, notable) must already exist before you create the tenant. The wizard references them; it does not create them.

Identity (all tenant types)

Every tenant begins with its identity:

- **Tenant identifier** (`tenant_id`) — unique and **immutable**. Lowercase letters, digits, and underscores, up to 40 characters.
- **Tenant alias** — optional human-readable display name (editable later).
- **Description** — optional free text.

Creating a splk-feeds tenant

The feeds wizard is the richest, because it configures live discovery of your data. You choose which feeds components to enable — **DSM**, **DHM**, **MHM** — and, per component, set:

- **Create tracker now** — optionally create a tracker during creation (this produces a *Hybrid Tracker*; you can also add trackers later from inside the tenant).
- **Splunk deployment** — whether the data is **local** to this Splunk, or on a **remote** Splunk deployment (see *Integrations & Remote Deployments*).
- **Splunk root search constraint** — the base constraint that governs which entities are discovered and maintained.
- **Restrict indexes discovery** — narrow the scope further with explicit and wildcard index patterns.
- **Advanced options** — Hybrid-tracker settings plus component-specific options.
- **Test now** — run a preview search against your settings and confirm entities are found before committing.

Component nuances:

Component	What you configure
DSM	Tracks data feeds by `(index, sourcetype)`, with an optional custom *break-by* field. Produces KPIs, ML outliers, and data-quality signals.
DHM	Tracks sourcetype activity per **endpoint**. The endpoint defaults to the `host` metadata but can be any custom field.
MHM	Tracks metric-category availability per endpoint. The endpoint defaults to `host` but can be any metric dimension.

Creating a splk-vol tenant

The **Splunk Volume (Outliers)** card creates a *Volume Outliers* tenant: one entity per Splunk **index**, fed by the license usage log, with ML outliers on the licensed volume (drops and / or spikes), inactivity detection, the daily licensed volume, a trend and a month projection. One tenant covers a whole Splunk deployment; there are no use cases to pick. VOL is available on the **Foundation** edition as well as Enterprise and Unlimited.





The wizard has five steps:

1. **Basics** — the tenant identifier, alias and description, with an optional impact score configuration. Expand **How it works** for the pipeline in one picture.
2. **Volume** — how the volume is collected and how the detection behaves:
   - **Target environment** — this Splunk deployment (`local`), or a **remote account** whose license usage is collected; a remote target runs the connectivity check.
   - **Detect** — the default detection direction of every new index: **drops and spikes**, drops only, or spikes only. Each index can be changed afterwards with one click.
   - **Backfill history (days)** — no backfill, 7, 14, 30 (default), 60 or 90 days, or **Custom…** for any number of days up to 400, capped by the retention of the license usage log. The history is replayed after creation, one license day at a time by the hourly tracker, and the models are trained once the replay completes: a long span takes longer to reach a normal confidence.
   - **Outliers on these priorities** — the priorities in scope for volume outliers, every priority by default

(volume outliers are the point of this component).

- **Outliers minimum history (days)** — the history a model needs before its confidence is *normal* (15 by default on a VOL tenant); the step tells you whether the backfill covers it.
- **Outliers KPIs** — the rolling KPIs every new index gets a model on (the rolling 24 hours by default).
- **Advanced** — the **license search constraint** (leave the default, `index=_internal source="*/license_usage.log"`, unless the license usage events live elsewhere — see *The license search constraint (advanced)*) and the **settle margin** before a 5-minute bucket is read.

3. **Inactivity** — the tenant's inactivity policy: after how long without licensed volume an index turns red. The default is **variable**: 1 day during working hours (Monday to Friday, 08:00–19:59 server time), 7 days for nights and week-ends. Quick templates, a static threshold, or your own day / hour slots.
4. **Indexes & RBAC** — the access-control and indexes step common to every tenant.
5. **Review** and create. The first collection runs immediately; the backfill job plans itself at its first hourly run, and the tenant card reads **Backfill pending** until then, then **Backfilling N %**.




Create a new TrackMe Virtual Tenant: Splunk Volume (Outliers)
VOL tracks the licensed volume of every index from the license usage logs, in 5-minute buckets. Outliers on the rolling 24 hours volume detect abnormal drops and / or spikes, and the licensed volume is recorded per license day with a month projection. There are no use cases to choose: pick the environment and how the detection behaves.
Target environment
local
The Splunk deployment whose license usage is tracked: this instance, or a remote account.
Detect
Drops and spikes
The default detection direction for every index; each index can be changed afterwards.
Backfill history (days)
30 days
History replayed after creation, one license day at a time by the hourly tracker: the rolling volume the outliers train on and the daily licensed volume of the cost trend. The models are trained once the replay completes, so a long span takes longer to reach a normal confidence. Capped by the _internal retention.
Outliers on these priorities
critical
high
medium
low
Indexes of these priorities are in scope for volume outliers. All priorities by default: volume outliers are the point of this component.
Outliers minimum history (days)
15
Metrics history an outliers model needs before its confidence is normal: a low-confidence model does not add to the impact score. A tenant setting (the system default is 30 days): it applies to every component of this tenant and can be changed later in the tenant configuration. 1 to 365.
Outliers KPIs
Rolling 24h volume
Every index gets one outliers model per KPI selected here. The rolling 24h volume follows the daily pattern; the shorter windows react faster to a sudden drop or spike. Changeable later in Tenant Home (Manage: Outliers default KPIs).
With 30 days of backfill, the outliers models reach a normal confidence once the replay completes (the hourly tracker replays one day at a time).
Advanced
Settle margin (seconds)
300
License usage events newer than now minus this margin are not read yet, so late events are counted. 60 to 3600, default 300.
License search constraint
1 index=_internal source="*/license_usage.log"
The base search giving access to the license usage events: every license search of the tracker is formed from it. Indexed fields only (index, source, sourcetype, host). Change it only to target another data set with the license_usage.log format, whose key=value fields (type, idx, b, pool) are extracted at search time. Default: index=_internal source="*/license_usage.log"
Back
Next


Create a new TrackMe Virtual Tenant: Splunk Volume (Outliers)
Basics
Volume
3
Inactivity
4
Indexes & RBAC
5
Review
An index whose last licensed volume is older than the threshold in force turns red. This global policy applies to every index without its own; each index can override it later (entity menu or bulk edit).
Inactivity policy
Variable (per day and hour)
Timezone notice
Threshold slot days and hours are evaluated in the Splunk server's local time, not your browser's time.
Splunk server: Sat 31 Oct 2026, 01:54:24 (UTC, +00:00) Your browser: Sat 31 Oct 2026, 01:54:25 (GMT, +00:00)
Your browser is in sync with the server. Enter slot hours as you want the Splunk server to read its clock — for example, to monitor 09:00–18:00 in your local time, apply the offset above before picking hours here.
Quick templates
Working hours vs. off-hours
Weekdays vs. week-ends
Three-tier (working hours / nights / week-ends)
Time slots (1)
+ Add slot
The first slot matching the current day and hour applies. Thresholds accept seconds or a unit suffix (30m, 12h, 1d, 1w); 0 disables the check in that slot.
working_hours
1d
1d
Remove
Days:
Select all
Mon
Tue
Wed
Thu
Fri
Sat
Sun
Hours:
Select all
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
Outside the slots:
7d
7d
Back
Next



Tip

A single VOL tenant per Splunk deployment is the natural shape: the license usage log already covers every index. Monitor several deployments from one TrackMe with one VOL tenant per remote account.

Creating a splk-uam tenant

The **User Activity Monitoring** card (the first of the chooser's dedicated workspaces) creates a *UAM* tenant: a **dedicated tenant type** — no entity component can be added to it — that inventories the scheduled searches, users and roles of one or more **search head tiers**, reads the search activity and the resource usage of the shared indexing layer, classifies the accounts and raises **findings** from a catalogue of policies. UAM is available on the **Enterprise** and **Unlimited** editions, the Unlimited trial and the **Developer** license; the card is refused on the Foundation edition and the Foundation trial.





The wizard has four steps:

1. **Basics** — the tenant identifier (lowercase letters, digits and hyphens, up to 20 characters for a UAM tenant), alias and description.
2. **Configure UAM** — the shape of the tenant:
   - **Search head tiers** — the local search head and / or remote accounts, every selected account feeding the **same indexing layer** (one remote account per tier of that layer). Every remote tier runs its connectivity test and must pass it before the creation is allowed. Every selected tier also runs a **UAM access check** (the local one with the owner chosen under *Indexes & RBAC*, again whenever it changes); its permission warnings do not block the creation. The **route** account of the indexing-layer searches (audit activity, resource usage, capacity, account signals) is derived: the local search head when selected, else the first tier.
   - **Historical backfill** — the range of audit and resource history collected before the live start, hourly in 5-minute buckets, until covered: **none** (the default), 7, 14, 30, 60 or 90 days, bounded by the retention of the sources (the introspection retention is typically shorter than the audit one). Changeable later in *Settings → Collection limits*.
   - **Activity scope (advanced)** — optional search head hostname patterns to include or exclude, written into the base searches so that the indexers return only the activity of the search heads in scope.
3. **Indexes & RBAC** — the access-control and indexes step common to every tenant. Prefer a **dedicated service account with administrative rights** as the owner (`svc-trackme`, for instance): the trackers and the scheduled AI review run as the owner, who must administer the tenant. Classify that account and the administrators as **system** in the workspace afterwards, so they never appear in the findings.
4. **Review** and create. The trackers are scheduled immediately — one inventory report per tier every 15 minutes with its full listing as a report of its own (the first within the hour, then daily off-peak), the audit activity and the resource usage every 5 minutes from their durable checkpoints, the account signals and the policies every 15 minutes — and, when a range was chosen, the backfill plans itself at its first hourly slot.

Several search head tiers. When the deployment has more than one search head over the same indexing layer — on Splunk Cloud the ad-hoc search head cluster and an Enterprise Security cluster, on Splunk Enterprise any search head with a TrackMe remote account — tick every tier in the *Configure UAM* step. Each remote account is tested as soon as it is selected and the creation waits until every selected account is reachable; the inventory then runs once per tier while the audit activity, the resource usage, the capacity and the account signals are collected once through the route account.


Create a new TrackMe Virtual Tenant: User Activity Monitoring
Basics
Configure UAM
Indexes & RBAC
Review
Select the search head tiers to inventory. Activity is collected through their shared indexing layer; use the advanced scope below to restrict its origins.
Local search head
route
The search head TrackMe runs on
checked
Local deployment selected
Local scheduled owner: nobody. User and role catalogs use TrackMe's authorized local identity reader; list_all_users/list_all_roles are not required on this owner.
Recheck local UAM access
hec_hf1
not selected
TrackMe remote account · additional search head tier
hec_hf2
not selected
TrackMe remote account · additional search head tier
hec_hf3
not selected
TrackMe remote account · additional search head tier
hec_hf4
not selected
TrackMe remote account · additional search head tier
prd5_cm
TrackMe remote account · additional search head tier
reachable
remote search connectivity check was successful, service was established
UAM listing capabilities and search head identities verified. Index permissions and catalog visibility still determine collection coverage.
Recheck UAM access
prd5_ds
not selected
TrackMe remote account · additional search head tier
prd5_ds2
not selected
TrackMe remote account · additional search head tier
UAM access is checked for every selected tier. Local checks use the owner chosen under Indexes & RBAC and run again when it changes. Permission warnings do not block creation; remote accounts must also be reachable.
ROUTE ACCOUNT
Local search head
The tier the indexing-layer searches (activity, resource usage, capacity) go through; derived from the selection: the local search head when it is selected, else the first remote tier.
ONE INDEXING LAYER PER TENANT
A UAM tenant monitors one indexing layer through one or more search head tiers. Each tier is one account: the local search head TrackMe runs on, or a configured TrackMe remote account. On Splunk Cloud, for example, TrackMe on the ad-hoc search head is the local tier and a remote account adds the Enterprise Security search head.
Saved searches, users and roles are inventoried once per tier. Audit, introspection and internal data of every tier land in the same indexing layer, so activity, resource usage and capacity are collected once per tenant through the route account.
Every tier selected for a tenant must share the same indexing layer — that is the administrator's responsibility; a tier on another indexing layer belongs to another tenant.
HISTORICAL BACKFILL
Range of history to collect
30 days
Preset: no backfill. Changeable later in Settings » Collection limits.
Collects the audit and resource evidence of the recent past through the route account, bucket by bucket, hourly, until the range is covered; the report then retires itself. Speed depends on the deployment. Introspection retention is typically 14 days, so resource history may end earlier than audit history.
Activity scope (advanced)
Back
Next


Create a new TrackMe Virtual Tenant: User Activity Monitoring
Basics
Configure UAM
Indexes & RBAC
Review
Select the search head tiers to inventory. Activity is collected through their shared indexing layer; use the advanced scope below to restrict its origins.
Local search head
route
The search head TrackMe runs on
checked
Local deployment selected
Local scheduled owner: nobody. User and role catalogs use TrackMe's authorized local identity reader; list_all_users/list_all_roles are not required on this owner.
Recheck local UAM access
hec_hf1
not selected
TrackMe remote account - additional search head tier
hec_hf2
not selected
TrackMe remote account - additional search head tier
hec_hf3
not selected
TrackMe remote account - additional search head tier
hec_hf4
not selected
TrackMe remote account - additional search head tier
prd5_cm
not selected
TrackMe remote account - additional search head tier
prd5_ds
not selected
TrackMe remote account - additional search head tier
prd5_ds2
not selected
TrackMe remote account - additional search head tier
UAM access is checked for every selected tier. Local checks use the owner chosen under Indexes & RBAC and run again when it changes. Permission warnings do not block creation; remote accounts must also be reachable.
ROUTE ACCOUNT
Local search head
The tier the indexing-layer searches (activity, resource usage, capacity) go through; derived from the selection: the local search head when it is selected, else the first remote tier.
ONE INDEXING LAYER PER TENANT
A UAM tenant monitors one indexing layer through one or more search head tiers. Each tier is one account: the local search head TrackMe runs on, or a configured TrackMe remote account. On Splunk Cloud, for example, TrackMe on the ad-hoc search head is the local tier and a remote account adds the Enterprise Security search head.
Saved searches, users and roles are inventoried once per tier. Audit, introspection and internal data of every tier land in the same indexing layer, so activity, resource usage and capacity are collected once per tenant through the route account.
Every tier selected for a tenant must share the same indexing layer — that is the administrator's responsibility; a tier on another indexing layer belongs to another tenant.
HISTORICAL BACKFILL
Range of history to collect
30 days
Preset: no backfill. Changeable later in Settings → Collection limits.
Collects the audit and resource evidence of the recent past through the route account, bucket by bucket, hourly, until the range is covered; the report then retires itself. Speed depends on the deployment. Introspection retention is typically 14 days, so resource history may end earlier than audit history.
Activity scope (advanced)
Back
Next


Create a new TrackMe Virtual Tenant: User Activity Monitoring
Basics
Configure UAM
Indexes & RBAC
Review
Choose indexes for TrackMe objects and assign ownership and roles.
Summary Index
trackme_summary
Audit Index
trackme_audit
Metric Index
trackme_metrics
Notable Index
trackme_notable
Splunk owner for created objects
svc-trackme
Admin roles
trackme_admin
Power roles
trackme_power
Readonly roles
trackme_user
Back
Next





Through the REST API, the same creation is `POST /trackme/v2/vtenants/admin/add_tenant` with `tenant_uam_enabled=true` and no other component flag, plus an optional `uam_config` object holding any subset of the *tenant options* — the wizard sends `tiers`, `scheduled` and `backfill_days`:

```
| trackme url="/services/trackme/v2/vtenants/admin/add_tenant" mode="post"
    body="{'tenant_name': 'user-activity', 'tenant_desc': 'User Activity Monitoring
↪',
          'tenant_roles_admin': 'trackme_admin', 'tenant_roles_user':
↪'trackme_user',
          'tenant_owner': 'svc-trackme', 'tenant_idx_settings': 'global',
          'tenant_uam_enabled': 'true',
          'uam_config': {'tiers': ['local'], 'scheduled': true, 'backfill_days':␣
↪30}}"
```

Tip

One UAM tenant per **indexing layer**: add every search head tier of that layer to the same tenant, so a user present on several tiers is one account and the activity is collected once. Monitor another deployment with another UAM tenant.

See also

- *User Activity Monitoring* — what UAM tracks, the workspace and the policies.
- *User Activity Monitoring — reference* — the tiers, the trackers, the evidence and every `uam_config` option.
- *Who is doing what on your Splunk? User Activity Monitoring (UAM)* — a UAM tenant from its creation to its first decisions, on a live Splunk Cloud deployment.

Creating a Cribl Stream or Cribl Edge tenant

The **Cribl Stream** and **Cribl Edge** cards create a dedicated Cribl tenant: no entity component can be added to it, and its objects are Cribl objects monitored through findings. Both need a *Cribl connection* declared beforehand, and an Enterprise, Unlimited, trial or Developer license.

Each wizard has four steps — the identity of the tenant, the connection with its Worker Groups (Cribl Stream) or Fleets (Cribl Edge), the indexes and roles, and a review — and is described step by step in its guide:

- *Cribl Stream* — create a Cribl Stream tenant, then explore its workspace.
- *Cribl Edge* — create a Cribl Edge tenant, then explore its workspace.
- *Cribl Stream & Cribl Edge* — which Cribl tenant to use, and the Cribl connections.

Creating a Splunk HEC tenant

The **Splunk HEC** card creates a dedicated tenant that monitors the HTTP Event Collector endpoints of one Splunk deployment: no entity component can be added to it, and its endpoints, pools and tokens are monitored through findings. It needs no connection: its source is this search tier or a TrackMe remote account to the search tier that sees the endpoints' internal indexes, and an Enterprise, Unlimited, trial or Developer license. The wizard has four steps — the identity of the tenant, the source with the discovered endpoints and their pools, the indexes and roles, and a review — described step by step in *Splunk HEC*.

Creating a Splunk S2S, Splunk UF Fleet or Splunk Forwarder Collectors tenant

The **Splunk S2S**, **Splunk UF Fleet** and **Splunk Forwarder Collectors** cards create a dedicated tenant that follows the forwarding of one Splunk deployment: the receivers, intermediate forwarders and their pools (Splunk S2S), the forwarding agents and their Deployment Servers (Splunk UF Fleet), or the forwarders used as collection points and the inputs they run (Splunk Forwarder Collectors). No entity component can be added to them, and what they follow is monitored through findings.

Like Splunk HEC, they need no connection: their source is this search tier or a TrackMe remote account to the search tier that sees the forwarders' internal indexes. They need an Enterprise, Unlimited, trial or Developer license, and a TrackMe administrator to create them. Each wizard starts with the identity of the tenant and its source, shows what it discovers before anything is created, then the indexes and roles and a review; each is described step by step in its guide:

- *Splunk S2S* — create a Splunk S2S tenant, then explore its workspace.
- *Splunk UF Fleet* — create a Splunk UF Fleet tenant, then explore its workspace.
- *Splunk Forwarder Collectors* — create a Splunk Forwarder Collectors tenant, declare its syslog collectors, then explore its workspace.

Creating a splk-wlk tenant

The Workload wizard walks through:

- **Deployment type** — for example Splunk Cloud, which creates a tracker for SVC consumption summary metrics.
- **Deployment target** — local or a remote account (searches switch transparently to the remote deployment).
- **Root search constraint** — optional filters for the introspection, scheduler, and SVC searches (by app, or by `host` / `splunk_server`).
- **ML Outliers** — auto-create and train ML models when entities are discovered (trains against the `elapsed` metric by default).
- **Inactive entities** — auto-purge scheduled entities that have been inactive beyond a configured period; this is handled by the Workload tracker named `inactive_entities`.

Tip

Multiple search head tiers: restrict the root constraint so it targets only the intended members (for example `host=sh1 OR host=sh2`). The simplest, most robust pattern is **one tenant per search head tier**.

Creating splk-flx, splk-fqm, splk-cim tenants

These wizards are minimal — you provide the identity, description, and the access-control step. The actual entity configuration happens later, inside the tenant UI, once it exists.

Access control, ownership & indexes (all tenant types)

Every creation path finishes with the same step:

- **Roles (RBAC)** — choose which Splunk roles administer the tenant (admin / power) and which have read-only access (user). See *Roles & access control* for the full model.
- **Owner** — the Splunk user that owns the tenant's knowledge objects and runs its trackers. Future objects are auto-assigned to this owner.
- **Indexes** — select the (already-existing) Splunk indexes the tenant will use.

Tip

You can preset the default owner and roles that the creation UI proposes, under **Configuration → System settings → General → Knowledge objects & RBAC defaults**, so every new tenant starts from sensible defaults.

What happens when you target a remote deployment

Choosing a remote target triggers an automatic **connectivity and authentication test**, and the result is shown before you can continue. Creation also checks that your license edition allows another tenant and that you are permitted to use the chosen remote account.

Automating with the REST API

Everything the wizard does is also available through the REST API — useful for scripting tenant creation or managing it as code. Browse the full endpoint list and options under **API & Tooling → TrackMe REST API Reference** in the UI. Tenants are created by posting to `/services/trackme/v2/vtenants/admin/add_tenant`; for example, a splk-dsm tenant:

```
| trackme url="/services/trackme/v2/vtenants/admin/add_tenant" mode="post"
    body="{'tenant_name': 'mytenant', 'tenant_desc': 'Demo tenant',
           'tenant_roles_admin': 'trackme_admin', 'tenant_roles_user':
```

(continues on next page)

(continued from previous page)

```
↪'trackme_user',
          'tenant_owner': 'admin', 'tenant_idx_settings': 'global',
          'tenant_dsm_enabled': 'true'}"
```

The body carries the same choices as the wizard — identity, RBAC roles, owner, indexes — and a per-component **enable flag**: `tenant_dsm_enabled` / `tenant_dhm_enabled` / `tenant_mhm_enabled` for feeds, `tenant_vol_enabled` for Volume Outliers (its options — `vol_account`, `vol_backfill_days`, `vol_default_direction`, the inactivity policy… — can be set in the same body, see *Volume Outliers — in depth*), and `tenant_flx_enabled` / `tenant_fqm_enabled` / `tenant_wlk_enabled` for the other tenant types. Set the flag(s) for the component(s) you want; the minimal tenant types (flx, fqm, cim) need only the identity, RBAC, and that flag. A **UAM** tenant is created with `tenant_uam_enabled` alone — it is a dedicated tenant type and cannot be mixed with a component flag — and an optional `uam_config` object (see *Creating a splk-uam tenant*). A **Cribl Stream** or **Cribl Edge** tenant is created the same way with `tenant_cribl_stream_enabled` or `tenant_cribl_edge_enabled` alone and a `cribl_config` object, by a TrackMe administrator (the `trackmeadminoperations` capability, see *Creating a Cribl Stream or Cribl Edge tenant*). A **Splunk HEC** tenant is created with `tenant_splunk_hec_enabled` alone and a `hec_config` object, by a TrackMe administrator too (see *Creating a Splunk HEC tenant*). **Splunk S2S**, **Splunk UF Fleet** and **Splunk Forwarder Collectors** tenants follow the same rule with `tenant_splunk_s2s_enabled` and `s2s_config`, `tenant_splunk_uf_enabled` and `uf_config`, or `tenant_splunk_collector_enabled` and `fwc_config` (see *Creating a Splunk S2S, Splunk UF Fleet or Splunk Forwarder Collectors tenant*).

See also

- *Managing tenants* — operate the tenant after creation.
- *Integrations & Remote Deployments* — remote Splunk deployments.
- *Monitoring Components* — what each component tracks.
- *Volume outliers on the Splunk license usage with Volume Outliers (VOL)* — a Volume Outliers tenant, from creation to the first alert.
- *Who is doing what on your Splunk? User Activity Monitoring (UAM)* — a User Activity Monitoring tenant, from creation to the first decisions.

8.2.3 Managing tenants

Once a tenant exists, everything you do to it is on its **three-dot menu** in the Virtual Tenants grid. This page covers that menu — configuration, access, enabling and disabling, deletion — and the two things you watch to keep a tenant healthy: its **operational status** and its **scheduling**.





The tenant menu

Everything you can do to a tenant is on its **three-dot menu** on the tenant card. The menu is grouped into *Access*, *Search*, and *Administration* actions, plus a *Personal preferences* entry to hide the tenant from your own grid; the administration actions are available to tenant admins.

Menu action	What it does
Open the Virtual Tenant	Opens the tenant's home page (you can also double-click the card).
Open SLA overview	Opens the SLA dashboard — the percentage of time each entity spent green.
Search summary / audit / notable events, Search metrics	Deep links into Splunk search, pre-scoped to the tenant's indexes.
View knowledge objects	Lists every knowledge object the tenant owns, with its type — useful for auditing.
Manage components	Add or remove monitoring components on the tenant.
Configure tenant	The per-tenant settings editor (see *Configuring a tenant*).
RBAC & ownership	Change the admin / power / user roles and the owner; existing knowledge objects are re-permissioned and, if the owner changes, re-assigned. See *Roles & access control*.
Restrict access by user	Limit the tenant to a named list of users, on top of role-based access; the card then shows a *Restricted* badge.
Clear operational status	Reset a stuck operational status once you have fixed the cause (see *Operational status*).
Disable / Enable the tenant	Pause or resume the tenant without deleting anything (see below).
Delete the tenant	Permanently destroy the tenant (see below).
Hide / Unhide this tenant	A personal preference that removes the tenant from *your* grid only; other users are not affected and the tenant is neither disabled nor restricted. Bring hidden tenants back with **Unhide this tenant** or with **Virtual Tenants visibility** in the Settings panel (see *Personal preferences*).

Warning

Removing a component (via *Manage components*) permanently deletes that component's knowledge objects and purges its data. The only way back is a restore from backup. Prefer disabling the tenant if you are unsure.

Configuring a tenant

Configure tenant opens a tabbed editor for everything about a tenant that is not its identity or its components. From here you set, per tenant: documentation and access control, entity discovery and labels, the **Tables & views** defaults of the Tenant Home (pagination, default time range, tab order, table group-by — see *Personal preferences*), feed delays, the delayed-entities inspector and auto-disablement, ML outliers, the *disruption queue* and *impact scoring*, *CMDB integration*, *AI settings* and bank holidays. Most of these features are explained in their own parts of this guide — this modal is where you tune them for one tenant.

Enabling and disabling

Disabling a tenant is the safe, reversible alternative to deleting it:

- **Disable the tenant** sets the status to `disabled`: all the tenant's knowledge objects are disabled in Splunk, no trackers run, no entities are maintained — but **no data is destroyed**. A disabled tenant stays on the Virtual Tenants grid with a **disabled** badge, and its entities no longer count in the page's alert totals.
- **Enable the tenant** brings it straight back.

Deleting a tenant

Warning

Deletion is permanent and irreversible. It drops the tenant's data collections, removes its configuration and its role triplet, deletes its scheduled trackers, and purges its Configuration Guardian alerts. Previously generated events and metrics are left untouched in your indexes. Hybrid trackers must be re-created if you ever rebuild the tenant. The only recovery path is a restore from backup — when in doubt, **disable instead of delete**.

Operational status

TrackMe continuously monitors itself. Every tracker and custom command reports failures to a central register, and a per-tenant **health tracker** (running every five minutes) turns that into health events you can see directly on the tenant card.

Common causes of a failing operational status:

- An overloaded environment with no available search slots.
- A search head in detention.
- Network or credential issues (especially toward a remote deployment — see *Integrations & Remote Deployments*).
- Corrupted configuration: missing knowledge objects or incorrect permissions.

To investigate:

- Click the operational-status chart on a tenant to see the affected tenants, related accounts, and the root cause; **Open in search** drills into the per-tracker results.
- Health events land in the tenant's audit index, for example `index=trackme_audit sourcetype="trackme:health" last_status="failure"`.
- Once you have fixed the cause, use **Clear operational status** from the tenant menu to reset the recorded status. It defaults to the current tenant but you can widen the scope to others — or all tenants — and optionally **re-run the health-check tracker** as it clears, with a note for the audit trail.
- The **Audit → Operational Statuses** dashboard gives the cross-tenant view.
- The per-tenant execution summary is stored centrally; inspect it with:

```
| inputlookup trackme_virtual_tenants | fields tenant_id,␣
↪tenant_objects_exec_summary
```

Scheduling and tracker health

TrackMe depends on its trackers being scheduled and running on time. You monitor that from the Virtual Tenants UI:

- **Aim for zero skipped searches.** Skipped trackers mean the deployment is short on search slots or CPU — review search performance or add capacity.
- Trackers log their own run-time KPIs; the **TrackMe Performance Deepdive** dashboard surfaces them.
- **Automatic re-attempts:** when maximum search concurrency is hit, TrackMe retries (logged as a temporary failure). If the retry limit is reached it logs a permanent failure and cancels that run.

Tip

`DEBUG` logging is very verbose — enable it only while troubleshooting, from **Configuration → System settings → Logging**, and turn it back down afterwards.

Under the hood and via the API

The tenant lifecycle is fully scriptable through the `vtenants/admin` endpoints (all `mode=post` unless noted): `update_tenant_rbac`, `enable_tenant`, `disable_tenant` and `del_tenant` (`mode=delete`). Operational status is produced by a per-tenant health tracker, `trackme_health_tracker_tenant_<tenant_id>` (every ~5 min, running `| trackmetrackerhealth tenant_id="<tenant_id>"`); read it with the `configuration/get_tenant_ops_status` endpoint and clear it with `vtenants/admin/clear_exec_summary`. The trackers themselves are orchestrated by the `trackmetrackerexecutor` command — inspect scheduling at `index=_internal sourcetype=trackme:custom_commands:trackmetrackerexecutor`.

See also

- *Roles & access control* — roles and ownership.
- *Operations & Administration* — backup & restore, maintenance, and scaling.
- *Personal preferences* — hidden tenants and the other personal preferences.

8.2.4 Virtual Groups

A **virtual group** is a read-only card, shown alongside your real tenants in the Virtual Tenants grid, that aggregates entities from **several tenants** into one view. Use it when you want a single pane that spans tenant boundaries — "all of finance across three regions", or "every high-priority entity regardless of tenant".





Important

A virtual group is **not a tenant**. It has no configuration stanza, no roles of its own, no trackers, and no entity ownership. It is **not seen by the decision maker** and **cannot open alerts**. It is a visibility layer only. If you need monitoring, ownership, or alerting, create a real tenant instead.

What a group shows

The card displays an aggregated entity count and a priority breakdown across all the tenants in scope. A **details** button opens a group overview. Summaries are **always computed live** from the underlying tenants' current state — nothing is cached — so a group is always as fresh as the tenants it aggregates.

Creating a virtual group

The creation wizard asks for:

1. **Tenants and components in scope** — which tenants, and which of their components, to aggregate.
2. **Optional priority filter** — restrict the group to (say) high and critical entities; leave empty for all.
3. **Allowed roles** — which Splunk roles may see the group.
4. **Alias and description** — the display name and purpose.
5. **Review and create.**

You also set a `group_id` (lowercase, up to 40 characters) and a display `group_alias`.

Behaviour and limitations

- **Graceful degradation.** Deleted tenants, disabled tenants, and disabled components in scope are silently skipped — the group simply reports fewer entities, never an error.
- **Empty groups are not auto-hidden.** A group whose scope yields nothing shows zero counts; delete it manually if you no longer want it.
- **Visibility follows roles.** A user who belongs to none of the group's allowed roles simply does not see it.
- **No SLA rollup and no group-level acknowledgment** today — entities are still acknowledged in their own tenant.

See also

- *Virtual Tenants explained* — real tenants, which groups aggregate.
- *Classification & Protection* — the priority levels used by the group filter.

8.2.5 Roles & access control

Start with the identity doing the work. The person using TrackMe, the owner of a scheduled report, and the account reading a remote system can be three different users. Giving one of them access does not grant it to the others.

This guide explains those boundaries for all tenants, with a setup map for the next-generation components: **UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge**. Use *Roles & access control — in depth* for the capability lists, index permissions, examples and checks.

Four identities: the TrackMe user (capability and tenant role), the scheduled owner (owner permissions), the source reader (search and index access), and the additional API account (destination permissions). A remote Splunk account needs both local permission to use it and destination permission to read.

Four questions before granting access

Table 1: Put each permission on the account that uses it

Question	Identity	Where to grant access
Who may use this tenant?	The person signed in to TrackMe	TrackMe capabilities, the tenant's role lists and access to its TrackMe indexes
Who runs its schedules?	The **tenant owner**: `nobody` or a named Splunk service account	On the search tier hosting TrackMe; see *The scheduled owner and service account*
Who searches the source data?	Locally: the scheduled execution identity. Remotely: the Splunk user behind the selected remote account's token	On the search tier where the search runs: search capability, source indexes and any component-specific reads
Who reads the additional API?	A separate Splunk remote-account user, or a **Cribl connection** credential	On the endpoint being read: HEC endpoint, cluster manager, Deployment Server or Cribl Leader

A remote account has **two access checks**. On TrackMe, its role list decides who may use it. On the destination, the token user's capabilities and index permissions decide what it can read. Both must work for scheduled collection; testing the account as a Splunk administrator does not prove that the tenant owner may use it.

The three TrackMe user levels

TrackMe ships three roles. You can assign them directly or inherit them into your own roles; `trackme_admin` inherits `trackme_power`, which inherits `trackme_user`.

Table 2: Day-to-day access to a tenant

Built-in role	Capability	What it is for
`trackme_user`	`trackmeuseroperations`	Read the tenant and its monitoring results.
`trackme_power`	`trackmepoweroperations`	Operate existing monitoring. In findings workspaces: triage findings, assign them, add notes and perform the actions the component exposes.
`trackme_admin`	`trackmeadminoperations`	Create and administer TrackMe content. For a tenant the user administers: source settings, policies, collection and automation settings.

The capability grants an **operation level**; the tenant's **user, power and admin role lists** grant the **scope**. A TrackMe administrator of one tenant does not thereby administer every tenant. Splunk administrators retain administrative access.

These are the permissions of **TrackMe users**. They do not automatically provide a remote Splunk account with cluster-read capabilities, allow a search of internal indexes, or satisfy the requirements of a named scheduled owner.

Important

UAM has a narrower workspace

UAM contains audit and account activity. Its workspace is for **tenant administrators** (`trackmeadminoperations` plus a tenant admin role), or a Splunk administrator with `admin_all_objects`. Tenant power users have selected REST read and action access; read-only tenant users have no UAM access. See *UAM: audit access and per-tier inventories*.

Choosing the scheduled owner

A dedicated service account makes scheduled activity attributable to a named identity. It is optional for ordinary collection: `nobody` is also a supported knowledge-object owner, with execution supplied by Splunk. **Scheduled AI Agents require a named account**; `nobody` and `splunk-system-user` cannot run them.

For **HEC, S2S, UF Fleet, Forwarder Collectors, Cribl Stream and Cribl Edge**, the current creation and collection checks require a named owner to hold the three TrackMe capabilities, `schedule_search`, `list_settings`, `list_storage_passwords` and `admin_all_objects` (or be a member of `admin` / `sc_admin`). Search and index access must also be configured. These broader permissions belong to the scheduled owner, not to every person using its workspace.

UAM's collection requirements are checked separately. Its identity and audit reads have different local and remote requirements. The general Configuration Guardian owner check is broader than the UAM collection check; see *The scheduled owner and service account* and *UAM: audit access and per-tier inventories* before treating them as the same requirement.

What each next-generation component reads

Table 3: Choose the source and the additional account independently

Component	Data searched in Splunk	Additional API access
UAM	`_audit`, `_introspection` and `_internal` through the chosen route account; inventories collected separately from each search head tier	User, role, knowledge-object and, for a cluster, SHC member reads on each tier
Splunk HEC	`_internal` and `_introspection` on the local or remote source search tier	Optional accounts on the HEC endpoints' **splunkd management API** for configuration comparisons
Splunk S2S	`_internal`; `_metrics` when used; `_introspection` for resources. Local or remote	`list_indexer_cluster` on the source reader for cluster discovery; optional **cluster manager account** for the manager's view
Splunk UF Fleet	Forwarding telemetry on its source tier; **separate Deployment Server source** for `_dsphonehome`, `_dsclient` and `_dsappevent`	Optional **Deployment Server API accounts** for current server-class membership; these accounts also need the three `_ds*` indexes on the Deployment Server
Splunk Forwarder Collectors	`_internal`, `_introspection` and `_configtracker` on the local or remote source search tier	Optional **collector API accounts**, one per collector, for the current list of its inputs: `search`, `rest_properties_get` and `list_inputs`
Cribl Stream	Cribl internal metrics in a metric index searchable from the **local TrackMe tier**; a remote metric-source account is not supported	Required **Cribl connection** with read access to Stream inventory, configuration and Leader-proxied Worker status
Cribl Edge	No Splunk source index required for core API monitoring; optional local System Metrics index for node CPU and memory	Required **Cribl connection** with read access to Edge Fleets, Nodes, configuration and Leader-proxied status

`local` describes **where a search is dispatched**, not where events are stored. A local TrackMe search head can search distributed indexers. Conversely, a remote Deployment Server account can search indexes held on that server

itself.

Tip

Internal indexes need an explicit grant

Access to all ordinary indexes (`*`) does not include internal indexes. Grant the source reader the actual indexes it needs, especially UF Fleet's three `_ds*` indexes. The detailed reference includes *index-access examples*.

Grant tenant access without exposing every tenant

1. Define your user, power and admin roles in Splunk, with the appropriate TrackMe capabilities and application access. SSO group mappings can assign these roles.
2. Put those roles in the matching **RBAC lists of each tenant**.
3. Give users access to that tenant's TrackMe result indexes. Grant source indexes and remote-account use separately when their investigations need them.
4. Configure the owner and each source/API account using the component requirements above.
5. Check a scheduled collection, not only a manual run. The two may execute under different identities. Follow *Validate each identity and diagnose the missing layer* for a repeatable check.

Warning

Role inheritance can widen tenant scope

A new tenant's default lists include the built-in TrackMe roles. If your custom role inherits one of them and it remains listed on other tenants, the inherited role can grant access there too. Replace the default tenant lists with your scoped roles, or grant capabilities directly to your custom roles. Include the lower-level capabilities needed by each higher-level role when using direct grants.

System configuration and tenant administration

Configuration → System settings and the dedicated account-management pages are Splunk administrator tasks. They are separate from managing a tenant you administer. Cribl credentials are managed under **Configuration → Manage Cribl connections**; Splunk remote-account credentials under **Configuration → Manage Remote deployment accounts**.

Set defaults for **new tenants** under **System settings → General → Knowledge objects & RBAC defaults**. To change an **existing tenant's** roles or owner, use its Virtual Tenant UI; TrackMe updates the tenant and its knowledge-object permissions. Changing defaults alone does not re-permission existing tenants.

See also

- *Roles & access control — in depth* — exact requirements, component details, account examples and diagnosis.
- *Creating a tenant* and *Managing tenants* — tenant lifecycle.
- *Configuration* — native configuration interfaces.
- *Remote Splunk deployments* and *Cribl Stream & Cribl Edge* — connection setup.

8.2.6 Roles & access control — in depth

Use this reference to provision the **right identity on the right system**. It covers the TrackMe user, the scheduled owner, the Splunk source reader and any additional API account, with separate requirements for UAM, Splunk HEC,

Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge. For the shorter setup guide, start with *Roles & access control*.

- *UAM*
- *Splunk HEC*
- *Splunk S2S*
- *Splunk UF Fleet*
- *Splunk Forwarder Collectors*
- *Cribl Stream*
- *Cribl Edge*

How TrackMe RBAC works

TrackMe combines **capabilities**, **tenant scope** and **Splunk object/index permissions**:

- A capability authorizes an operation. The user's effective capabilities include grants inherited through Splunk roles.
- The tenant's admin, power and user role lists determine which tenants the user can administer, operate or read. Higher-level access includes the lower-level operations.
- Splunk ACLs protect the tenant's knowledge objects. Index permissions govern searches of its results and of source data. A capability does not replace index access.
- TrackMe authorizes a caller before performing privileged work on their behalf. A successful UI operation does not mean the caller's own search session, a scheduled owner, or a remote token can perform every underlying read directly.

The built-in roles

Table 4: User permissions on the TrackMe search tier

Role	Capability	Tenant operations
`trackme_user`	`trackmeuseroperations`	Read the tenant's monitoring results, subject to the component's access rules.
`trackme_power`	`trackmepoweroperations`	Operate existing monitoring. In HEC, S2S, UF Fleet, Forwarder Collectors, Stream and Edge: triage findings and Finding groups, assign, annotate and use permitted actions.
`trackme_admin`	`trackmeadminoperations`	Create tenants and trackers; administer source settings, policies, collection and automation for tenants in the user's admin scope.

`trackme_power` inherits `trackme_user`; `trackme_admin` inherits `trackme_power`. Membership and inheritance both work. If you grant capabilities directly instead, include all the levels the role needs. On the six findings workspaces above, read access accepts any tenant role, triage requires a power or admin role, and configuration requires an admin role, in addition to the matching capability. `admin` / `sc_admin` and Splunk's internal execution context have administrative access.

UAM is different: its workspace requires tenant-administrator access. Its limited power access is through selected REST operations; a tenant user has no UAM access. See *UAM: audit access and per-tier inventories* for the complete distinction.

The scheduled owner and service account

A tenant's **owner** owns its reports and scheduled monitoring. It is set at creation and can be changed through the tenant's RBAC/ownership controls. A service account is simply a named Splunk user chosen for that role; it is not the person currently viewing the tenant, nor automatically the user behind any remote token.

`nobody` and a named owner

- ``**nobody**`` is supported for ordinary collection. It is a knowledge-object ownership value: Splunk supplies the scheduled execution identity. It is not a remote username or a credential to create on another system. Source/API authorization still applies.

- **A named owner** must be able to run the scheduled work and access its inputs. Configure its permissions on the TrackMe search tier, then separately configure every remote user and Cribl credential it uses.
- **Scheduled AI Agents require a named owner.** `nobody` and `splunk-system-user` are rejected for scheduled agent execution. The named owner also needs the agent's tenant access and access to the records the agent uses; changing the owner is not itself proof that an agent run will succeed. `splunk-system-user` is not a selectable replacement for a named owner in the HEC/S2S/UF Fleet/Forwarder Collectors/Cribl creation flows.

Current owner capability checks

For **Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge**, the current creation and collection checks accept `nobody`, a member of `admin` / `sc_admin`, or an account holding **all seven** of these capabilities:

Table 5: Named-owner requirement on the TrackMe tier

Capability	Scope
`trackmeuseroperations`	TrackMe user operations
`trackmepoweroperations`	TrackMe power operations
`trackmeadminoperations`	TrackMe administration operations
`schedule_search`	Scheduled searches
`list_settings`	Splunk settings reads
`list_storage_passwords`	Splunk credential-store reads
`admin_all_objects`	Splunk knowledge-object administration

This is an **enforced owner requirement**, not a claim that every individual collection read needs all seven. They may be inherited or granted directly. Add `search` and the source/result index permissions needed by the work; the owner check alone does not verify search coverage. A custom owner role can inherit `trackme_admin` and `power` and add the remaining required capabilities, but it is still a privileged role.

The general Configuration Guardian check ``**insufficient_tenant_owner_capabilities**`` uses this same seven-capability list for named owners of enabled tenants, including UAM. **UAM's own collection preflight is narrower**: it checks the identities and capabilities in *UAM: audit access and per-tier inventories* and does not require `list_settings` or `list_storage_passwords` for those reads. A successful UAM collection check therefore does not necessarily clear the general owner warning. Diagnose the named check rather than combining the two lists into a requirement for every remote UAM account.

Important

Keep owner permissions separate from user and remote-account permissions

Do not copy this seven-capability set onto every TrackMe user, or onto a cluster manager or Deployment Server reader. The first needs the relevant tenant role; the latter need their destination-specific reads. Their requirements are listed below.

Local searches, remote searches and API accounts

A **search tier** is the search head or search head cluster that dispatches searches. It may search distributed indexers; choosing `local` does not mean the data must be stored on the TrackMe search head.

- Local search: tenant report → local execution identity → searchable source indexes.
- Remote search: tenant owner may use the remote account → destination token user → source indexes and REST reads on the remote tier.
- Additional API: component collection → selected API credential → endpoint or Leader.

Interactive fetches may use the caller. Validate the schedule separately.

Table 6: Which identity must have the permission?

Read	Acting identity	Access to check
Scheduled local search	Named tenant owner, or Splunk's execution context for `nobody`	Search and index permissions on the TrackMe search tier
Scheduled remote search	The owner starts the work; the destination search runs as the **remote token user**	The owner's effective roles may use the TrackMe remote account; the token user may search the destination's indexes and perform the required REST reads
Additional Splunk API read	Token user of the selected HEC, cluster manager or Deployment Server account	That system's endpoint permissions; these can differ from the telemetry source
Cribl API read	Cribl.Cloud API client or self-managed Cribl user in the **Cribl connection**	Cribl product, group/Fleet and API read permissions
Interactive search or live fetch	The caller, or the remote token selected for that action	The caller's source/result index access and permission to use the remote account when applicable; do not infer these from a successful schedule

Remote-account RBAC is configured on TrackMe. Its `rbac_roles` allows local roles to use the stored account. Inherited roles count. For HEC, S2S, UF Fleet and Forwarder Collectors, creation/settings check that the selected owner may use the source and enrichment accounts; an explicit refusal is reported as `remote_account_refused`. UAM also checks account use for its collection and live reads. Internal scheduler handling does not grant additional capabilities to the destination token.

Destination permissions are configured on the destination. A token authenticates a Splunk user on that system. Grant that user the required capabilities and index access there. Giving the local owner `list_indexer_cluster` cannot repair a remote source token that lacks it. TrackMe need not be installed on a remote telemetry tier merely to run Splunk searches; UAM's remote administrative check has a specific consequence when the TrackMe capability is absent (see below).

The account's endpoint matters. The telemetry source usually points to a search head. A cluster manager account must point to the manager; a Deployment Server API account to the Deployment Server; HEC enrichment to the endpoint's splunkd management API. They are independent account selections, even when some use the same user or system.

Source indexes and TrackMe result indexes

Grant index access to the identity that **actually searches on that tier**. The source index names below are defaults; use the configured names when they differ.

Table 7: Source data required by each component

Component	Indexes	Reader / condition
UAM	`_audit`, `_introspection`, `_internal`	The route account that searches audit, resource and indexer activity. Inventory accounts on other tiers have separate REST/object requirements.
Splunk HEC	`_internal`, `_introspection`	Owner for a local source; token user for a remote source
Splunk S2S	`_internal`; `_metrics` when the metrics form is used; `_introspection` for resource observations	Source reader. Keep `_internal` available for logs and qualification even when connection telemetry uses `_metrics`.
Splunk UF Fleet	Telemetry: `_internal`, plus `_metrics` / `_introspection` when used. Deployment Server data: `_dsphonehome`, `_dsclient`, `_dsappevent`	Telemetry reader and Deployment Server source reader can be different accounts on different systems. The optional DS API user also needs the three DS indexes.
Splunk Forwarder Collectors	`_internal`, `_introspection`, `_configtracker` (Splunk 9.0 or later on the collectors)	Owner for a local source; token user for a remote source
Cribl Stream	The configured Cribl internal-metrics index, for example `cribl_metrics`	Local TrackMe execution identity; no remote metric-source account
Cribl Edge	Optional configured System Metrics index	Local TrackMe execution identity, for node CPU and memory. Core API monitoring does not depend on a Splunk source index.

TrackMe result indexes are a second set. The tenant's summary, audit, metric and notable indexes store TrackMe's own results. Give the people investigating the tenant access to the results they need, and ensure the scheduled owner can use the configured stores and searches. These indexes are not the same as Splunk's `_audit` or source `_metrics`. When results are on another deployment, follow the tenant's configured storage/search arrangement as well as its source arrangement.

``*`` does not include internal indexes. In a Splunk role, `srchIndexesAllowed = *` covers ordinary indexes, not names beginning with _. Grant the required internal indexes explicitly; `_*` grants all internal indexes and is broader. Search filters and explicit disallowed indexes can still restrict visibility. See the Splunk authorize.conf reference.

For example, a **remote HEC telemetry reader** can inherit basic search capabilities and be restricted to its two source indexes:

```
[role_trackme_hec_telemetry]
importRoles = user
srchIndexesAllowed = _internal;_introspection
```

This is a **source-reader example**, not the complete role for a named TrackMe tenant owner. Check the user's other inherited roles too: this stanza does not revoke broader access already granted elsewhere. Index permission also cannot make data visible if it was never forwarded to the indexers searched by that tier.

UAM: audit access and per-tier inventories

UAM separates the **route account**, which searches the audit/resource data, from the **inventory accounts**, which read users, roles and knowledge objects on each search head tier. One tier can serve both purposes. Do not assume that a search head holding all the audit data can also enumerate another tier's local users and private objects.

Who may use UAM

Caller	Access
Splunk administrator with `admin_all_objects`	Administrative access across UAM tenants
`trackmeadminoperations` + a tenant admin role	UAM workspace, configuration, collection, policies and decisions
`trackmepoweroperations` + a tenant power role	Selected REST read operations, user communications and findings actions; no UAM workspace
Tenant user	No UAM access

Workspace searches and live **Fetch SPL / Fetch definition** operations can use the caller's own access. The user needs the relevant TrackMe/source indexes and, for a remote fetch, permission to use that remote account. An owner collecting successfully does not grant these rights to the person investigating.

Collection-account requirements

Table 8: Requirements on the tier being read

Capability / access	Local tier: named owner	Remote tier: token user
`search`	Required	Required
`trackmeadminoperations` or `admin_all_objects`	Required; with TrackMe admin access, the owner also needs a tenant admin role	Required. Without TrackMe installed there, `trackmeadminoperations` is not defined: `admin_all_objects` is the available alternative.
`list_all_users` and `list_all_roles`	Not needed for the local identity catalogs: TrackMe's authorized identity reader performs those reads	Required to enumerate all users and roles. `edit_user` / `edit_roles` also satisfy the corresponding checks, but are broader than the read capabilities.
`list_search_head_cluste`	Required for a search head cluster	Required for a search head cluster
Source index access	`_audit`, `_introspection`, `_internal` when this is the route account	The same indexes when this is the route account
Knowledge-object / lookup access	Determines which objects and definitions are visible	Determines which objects and definitions are visible; REST access alone does not guarantee all private objects are visible

Before Splunk 10.4, UAM also reads `admin/file-explorer` for CSV lookup sizes; that read must be allowed on each inventory tier for lookup-footprint coverage, and a least-privilege account is refused it by default. A refused read is shown as **Lookup sizes refused**. From Splunk 10.4 the lookup listing reports the size, with no extra access. Granting `list_all_users` does not solve an unrelated lookup-size or object-ACL refusal.

The UAM collection check does not require `list_settings` or `list_storage_passwords`. It runs at tenant creation and through Configuration Guardian, which reports **UAM collection permissions need attention** with the affected tier. It checks capabilities, not complete data coverage: ``**identity_visibility_limited**`` means the user/role catalog still reflects only the reading account. See *UAM collection accounts* for collection details and *The scheduled owner and service account* for the separate general owner check.

Splunk HEC: telemetry and optional configuration reads

The telemetry reader needs `search` and access to `_internal` and `_introspection` on the selected source tier. For `local`, this is the scheduled execution identity; for a remote source, the remote token user. The owner must be allowed to use the remote account. These searches support monitoring without management access to individual HEC endpoints.

Optional splunkd API accounts add HEC token consistency and the HEC-settings portion of configuration drift. Each account points to the endpoint's **management port**, not the HEC ingestion port, and uses a Splunk authentication token, not an HEC ingestion token. The owner must be allowed to use each of these accounts independently of the source account.

The destination user must be allowed to GET:

- `/services/server/info`;
- `/services/data/inputs/http/http` (global HEC settings);
- `/services/data/inputs/http` (HEC input configuration).

Splunk defines ``**list_token_http**`` for viewing HTTP token-input settings; use that read capability where supported by the target version and check that the account can read all three endpoints. `edit_token_http` is broader: it also allows changes. TrackMe performs GETs only, but that does not turn a broadly privileged token into a read-only credential. The capability definitions are in the Splunk authorize.conf reference. The exact minimum for this three-endpoint combination is not established by TrackMe's capability preflight; verify the reads on the target Splunk version instead of applying the owner capability bundle to the endpoint user.

HEC token **values** are discarded while parsing the response. If an enrichment read is refused, Collection health reports the affected account; telemetry monitoring continues. On **Splunk Cloud**, customer access to the indexers' management API is not used by this enrichment. Monitor Cloud HEC through the source search tier's internal indexes. See *Splunk HEC* for the complete workflow.

Splunk S2S: two levels of indexer-cluster access

Telemetry uses the source-reader permissions in *Source indexes and TrackMe result indexes*, plus the named owner requirements on TrackMe. Cluster discovery adds a requirement on that **source reader**, even when no cluster manager account has been configured.

Table 9: Keep the two cluster identities separate

Read	Account and capabilities	What it adds
Search head's cluster view	Local owner or remote **source token user**: `search`, `rest_properties_get` and `list_indexer_cluster`	`cluster/searchhead/generation` and peer `cluster/peer/info` reads: cluster membership and the view available through the search tier
Optional manager view	**Cluster manager account's token user**, on the manager: `search`, `rest_properties_get`, `list_indexer_cluster`	Manager configuration, maintenance/rolling-restart state and peers from `cluster/config/config`, `cluster/manager/info` and `cluster/manager/peers`

A missing source-reader `list_indexer_cluster` can appear as no rows from `| rest`. TrackMe reports ``**cluster_read_refused**`` rather than treating that as an empty cluster. For a deployment without an indexer cluster, **Indexer clusters as pools → Off** disables that discovery; it is not a fix for a clustered deployment with missing access.

The optional manager account is configured per cluster pool, and its local account RBAC must allow the owner. TrackMe recommends this **read-oriented role on the manager**:

```
[role_trackme_cluster_reader]
importRoles =
search = enabled
rest_properties_get = enabled
list_indexer_cluster = enabled
```

This is the capability set recommended by the product's account check, not a certified minimum for every Splunk release. Check all three manager reads in the wizard or Settings; the result identifies refused reads. `edit_indexer_cluster` is not part of the recommendation: TrackMe observes the cluster and does not change its configuration.

Without a manager account, the search head's cluster view remains available. A failing manager account is reported as ``**cluster_manager_unreadable**``; manager-specific facts are unavailable and maintenance handling relies on the search head's view. It does not remove the source reader's own cluster-read requirement.

Splunk UF Fleet: three independent access paths

UF Fleet can read its forwarding telemetry, Deployment Server event data and Deployment Server API from different systems. Configure all the paths you use, even when they happen to share a credential.

UF Fleet has three paths:

1. Telemetry source: search on the local or remote tier; internal/metrics/introspection indexes used by the configured collection.
2. Deployment Server source: search where the phonehome, client and app-event indexes live.
3. Optional Deployment Server API: search, rest_properties_get, list_deployment_server, and all three DS indexes on the Deployment Server itself.

Access to DS event copies on the indexers does not grant the API user access on the DS.

Table 10: UF Fleet account map

Selection	Required access	Purpose
Source	Local execution identity or remote source token: `search` and forwarding telemetry indexes (`_internal`; `_metrics` / `_introspection` when used)	Receivers' connection records, agent logs and resource observations
Deployment Server source	Local execution identity or its own remote token: `search` and `_dsphonehome`, `_dsclient`, `_dsappevent` (or configured replacements)	Phonehomes, client properties and app-deployment events
Deployment Server API accounts — optional	On each Deployment Server: `search`, `rest_properties_get`, `list_deployment_server` **and access to the three DS indexes**	Read `/services/deployment/server/clients` for current server-class membership

The **Deployment Server source** is a search location, separate from the main telemetry source. When the DS indexes are searchable on the TrackMe tier, it can be local. Otherwise choose a remote account on the tier that holds them. In particular, when the main source is Splunk Cloud and the DS data remains on an on-premises Deployment Server, select a remote account on that Deployment Server for the DS source.

The optional **DS API account** is a separate selection. Adding it does not change where telemetry or DS events are searched. Without it, server-class groups use clients observed in app events over the last 60 days, labelled **App events (60 days)**; this is not a complete current client listing, and its coverage is not judged as such.

Important

An HTTP 200 with no clients can be an index-permission problem

The Deployment Server builds its client listing from searches of `_dsphonehome`, `_dsclient` and `_dsappevent` as the API caller. `list_deployment_server` alone can therefore return an empty listing. Grant the API user access to these indexes **on the Deployment Server**, even if another account already searches copies on your indexers. Splunk documents this behaviour in Deployment clients are not visible for non-admin users.

A dedicated **DS API reader** role can be:

```
[role_trackme_ds_reader]
importRoles =
```

(continues on next page)

(continued from previous page)

```
search = enabled
rest_properties_get = enabled
list_deployment_server = enabled
srchIndexesAllowed = _dsphonehome;_dsclient;_dsappevent
```

This does not need a Splunk administrator role or Deployment Server edit capabilities. If the same account also serves the DS telemetry source, add whatever other indexes that source reads. If it also serves the general telemetry source, add those indexes too.

The wizard and Settings check the selected DS API account's identity, capabilities and client listing. If the listing is empty they also check visible phonehomes. No phonehomes can mean missing index access **or no recent data**; if phonehomes are visible, a clustered Deployment Server may have another member that serves the listing. Check the member and the actual data rather than treating an empty response as proof of a healthy empty fleet.

Expected-agent and group lookup tables introduce another read: the scheduled owner needs the lookup's ACLs locally, or permission to use the chosen remote lookup account, whose user must be able to read the lookup there.

Splunk Forwarder Collectors: the source and optional collector accounts

Telemetry uses the source-reader permissions in *Source indexes and TrackMe result indexes*: the collectors forward their `_internal`, `_introspection` and `_configtracker` to indexers the source search tier can read, which is Splunk's default forwarding of internal indexes. Every policy works from this telemetry alone.

Table 11: Splunk Forwarder Collectors account map

Selection	Required access	Purpose
Source	Local execution identity or remote source token: `search` and `_internal`, `_introspection`, `_configtracker`	Each collector's metrics and logs, its processes and disks, and its configuration changes, which list the inputs it has
Collector API accounts — optional	On each collector: `search`, `rest_properties_get` and `list_inputs`	The current list of the collector's inputs and their state, read from its management API

The optional **collector API account** is a TrackMe remote account on the collector itself. The wizard and Settings check each one as it is added: who the account is, which instance answers, and how many inputs it reads; a missing capability is named with the read-only role that fixes it. A failing account can still be chosen, because no policy depends on it. A Splunk Cloud search tier usually cannot reach a customer's collectors: the tenant then works from the telemetry, and an input appears from its configuration changes or while it delivers.

A dedicated **collector reader** role can be:

```
[role_trackme_collector_reader]
importRoles =
search = enabled
rest_properties_get = enabled
list_inputs = enabled
```

The scheduled owner must be allowed to use the source account and every collector API account the tenant reads, like the other Splunk dedicated tenants.

Cribl Stream: local metrics plus a Cribl connection

A Stream tenant has **two required inputs with different credentials**:

1. **Cribl internal metrics** in a Splunk metric index searchable from the local TrackMe tier. The scheduled execution identity needs `search` and access to that index. The index may be on distributed indexers, but the current Stream metric source is `local`; a TrackMe remote metric-source account is not supported.
2. **A Cribl connection** to Cribl.Cloud or the self-managed Leader. This uses a Cribl API client/user, not a Splunk user or a Splunk remote-deployment account.

The Cribl credential needs read access to the selected **Worker Groups and Workers**, Sources, Destinations, Pipelines, Routes and Packs, Leader system information, and Leader-proxied Worker system information and input/output status. Scope it to the groups being monitored while allowing those reads. TrackMe does not prescribe a universal Cribl role name; permissions depend on the Cribl deployment and its role model.

Both inputs matter: successful Cribl authentication does not grant access to the Splunk metrics index, and a metrics search succeeding does not prove inventory/status API coverage. Configure the named owner's capabilities separately using *The scheduled owner and service account*. See *Cribl Stream* and *What the Cribl account needs* for setup and connection details.

Cribl Edge: API monitoring and optional System Metrics

Edge's core monitoring reads the **Cribl Leader API** through a Cribl connection. The credential needs read access to the selected **Fleets and Edge Nodes**, their Source and Destination configuration, Leader system information, and Leader-proxied node status, system information and system metrics. This is a Cribl permission grant; adding Splunk capabilities to a remote Splunk user does not provide it.

A Splunk source index is **not required for core API monitoring**. For node CPU and memory policies/charts, configure the optional **System Metrics index** and grant the local TrackMe execution identity search access to it. A remote metric-source account is not supported. Without this index, the relevant features report that System Metrics are needed; that does not mean the Cribl connection has failed.

The tenant still has its own TrackMe result stores and scheduled-owner requirements. For both Stream and Edge, a Splunk administrator manages the stored Cribl connection; tenant users operate the workspace through their TrackMe role without being given the Cribl credential. All monitoring reads leave Cribl configuration unchanged. See *Cribl Edge* and *What the Cribl account needs*.

Validate each identity and diagnose the missing layer

1. **Record the account map.** Note the tenant owner, local/remote source, optional API accounts, and the actual user behind each remote token. For UF Fleet, record the Deployment Server source separately. For UAM, record the route and every inventory tier.
2. **Check local entitlement.** Verify the operator's TrackMe capability and tenant role, and the scheduled owner's required capabilities. Verify permission to use each selected remote account for the owner and for users making live remote reads.
3. **Check the destination identity.** Test connectivity in the native account-management UI, then run the component's account/collection checks. A valid token is not proof of cluster access, a complete user catalog, or access to the source indexes.
4. **Check data visibility as the reader.** Confirm that known recent events or metric series are visible in the configured indexes and time range, with the required host scope. Check lookup/object ACLs too. A test performed as your administrator account does not test the token user's or the owner's visibility.
5. **Check a scheduled run.** Use Collection health and Configuration Guardian to confirm successful, recent collection after the changes. Manual collection can be dispatched after privileged authorization and can succeed where the scheduled owner fails.
6. **Check the intended user experience.** Sign in as a scoped user/power/admin account and verify the tenants, results and operations each should see. For UAM, verify its stricter workspace boundary. If scheduled AI is enabled, verify an actual agent run under its named owner as well.

Table 12: Symptoms identify different permission layers

Symptom	Check first
Tenant missing or an action forbidden	TrackMe capability, tenant role lists, inherited roles and the username allowlist
`insuffi-cient_tenant_owner_capabilit`	The **named owner's** general seven-capability requirement on TrackMe; do not add these capabilities to every remote account
Manual collection works; scheduled collection fails	The owner, its remote-account use, source/result index access and collection receipt
`remote_account_refused` / `account_access_denied`	Local permission to **use** the account, distinct from destination token permissions
UAM catalog contains only the reader	Remote `list_all_users` / `list_all_roles` and effective identity visibility
`cluster_read_refused`	`list_indexer_cluster` on the **source reader**, not only on the manager account
`cluster_manager_unreadable`	The separate manager account, its destination, connectivity and three manager reads
DS API returns no clients	The API user's `_ds*` index access, recent phonehomes and the DS cluster member
Source search succeeds but returns no expected data	Allowed/disallowed indexes, search filters, forwarding, retention, host scope and correct search tier; transport success alone does not establish coverage
Cribl API works but traffic or CPU charts are empty	Stream metrics-index access or Edge's optional System Metrics configuration and data
Scheduled AI Agent reports a system owner	Reassign to a named owner with the required tenant and automation access

Configuration-UI requirements

TrackMe provides native interfaces under **Configuration**, replacing the Splunk UCC configuration page. System settings and account management require a Splunk administrator-class user with TrackMe admin capabilities, typically `trackme_admin` together with `admin` / `sc_admin`. A denied request produces an **Insufficient permissions** message.

- **System settings → General → Knowledge objects & RBAC defaults** sets the proposed owner, sharing and roles for **new** tenants.
- **Manage Remote deployment accounts** manages Splunk source/API connections and their account RBAC. **Manage Cribl connections** manages Cribl API credentials and scope.
- **Manage Email delivery accounts**, **Manage S3 export accounts** and **Manage Webhook accounts** manage the other integrations.
- An existing tenant's roles and owner are changed through its Virtual Tenant UI, as described below. These changes are separate from system-wide defaults.



See *Configuration* and *System settings*.

RBAC at tenant creation

RBAC and ownership are defined as part of creating a Virtual Tenant.

Hint

Preset the creation defaults

You can preset the proposed owner and roles for new tenants under **Configuration → System settings → General → Knowledge objects & RBAC defaults**.

See *System settings* for the native configuration interface. These defaults do not change the permissions of existing tenants.

Defining RBAC while creating a tenant:



Once the tenant exists, you can review the permissions and ownership TrackMe applied:





Worked example: per-team tenant access

This example shows how to scope access to Virtual Tenants by role. We have three tenants, each dedicated to a regional security-operations team:

- `secops-emea` — EMEA team, Splunk role `secops_emea`
- `secops-amer` — Americas team, Splunk role `secops_amer`
- `secops-uk` — UK team, Splunk role `secops_uk`

The administrator in this example is granted admin access to all three tenants. Each team receives its own scoped role for access:





Each role must also be granted access to the TrackMe application namespace:





Warning

Inheritance can over-grant access

By default, a new tenant lists all three built-in roles in each RBAC category. So if the role you use for access **inherits** from one of the built-in roles and you leave that default in place, its members are granted access and will see **every** tenant. To scope access cleanly, either make the desired TrackMe capability **native** to the user's role, or remove the built-in roles from the tenant's RBAC configuration.

When a member of the configured roles connects, they see only the tenants they have access to:



If a user is granted TrackMe app access but lacks the required TrackMe capabilities, they get this error screen instead:



Here the roles carry the TrackMe capabilities natively, so the user can access the resources as intended:



Updating RBAC for an existing tenant

You can update a tenant's RBAC policies and ownership at any time — through the UI or the REST API.

Through the Virtual Tenant UI

Open the Virtual Tenants interface, double-click the tenant, and open its administration screen:







Update the ownership and/or RBAC policies carefully: TrackMe updates the tenant configuration and re-assigns all of its knowledge objects if needed, updating their permissions accordingly.

Through the REST API

The same update is available from the `update_tenant_rbac` endpoint. The best way to discover any endpoint's usage — its parameters, HTTP method, and ready-to-copy **curl** and **SPL** examples — is the built-in **TrackMe REST API Reference** UI (open it from **API & Tooling → TrackMe REST API Reference**), which documents every endpoint:



For instance, to add the `siem_users` role to the user access of the `siem-quality-control` tenant:

```
| trackme url=/services/trackme/v2/vtenants/admin/update_tenant_rbac mode=post
    body="{'tenant_id': 'siem-quality-control', 'tenant_roles_admin':
```

(continues on next page)

(continued from previous page)

```
↪'emea_siem_admin',
          'tenant_roles_power': 'emea_siem_power',
          'tenant_roles_user': 'emea_quality_control,siem_users',
          'tenant_owner': 'srv-trackme'}"
```

Restricting a tenant to specific users

On top of the role model, a tenant can be restricted to an **allowlist of Splunk usernames**. When the allowlist (`tenant_allowed_users`) is non-empty, only those users — plus the tenant **owner** — see the tenant in the Virtual Tenants UI and via `trackmeload`; other ordinary users do not, even with a granting role. `splunk-system-user` bypasses this filter so scheduled searches and alert actions keep working. The dedicated findings workspaces also retain the `admin` / `sc_admin` administrative bypass: this is not an isolation boundary against Splunk administrators. It is an extra restriction layered *over* RBAC, not a replacement — a listed user still needs a role that grants access. Clear the list to remove the restriction. Set it from the `vtenants/admin/update_tenant_visibility` endpoint:

```
| trackme url=/services/trackme/v2/vtenants/admin/update_tenant_visibility␣
↪mode=post
    body="{'tenant_id': 'siem-quality-control', 'tenant_allowed_users': 'alice,bob
↪'}"
```

See also

- *Roles & access control* — the readable overview of the role model.
- *Creating a tenant* — where RBAC and ownership are first set.
- *Managing tenants* — updating a tenant after creation.
- *REST API reference* — the REST API the `update_tenant_rbac` call uses.
- *User Activity Monitoring* — User Activity Monitoring tenants.

8.2.7 Personal preferences

Each user can tailor the TrackMe interfaces to their own taste from the **Settings** panel. These preferences are **private**: changing them never affects anyone else, and they never change tenant data, access control or monitoring behaviour. Administrators set the **defaults** that apply until a user makes their own choice (see *Application-wide defaults* below).

Note

One theme choice: Light or Dark

TrackMe's interfaces are native Splunk UI (React) components, and the colour scheme is a single **Light** or **Dark** choice. Earlier releases styled their entity tables with the Tabulator JavaScript library, which offered named table themes (*Dark Site*, *Dark Midnight*, *Light Site*, *Light Modern*) and a separate icon theme. That component and those preferences no longer exist, and neither does the old `trackme_theme_default` system setting: the application default is now `default_theme` (see below).

Opening the Settings panel

Click the **settings** icon (the sliders button, *Open settings*) at the top right of the **Virtual Tenants** page, of any **Tenant Home** or of **Topology Studio**. The panel slides in from the right and every change applies immediately — there is no Save button.



Left: the panel on the Virtual Tenants page. Right: the panel on a Tenant Home, with its per-tenant section.

What you can set

Preference	Where	Effect
Volume unit labels	Virtual Tenants, Tenant Home	How volumes are labelled: **System default**, **IEC (MiB / GiB / TiB)** or **Splunk binary (MB / GB / TB)**. Only the label changes: every volume is computed in base 2 (1 GiB = 1024 MiB = 1024^3 bytes), Splunk style simply labels the same binary values MB / GB / TB.
Color Scheme	All three	**Light** or **Dark**. The choice is global: it applies to every TrackMe interface, including embedded views, the AI assistant and notifications.
Auto Refresh	All three	**Enabled** or **Disabled**. Whether the page refreshes its data automatically (on Topology Studio, the live-status polling of the canvas). Until you choose, the system default applies.
Tenant Card Detail Level	Virtual Tenants	**Enabled** shows the priority breakdown (low, medium, high, critical) on each tenant card; **Disabled** keeps the compact cards.
Virtual Tenants visibility	Virtual Tenants	Appears once you have hidden at least one tenant (tenant menu → **Hide this tenant**). **True** shows your hidden tenants again; **False** keeps them out of your grid. Hiding is personal and never restricts access (see *Managing tenants*).

Personal preferences for this tenant

On a **Tenant Home**, the panel adds a section that applies to **that tenant only** (it names the tenant). The controls depend on the kind of tenant:

Tenant	Controls
Entity tenants (DSM, DHM, MHM, FLX, FQM, WLK, VOL)	**Pie Charts** and **Single Values** (Show / Hide) — the priority and state charts and the KPI tiles above the entities table. **Table height — Auto** (58 % of the window, down to the window bottom when the pie charts or single values are hidden), **Fill window**, **Compact** (40 %) or **Tall** (75 %).
User Activity Monitoring	**Collection monitors**, **Single stats** and **Charts** (Show / Hide) — see *User Activity Monitoring*.
Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge	**Workspace context**, **Single stats** and **Charts** (Show / Hide), and **Rows per page** (25, 50 or 100) for the workspace tables. Hiding a layer never stops collection, findings or safety warnings.

Other choices TrackMe remembers

A few selections made elsewhere in the interfaces are kept with your preferences too: the **Saved View** selected on the Virtual Tenants page (see *Saved Views*), the list of tenants you have **hidden**, the **display mode** of the Topology Studio views browser (see *Views browser and display modes*) and the **Policies** layout (cards, compact or list) of the findings workspaces: one choice for UAM, one shared by Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge.

Where preferences are stored

Preferences live in a single browser cookie, `trackme_user_preferences`, and are read and written by every TrackMe interface. As a consequence:

- they follow **your browser**, not your Splunk account: another browser or another device starts from the system defaults;

- the cookie holds UI state only — no personal data, no tenant data;
- deleting the site's cookies in your browser resets them.

Clearing your preferences

Clear Preferences, at the bottom of the panel, asks for confirmation, then deletes the whole cookie and reloads the page: every choice above — colour scheme, auto-refresh, unit labels, per-tenant layouts, hidden tenants, the selected Saved View — returns to the system defaults. A few views remember purely local display options (for example chart options in the VOL views) in the browser's local storage; Clear Preferences leaves those untouched.

Application-wide defaults

Administrators set the defaults for all users from **Configuration → System settings → User interfaces**, the `[trackme_ui_defaults]` stanza of `trackme_settings.conf`. Like every Configuration interface, this page is reserved for Splunk admins (see *Configuration*).



Setting	Key	Default	Notes
Default theme	`default_theme`	`dark`	`dark` or `light`. Users override it with **Color Scheme**.
Auto-refresh	`auto_refresh`	`1`	Auto-refresh on (`1`) or off (`0`) by default. Users override it with **Auto Refresh**.
Action progress refresh rate	`auto_refresh_inte:`	`15`	Seconds between progress refreshes while an action runs (creating a tenant, running a tracker, training models). 5–300; raise it on slow systems.
Tenant Card Detail Level	`vtenants_card_det:`	`0`	Show the priority breakdown on tenant cards (`1`) or not (`0`). Users override it.
High-scale shadow page size	`shadow_page_size`	`25000`	Records fetched per batch when large entity tables load from shadow collections. 5,000–50,000.
Topology default entity count	`topol-ogy_default_max_e:`	`150`	Member entity nodes drawn when the topology graph view opens. 10–10,000; analysts can adjust it per view.
Topology maximum entity count	`topol-ogy_max_entities`	`10000`	Ceiling on the member entity nodes an analyst can request. 100–100,000.
Entity table max visible labels	`ta-ble_labels_max_vi:`	`4`	Label chips shown per entity before the rest collapse into **+N**. 1–10.
Volume unit labels	`vol-ume_unit_labels`	`iec`	`iec` (MiB / GiB / TiB) or `splunk` (MB / GB / TB); values stay binary either way. Users override it with **Volume unit labels**.

The precedence is always the same: a user's own choice wins, then the system default, then a built-in fallback. Changing a default affects users who have not made their own choice; it never overwrites a personal one. The deployment-wide default display mode of the Topology Studio views browser is set in the **Topology Studio** section of the same page.

Per-tenant table defaults

How a tenant's entity tables load and group is a **tenant setting**, not a personal one. Tenant administrators set it from the tenant menu → **Configure tenant** → **Tables & views** (see *Configuring a tenant*):



- **Pagination** — the table pagination mode (**Local**, client-side, the default; or **Remote**, server-side for very large entity sets) and size (`10000` records per page). New tenants inherit both from **System settings → General** (`pagination_mode`, `pagination_size`).
- **Tenant Home** — the default time range, the minimum width of the object column, whether metric rows open expanded, and the order of the Tenant Home tabs.
- **Table group-by** — per component, the field or fields the entities table is grouped by when it opens (for example `data_index` for DSM, `overgroup` for WLK, `group` for FLX and VOL). Users can still regroup on the fly with the table's **Group by** selector.

Note

The group-by settings keep their historical keys in `trackme_vtenants.conf` (`splk_dsm_tabulator_groupby`, `splk_flx_tabulator_groupby` and so on). The word *tabulator* in the key is a leftover of the former table component; the setting drives the grouping of today's entity tables.

See also

- *Navigating the TrackMe UI — in depth* — a visual tour of the interfaces, in dark and light mode.
- *Managing tenants* — hiding a tenant from your grid, and configuring a tenant.
- *Configuration — in depth* — every System settings section.

8.3 Monitoring Components

Everything TrackMe tracks is an **entity**, and every entity belongs to one of seven **monitoring components**. The component decides *what* is measured and which tracking logic applies. This part introduces the shared model, gives each component its own page, then covers the advanced tracker types that extend them.

- **Overview** — the seven components at a glance, the shared entity lifecycle, and how to choose between them.

- **DSM, DHM, MHM —** the data-flow family ("is my data arriving?").
- **VOL —** is my license volume normal? Drops, spikes and silence on the licensed volume of every index, with a trend and a month projection.
- **FLX —** track anything you can express in SPL.
- **FQM —** is my data well-parsed and CIM-compliant?
- **WLK —** are my scheduled searches healthy?
- **UAM —** *Who is doing what on my Splunk?* A dedicated tenant type outside the seven entity components: the accounts, their scheduled searches, their activity and their resource usage, turned into findings.
- **Cribl Stream —** *Is my Cribl Stream deployment healthy?* A dedicated tenant type that monitors one Cribl Stream deployment from the Cribl metrics already in Splunk and the Cribl Leader API: Sources, Destinations, Pipelines, Routes and Workers, findings from a policy catalogue and traffic ML models.
- **Cribl Edge —** *Are my Cribl Edge Fleets healthy?* A dedicated tenant type that follows the Fleets of a Cribl Edge Leader and their Edge Nodes, Sources and Destinations, read in bulk from the Leader at any fleet size, with findings from a policy catalogue.
- **Splunk HEC —** *Are my HTTP Event Collector endpoints healthy?* A dedicated tenant type that monitors the HEC endpoints, pools and tokens of one Splunk deployment from the telemetry Splunk already writes (`_introspection` and `_internal`), with findings from a policy catalogue and traffic models.
- **Splunk S2S —** *Is every receiver accepting, and is my intermediate tier keeping up?* A dedicated tenant type that watches the receivers, intermediate forwarders and pools of one Splunk deployment, traces back pressure to its origin, and learns traffic ML models per pool.
- **Splunk UF Fleet —** *Which forwarding agents are connected, at any fleet size?* A dedicated tenant type that follows the forwarding agents and the Deployment Servers that manage them, with groups, expected agents and mass events as one finding.
- **Splunk Forwarder Collectors —** *Is every input collecting?* A dedicated tenant type that follows the forwarders used as collection points — heavy forwarders running add-ons, scripted and modular inputs, and the forwarders reading syslog files — and every input they run, with groups, a minimum of healthy members and traffic models.
- **Findings —** the model the dedicated tenant types share: policies, findings, Finding groups, notifications, findings actions and the AI Agents.
- **Hybrid & replica trackers —** advanced tracker types that feed entities into those components: build a tracker from your own SPL (local or remote), or mirror entities from another TrackMe deployment.

8.3.1 The seven components at a glance

A **component** is the kind of thing an entity is. It determines what TrackMe measures, which thresholds apply, and what "healthy" means. Every entity belongs to exactly one component, and the seven codes — DSM, DHM, MHM, VOL, FLX, FQM, WLK — appear throughout the interface, the metrics, and the REST API.

Code	Component	What it tracks	Use it to answer
DSM	Data Source Monitoring	`(index, sourcetype)` data feeds — delay, latency, volume, quality.	*Is my data flowing?*
DHM	Data Host Monitoring	Host × sourcetype activity per endpoint.	*Who is sending, and what?*
MHM	Metric Host Monitoring	Hosts sending metric data, by metric category.	*Are my metrics arriving?*
VOL	Volume Outliers	Licensed volume per index — drops, spikes, inactivity, trend and projection.	*Is my license volume normal?*
FLX	Flex Objects	Anything you can express in SPL, with custom KPIs.	*Is this custom thing healthy?*
FQM	Field Quality Monitoring	Field extraction quality and CIM compliance.	*Is my data well-parsed?*
WLK	Workload Knowledge	Scheduled-search execution health.	*Are my searches healthy?*

DSM, DHM, and MHM form the **Splunk Feeds** family and share a delay/latency model. FLX, FQM, and WLK share a **dynamic-thresholds + metrics** model. VOL stands on its own: one entity per index fed by the license usage logs, with ML outliers as its state.

Note

UAM — User Activity Monitoring is not an eighth entity component but a dedicated **tenant type**, outside the seven: it has no entities, no state and no scoring — it answers *who is doing what on my Splunk?* with **findings** about accounts and their scheduled searches, activity and resource usage. See *User Activity Monitoring*.

Cribl Stream is another dedicated **tenant type**: it monitors one Cribl Stream deployment — its Sources, Destinations, Pipelines, Routes and Workers — from the Cribl metrics already in a Splunk metric index and the Cribl Leader API, and raises **findings** from a policy catalogue and per-object traffic ML models. See *Cribl Stream and Cribl Edge*.

Cribl Edge, the third dedicated **tenant type**, follows the Fleets of a Cribl Edge Leader and their Edge Nodes, Sources and Destinations, and raises **findings** from a policy catalogue. See *Cribl Edge*.

Splunk HEC, the fourth dedicated **tenant type**, monitors the HTTP Event Collector endpoints of one Splunk deployment, the pools they form and the tokens senders use, from the HEC telemetry Splunk already writes, and raises **findings** from a policy catalogue. See *Splunk HEC*.

Splunk S2S and **Splunk UF Fleet**, the fifth and sixth dedicated **tenant types**, follow the forwarding of one Splunk deployment from what Splunk already writes: Splunk S2S watches the receivers, intermediate forwarders and pools of the S2S receiving side (see *Splunk S2S*), Splunk UF Fleet the forwarding agents and the Deployment Servers that manage them (see *Splunk UF Fleet*).

Splunk Forwarder Collectors, the seventh dedicated **tenant type**, follows the forwarders a Splunk deployment uses as collection points and every input they run — add-ons, scripted and modular inputs, syslog files — and raises **findings** when an input stops, fails, goes quiet or is disabled, or when a collector or a group of collectors falls short. See *Splunk Forwarder Collectors*.

The seven dedicated tenant types share one findings model: *Findings, Finding groups and findings actions*.

The shared entity model

However different the seven components look, every entity has the same shape and lifecycle.

An **entity** always has:

- an **object** — its canonical, unique key (what it identifies differs per component);
- an **alias** — an editable display name;

- a **state** — green / orange / red / blue (see *Entity State & Scoring*);
- a **score**, **thresholds**, anomaly fields, and a cache of its latest **metrics**.

And every entity goes through the same lifecycle each cycle:

```
DISCOVER  →  TRACK  →  EVALUATE  →  ALERT / LIFECYCLE
a tracker     each run    decision maker   stateful alerting,
enumerates    refreshes   sets state +     acknowledgment,
units and     metrics     score from       purge of inactive
creates       & times     thresholds       entities
entities
```

Note

Trackers are scheduled searches, usually running every five minutes. The freshness of a delay measurement is therefore only as precise as the tracker cadence — a five-minute schedule can report a delay up to five minutes after the fact. Entities can also be **declared before they exist** (see expected sources / hosts for *DSM* and *DHM*); a declared entity starts red until real data arrives.

Choosing a component

You want to track...	Use
The freshness and volume of an index + sourcetype feed	**DSM**
Which hosts are sending, and what sourcetypes, per host	**DHM**
Hosts sending data to a metrics index	**MHM**
The licensed volume of every index — drops, spikes, silence, trend, projection	**VOL**
Field parsing quality / CIM coverage	**FQM**
Scheduled-search execution health (skips, errors, delay, cost)	**WLK**
Anything else you can write a search for	**FLX**

What every component shares

These concerns apply to all seven and are documented once, in their own parts:

- **Classification** — priority, tags, labels, and SLA (*Classification & Protection*).
- **Protection** — logical groups, the disruption queue, and acknowledgments (*Classification & Protection*, *Alerting*).
- **Stateful alerting** — one state machine for all seven (*Alerting*).
- **Scoring** — one hybrid score; the per-anomaly weights differ per component (*Entity State & Scoring*).
- **ML outlier detection** — a natural fit for DSM and FLX, and the very purpose of VOL; less useful, and off by default, for DHM, MHM, FQM, and WLK (*Machine Learning*).
- **Topology graphs** — every component except VOL can render its entities as an interactive link / network graph (see *Topology graphs* below).

Where the components differ

A few capabilities are specific to certain components. Keep this map in mind:

Capability	Available for
Variable delay & lagging classes	DSM, DHM (MHM uses static thresholds)
Data sampling, elastic sources, lookup monitoring	DSM only
Expected-sources / hosts injection & coverage-gap analysis	DSM, DHM
Dynamic, time-sliceable metric thresholds	FLX, FQM
Versioning, orphan detection, app enablement, SVC cost	WLK
Free-form KPIs, template catalog, converging trackers	FLX
ML outliers as the primary state, detection direction per index (drops / spikes / both)	VOL
Inactivity policy (static or by day-of-week / hour-of-day), history backfill, month projection, Global license usage analytics	VOL

Topology graphs

Every component except VOL can render its entities as an interactive **topology graph** — open it from the graph icon in the **Tenant Home header** (the icon is contextual to the component tab you are viewing). The modal offers two views, a **Link graph** and a **Network graph**, and lays the tenant's entities out as a hierarchy:

Component	Hierarchy
DSM	tenant → index → entity
DHM	tenant → host → index → sourcetype (→ extras)
MHM	tenant → host → metric category
WLK	tenant → app → entity
FQM	tenant → data model → entity
FLX	plus dedicated *converging* and *use-case* graphs (tenant → service / group → members)

Note

VOL has no Tenant Home topology graph — one entity per index leaves no hierarchy to draw. VOL entities can still be placed in *Topology Studio* views like any other component.

Each node is coloured by status (/ / /), and a parent rolls up to the **worst** status among its descendants — so a single red leaf turns its whole branch red, letting you trace a problem from the tenant down to the offending entity at a glance. You can search the graph, filter to **All / Down / Up**, and cap the number of leaf nodes drawn (default 150) on large tenants.

See also

- *Entity State & Scoring* — how state and score are computed from thresholds.
- *Core concepts and vocabulary* — the core vocabulary.
- *VOL — Volume Outliers* — the seventh component, Volume Outliers.
- *User Activity Monitoring* — User Activity Monitoring, a dedicated tenant type outside the seven components.
- *Cribl Stream and Cribl Edge* — Cribl Stream and Cribl Edge, two other dedicated tenant types.
- *Splunk HEC* — Splunk HEC, the dedicated tenant type for HTTP Event Collector endpoints.
- *Splunk S2S* — Splunk S2S, the receivers, intermediate forwarders and pools.

- *Splunk UF Fleet* — Splunk UF Fleet, the forwarding agents and their Deployment Servers.
- *Splunk Forwarder Collectors* — Splunk Forwarder Collectors, the collection points and every input they run.

8.3.2 DSM — Data Source Monitoring

Data Source Monitoring is TrackMe's original and most fundamental component. Each entity is a **data feed** — an `(index, sourcetype)` combination, optionally split further by a *break-by* field — and DSM answers the first question of any healthy platform: **is my data still flowing, on time, in the expected volume, and well-formed?**





What DSM tracks

Signal	What it means
Delay	How old the most recent event is (`now - last_event_time`). Breached when the latest data is older than the **delay max** threshold (`data_max_delay_allowed`).
Latency	The lag between an event's timestamp and when Splunk indexed it (`_indextime - _time`). Breached above the **lag max** threshold (`data_max_lag_allowed`).
Volume outliers	Unusual event volume, detected by ML density-function modelling — a natural fit for DSM (see *Machine Learning*).
Future events	Events arriving with a timestamp in the future (beyond tolerance) — usually a time-zone or parsing problem.
Data quality	Unknown formats, mixed formats, and format drift, surfaced by **data sampling** (see below).

In an entity's overview you can switch the source of these figures between **TrackMe metrics** (fast, served from the metrics index) and **Splunk queries** (direct and exact, but slower and requiring index access).

Tip

Delay and **latency** are the two most-confused KPIs in TrackMe (people call everything "latency"). If the difference is not crystal clear, read *Delay vs latency: what they actually mean* — it is short, diagram-led, and worth five minutes.

Defining what gets tracked

- **Root constraint** — the base SPL filter that defines a tenant's data scope.
- **Allowlist / blocklist** — regex rules that include or exclude specific `index + sourcetype` combinations from automatic entity creation.
- **Break-by** — an optional extra field that splits one feed into finer entities.

Note

Sourcetype-explosion safeguard. If a single index produces more sourcetypes than the configured cap, bulk DSM creation is refused and an alert is raised. The fix is to blocklist the noisy sourcetypes and re-cover the index with a single `<index>:@all` aggregate entity.

Delay thresholds, in depth

DSM delay handling is rich, because "late" means different things at different times:

- **Static or variable delay.** A delay threshold can be a single fixed value, or a **variable** time-slot policy (day-of-week × hour-of-day) — tighter during business hours, relaxed overnight and at weekends.
- **Lagging classes** — named, reusable delay/lag rule sets by cadence (high-frequency, nightly batch, quarterly…). Assign a feed to a class for consistent thresholds.
- **Adaptive delay** — TrackMe can compute delay thresholds automatically from an entity's own history.
- **Threshold lock** — lock a feed's delay and lag thresholds so neither adaptive delay nor a lagging class can rewrite them; a red lock icon marks locked entities.

The full thresholds-and-scoring model lives in *Entity State & Scoring*.

Note

One authority manages delay per tenant. If the AI *Feed Lifecycle Advisor* is set to manage DSM delay, it takes over and the legacy adaptive-delay / variable-delay auto-review settings are disabled automatically. Turning the advisor back off does not re-enable them — re-toggle them by hand if you want them back. See the *Artificial Intelligence* section.

Elastic sources

Elastic sources are virtual DSM entities built from arbitrary SPL — `tstats`, `raw`, `mstats`, or `from` a data model or lookup — rather than from straightforward `index/sourcetype` discovery. One search defines one entity. They suit syslog collectors, data-model- or lookup-derived sources, and cases that rely on search-time fields. They come in two flavours: **shared** (one pooled, concurrency-capped tracker) and **dedicated** (an independent scheduled report). See *Elastic sources* for the full treatment.

Tip

For high-cardinality cases, *Hybrid trackers* have largely superseded elastic sources — but elastic sources remain the right tool for `from` / `raw` single-entity edge cases.

Lookup monitoring

DSM can track Splunk **lookups** (CSV and KV Store) as first-class entities — one entity per lookup — to detect whether a lookup is still being refreshed. Only the delay threshold applies (latency is not meaningful for a lookup). This is covered with the other integrations in *Integrations & Remote Deployments*.

Data quality: sampling

Data sampling periodically samples a feed's events and matches them against format models to detect unknown formats, multi-format feeds, and format drift. TrackMe ships out-of-the-box regex models, and you can add your own. Data sampling is configured under operations — see *Operations & Administration*.

Declaring feeds before they exist

You can declare feeds that **should** exist from a Splunk lookup — **expected sources injection**. For DSM both `index` and `sourcetype` are required. Injected feeds appear **red** until real data arrives, then their metadata auto-populates. Imports can be one-shot or recurring.

Coverage-gap analysis complements this: a read-only reconciliation of a reference (a lookup, or a live `tstats` / `raw` search) against the feeds TrackMe actually tracks, returning three downloadable lists — only in the reference, only in TrackMe, and in both.

Delayed and inactive entities

When a feed falls so far behind that it drops out of the tracker's search window, TrackMe does not lose track of it — two cooperating processes keep its delay metadata fresh (this behaviour is shared with *DHM*):

- **Health Tracker — the ``untracked_entities`` task.** The tenant's Health Tracker runs every **5 minutes**; its `untracked_entities` task detects entities no longer covered by any tracker and refreshes their calculated delay, stamping `tracker_health_runtime` on each entity it handled. Troubleshoot it with:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth task=
↪"untracked_entities"
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪tenant_id="mytenant" component="splk-dsm" task="untracked_entities"
```

- **Delayed entities inspector.** A scheduled job named `trackme_<component>_delayed_entities_inspector_tracker_` runs the `trackmesplkfeedsdelayedinspector` command. It picks up the entities the Health Tracker flagged (via `tracker_health_runtime`) and runs one targeted search per entity to refresh its metadata, on a tiered cadence keyed to current delay — entities delayed **< 24h** every ~30 min, **< 7 days** hourly, and

beyond that once every 24 hours — until the entity reaches the tenant's auto-disablement period. Configure it per tenant under **Delayed entities inspector** in the tenant settings. The last inspection time is stored on the entity as `tracker_runtime` (epoch), shown human-readably in the entity's right-click contextual menu.

Each inspected entity logs a pair of lines — `Executing entity delayed tracking resulting search=` then `delayed_entity_search_results=`:

```
index=_internal␣
↪sourcetype=trackme:custom_commands:trackmesplkfeedsdelayedinspector tenant_id=
↪"mytenant" component="splk-dsm" object="myentity"
```

You can also follow the inspector (and TrackMe's other backend logics) in the **Handler events** tab — the `trackme:handlers` sourcetype in the tenant's summary indexes.

See also

- *Feeds thresholds (delay & latency)* — delay & latency thresholds in full: static, lagging classes, adaptive (ML), and variable delay.
- *Expected sources & hosts — in depth* — alert on sources that stopped sending.
- *Entity State & Scoring* — delay, latency, and the impact score in detail.
- *DHM — Data Host Monitoring* — the host-centric counterpart of DSM.
- *Machine Learning* — native outlier detection.
- *Operations & Administration* — data sampling configuration.
- *Metrics reference* — the DSM delay, latency, and volume metrics.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.3 DHM — Data Host Monitoring

Data Host Monitoring tracks **host × sourcetype activity** — which hosts are sending data, and what they are sending. Where *DSM* watches feeds, DHM watches the *senders*, which makes it the natural component for forwarder fleets and for declaring the hosts that *should* be reporting.





What DHM tracks

Each entity is a **host**, carrying the index × sourcetype combinations it sends. DHM shares DSM's delay/latency model:

- **Delay** and **latency** against the configured thresholds.
- **Volume outliers** per host/sourcetype (off by default — host volume is noisy).
- **Future events** and **distinct-host drift**.

> **Note**
>
> **The asset field.** Each host also carries an `asset` field — a de-duplicated set of every known variation of the host name (canonical value, alias, short hostname, FQDN). TrackMe uses it to match short names against fully-qualified ones, so a host is not tracked twice as `web01` and `web01.corp.example`.

Per-host granularity

DHM lets you choose how finely a host is broken down, via the tracker's break-by mode:

Mode	Grain
Standard	Combos at `(index, sourcetype)`.
Merged	Sourcetype collapsed — `(index, all sourcetypes)`.
Custom	Keyed by any single field you choose (for example `cmdb_ci`), with up to five extra ordered metadata dimensions (such as `source`).

Declaring hosts that should be sending

DHM's signature feature is **expected-hosts injection** — telling TrackMe which hosts *ought* to be reporting, so a host that goes silent (or never starts) is visible as a problem rather than an absence.

- Declare hosts from a Splunk lookup (local or remote, CSV or KV Store). For DHM only the `host` field is required.
- The wizard previews the lookup, maps fields, and **simulates** the import (reporting how many rows are new, existing, or already tracked) before you commit. Imports can run once or on a recurring schedule.

- Injected hosts appear **red** until data arrives; their index/sourcetype placeholders fill in automatically — and clean up automatically — once the DHM tracker sees real data.
- Existing entities are never duplicated; the `asset` field makes the matching short-name/FQDN aware, and large lookups are batched automatically.

Host coverage-gap analysis complements injection: a read-only reconciliation of a reference host list (a CMDB lookup, or a `tstats` / `raw` search) against the hosts TrackMe tracks — returning the hosts only in your reference (the coverage gap), only in TrackMe, and in both.

Shared with DSM

DHM shares DSM's configuration model: root constraint, allowlist/blocklist, variable delay, lagging classes, adaptive delay, threshold lock, the AI Feed Lifecycle Advisor delay mutex, and the delayed/inactive-entity inspector. See *DSM — Data Source Monitoring* for those.

Note

DHM has **no data sampling and no elastic sources** — those are DSM-only. On large fleets (10k+ hosts), keep SLA thresholds moderate to avoid alert storms, and remember that inactive hosts are auto-disabled after the tenant's auto-disablement period.

See also

- *Feeds thresholds (delay & latency)* — delay & latency thresholds (shared with DSM): static, lagging classes, adaptive (ML), and variable delay.
- *Expected sources & hosts — in depth* — inject expected hosts so a host that stops sending turns red.
- *DSM — Data Source Monitoring* — the feed-centric counterpart and the shared delay model.
- *Integrations & Remote Deployments* — CMDB enrichment for host metadata.
- *Entity State & Scoring* — thresholds and scoring.
- *Metrics reference* — the DHM host metrics.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.4 Feeds thresholds (delay & latency)

For the splk-feeds family — *DSM* and *DHM* — the core question is *"is my data on time?"*. Two KPIs answer it, and each has a threshold an entity must stay under:

- **Delay** — how old the most recent event is (`now - last_event_time`); breached above `data_max_delay_allowed`.
- **Latency** — the gap between an event's timestamp and when Splunk indexed it (`_indextime - _time`); breached above `data_max_lag_allowed`.

Tip

New to these two, or not sure which is which? Start with *Delay vs latency: what they actually mean* — a plain-English, diagram-led explanation of the difference (and how they appear in the UI).

When discovered, every entity starts with a **1-hour** (3600-second) maximum for both. From there, TrackMe gives you several ways to set the *right* threshold for each feed — from a fixed value to fully automated, machine-learning-driven thresholds.

Ways to set a delay threshold

Approach	When to use it
Static threshold	A single fixed value — the default. Set it per entity, simulate it first, and read the two worked anatomy examples.
Lagging classes	Rule-based thresholds at scale — match entities by index, sourcetype or priority and apply a shared delay (and optional latency) threshold.
Adaptive delay (ML)	Let TrackMe learn each feed's normal lag from history and maintain the threshold for you.
Variable delay	Time-aware thresholds that change by day-of-week and hour-of-day — tight during business hours, relaxed at night and on weekends.

Who manages a threshold: the Threshold Lock and the AI Advisor

Two cross-cutting ideas decide *who* owns an entity's delay threshold.

The Threshold Lock. Setting thresholds used to mean answering two easily-confused per-entity questions — *"allow adaptive delay to auto-tune this?"* and *"should this entity override lagging classes?"* — that interacted in non-obvious ways and could each silently rewrite a value you had set by hand. A single per-entity **Threshold Lock** (a red , pinning delay *and* latency together) replaces both with one yes/no:

- **Locked** — *this value is mine.* TrackMe pins your manual thresholds and **every** background writer skips the entity: adaptive delay, variable-delay auto-review, and lagging-class management. A periodic reconcile even restores your pinned values (and, for a variable entity, your slot schedule) if anything drifts.
- **Unlocked** — *TrackMe, manage it.* Lagging classes and adaptive / variable adaptive delay are free to set and maintain the thresholds.

(The lock is the single control behind the older `allow_adaptive_delay` and `data_override_lagging_class` fields, which are now derived from it and no longer edited directly. Locking is governed per tenant by `delay_threshold_lock_enabled`.)

Important

The AI Feed Lifecycle Advisor takes precedence. When the AI *Feed Lifecycle Advisor* is enabled to cover DSM/DHM, it becomes the authority for delay thresholds: the legacy **adaptive-delay** and **variable-delay auto-review** mechanisms are automatically disabled (their per-tenant toggles are forced off and their trackers short-circuit at runtime). Turning the advisor back off does **not** re-enable them automatically — you re-toggle them by hand if you want them back. See the *Artificial Intelligence* section.

Delay vs latency: what they actually mean

People call everything "latency" — a feed is late, a dashboard is stale, an alert fired, and it all gets labelled *latency*. In TrackMe these are **two different measurements** that answer **two different questions**, and telling them apart is the single most useful thing you can learn about feeds monitoring. Get this right and the entities table, the thresholds, and the alerts all suddenly make sense.

Two timestamps

Every event in Splunk carries two moments in time:

- `_time` — **when the event happened** (the timestamp inside the event itself).
- `_indextime` — **when Splunk indexed it** (when it actually arrived and was stored).

Those two are never exactly equal, and the whole idea of delay and latency falls out of the gap between them and the gap to *now*:

- **Latency** = `_indextime - _time` — *"when an event arrives, how late is it?"* A measure of the **ingestion pipeline** (forwarding, queues, network, clock skew). It is a **per-event** property.

- **Delay =** `now - last_event_time` **—** *"how long since we last heard anything at all?"* A measure of **freshness**: is the feed still alive right now?

The mailbox analogy

Think of a feed as your postbox:

- **Latency** is how long a letter takes from the day it was **written** (the postmark — `_time`) to the day it **lands in your box** (`_indextime`). A letter postmarked Monday that arrives Thursday has three days of latency.
- **Delay** is how long since the **last letter of any kind** turned up. If nothing has arrived for five days, your delay is five days — the box has gone quiet.

The two are independent. Someone can drop a stack of old, week-old postmarked letters in today (**low delay, high latency** — fresh arrivals, but every one is stale). Or your reliable daily letter can simply stop coming (**high delay, low latency** — when letters *did* arrive they were same-day, but now there is nothing).

Why you can't collapse them into one

Because they are independent, a feed sits in one of four states — and the cause you go looking for is different in each:

The practical upshot: **an alert on latency and an alert on delay are telling you different things.** A latency breach points at your **ingestion path**; a delay breach points at the **source** (has it stopped?). Fixing one does nothing for the other.

How TrackMe measures each

- **Latency** is computed per event as `_indextime - _time`. On an entity, TrackMe surfaces the **current**, **average**, and **95th-percentile** latency over a recent window (so a single odd event does not dominate).
- **Delay** is computed as `now - last_event_time` — always relative to *this moment*, which is why a delay figure keeps climbing on its own while a feed is silent.

Each has its own threshold, and an entity turns red when it goes over:

- `data_max_delay_allowed` — the **delay max** (freshness ceiling).
- `data_max_lag_allowed` — the **lag max** (per-event latency ceiling).

New entities start with a 1-hour default for both. See *Feeds thresholds (delay & latency)* for the ways to set the *right* value per feed (static, lagging classes, adaptive, and variable).

Where you see them in the UI

In the **entities table**, each feed row shows:

Column	Meaning
lag (event / ingest)	A combined summary cell: the current **delay** (event) and **latency** (ingest), in that order.
Latest event	The newest event seen, by `_time` (drives **delay**).
Latest ingest	The newest event seen, by `_indextime`.
Delay max	The current delay threshold for the entity (`data_max_delay_allowed`).
Lag max	The current latency threshold for the entity (`data_max_lag_allowed`).

On an entity's own view, the KPI stat tiles read:

- **CURRENT EVENT DELAY —** `now - last_event_time`.
- **AVG INGESTION LATENCY / PERC95 INGESTION LATENCY** — the mean and 95th-percentile latency over the recent window.

All of these display in **seconds**, switching to a `[D+]HH:MM:SS` duration format (via Splunk's `tostring(…, "duration")`) once the value exceeds 60 seconds — so `4200` shows as `01:10:00` and a two-day-stale feed as `2+00:00:00`.

See also

- *Feeds thresholds (delay & latency)* — setting the delay and latency thresholds per feed.
- *DSM — Data Source Monitoring* — the DSM component these KPIs describe.
- *Entity State & Scoring* — how a breach feeds the entity state and impact score.

Delay & latency thresholds

This is the starting point for splk-feeds thresholds: how the two KPIs work, how to read an entity's current thresholds, how to set them by hand, and how to validate a value before you commit it. For the automated and rule-based options, see *lagging classes*, *adaptive delay*, and *variable delay*.

Delay and latency, briefly

TrackMe measures two independent things for every feed entity, and gives each its own threshold.

Delay — is the data fresh?
: How old the most recent event is: `delay = now() - max(_time)`, where `_time` is when the event happened *at the source*. It answers *"how long ago did the most recent event arrive?"* High delay means the source has stopped producing fresh data — it is silent, or only sending old, backlogged events.

Latency — how slow is the pipeline?
: How long an event took to travel from the source to Splunk's index: `latency = _indextime - _time`, where `_indextime` is when Splunk *indexed* it. It answers *"how delayed was the pipeline between the source and Splunk?"* High latency points to a forwarding, buffering, or queuing problem — the events exist, but they arrived late.

The two are **independent** — you can have one without the other, and each tells a different story:

Pattern	What it means
High latency, low delay	Events are arriving, but they were already old when they got there — the source is fine, the transport is slow (pipeline buffering).
High delay, low latency	Events are indexed promptly once they arrive, but the source has stopped sending fresh data (a gap or silence).
Both high	The source is sending delayed, backlogged events that *also* took a long time to transit — the most concerning pattern.

Because they mean different things, **tune them independently**. On every monitoring cycle TrackMe checks each against its own threshold — delay against the entity's delay policy (static or variable), latency against `data_max_lag_allowed` — and each breach contributes to the *impact score* that drives the entity's state. A new entity starts with a **1-hour** (3600-second) maximum for both; the rest of this page is about setting better values.

Reviewing current thresholds

An entity's active thresholds — and which delay policy is in force (static or variable) — are shown in the entities table (a two-value cell for delay and latency) and on the entity's own screen.

Setting a threshold per entity

To set thresholds by hand, open the entity, go to its threshold settings, and you can:

- set the maximum acceptable **latency**;
- set the maximum acceptable **delay**;
- choose whether to alert on **both** KPIs (the default) or only one;
- **lock** the threshold so neither a *lagging class* nor *adaptive delay* will overwrite your value (see *Who manages a threshold: the Threshold Lock and the AI Advisor*).

Tip

The **Threshold Lock** replaces the old per-entity "override lagging classes" checkbox. Lock a threshold when you have deliberately tuned it and want it left alone; leave it unlocked to let lagging classes or adaptive delay manage it.

Simulating a threshold before you apply it

Before committing a value, use **Simulate thresholds** to test it against the history TrackMe has already accumulated. Provide candidate latency and delay values and TrackMe replays them over the entity's summary events, returning one event per breach that *would* have occurred — so you can see whether a value is too tight before it pages anyone. For a provider that sends data once per hour but indexes it promptly, you would keep latency tight and allow a little over an hour of delay, confirm the simulation is quiet, then apply.

Worked example: an entity suffering from latency

Latency means events are indexed well after their event time — you may *also* see delay, but not necessarily; a feed can mix near-real-time events with late ones. The two measures will usually differ, which is expected.

When an entity breaches its latency threshold, the rising latency is visible in both the overview and Performance Metrics charts, the **status message** spells out the problem, and **Smart Status** adds an automated investigation. To review the offending events by hand — here, events indexed more than 15 minutes (900 s) late over the last 4 hours:

```
index=myindex sourcetype=mysourcetype _index_earliest="-4h" _index_latest="+4h"␣
↪earliest="-4h" latest="+4h"
| eval indextime=_indextime, latency=(_indextime-_time)
| where latency>900
| eval indextime=strftime(indextime, "%c")
| table _time indextime latency _raw
```

Once the root cause is fixed and latency returns to normal, TrackMe brings the entity back to green after a short while.

Worked example: delay without latency

Some feeds are perfectly on time when they *do* arrive, but arrive infrequently — batched data. A provider that emits one batch per day shows no latency but accrues delay between batches; TrackMe surfaces the current delay alongside the latest event (`_time`) and the latest ingested event (`_indextime`). The fix is to set a delay of a little over 24 hours while keeping latency low, after which the entity returns to green.

A feed that arrives once per hour is the more common case: the default 3600-second delay will alert if the provider runs even slightly late, and too tight a value (say 3000 s) produces constant delay alerts — so you settle on something like 4200 s (1 hour 10 minutes). As always, tolerating delay in *delivery* does not mean tolerating latency in *indexing* — tune the two KPIs independently.

See also

- *Lagging classes* — rule-based thresholds at scale.
- *Adaptive delay (ML)* — let ML maintain the delay threshold.
- *Variable delay* — time-aware thresholds.
- *Entity State & Scoring* — how a breach feeds the impact score and state.

Lagging classes

A **lagging class** is a rule that sets delay (and optionally latency) thresholds for every entity that matches it — the scalable way to manage thresholds without touching entities one by one. Define a class once for "all firewall syslog feeds" or "everything in the `sec_*` indexes", and matching entities pick up its thresholds automatically.

Lagging classes are **per component**: DSM, DHM and MHM each keep their own set (MHM matches on metric category rather than delay/latency). This page covers the splk-feeds (DSM/DHM) classes.

How a lagging class is defined

Each class is a matching rule plus the thresholds to apply:

Field	Meaning
name	The pattern to match against (an index name, a sourcetype, or a priority value).
level	What the pattern matches on: `index`, `sourcetype`, or `priority`.
match mode	How the pattern is compared: `exact`, `wildcard`, or `regex` (regex is a **full match**).
delay	The delay threshold to apply — either a single **static** value, or a full **variable** schedule (named time slots, see *Variable delay*).
latency *(optional)*	An optional latency (lag) override; leave it empty to not touch latency.

So a single class can apply a static delay, a time-aware variable delay, and/or a latency override, to any set of entities you can describe by index, sourcetype or priority.

How a class is matched to an entity

When several classes could apply, TrackMe resolves a single winner deterministically:

1. **By level**, most specific first: an `index` match beats a `sourcetype` match, which beats a `priority` match. The first level with any match wins.
2. **Within a level**, the most specific match mode wins — `exact` over `wildcard` over `regex` — with the oldest rule breaking any tie.
3. For multi-value entity fields (common in DHM), a class matches if **any** value matches.

You can preview which entities a class would match with the lagging-class **simulation** before saving it.

Note

An entity whose threshold is **locked** (see *Who manages a threshold: the Threshold Lock and the AI Advisor*) is skipped by lagging classes entirely — its manual value stands. Likewise, latency rules from a class apply independently of the delay policy, so an entity on *variable delay* still honours a class's latency override.

Example: a class for a firewall sourcetype

Here we define a class at the **sourcetype** level for `netscreen-firewall`, applying a delay of 2 hours (7200 s) and a latency of 10 minutes (600 s). Once the trackers have run at least once and the matching entities are active, their thresholds are set from the class automatically.

See also

- *Delay & latency thresholds* — per-entity thresholds and the Threshold Lock.
- *Variable delay* — the time-slot schedule a class can carry.
- *Priority* — the priority a `priority`-level class matches on.

Adaptive delay (ML)

Adaptive delay lets TrackMe **learn each feed's normal lag from history and maintain its delay threshold automatically** — so a feed whose natural cadence drifts over time keeps a sensible threshold without anyone re-tuning it. It is handled by a dedicated tracker, `trackmesplkadaptivedelay`, which inspects entities in a delay breach and, when the conditions are met, updates the delay threshold and records an audit trail.

The more history TrackMe has for an entity, the more accurate the computed threshold.

Important

When the AI **Feed Lifecycle Advisor** covers DSM/DHM, it owns delay management and adaptive delay is **disabled** automatically (the tracker short-circuits and its tenant toggle is forced off). See *Who manages a threshold: the Threshold Lock and the AI Advisor*.

How a run works

The adaptive-delay tracker runs **every 20 minutes** by default, as a **per-entity batch under a runtime budget** (900 seconds by default, never more than the schedule interval — see *batches under a runtime budget*). Each occurrence selects the entities that need review, processes as many as the budget allows, and resumes with the rest on the following cycle — so on a large tenant a multi-minute run simply means a large batch, not a stuck search.

Adaptive delay is also **self-limiting by design**: an entity only enters the review bucket when it is actually in a delay-breaching state, and it is further gated by the entity's SLA compliance — entities comfortably within SLA are skipped. The cost therefore scales with how much of the estate is *unstable*: the cleaner and more stable the environment, the fewer entities adaptive delay has to review.

How an entity is selected

Each run, an entity becomes a candidate only if **all** of these hold — otherwise it is skipped:

```
enabled  and  currently red
  └─ in a delay breach (delay_threshold_breached)
       └─ threshold UNLOCKED                  ── locked? → skip (your pinned value␣
↪stands)
            └─ current delay ≥ min_delay_sec  (3600 s by default)
```

(continues on next page)

(continued from previous page)

```
                └─ AI Feed Lifecycle Advisor not covering DSM/DHM
                                                  ── advisor covers? → skip (it owns␣
↪delay)
                     └─ ✓ candidate → compute & update the threshold
```

(See *Who manages a threshold: the Threshold Lock and the AI Advisor* for the lock and the AI Advisor gate.)

It also reviews entities it has already managed: an entity updated in the last 4 hours is left to settle; one updated 4–24 hours ago whose threshold was *increased* is re-checked; and beyond that, managed entities are reviewed on a rolling 7-, 15- or 30-day cadence (`review_period_no_days`) so a feed that has stabilised can have its threshold tightened back down.

How the threshold is computed

For a selected entity, TrackMe models the observed lag metric (`trackme.splk.feeds.lag_event_sec`) with its **native ML engine** (`trackmefit` — no Splunk MLTK dependency) to find an upper bound on normal lag, then rounds to the nearest hour and adds a one-hour safety buffer (the variable-delay path rounds up instead). In essence:

```
| mstats latest(trackme.splk.feeds.lag_event_sec) as lag_event_sec
    where `trackme_metrics_idx(mytenant)` tenant_id="mytenant"
    object_category="splk-dsm" object="myobject" by object span=5m
| trackmefit ...                          ``` native ML upper-bound on normal lag ```
| eval adaptive_delay = (round(UpperBound/3600, 0) * 3600) + 3600
```

When **re-reviewing** an entity after an earlier update, TrackMe runs the same logic over three windows — last 24 hours, 7 days and 30 days — and aggregates them, which lets it react faster when a feed returns to stability after an interruption rather than waiting for the long window to catch up. Every search and its result are logged.

Tracker arguments

The behaviour is tuned with per-tracker arguments (edit them in the tracker configuration):

Argument	Default	Effect
`min_delay_sec`	`3600`	Minimum current delay (seconds) for an entity to be considered.
`max_auto_delay_sec`	`604800`	Cap on the value the engine may set; a higher computed value is clamped to this (the entity then stays under review).
`max_changes_past_7days`	`10`	Maximum updates per entity in a rolling 7 days; once reached, the entity is left alone until the counter resets.
`min_historical_metrics_day`	`7`	Minimum days of accumulated metrics before the threshold may be updated.
`review_period_no_days`	`30`	Re-review window for already-managed entities (`7`, `15` or `30`).
`max_sla_percentage`	`100`	Skip entities whose current SLA % is at or above this — protecting a stable entity hit by a genuine one-off anomaly from being widened (defaults effectively to 90%).

Automatic updates and audit

When the command updates a threshold it writes an audit record (context: *automated adaptive delay update*). Review them with:

```
`trackme_audit_idx` tenant_id=* "automated adaptive delay update"
| table _time, tenant_id, object_category, object, action, comment
| sort - 0 _time | trackmeprettyjson fields=comment
```

A dedicated dashboard, **TrackMe - Adaptive delay threshold audit**, is available under **Audit & Troubleshoot**.

Automated summary notes

When the framework updates an entity's threshold, it also attaches a **Markdown summary note** to that entity (author `trackmesplkadaptivedelay`), making the change self-documenting. The note records the change **direction**, the **previous and new** values (as a duration and in seconds, e.g. *1h (3600s) → 4h (14400s)*), a short **reason**, and a **reasoning table** (95th-percentile, max and standard deviation of the observed lag plus the ML upper bound, per window), and flags when the value was capped at `max_auto_delay_sec`.

Find it under **Tenant Home → the entity → Notes**, or in the entity's **Audit changes** tab. Notes are controlled per tenant by the `adaptive_delay_notes` setting (*Feature Behavior*, enabled by default); a note is created only when a threshold actually changes, and note creation never blocks the update itself.

Activity logs

The tracker logs every entity it considered and the action taken:

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkadaptivedelay␣
↪object="myobject"
```

Opting an entity or tenant out

To stop adaptive delay managing a **single entity**, **lock** its threshold (see *Who manages a threshold: the Threshold Lock and the AI Advisor*) — the lock pins your value and excludes the entity from automatic management. To disable adaptive delay for a **whole tenant**, turn off **Adaptive delay (auto-tune thresholds)** (`adaptive_delay`) in the tenant menu → **Configure tenant** → **Feeds & delays** (see *Configuring a tenant*). The toggle is locked while the AI Feed Lifecycle Advisor manages DSM/DHM on the tenant.

Disabling it at the tenant level also takes the backend off the Splunk scheduler: the tenant's health tracker **unschedules the adaptive-delay tracker** on its next cycle (and the tracker short-circuits at runtime in any case). The only effect is that thresholds are no longer auto-reviewed — they simply hold their current values, or whatever your policies and manual settings define. Adaptive delay is a convenience feature, valuable but entirely optional, which makes it the classic first lever in a *footprint review*.

Variable adaptive delay

Adaptive delay is no longer limited to the static policy — for an entity on the *variable delay* policy, the same `trackmesplkadaptivedelay` tracker (same `adaptive_delay` tenant setting, same per-entity management) also keeps the **slot thresholds** current. It **recomputes each slot's** `max_delay_allowed` **and the fallback** `variable_delay_default` from history, while **preserving your slot layout** — the slot names, days and hours you defined are kept; only the threshold values are refreshed.

This is different from the variable-delay *auto-review* reviewer (`trackmesplkvariabledelayreview`), which *regenerates* the slot layout. The two are independently opt-in and can both apply to a variable entity:

Mechanism	Opt-in	What it changes
Variable adaptive delay	`adaptive_delay` + entity managed (unlocked)	Refreshes per-slot thresholds + the default; **keeps your slot names/days/hours**.
Variable-delay auto-review	`variable_delay_auto_review_ena`	**Regenerates** the slot layout (`auto_slot_N`) when patterns shift.

A **locked** entity (*Threshold Lock*) is skipped by both, and its pinned slot schedule is restored if it ever drifts.

See also

- *Delay & latency thresholds* — manual thresholds and the Threshold Lock.
- *Variable delay* — time-aware thresholds.
- *Artificial Intelligence in TrackMe* — the AI Feed Lifecycle Advisor that supersedes this when enabled.

Variable delay

Variable delay lets an entity's delay threshold change by **day of the week** and **hour of the day**, instead of using one value around the clock. It solves a common problem: feeds have natural rhythms — a business application produces data through weekday office hours but little overnight or at weekends. A single static threshold must be loose enough for the quiet periods, which makes it too forgiving when the feed is busy. Variable delay fixes this by letting you define **named time slots**, each with its own threshold.

Variable delay applies to **splk-dsm** and **splk-dhm**.

Static vs variable delay

Aspect	Static (default)	Variable
Threshold	One fixed value at all times	Multiple values that change by day-of-week and hour-of-day
Configuration	A single number per entity	Named slots (days, hours, threshold) + a fallback default
Quiet periods	Must keep the value loose, reducing sensitivity when busy	Relax overnight/weekends, tighten during business hours
Anomaly reason	`delay_threshold_breached`	`variable_delay_threshold_breached`
Entity field	`variable_delay_policy="static"`	`variable_delay_policy="variable"`

Setting the tenant default

The **default** delay policy for new entities is set per tenant, separately for DSM and DHM — both at tenant creation (static, or variable with slot definitions) and afterwards from tenant management.

Setting variable delay on an entity

Per entity, variable delay is configured from the entity's lagging policy / variable-delay modal.

How slots work

Each slot specifies a **name**, the **days** it covers (`0` = Monday … `6` = Sunday), the **hours** it covers (`0–23`), and the **max delay allowed** (seconds). Evaluation is **first match wins**: slots are checked in order, the first whose day and hour match the current time is used, and if none match, the **fallback default** threshold applies.

```
{
  "slots": [
    {
      "slot_name": "business_hours",
      "days": [0, 1, 2, 3, 4],
      "hours": [8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19],
      "max_delay_allowed": 3600
    },
    {
      "slot_name": "weekends",
      "days": [5, 6],
```

(continues on next page)

(continued from previous page)

```
      "hours": [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,␣
↪19, 20, 21, 22, 23],
      "max_delay_allowed": 86400
    }
  ]
}
```

Here weekday business hours allow 1 hour of delay, weekends allow 24 hours, and weekday nights fall through to the default. The UI ships **presets** — *business hours*, *weekday/weekend*, and a three-tier *business hours / nights / weekends* — for quick setup.

Auto-computation from history

TrackMe can propose slots from an entity's own history, using `trackme.splk.feeds.lag_event_sec`: it buckets 5-minute metrics by day-of-week × hour-of-day, applies a statistical function (`perc95` by default, or `perc99`), rounds each window up to the next hour plus a one-hour buffer (minimum 3600 s), groups adjacent hours with the same value into named slots, and applies a safety cap (default 604800 s = 7 days).

Auto-review

An optional review process keeps slots current as patterns shift. A scheduled report per tenant per component (DSM, DHM) runs daily — `trackmesplkvariabledelayreview` — recomputing thresholds from history and updating slots **only** when the largest deviation exceeds a configurable threshold (default **20%**). It can be enabled at the tenant level and per entity.

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkvariabledelayreview
```

Note

The periodic reviewer regenerates slot layouts (`auto_slot_N`); the operator "honour" strategy and the adaptive-delay branch preserve your slot names.

Important

When the AI **Feed Lifecycle Advisor** covers DSM/DHM, variable-delay **auto-review** is disabled automatically (and does not auto-re-enable when the advisor is turned off). See *Who manages a threshold: the Threshold Lock and the AI Advisor*.

Interactions with other features

- **Adaptive delay.** Variable delay and *adaptive delay* are no longer mutually exclusive — adaptive delay also keeps variable-delay slot thresholds refreshed. Whether either mechanism may touch a given entity is governed by the *Threshold Lock*.
- **Lagging classes.** A *lagging class* can itself carry a variable-delay schedule; and a class's **latency** override still applies independently of the delay policy.
- **Monitoring time policy.** Independent — the monitoring policy controls *whether* an entity is monitored; variable delay controls *which threshold* applies when it is.
- **Bulk edit.** Variable delay can be applied to many entities at once from the bulk-edit actions.
- **Upgrades.** Existing entities default to `variable_delay_policy="static"` with no variable-delay record, so behaviour is unchanged until you opt in.

See also

- *Delay & latency thresholds* — static thresholds and the Threshold Lock.
- *Adaptive delay (ML)* — ML-maintained delay thresholds.
- *Lagging classes* — rule-based thresholds, which can carry slots too.

8.3.5 Expected sources & hosts — in depth

Tip

This is the **in-depth reference** for injecting *expected* data sources (DSM) and hosts (DHM) so TrackMe alerts on entities that have **stopped** sending — even ones it never auto-discovered. For the component basics, see *Data source monitoring* and *Data host monitoring*. This page covers **coverage gap analysis**, the guided injection wizard, the `trackmepushdatasource` command, entity-key formats, and the recurring-import mechanics.

Inject expected data sources or hosts from a Splunk lookup — with a guided wizard, simulation preview, and optional recurring schedule

- This guide walks you through the **Inject Expected Sources / Hosts** wizard, available directly from the Tenant Home actions menu.
- Use this feature to manually insert entities into TrackMe that have not yet been discovered, based on a CMDB, asset inventory, or any Splunk lookup table.
- When injected, these entities are added as if they were discovered by TrackMe under normal circumstances.
- If these entities are not yet sending data to Splunk, they will appear in a **red** state. As soon as real data arrives and the trackers run, the entities will be updated and their status will reflect the actual health of the data.
- The wizard supports both **one-shot injection** and **recurring scheduled imports**, so new entries added to the lookup over time are automatically picked up.
- Supported for **Data Source Monitoring (DSM)** and **Data Host Monitoring (DHM)**.
- This feature requires **TrackMe 2.3.19** or higher.

Coverage gap analysis

Before you inject anything, **coverage gap analysis** tells you *what* is missing. It is a **read-only** reconciliation of a **reference inventory** — a Splunk lookup such as a CMDB or asset table — against the entities TrackMe actually tracks for the tenant, returning the gaps on **both** sides as downloadable CSVs. Open it from the Tenant Home **three-dot menu → Ops**: **Host coverage gap analysis** (DHM) or **Data source coverage gap analysis** (DSM).



It runs as a four-step wizard.

Step 1 — Splunk deployment. Choose where the reference lookup lives: the local instance, or a configured remote account.



Step 2 — Source. Pick the comparison source — a reference **lookup** (CMDB / inventory table) or the result of a live Splunk search — and the wizard previews its content so you can confirm the right table and columns.



Step 3 — Compare options. Decide how a reference row is matched to a tracked entity:

- **Match short ↔ FQDN via asset variations** *(DHM)* — when on, a short hostname matches its FQDN form (and vice-versa); turn it off to compare the exact host value only.
- **Host field mapping** — choose the lookup column that holds the host / endpoint name. Add an **OR condition block** when the name can live in more than one column (for example `ci_name` OR `asset_tag`). *(For DSM you map the index and sourcetype columns instead.)*
- **Filter lookup rows** *(optional)* — restrict the reference with a simple expression: `field=value`, wildcards `*` (any sequence) and `?` (one character), space = implicit AND, `OR` for alternatives, parentheses for grouping; all comparisons are case-insensitive — e.g. `environment=production site=ams* OR site=emea*`.



Step 4 — Analyze. Run the analysis. TrackMe compares the reference rows against the tracked entities and returns three buckets, each previewable and downloadable as CSV:

- **Only in the reference** — entities in your inventory that TrackMe is **not** tracking: the coverage gap to act on.
- **Only in TrackMe** — tracked entities **absent** from the reference (decommissioned, or unknown to the CMDB).

- **In both** — covered entities, with the matched tracked object.



Tip

Gap analysis and injection are a pair: run the analysis to find the **Only in the reference** set, then inject those expected entities (below) to start monitoring them and close the gap.

Injecting expected data hosts (DHM)

In this example, we inject 400 expected hosts from a lookup into a DHM tenant. The lookup contains a `host` field with the hostnames we expect to see in Splunk.

Step 1 — Open the wizard

From the Tenant Home for a DHM tenant, click the **three-dot menu** (top-right) and select **Inject expected hosts** under the **One shot actions** section:



Step 2 — Select the Splunk deployment

The wizard opens. The first step lets you choose the Splunk deployment where the lookup resides. Select **local** for the local instance, or pick a configured **remote account** if the lookup lives on a different Splunk deployment:



Step 3 — Select the lookup and preview its content

Select the lookup transform from the dropdown. The wizard loads the lookup fields and displays a **preview table** showing sample rows, so you can verify the content before proceeding:



Step 4 — Map lookup fields to entity fields

Map the lookup fields to the required entity fields. For DHM, the **host** field is required. The wizard auto-detects matching field names — if your lookup has a column called `host`, it will be pre-selected automatically:



Step 5 — Simulate the injection

Before executing, run a **simulation** to preview exactly what will happen. The simulation checks each row against the existing KV store and reports:

- **Lookup rows** — total rows in the lookup
- **New entities** — entities that will be created (not yet in TrackMe)
- **Already existing** — entities that already exist and will be skipped

A collapsible table shows the full list of new entities to be created, with their object name and host value:



Step 6 — Configure options and execute

The final step lets you optionally enable a **recurring import** before executing. When toggled on, a saved search is created to re-execute the injection on the configured schedule. You can customize both the **report name** and the **cron schedule**:



Click **Inject expected hosts** to execute. The wizard reports the results — entities created, entities skipped, and whether the recurring schedule was created:



Result — Entities visible in Tenant Home

After closing the wizard and refreshing, the 400 injected hosts are now visible in the Tenant Home. Entities that are not yet actively sending data to Splunk appear in a **red** state, which is expected:



Hint

For DHM entities, the index and sourcetype fields will initially appear empty for hosts that have not yet sent data to Splunk. This is normal — once real data arrives and the DHM tracker runs, these fields are automatically populated and the empty placeholders are cleaned up.

Clicking on an entity that has not yet sent data shows the indexes/sourcetypes summary with empty placeholder values. Once the host starts sending data, these values are automatically replaced with the real indexes and sourcetypes:



As soon as trackers are executed and the hosts start sending data to Splunk, the entities will transition to a **green** state and their metrics will be fully populated.

Injecting expected data sources (DSM)

The process for data sources is very similar, with the key difference that DSM requires both an **index** and a **sourcetype** field from the lookup.

Open the wizard (DSM)

From the Tenant Home for a DSM tenant, click the **three-dot menu** and select **Inject expected sources** under the **One shot actions** section:



Select the Splunk deployment (DSM)

Choose the Splunk deployment where the lookup resides:



Select the lookup and preview (DSM)

Select the lookup transform. The preview table shows the index and sourcetype columns from the lookup:



Map lookup fields (DSM)

For DSM, both **index** and **sourcetype** mappings are required. The wizard auto-detects matching field names:



Simulate the injection (DSM)

Run the simulation to preview the results. In this example, 12 new data source entities will be created:



Configure options and execute (DSM)

Enable the recurring import if desired, customize the report name and cron schedule, then click **Inject expected sources**:



The wizard confirms the injection was successful and the recurring schedule was created:



Entities visible in Tenant Home (DSM)

The 12 injected data sources are now visible in the Tenant Home. Entities not yet covered by the tracker scope appear in red:



As soon as the trackers are executed and the data sources become active within their scope, the entities transition to a **green** state:



Recurring imports

When you enable **Set as recurring import** in the wizard, a Splunk saved search is created with the specified name and cron schedule. This saved search:

- Runs on behalf of the **tenant owner** (automatically re-assigned)
- Reads the lookup on each execution and injects any **new** entities not already in TrackMe
- **Skips** entities that already exist — no duplicates, no overwrites
- For recurring runs where nothing new is found, **no audit event** is recorded and no searches are executed (the pre-filter detects that all entities already exist and short-circuits)

You can also create a recurring schedule **after** the initial injection by toggling the recurring import switch and clicking **Create schedule** in the footer.

The recurring schedule can be managed from Splunk's **Settings > Searches, Reports and Alerts** page. Look for reports with the naming convention `TrackMe - Inject Expected {Sources|Hosts} - {tenant_id} - {lookup_name}`:



How it works under the hood

The inject wizard leverages the `trackmepushdatasource` streaming command, which is the same mechanism used by the TrackMe push API. This ensures injected entities are created with all required fields properly initialized — delay/latency statistics, timestamps, and other metadata that the decision maker expects.

For **DSM**, each lookup row with an index and sourcetype produces a data source entity with the format `{index}:{sourcetype}`.

For **DHM**, each lookup row with a host value produces a host entity with the format `key:host|{hostname}`. The raw hostname is stored as the entity's **alias** for display purposes.

Key behaviors

- **Existing entities are never overwritten.** The wizard checks against the KV store before injection and skips any entity that already exists.
- **Empty placeholder cleanup for DHM.** Injected hosts start with empty index/sourcetype values. Once real data is detected by the DHM tracker, these empty placeholders are automatically cleaned up — no manual intervention required.
- **Batching for large lookups.** Lookups with more than 500 entities are automatically processed in batches to stay within Splunk's SPL query length limits.
- **Remote deployment support.** The lookup can reside on a different Splunk deployment. Select the configured remote account in the first step of the wizard.

Troubleshooting

If you encounter issues with the injection, check the following logs:

Inject expected REST handler logs:

```
index=_internal sourcetype=trackme:rest_api source=
↪"*trackme_rest_api_splk_inject_expected*"
```

trackmepushdatasource command logs:

```
index=_internal sourcetype=trackme:custom_commands:trackmepushdatasource
```

Advanced usage with the trackmepushdatasource command

The inject wizard uses the `trackmepushdatasource` streaming command under the hood. For advanced use cases, you can call this command directly in SPL.

Command arguments

Argument	Required	Default	Description
tenant_id	Yes	None	The tenant identifier
component	Yes	None	The component to use (`dsm` or `dhm`)
search_type	Yes	None	The type of search to perform (`tstats` or `raw`)
show_search_query	No	False	If true, includes the search query in the summary output
show_search_results	No	False	If true, includes the search results in the summary output
pretend_latest	No	-24h	Relative time value in Splunk format for data_last_time_seen

Example — Pushing expected data sources via SPL

```
| inputlookup example_expected_data_sources.csv
| trackmepushdatasource search_type=tstats tenant_id=demo show_search_query=True␣
↪show_search_results=True pretend_latest="-24h" component="dsm"
```

Example — Pushing expected hosts via SPL

```
| inputlookup example_expected_hosts.csv
| trackmepushdatasource search_type=tstats tenant_id=demo show_search_query=True␣
↪show_search_results=True pretend_latest="-24h" component="dhm"
```

Controlling the break-by logic for DSM

For expected data sources, if the TrackMe tracker logic includes a specific break-by logic, you can submit the value for the object accordingly:

```
| inputlookup ds_expected.csv
| fields index, sourcetype, cribl_env
| eval object = index . ":" . sourcetype . ":" . "|key:cribl_env|" . cribl_env
| trackmepushdatasource search_type=tstats tenant_id=secops show_search_query=True␣
↪show_search_results=True pretend_latest="-24h" component="dsm"
```

Controlling the host metadata key for DHM

For expected hosts, you can control the metadata key for the host:

```
| inputlookup dh_expected.csv
| fields host
| eval host = "key:forwarder|" . host
| trackmepushdatasource search_type=tstats tenant_id=endpoints␣
```

(continues on next page)

(continued from previous page)

```
↪show_search_query=True show_search_results=True pretend_latest="-24h" component=
↪"dhm"
```

8.3.6 MHM — Metric Host Monitoring

Metric Host Monitoring is the metric-index counterpart of *DHM*. It tracks **hosts sending metric data**, grouped by metric category, answering "are my metrics still arriving for every host that should be emitting them?"



What MHM tracks

Each entity is a **host × metric category** pair (for example `os.cpu` or `os.mem`). The host is the group; its categories inherit the host's state. MHM watches:

- **Metric lag per category** — the freshness of each category.
- **Inactivity** — a host or category that has stopped sending.
- **Missing expected categories** — a host no longer emitting a category it used to.

MHM is scoped to a **metrics index** and queries it with `mstats`. The creation wizard enumerates the hosts in that index, groups their metric categories, and lets you choose which to track. A host's **Overview** tab summarises its lag, impact score and SLA, and the **metrics availability table** lists every metric category with its last-seen time, lag, and state:



How MHM differs

Important

MHM is deliberately simpler than DSM/DHM:

- **Static thresholds only** — no variable delay, no lagging classes.
- **No data sampling, no elastic sources.**
- **Outlier detection is not available** — unlike the other components, MHM does not support the native ML outlier engine at all (see *Outlier detection*).

Tip

Many infrastructure metrics report on an irregular cadence, which produces "metric lag" false positives. If a host legitimately emits a category intermittently, raise its lag threshold or disable that entity.

See also

- *DHM — Data Host Monitoring* — the event-data host-centric component.
- *Entity State & Scoring* — thresholds and scoring.
- *Classification & Protection* — priority, tags, and SLA.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.7 VOL — Volume Outliers

Volume Outliers tracks the **licensed volume of every Splunk index** — the volume you pay for — and answers a question the other components leave open: **is my license volume normal?** Each entity is one **index**; its state comes from **ML outlier detection** on the rolling volume (abnormal drops and / or spikes) and from **inactivity** (an index

that silently stops consuming license). On top of the detection, VOL records the **daily licensed volume** per license day, a **30-day trend** and a **month projection**, per index and for the whole environment.

Where DSM watches feeds arrive on time, VOL watches *how much* arrives, from one source that is authoritative, cheap to read and identical on every Splunk deployment: the **license usage log**. One tenant covers a whole Splunk deployment — local or remote, Splunk Enterprise or Splunk Cloud — in a few clicks, with the history **backfilled automatically**: detection and the cost trend are usable as soon as the replay completes, not weeks later.

Volume detection has always been part of TrackMe: per feed with the DSM volume models, per index with the Flex Objects license usage templates. VOL is its **standardised, first-class form at scale**: one entity per index, the licensed volume read incrementally, outliers on drops and spikes, and a purpose-built UI. It is the seventh monitoring component, next to DSM, DHM, MHM, FLX, FQM and WLK, and on top of the detection it answers the license questions people otherwise build dashboards for: who consumes, how the trend goes, how the month will end, which days went over quota, what a working day costs and which team pays for what.





What VOL tracks

Signal	What it means
Volume outliers	Each index has its own ML model on its **rolling 24-hour licensed volume** (the shorter windows — 60 min, 4 h, 12 h — can be modelled too). A value outside the learned range is a **drop** (lower bound) or a **spike** (upper bound). The **detection direction** — drops, spikes, or both — is chosen per index; the tenant sets the default. Outliers are the *state* of the component, not an option.
Inactivity	An index whose last licensed volume is older than the threshold in force turns **red**. The threshold is a tenant policy — static, or **variable** by day of week and hour of day (by default 1 day during working hours, 7 days for nights and week-ends) — and can be overridden per index.
Daily licensed volume	The volume of each **license day** (the day rolls at the license manager's local midnight, whatever your search head's time zone), the figure your license is measured on. Reported per index and summed for the environment.
Trend and projection	A 30-day trend, the month-to-date volume and a **month projection** that follows the weekday pattern of the index — recomputed every 5 minutes, per index and for the whole environment. Both are built on medians, so a one-off spike or outage does not dominate them.
Pool usage	On Splunk Enterprise, the share of the license pool consumed over the last 24 hours; the entity's *group* is its license pool. On Splunk Cloud a pool has no limit, so no pool usage is reported.

Every index is a regular TrackMe entity: priority, tags, labels, SLA, logical groups, the disruption queue, acknowledgments, notes, maintenance, stateful alerting and the AI Assistant all apply. VOL has **no thresholds** to configure and no delay / latency — there is nothing to tune before it delivers value.

How it collects

One tracker per tenant runs every 5 minutes and reads **only the new events** of the license usage log — one bounded search, whatever the number of indexes — into **5-minute buckets** per index. The rolling KPIs, the daily volumes and the projection are derived from those buckets, never by rescanning `_internal`. The tracker writes the `trackme.splk.vol.*` metrics with explicit bucket timestamps, so the charts and the ML models see a continuous, regular series.

At creation the tenant **backfills** its history: 30 days by default, a choice from 7 to 90 days, or **Custom…** for any number of days up to **400** (13 calendar months, enough for a year of chargeback statements), always bounded by the retention of the license usage log. The backfill replays the same collection over the past, one license day at a time, and writes the daily licensed volume of every day it covers, so the license view, the daily chart and the projection are complete when it finishes; the outliers models train as soon as it completes. Progress is visible on the tenant card, in the Tenant Home header and in the *Manage: volume collection* screen. See *Backfill* for the details.

Global license usage

Next to the VOL component tab, Tenant Home offers the **Global license usage (VOL)** tab: the license consumption of the whole environment, the dashboards people otherwise build by hand. It reads what the tenant already collected, so it adds no search on the license usage log, raises no alert and works the same on a remote deployment. It covers closed license days only, over a period of 30, 60 or 90 days, and every tenant user can open it.

The view has six sub-tabs:

Sub-tab	What it answers
Overview	The key figures (licensed yesterday, daily average and working-day average, peak day, trend, month to date, month projection, top 5 share), insights in plain words, the daily volume stacked by top consumers with the quota line, the whole environment per day with its projection, the consumers table, the week-over-week movers and the license pools. With costs on, the cost of the period, the cost per GB and the unused capacity; with a quota, the days over quota.
Scopes	One tab per **named scope**: a set of indexes analysed together, such as a team, a use case or the indexes of Enterprise Security. A scope is defined by index patterns or follows the CIM datamodel index macros. Each tab shows the scope's share of the license (or its cost), its daily average, its peak day, the last 7 days against the 7 before, its month projection and its indexes.
Working days & holidays	Weekends and public holidays pull a daily average down. This tab shows the working-day, weekend and holiday averages, the average per weekday, and the top consumers and pools on a working day. Holidays come from the *Bank Holidays* calendars.
Over quota	The days a license pool went over its daily quota, over 3, 6 or 12 months: per day against the quota, per calendar month and over a rolling 60 days. **What made the day** shows, for any day you pick, the pool's largest indexes against their usual day.
Chargeback	A monthly statement of the license cost per **cost centre**, over 13 calendar months: the bill of the month, what each centre is charged, its budget, the unallocated volume, the unused capacity, a 13-month trend and a CSV export.
Configuration	The scopes, the working days, a daily quota per pool, the cost centres and the costs with their price history, saved together.

Everything beyond the overview is **opt-in**: nothing is priced, scoped or charged until a power or admin user of the tenant configures it. The **AI Assistant** of Tenant Home switches to this view while it is open: it knows the sub-tab and the figures on screen, and with your consent it can change the configuration for you through the Concierge.





Creating a VOL tenant

1. **Create a tracking tenant** and pick the **Splunk Volume (Outliers)** card.
2. **Basics** — the tenant identifier, alias and description.
3. **Volume** — the target environment (this Splunk, or a remote account), the default **detection direction**, the **backfill history** (up to 400 days with **Custom…**), which priorities are in scope for outliers, the minimum history a model needs before its confidence is *normal*, and the default **outliers KPIs**. *Advanced* holds the license search constraint and the settle margin.
4. **Inactivity** — the tenant's inactivity policy (static, or variable by day and hour).
5. **Indexes & RBAC** — the TrackMe indexes, the roles and the owner, as for every tenant.
6. **Review** and create. The first collection runs at once; the backfill follows on its own schedule.

The wizard is walked step by step in *Creating a splk-vol tenant*, and end to end — from creation to the first alert — in the *Volume Outliers white paper*.

Tip

Foundation customers get it too. VOL is part of the **Foundation** edition, next to DSM, DHM and MHM — a Foundation deployment can monitor its license volume out of the box, without Flex Objects. Enterprise and Unlimited include it as well.

Note

Replaces the Flex Objects license-usage recipes. Before 2.4.18, volume outliers on the license usage were delivered with Flex Objects templates (`splk_license_usage_per_index_*`) and, per feed, with the DSM volume models. VOL packages that recipe as a first-class component: cheaper to run, backfilled, with a purpose-built UI. The templates remain available for custom needs.

See also

- *Volume Outliers — in depth* — the collector, license days and time zones, the backfill, outliers and inactivity policies, the projection, the Global license usage tab, alerting, REST and metrics.
- *Volume outliers on the Splunk license usage* — the white paper: an end-to-end walkthrough on a live deployment.
- *Machine Learning* — how outlier detection works.
- *Entity State & Scoring* — states and scoring.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.8 Volume Outliers — in depth

Tip

This is the **in-depth reference** for the Volume Outliers (splk-vol) component. For the readable overview of what VOL is and when to use it, start with *VOL — Volume Outliers*; for a guided, end-to-end walkthrough on a live deployment, read the *Volume Outliers white paper*. This page covers the collector, license days and time zones, the closed-day reconciliation, the backfill, the outliers and inactivity policies, the projection, the Global license usage view and its configuration (scopes, working days, daily quotas, costs, chargeback), alerting, the REST endpoints, the metrics and troubleshooting.

The entity model

- An entity is one Splunk **index**: `object` is the index name, `object_category` is `splk-vol`, and the entity's **group** is its license **pool** (a single-pool deployment has one group; on Splunk Cloud, where a pool has no limit, the group is `default`). An index reported under several pools follows its **primary pool** — the one carrying most of its bytes in the window — with hysteresis: it keeps that pool up to 24 hours while the pool reports it, and moves only when another pool carries twice its volume.
- The state is **green by default**: collection never turns an index orange or red on its own. The state comes from **ML outliers** (a lower-bound breach is a *drop*, an upper-bound breach a *spike*) and from **inactivity**, through the same hybrid impact scoring as every component (*Entity State & Scoring*). The tenant impact keys are `impact_score_vol_outliers` and `impact_score_vol_inactive` (default 100: an inactive index turns red).
- Per index the analyst chooses the **detection direction** — *Detect drops*, *Detect spikes*, or both — with the inline **Detect** toggle of the VOL table, in bulk edit, or in the entity's *Volume & cost* tab. The direction is stored on the index's outliers model (`alert_lower_breached` / `alert_upper_breached`); the tenant default is `vol_default_direction` (`both`).
- Everything generic applies: priority, tags, labels, SLA, logical groups, disruption queue, acknowledgments, notes, maintenance, monitoring time, enable / disable, deletion. An index deleted *temporarily* keeps its bucket history when it had licensed volume in its last 24 hours (it comes back with its true rolling volume); a *permanent* deletion drops it and the index is never collected again.

The collector

One report per tenant, `trackme_vol_tracker_tenant_<tenant_id>`, runs every 5 minutes through the standard tracker executor (execution summary, licensing checks and the scheduled-time anchor come with it). Each run:

1. **Loads the checkpoint** of the tenant — the `cursor` (the end of the last finalized 5-minute bucket), the validated history start, the license manager's UTC offset and the deployment type.
2. **Establishes the license writer** — the one host writing `type=Usage` over the last 3 days — and scopes every license search to it. No writer, or several (two license managers in one license search, a test stack

forwarding its log to the same index), holds the collection with the `VOL_LICENSE_WRITER_UNKNOWN` / `VOL_MULTIPLE_LICENSE_WRITERS` diagnostic rather than summing licences: the cursor does not move, the history is kept, and the license view and the *Manage: volume collection* screen say so. Scope `vol_license_search` with `host=<license manager>` to resolve it; after a license manager is replaced, collection resumes by itself once the previous host has written no `Usage` for 3 days.

3. **Computes the window** `[cursor, floor(min(now, watermark) - settle))`, aligned on 5-minute boundaries. The **source watermark** (2.4.22) is how far the writer has delivered: the newest `_internal` event of the writer host, which travels the same forwarding path as the license usage log (for a custom license search, the earlier of that and the newest license event). The window waits for a late source instead of closing on events that have not arrived; the **settle margin** (`vol_settle_seconds`, default 300 s) still leaves the time to write every `Usage` event of a bucket before it is read. A run catches up at most 6 hours at a time; a gap above 24 hours re-seeds the history (see *Validated history* below).

4. **Runs one bounded search** on the license usage log, local or on the tenant's remote account, starting `vol_late_overlap_seconds` (default 1800 s) before the cursor so stragglers of the previous buckets are read again:

```
search (index=_internal source="*/license_usage.log" host=<writer>) type=Usage␣
↪idx=* earliest=<cursor - overlap> latest=<end>
| bin _time span=300s
| eval effective_bytes=b*coalesce(licensed_weight,1)
| stats sum(b) as b, sum(effective_bytes) as effective_b, latest(poolsz) as␣
↪poolsz by _time, idx, pool, host
```

   plus, once a day (and on every run around the license manager's midnight), a cheap indexed probe on the `RolloverSummary` event that returns the license manager's UTC offset and the deployment type. `b` is the **raw ingest**; `effective_b` the **licensed volume** — data Splunk licenses at a reduced weight carries a `licensed_weight` on its `Usage` rows (2.4.22). The 5-minute buckets and the rolling values the outliers models train on stay raw ingest; the daily figures (yesterday, month to date, trend, projection and the whole *Global license usage* view) are licensed volume, and the raw ingest is shown next to *Licensed yesterday* when the two differ.

5. **Updates the ring** of each index — the last 288 buckets (24 hours) kept in the tenant's KV store: appends the new buckets (a bucket with no event is 0), recomputes the rolling sums at each bucket end, accumulates the current license day and closes any license day that ended. A bucket read again through the overlap is **raised** to the re-read total, never lowered (license events are never removed, so a re-read cannot take volume away); the difference goes to the ring and to the open or closed day it belongs to, and the 5-minute and rolling points already written are not written twice.

6. **Writes the metrics** with explicit bucket timestamps, then advances the cursor — the cursor moves only once the metrics and the rings are persisted, so a run that dies mid-way is replayed, never skipped.

7. **Emits one entity row per known index**, carrying the latest rolling values, the daily figures (yesterday, month to date, 30-day average, trend, projection), the pool usage on Splunk Enterprise, the detection direction, the effective inactivity policy and the **hold** the run measured on the source (see *Late license events* below).

8. **Reconciles closed days** — after 03:00 license-manager time, up to two of the last seven closed license days are reread from the license usage log, repaired if needed and compared with the license manager's own daily summary (see *Closed-day reconciliation*).

Note

Why a ring instead of querying the metrics index. The rolling volume must not depend on the metrics pipeline's own indexing latency: a blocked queue would read as a drop and raise false lower-bound outliers. The rolling values are computed from the buckets TrackMe holds and *then* written as metrics.

Integrity rules

- **Validated history.** Every ring and the tenant checkpoint carry the time from which the history is continuous. A rolling KPI is emitted only once its whole window lies inside the validated history, and a license day is reported only when it was observed from its start — the first run's 24-hour seed writes the 5-minute volumes for every bucket, but the 24-hour trend only from the last seed bucket. No ramp-up or gap ever reaches the series the models train on.
- **Search errors fail the run.** An error from the usage search — or a warning saying the results are incomplete (a peer down, a search process ended prematurely) — stops the run before anything is written; the cursor does not advance and the missing volume is never recorded as a drop.
- **One collecting run at a time.** A lock protects the collection; an overlapping run (for instance *run the tracker now* during the schedule) rebuilds the entity rows from the stored rings without writing any metric point. A run that meets the lock of a tenant being seeded waits for it (up to 4 minutes) and then collects normally.
- **A new source re-seeds.** Changing the target environment (`vol_account`) or the license search constraint discards the rings and the checkpoint and re-seeds 24 hours from the new source, emitting only the points *after* the previous source's cursor — a series never holds two points for the same bucket. The previous source's indexes stop being updated, turn inactive and can be deleted.
- **Fail-closed reads.** The rings, the per-index directions and the permanently deleted indexes are read without a fallback: a failed read fails the run instead of resetting a direction or resurrecting a deleted index.
- **Squashed rows keep the index.** When the license manager squashes `Usage` rows (`squash_threshold`), it drops the host and source, never `idx` — squashing never reads as a drop.
- **One license writer.** Every search is scoped to the host established as the `Usage` writer; a held collection (no writer, several writers) never sums volumes and never moves the cursor.

Late license events

The license usage log reaches the indexers through the license manager's own forwarding. A blocked output queue on the license manager, an indexer restart or a license manager clock behind the indexers used to turn into a sudden, lasting drop on **every** index at once while the events were all there, just late. Since 2.4.22 three layers keep the figures exact whatever the latency:

- **The window waits for the source** — it ends at the writer's watermark (step 3 above), so on an in-order forwarding path every bucket the collector reads is complete.
- **The late overlap** — each run rereads the last `vol_late_overlap_seconds` before its cursor (default 30 minutes, 0 disables, up to 4 hours) and raises the buckets it finds short, never lowers one.
- **The closed-day reconciliation** — a straggler that arrives after its license day closed is repaired in the day's figures by the reconciliation (next section).

A window held 15 minutes or more behind the wall clock raises the `VOL_LICENSE_SOURCE_DELAYED` diagnostic in the license view: the collection is waiting for the events instead of recording missing volume — check the license manager's forwarding to the indexers and its clock. The hold each run measures is written on every index and **discounted from the inactivity clock**, up to 4 hours, so a late source does not turn every index red; a source delivering nothing for more than 4 hours is an outage and does. The measure is the run's, not "now minus the cursor": a tracker that stops running keeps its last measure and the inactivity clock keeps running.

License days and time zones

Splunk measures the license on **license days** that roll at the **license manager's local midnight**, whatever the time zone of your search heads or of your Splunk user. VOL follows the license manager:

- The daily probe reads the license manager's UTC offset from the `RolloverSummary` event (its raw timestamp carries the offset, and its time is the true day boundary). On a deployment younger than a license day no rollover has been logged yet: the offset is then read from the raw timestamp of the latest `Usage` event (every line of the license usage log carries it) and the *Manage: volume collection* screen says so. UTC is assumed only when there is no license event at all, and the open day is voided when the real offset turns out to differ.
- The probe runs on every run between 22:00 and 03:00 license-manager time, so a **daylight-saving transition** is caught within the run that follows it. The buckets between the old and the new day boundary move to the day they belong to, and a corrected closed day is re-emitted (a fall-back day lasts 25 hours, a spring-forward day 23 hours).
- The `daily_volume_mb` metric of a license day is stamped in the middle of that day, so a chart binned by day places it on the right date whatever the viewer's time zone.

The **daily figures** (`yesterday`, `month to date`, the trend and the projection) refer to license days, and the *Volume & cost* tab says which license date "today" is for the collector.

Closed-day reconciliation

The live collection reads each 5-minute bucket once, shortly after it ends. A `Usage` event that reaches the indexers later than the settle margin (a blocked forwarding queue on the license manager, an indexer restart) would be missing from the day's figure, and Splunk's own daily summary — the `RolloverSummary` event the license manager writes at its midnight — can itself differ from the sum of the detailed `Usage` records. Since 2.4.22 the tracker therefore **reconciles every closed license day** against the license usage log and says what it found.

How it runs

- The reconciliation is a phase of the 5-minute live tracker, under the collection lock: a run that collects (one `Usage` writer identified, no diagnostic hold) checks at most **two closed days** among the **seven most recent** the tenant holds, in due-time order, so a day that keeps failing never starves the others.
- A closed day becomes due at **03:00 in the license manager's time zone** of the following day, leaving late events and the rollover itself the time to be indexed. A verified day is checked again every day at 03:00 while it stays in the seven-day window (a late redistribution between indexes is caught); a day that could not be read, or whose totals disagree, is retried **every hour**.
- Each attempt reads the day's `RolloverSummary` (the day's exact bounds come from the two adjacent rollovers and their UTC offsets, so a 23-hour or 25-hour day is read as such), then rereads the **whole day per index** from the `Usage` records — the same source and the same remote account as the live collection, under the tenant's license search constraint.
- A reread that answers **repairs the stored figures** of the day: the raw ingest and the licensed volume of every index of that day, including an index that stopped consuming (its day is zeroed, never left at a stale value) — never an index first seen on a later day, never a permanently deleted index. The repair happens whether or not the rollover agrees: the detailed `Usage` records are the measurement.
- A repaired day is re-emitted: one corrected daily point per index (`daily_volume_mb`, `daily_ingest_mb`), stamped later than the original point inside the same license day and carrying the `license_day` and `reconciliation_revision` dimensions (`r1`, `r2`…). The entity searches read the daily series with `latest()`, so a corrected point replaces the original one in every chart and figure without summing the two; the stored daily figures the license view reads carry the same snapshot.
- Nothing is invented: when the rollover or the `Usage` records of the day are no longer available (retention), the day boundaries cannot be established, or the search fails, the previous figures are kept untouched and the attempt is retried an hour later. A day that leaves the seven-day window without a successful reread keeps its figures and is marked as outside the automatic window; it is never set to zero.

Verified, or reconciled with a difference

Repairing the figures and agreeing with the license manager are two different things. Once the day is reread, its raw and licensed totals are compared with the `RolloverSummary` totals to the byte (8 bytes of tolerance, independent of the displayed rounding):

- **Verified —** the `Usage` records and the rollover agree (*Verified for collected scope* when permanently deleted indexes are excluded from the comparison).
- **Source reconciled · summary differs —** TrackMe matches the detailed `Usage` records, but Splunk's own `RolloverSummary` reports more or less than those records. The card gives both totals, the difference in MiB and in percent of the summary, and the next comparison check. This is information, not an alert: TrackMe's reconciliation is complete, the figure is the one the `Usage` records measure, and the comparison keeps running while the day stays in the window.
- **Reconciliation incomplete —** the day could not be reread (the reason and the last attempt are on the card) and keeps its previous figures; the retry is automatic.

Where to see it

About this deployment — the button at the top of the *Global license usage* tab, and the same card on the *Manage: volume collection* screen — leads with the verification state: *N closed license days verified*, *Daily figures successfully reconciled* with one line per day whose summary differs, or *Collection or verification needs attention* with the days to look at. *Technical details and verification dates* lists every day of the window with its status, revision, last check, both byte totals and the next check. The cards of the *Global license usage* views whose figure comes from the license usage log offer **Verify against license_usage.log** in their menu: the search that sums the `Usage` records of that exact day and scope, so you can confirm the figure yourself.


About this deployment
LICENSE VERIFICATION
Daily figures successfully reconciled
TrackMe's daily totals and per-index figures match the detailed Usage records in license_usage.log.
2026-10-24 · Splunk's own records disagree: its RolloverSummary reports 2.18 MiB more (0.033%) than the detailed Usage records for this day.
2026-10-28 · Splunk's own records disagree: its RolloverSummary reports 1.74 MiB more (0.026%) than the detailed Usage records for this day.
2026-10-30 · Splunk's own records disagree: its RolloverSummary reports 1.45 MiB more (0.016%) than the detailed Usage records for this day.
2026-11-02 · Splunk's own records disagree: its RolloverSummary reports 0.74 MiB more (0.009%) than the detailed Usage records for this day.
TrackMe's reconciliation is complete for the listed days. Automatic checks will continue to monitor the difference between Splunk's records.
2026-10-21 → 2026-11-03 · matched against the license manager
Deployment
Platform
Splunk Enterprise
Collection
Local
License Usage writer
license-mgr-01
Monitored indexes
61
Collection
Collected through
2026-11-04 07:20:00 UTC
License day
license manager midnight, UTC+0000
Unit labels
IEC · MiB / GiB / TiB
All volume calculations use binary units (1024). Changing labels does not change the values.
Close



Note

After an upgrade to 2.4.22. Nothing to request and nothing to wait for: the closed days are those the tenant already holds, so the first collecting runs after the upgrade reconcile the last seven of them, two per run. There is no manual reconciliation request; **Run tracker now** on the *Manage: volume collection* screen runs the same tracker at once, under the same rules — it cannot pick a day, nor reconcile a day before 03:00 license-manager time the day after it closed.

Backfill

At creation the tenant requests a **backfill** of its history (`vol_backfill_days`, default 30) so the outliers models train as soon as it completes and the daily history, the trend and the month projection are complete. The wizard offers **No backfill**, 7, 14, 30, 60 or 90 days, or **Custom…** for any whole number of days up to **400** — 13 calendar months, the depth of the chargeback statements. It caps the depth to the retention of the license usage log it can reach, shows the reachable span, and shows the estimated cost in metric points before you confirm.

- The backfill is a separate report, `trackme_vol_backfill_tracker_tenant_<tenant_id>`, scheduled **hourly** while a plan is open and un-scheduled once complete. It walks **forward** in time (every rolling value needs the 24 hours before it) in day-sized chunks, each chunk replaying exactly the live collection with historical timestamps, and runs for at most 50 minutes per slot.
- The backfilled range ends where the live collection started: the live tracker and the backfill **never overlap** and no point is written twice. The first backfilled day is aligned on a license-day start, so it is complete; every complete license day of the range emits its `daily_volume_mb`.

- The live collection emits a rolling value only once its whole window lies in its own history, so the **first live day** has no 24-hour value (and no 12-hour, 4-hour or 60-minute one for its first hours). About 24 hours after the live start, the hourly backfill job replays that day and writes the missing rolling values; until then the *Manage: volume collection* screen reads *First live day trends: pending* and the rolling chart shows a gap at the seam.
- Daylight-saving transitions inside the range are replayed with the offset in effect at the time, from the `RolloverSummary` history.
- On completion the backfilled days are merged into each index's live history (the entity figures reflect them at once), the seam day around the live start is completed, and a **training of every VOL model** is requested, so outliers are usable without waiting for the next training cycle.

Progress is visible in three places — the tenant card on the Virtual Tenants page (`Backfilling 42 %`), the vignette in the Tenant Home header band, and the **Manage: volume collection** screen (live cursor, validated history, license-manager offset, deployment type, the backfill progress with **pause / resume / cancel / restart**, and *run the tracker now*). From the creation until the backfill job's first hourly run, the three read **Backfill pending** with the requested depth and the job's next run.

An upgrade to 2.4.22 runs a **Historical license update** through the same job: the reachable source history is reread once to apply the licensed weight to the stored daily volumes (*Updating history* on the vignette, *Historical license update* on the collection screen, with the same pause, resume and cancel). The raw rolling history and the outlier models are untouched; a day whose source has expired keeps its raw value, and the *Adjusted history from* date on the screen says where the licensed history starts.





Note

A **restart** re-plans only the history *before* the earliest point already written: it extends the history backward and never rewrites it. A change of the target environment or of the license search constraint while a plan is open cancels the plan (the new source is re-seeded first); restart it once the live collection runs on the new source.

The license search constraint (advanced)

Every license search of the tracker, the backfill and the probes is formed from one tenant-level constraint, `vol_license_search`. Its default targets the license usage log of the deployment the tenant collects from:

```
index=_internal source="*/license_usage.log"
```

You normally never change it. The **Advanced** section of the wizard's *Volume* step (and the *Configure tenant* form afterwards) lets you point the tenant at another data set holding events in the `license_usage.log` format — for

instance a copy of the license usage events forwarded to a dedicated index, or a deployment where `_internal` is not reachable under that name. The constraint is a plain base search on indexed fields:

- no pipe, subsearch (square brackets), macro (backtick), line break, `earliest` or `latest` — the tracker sets the time range;
- it must name an **index** as a real search term (`index=` inside a quoted value does not count), and not every index: a bare wildcard (`index=*`) is refused, `index=lic*` is fine;
- every `OR` alternative must be index-scoped (`index=a OR index=b` is fine, `index=a OR source=x` is refused) and parentheses must balance.

The wizard validates the constraint as you type, and it is validated again at every save.

Important

The data behind a custom constraint must have the `license_usage.log` key=value fields extracted at search time — `type`, `idx`, `b`, `pool`, `poolsz` — since the tracker filters on `type=Usage idx=*` and sums `b`. The real `license_usage.log` in `_internal` (`sourcetype=splunkd`) is extracted out of the box; a copy stored under a sourcetype with `KV_MODE = none` (`stash`, for instance) needs a `props.conf` stanza with `KV_MODE = auto` for its source, or the tracker reads no usage row at all.

Changing the constraint on a running tenant re-seeds the collection from the new source (see *Integrity rules* above) and cancels an open backfill plan.

Outliers

VOL rides TrackMe's native outliers engine (*Machine Learning*, *Outlier detection — in depth*) with component-specific defaults:

- **Every new index gets one model per default KPI** — `vol_outliers_kpis`, default `last_24h_volume_mb`, any subset of the four rolling KPIs. The rolling 24-hour volume follows the daily pattern and is the proven choice; the shorter windows (12 h, 4 h, 60 min) react faster to a sudden drop or spike and are noisier. One model per index is the deliberate default: several models per index multiply the noise. The models use the day-of-week × hour-of-day time factor over a 90-day calculation period.
- The default KPIs are set in the wizard and changed later in Tenant Home (*Manage: Outliers default KPIs*), with an opt-in **apply to the existing indexes** that adds the models of newly selected KPIs and removes the models of deselected rolling KPIs — models of kept KPIs and of non-rolling metrics are untouched. A model can also be added on, or switched to, any of the four KPIs from the entity's Outliers tab.
- **Direction** — a drop is a lower-bound breach, a spike an upper-bound breach; the per-index detection direction enables one or both bounds on the model.
- **Scope** — VOL has its own scope: `vol_mloutliers_priority_filter` (default: every priority — volume outliers are the point of the component) and `vol_mloutliers_filter_expression` (the TrackMe filter DSL on `object`, `priority`, `tags`, `labels` and `pool`). The tenant-wide `mloutliers_priority_filter` and `mloutliers_filter_expression` do **not** apply to VOL entities. The *Manage: ML Outliers scope* modal edits the VOL pair on a VOL tenant, and the table's Outliers column shows *out of scope* with the reason for an excluded index.
- **Minimum history** — the wizard exposes the tenant override of `splk_outliers_min_days_history` (15 days on a VOL tenant by default, against a system default of 30) and tells you whether the requested backfill covers it: with 30 days of backfill, the models reach *normal* confidence once the replay completes.
- **Reading a model** — the entity's *Volume & cost* tab draws one outliers chart per enabled model, each with its own **History** slider, from 24 hours (the default) to 90 days. The chart reloads once, when you release or leave the slider, and its subtitle gives the number of **anomaly windows** over that range with their points (for instance *5 anomaly windows (36 points)*: consecutive anomalous points form one window, as the chart's legend counts them) and the model's confidence.
- ML is **always on** for VOL. A tenant where ML is off for the other components because it only runs DHM / MHM keeps ML on for VOL alone (`mloutliers_allowlist=vol`).

- The **AI ML Advisor** — interactive and its automated batch — covers VOL models like any other (*AI ML Advisor*).

Note

A one-off spike right after a short backfill. The models use the day-of-week × hour time factor over a 90-day period. Trained on a short history, a model learns a single spike as a weekly pattern: its bounds widen on that weekday and hour until the day ages out of the calculation period. To remove it sooner, add a **period exclusion** on the entity's Outliers tab and retrain, or let the **AI ML Advisor** in act mode do it.







Inactivity policy

An index whose last licensed volume is older than the threshold **in force** turns red (`anomaly_reason=inactive`, `impact_score_vol_inactive`). Two policy shapes exist:

- **static** — one threshold (`vol_max_sec_inactive`, default 7 days);
- **variable** — slots by day of week and hour of day, the DSM / DHM variable-delay model: `{slot_name, days (0 = Monday .. 6), hours (0-23), max_sec_inactive}`; the **first matching slot** applies, the default (`vol_inactive_default`) outside every slot.

The tenant policy (`vol_inactive_policy`, **variable by default: 1 day during working hours, Monday to Friday 08:00–19:59, and 7 days for nights and week-ends**) applies live to every index without its own override. A **per-index override** — static or variable — is set from the entity menu (*Modify → Inactivity threshold*), from the bulk edit category *Inactivity Threshold*, or by clicking the **Inactivity** column, and can be sent back to the tenant policy. A threshold of 0 disables the check (the whole static policy, or one slot); otherwise it runs from 300 seconds to 90 days. Durations accept seconds, a unit suffix (`30m`, `12h`, `1d`, `1w`) or `[D+]HH:MM:SS`.

Note

The slot clock is **splunkd's system time zone** — the frame the editor's hour grid is drawn in (the editor shows the server time and converts your browser hours). It is not the time zone of the user running the tracker or viewing the page, so the scheduled tracker, a Tenant Home refresh and an inactivity save all pick the same slot.

The red status message names the slot in force (and says *set for this index* when an override applies); the table's compact *Inactivity* column shows the threshold in force, with the slot and the source in its tooltip.





Trend and projection

The collector computes, per index and on every run, from the closed license days it holds:

- `yesterday_volume_mb`, `mtd_volume_mb` (month to date), `avg_daily_30d_mb`;
- `trend_30d_pct` — a **robust (Theil–Sen) slope** over the last 30 closed days (the median of the slopes between every pair of days), as a percentage of their **median**, never below −100 % (a volume cannot fall by more than all of it), from 7 days of history;
- `projected_month_mb` — the month to date plus, for each remaining day, its expected rate: the **median** of the same weekday over the last 28 license days once two samples exist, else the mean of the last 7 days. The open day counts for the larger of its volume so far and its expected rate. **No projection before 7 closed license days**: a day or two of history gives no sensible monthly estimate. A day missing *inside* the index's history (a collector outage past the catch-up) is estimated at its rate; the days before the index's first recorded day are not.

Both figures use medians on purpose: a **single outlier day cannot dominate them**. A one-off spike or outage inside the 28 days is not repeated on every remaining same weekday of the month, and does not turn a flat index into a steep trend. With only two samples of a weekday, the median is their mean.





The *Volume & cost* tab draws the daily licensed volume of the last 60 days with the per-day projection to the **end of this month** or **of next month** (a switch remembered per viewer), scaled to the collector's figure so the chart and the card agree; **Open in search** carries the same rule in SPL. There is deliberately no `predict`: a weekday-weighted rate is what the license bill follows.

Global license usage (VOL)

A Tenant Home tab next to the VOL component tab: the license consumption of the whole environment, with an opt-in configuration layer (named scopes, working days, daily quotas, costs and chargeback). It reads what the tenant already keeps (one daily point per index and closed license day, and the entity figures), so there is **no monitoring, no alerting and no search on the license usage log**: a few `mstats` searches, the same for a remote deployment. It is as deep as the tenant's history, live collection plus backfill.

Principles

- **Closed license days only**, in the license manager's time zone. Today, the open day, is never in a chart; the month figures (month to date, month projection) come from the entities.
- **One value per index and license day**, placed in the middle of the day, so any viewer within 12 hours of the license manager sees it on its own date.
- **Analysis period**: every view has its selector — rolling 7, 14, 30, 60 or 90 closed license days, 6 or 12 calendar months, the current or previous week, or the year to date (Overview opens on 30 days, *Daily changes* on 14, *Investigate an index* on 90). A banner says when a backfill is still filling the history in, or where the history starts when it is shorter than the period.
- **Binary units, IEC labels.** Every volume is computed in base 2 (1024) and labelled **MiB / GiB / TiB** by default (2.4.22). The system setting `[trackme_ui_defaults] volume_unit_labels = splunk` (*System settings → User interfaces*) switches the whole app to Splunk's own MB / GB / TB labels for the same values, and each user can pick either style, or the system default, in the settings panel of Tenant Home and Virtual Tenants; the values never change, only the labels. Every chart draws in one unit picked from its largest value; tables and tiles follow the same rule.
- **Exact dates, stated frame.** *Licensed yesterday* and the pool meters show the last closed license day or **Not recorded** (the last recorded day is named separately), never an older day under that label; every sub-tab states the license-day frame (license manager midnight and its UTC offset) and names the collection scope — a custom license source, permanently deleted indexes, missing or partial days.

- **No auto-refresh.** Tenant Home's auto-refresh is paused while the license view is open, so an analysis or a configuration draft is never reloaded under you; **Refresh** reloads.

Sub-tabs and deep links

Two rows of sub-tabs: the views on the first — **Overview**, **Daily changes**, **Investigate an index**, **Scopes**, **Working days & holidays**, **Over quota**, **Chargeback**, **Configuration** — and, under **Scopes** only, a second row with **All scopes** and one tab per enabled scope (tabs while they fit, a filterable selector beyond, up to 50 scopes). Each view has a deep link:

```
TenantHome?tenant_id=<tenant>&component=vol&tab=license_usage&view=<view>
```

where `<view>` is `overview` (the default), `daily-changes`, `investigate-index`, `scopes`, `scope-<scope id>`, `working-days`, `over-quota`, `chargeback` or `configuration`. A link to a scope that no longer exists, or is disabled, opens the overview. *Daily changes* and *Investigate an index* keep their selection in the link — `vol_index=<index>`, `vol_day=YYYY-MM-DD` and `vol_compare=weekday` (the previous-day comparison is implicit) — so a dated drilldown can be shared; without a date, a link opens on the last closed day against the previous day.

Overview

- **Key figures —** Licensed yesterday, Daily average (with the working-day average once working days are classified), Peak day, Trend (the 30-day rule of the entities, over the environment), Month to date, Month projection (the sum of the per-index projections, with the number of indexes that have one), Top 5 share. With costs on: the **Licence cost** or the **Cost of the volume** of the period (see *How costs are computed*), the **Cost per GB**, and the **Unused capacity** (or **Over the licence**). With a quota in force: the **Quota used yesterday** and the **Days over quota** of the period.
- **Insights** in plain words, most important first: a **Yesterday** block (the licensed volume of the last closed day, its change against the previous day and against the same weekday last week, the top changes per direction), then days over quota and pools close to it, the trend, the largest increase and decrease, indexes that stopped consuming, new consumers, the concentration, the weekly pattern and the licensing model.
- Every card's menu offers **Refresh**, **Open in search** (the `mstats` search behind the figure) and, for a figure measured in the license usage log, **Verify against license_usage.log** (see *Closed-day reconciliation*).
- **Licensed volume per day —** stacked by top consumers, with a **Top consumers** slider from 5 to 100 (remembered per viewer; the rest is *All others*) and the daily quota as an overlay line when a quota is in force.
- **The whole environment per day —** the daily total with the projection to the end of this month or of next month, the same rule as the entity chart.
- **Top consumers —** every index, sortable and filterable: rank, share, total, average, peak, the last 7 closed days against the 7 before, pool, projection and 30-day trend, plus the cost columns when costs are on. A click opens the entity.
- **Movers —** rising, falling, new consumers and stopped consuming, last 7 closed days against the 7 before. They need 14 recorded days: on a young tenant the card says so instead of calling every index new.
- **License pools —** one card per pool with its meters (yesterday, average, peak day), the number of recorded days over quota and the headroom; *(configured)* marks a quota set in the configuration. Each pool's quota is read from the pool's own rows, and the view lists the pools configured now: a pool the license manager has not reported for 4 days leaves the quotas (the card says so after a day without report — a pool fed on weekdays only reads so every weekend) while its recorded days stay in the charts under its name.


splunk>enterprise
Apps
Jordan Wildinson
Messages
Settings
Activity
Help
Find
Virtual Tenants
Topology Studio
Configuration
Artificial Intelligence
Maintenance & holidays
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
TrackMe - Tenant Home
Monitoring the great value in your Big Data
AI assistant
Volume Outliers (VOL)
Global license usage (VOL)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Overview
Daily changes
Investigate an index
Scopes
Working days & holidays
Over quota
Chargeback
Configuration
About this deployment
Global license usage
Last 30 license days
Refresh
LICENSED YESTERDAY
1.71 TiB
DAILY AVERAGE
1.72 TiB
PEAK DAY
2.35 TiB
TREND
-5.5%
MONTH TO DATE
5.28 TiB
MONTH PROJECTION
50.13 TiB
TOP 5 SHARE
50%
LICENSE COST (PERIOD)
£30,000
COST PER GIB
£0.5684 / GiB
QUOTA USED YESTERDAY
171%
DAYS OVER QUOTA
30
Insights
Yesterday : 2026-11-03
Largest increases
Largest decreases
In the selected period
Licensed volume is stable
Largest increase: business_data
Largest decrease: slam-wineventlog-islam
Where the license goes


Licensed volume per day
Top consumers
10 indexes (5 to 100)
Licensed volume per day — the whole environment
Projection
End of this month
Top consumers
Movers
RISING
FALLING



Daily changes

Which indexes moved, day by day. Each closed license day of the analysis period is compared with the **previous day** (the default) or with the **same weekday last week**:

- **Cards —** the environment volume on the selected day and its change against the comparison day, then the **largest increase** and the **largest decrease** of that day, each naming its index.
- **Largest daily volume changes by index —** one bar per mover and day, above zero for an increase and below for a decrease: the top 3, 5 or 10 per direction, chosen each day from every index (never only the Overview's top consumers). Hover a bar for its index and percentage; click a day to select it.
- **What changed on <day> —** every index of the selected day: comparison volume, selected-day volume, change, change in percent and the recorded volume of the last 7 days as a sparkline; sortable (the **Change** header cycles largest change → increases first → decreases first) and paged by 25. A row opens *Investigate an index* on that index and day.
- **Filter indexes** narrows the chart and the table by name, and the index selector next to it focuses the cards and the chart on one index, without the ranking cap.

Missing days stay gaps and a recorded zero is compared as zero: a change needs a recorded volume on both days, and an index with none on the comparison day reads *No prior volume*. These are changes, not outlier classifications.





Investigate an index

Everything the tenant knows about one index, read-only. Pick it in the selector (every tracked index, including one without a closed day yet) or arrive from a row of *Daily changes*, from the Overview's top consumers, or from a deep link:

- The index's **pool** and the **scopes** it belongs to, with **Incidents & entity settings** (the entity's own modal, where the detection and inactivity settings live), **Search licensed volume** and **Search index events**.
- **Licensed in analysis period**, **Peak day in analysis period** and, with costs on, **Cost in analysis period**.
- The daily-change explorer of *Daily changes* for this index alone: the selected day and its comparison, the index volume and change cards, the daily bars and the selected-day comparison.
- The entity's own charts on their own windows — the daily licensed volume with its projection, the rolling volumes and their history, the outliers model and the hourly profile — as on the entity's *Volume & cost* tab.





Scopes

A **named scope** is a set of indexes analysed together: a team, a use case, the data of Enterprise Security. Up to 50 scopes; an index may belong to several (overlaps are allowed and shown). Two sources:

- **Index patterns** — include and exclude patterns, `*` and `?` as the only wildcards, case-insensitive.
- **CIM datamodel macros** — the scope follows the `cim_<datamodel>_indexes` macros, read where Enterprise Security configures them (**Read macros from**, per scope): this search head, the default, or a TrackMe remote account — the Enterprise Security search head tier when TrackMe runs on its own monitoring search head — as the caller, with the account's role restrictions. A remote source that cannot be read is reported as the scope's own error, never replaced by this search head's macros. A macro left at its default `()` is **unconstrained** and is left out, never read as every index; a macro that is not a plain list of index terms joined by `OR` is **unreadable** and left out too. Datamodels can be excluded, and exclude patterns apply on top. When every macro is at its default the scope is empty and the tab says so.

All scopes (`view=scopes`) compares every enabled scope side by side: its source (with the read error when unavailable), its indexes, **Licensed (period)**, **Share of the licence** or **Cost (period)**, the last 7 days against the 7 before and the month projection — sortable, filterable by name, a row opens the scope. It is computed from the per-index search the view already runs, so there is no daily average or peak here.



A scope tab shows **Licensed (period)**, **Share of the licence** (or **Cost (period)**), **Daily average**, **Peak day** (the peak of the scope's own daily series, not the sum of its indexes' peaks), **Last 7d vs prior 7d** and **Month projection**; the daily volume stacked by index with its own top consumers slider (50 by default); the scope per day with its projection; and its **Indexes** table.



Working days & holidays

Weekends and public holidays pull a daily average down, so the view gives a working-day average everywhere and a dedicated sub-tab: **Working-day average**, **Weekend average**, **Bank holidays** (the license days of the period that fall on a holiday, shown when holiday calendars apply), **Holiday average**, **Working vs non-working**, **Peak working**

day and **Quota used on a working day**; the daily volume coloured by kind of day; the average per weekday (holidays left out); the top consumers and the license pools on a working day; and the holidays of the period.



A license day is a **holiday** when its date is covered by a holiday of the calendars in use, else a **working** day or a **weekend** day from the working weekdays (Monday to Friday by default). The holidays are either not counted (default), those of the tenant's *Bank Holidays* calendars, or those of calendars you select. Only the holidays that count in a calendar are used, with their exact date every year (Easter Monday, Thanksgiving…). Working days follow the tenant's calendars, not the four Bank Holidays effects.

Over quota

The days a license pool went over its daily quota. It needs a quota in force: the license manager's pool size (Splunk Enterprise) or a **daily quota** configured per pool (any deployment, Splunk Cloud included). Without one, the tab offers *Configure the daily quota*.

- **Window —** 3, 6 (default) or 12 calendar months, ending with the last closed day.
- **Tiles —** In the window, This month, Last month, Rolling 60 days and Worst day.
- **Licensed volume per day against the quota —** one column per day, blue within the quota, red over it, with the quota line; a pool selector when several pools have a quota (the one most often over by default).
- **What made the day —** pick a day with the date selector or **Previous day / Next day** (a badge says *Over quota +N%* or *Within quota*): the pool's 10 largest indexes that day against their **usual day**, the average of their previous recorded days (up to 30). Each index links to its entity. Clicking a column of the chart presets the selector.
- **Days over quota —** the list, with the three largest indexes of each day and their change; **Per calendar month**; and **Splunk license warnings** for context.


splunk>enterprise
TrackMe - Tenant Home
Monitoring the great value in your Big Data
Volume Outliers (VOL)
Global license usage (VOL)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Overview
Daily changes
Investigate an index
Scopes
Working days & holidays
Over quota
Chargeback
Configuration
About this deployment
Over quota
Window
Last 6 months
Refresh
Configure
IN THE WINDOW
102
days over quota since Mon 1 Jul
THIS MONTH
3
days over quota in November 2025
LAST MONTH
31
days over quota in October 2025
ROLLING 60 DAYS
60
WORST DAY
235%
Licensed volume per day against the quota
What made the day
Previous day
Next day
Over quota +71%
Within quota
Over quota
Daily quota (TiB)


What made the day
Day
11/3/2026
Previous day
Next day
Over quota +71%
That day
Usual day
Index
That day
Usual day
Difference
Change
Share of the day
Days over quota
Open in search
Day
Pool
Licensed
Quota
Over by
Largest indexes that day (against their usual day)
Tue 3 Nov
Mon 2 Nov
Sun 1 Nov
Show

Note

Days over quota are not Splunk license warnings. Splunk counts one warning per pool and license-manager day over the pool's quota. A stack under 100 GB/day has search disabled for a pool after 45 warnings in a rolling 60 days; from 100 GB/day, warnings are issued but search is not disabled. TrackMe does not know the size of the stack (a pool is a slice of it), so the view states both rules and never claims which applies. Its warning count uses the license manager's quota only: a configured quota never raises one. Past days are measured against today's quota.

Chargeback

The license cost of each **calendar month**, charged to **cost centres**. It needs costs (a price) and at least one centre; otherwise the tab says what to configure.

- **Partition —** the centres are an ordered list, each with include and exclude index patterns. An index belongs to the **first** centre that matches it, so every GB is charged once; the configuration shows the indexes an earlier centre already took, so you can check the order. Indexes no centre matches form the **Unallocated** line, or are **spread** across the centres in proportion to their volume.
- **Statement —** for the month you pick among the last 13: the **Bill of the month** (the calendar days of the month; the closed days so far for the current month, marked *to date*), **Charged to the centres**, **Unallocated**, **Unused capacity** or **Over the licence**, and **Over budget**. Per centre: volume, share, cost, budget used and the change against the previous month, shown only between two complete months. Each line opens on its indexes. A 13-month trend chart and **Export CSV** complete it.
- **Prices with an effective date —** a renewal does not reprice the past: each previous price applies up to the month you give it, on its own basis (annual or per GB).
- A collection gap never shrinks the bill: the days are flagged and the rules under the statement say how each basis treats them.



Important

Statements are computed live from the **current** cost centres and price history: no statement is frozen. To keep the statement of a closed month as it was, export it to CSV.

Configuration

The **Configuration** sub-tab holds five sections, saved together with **Save configuration** (an optional comment goes to the audit trail), with live previews as you type. The sub-tab keeps an unsaved draft while you visit the other sub-tabs; **Reload saved** discards it.

Section	What you set
Scopes configuration	Add, remove, enable or disable a scope; its name, description, source (**Index patterns** or **CIM datamodel macros**), include and exclude patterns, excluded datamodels. The preview lists the indexes each scope resolves to.
Working days configuration	The working weekdays and the holidays: **Not counted**, **The tenant's calendars** or **Selected calendars** (with this year's holidays of each). The preview classifies the last 30 license days.
Daily quota configuration	One row per pool with the license manager's quota and an optional daily volume in GB. A configured quota becomes the view's reference (or gives a Splunk Cloud stack the volume it was sized for); it never changes Splunk's own warnings. Pools no longer reported stay listed so you can clear them.
Chargeback configuration	Enable the chargeback, choose what happens to unallocated indexes (**Its own line** or **Spread across the centres**), then the ordered cost centres (**Up / Down**) with a name, an owner, a monthly budget, a description and the patterns. **Seed from scopes** creates one centre per enabled pattern scope. The preview shows the partition.
Costs configuration	Enable the costs; the basis (**Annual licence cost**, with the licensed volume per day, or **Cost per GB**); the currency (ISO code, USD by default); the allocation (**By rate** or **By share of the bill**); the display (**Next to the percentages** or **Instead of the percentages**); and the **Price history**: **Record a price change** copies the saved price to last month before you enter the new one, **Add a previous price** enters an older one.

The configuration is one document per tenant with a revision: a save based on an older revision is refused, so two people editing at once never overwrite each other silently. It is backed up with the tenant. Nothing is configured by default.

Import / Export (an inner tab of *Configuration*, 2.4.22) moves the whole document between tenants and deployments. **Export** downloads the saved configuration — the five sections, the price history included, never unsaved edits — as a JSON file of format `trackme.vol.license-configuration`. **Import** reads such a file (up to 2 MiB), validates it on the server, shows the summary and the JSON, and **Replace configuration** replaces the destination's whole document after a confirmation that names the tenant: no merge (an empty section removes what was there), the destination's loaded revision must still be current, and unsaved edits or read-only access block it. Remote CIM accounts and Bank Holidays calendars are referenced by name, not exported: they must exist on the destination.


splunk>enterprise
Apps
Jordan Wilkerson
Messages
Settings
Activity
Help
Find
Virtual Tenants
Topology Studio
Configuration
Artificial Intelligence
Maintenance & holidays
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
TrackMe - Tenant Home
Monitoring the great value in your Big Data
AI assistant
Volume Outliers (VOL)
Global license usage (VOL)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Overview
Daily changes
Investigate an index
Scopes
Working days & holidays
Over quota
Chargeback
Configuration
About this deployment
License view configuration
Reload saved
Save configuration
Scopes configuration
Working days configuration
Daily quota configuration
Chargeback configuration
Costs configuration
Import / Export
Export configuration
Download the complete saved VOL configuration as JSON. Unsaved edits are not included.
Download configuration
Import configuration
Choose a VOL export, validate it, then confirm replacement of the entire VOL configuration of this tenant.
Choose configuration file


splunk>enterprise
Apps
Jordan Wilkerson
Messages
Settings
Activity
Help
Find
Virtual Tenants
Topology Studio
Configuration
Artificial Intelligence
Maintenance & holidays
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
TrackMe - Tenant Home
Monitoring the great value in your Big Data
AI assistant
Volume Outliers (VOL)
Global license usage (VOL)
Investigate Status Flipping
Investigate Audit Changes
Tracking Alerts
Overview
Daily changes
Investigate an index
Scopes
Working days & holidays
Over quota
Chargeback
Configuration
About this deployment
License view configuration
Reload saved
Save configuration
Scopes configuration
Working days configuration
Daily quota configuration
Chargeback configuration
Costs configuration
Import / Export
What is the chargeback?
Every GiB charged once
Unallocated
Priced month by month
Budgets
Enable the chargeback
Unallocated volume
Its own line
Spread across the centres
Name
Owner
Monthly budget
Security AMER (demo)
Demo - Americas SOC
5000
Up
Down
Remove
Description
Demo/testing allocation and illustrative GBP monthly budget, not an actual invoice. Americas SIEM feeds.
Include
Exclude
sim* amer



How costs are computed

Every figure uses one **effective rate per GB**:

- **Annual licence cost, by rate** — the annual cost divided by 365 and by the licensed GB per day (the configured value, else the pools' quota in force). Each GB costs the contract rate, and the **Unused capacity** is the bill of the recorded days minus the cost of the volume used (negative: **Over the licence**).
- **Annual licence cost, by share of the bill** — the bill of the recorded days split across the indexes by volume, so the costs add up to the bill. It needs no licensed volume per day: the natural choice on Splunk Cloud.
- **Cost per GB** — that cost, always by rate.

Amounts are never rounded, so a small per-GB price never turns into zero.

Permissions

Action	Who
Open the tab and every sub-tab	Every TrackMe user who can open the tenant (`trackmeuseroperations`); the searches and the reads run with the user's own session.
Change the configuration (in the page or through the AI Assistant)	`trackmepoweroperations` **and** a power or admin role on the tenant. Otherwise the Configuration sub-tab is read-only. Changes are refused while the TrackMe license is read-only.

Every saved change is audited (`update license view configuration`).

AI Assistant on the license view

While the Global license usage tab is open, the Tenant Home **AI Assistant** works on this view: it knows the view, its configuration and the figures of the sub-tab on screen (the period, the day selected in Over quota, the statement month…), offers suggested questions per sub-tab, and keeps one conversation across sub-tabs.

With your consent it can **change the configuration** through the *Concierge*: add, change, remove or reorder scopes and cost centres, set the costs, record a price change or edit the price history, set the working days, set or clear a pool's quota, enable the chargeback. Each change is applied to the latest saved configuration, audited, and the view reloads it (an unsaved draft of your own is kept). The same permissions apply as in the page.





Known limits

- As deep as the recorded history: live collection plus backfill, up to 400 days within the retention of the license usage log. The movers need 14 recorded days, a month projection 7 closed days per index.
- Past days are measured against **today's** quota (there is no quota history).
- Statements follow the current cost centres and prices (see above).
- CIM scopes read the macros of the search head TrackMe runs on.
- The periods (30 / 60 / 90 days), the over-quota windows (3 / 6 / 12 months) and the chargeback depth (13 months) are fixed choices.

Alerting

VOL entities flow through *stateful alerting* like every component. What is VOL-specific:

- The **Create Alert** wizard offers the `SPLK-VOL` technical alert type (a filter on `object_category=splk-vol`); the outliers trigger stays on by default, the outliers being the VOL state. VOL is selectable in the stateful email component filter and in the SLA breaches alert.
- **Stateful alert emails** embed the rolling volume charts (24 h / 12 h / 4 h / 60 min over the alert's chart window), the outliers chart of each model, and a **daily licensed volume** bar chart over the last 30 license days.
- **AI status reports** run the ML outliers investigation for the index.
- Auto-acknowledgment, notable events, the free-style REST call, maintenance mode, CMDB and alert enrichment (`splk_general_vol_cmdb_search`, `splk_general_vol_alert_enrichment_search`)

and the drilldown (`TenantHome?component=vol&keyid=…`) need nothing special.

See *Alerting — in depth* for the alert types and the email anatomy.

REST endpoints

Resource group `splk_vol` (browse it with **API & tooling → TrackMe REST API Reference**, every endpoint supports `describe=true`):

Endpoint	Purpose
`POST /trackme/v2/splk_vol/vol_get_table`	The VOL entity table of a tenant.
`POST /trackme/v2/splk_vol/vol_entity_info`	One index: its figures, direction, inactivity policy in force and the entity searches.
`POST /trackme/v2/splk_vol/vol_get_state`	The collector state (cursor, validated history, license-manager offset, deployment type) and the backfill progress.
`POST /trackme/v2/splk_vol/vol_license_overview`	The Global license usage context and searches (`period_days` 30 / 60 / 90, `top_n` 1 to 100, `over_quota_months` 3 / 6 / 12), with the license verification context: the `Usage` writer, the effective coverage and the per-day `reconciliation` receipts (also carried by `vol_license_config` and `vol_entity_info`).
`POST /trackme/v2/splk_vol/vol_license_config`	The license view configuration with its resolved scopes, cost-centre partition, calendars and working-days preview, and whether the caller can change it; with a `draft`, the draft validated section by section and previewed.
`POST /trackme/v2/splk_vol/write/vol_license_config_update`	Save the whole configuration (`revision` of the document it is based on, `config`, optional `update_comment`). A stale revision is refused (HTTP 409).
`POST /trackme/v2/splk_vol/write/vol_license_config_patch`	Targeted changes on the latest saved configuration (`operations`, 1 to 50, all applied or none): `scope_upsert` / `scope_remove` / `scope_move`, `centre_upsert` / `centre_remove` / `centre_move`, `chargeback_set`, `costs_set`, `price_change`, `price_history_upsert` / `price_history_remove`, `working_days_set`, `quota_set`. The AI Assistant uses this endpoint.
`POST /trackme/v2/splk_vol/write/vol_set_direction`	The detection direction of one or more indexes (`object_list` or `keys_list`).
`POST /trackme/v2/splk_vol/write/vol_set_inactivity`	The inactivity policy of one or more indexes: `policy` `tenant` / `static` / `variable` with `max_sec_inactive` or `slots`.
`POST /trackme/v2/splk_vol/write/vol_backfill_control`	`action` `pause` / `resume` / `cancel` / `restart`, optional `run_now`.
`POST /trackme/v2/splk_vol/write/vol_bulk_edit`, `vol_update_priority`, `vol_monitoring`, `vol_delete`, `vol_update_wdays`, `vol_update_hours_ranges`, `vol_update_monitoring_time`, `vol_update_manual_tags`, `vol_update_sla_class`	The generic entity operations, as for every component.
`POST /trackme/v2/vtenants/admin/update_tenant_vol_inactivity_policy`	The tenant inactivity policy (dispatches the tracker so the indexes apply it within a minute); read it in the `vol` block of `GET vtenants/tenant_default_delay_config`.
`POST /trackme/v2/vtenants/admin/update_tenant_vol_outliers_kpis`	The default outliers KPIs, with the opt-in reconcile of the existing indexes.

The tenant is created with `tenant_vol_enabled` (see *Creating a tenant*); every `vol_*` option below can be set in the same call.

Tenant options

Stored in the tenant's `vtenant_account` and editable from *Configure tenant*:

Option	Default	Meaning
`vol_account`	`local`	This Splunk, or the remote account the license usage is collected from.
`vol_license_search`	`index=_interna source="*/license_usag log"`	The license search constraint (advanced, see above).
`vol_settle_seconds`	`300`	Settle margin (60–3600 s) before a 5-minute bucket is read.
`vol_late_overlap_seconds`	`1800`	How far before its cursor each run rereads the license usage log for late events (0 disables, max 14400).
`vol_backfill_days`	`30`	Days of history replayed at creation (0 disables, max 400).
`vol_default_direction`	`both`	Default detection direction of new indexes: `both` / `lower` (drops) / `upper` (spikes).
`vol_outliers_kpis`	`last_24h_vol`	The rolling KPIs every new index gets a model on.
`splk_outliers_min_days_hist`	`15`	Minimum history before a model's confidence is *normal* (the wizard's value on a VOL tenant; the system default is 30).
`vol_mloutliers_priority_filt`	every priority	Priorities in scope for outliers.
`vol_mloutliers_filter_expres`	empty	Filter expression restricting the indexes in scope.
`vol_inactive_policy`	`variable`	The tenant inactivity policy shape.
`vol_max_sec_inactive`	`604800`	Static policy threshold (7 days).
`vol_inactive_slots`	working hours	Variable policy slots (Mon–Fri 08:00–19:59 at 1 day).
`vol_inactive_default`	`604800`	Variable policy threshold outside every slot (7 days).
`impact_score_vol_inactive`	`100`	Impact score of an inactive index.

The Global license usage configuration (scopes, working days, quotas, costs, chargeback) is not a tenant option: it lives in its own document, edited in the view's **Configuration** sub-tab or through `vol_license_config_update` / `vol_license_config_patch`.

Metrics

Written to the tenant's metric index under `trackme.splk.vol.*` (`object_category=splk-vol`, dimensions `tenant_id`, `object`, `object_id`):

Metric	Meaning	Unit
`trackme.splk.vol.volume_5m_mb`	Licensed volume of the 5-minute bucket (the base series; the hourly profile is `mstats sum(…) span=1h`).	MB
`trackme.splk.vol.last_60m_volume_mb`, `last_4h_volume_mb`, `last_12h_volume_mb`, `last_24h_volume_mb`	Rolling volume ending at the bucket end — the outliers inputs.	MB
`trackme.splk.vol.last_24h_pool_pct_used`	Share of the license pool over the last 24 hours (Splunk Enterprise only).	percent
`trackme.splk.vol.daily_volume_mb`	Licensed volume of a closed license day, stamped mid-day. Since 2.4.22 it carries the effective license weight (`daily_effective_mb` is written alongside as the marker; an older point without it holds the raw ingest). A point corrected by the *closed-day reconciliation* is stamped later in the same day with the `license_day` and `reconciliation_revision` dimensions, so `latest()` selects it.	MB
`trackme.splk.vol.daily_ingest_mb`	Raw ingest of the closed license day, before the license weight (2.4.22).	MB
`trackme.splk.vol.volume_5m_ingest_mb`, `volume_5m_licensed_mb`	The 5-minute bucket as raw ingest and as licensed volume (2.4.22; `volume_5m_mb` stays the raw series the outliers models train on).	MB
`trackme.splk.vol.status`	Decision-maker status.	status code

Every point costs 150 bytes of metric-index licensing: about 288 buckets × 5 measures per index per day live (≈ 216 KB per index per day); the backfill writes the same series over the requested depth, and the wizard shows that estimate before you confirm.

The entity searches (*Open in Search* on every chart) read those metrics:

```
| mstats latest(trackme.splk.vol.daily_volume_mb) as daily_volume_mb
    where index=<metric_index> tenant_id="<tenant>" object="<index>" span=1d
| timechart first(daily_volume_mb) as daily_volume_mb span=1d
| eval daily_volume_gb=round(daily_volume_mb/1024, 3)
```

Troubleshooting

Symptom	What to check
The tracker succeeds but the tenant has no entity	The license search returns no `Usage` row: on a custom constraint, the `type` / `idx` / `b` fields are not extracted (add `KV_MODE = auto` for the source); on a remote account, the account's roles cannot read `_internal` on the remote deployment.
The daily chart shows no bar, or a bar a day early	No closed license day yet (a day sits mid-day; the entity overview chart carries its own 30-day window) — or your Splunk user's time zone differs from the license manager's: the daily point is placed on its license date, check the user preferences.
Every index turned red at once	Inactivity in force: the collection stopped (a failed usage search, the tracker disabled, a remote account failing) — the *Manage: volume collection* screen shows the cursor; the execution summary shows the run errors. Fix the source, the next run catches up 6 hours at a time.
No projection on the *Volume & cost* tab	Fewer than 7 closed license days, or the backfill has not merged its days yet (wait for the plan to complete).
The movers card says it needs 14 recorded days	Expected on a young tenant: the week-over-week comparison needs both weeks recorded.
A CIM scope is empty	The `cim_<datamodel>_indexes` macros are at their default `()` on the search head TrackMe runs on: configure CIM Setup there so the macros list the indexes.
Over quota is empty on Splunk Cloud	A Splunk Cloud pool has no quota: set a daily quota per pool in the **Daily quota configuration** section.
The Chargeback tab only explains what to configure	Costs are off, or no cost centre is defined.
Saving the configuration is refused	Someone saved since your draft started (reload, then reapply your change), you have no power or admin role on the tenant, or the TrackMe license is read-only.
Collection is held: *Usage was found from multiple license writers* / *No Usage writer could be established*	Two hosts write `type=Usage` into the license search (two license managers, a test stack forwarding its log to the same index), or none does yet. Scope `vol_license_search` with `host=<license manager>`; after a license manager replacement, collection resumes by itself 3 days after the old host stops writing. Nothing is lost meanwhile: the cursor waits.
The license view says the license usage events are reaching the indexers late	The collector's window is held behind the wall clock (15 minutes or more) because the writer's own events have not arrived; it waits rather than recording missing volume, and the hold is discounted from inactivity up to 4 hours. Check the license manager's forwarding to the indexers and its clock.
Licensed yesterday reads *Not recorded*	The last closed license day has no figure yet: it closes at the license manager's midnight and is recorded by the next collecting run, or the collection is held (see above). The last recorded day is named under the tile; an older day is never shown under that label.
About this deployment says a license day needs reconciliation attention	The day could not be reread: its `RolloverSummary` or `Usage` records are no longer searchable (retention of the license usage log, a remote account that cannot read it), the day boundaries could not be established, or the search failed. The previous figures are kept and the tracker retries every hour while the day stays in the seven-day window; the card gives the reason and the next attempt.
A day reads *Source reconciled · summary differs*	Not an error: TrackMe matches the detailed `Usage` records and Splunk's own `RolloverSummary` reports a different total for that day. Both totals and the difference are on the card; the comparison keeps running daily.
Yesterday's figure changed after the fact	The closed-day reconciliation repaired it (late `Usage` events, or the first reconciliation after an upgrade); the day's revision is on the card and the corrected point carries `reconciliation_revision`.
A backfill plan is *cancelled (live_history_reset)*	The live collection re-seeded (a gap above 24 hours, a source change) while the plan was open; **restart** the backfill from the *Manage: volume collection* screen once the live cursor advances again.

See also

- *VOL — Volume Outliers* — the overview.
- *Volume outliers on the Splunk license usage with Volume Outliers (VOL)* — the end-to-end walkthrough.
- *Outlier detection — in depth* — the outliers engine reference.
- *Alerting & third-party integration — in depth* — alert types and emails.
- *Metrics reference* — the TrackMe metrics catalogue.

8.3.9 FLX — Flex Objects

Flex Objects turn the result of **any Splunk search** into fully tracked, scored, and alertable entities, with KPIs and thresholds you define. If you can write the search, TrackMe can monitor it. FLX is the most flexible component — and, with a 60+ template catalogue, often the fastest to get value from.

Note

Flex Objects are an **Enterprise / Unlimited Edition** capability.





The Flex search contract

A Flex tracker is a search whose **each result row becomes one entity**. TrackMe reads meaning from specific field names in that output:

Field	Required	Meaning
`object`	yes	The unique entity identifier (any ASCII string).
`status`	yes	The entity state: `1` green, `2` orange, `3` red, `0` unknown.
`group`	no	A logical grouping (defaults to the tracker name); an entity should belong to one group.
`metrics`	no	A JSON object of KPIs — one numeric value per key.
`outliers_metrics`	no	Per-metric ML outlier rules.
`alias`, `object_description`, `status_description`	no	Display name and human-readable context.
`max_sec_inactive`, `default_metric`, `default_threshold`	no	Inactivity window and default KPI / threshold.

Each metric you emit is stored on the entity **and** ingested into the metrics index (prefix `trackme.splk.flx.`), so your KPIs are immediately queryable with `mstats` for your own dashboards.

Dynamic thresholds

You set per-metric threshold rules (an operator and a value) entirely through the UI — no need to edit the SPL. Thresholds can be **time-sliced** (day-of-week × hour-of-day) for tighter limits during business hours. An entity's state is driven by threshold breaches, inactivity, outliers, and metric-shape drift.

The template catalogue

Rather than start from a blank search, load one of **60+ pre-built templates** from the **Flex Objects Library**, organised by vendor and category — Splunk infrastructure, data collection & quality, license management, Splunk Cloud / SVC, SOAR, Cribl, host monitoring, and more. Each template ships a complete search, pre-configured KPIs and outlier rules, a recommended schedule, and inline guidance. Load it, adapt the SPL to your environment, simulate, and create.

Tip

FLX is the most powerful component, and the steepest. Its value depends on the author's SPL skill — so start from a template whenever one fits. See also the *Use Cases* for worked examples.

Creating a Flex tracker

1. **Name & deployment** — local, or a remote account (connectivity is validated).
2. **Define the SPL** — from a template or from scratch.
3. **Time ranges** — short (for example `-5m` for `| rest`) or longer for `tstats` / `stats`.
4. **Simulate** — preview the discovered entities with their parsed metrics and outliers.
5. **Schedule & create.**

Outlier detection

FLX is, with DSM, the natural home for ML outlier detection. Per metric you can enable upper/lower breach alerting, with optional seasonality (hour, day-of-week) and a training window. Allow at least a week — ideally a month — of history before relying on outlier alerts. See *Machine Learning*.

Working at scale

A single tracker can manage from a handful to tens of thousands of entities. Use the `group` field deliberately to scope thresholds and SLA and to keep large trackers organised — a common convention is `object = group + ":" + host`. Prefer `tstats` over raw events and keep time ranges narrow.

Converging trackers

A **converging tracker** is a Flex flavour that aggregates the up/down state of many member entities — possibly across several tenants and components — into a single object with a **percentage-availability** KPI (for example "Authentication service: 98.5% available"). It is created from its own wizard, and a searchable inspector lists its members with a link back to each member's own tenant. Membership scope, the *orange-as-up* toggle, and the minimum-green percentage are editable in place; the object name and group are fixed once created.

Managing trackers

Warning

Manage a tracker's lifecycle only through TrackMe — do **not** manually delete its reports or lookups. Deleting a tracker removes its knowledge objects but does **not** remove entities it already discovered; those simply stop being maintained. Tune `max_sec_inactive` so entities that naturally go quiet do not raise false inactivity alerts.

See also

- *Flex Object trackers — in depth* — the tracker library, search contract, KPI/metrics JSON, the wizard, and high-scale grouping.
- *Flex Objects use cases* — worked examples.
- *Entity State & Scoring* — dynamic thresholds and scoring.
- *Machine Learning* — native outlier detection.
- *Hybrid trackers* and *Replica trackers* — advanced tracker types.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.10 Flex Object trackers — in depth

Tip

This is the **in-depth reference** for Flex Object (splk-flx) trackers. For the readable overview of what FLX is and when to use it, start with *Flex Objects*. This page covers the full tracker library, the search contract and KPI/metrics JSON, the creation wizard, outlier detection, high-scale grouping, converging trackers, and managing trackers.

Introduction

The splk-flx component (Flex Objects) allows you to turn the results of any Splunk search into fully tracked, monitored, and alerted entities within TrackMe.

- Flex trackers are scheduled backend jobs that orchestrate entity discovery, KPI tracking, and lifecycle management
- A Flex tracker can monitor anything you can query in Splunk: modular input statuses, Data Model Acceleration, host availability, infrastructure health, IoT devices, or any custom use case
- The component ships with a **library of 60+ pre-built use case templates** organized by vendor and category, ready to deploy in minutes
- A single tracker can discover and manage a few or many entities according to the needs
- **Enterprise Edition & Unlimited Edition** feature

Hint

For a high-level overview of Flex Objects concepts and real-world use cases, see *Flex Objects - Adapt TrackMe to any monitoring use case (splk-flx)*.

The Flex Objects Library

One of the biggest challenges with a component as flexible as Flex Objects is knowing where to start. The **Flex Objects Library** solves this by providing 60+ pre-built use case templates that you can deploy in minutes.

Each template includes:

- A complete Splunk search with best-practice field definitions
- Pre-configured KPIs and metrics
- Outlier detection rules where applicable
- Recommended scheduling and time ranges
- Implementation comments and guidance



The Flex Objects Library with vendor and category filters. Select a use case to view its full definition including the search logic, metrics, and configuration.

Available categories

The library organizes use cases by vendor and category:

Category	Count	Examples
Splunk Infrastructure	17	KVstore status/size, cluster health, SHC status, CPU/memory usage, queue filling, HEC errors, bundle size, log level variations
Data Collection & Quality	10	Host tracking, deployment server clients, UF version tracking, dynamic sourcetype detection, volume variations, events drop detection, fields quality
License Management	5	Pool usage, global enterprise/cloud usage, per-index usage tracking
Splunk Cloud	4	SVC usage (global, per-app, per-consumer service), storage usage
Data Model Acceleration	1	DMA completion, size, runtime, and bucket monitoring
SOAR (Splunk Automation)	10	Asset health, service health, concurrent playbooks, infrastructure load/memory, action/playbook failures, automation brokers, Splunk forwarding
Cribl LogStream	8	Input/output health, CPU usage, destination pressure, pipeline metrics, route traffic, total traffic
Host Monitoring	4	Linux/Windows CPU usage, Linux/Windows memory usage (via TA-nix and TA-windows)

Hint

Loading a use case from the library

When creating a new Flex tracker, select "Vendor" and "Category" to filter the available use cases. Once you select a use case, its full definition is displayed including the search logic, expected metrics, and implementation comments. You can then customize the search to match your environment before deploying.

The Flex Search Contract

At the heart of every Flex tracker is a Splunk search that produces entities. This search must follow a simple contract: each result row represents one entity, and specific field names carry specific meanings.

This is what makes Flex Objects both powerful and approachable - if you can write a Splunk search, you can create a Flex tracker.



The Flex Objects field contract showing required and optional fields with their descriptions and examples.

Required fields

Every Flex search **must** produce these two fields:

Field	Type	Description
`object`	String	The unique identifier of the entity. Any ASCII character is accepted. This is the primary key that TrackMe uses to track the entity over time. Example: `Okta:Production`, `my_index`
`status`	Integer	The health status of the entity: **1** = green (healthy), **2** = orange (warning), **3** = red (critical). If set to **0**, the entity state is considered unknown.

Optional fields

These fields enrich the entity with metadata, KPIs, and anomaly detection:

Field	Type	Description
`group`	String	Logical grouping of entities. Groups can span multiple trackers but overlap should be avoided. If not set, the tracker name is used as the group.
`alias`	String	A human-readable display name for the entity, shown in the UI.
`object_descriptior`	String	Describes the entity for easier management. Example: `Datamodel: Authentication, app: Splunk_SA_CIM, retention days: 7.0`
`status_description`	String	Explains the current status condition. Example: `acceleration is completed`, `Input is enabled`
`metrics`	JSON String	A JSON object containing one or more Key Performance Indicators (KPIs). Example: `{'dma.complete_pct': 100.0, 'dma.size_mb': 136.55}`
`outliers_metrics`	JSON String	A JSON object defining Machine Learning Outlier detection rules per metric. Example: `{'dma.runduration_sec': {'alert_lower_breached': 0, 'alert_upper_breached': 1}}`
`max_sec_inactive`	Integer	Maximum number of seconds an entity can be inactive before turning red. Set to `0` to disable inactivity alerting.
`default_metric`	String	The primary metric to display in the entity table. Example: `dma.complete_pct`
`default_threshold`	JSON String	A threshold rule applied to the default metric for status determination.



Metrics, thresholds, and ML Outliers configuration - fully UI-driven for effortless setup.

Hint

Fully UI-driven configuration

Starting with TrackMe 2.3, the configuration of **metrics**, **thresholds**, and **ML Outliers detection** is entirely managed through the user interface. While the search contract fields remain available for programmatic definition, you can now configure every aspect of your KPIs and anomaly detection directly from the tracker UI - making it easier than ever to fine-tune your monitoring without editing SPL.

Understanding the metrics JSON

The `metrics` field is a JSON object where each key is a metric name and each value is a numeric measurement:

```
| eval metrics = "{'dma.complete_pct': " . complete_pct . ", 'dma.size_mb': " .␣
↪size_mb . ", 'dma.runduration_sec': " . round(runDuration, 2) . "}"
```

When metrics are provided:

- A metric object is automatically created in the Splunk metrics index for each KPI
- Metric names are prefixed with `trackme.splk.flx.` in the metrics index
- You can query these metrics with `mstats` for dashboards, reports, and further analysis

Understanding the outliers_metrics JSON

The `outliers_metrics` field defines which metrics should be monitored for anomalies using Machine Learning:

```
| eval outliers_metrics = "{'dma.runduration_sec': {'alert_lower_breached': 0,
↪'alert_upper_breached': 1, 'time_factor': 'none'}}"
```

Each metric entry contains:

- `alert_lower_breached`: Set to `1` to alert when the metric drops below the lower threshold (e.g., event count drops)
- `alert_upper_breached`: Set to `1` to alert when the metric exceeds the upper threshold (e.g., runtime spikes)
- `time_factor` *(optional)*: Defines the seasonality model for the ML algorithm:
  - `none` or omitted: No seasonality, the model uses the full history
  - `%H`: Hour of day (e.g., "traffic is normally lower at 3 AM")
  - `%H%M`: Hour and minute granularity
  - `%w%H`: Day of week + hour (e.g., "Monday mornings are different from Friday afternoons")
  - `%w%H%M`: Day of week + hour + minute for the finest granularity
- `period_calculation` *(optional)*: How far back to look for the ML model training. Example: `-90d` for 90 days of history.

Creating a Flex Tracker

TrackMe provides a guided wizard that walks you through the entire creation process. You can either start from a library template or define a custom search from scratch.

Step 1: Name and deployment

Choose a name for your tracker identifier and select the Splunk deployment (local or a configured remote account).

The tracker name is used as a prefix for entity grouping and must be unique within the tenant.



Step 1: Enter the tracker name and select the deployment. The name is validated for uniqueness.

When using a remote deployment, TrackMe automatically validates the connectivity to the remote Splunk instance:



Remote deployment connectivity check showing a successful connection to the remote Splunk instance.

Step 2: Define the search logic

Enter your Splunk search or load one from the library. The search must produce at least the required fields (`object` and `status`).



Step 2: Enter or paste your Splunk search logic. The wizard displays the field requirements and examples for reference.

Step 3: Set time ranges

Define the earliest and latest time ranges for the search. Some searches (like those using `| rest`) don't rely on time ranges, in which case you can set a very short window.



Step 3: Configure the search time range. Use short ranges like -5m for REST-based searches, or longer ranges for tstats/stats-based searches.

Step 4: Test and review

Before creating the tracker, simulate the search to verify the results. The simulation shows the raw JSON output for each discovered entity, including the parsed metrics and outliers:



Step 4: Simulate the search. Click "Simulate the search" to preview the entities that will be discovered.



The simulation results showing discovered entities with their metrics and outliers_metrics successfully parsed.

Step 5: Schedule and create

Set the cron schedule for the tracker execution. TrackMe automatically randomizes the schedule to distribute load across your environment.



Step 5: Define the cron schedule. A typical every 5 minutes schedule is randomly distributed for optimization.

Finally, validate and create the tracker:



Promote to Live and create the new Flex tracker.



Success! The new tracker has been created with all its associated knowledge objects (reports, macros, lookups).

KPIs and Metrics

When your Flex search defines the `metrics` field, TrackMe automatically:

1. **Stores the KPIs** as part of the entity record for display in the UI
2. **Ingests metrics** into the Splunk metrics index (`trackme_metrics`) for trending and analysis
3. **Sets a default metric** for the entity table view if `default_metric` is defined

Viewing KPIs in the entity detail

Click on any Flex entity to open its detail view. The **Overview entity** tab shows the current status, description, and key metrics:



Entity detail view showing a CIM Data Model Acceleration entity with its metrics (dma.complete_pct: 99.30%) and status information.

Querying Flex metrics

All Flex metrics are available in the `trackme_metrics` index with the prefix `trackme.splk.flx.`. You can query them with `mstats` for dashboards and reports:

```
| mstats avg(trackme.splk.flx.dma.complete_pct) as avg_complete,
    max(trackme.splk.flx.dma.size_mb) as max_size
where index=trackme_metrics tenant_id="my-tenant" object_category="splk-flx"
by object span=1h
```

Exploring metric events

You can also explore the raw metric events ingested by TrackMe for your Flex entities:



Splunk search showing Flex metric events in the trackme_metrics index with individual metric fields extracted.

Machine Learning Outlier Detection

Flex Objects integrates with TrackMe's Machine Learning Outlier detection to automatically identify anomalies in your KPIs. This is particularly powerful for use cases where you cannot define a static threshold - like event count variations that follow seasonal patterns.

How it works

When you define `outliers_metrics` in your Flex search, TrackMe:

1. **Builds an ML model** for each metric, learning the normal behavior over time
2. **Calculates dynamic thresholds** (upper and lower bounds) based on historical patterns
3. **Detects anomalies** when the current value falls outside the expected range
4. **Triggers alerts** based on your configuration (lower breach, upper breach, or both)

The `time_factor` parameter is key to accuracy: it tells the ML model about the seasonality of your data. For example, if your event counts are naturally lower on weekends, using `%w%H` (day of week + hour) ensures the model doesn't flag normal weekend dips as anomalies.

Viewing outlier detection

The **Outliers anomaly detection** tab in the entity detail shows the ML model's predictions alongside actual values:



Outlier detection view showing the ML model's predicted range (blue band) and actual values over time. The "0 outliers" indicator confirms no anomalies were detected in this window.

The chart displays:

- The **actual metric values** as a line chart
- The **predicted normal range** as a shaded band (upper and lower bounds)
- The **number of outliers** detected in the current time window
- A time range selector to zoom in or out on the data

You can manage the ML models, select which metric to visualize, and fine-tune the detection parameters directly from this view.

Hint

Outlier detection best practices

- Start with `time_factor: 'none'` for metrics without clear seasonality (e.g., DMA completion percentage)
- Use `time_factor: '%w%H'` for metrics with weekly patterns (e.g., event counts, search activity)
- Use `time_factor: '%H'` for metrics with daily patterns but no weekly variation
- Allow sufficient training time (at least 7 days, ideally 30+) before relying on outlier alerts
- Use `period_calculation: '-90d'` to limit the ML training window and adapt faster to changes

High Scale and Custom Grouping

Flex Objects is designed to handle high-scale environments with thousands of entities. Here are key strategies for managing large deployments effectively.

Custom grouping strategies

The `group` field in your Flex search determines how entities are organized. Smart grouping helps you:

- **Scope monitoring** - Different groups can have different alert thresholds and SLA policies
- **Organize at scale** - Group entities by function, environment, tier, or business unit
- **Avoid overlap** - Each entity should belong to exactly one group to prevent duplicate tracking

For example, in a Splunk infrastructure monitoring scenario, you might create separate groups for:

```
| eval group = case(
    tier=="indexers", "infra-indexers",
    tier=="search_heads", "infra-search-heads",
    tier=="heavy_forwarders", "infra-hf",
    1=1, "infra-other"
)
```

Entity naming conventions

The `object` field is the unique identifier for each entity. A common best practice is to prefix the object name with the group:

```
| eval object = group . ":" . host
```

This produces entity names like `infra-indexers:idx01`, `infra-search-heads:sh01`, making it easy to identify entities at a glance.

Performance considerations

- **Optimize your search** - Use `tstats` over raw searches when possible, as it's orders of magnitude faster for large datasets
- **Tune the time range** - Keep the search window as narrow as possible (e.g., `-5m` for REST-based searches, `-30m` to `-2h` for tstats)
- **Schedule wisely** - A 5-minute cron schedule is typical, but less critical use cases can use longer intervals (e.g., `*/15 * * * *`)
- **Use max_sec_inactive** - Set appropriate inactivity thresholds to avoid false positives for entities that naturally go quiet
- **Batch processing** - TrackMe automatically uses batch KVstore operations and parallel processing for optimal performance at scale

Converging Trackers

While standard Flex trackers (use_case type) create entities from a Splunk search, **converging trackers** take a different approach: they aggregate the status of multiple existing Flex entities into a single availability metric.

This is particularly useful for:

- **Service-level monitoring** - Aggregate the health of all components of a service into a single entity (e.g., "Authentication Service: 98.5% available")
- **Executive dashboards** - Provide a high-level view of complex multi-entity monitoring
- **SLA tracking** - Measure overall availability across a group of entities

A converging tracker calculates the percentage of entities in a healthy state within a specified group, providing a single aggregated status and availability metric.

Managing Flex Trackers

The **Manage existing flex trackers** interface allows you to view, inspect, and delete Flex trackers:



The Manage flex trackers interface showing existing trackers with their knowledge objects (reports, properties, schedule).

From this interface you can:

- **View tracker properties** - See the root constraint, cron schedule, time ranges, and associated reports
- **Select trackers for deletion** - Use the checkboxes to select one or more trackers for removal
- **Access the executor job** - Monitor the TrackMe executor job for tracker operations



Selecting trackers for deletion. Multiple trackers can be selected and removed at once.

Important

TrackMe keeps records of all knowledge objects related to a tracker. You must manage the tracker lifecycle through TrackMe - do not manually delete scheduled reports or lookups outside of the interface.

When a tracker is deleted, its associated knowledge objects are purged. However, entities previously discovered by the tracker are **not** automatically removed - they will simply no longer be maintained.

Deleting a Flex Tracker through REST

You can also delete a tracker programmatically via the REST API:

```
| trackme mode=post url="/services/trackme/v2/splk_flx/admin/flx_tracker_delete"␣
↪body="{'tenant_id': 'mytenant', 'hybrid_trackers_list': 'Okta:prod'}"
```

8.3.11 FQM — Field Quality Monitoring

Field Quality Monitoring tracks whether your data is **well-parsed**: how completely and correctly fields are extracted, and how compliant a sourcetype or data model is with the Splunk **Common Information Model (CIM)**. Where the feeds components ask "is the data arriving?", FQM asks "is the data *usable*?"

Unlike the other components, FQM works in **two phases**: a `collect` job samples and scores your events, and a `monitor` job then turns those results into TrackMe entities and alerts.







What FQM tracks

Each entity is the **extraction quality of one field** — typically one entity per `(data model, dataset, field)`, or per a custom SPL pattern. FQM measures:

Metric	Meaning
Coverage %	The share of events in which the field is present / extracted.
Success %	The share in which the field's value is valid / well-formed.
CIM compliance	Drift against CIM-specific expectations.
Inactivity	The dataset has stopped producing data.

You set coverage and success thresholds per metric — the same dynamic, optionally time-sliced threshold model as *FLX*.

Creating an FQM tracker

1. **Pick the source** — a CIM data model and dataset, or a custom SPL pattern.
2. **Choose the fields** to evaluate.
3. **Set thresholds** — coverage % and success %.
4. **Review and create.**

FQM leans on the Common Information Model: standard CIM lookups provide ready-made field patterns, while custom SPL covers non-CIM fields.

Tip

Aim for a realistic bar. 95% coverage is a common, healthy target — insisting on 100% will generate noise. Outlier detection is supported but **not recommended** for FQM, as field-quality metrics are slow-moving.

See also

- *Field quality use cases* — worked examples.

- *FLX — Flex Objects* — the shared dynamic-threshold model.
- *Entity State & Scoring* — thresholds and scoring.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.12 WLK — Workload Knowledge

Workload Knowledge tracks the **execution health of your scheduled searches** — are they running on schedule, completing without errors, not skipping, and not quietly costing more than they should? It turns Splunk's own scheduler activity into tracked, scored entities.

Note

WLK is a licensed / restricted component.





What WLK tracks

Each entity is **one scheduled search in one context** — its app, owner, and saved-search name together. The state is driven by these anomaly reasons:

Anomaly	Meaning
Skipping searches	The scheduler is skipping runs. Orange below 5%, red at or above 5% by default.
Execution errors	The scheduler reported execution errors.
Execution delayed	The search has not run on its cron schedule (within a grace period).
Orphan search	The search's owner is no longer valid, so it cannot run.
Runtime / cost outliers	An ML-detected spike in run time or cost.
Versioning drift	The search definition changed.
App disabled	The parent app is disabled, so its scheduled searches do not run.

The KPIs behind these include execution / completed / skipped / error counts, run time (`elapsed`), CPU and memory percentages, scan count, and — on Splunk Cloud — **SVC consumption** per search, for cost visibility.

Versioning

WLK can fingerprint each saved search's definition and flag when it **changes** — recording a versioning event and, when the change was made through Splunk Web or the API, a **diff** plus who changed it and when. This is invaluable for correlating a sudden jump in run time, cost, or skip rate with an edit to the search. Versioning is opt-in per app.

The tenant default threshold

WLK has a special tenant-wide fallback: a **default threshold row** that applies to any entity without its own per-entity threshold, so you can set sensible limits once and override only where needed.

Setup notes

- The creation wizard auto-builds the trackers it needs (scheduler, metadata, orphan, introspection, and — on Splunk Cloud — SVC).
- Entities are grouped by **app**, or by a custom **overgroup** if you want to host several search-head tiers in one tenant.

- On Splunk Cloud, point WLK at a remote deployment targeting the relevant search-head tier (see *Integrations & Remote Deployments*).

Tip

Dedicate a Workload tenant per search-head tier, and use an overgroup only when you deliberately want to combine tiers. Outlier detection is not recommended for the scheduler counters (too discrete), but it *is* useful for run-time and cost trends.

See also

- *Workload monitoring — in depth* — Splunk Cloud setup, status-macro internals, the overgroup procedure, metadata versioning, and the anomaly-reason and metrics reference.
- *Entity State & Scoring* — thresholds, scoring, and the default-threshold row.
- *Alerting* — turning execution problems into notifications.
- *Integrations & Remote Deployments* — Splunk Cloud / remote-tier setup.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.13 Workload monitoring — in depth

Tip

This is the **in-depth reference** for Workload (splk-wlk) monitoring. For the readable overview, start with *Workload monitoring*. This page collects the full administration and usage detail: Splunk Cloud setup, status-macro internals, the overgroup procedure, metadata versioning, the anomaly-reason and metrics tables, and the discovery / investigation workflow.

Introduction to the Workload component

The TrackMe workload component (splk-wlk) is a licensed users restricted component which tracks the activity from any kind of scheduled search in your Splunk deployment, from reports to alerts and Data Model Acceleration. (DMA)

The Workload component tracks various use cases such as execution errors which can slightly affect the consistency of your Splunk use cases, providing powerful features to detect issues and track the performance of the Splunk scheduling activity.

For the User Guide, consult: *WLK — Workload Knowledge*

Note

Grouping in the Workload component: group and overgroup

- By default, the Workload component groups entities by `app` which represents the Splunk application namespace hosting the scheduled report or alert.
- When creating the tenant and also later on when creating trackers manually, since TrackMe 2.0.70, you can optionally override this behaviour by setting up an `overgroup` value.
- By doing so, the Workload will group entities based on a custom term instead of applications, this can be useful if for instance you want to have multiple Search Head tiers in the same tenant.
- Refer to the *Creating a Workload tenant to host multiple Search Head tiers with overgroup* section for more information.

Note

Data collection enhancements in TrackMe v2.0.75

- Since TrackMe 2.0.75, the Workload components has been enhanced to automatically support data collection overlap between execution, relying on the tracking of the last known metrics for a given account/group and type of metrics.
- This allows TrackMe to look at multiple slices of times when tracking the Workload activity (scheduler, introspection, etc) and avoids missing any data point in case of Splunk delaying the execution of the workload trackers. (Splunk skew parameter, time_window and/or over busy scheduler)
- This process is fully automated and transparent, the last seen metadata are stored in a KVstore called `trackme_wlk_last_seen_activity_tenant_<tenant_id>`.

Creating and configuring a Workload Tenant

The creation and configuration of a Workload Tenant is guided through the wizard, click on the "create new tenant" and choose the associated component to start the configuration process:



Workload component in Splunk Cloud

Generally, in Splunk Cloud have two or more Search Head layers:

- The ad-hoc Search Head tier, where TrackMe should be used (standalone or SHC)
- The premium app Search Head tier, running Enterprise Security or ITSI (standalone or SHC)

In a nutshell, the configuration path is the following:

- configure a TrackMe Splunk Remote Deployment which targets the premium app Search Head tier (this is required notably for the versioning tracking of scheduled searches, even through both Search Head tiers access the same indexing tier)
- following these instructions to identify the Search Head tiers names as automatically defined per Splunk Cloud summary data, and recycle this information using a simple subsearch (see in the next sections)
- create and configure the Workload tenant in TrackMe

Step 1: configure the remote Splunk Deployment

For more information about TrackMe's Splunk Remote Deployments capabilities and configuration, consult: *Remote Splunk deployments*

Proceed as following:

- Create a bearer token in the premium app Search Head tier, it is good practice to associate the bearer token with a service account dedicated for that purpose



- Identify the Splunk Cloud label per Search Tier, we will use this logic as a subsearch when creating the Workload tenant/trackers:

```
index=summary source="splunk-svc-consumer" earliest=-7d latest=now
| stats values(usage_source) as members by search_head_names
```

Hint

Using the search_head_names label with a subsearch to dynamically restrict the hosts

- In Splunk Cloud, use the following subsearch technique to assign dynamically the list of Search Heads for that given Search Head tier
- If Search Head members are re-created, the subsearch will automatically pick up the new hosts and you have no changes to perform yourself

for instance, if the target Search head tier is called "es" (Enterprise Security), we will use the following subsearch to filter out the hosts dynamically:

```
[ search index=summary source="splunk-svc-consumer" search_head_names="es"␣
↪earliest=-24h latest=now | stats count by usage_source | table usage_source |␣
↪rename usage_source as host | format | eval search = if(search="NOT ()", "host=*
↪", search) ]
```

Once you have validated the connectivity, you can continue and start the configuration of the Workload tenant.

Step 2: create and configure the Workload tenant

Open the assistant wizard, and proceed as follows:

- Select "Splunk Cloud" in the Splunk environment dropdown:



- Select the Remote deployment target:



- Set the search restrictions:

Hint

Search restrictions per Workload component

- Since the release 2.0.57, you can define the search restrictions per component, to ease and improve the configuration process
- You can directly validate that the search will return results by clicking on the search preview link
- This opens a pre-populated search interface, it also sets the restrictions and calls splunkremotesearch if the search has to run against a remote Search Head tier



Complete the rest of the setup, it can take up to 5 minutes before entities start to be visible in the Workload tenant.

Workload component in Splunk Enterprise

If you are running Splunk Enterprise, then the setup slightly depends on your deployment:

- If you have multiple Search Head tiers, you can use the same tag approach as documented in the Splunk Cloud setup above
- You can also create a custom indexed or search time field on the Search Head tier level, which automatically identifies the Search Head tiers to be targeted
- Use different combinations up to your preference

Managing Workload Trackers

There are a certain number of Workload trackers which are created automatically by the wizard, depending on your Splunk environment.

Through the trackers management user interface, you can eventually add new trackers if needed, either if you want to re-create the trackers or extend the tracking to additional context:



However, the general recommendation is to dedicate a Workload Tenant to a given Search Head tier.

The REST API UI also allows you to manage all types of Workload trackers:



Managing Workload detection

When a Workload Virtual Tenant is created, TrackMe automatically creates a macro which defines the main status definition behaviour for that tenant.

The macro is named as follows:

- `trackme_wlk_set_status_tenant_<tenant_id>`

You can update this macro to modify how TrackMe defines the status of Workload entities, for instance these lines define the grace time period for the detection of delayed entities:

```
``` Calculate the delta in sequence between now and the last execution compared␣
↪against the requested cron schedule sequence, add 1h of grace time, detect if␣
↪the execution has been delayed ```
| eval status=if(cron_exec_sequence_sec>0 AND ( now()-last_seen >␣
↪(cron_exec_sequence_sec + 3600) ), 2, status)
```
```

This also defines the level of skipping searches (percentage) which leads entities to turn orange or red:

```
``` If there are skipping searches, define two levels of alerting, less than 5% is␣
↪3 (orange), more is 2 (red) ```
| eval status=case(skipped_pct>0 AND skipped_pct<5, 3, skipped_pct>0 AND␣
↪skipped_pct>=5, 2, 1=1, status)
```

The macro is part of the life cycle of the tenant's knowledge objects, therefore it will be removed automatically if the tenant itself is deleted.

This macro is executed by the "main" tracker, which naming convention is the following:

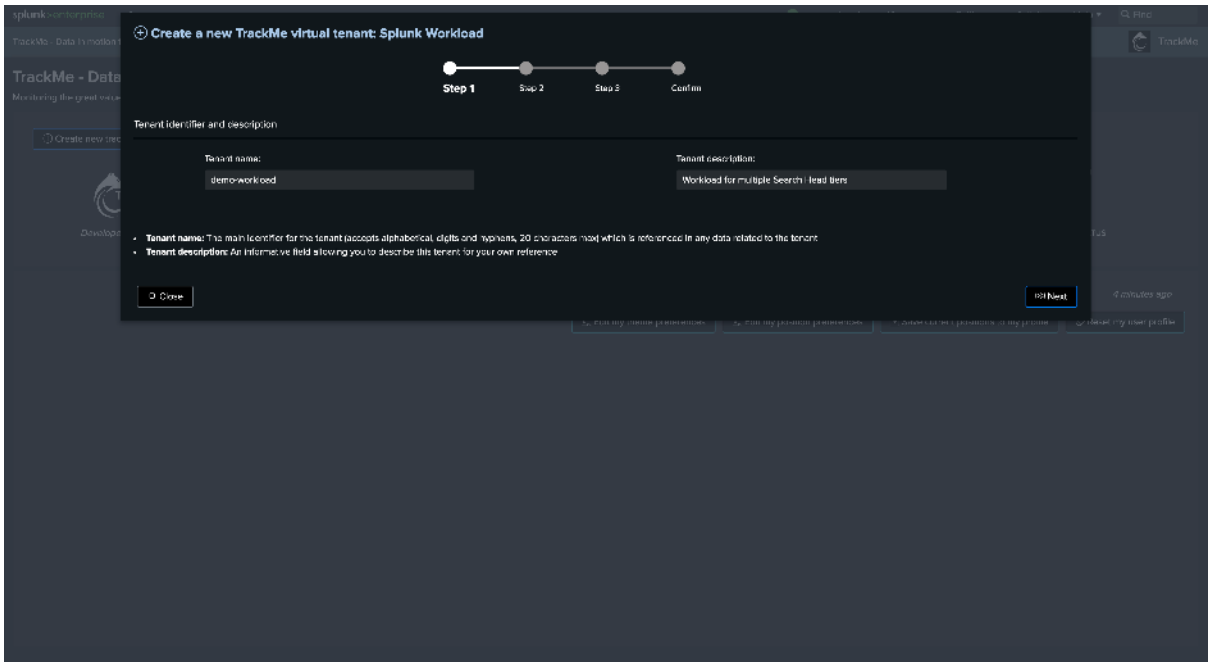
- `trackme_wlk_hybrid_main_ce555e75db1643a_wrapper_tenant_<tenant_id>`

Creating a Workload tenant to host multiple Search Head tiers with overgroup

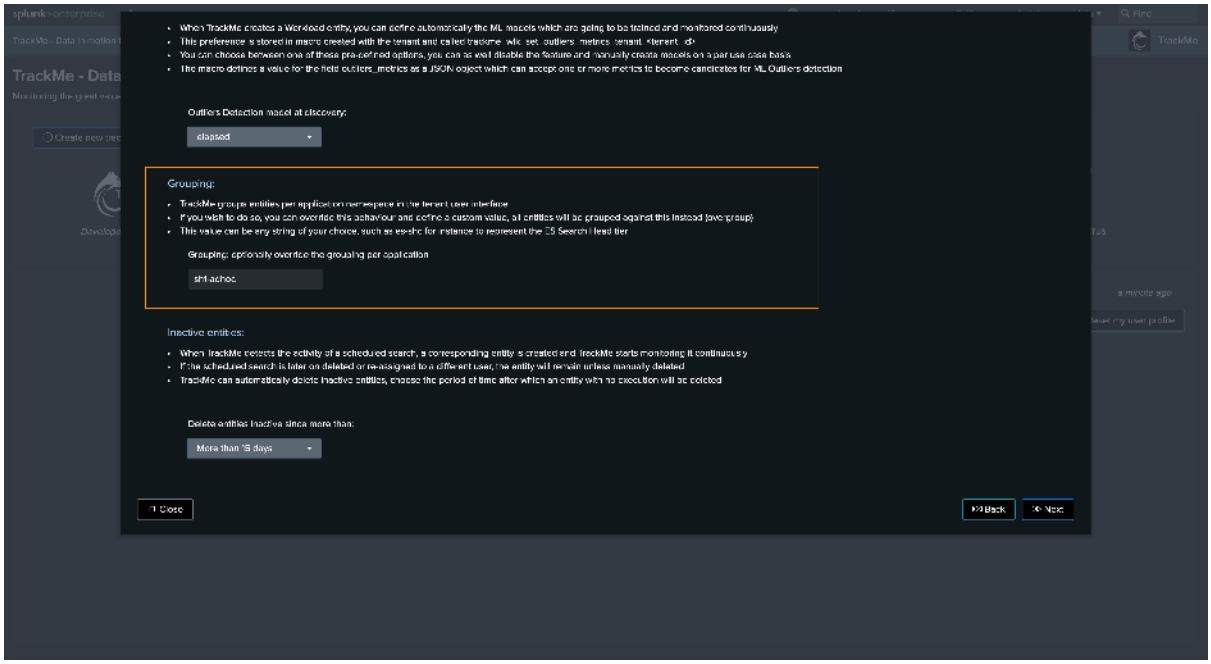
To create a tenant which hosts multiple Search Head tiers, you can use the overgroup feature, follow the steps below:

Step 1: Create the tenant with a first Search Head tier and the corresponding overgroup

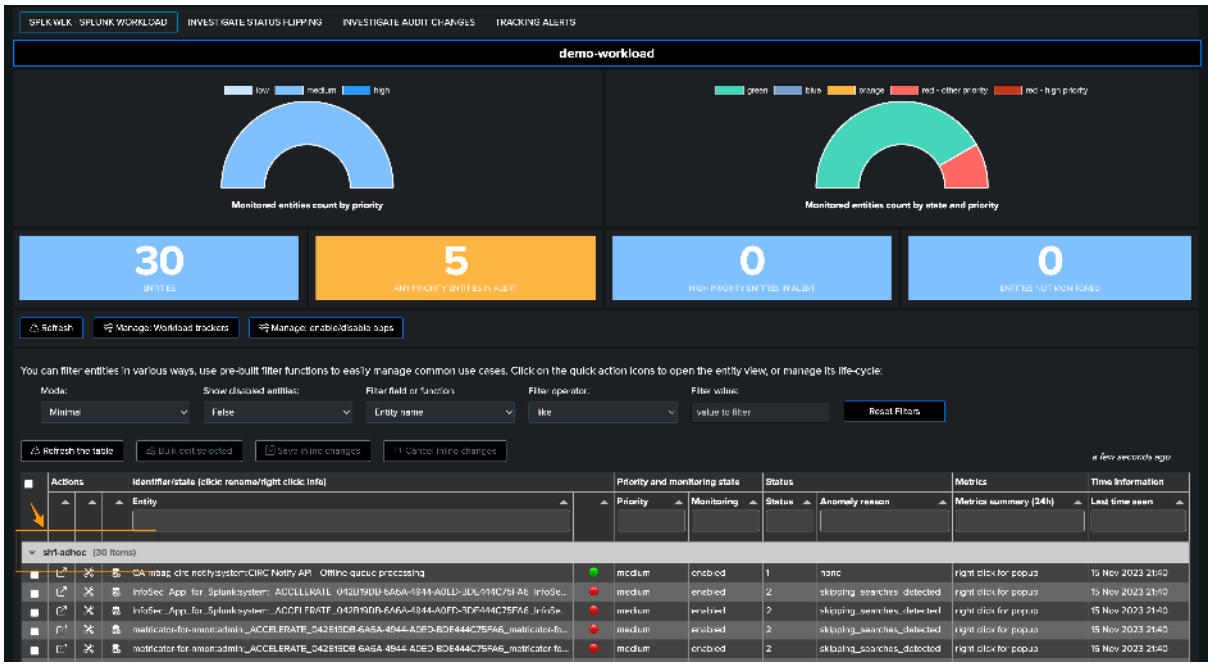
We start by creating a brand new Workload tenant, where we define the first Search Head tier constraint, target (account) and overgroup value:



Define everything as needed (account, constraints, etc) then enter a custom value for the overgroup:
```



Once created and after 5/10 minutes, entities will be visible under the overgroup:



Step 2: In the Workload tenant, create the additional trackers for the second Search Head tier

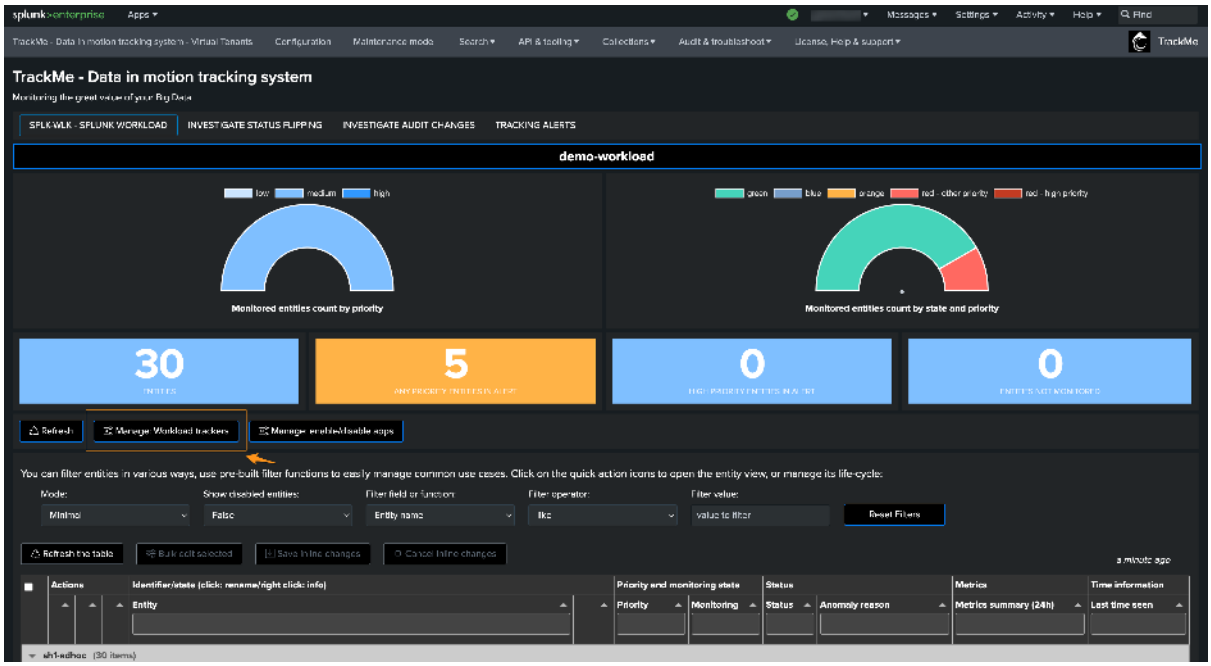
Now that the tenant is created, we can create the additional trackers for the second Search Head tier, the following types of trackers can be created:

- scheduler
- introspection
- Splunk Cloud SVC (optional, for Splunk Cloud only)
- Notable (optional, if you wish to track notable activity)

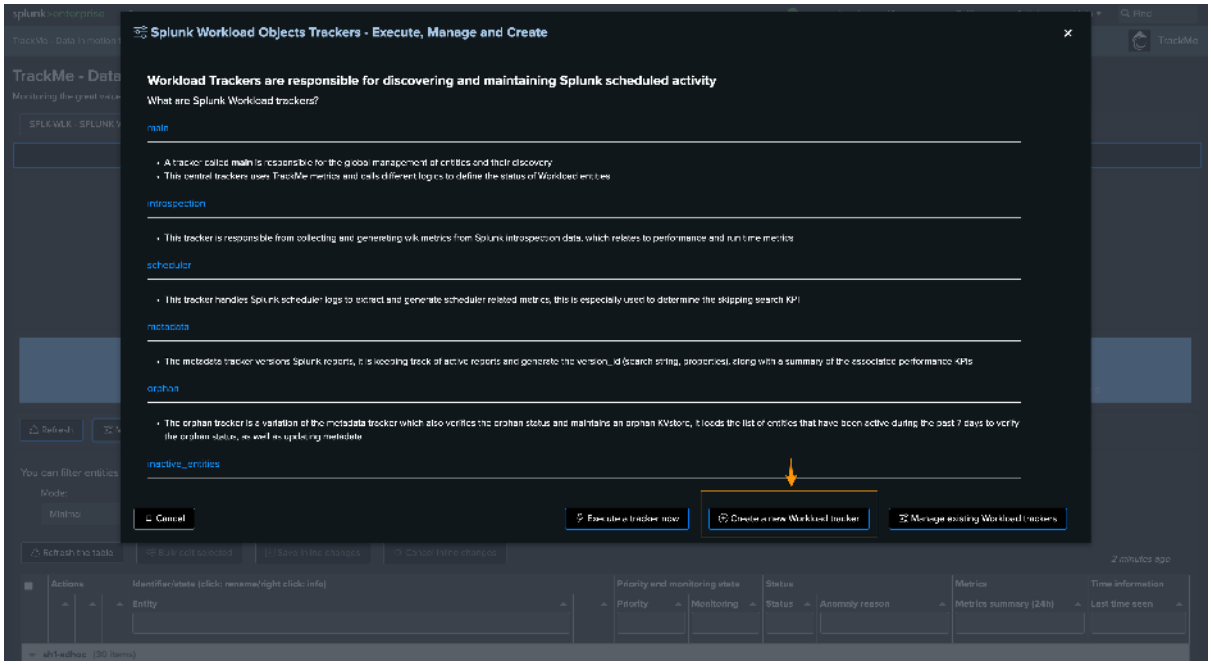
Note

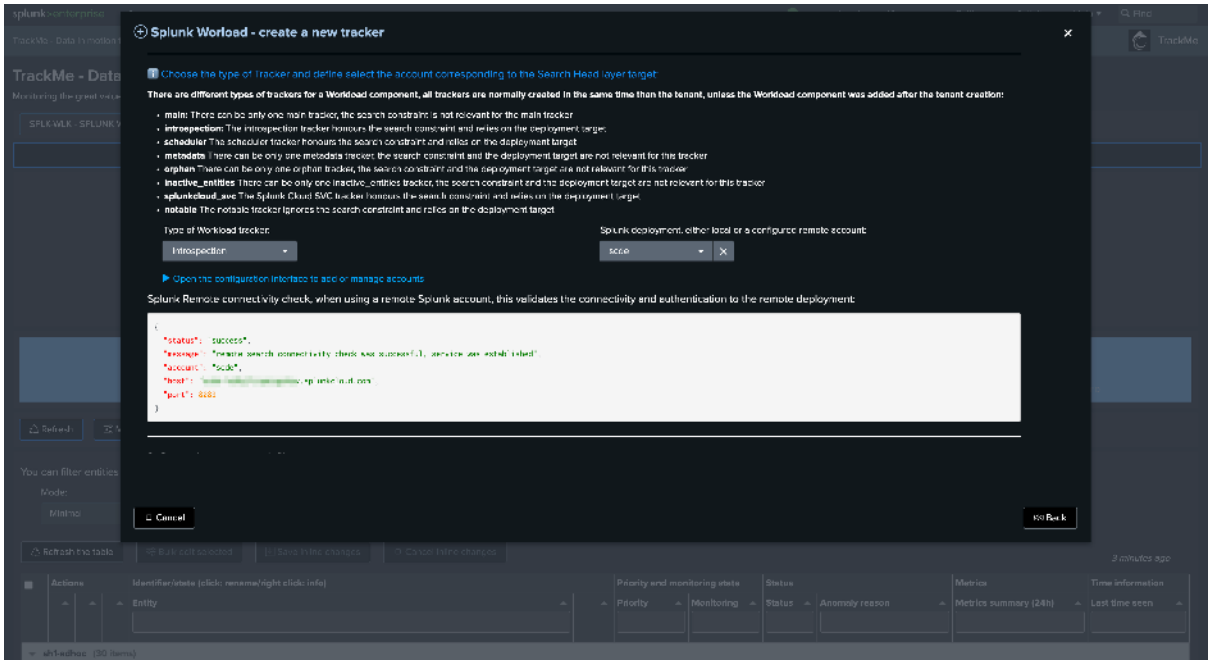
Common trackers:

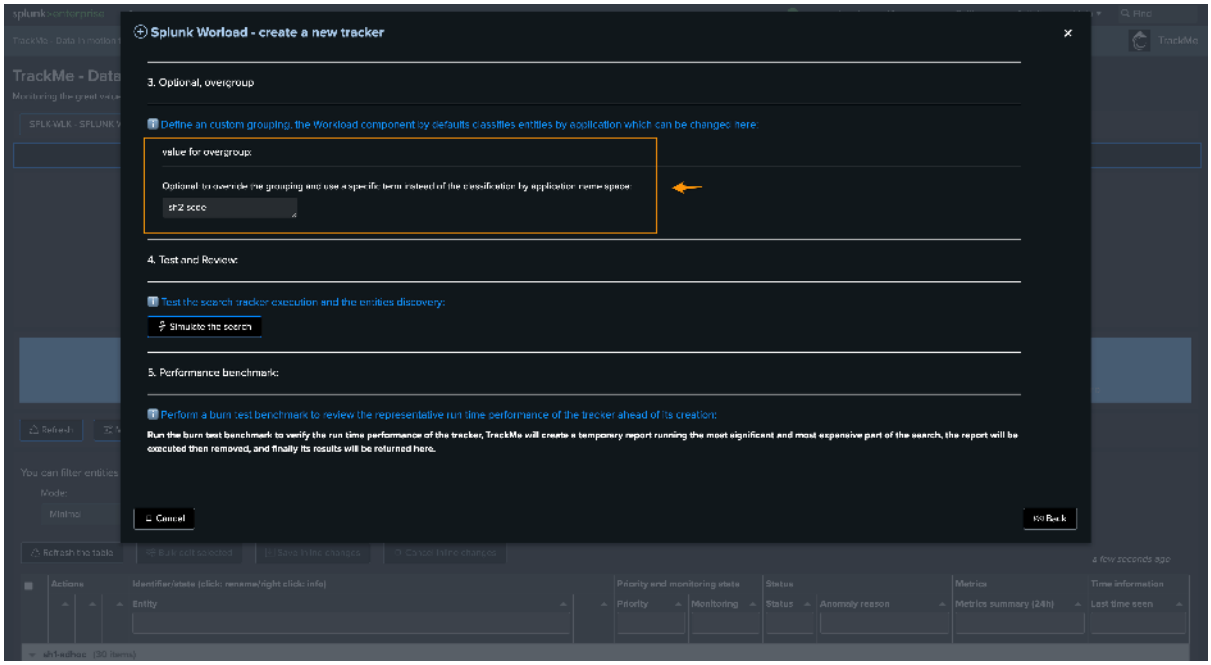
- You do not need to create additional trackers for main, metadata and inactive entities
- There can be only one copy of these trackers per tenant, and these trackers can manage entities accordingly as long as these entities refer to a different account than the one used when creating the tenant
- Therefore, we can manage multiple Search Head tiers in a same tenant as long as each Search Head tier corresponds to a different account (which in fact will be the case systematically in the Workload component)



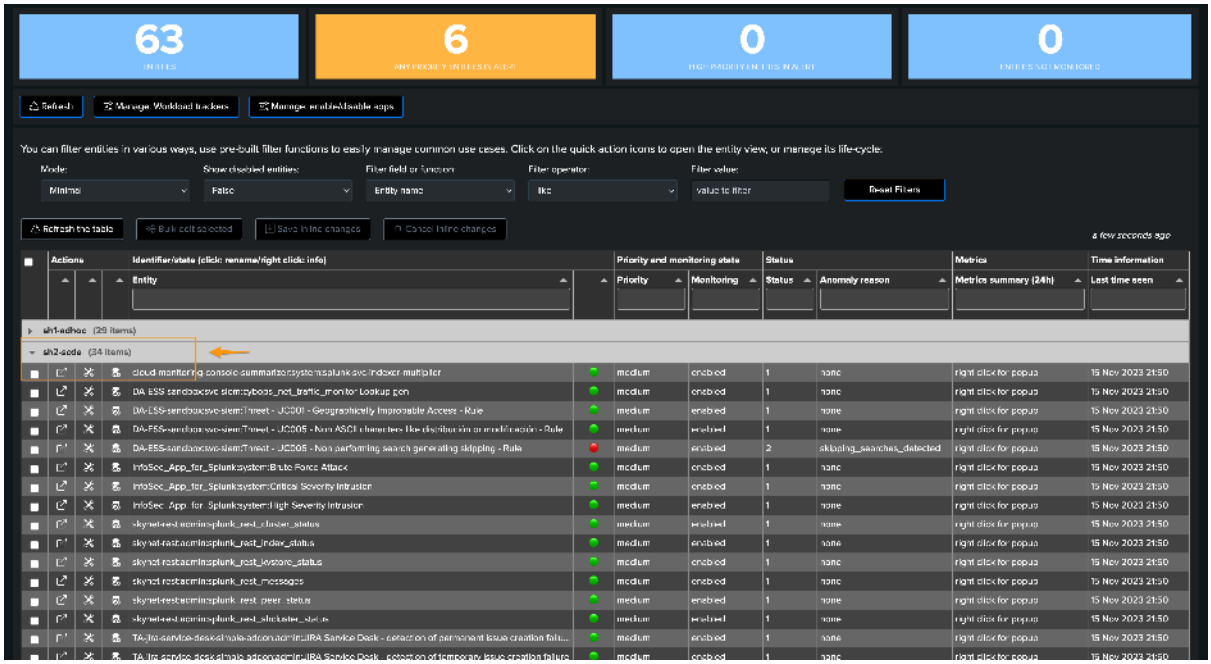
For instance, we will create trackers for a second Search Head tier targeting a remote account and will name the overgroup `sh2-scde`:



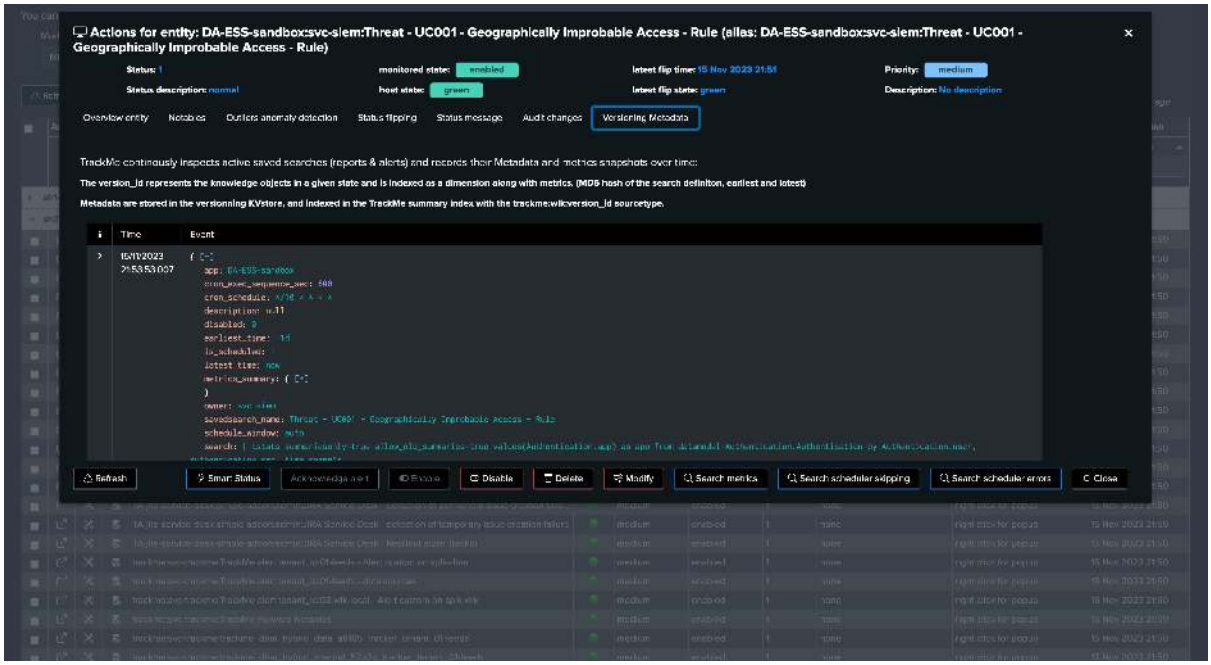




After 5/10 minutes, entities are now also created under the second overgroup as expected:



Metadata are managed transparently depending on their parent account, so TrackMe knows which Search Head tier to query for:



Workload monitoring — concepts & usage

The following material covers the Workload component from the user's perspective — what it discovers, how to read its anomaly reasons, metrics and status messages, and the investigation workflow.

Note

TrackMe Workload component

- This feature is part of TrackMe's restricted features and allows tracking the key value activity and KPIs of your Splunk scheduling, from alerts to scheduled reports and **Enterprise Security correlation searches**.
- We track and alert on sophisticated concepts such as the detection of **execution anomalies** or **delayed**

execution, **consumption behavior** changes using **Machine Learning** as well as **version control** and changes in the searches themselves.

- The **TrackMe Workload component** is extremely powerful and provides the missing deep visibility on your critical Splunk workflow, whether you are using Splunk for security or IT operations, or any other use case.

Hint

Evolutions in TrackMe 2.1.28

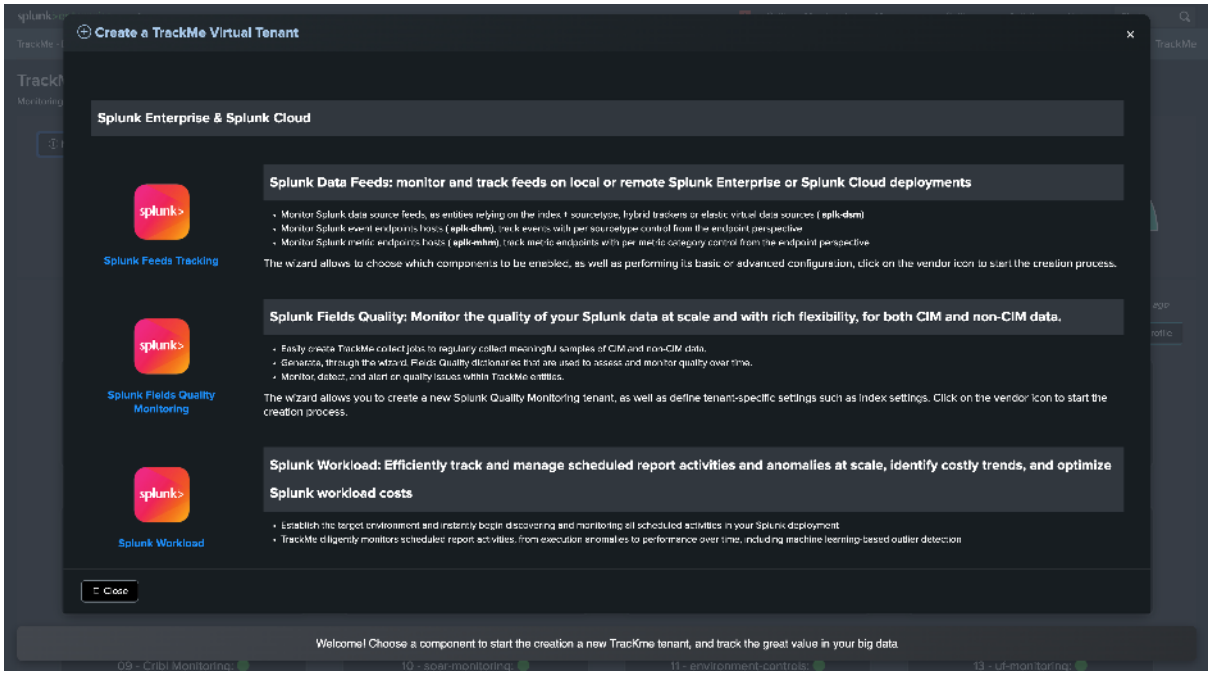
- Since TrackMe 2.1.28, you can now control the list of parameters that are taken into account to calculate the version_id hash.
- To do so, go in Configure / splk-general and update the `splk_general_workload_version_id_keys` parameter.
- The default is to take into account the following parameters: search,dispatch.earliest,dispatch.latest,description,cron_schedule

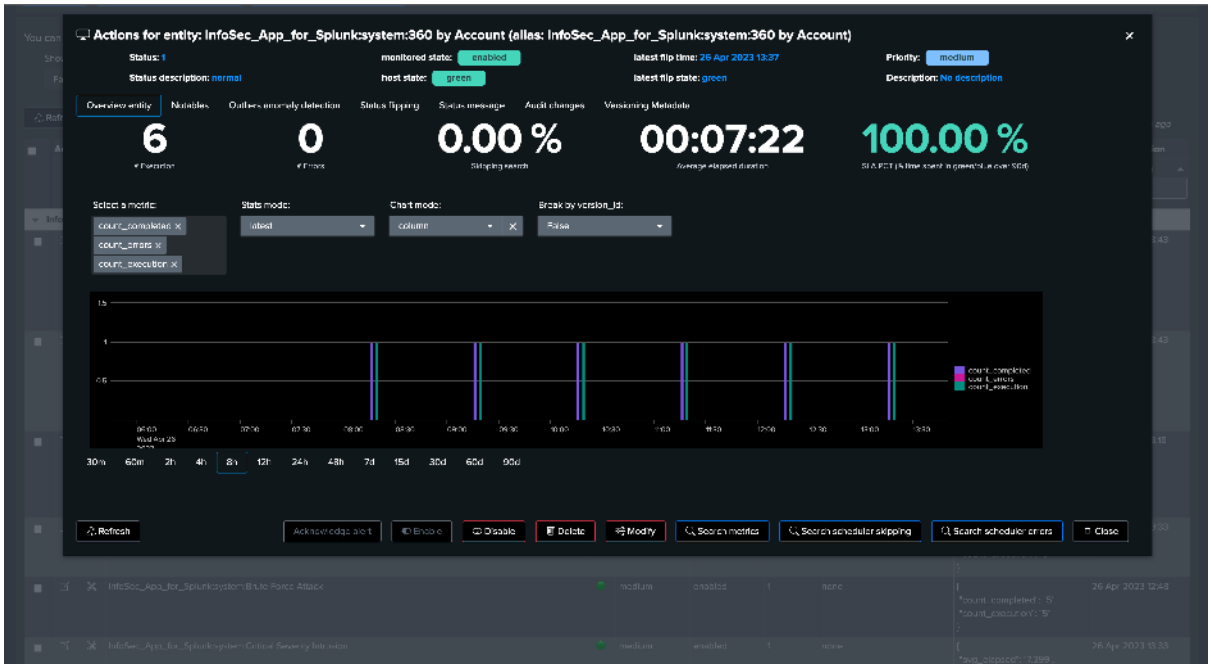
1. Introduction

The objective of the Splunk Workload component for TrackMe is to continuously track the scheduling activity of a Splunk deployment, and perform key activity and detection:

- Discover Splunk scheduled activity across the environment; an entity will be the association of the Splunk application namespace, the owner, and the search name
- Track the scheduled activity behavior, and detect issues that can affect the well-being of Splunk scheduled reports and alerts
- Track and identify the scheduled objects' Metadata, such as their definition, to investigate and detect changes impacting searches' behaviors over time
- Generate consistent and key value metrics from Splunk introspection, scheduler and SVC usage for Splunk Cloud customers
- Use Machine Learning Outliers detection to track for abnormal behaviors and changes, such as an abnormal increase in costs, run time or consumption
- Alert when scheduled reports and alerts are affected by issues, and get benefit from the full set of TrackMe's features and workflow to handle the lifecycle of Splunk scheduled activity

The TrackMe Workload component, associated with TrackMe's unique workflow, is a game-changing companion for Splunk, allowing it to quickly detect and alert on Splunk's most critical activity, as well as providing the keys to understand scheduled related costs like never before.


Create a TrackMe Virtual Tenant
Splunk Enterprise & Splunk Cloud
Splunk Data Feeds: monitor and track feeds on local or remote Splunk Enterprise or Splunk Cloud deployments
Splunk Feeds Tracking
Splunk Fields Quality: Monitor the quality of your Splunk data at scale and with rich flexibility, for both CIM and non-CIM data.
Splunk Fields Quality Monitoring
Splunk Workload: Efficiently track and manage scheduled report activities and anomalies at scale, identify costly trends, and optimize Splunk workload costs
Splunk Workload
Close
Welcome! Choose a component to start the creation a new TrackMe tenant, and track the great value in your big data


Actions for entity: InfoSec_App_for_Splunk:system:360 by Account (alias: InfoSec_App_for_Splunk:system:360 by Account)
6
0
0.00 %
00:07:22
100.00 %
Refresh
Close

Note

Grouping in the Workload component: group and overgroup

- By default, the Workload component groups entities by `app` which represents the Splunk application namespace hosting the scheduled report or alert.
- When creating the tenant and also later on when creating trackers manually, since TrackMe 2.0.70, you can optionnally override this behaviour by setting up an `overgroup` value.
- By doing so, the Workload will group entities based on a custom term instead of applications, this can be useful if for instance you want to have multiple Search Head tiers in the same tenant.
- Refer to the *Creating a Workload tenant to host multiple Search Head tiers with overgroup* section for more information.

2. Workload entities

Once a Workload TrackMe Virtual Tenant has been configured, TrackMe starts to track for any scheduled activity and will create and maintain associated entities:

An entity name is composed of:

- `app` + ":" + `owner` + ":" + savedsearch name

Should any of this information change, TrackMe will consider this as a new entity to be created and maintained, for instead of a search reassignment, if the search is moved between different application name spaces, or if the search knowledge object identifier is changed.

3. Anomaly reason

3.1 Anomaly reason definition

TrackMe considers each entity individually, and will trigger a status change based on the following criteria:

Anomaly Reason	Conditions
none	The entity is green and is healthy, there are no issues detected currently
skipping_searches_de	TrackMe detected skipping executions for that scheduled, by default orange<5% and red>5%
orphan_search_det	A previously active scheduled search is now orphaned, which means the owner is not valid any longer and the search cannot be executed
execution_errors_detec	Execution errors are detected and the scheduled report/alert is not working properly
anomaly_outliers	The Machine Learning outliers engine detected issues in one or more active ML models, this can be for instance an abnormal increase in the schedule runtime
execution_delayed	TrackMe uses the Metadata and the cron schedule translation to monitor if the search is delayed compared to its expected schedule (with 5 min of grace time by default)
status_not_met	The status of the entity is red, for unclassified reason
out_of_monitorir	The status of the entity is not healthy, however the current period is out of the monitoring window set for this entity

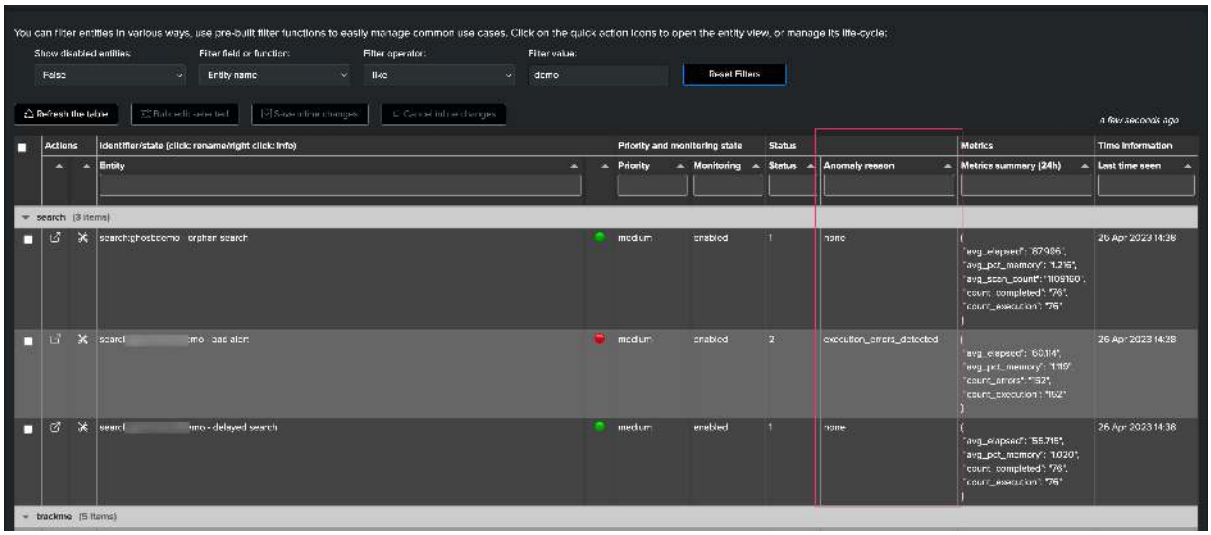
3.2 Checking the Anomaly Reason

You can observe the current anomaly reason in different locations:

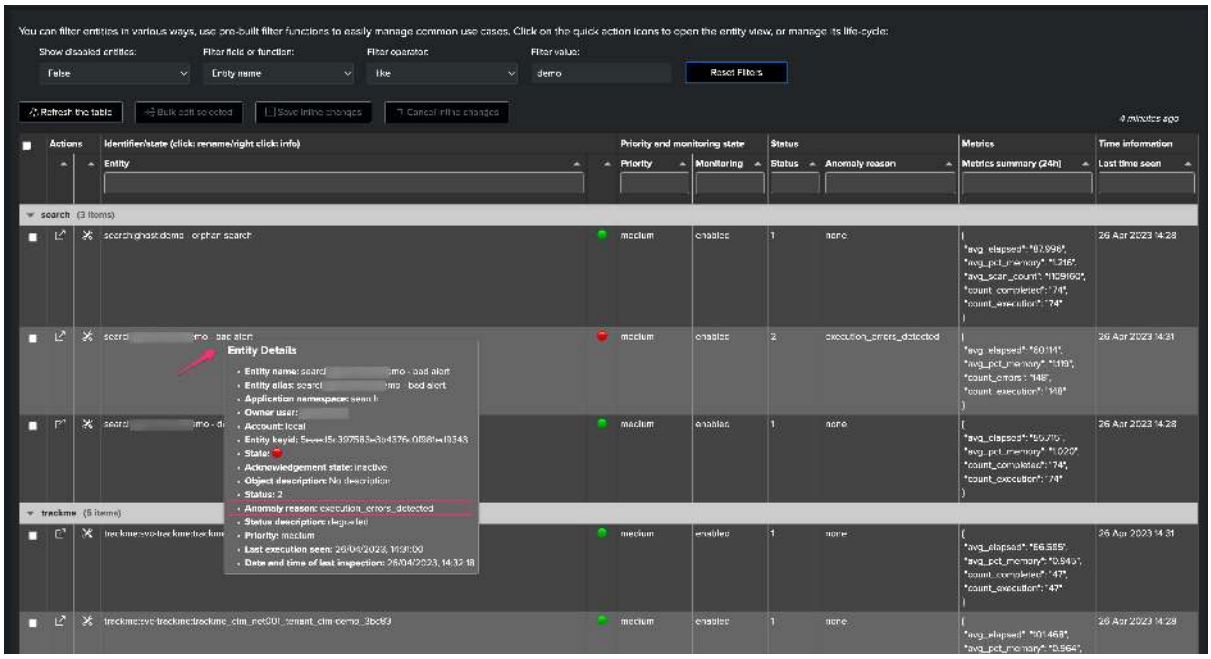
- In the table column called "Anomaly Reason"
- By right clicking on the entity, in the contextual menu
- When the alert is sent over, as part of the fields resulting from the alert

- When the alert fires, as part of the TrackMe notable event

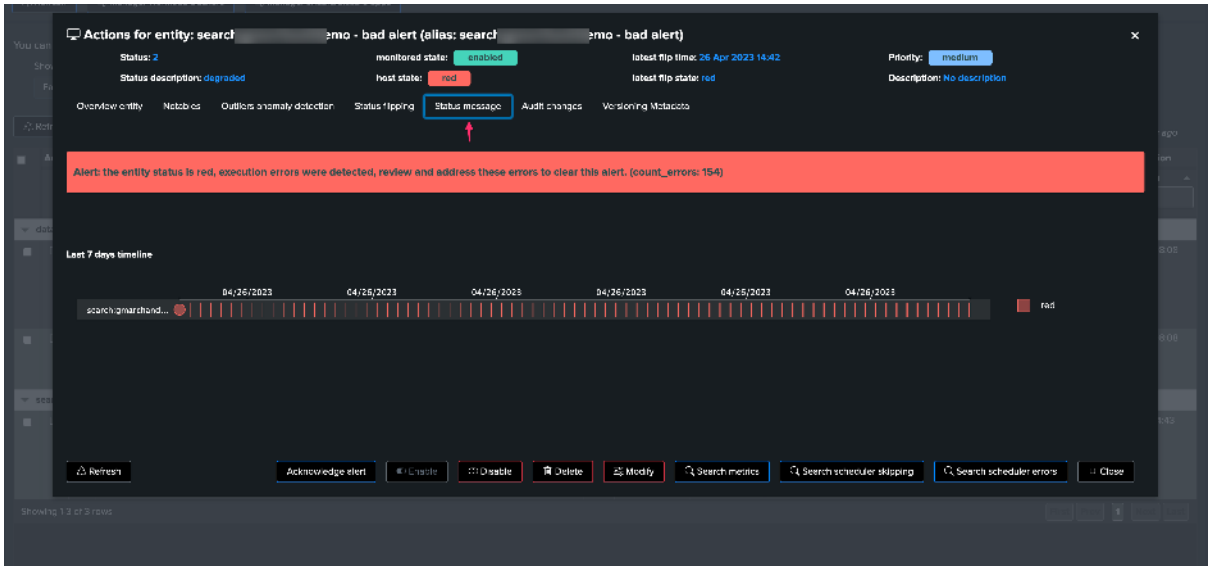
table anomaly reason column:



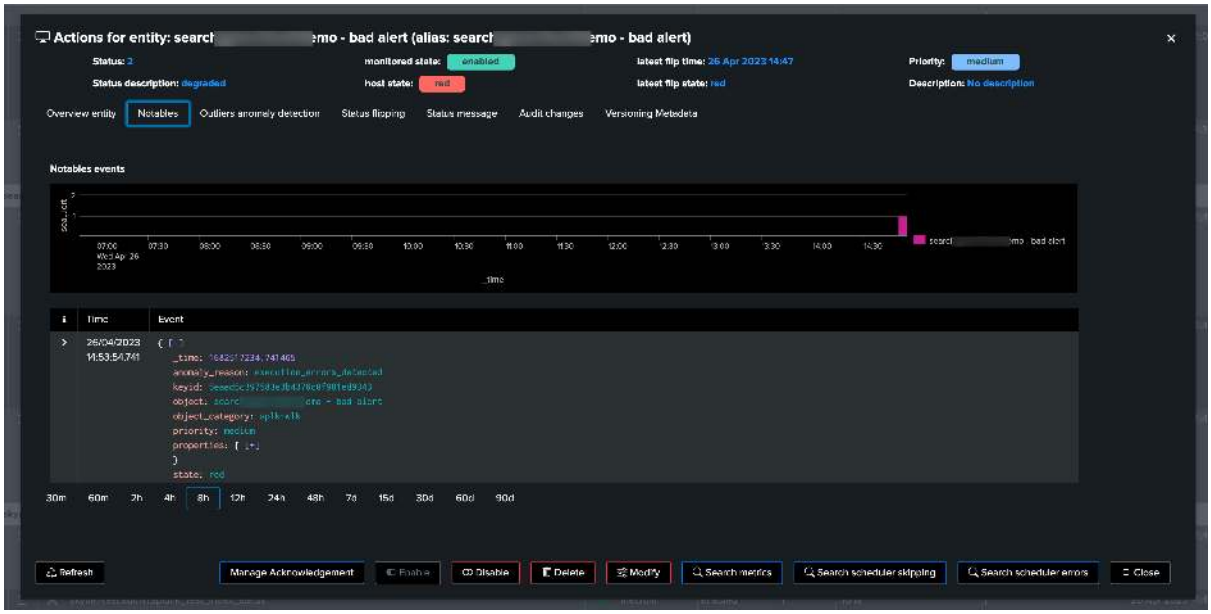
right click contextual menu on the entity name:



The status message is conditioned by the anomaly reason value and translated into a detailed message:

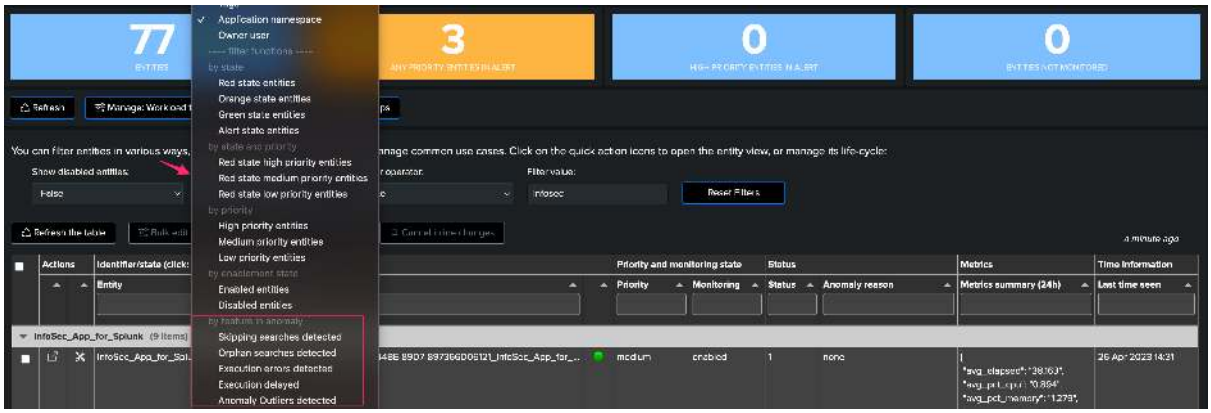


The anomaly reason is part of a notable event when an alert triggered:



3.3 Filtering on Anomaly Reasons

You can use the filter functions to check all entities with a given anomaly reason:



4. Workload metrics

Once a scheduled entity was discovered, TrackMe tracks its activity and generates various key metrics:

Metric name	Description
count_executior	Scheduler metric: the number of requested executions for this schedule
count_complete	Scheduler metric: the number of successfully completed executions for this schedule
count_skipped	Scheduler metric: the number of skipped executions for this schedule
count_errors	Scheduler metric: the number of executions errors detected for this schedule
elapsed	Introspection metric: the run time of the search, in seconds
pct_cpu	Introspection metric: the aggregated percentage of CPU used for this schedule (which can be more than 100%)
pct_memory	Introspection metric: the aggregated percentage of Memory used for this schedule (which can be more than 100%)
scan_count	Introspection metric: the number of events scanned for this schedule
svc_usage	Splunk Cloud metric: the SVC usage for this consumer

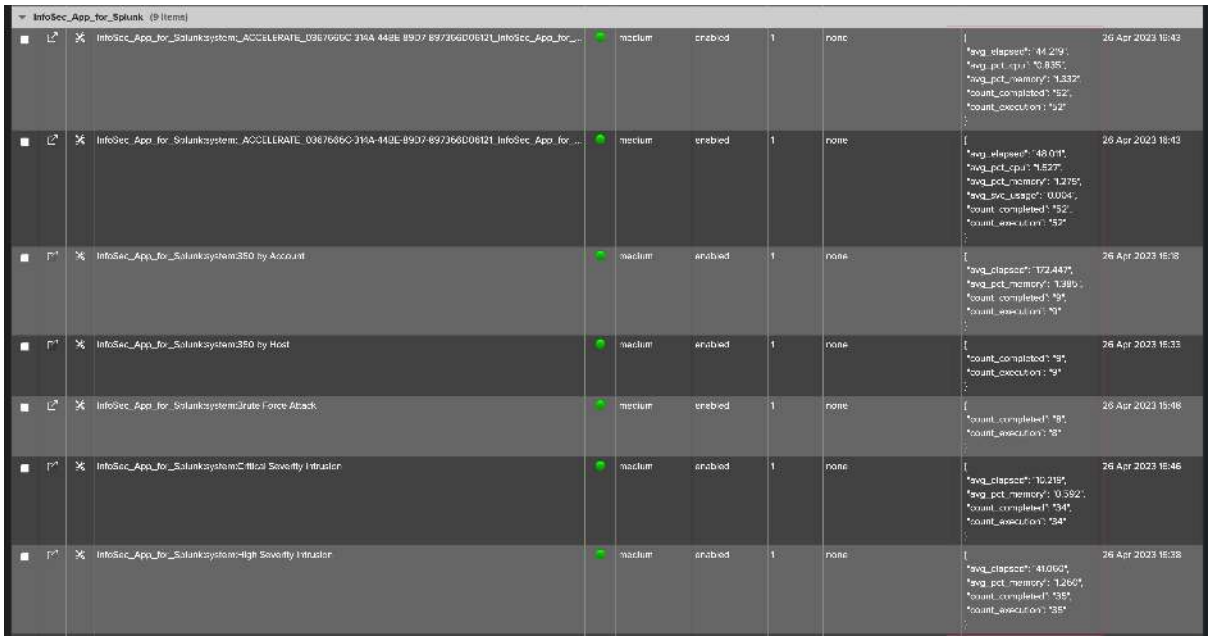
The Workload metrics are then used to condition the status of the entity, and feed the Machine Learning Outliers engine.

4.1 Accessing metrics

4.1.1 Metrics summary table

Metrics summary in the table:

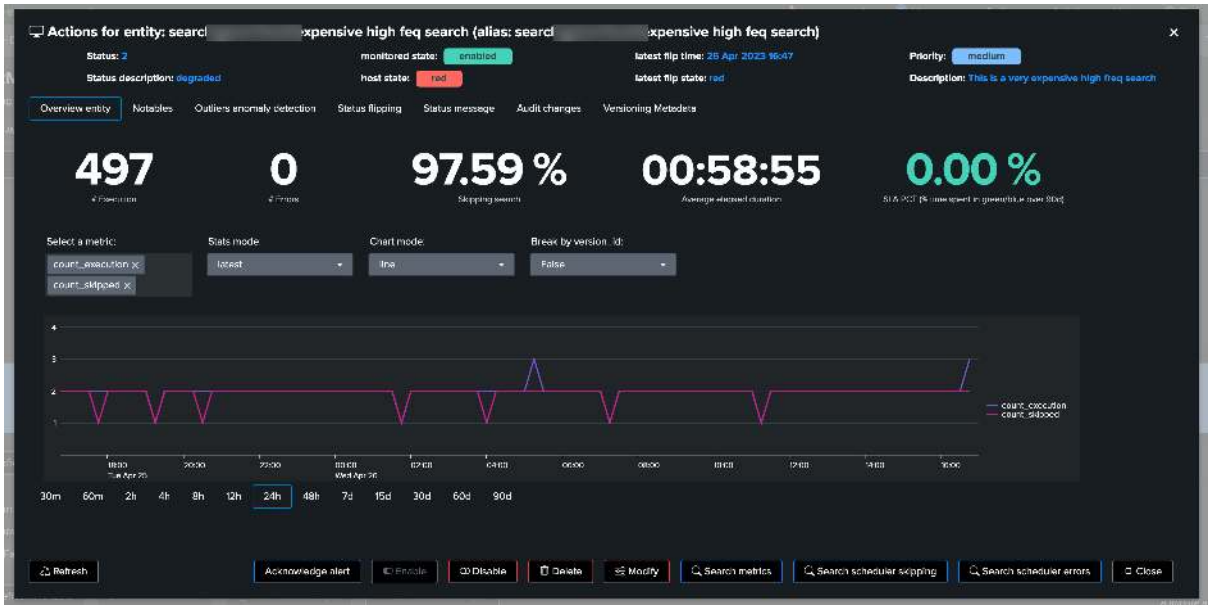
TrackMe shows a summary of the metrics of the last known 24 hours per entity:



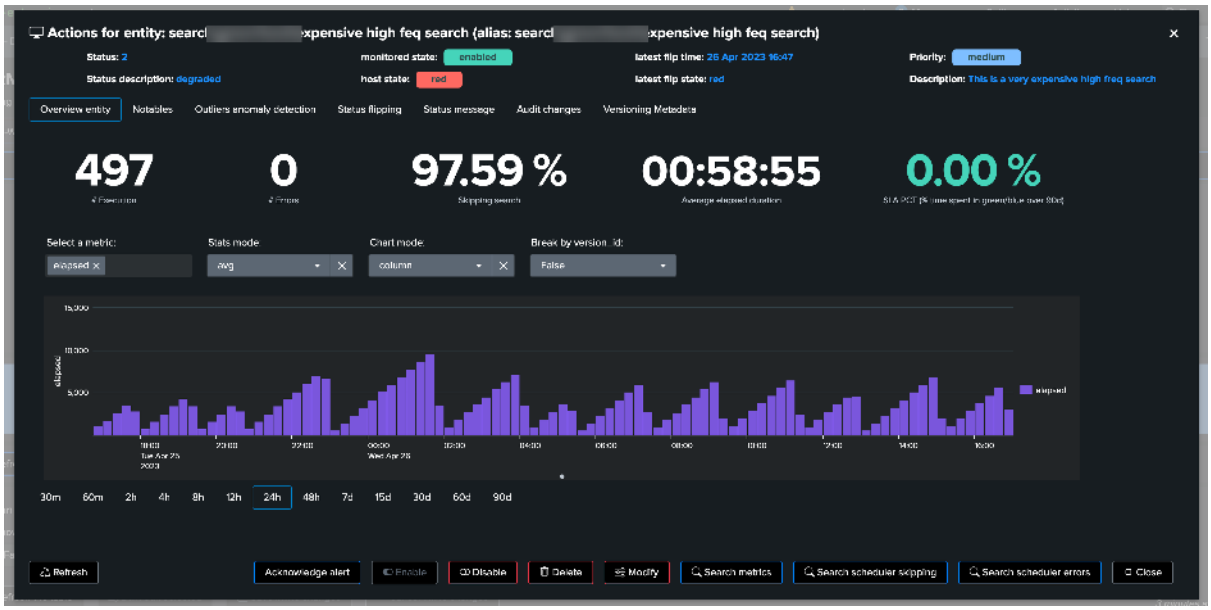
Note that metrics shown in the summary JSON vary depending on the metrics available, for instance an entity which has not been experiencing skipping searches will not show the metric as it is null.

4.2.2 Metrics chart over time

You access to the metric chart selector by opening an entity overview:



Use the metric selector, and optionally other settings according to your needs, in the following examples, we are looking at the average elapsed for that entity:



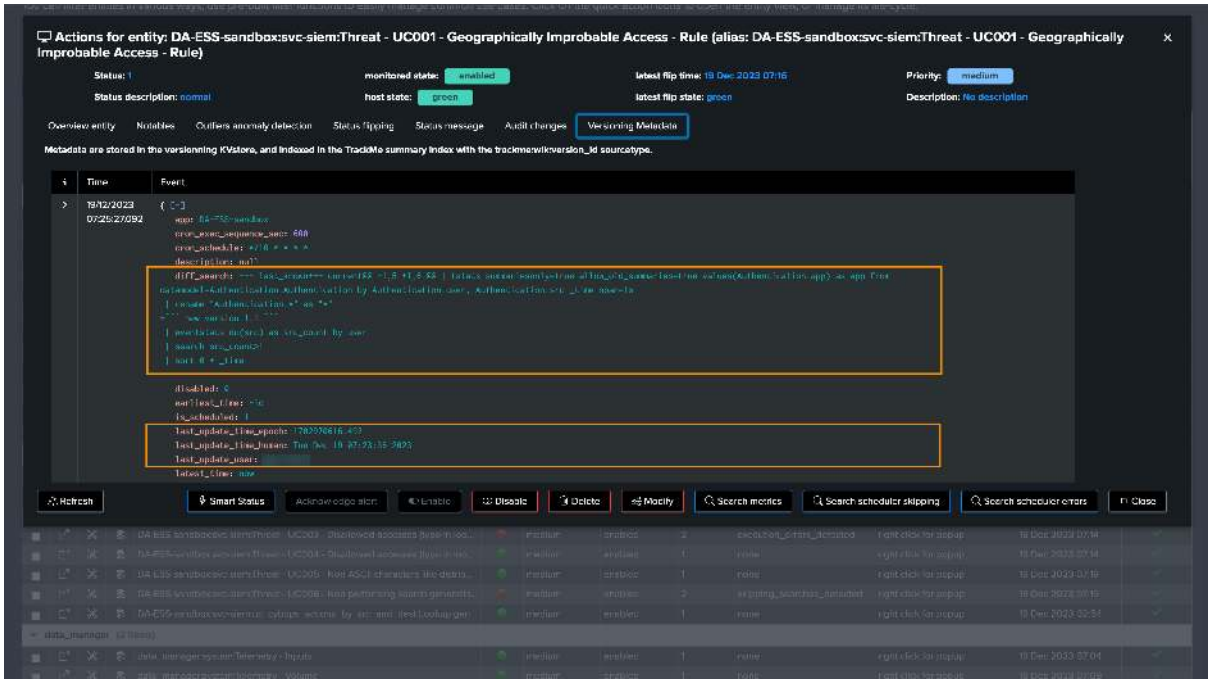
5. Metadata Versioning

TrackMe monitors the versioning of scheduled entities, to detect changes and allow linking a knowledge object state with its performance metrics.

Hint

Diff search/earliest/latest (since TrackMe 2.0.72)

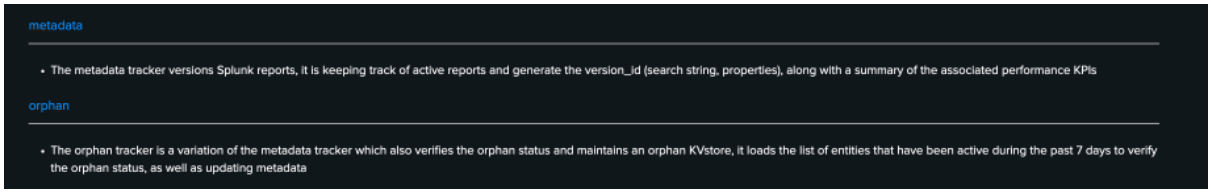
- Since TrackMe 2.0.72, the diff of the search and/or earliest/latest quantifiers is performed and stored as part of the versioning events and records
- This means that TrackMe automatically identifies the difference of the search when a change is detected against the previously known version, stored as part of the `diff_<context>` fields
- It will also attempt to identify the user that performed the change and when this change occurred (if the change occurred via Splunk Web/Splunk API, not application configuration push)



5.1 How the Metadata Versioning works under the cover

This activity is performed by two associated scheduled TrackMe trackers:

- metadata
- orphan

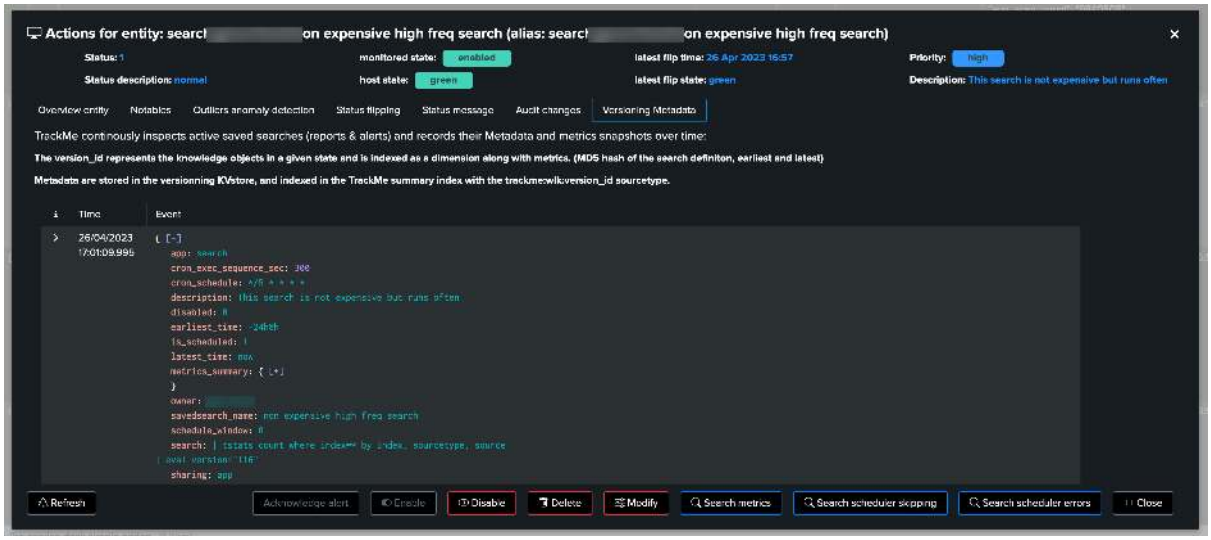


Via a Python based integration, TrackMe maintains the knowledge of scheduled entities, and identity a certain state as an MD5 hash which is composed by:

- The search code definition, basically the whole SPL statement whatever its complexity is
- The earliest time quantifier
- The latest time quantifier

5.2 Accessing Metadata information via the UI

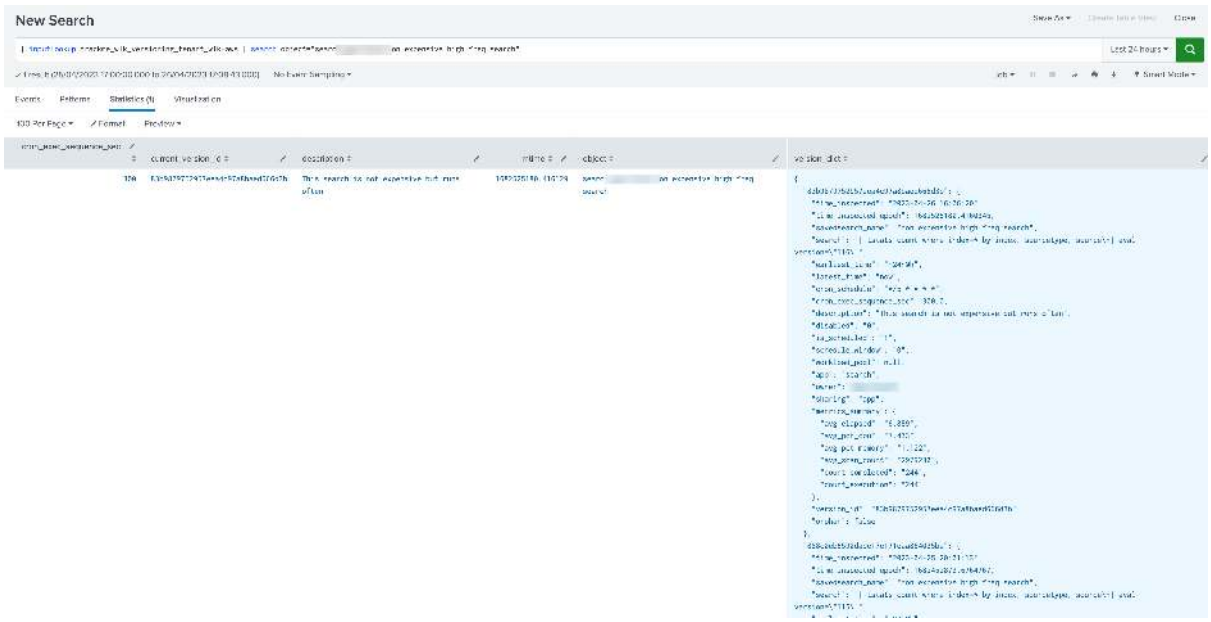
Regularly, active entities are inspected, if a change is detected, a new version_id is defined, a new set of Metadata are extracted and stored in a KVstore collection:



5.3 Accessing Metadata information via the KVstore

The Metadata version is stored in a JSON structure within the KVstore record associated with a given entity, in a persistent fashion:

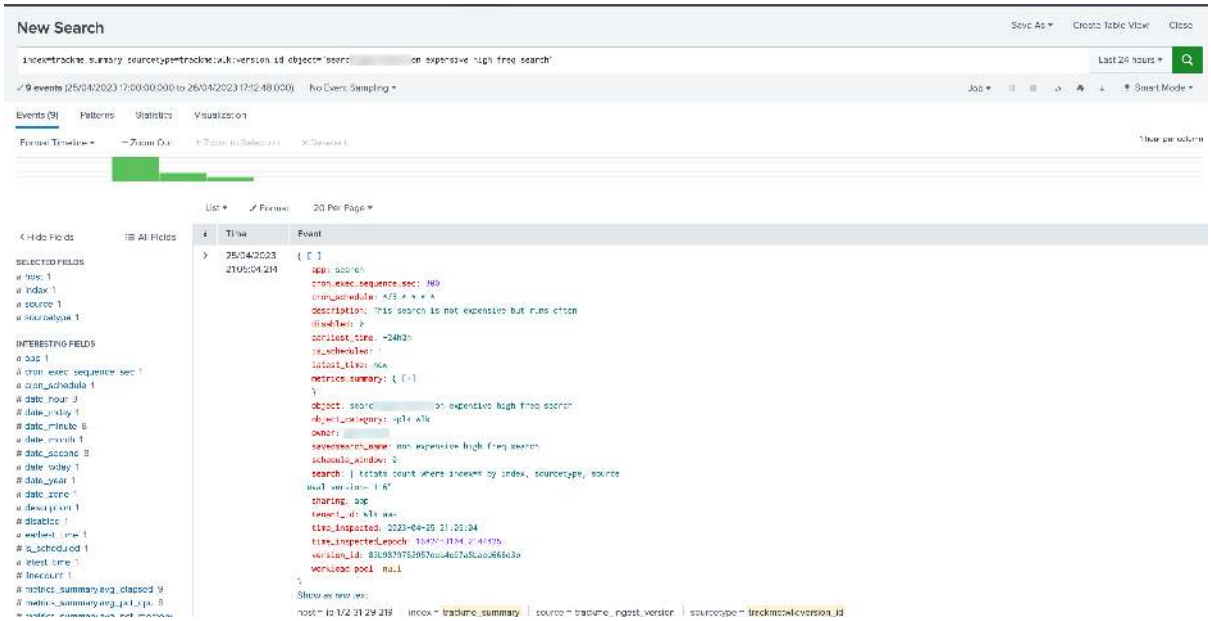
```
| inputlookup trackme_wlk_versioning_tenant_<replace with tenant_id> | search␣
↪object="<replace with object name>"
```



5.4 Accessing Metadata information via TrackMe indexed events

When TrackMe detects a new version of a monitored scheduled entity, it will as well generate an event in the trackme_summary index of the tenant, with the sourcetype:

```
index=trackme_summary sourcetype=trackme:wlk:version_id tenant_id="<replace with␣
↪tenant_id> object="<replace with object name>"
```

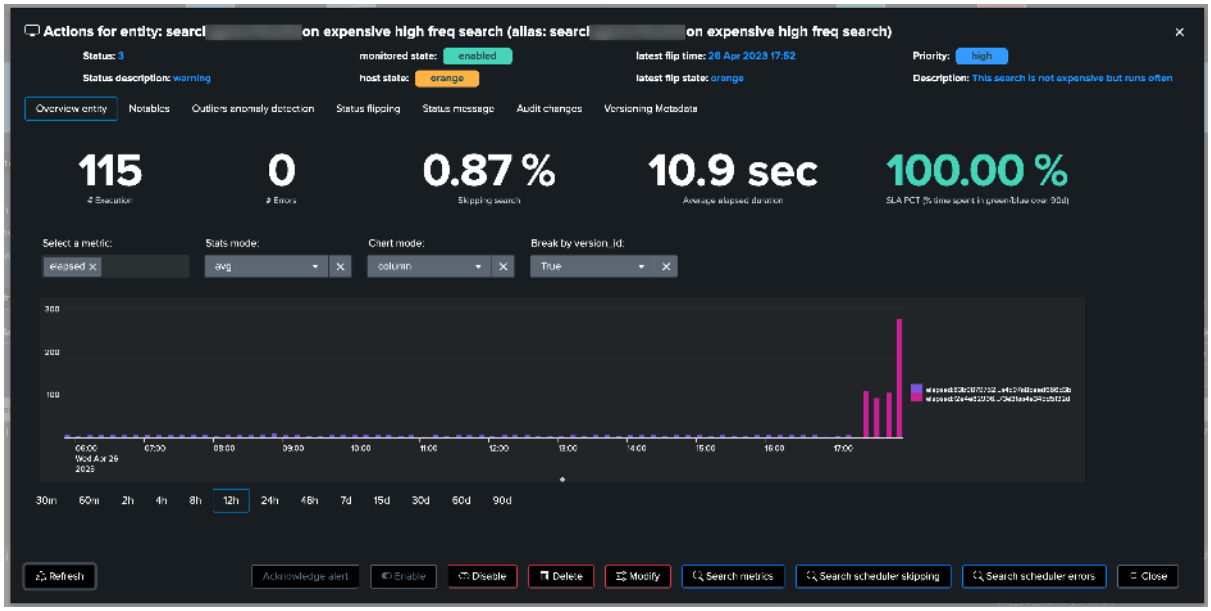


5.5 Use cases for versioning

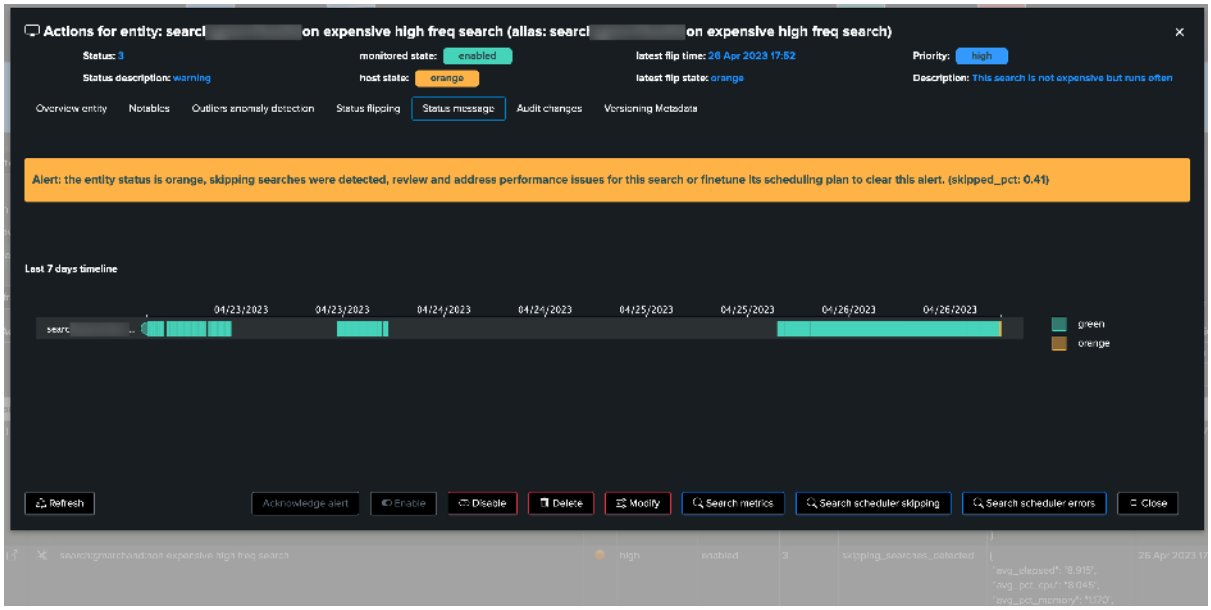
So, what are the use cases for the versioning TrackMe feature, associated with TrackMe's workflow, incident management, metrics generation and so forth? There are many valuable use cases of course.

For instance, it becomes fairly easy to link a change in the search logic to a massive increase of its run time or computing costs, potentially suddenly leading to alerts in TrackMe caused by skipping searches.

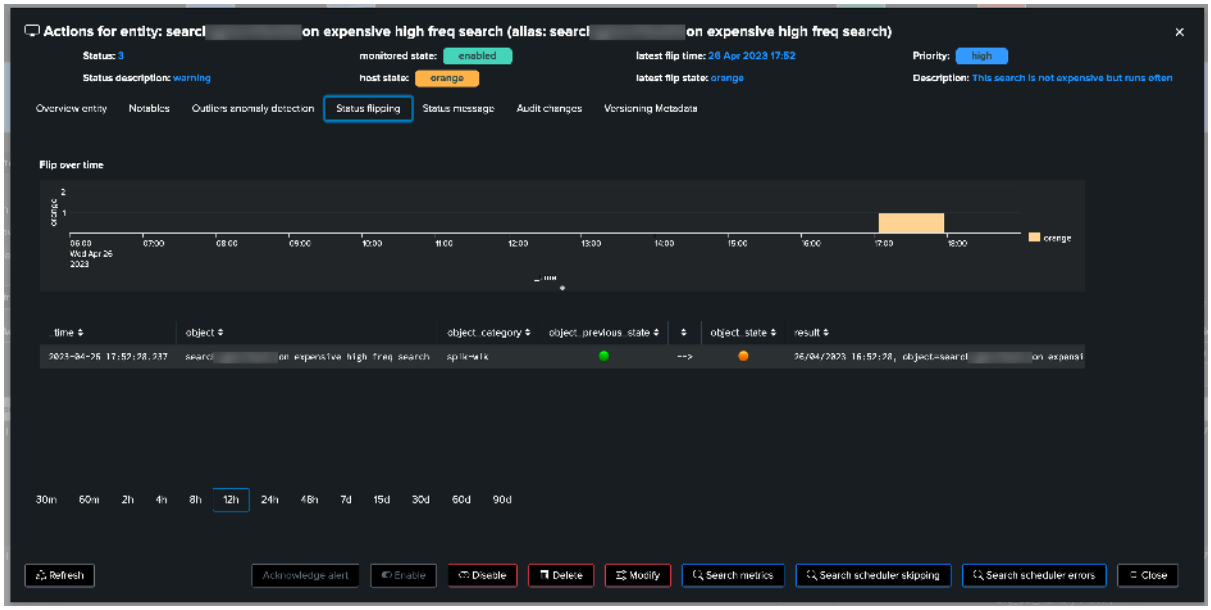
When accessing the metrics overview, use the break down selector to add the version_id as part of the break by statement:



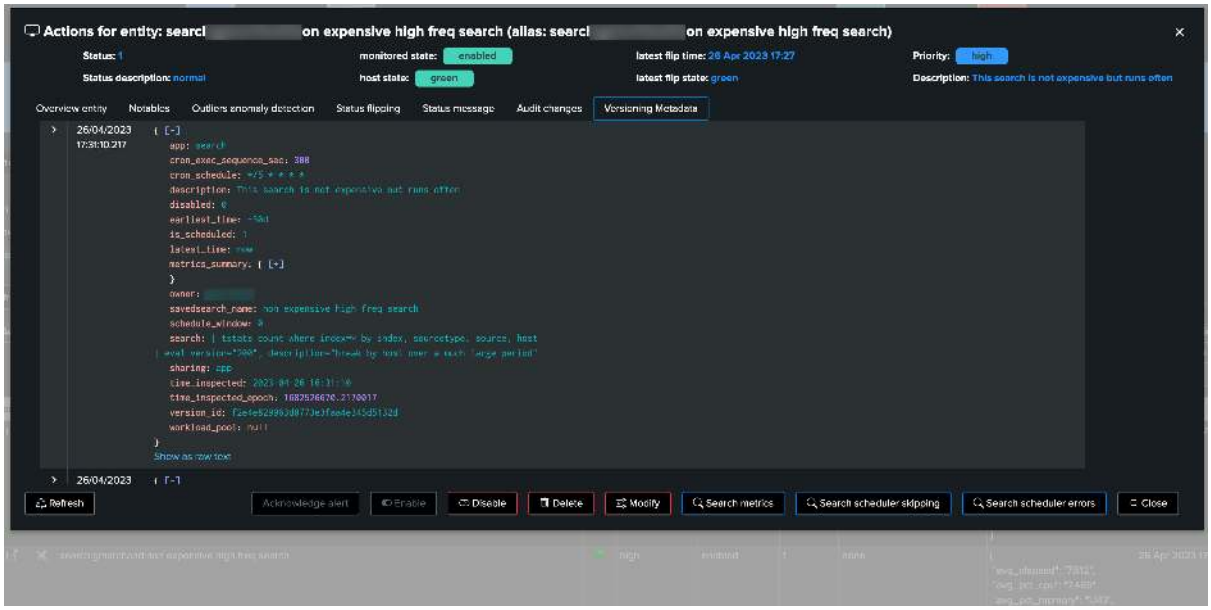
Perhaps TrackMe started to detect bad behaviour of the scheduled object, for instance starting to generate a certain level of skipping searches:



Wait what, don't we have information about flipping statuses? We also know the "when":



As we can see in the chart, a clear increase in in the elapsed (run time in seconds) became suddenly visible, which we have been able to easily associate with a change of the Splunk knowledge object itself:



A search which was very efficient suddenly starts consuming a huge amount of resources, potentially impacting the functional use case itself, perhaps even the platform and starting to cost real money to the company, with the need for additional resources and so forth!

Use cases for the Workload features and the versioning are legions, its value grows exponentially along with your deployment size, scale and maturity.

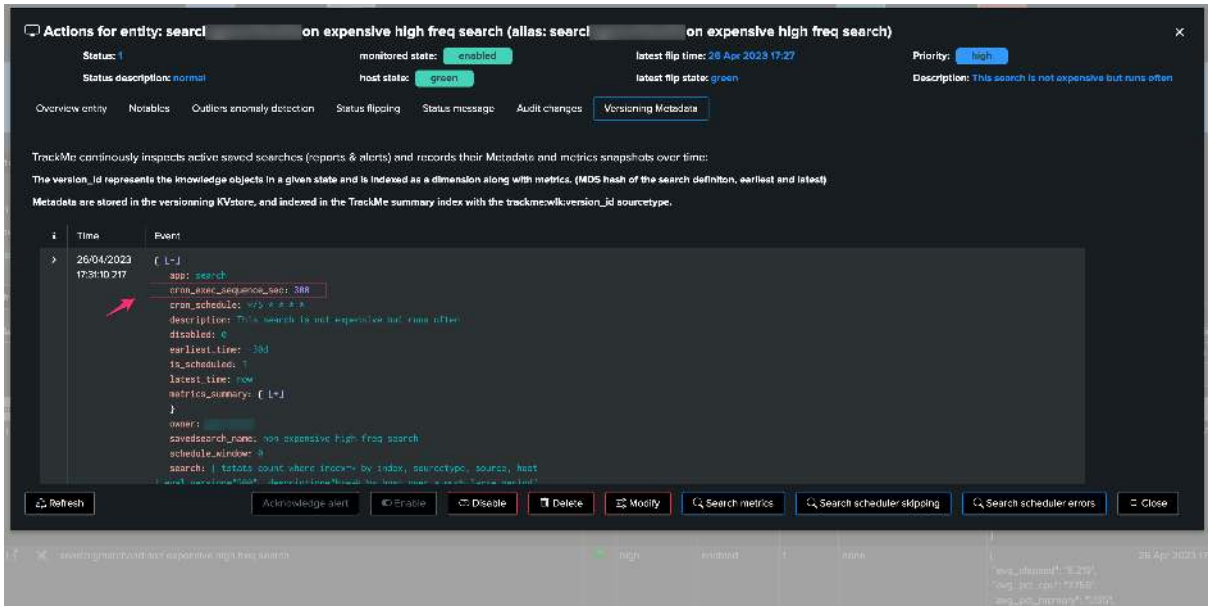
6. Delayed searches execution detection

As part of the Workload component, TrackMe verifies that scheduled entities have been executing properly according to their requested schedule.

A delayed search execution can have severe implications, from a functional perspective this can mean security events which are not considered properly for instance, with all the range of potential consequences.

This verification is called "execution_delayed" anomaly reason status, and works the following way:

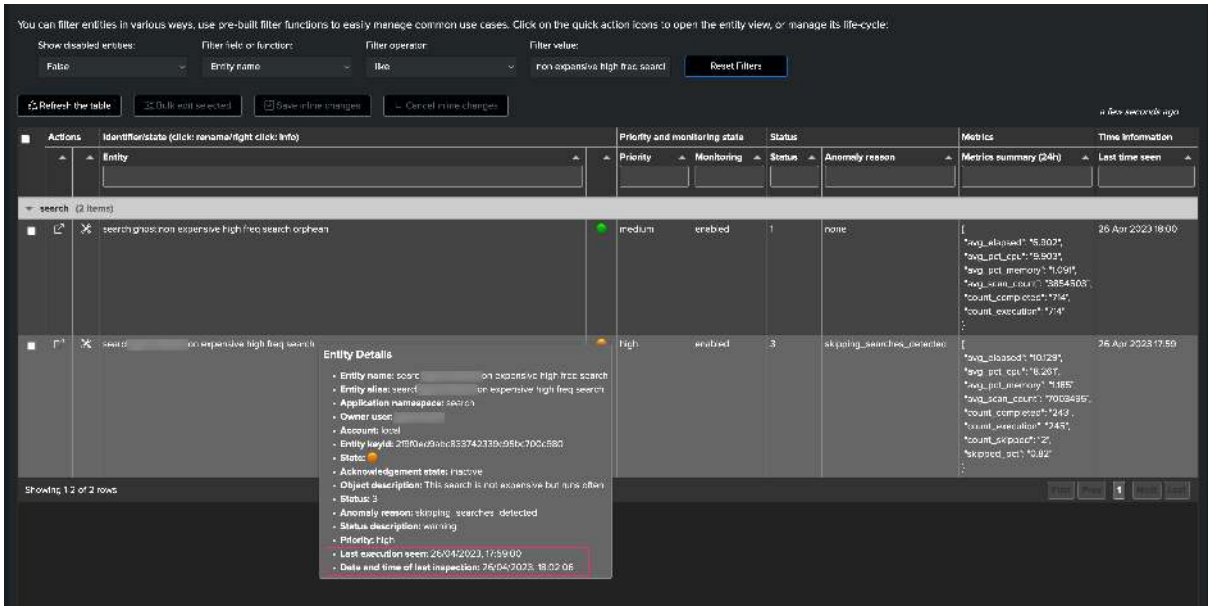
- When the versioning components inspect the Metadata of an entity, it retrieves the cron schedule expression too, which defines how often a scheduled is going to be scheduled
- This cron schedule is transformed into a per seconds value defining how often the search should have been executing (using the croniter Python library), and stored in the field `cron_exec_sequence_sec` along with the metadata record of the entity



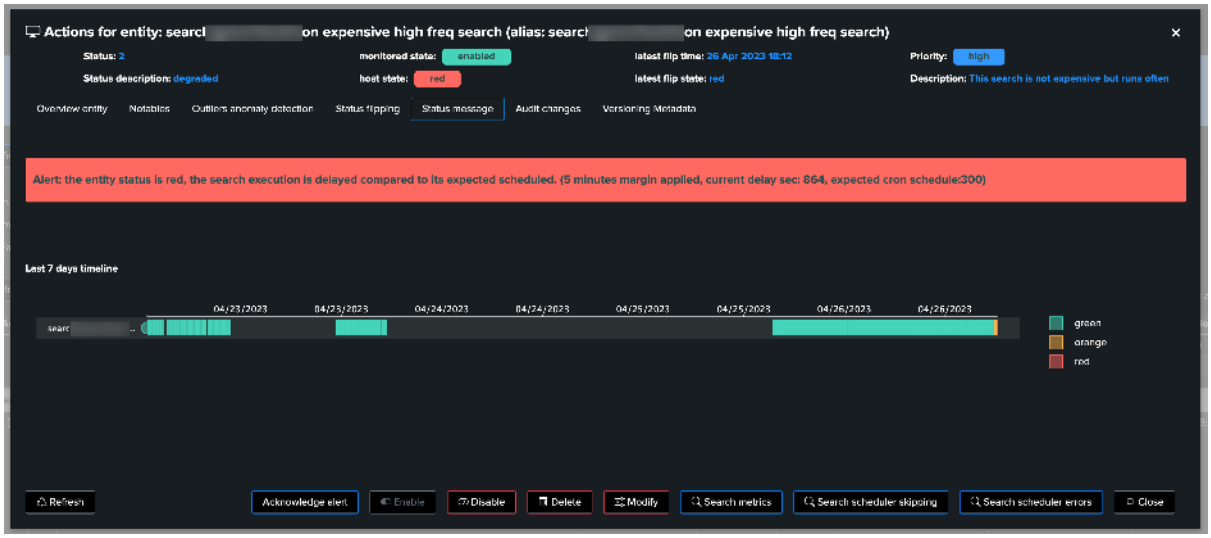
Finally, TrackMe verifies and stores the scheduled successful execution traces, if a search which should have executed at least once in the past 5 minutes has not been since active since more than 5 minutes, plus the grace period, then TrackMe can consider this entity as unhealthy and trigger and alert accordingly,

For instance, let's disable the scheduling of an active schedule (a mistake? bad behaviour?), after a short period of time, TrackMe detects the conditions which immediately impacts the entity, generates an alert and TrackMe notable!

we can observe the dates of latest inspection and latest execution in the contextual menu:



After some minutes, TrackMe detected that the scheduled search is delayed:

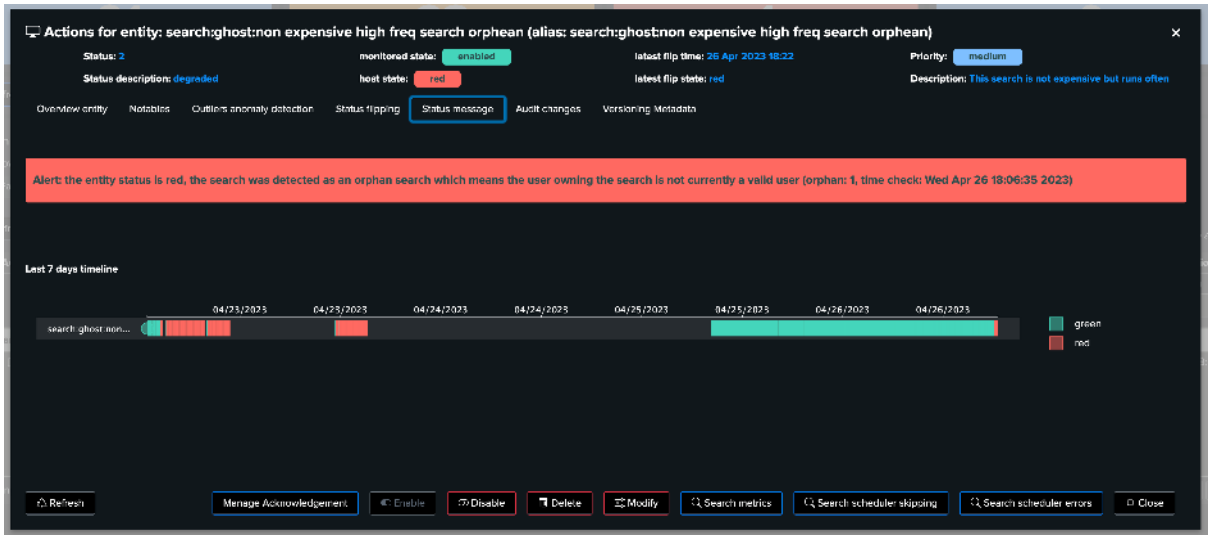


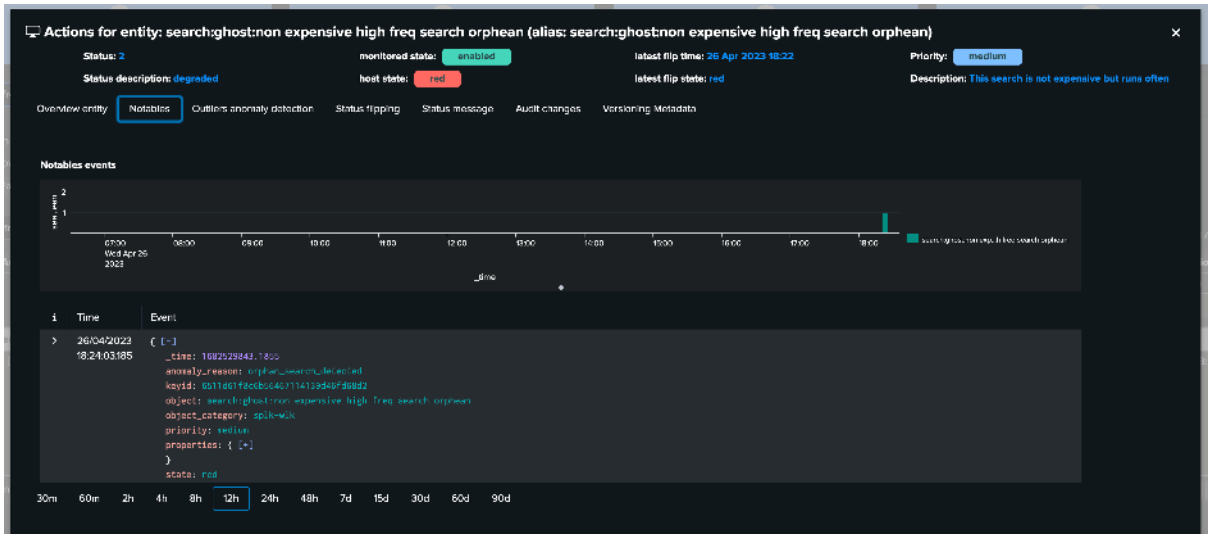
From this stage, we automatically detect an abnormal condition for our critical scheduled activity, and we can start acting accordingly before it starts having any severe consequences!

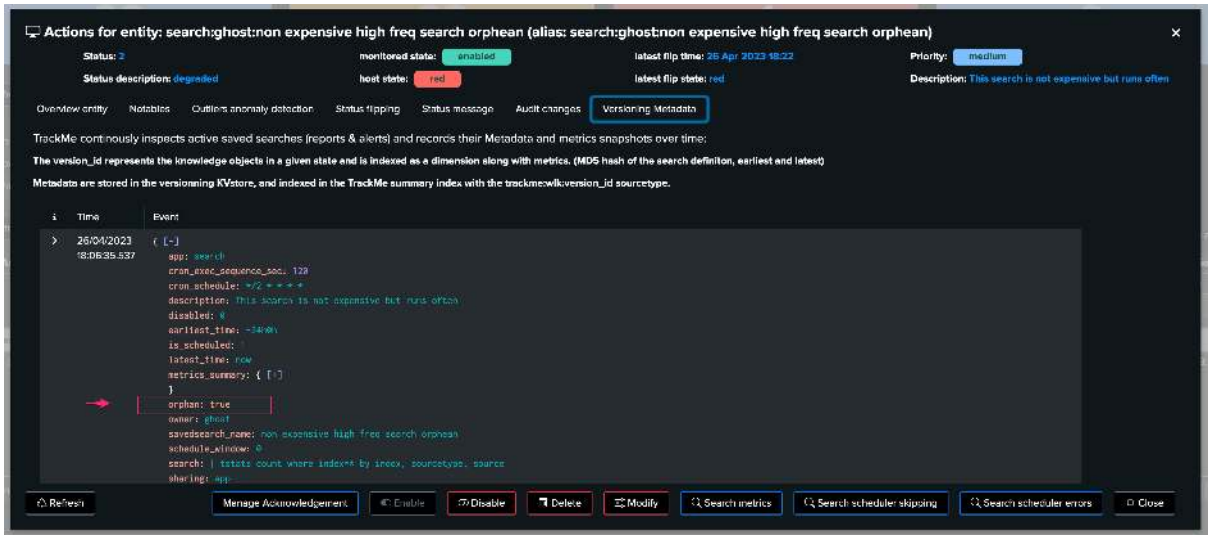
7. Orphan searches detection

Orphan search condition is also detected by TrackMe, for any search that has been actively discovered, in short:

- Once per hour, TrackMe verifies via the orphan tracker the current status of all active scheduled objects for the past 7 days
- It updates the Metadata information with the `orphan` boolean
- In return, TrackMe as part of its condition verifies that the search has not been detected as an orphan search, which otherwise will be considered as a critical condition for this entity



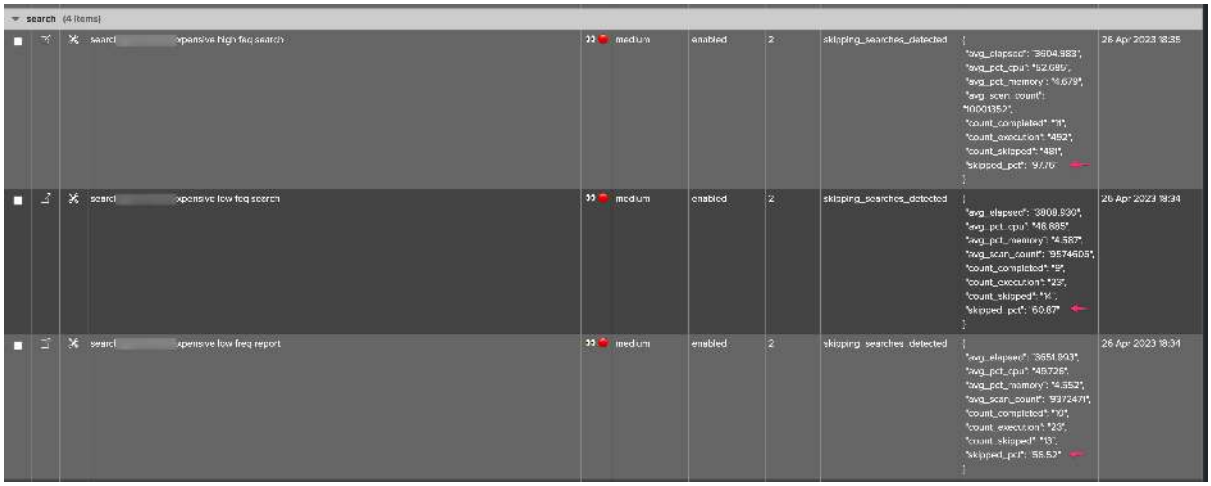


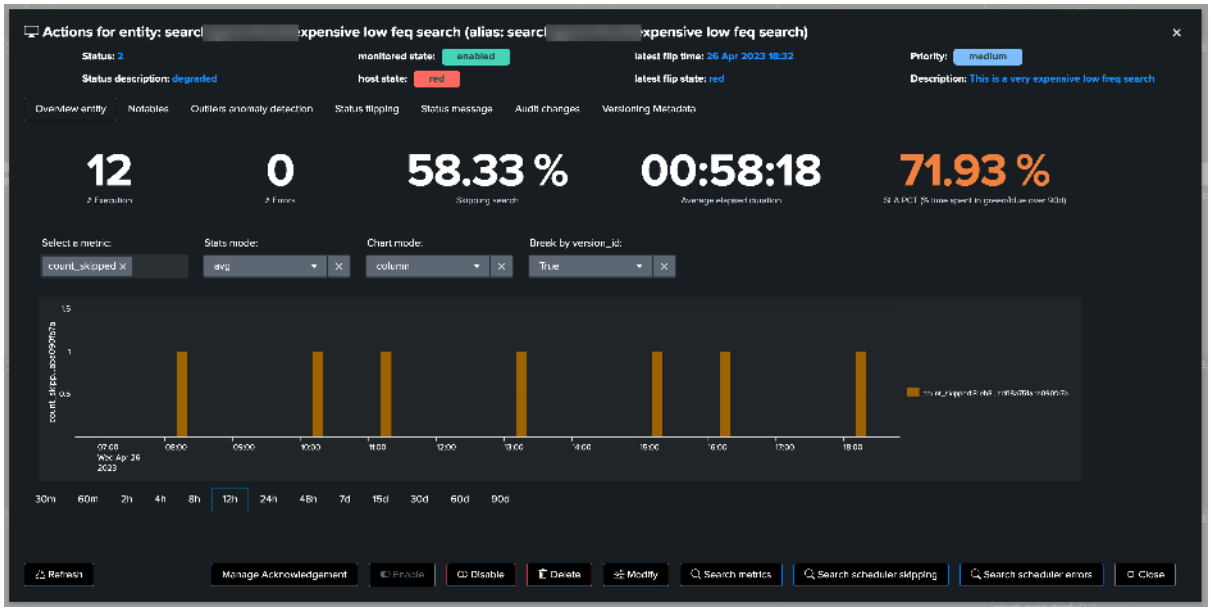


8. Skipping searches detection

Skipping search condition is achieved by comparing the collected count_execution and count_completed collected metrics, which gives the skipping search percentage:

- If the skipping search percentage is somewhere between 0 to less than 5%, the entity status will be **orange**
- Over 5% of skipping search percentage, the entity will turn **red**
- The skipping percentage can be seen in the summary metrics JSON in the table, or via the metric over time inspector

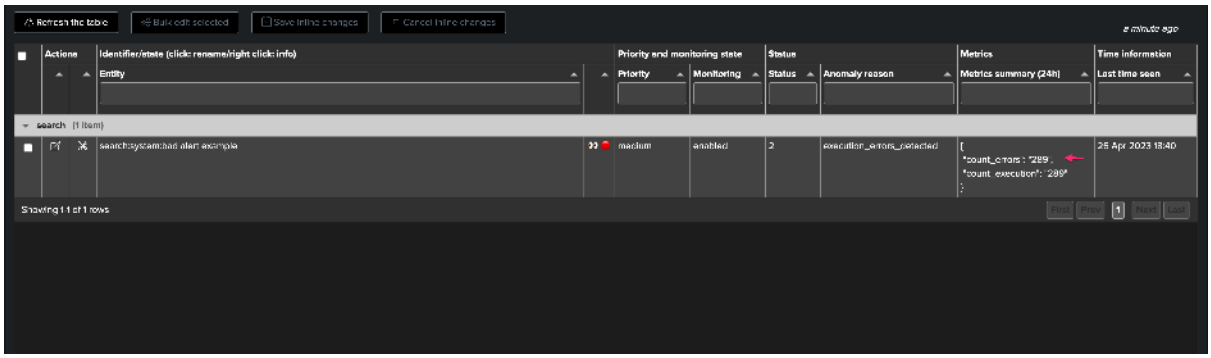


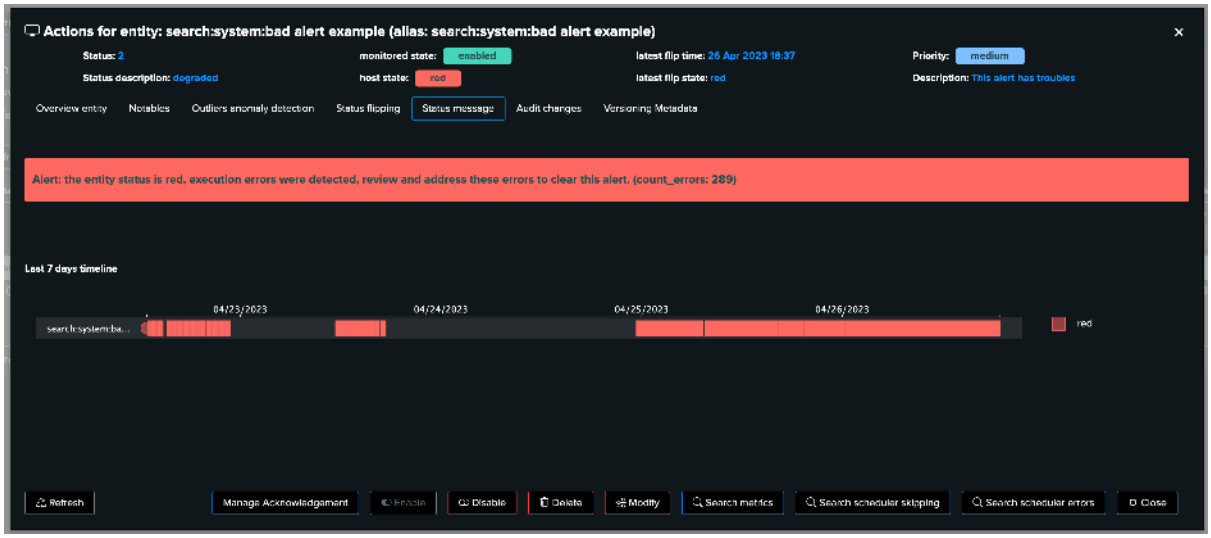


9. Error executions detection

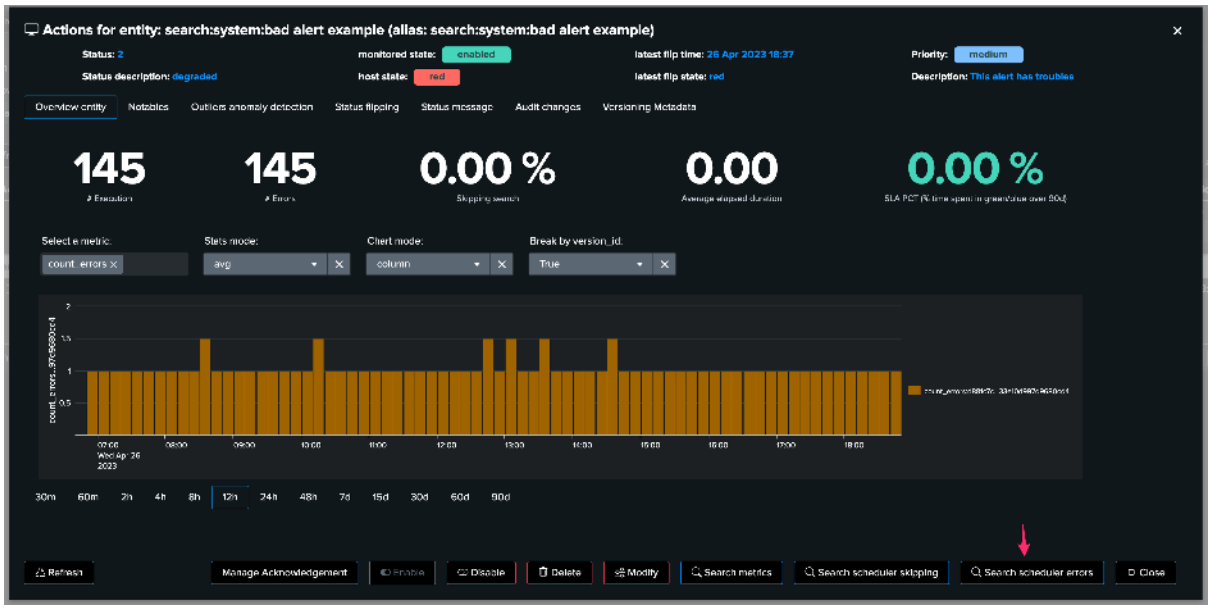
Detecting at scale and automatically execution errors is another challenge TrackMe tackles in the Workload component:

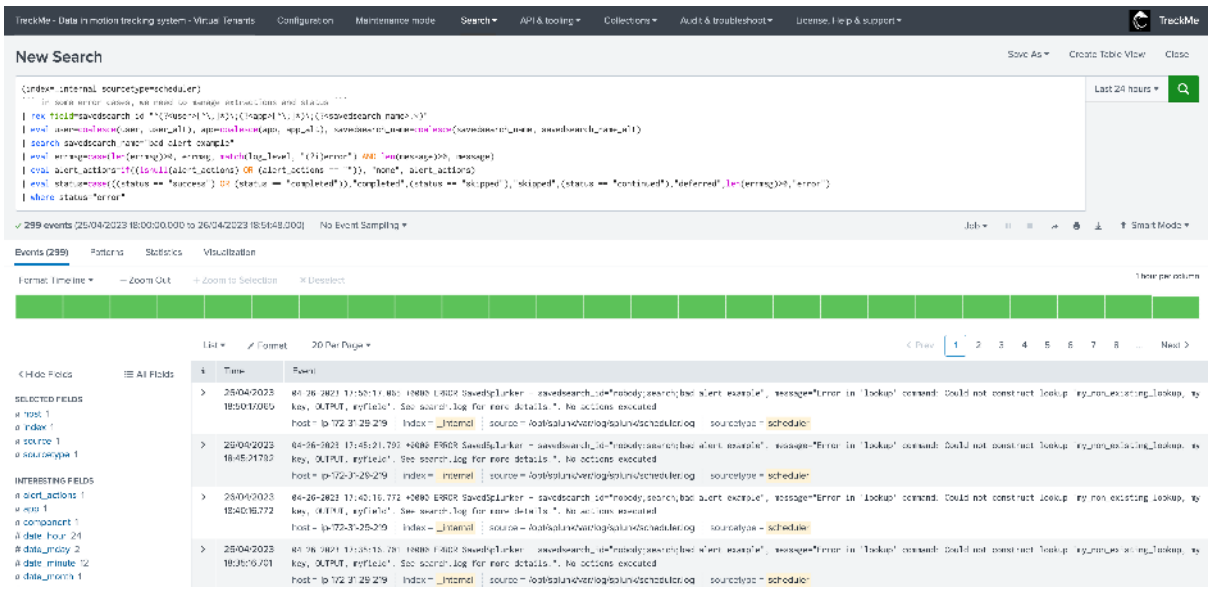
- Execution errors can happen for all sorts of reasons, for instance a related knowledge object (macro, lookup etc.) which is not available to the search logic
- Errors in the application development, lack of qualification, unexpected deployment side effect, etc. many combinations can lead to a scheduled search to be failing pretty much silently
- TrackMe handles the challenge by continuously looking at the scheduler activity, and extracting the executions errors turning these into a metric which we can take into account





From the overview (currently looking at the metrics), you can as well directly access to the scheduler errors:



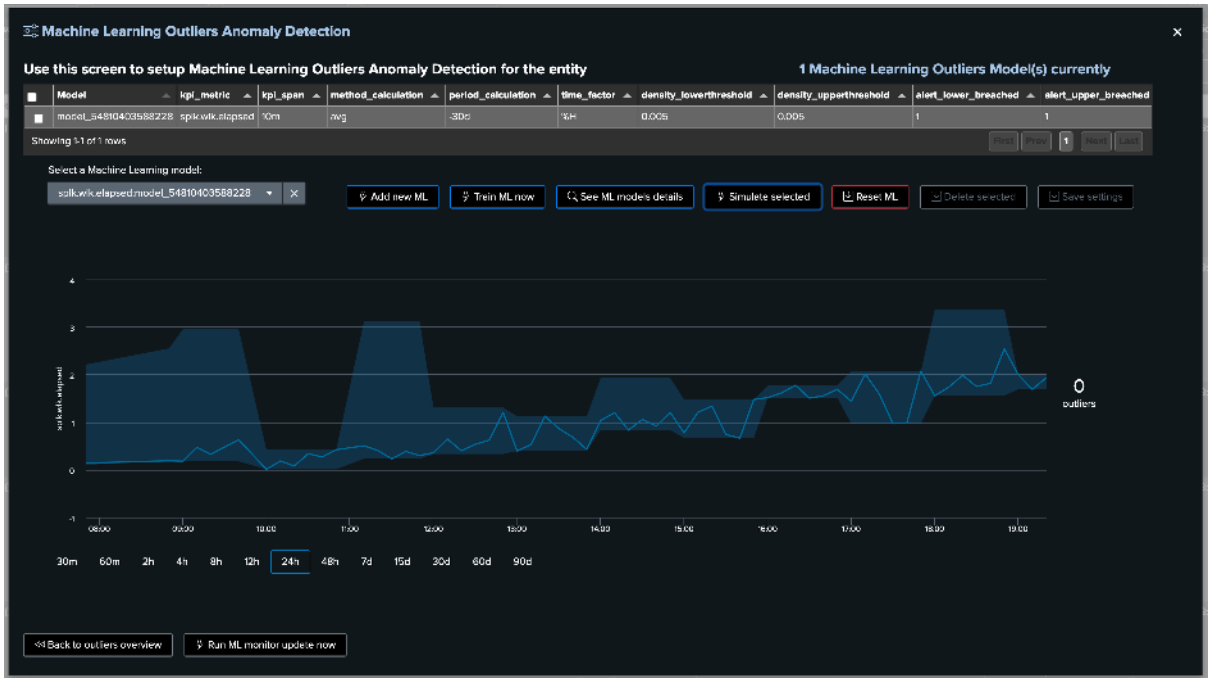


10. Machine Learning Outliers detection

Machine Learning Outliers detection can be used to detect abnormal trends for the scheduled entities, based on any of the available KPIs:



Machine Learning models created and trained by default depend on the Virtual Tenant configuration, additional ML models can be created on demand:

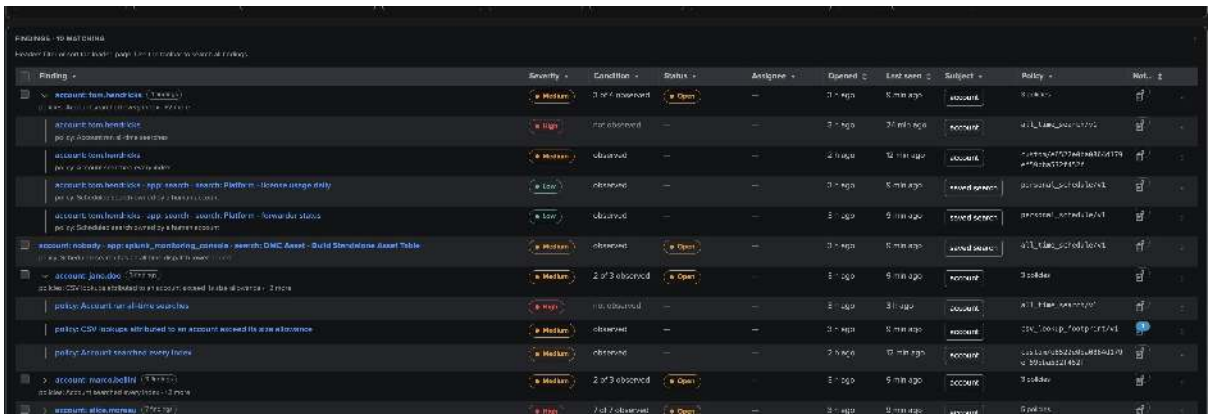


8.3.14 User Activity Monitoring

TrackMe has always watched the data. **User Activity Monitoring** watches the other half of a Splunk platform: the people and the automation that use it. It answers a question none of the entity components asks — **who is doing what on my Splunk?** Which accounts run searches, how often and at what cost; which scheduled searches are owned by a person who may leave tomorrow; which account started to run all-time searches; who consumes the search capacity of the platform; whose CSV lookups keep growing; which indexes are read, by whom, and which receive data no one reads.

A User Activity Monitoring tenant (UAM) reads what Splunk already keeps — the audit trail, the search resource

records and the knowledge objects of every search head — and turns it into **findings**: one finding per policy and subject, with its evidence, a severity, an investigation status, an assignee, notes and a history. You investigate, decide, notify the people concerned and move on.



UAM is a **dedicated tenant type**, not an eighth entity component: a tenant is either an entity tenant (DSM, DHM, MHM, FLX, FQM, WLK, VOL) or a UAM tenant, never both. There is no decision maker, no impact score and no state colour behind it: what you work with are findings. UAM shares its findings model — Finding groups, triage, assignment, notifications, findings actions, the AI Agent — with the other dedicated tenant types (Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge); those concepts are explained once in *Findings, Finding groups and findings actions*, and this page documents how they work in UAM.

Tip

This page is the task-oriented guide: set the tenant up, find your way in the workspace, triage, communicate and tune. Every setting with its default and bounds, the schedules, the indexed evidence, the REST API and the troubleshooting table are in *User Activity Monitoring — reference*.

What UAM watches

One UAM tenant monitors **one indexing layer** seen through **one to ten search head tiers**: the local search head and / or remote Splunk deployments reached through TrackMe remote accounts, on Splunk Enterprise or Splunk Cloud.

What	What UAM knows about it
Accounts	Every account seen in the users inventory or in the search activity — people, service accounts and Splunk's own system identities — with its **class** (human, service or system), its roles, the tiers it exists on and its email address when Splunk has one.
Search activity	One compact audit record per search and collection window, read from `_audit`: the account Splunk reports, the app, the search type, the outcome, the run time, and whether the search covered all time. Executions are correlated per search.
Resource usage	The CPU time and peak memory of every search, on the search heads and on the indexers, and each account's share of the environment's search capacity. CPU follows Splunk's own accounting where it exists.
Saved searches	The saved-search inventory of every tier: owner, app, schedule, dispatch time range, enablement, and every change (added, changed, removed).
Other knowledge objects	Dashboards, macros, event types, field extractions and transforms, field aliases and calculated fields, lookup definitions, automatic lookups, CSV lookup files (size, owner, modification time — never their contents) and tags, read from the Splunk REST API of each tier.
Data usage	The indexes every search names, by account, app and path (literal term, macro, event type, saved search, data model or wildcard), the knowledge objects that declare each index, and — through linked VOL and DSM tenants — what each index ingests and costs. See *Data usage*.
Findings	What the **policies** raise: sixteen built-in policies and the custom policies you write in SPL.

Note

What UAM claims, and what it does not.

- An account in UAM is the account **reported by Splunk**: it is not a verified person. Shared credentials or an integration running as a person's account look like that person.
- Collection is bounded by time only, and nothing is dropped to fit a row count: what a run cannot finish in its time continues at the next run. Coverage still depends on what the Splunk sources retain and what the collection accounts can see; gaps and partial coverage are recorded and shown in **Collection health**, never hidden.
- There is no machine learning in UAM: thresholds are explicit, and the daily CPU baseline is a percentile over the account's recent days.
- UAM never changes, dispatches or deletes anything it observes. A decision on a finding stays in TrackMe.

Before you start

A licence beyond Foundation. UAM is available with the **Enterprise** and **Unlimited** editions, the Unlimited trial and the Developer licence. It is not available on the Foundation edition or the Foundation trial. When a licence expires, the workspace stays readable while changes are refused, and scheduled collection continues until the licence enforcement ladder stops the tenants.

The Splunk sources. UAM reads `_audit` (search activity), `_introspection` (search resource records and capacity) and the indexers' search completion records (CPU accounting), through one search head tier called the **route account**. Splunk Enterprise 9.4, 10.4 and 10.6 and Splunk Cloud 10.5 are qualified; other versions run as well and only add a soft diagnostic in Collection health. Three things to know:

- The automatic detection of **service accounts** relies on the classic `key=value` format of the `_audit` lines. A deployment whose `_audit` only contains the JSON format gets no automatic service detection (human detection and your own rules still work).
- What UAM can look back on is bounded by the retention of these indexes. `_introspection` is often kept for a shorter time than `_audit`, so resource history may end earlier than audit history.

- Before Splunk 10.4, the lookup file listing gives no file size: UAM reads the sizes of the CSV lookup files through Splunk's file explorer endpoint (`admin/file-explorer`), so the collection account of each tier must be allowed to read it. By default only the `admin` and `splunk-system-role` roles can, so a least-privilege account cannot retrieve CSV lookup sizes before Splunk 10.4. When the read is refused, Collection health reports *Lookup sizes refused* and the CSV lookup footprint policy cannot judge that tier. From Splunk 10.4, the listing reports the size and no extra access is needed.

Remote tiers. Each additional search head tier is a TrackMe remote account (*Configuration → Remote accounts*, see *Remote Splunk deployments*). The wizard tests each one. Live reads you make from the workspace as yourself — **Fetch SPL**, **Fetch definition** — also need your roles to be allowed on that remote account.

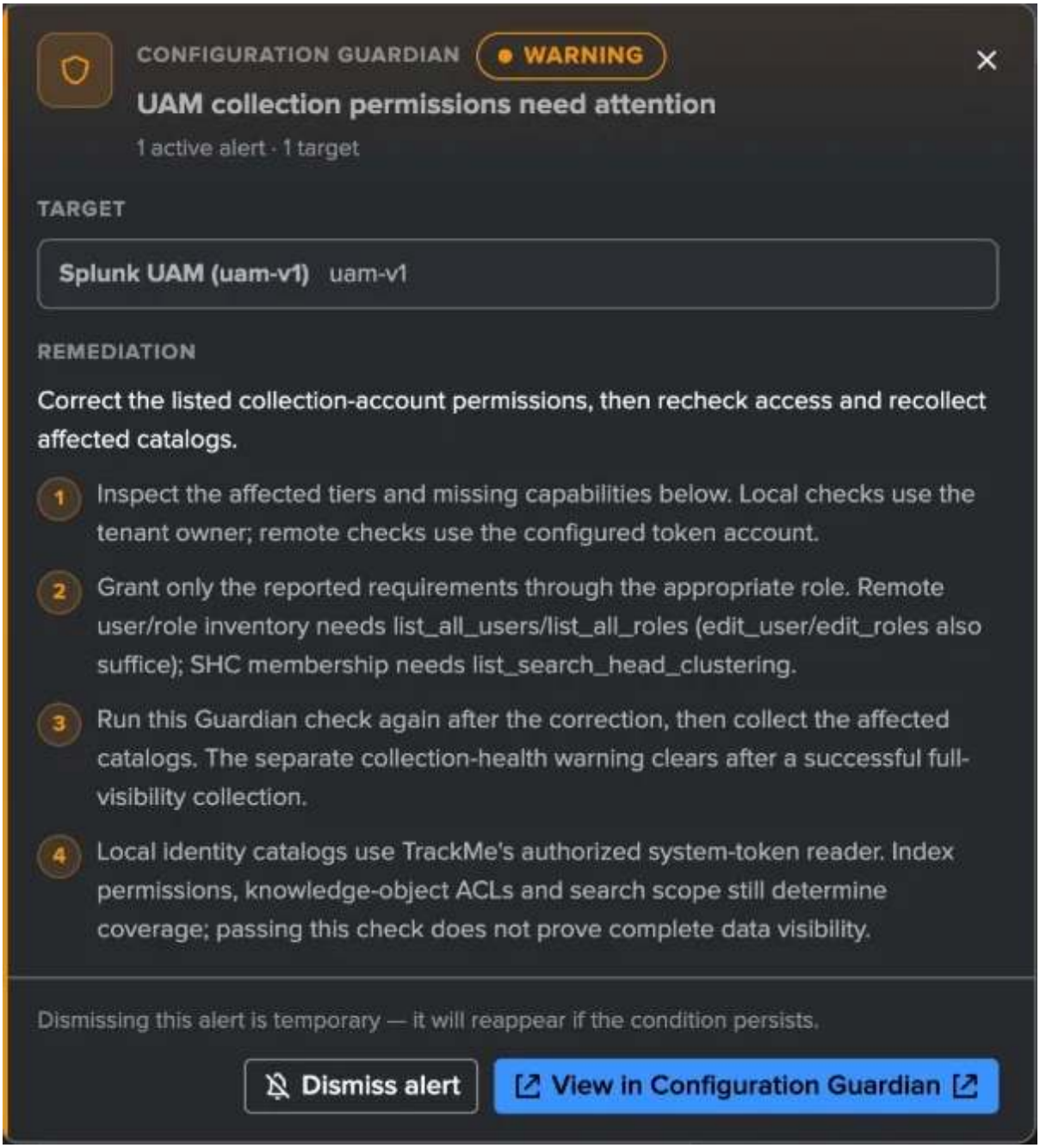
A tenant owner. Scheduled collection runs as the tenant owner. Prefer a dedicated service account (`svc-a` in the examples of this page) over `nobody`: the scheduled UAM AI Agent cannot run as `nobody` or `splunk-system-user`.

The collection accounts and their capabilities. Each search head tier is collected as one account: the **tenant owner** on the local search head, the **token account** of its remote account on a remote tier. UAM needs no `list_settings` and no `list_storage_passwords`. TrackMe checks these accounts when the tenant is created and on every health cycle. A gap raises the Configuration Guardian alert *UAM collection permissions need attention*, which names the tier and what is missing (*Configuration Guardian*).

Capability	Local tier (tenant owner)	Remote tier (token account)	What it is for
`search`	Required	Required	Run the collection searches.
`trackmeadminoperations` (or `admin_all_objects`)	Required, with one of the tenant's **admin roles**	Required	The UAM administrative boundary. On the local tier, users and roles are read through TrackMe's identity reader, which serves the tenant's administrators only. The capability is defined by the TrackMe app: on a remote deployment where TrackMe is not installed, only `admin_all_objects` satisfies it.
`list_all_users` and `list_all_roles` (`edit_user` and `edit_roles` also suffice)	Not required	Required	List every account and role of the tier. Without them, Splunk lists only the token account and its own roles, and the tier's users and roles catalogs hold one account. On the local tier, TrackMe reads them with its system token for the tenant owner.
`list_search_head_cluste`	On a search head cluster	On a search head cluster	Read the cluster members, to attribute activity to the tier.

Beyond capabilities:

- The **route account** must be allowed to search `_audit`, `_introspection` and `_internal` (the indexers' search completion records, and the searches they received, which Data usage reads).
- Before Splunk 10.4, the collection account of each tier reads `admin/file-explorer` for the lookup sizes (see above).
- Index permissions, knowledge-object permissions and the search scope of each account still decide what UAM sees. The check confirms the capabilities, not complete visibility: Collection health reports *Users or roles listed as one account sees them* (`identity_visibility_limited`) when a tier's listing still holds only the reading account.



Who can use it. UAM evidence is audit data, so the workspace is for the **TrackMe administrators of the tenant** — accounts holding the TrackMe administrator capability and one of the tenant's admin roles; Splunk administrators can do everything on every tenant. Power users of the tenant have read access through the REST API (and can send user communications and run findings actions through it); read-only users have no UAM access. The workspace's own searches (the Data usage views, the Privileged users activity, Fetch SPL) run as you, so your roles also need the tenant's TrackMe indexes, and the search text index when it is a restricted one.

Email delivery (optional). Notifications and communications are sent through a TrackMe email delivery account (*Configuration → Emails*), or the local mail transfer agent when none is selected. The account's allowed-domain rules apply.

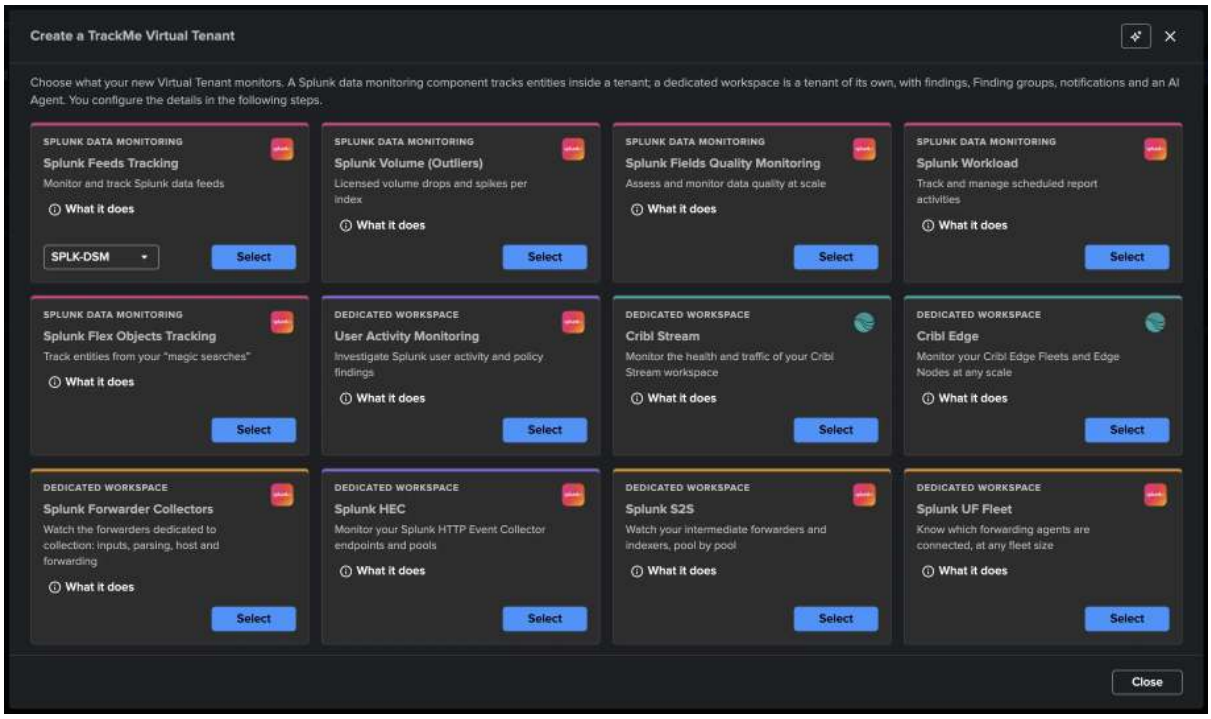
Note

UAM indexes its evidence in the tenant's summary index and its finding events in the tenant's notable index, as `trackme:uam:*` events. This evidence counts against your Splunk licence like any TrackMe summary data; it is compact by design (one record per search and window). Some indexed fields of these events take effect after

the splunkd restart that follows the app installation or upgrade; until then, reads fall back to slower searches.

Create the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **User Activity Monitoring** card (*Investigate Splunk user activity and policy findings*). The wizard has four steps.



Step 1 — Basics

The **tenant identifier** (lowercase letters, digits and hyphens, up to 20 characters; it cannot change later), an optional **alias** shown on the tenant card and an optional description.

Step 2 — Configure UAM

Search head tiers. *Local search head* is selected by default; each TrackMe remote account can be added as another tier, up to ten. Select every search head tier that shares the indexing layer you want to monitor — for example an ad hoc search head cluster and an Enterprise Security cluster on the same indexers. Remote accounts are tested as soon as you select them, and creation waits until every selected one is reachable (**Retry connection test** if one fails).

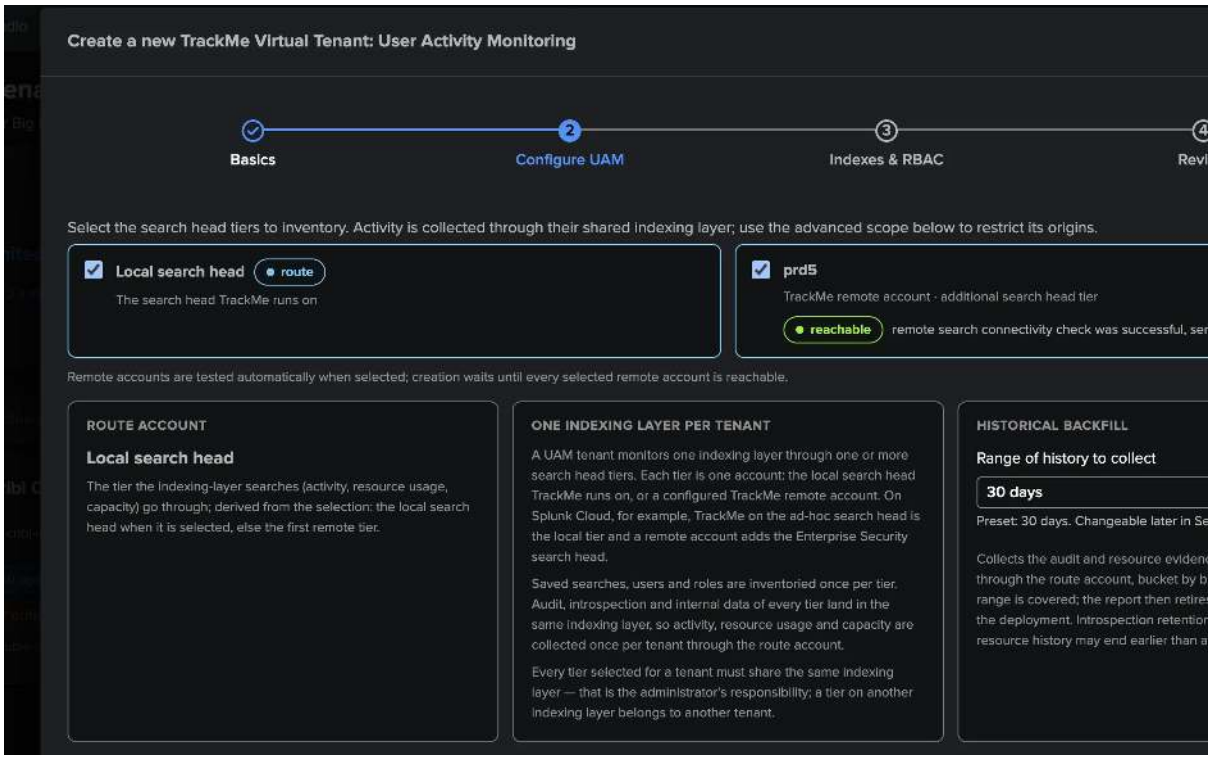
Route account. Derived from your selection: the local search head when it is selected, otherwise the first remote tier. The indexing-layer searches — audit activity, resource usage, capacity, logins and automation — go through it, once for the whole layer, while the knowledge objects are read on every tier.

One indexing layer per tenant. All the tiers of a tenant must search the same indexers. Monitoring two independent deployments takes two tenants.

Historical backfill. How far back UAM collects the audit and resource evidence that existed before the tenant: **No backfill** (the default), 7, 14, 30, 60 or 90 days. The backfill runs hourly in 5-minute buckets, newest history first, and retires itself once the range is covered or the retention of the sources is reached. It makes the account classification and the activity history useful from the first day, at the cost of reading that history from the indexers: on a busy deployment, start without it, or with a short range, and extend it later in *Settings → Collection limits*.

Activity scope (advanced). Optional hostname patterns (`*` and `?`, up to 20 include and 20 exclude patterns) that restrict which search heads' activity is collected, for example to leave out a test search head that shares the indexers. Exclusions win. The scope is part of the base searches, so the indexers only return the events of the search heads

in scope: on a large indexing layer shared by several search head clusters, it is also the first way to make collection lighter.



Step 3 — Indexes & RBAC

The TrackMe indexes of the tenant (summary, audit, metric and notable), the **tenant owner** and the admin, power and user roles. The summary index receives the evidence and the notable index the finding events; each family can be pointed at another index later in *Settings → Indexes*.

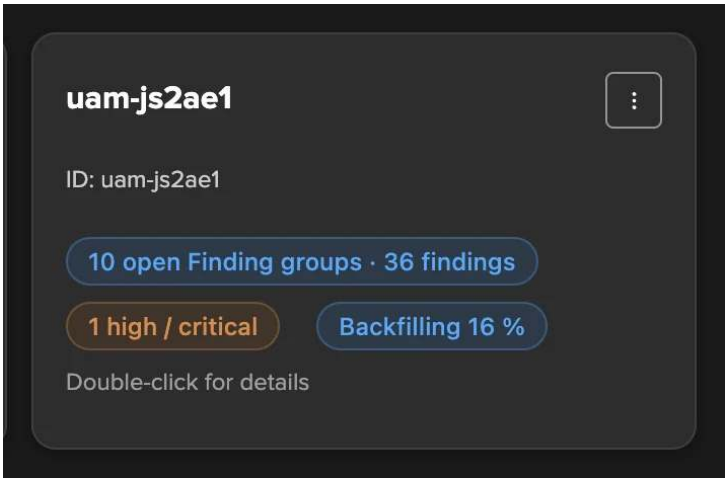
Step 4 — Review

Check the summary — the tiers with the route, the owner, the activity scope, the backfill — and create. Everything is provisioned and scheduled at once: the inventory, audit activity and resource usage reports start within minutes, the account signals and the policies follow, and each tier's first full saved-search listing runs within the hour. If the creation result is incomplete, the wizard says so and the workspace offers **Repair UAM workspace**.

The step-by-step creation is also described in *Creating a splk-uam tenant*.

After creation

The tenant appears on the *Virtual Tenants* page with its open Finding groups and findings, the high or critical ones, and its collection status — here the historical backfill in progress. Selecting it opens the workspace; double-clicking it opens a read-only overview.



Your first hour with a new tenant:

1. Open **Collection health**: the **Status** view says whether anything needs attention, and **Trackers** shows the first inventory, audit activity, resource usage, account signals and policies runs of every tier.
2. **Classify the monitoring accounts** as system — the tenant owner, your administrator accounts — so that they never appear in findings (see *Accounts and classification*).
3. Review the **Policies** tab: the enabled policies, their severities and exclusions.
4. Come back to **Findings** after the next policies run (every 15 minutes).

The workspace

The workspace opens on the **Findings** tab. At the top, the **status strip** tells you that the tenant is working: the search head tiers (the route account marked · *route*), the age of the last **Inventory**, **Audit activity**, **Resource usage**, **Account signals**, **Data usage** and **Policies** runs, the activity checkpoint lag and coverage, the backfill progress, a **Configuration changed since last run** warning after a settings change, **Refresh**, and **Collect now** — which runs every tier's inventory, then the audit activity, the data usage, the resource usage and the account signals, then the policies, with the current configuration, and every tier's full saved-search listing alongside.



Eleven tabs follow: **Findings**, **Users**, **Privileged users**, **Data usage**, **Activity**, **Scheduled searches**, **Policies**, **Audit changes**, **Collection health**, **AI Agent** and **Settings**.

A few conventions apply everywhere, shared with the other findings workspaces (*The shared workspace conventions*):

- Column headers filter and sort the loaded page; the toolbar filters search everything.
- Every panel has a ⍰ menu with *Open in Search* (the search behind the figure, ready to edit) and *Refresh*.
- Three personal display preferences per tenant live in the Tenant Home settings panel, under *Personal preferences for this tenant*: **Collection monitors** (the status strip), **Single stats** and **Charts**. They change your view only.
- **Exclude TrackMe activity** (Users, Activity and the account view) leaves out the searches run in the `trackme` app — TrackMe's own scheduled trackers (recognised by their search id) and your own use of TrackMe. An ad-hoc search that reports another app or none, such as the tracker searches TrackMe dispatches over REST, cannot be attributed and stays in. Findings are not affected. It is remembered per viewer.
- Notification emails link straight to a finding or a Finding group, opened in its investigation.

Findings

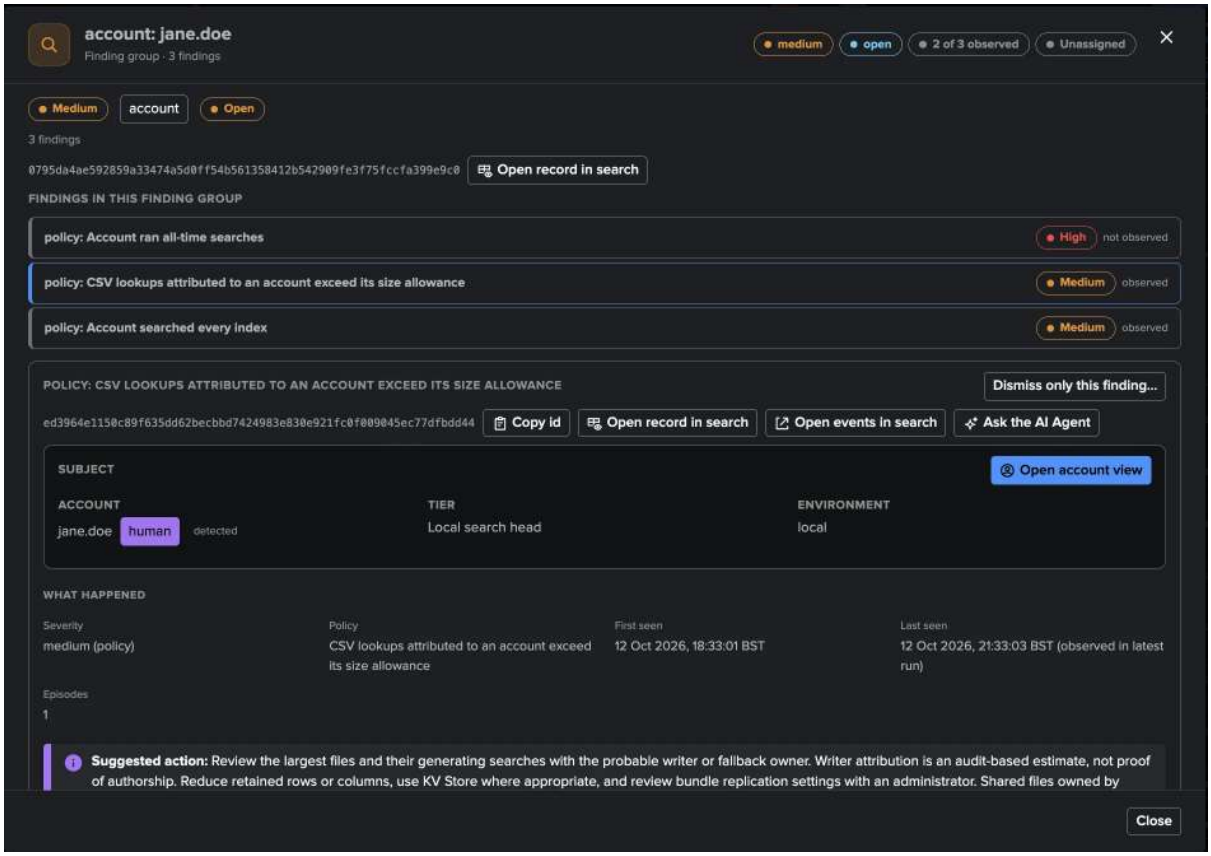
Your investigation backlog. Tiles count **Open groups**, **High / critical groups**, **Groups assigned to me**, **Resolved groups** and **Dismissed groups** — Finding groups — plus **New findings in range** (findings first seen in the time range), and drive the filters; charts show the finding transitions over the time range (24 hours by default) and the open

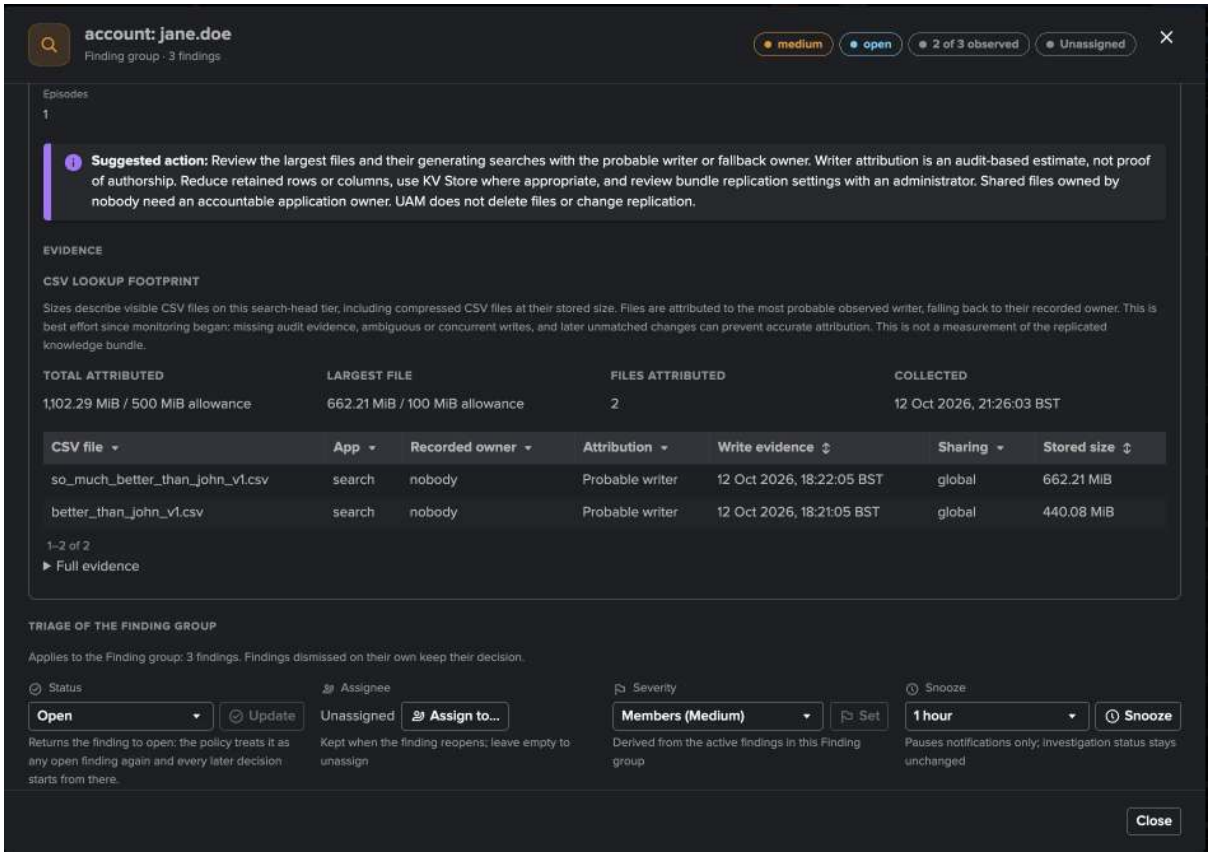
findings by severity and by policy. Filters cover the status, severity, policy, subject (accounts, saved searches or roles), first seen, assignment (including assignment groups) and free text (owner, app, search name).

The table lists **one row per Finding group**: an account and the saved searches it owns. A group of several findings has an expander showing each member with its own policy, severity and whether the latest policies run still **observed** it. Selecting rows offers **Merge into one Finding group**, **Add to the Finding group** (a manual group), **Run action**, **Assign to…** and **Bulk edit**; **AI Agent review** asks the AI Agent to review the open and investigating findings.



Selecting a row opens the **Investigate** view: the subject, the findings of the Finding group, *What happened* in plain words, the suggested remediation, the evidence of the selected finding, **Triage** of the Finding group (status, dismissal, assignee, notification snooze, severity, investigation note), the notes and both histories — the finding's and the group's. From there, open the account view or the saved search view, or the record and its events in Search.

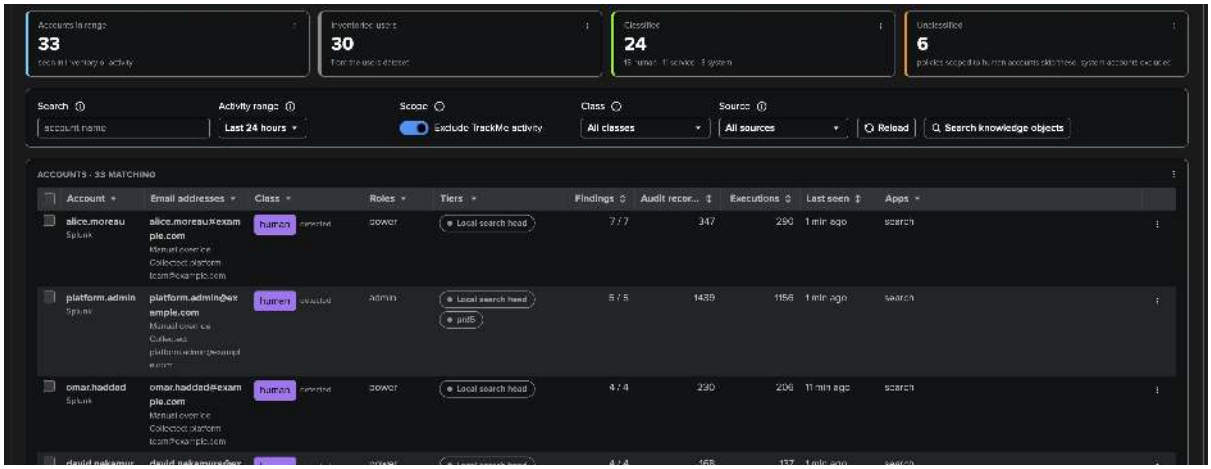




Each finding explains its own evidence. A CSV lookup footprint finding, for instance, lists the files attributed to the account, their size and how each was attributed: to its **probable writer** — the account whose searches wrote the file — or, when no writer could be identified, to its **recorded owner**.

Users and the account view

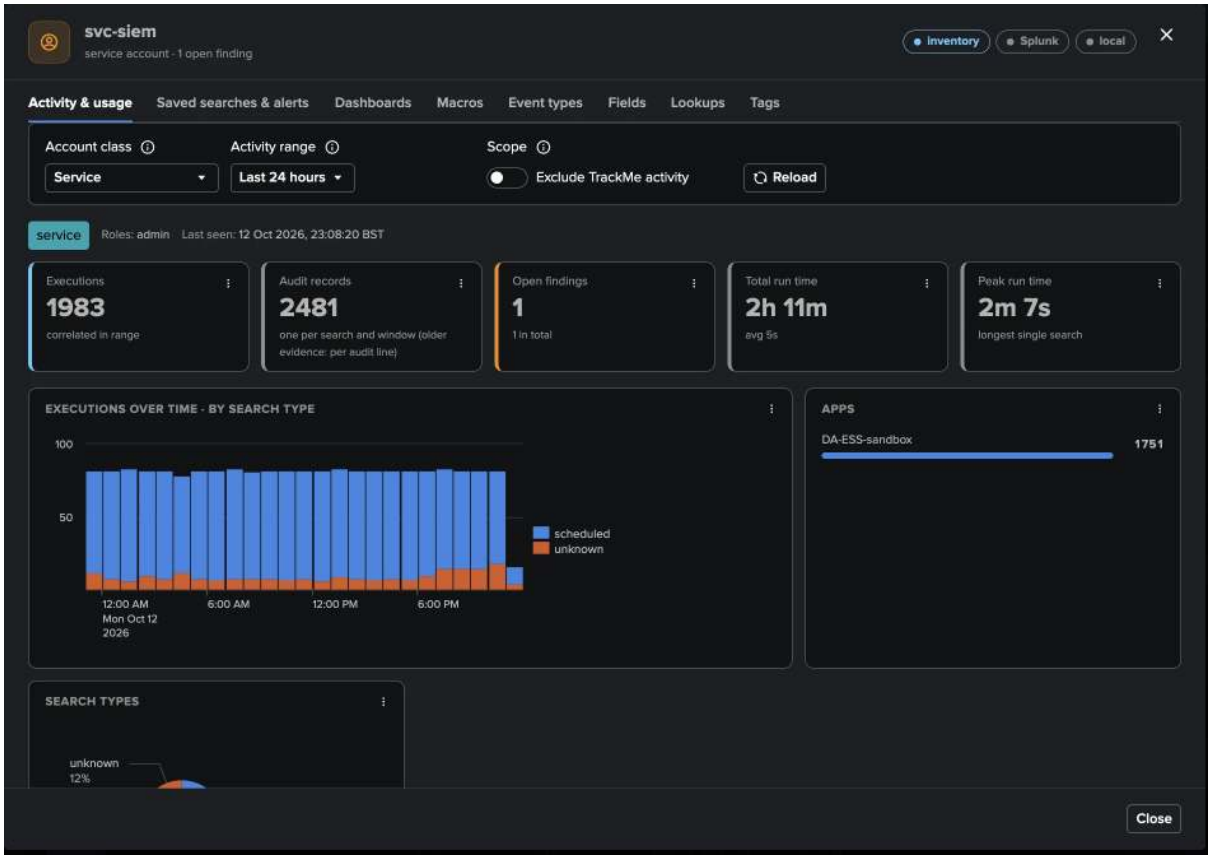
Every account across the tiers: tiles for the accounts in range, inventoried, classified and unclassified; filters by class, source and activity range; columns for email addresses, class (with a *detected* marker when a detection set it), roles, tiers, findings, audit records, executions, last seen and apps. The row menu sets the class or overrides the email address used for user notifications (**Override email address…**); a selection can be classified at once, or decided on as a whole (resolve, dismiss or mark investigating its open findings).

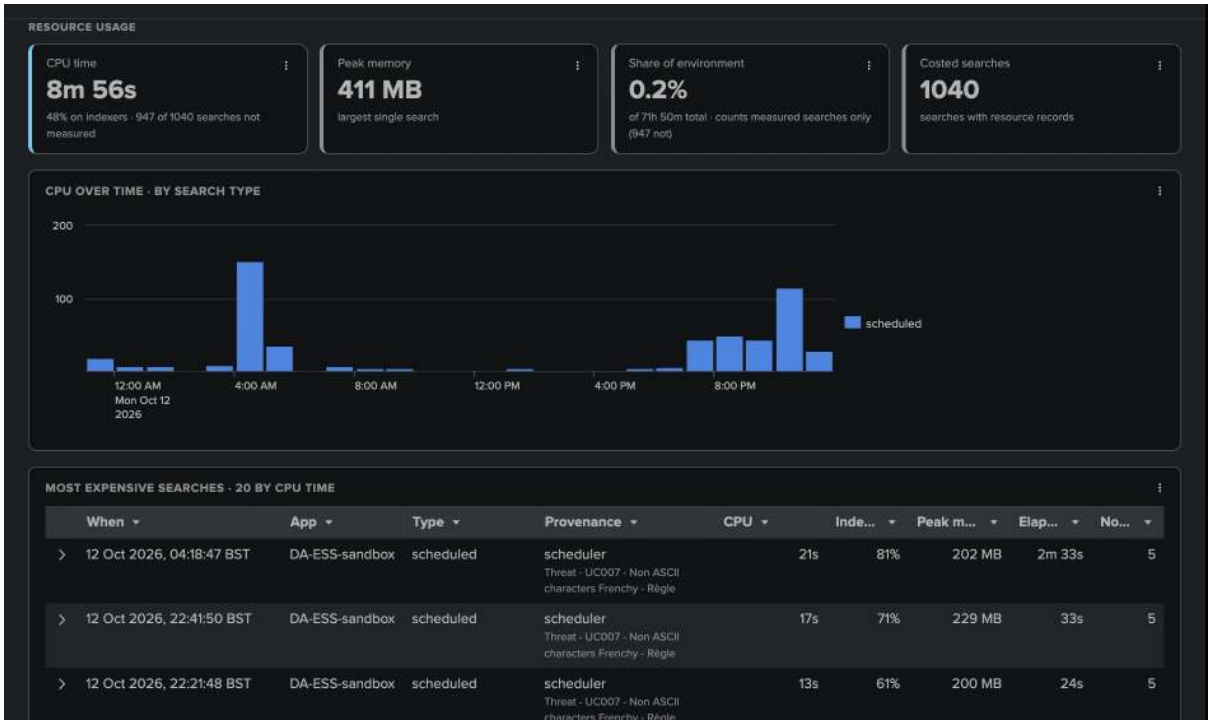


Selecting an account opens the **account view**:

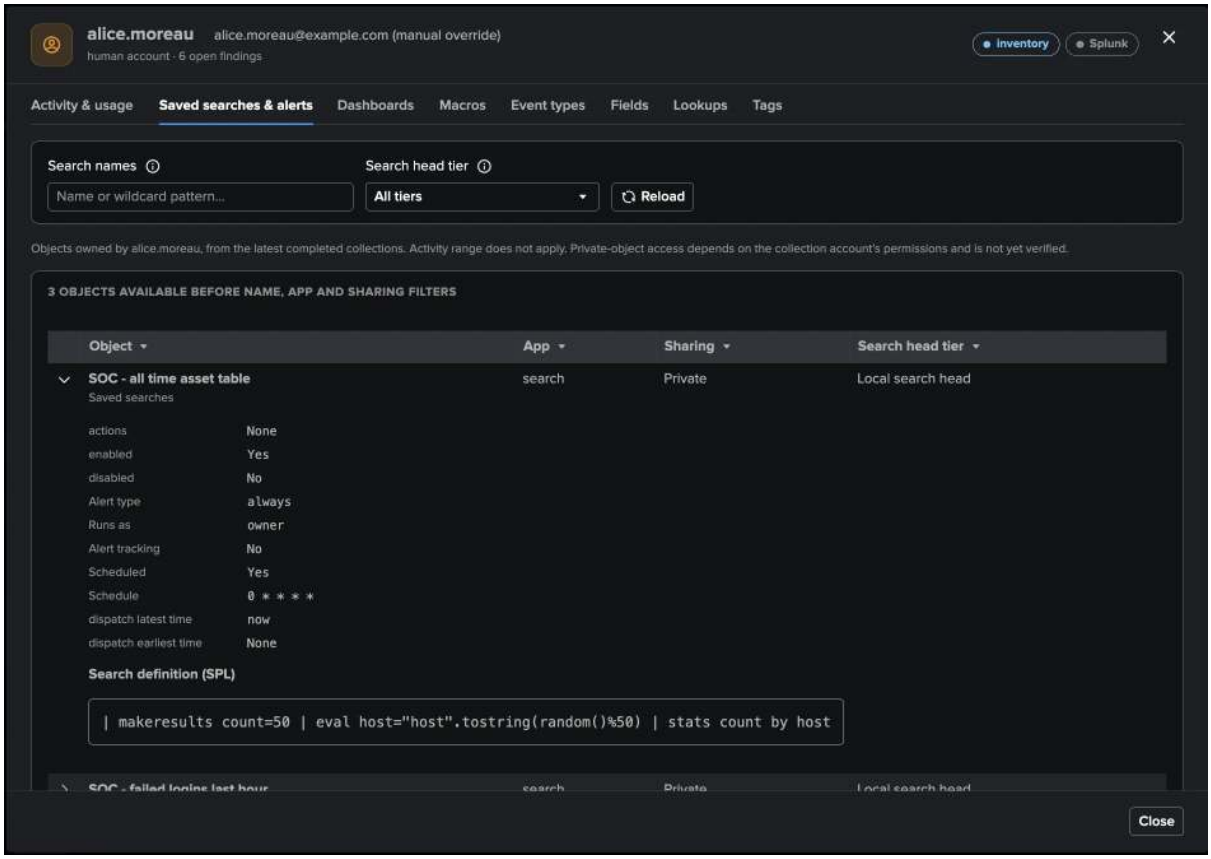
- **Activity & usage** — executions and run time over time by search type, CPU over time, the daily CPU baseline, the resource usage (CPU time, peak memory, share of the environment), the most expensive and the most recent searches with **Fetch SPL**, the account's findings and the scheduled searches it owns.

- **Data accesses** — which indexes the account read, over time: a chart of its named reads per index and one lane per index with a marker for each observation (see *Data usage*).
- The account's **catalogs**, one tab per family: *Saved searches & alerts*, *Dashboards*, *Macros*, *Event types*, *Data models*, *Indexes*, *Fields*, *Lookups* and *Tags* — what this account owns on each tier.



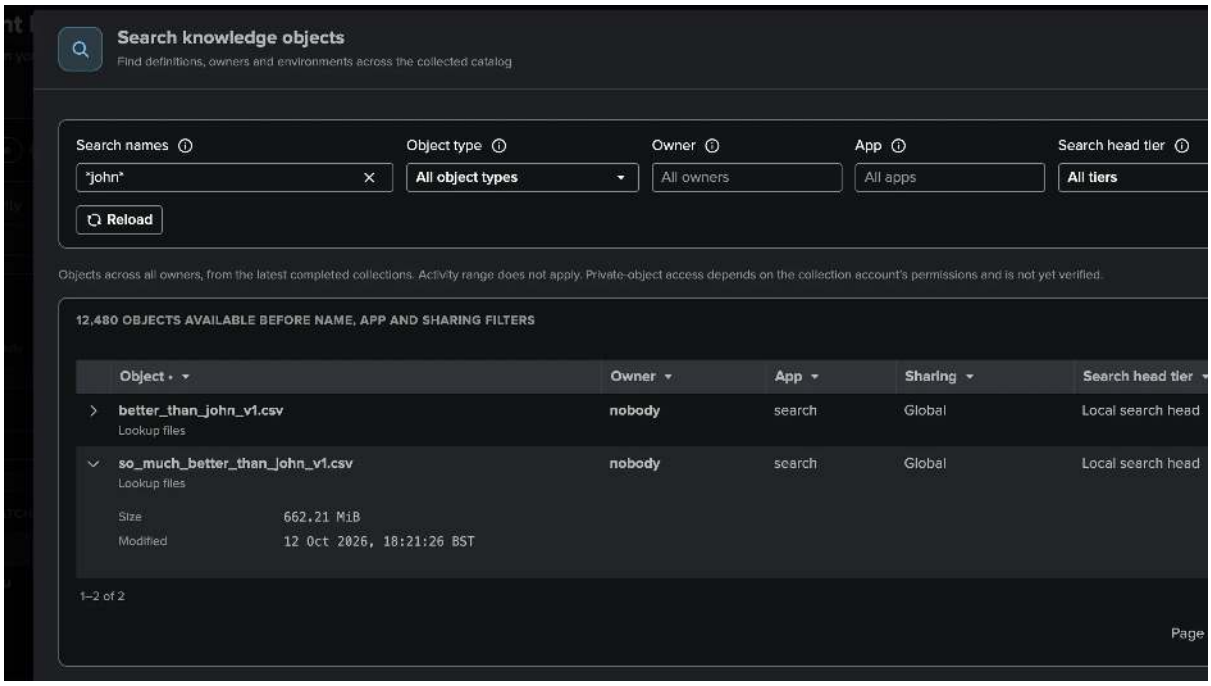


The resource usage tells what an account costs: here a service account whose scheduled detection searches use 0.2% of the environment's search CPU over the day, with the most expensive runs listed first. Searches the resource records could not measure are counted and shown as such, never estimated.



Fetch SPL shows the text of a search. By default UAM does not store search text: it reads it live from `_audit`, as you, and stores nothing. If you switch on *Store SPL search* (*Settings → Collection*, off by default), the text is indexed in its own record in the search-text index — point that index at a restricted index first, since search text can contain literals and secrets. When a text cannot be shown, the reason is given.

Search knowledge objects (on the Users tab) searches the collected catalogs across every owner: a name or a wildcard pattern, filtered by object type, tier, app, owner and sharing. Use it to answer "who owns the macro `app_a_base`?" or "which dashboards use this name on which tier?".



Note

The catalogs reflect what the collection account can see on each tier: the coverage of private objects depends on its permissions. They are collected on a cadence (daily in the off-peak window by default for most families, every 15 minutes for CSV lookup files, hourly for users and roles), not as atomic snapshots. A listing too long for one run continues at the next one, and the previous catalog stays in use until the new one is complete.

Privileged users

Which accounts hold high privileges on the monitored search head tiers, and why. An account is **privileged** when one of its roles — or a role that role imports, however indirectly — is a **privileged role** or grants a **privileged capability**:

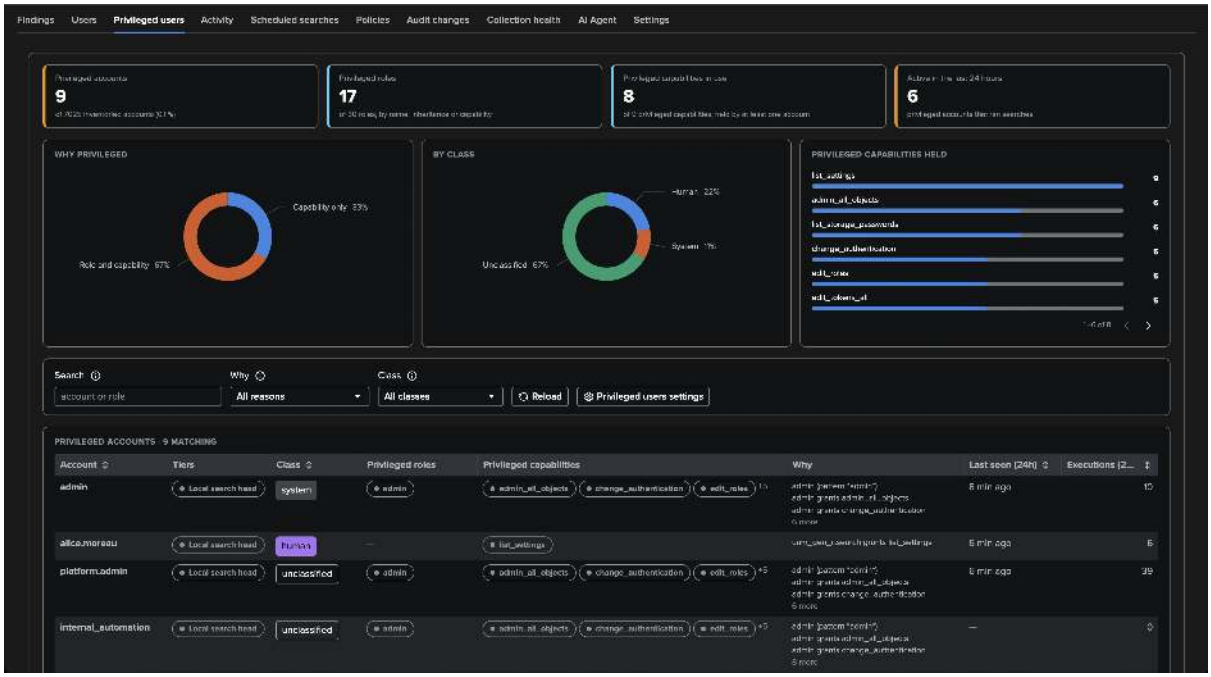
- **Privileged roles**: every role whose name matches a pattern (`*admin*` by default, case-insensitive, `*` and `?`), plus the roles you add by name, minus the roles you exclude. The patterns are live: a role created later is privileged as soon as its name matches. `trackme_admin` is excluded by default: TrackMe's own administrator role holds TrackMe capabilities, not Splunk-level ones.
- **Privileged capabilities**: `admin_all_objects`, `change_authentication`, `delete_by_keyword`, `edit_roles`, `edit_tokens_all`, `edit_user`, `list_settings`, `list_storage_passwords` and `rest_apps_management` by default, granted directly or through an imported role.
- An exclusion only removes a role's **name** rule: an excluded role that grants a privileged capability still makes its users privileged, through the capability.

The tab shows the privileged accounts and their share of all accounts, the privileged roles, the privileged capabilities in use and the privileged accounts active in the last 24 hours; charts of why accounts are privileged (role, capability or both), of their classes and of the accounts holding each privileged capability; the **privileged accounts** with their reasons in words (`team_lead imports admin (pattern *admin*)`, `secops grants list_storage_passwords`), last seen and executions; and the **roles** of every tier, the privileged ones by default. An account present on several tiers is one account, each reason naming the tier that grants it.

Everything is read from the users and roles catalogs UAM already collects: the view follows their cadence (hourly by default) and makes no call to Splunk. The rules are set in *Settings → Privileged users*: the patterns, and two dual lists — the roles and the capabilities the tiers hold — where moving a role a pattern matches out of the privileged list excludes it, and moving any other role in adds it by name.

Splunk grants roles and capabilities through direct membership and through inheritance, and a role inherits from roles that inherit further: every reason keeps its chain — *team_lead imports admin*, *team_lead inherits admin through ops*, *leaf inherits list_settings from base through mid*. The full rules are in *Privileged users*.

Four **privileged** policies build on this assessment, all off by default: an account that became privileged, a role that gained a privileged capability, a dormant privileged account, and privileged accounts that differ from a reference you maintain. See *Privileged policies*.



Data usage

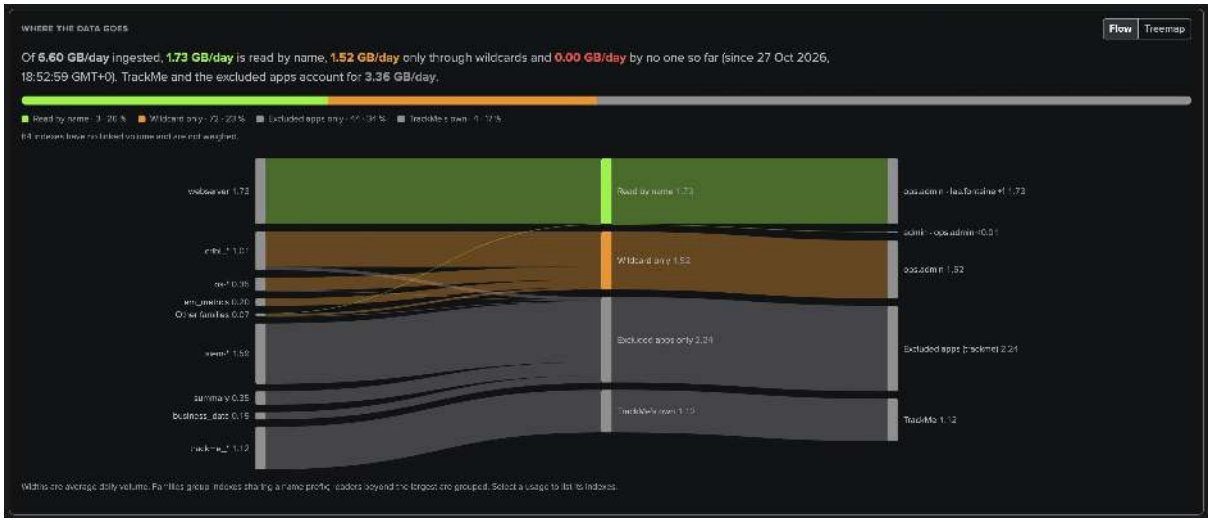
Which indexes are read, by whom, and which receive data no one reads? UAM already reads every search from the audit trail. The **Data usage** tab turns those searches into a map of your data: for every index, the searches that named it, the accounts and apps behind them, the path they took (a literal `index=` term, a macro, an event type, a saved search, a data model or a wildcard), the knowledge objects that declare it, and — when the tenant is linked to a VOL or DSM tenant — what the index ingests per day and what it costs.

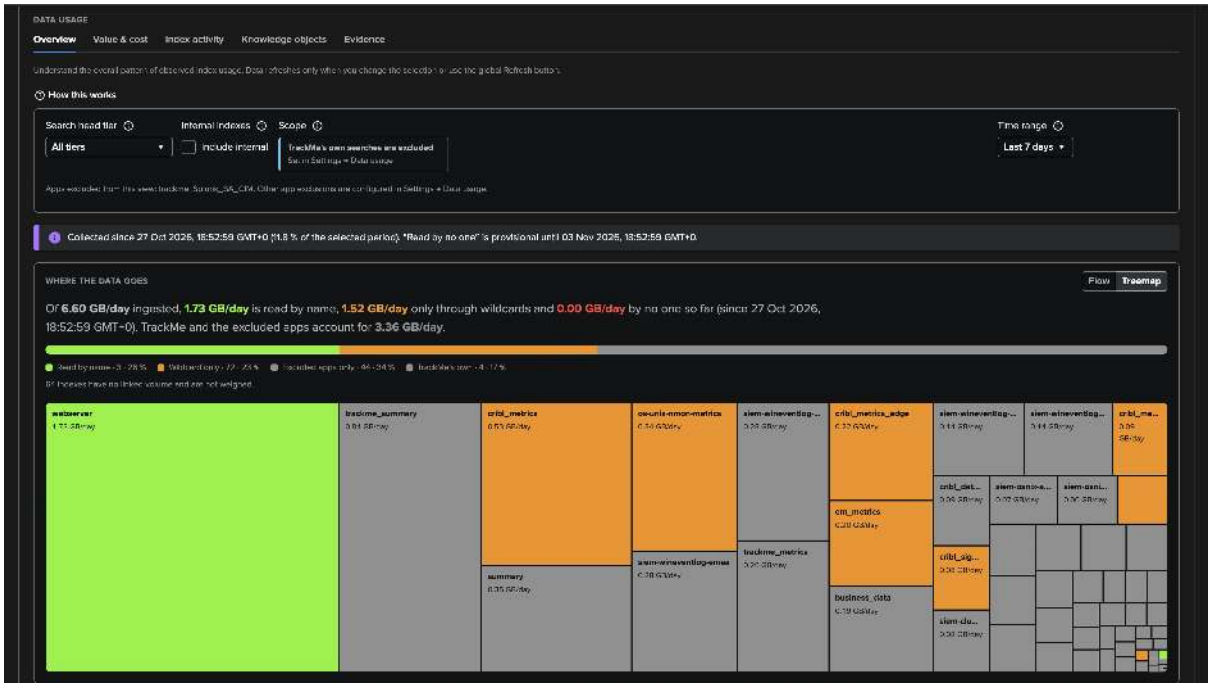
It rests on one rule: **an absence of reads is a claim, made only when the evidence supports it.** An index is never called unread from a few hours of collection, an incomplete read or a wildcard that may reach it. Every view says what it covers, in one **coverage card** under its controls: when the collection started and what that means for the view ("Collected since 27 Oct 2026 (11.8 % of the selected period). "Read by no one" is provisional until 3 Nov 2026."), a fault with its cause, or *Coverage complete*.

Five views share the same controls: the **search head tier**, **internal indexes** (left out by default), the **scope** (TrackMe's own searches and the excluded apps, set in *Settings → Data usage*) and the **time range** (the last 7 days by default).

Every tile, chart and table of the Overview, Value & cost, Knowledge objects and Evidence views has a three-dots menu with **Open in Search**. It opens the search over the recorded facts behind that view, with a note on what only the map adds when you read the view: the catalog expansion of wildcards, the role scopes and the VOL volumes. How each kind of claim is recorded is described in *Data usage reference: how each claim is processed*.

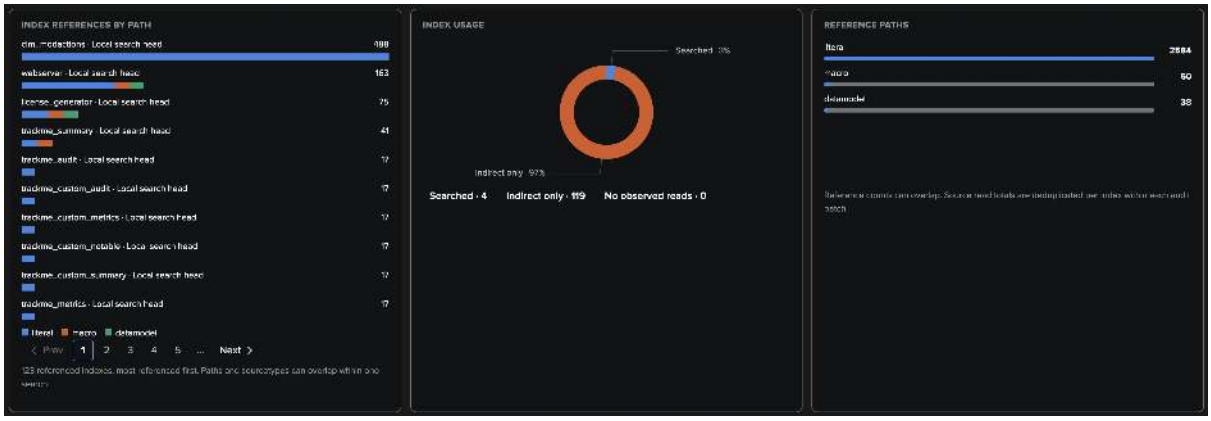
Overview — where the data goes. One sentence answers the question: *Of 6.60 GB/day ingested, 1.73 GB/day is read by name, 1.52 GB/day only through wildcards and 0.00 GB/day by no one so far.* A bar splits the volume by usage, and the hero shows it either as a **flow** — from data families (indexes sharing a name prefix) to their usage, then to the accounts that read them — or as a **treemap** of ingestion, one tile per index sized by its daily volume and coloured by its usage.



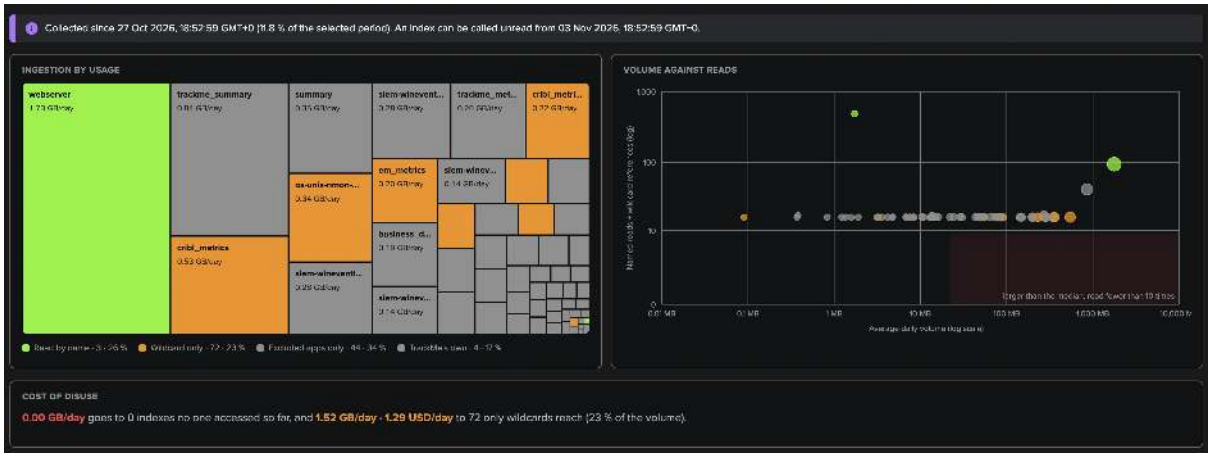


The usage classes are decided across the whole indexing layer, never per tier: **Read by name** (a search named the index), **Wildcard only** (only wildcards such as `index=cribl_*` reach it: they do not prove which index supplied the events), **No reader**, **Excluded apps only** (only the apps you exclude read it, for example `trackme`) and **TrackMe's own** (the indexes TrackMe writes to, never a review candidate). An index without a linked volume is counted but not weighed.

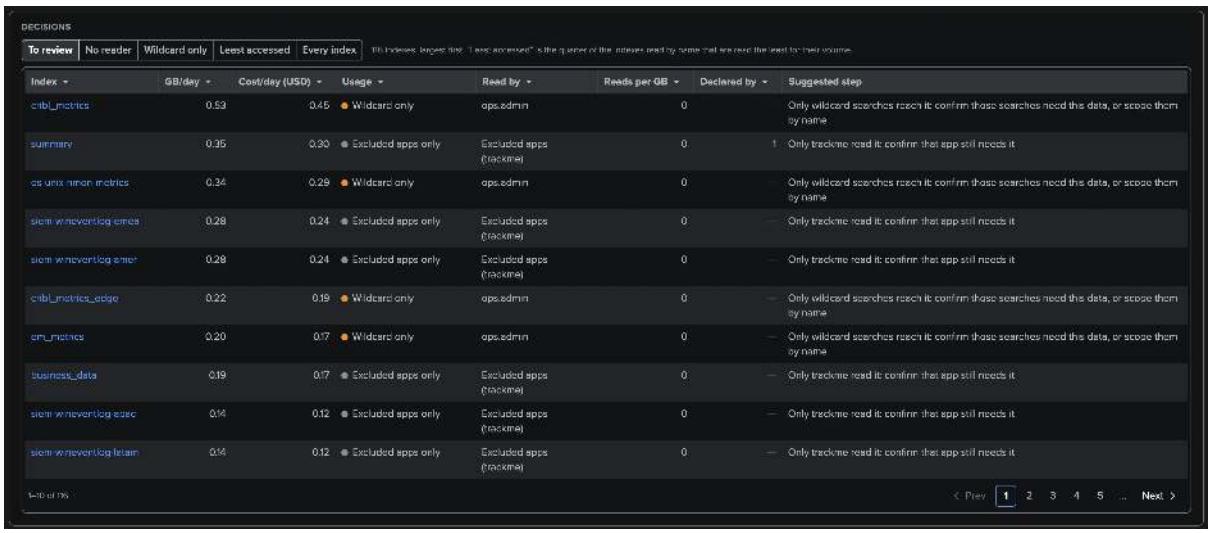
Under the hero, three panels list the **index references by path**, the **index usage** verdicts and the **reference paths** (literal, macro, event type, saved search, data model).



Value & cost — weigh what an index costs against how it is read. The treemap of ingestion by usage sits beside **Volume against reads** (log scales, one dot per index: the shaded corner holds the indexes larger than the median and read fewer than ten times), and **Cost of disuse** states what the indexes no one accessed, and those only wildcards reach, cost per day.

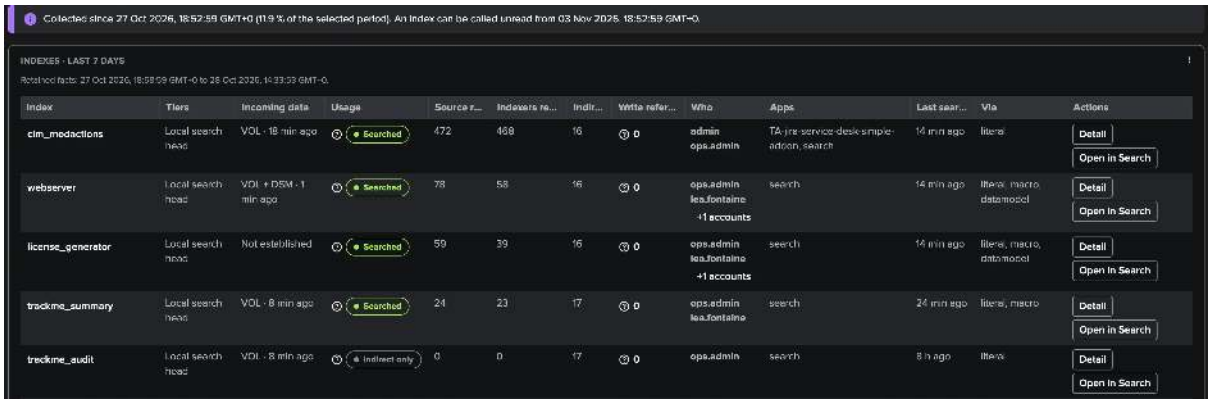


The **Decisions** list turns it into work: one row per index, largest first, with its daily volume and cost, its usage, who read it, its reads per GB, the knowledge objects that declare it and a **suggested step** ("Only wildcard searches reach it: confirm those searches need this data, or scope them by name"). Five views filter it: **To review** (no reader, wildcard only, excluded apps only and the least accessed), **No reader**, **Wildcard only**, **Least accessed** (the quarter of the indexes read by name that are read the least for their volume; the ranking needs at least four such indexes with a daily volume) and **Every index**. An empty list says why — for example, while the collection is younger than the period, nothing is called unread. The list pages 10, 25, 50 or 100 rows.

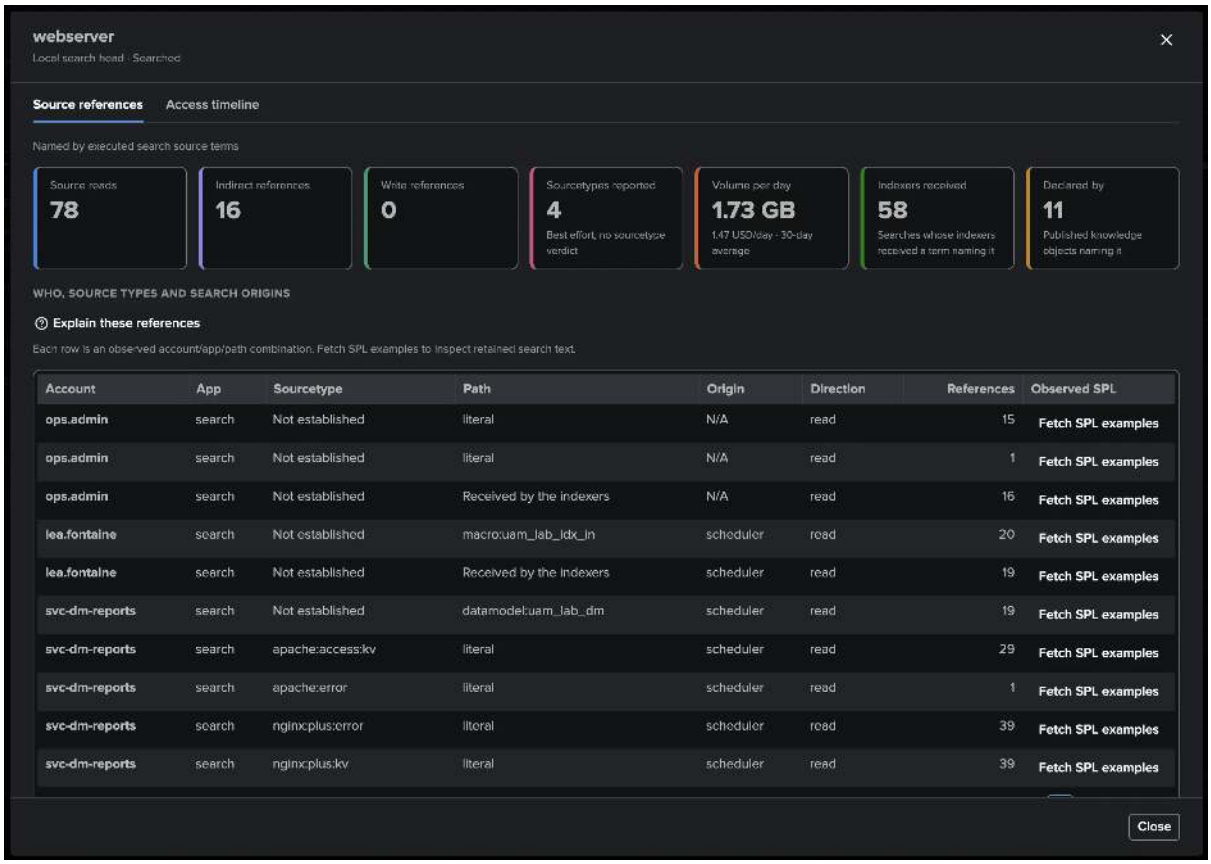


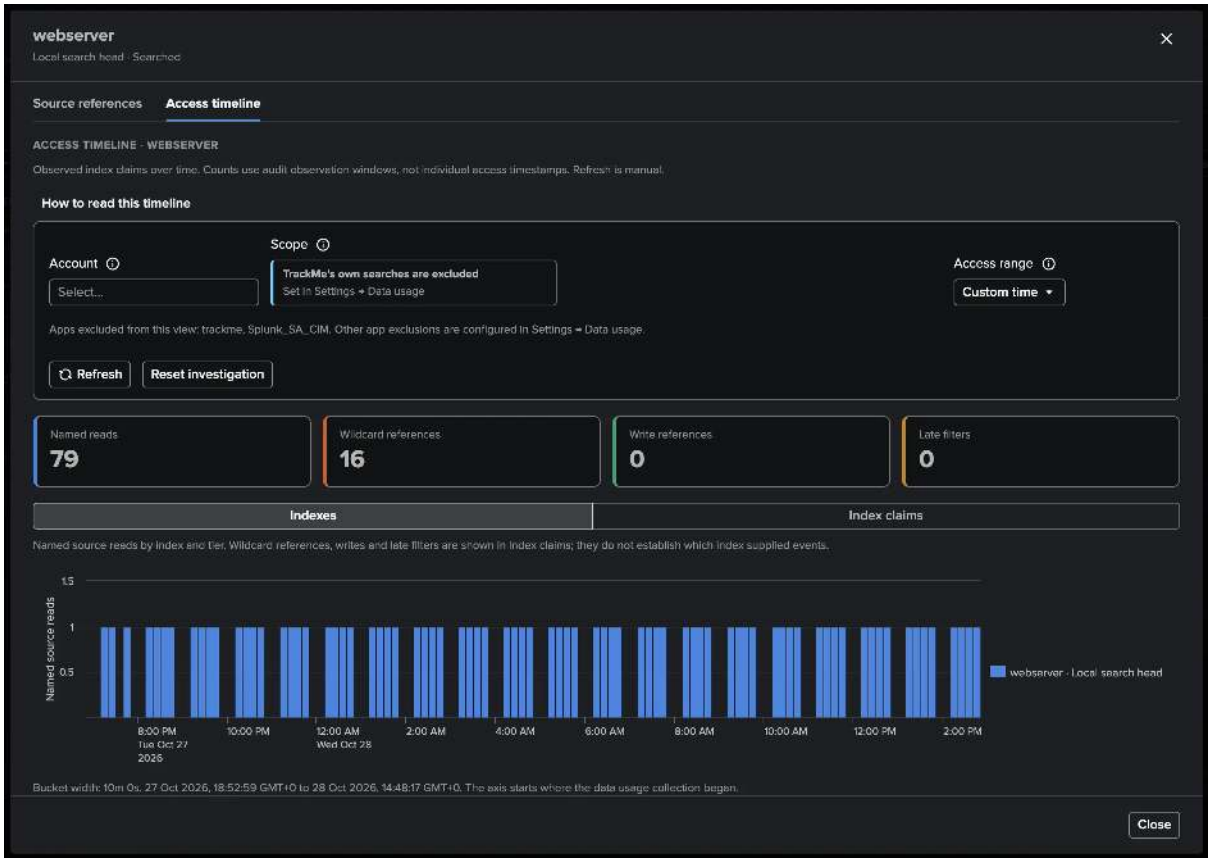
Index activity — investigate an index. One row per index and tier: the incoming data (from the linked VOL or

DSM tenant), the usage verdict, the source reads, the reads the **indexers received** (on distributed deployments, the indexers' own record of the searches they ran, shown beside the search head's figures, never added to them), the wildcard references, the write references (`collect` destinations, and the target index of a saved search's summary indexing or Log event alert action), who read it, from which apps, when, and through which paths. Filters cover the usage, the path (its **wildcard** choice selects the wildcard references), the account class, the index name, the account and the app.



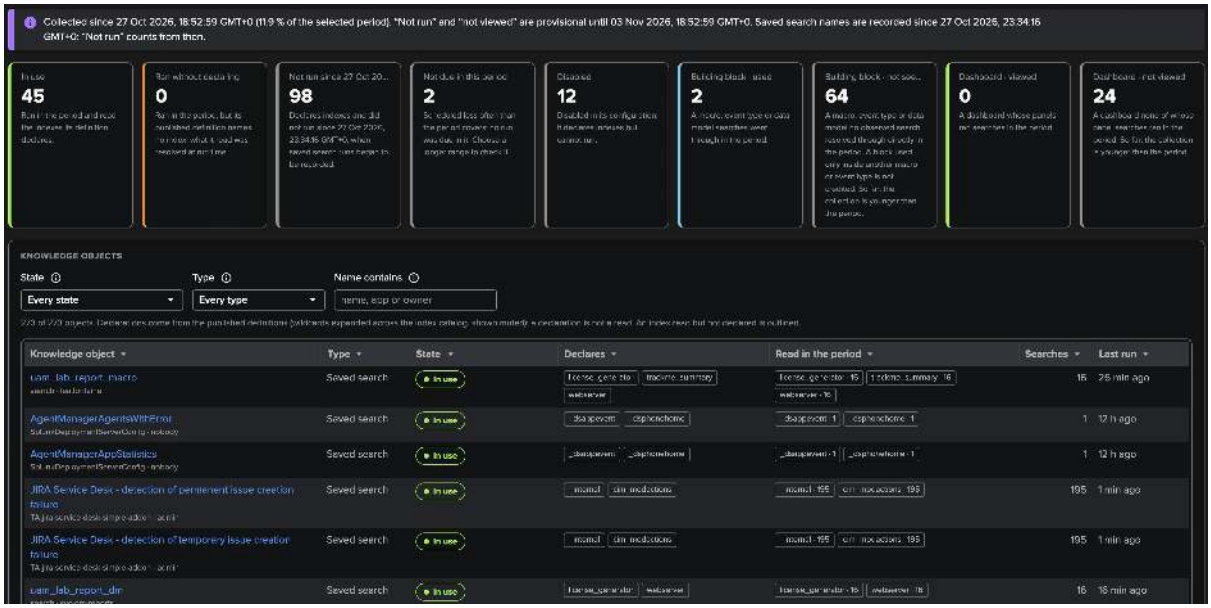
Detail opens the index. **Source references** gives the tiles — source reads, indirect and write references, the source-types reported (best effort: a sourcetype never gets a verdict of its own), the volume per day and its cost, the reads the indexers received and the knowledge objects that declare it — and one row per account, app, sourcetype and path, with **Fetch SPL examples** to read the retained searches. **Access timeline** charts the named reads of the index over the period, clock-aligned, from the start of the collection.



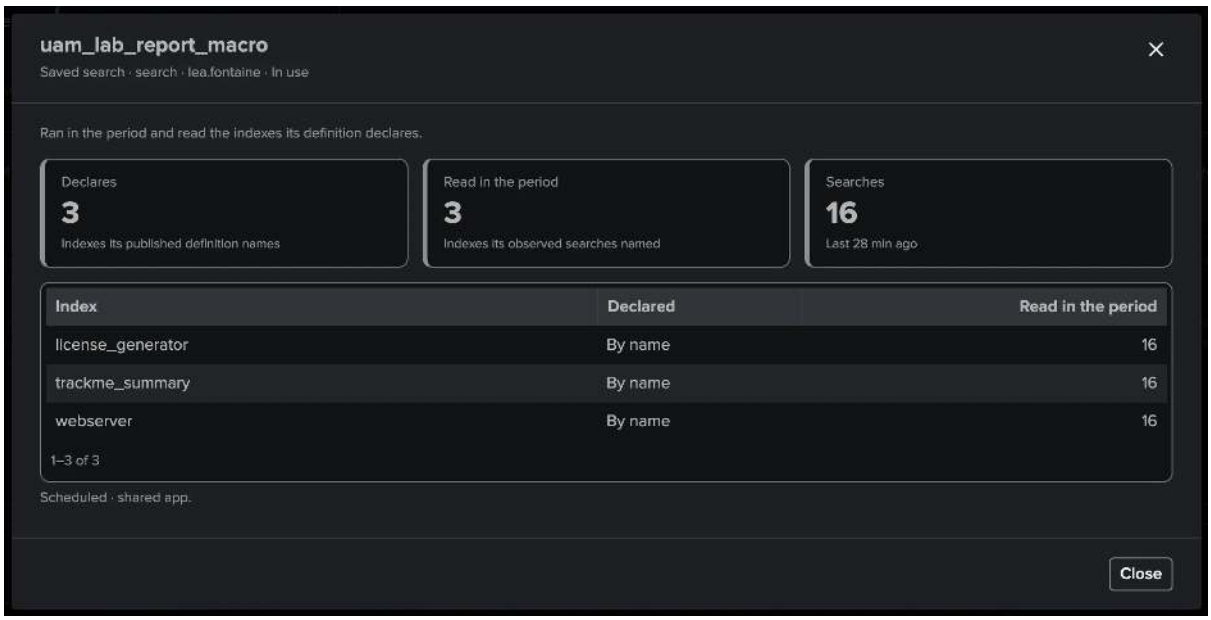


Knowledge objects — declared against observed. Each published saved search, macro, event type, dashboard and data model, with the indexes its definition **declares** (wildcards expanded across the index catalog, shown muted) against the indexes its searches **read** in the period (an index read but not declared is outlined). Tiles count the states and filter the list:

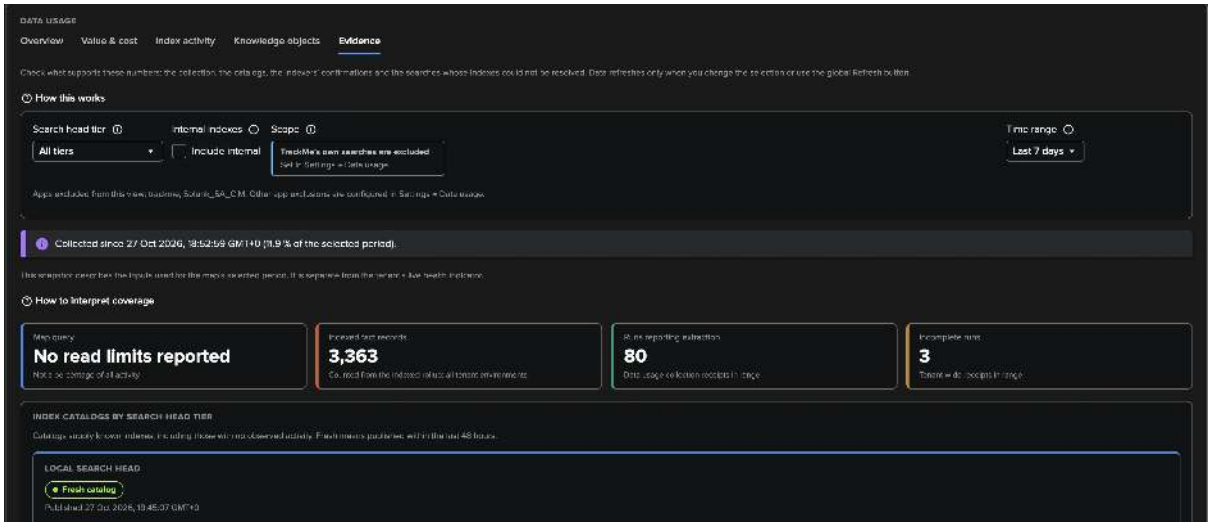
- **In use** — ran in the period and read the indexes its definition declares.
- **Ran without declaring** — ran, but its definition names no index: what it read was resolved at run time.
- **Not run** — declares indexes and did not run in the period, dated from when saved search runs began to be recorded.
- **Not due in this period** — a scheduled search whose schedule leaves a longer gap than the period covers: it was not due, so it is not called *Not run*.
- **Disabled** — disabled in its configuration.
- **Building block · used / not seen** — a macro, event type or data model that searches went through, or that no observed search resolved directly.
- **Dashboard · viewed / not viewed** — a dashboard whose panels ran searches in the period, or not.



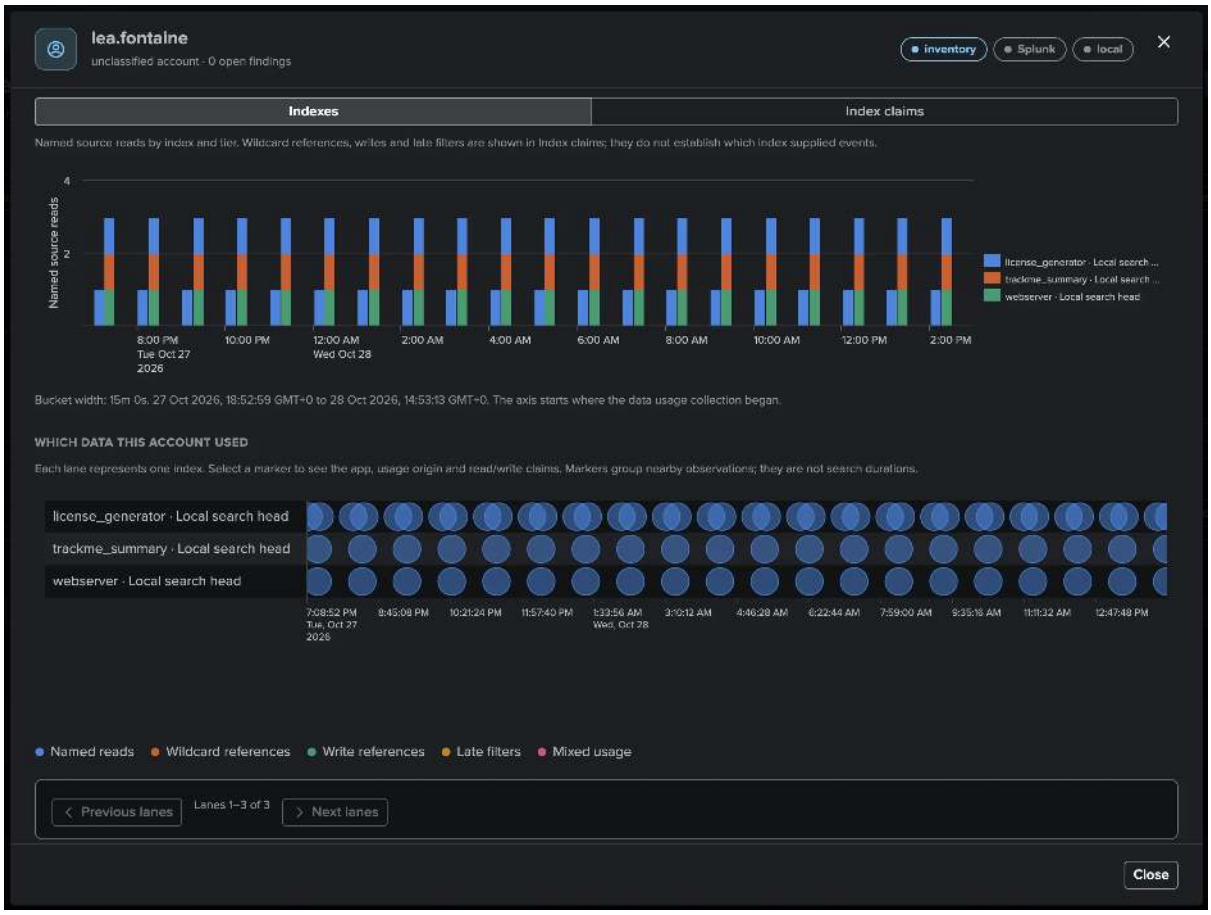
An object opens its detail: what it declares, what it read, how many searches ran and when.



Evidence — what supports these numbers. The collection behind the figures: the fact records read, the collection runs and their outcome, the index catalog of each tier, the indexers' confirmations, and the searches whose indexes could not be resolved, with the reason for each (a macro or saved search that cannot be found, a dynamic term, a wildcard role default) and the steps to resolve it.



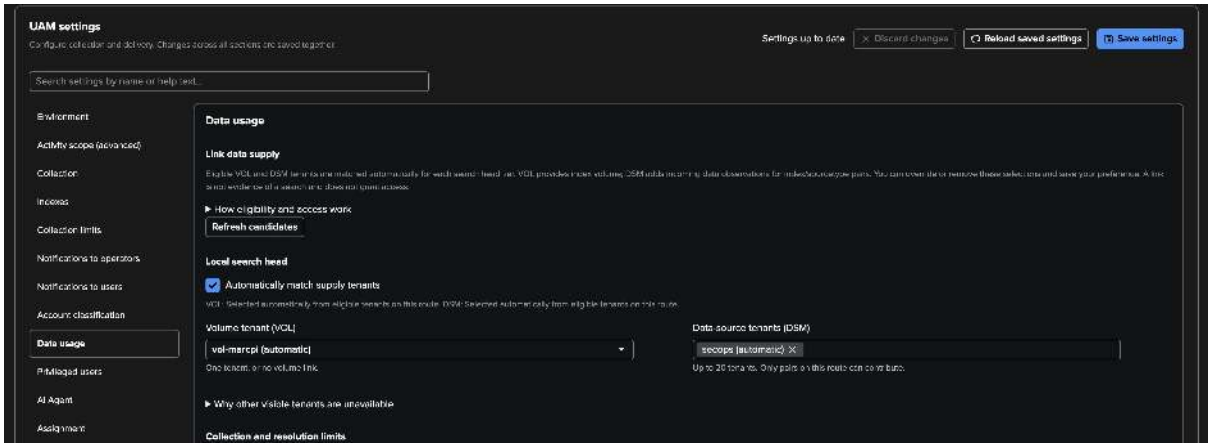
An account's data accesses. The account view has a **Data accesses** tab: the named reads of the account per index over the period, and **Which data this account used**, one lane per index with a marker for each group of nearby observations. Select a marker for the app, the usage origin and the claims behind it. An index view of the same timeline shows **who used this index**.



Searches run as long as they need. Every view reads the evidence through your own Splunk search jobs, started when the view opens and followed by the UI: the loader names what is still being read (*Still reading Collection runs…*), and the searches are cancelled when you change the selection or close the view. Nothing is cut after a fixed time: a long period or a large tenant takes longer, and is complete when it is shown. Only a safety timeout of 10 minutes applies.

Settings → Data usage. Each search head tier is linked to the VOL tenant that gives its index volumes and to the DSM tenants that give its incoming data. Matching is **automatic** by default — the first VOL tenant and every DSM

tenant you can see that is enabled and has the component, whatever account it collects through (2.4.23) — and you can override or remove a link. Indexes are matched by name: unselect a tenant that measures other indexers. A link grants no access: every read repeats the visibility check of the caller. The same section sets the apps whose reads are left out (`trackme` and `Splunk_SA_CIM` by default), whether internal indexes are shown, and the collection's own limits.



Note

What Data usage claims, and what it does not.

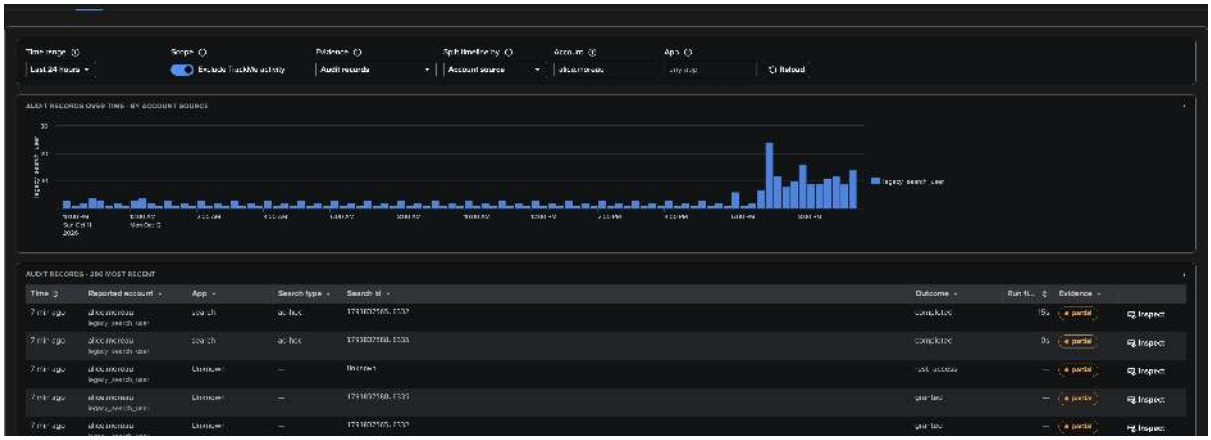
- A read is a search that **named** an index — directly or through a macro, event type, saved search or data model that resolves to it. A wildcard is a reference, not a read: `index=cribl_*` does not prove which index supplied the events.
- Each search is counted once. A search Splunk refused at its quota is not a read; a canceled or failed search that started is.
- A sourcetype is reported on a best-effort basis and never gets a verdict.
- The map covers what the audit trail of the indexing layer records and what the collection has seen so far. Until the collection covers the period, or seven days of it, nothing is called unread.
- A declaration in a knowledge object is not a read, and a link to a VOL or DSM tenant is not evidence of a search.

The collection behind the tab — the Data usage report, its facts, the resolution limits and the REST operations — is described in *Data usage*.

How each kind of claim is processed (named reads, wildcards and their narrowing to the reader's roles, searches the roles deny, definitions, index-less searches, late filters, writes, indexer confirmations, declarations, incoming data, verdicts and coverage), with diagrams, worked examples and limits, is the *Data usage reference: how each claim is processed*. The same reference opens in the product from every **How this works** link of the tab.

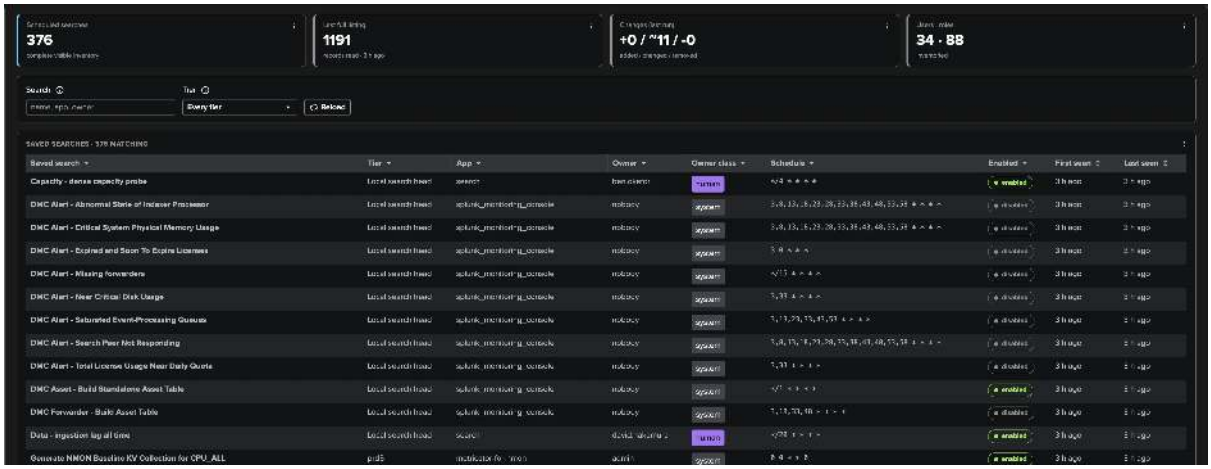
Activity

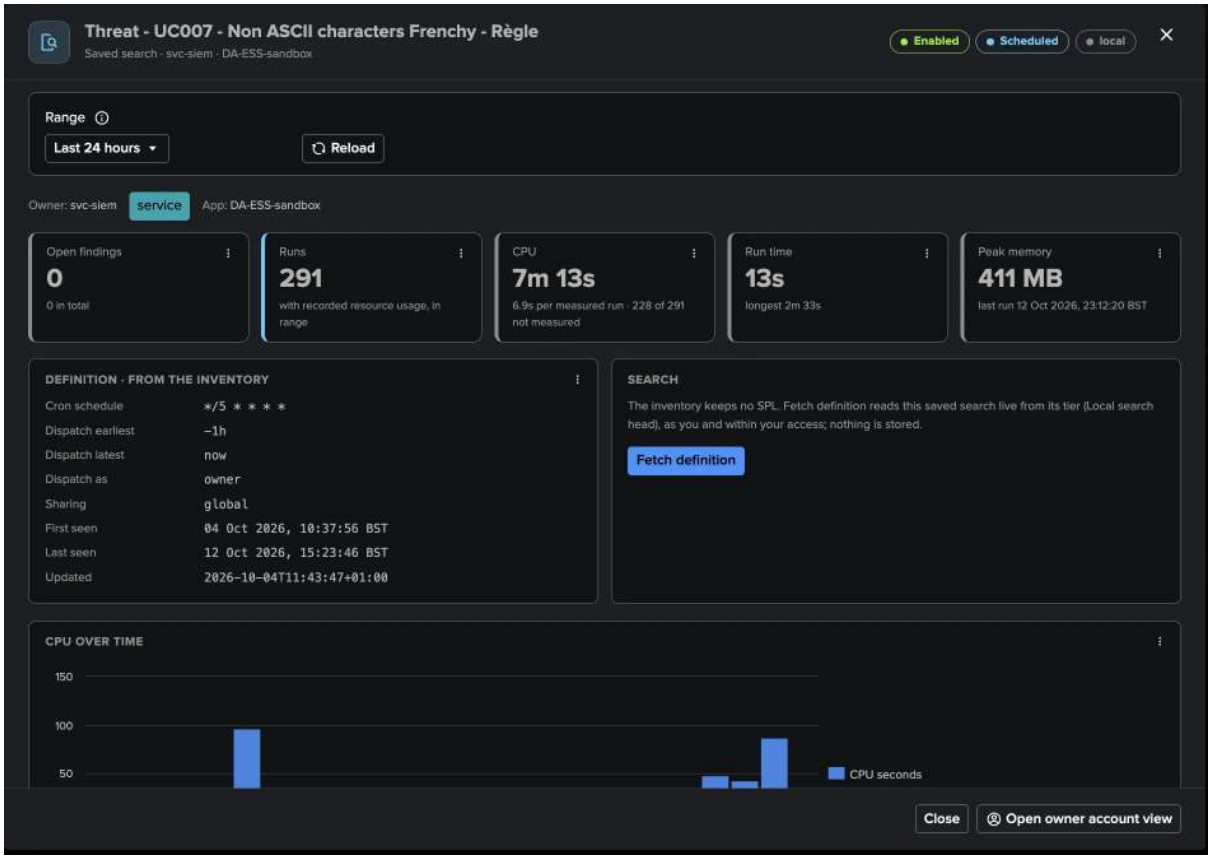
The raw evidence: **Audit records** or **Correlated executions** over a time range, a timeline split by account source, evidence status or account, account and app filters, and the 500 most recent records with an evidence inspector (**Inspect**). Executions are correlated from the audit records when you read them — the start and the completion of each search, the accounts and apps reported — and a notice says when an answer covers part of the range only.



Scheduled searches and the saved search view

The saved-search inventory per tier: tiles for the scheduled searches, the last full listing, the changes of the last run and the users and roles counts; the table with owner, owner class, schedule, enablement and first and last seen. Selecting a name opens the **saved search view**: its definition from the inventory, **Fetch definition** for the live definition, its runs with run time, CPU and peak memory, its findings and its change history, and a way to the owner's account view.

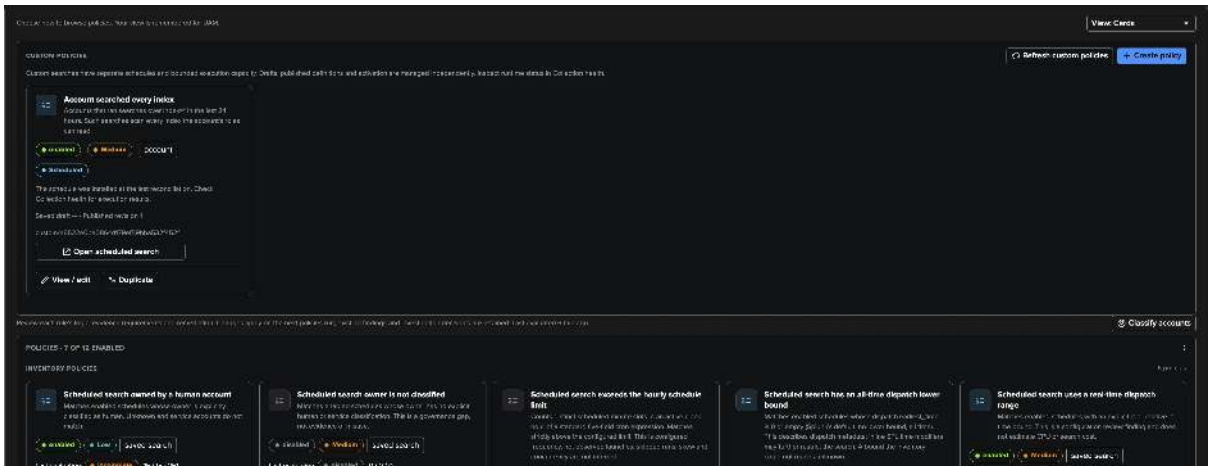




Peak memory can read *not measured* for short searches that Splunk's resource records never sampled. A saved search with the same name, app and owner on two tiers shares one set of run figures.

Policies

The built-in catalogue grouped by family (inventory, activity, resource, privileged) and your **custom policies**, shown as **Cards**, **Compact cards** or a **List** (remembered per user). Each policy shows its status, severity, account scope, thresholds (the effective values when they are derived from the capacity), exclusions, what it does on bank holidays and its **last evaluation**. Selecting the last evaluation opens the **policy insight**; **Edit** opens the policy editor. See *Tuning the policies*.





Audit changes

TrackMe's own audit of this tenant: every configuration change, decision and collection request, with who did what and when — the same view every tenant type offers.

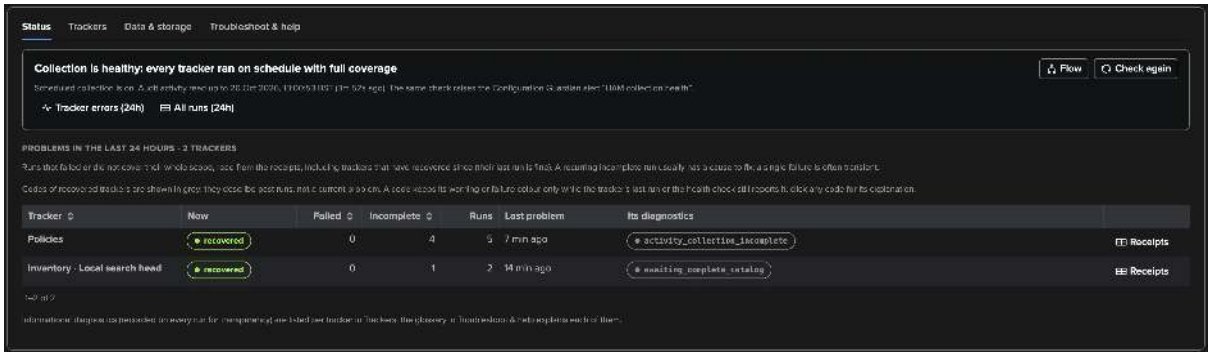
Collection health

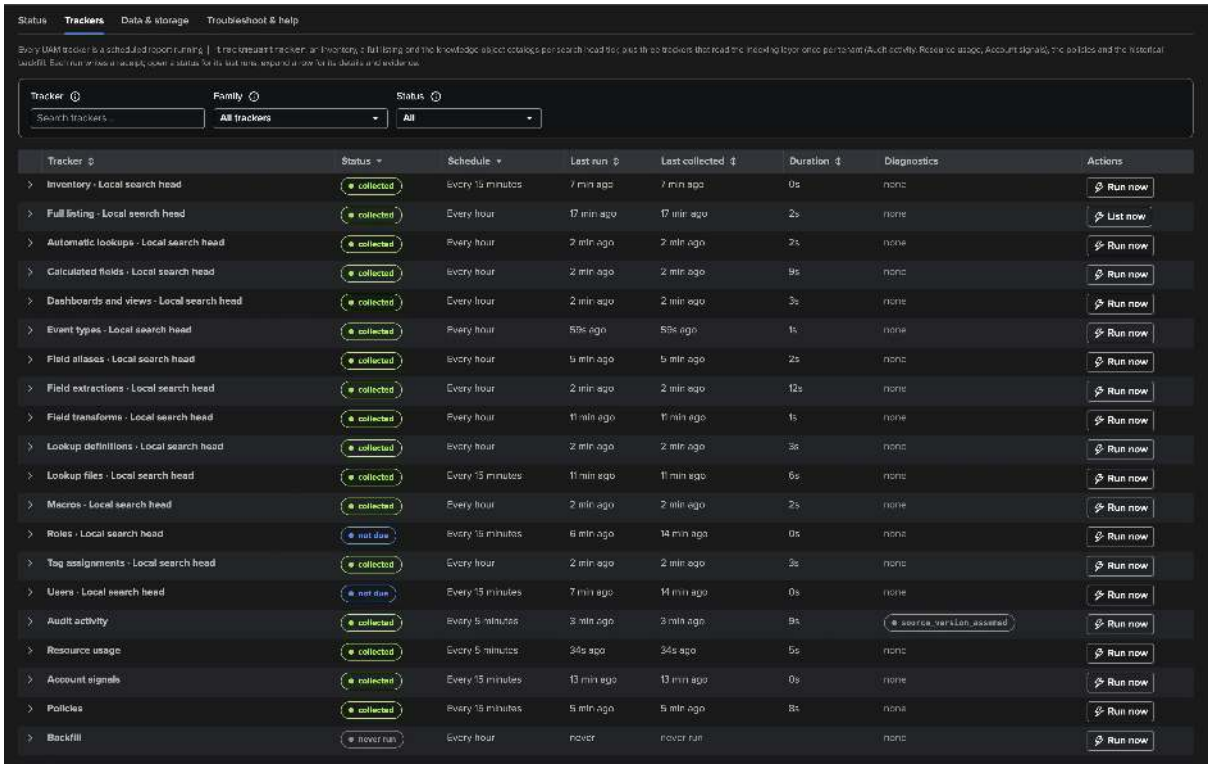
Is UAM collecting what it should, and if not, why and what to do? Everything that says something is wrong comes with its explanation, its evidence and its action. Four views:

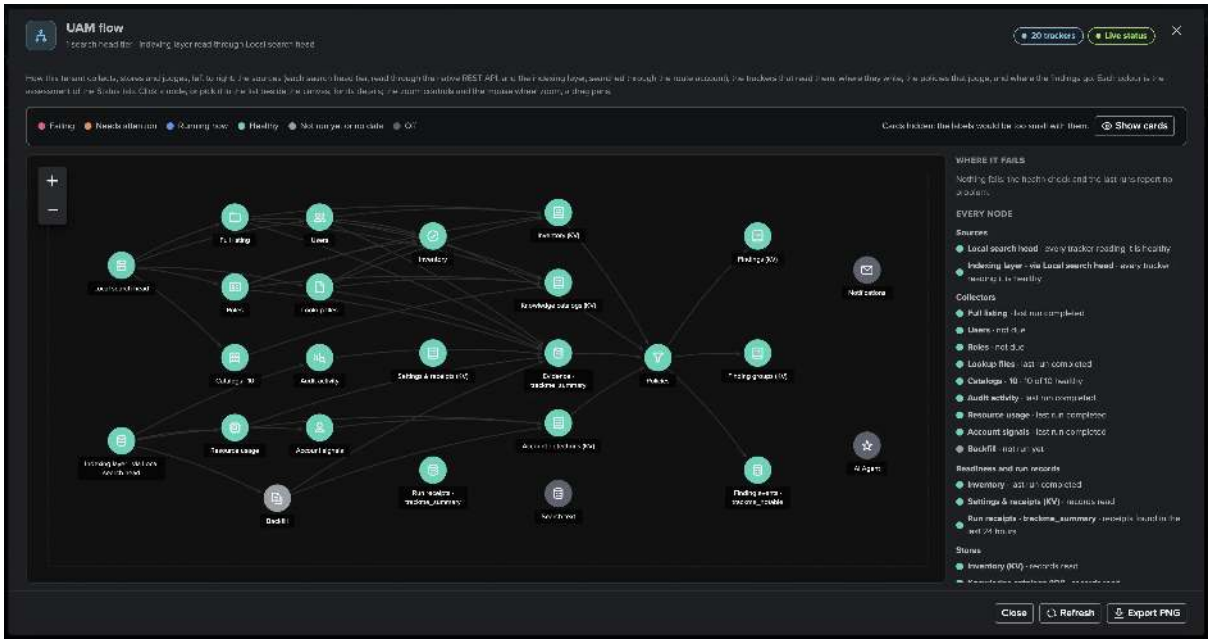
- **Status** — one row per tracker that needs attention, from the same assessment the Configuration Guardian raises, so the page and the alert never disagree. A row expands to every reason with its meaning and where it was reported, **What to do**, the facts of the last run (status, finish, consecutive failures, search job, report and its live schedule, lock), the policies a run could not complete (with **Show why**), shortcuts to the evidence (**Receipts**, **Tracker log**, **Search job**, **Report**, **Scheduler log**) and **Run now / Recover**. Below, **Problems in the last 24 hours** lists the failed and incomplete runs of each tracker, recovered or not. **Flow** draws UAM's own collection pipeline — the search head tiers and the indexing layer, the trackers, what they store and the policies — coloured by health.

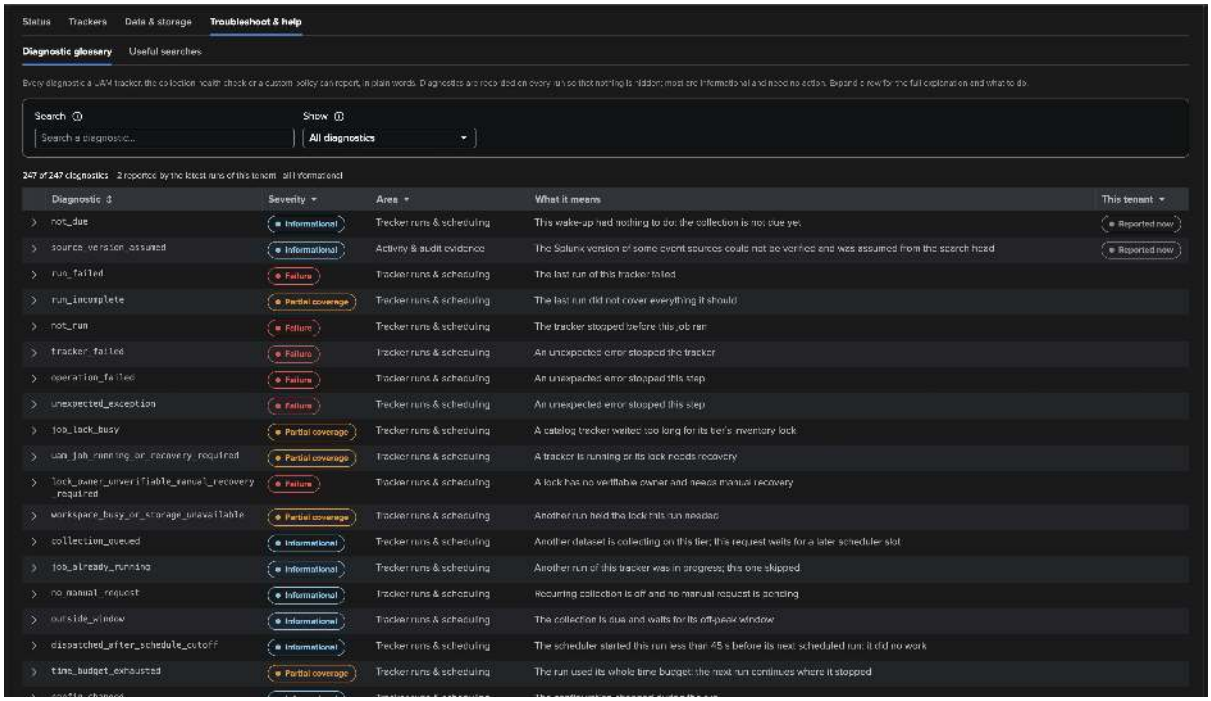
- **Trackers —** every tracker in one table: status, live schedule (read from Splunk), last run, **last collected**, duration, diagnostics, **Run now** (**List now** for a full listing) and **Recover**, filtered by name, family, status or tier. A row expands to its details: the activity checkpoint and the windows of the last run for Audit activity, the facts written, the claims and the unresolved searches for Data usage, the resource windows and the capacity for Resource usage, the login and automation scans for Account signals, the datasets of a catalog, the backfill progress and its controls (**Pause**, **Resume**, **Cancel backfill**, **Plan again**). For Audit activity, Data usage, Resource usage, Account signals and Policies, **Searches of the last run** lists each search the run sent, exactly as sent, with its time range, **Copy** and **Open in Search**. Custom policies have their execution health in their own family.
- **Data & storage —** what UAM writes for the tenant and how to read it: the events per index and sourcetype (volume over a range, latest event), the KV collections (records, size, lookup, and a record viewer), the logs and every scheduled report with its live state, each with a shortcut.
- **Troubleshoot & help —** the **diagnostic glossary** (every code UAM can report: what it means, where it comes from, possible causes and what to do; the codes this tenant reports now come first) and ready-made searches (tracker errors, failed or incomplete runs, skipped scheduled reports, REST calls, who changed UAM).

Every diagnostic code, wherever it appears, opens its explanation. A code is coloured as a warning or a failure only while its condition is current; a code that only describes a past, recovered run stays neutral. Only one run of a report happens at a time; a run whose search provably stopped is recovered automatically after five minutes, otherwise **Recover stopped run** releases it. A lock is never taken over on age alone.





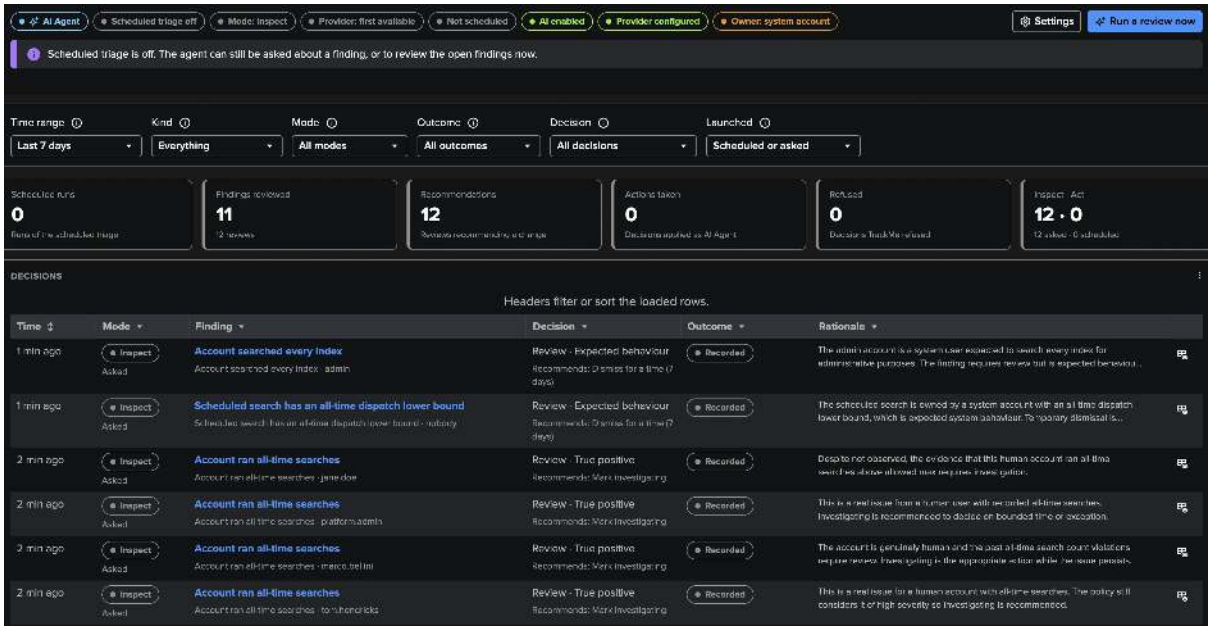




The Configuration Guardian check `uam_collection_health` warns when collection is not running as configured — a missing or unscheduled report, missed schedules, repeated failures, a checkpoint that does not move, a range held for 15 minutes, windows or recounts waiting more than a day, an overdue listing, failing notification delivery, a custom policy that drifts from its schedule. See *Configuration Guardian*.

AI Agent

The **UAM AI Agent**: whether scheduled triage is on, its totals over 24 hours to 90 days, **Run a review now**, and the decision log with its filters (decision, mode, outcome, scheduled or asked). See *The UAM AI Agent and the AI Assistant*.



Settings

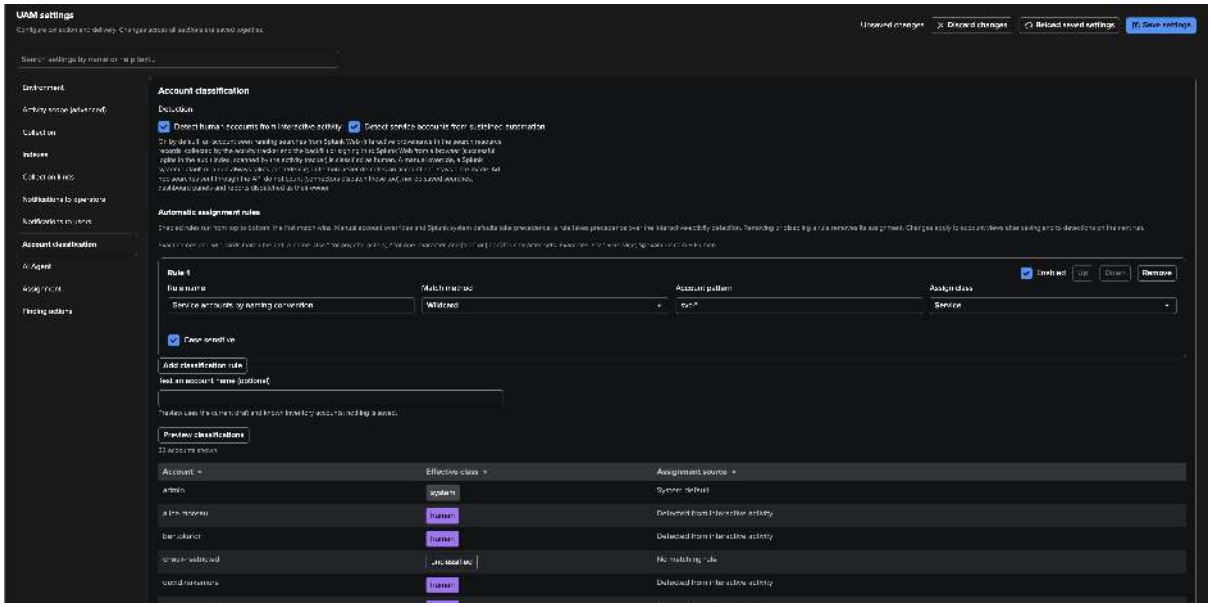
Every tenant setting in one page with one **Save settings**: **Environment** (the tiers and the route account), **Activity scope (advanced)**, **Collection** (scheduled trackers, *Store SPL search*, resource collection), **Indexes**, **Collection limits** (time budgets, windows and cadences: there is no row limit), **Notifications to operators**, **Notifications to users**, **Account classification**, **Data usage** (the VOL and DSM links, the excluded apps and the resolution limits), **Privileged users**, **AI Agent**, **Assignment** and **Finding actions**. Changes apply from the next runs; **Collect now** applies them immediately. If someone else saved in the meantime, your save is refused and you are asked to reload. Every setting is described in *Settings reference*.

Accounts and classification

A finding about an account only makes sense once you know what the account **is**. UAM keeps three classes, and the class decides which policies apply:

- **human —** a person. Detected automatically from **interactive activity**: a search run from Splunk Web (the Search app, a dashboard, a report opened by a person) or a browser login to Splunk Web. One is enough, and the class stays.
- **service —** automation. Detected after **seven distinct days** of automation-only activity (API jobs, the scheduler, acceleration) and nothing a person does; withdrawn as soon as the account does what a person does. This detection needs the classic `key=value _audit` format, and the seven days must be inside the audit retention.
- **system —** Splunk's context identities (`nobody`, `splunk-system-user`) and the built-in `admin`, preset. System accounts are never evaluated by the account policies and never receive email.

On top of the detection, set **manual classes** per account (Users tab or *Settings → Account classification*) and ordered **classification rules** (exact names or wildcard patterns such as `svc-*`), with a preview of the effective class before saving. Precedence is always: manual class, system defaults, the first matching rule, human detection, service detection. An account nothing classifies stays **unclassified** and is left out of the policies scoped to human accounts.



Tip

Classify the monitoring itself first. Classify the tenant owner (`svc-a`), your administrator accounts and other platform automation as **system** or **service**: they run TrackMe's collection and the platform's housekeeping and should not fill the findings. The `trackme` and `Splunk_SA_CIM` apps are excluded from the policies by default for the same reason.

Triage findings

Observed, not observed, and the status

A finding has two independent axes:

- **Whether the policy still sees the condition.** While it does, the finding is **observed**. After a complete policies run that no longer sees it, the finding reads **not observed** — UAM never resolves it for you: a configuration or a behaviour that disappears from one evaluation still deserves a human decision. A run that could not cover everything (an incomplete inventory, an activity gap) may add findings, never clear them.
- **Your decision**, the status: **Open**, **Investigating**, **Resolved** or **Dismissed**.

A resolved finding that is observed again is **reopened**. A dismissed finding stays dismissed until its dismissal ends — then it returns to open, so you get another look. The lifecycle shared with the other findings tenants is described in *A finding and its lifecycle*.

Finding groups

All the findings about **one account and the saved searches it owns** form one **Finding group**: the all-time searches and resource consumption of `analyst-a` together with the personal-schedule and real-time-schedule findings of the searches `analyst-a` owns. A saved search without an owner keeps a group of its own. You decide once, for the group: status, dismissal, assignee, notification snooze and severity apply to every member, and a finding that joins the group later takes its state.

Two UAM specifics:

- **Dismiss only this finding** (member menu) sets one finding aside — for example a personal schedule you accept inside a group that also holds a real problem. It stays dismissed on its own while the rest of the group is decided on; **Reopen this finding** brings it back into the group's state.
- A new kind of finding on a dismissed group returns the group to open with the new finding, while the findings the dismissal covered stay dismissed: a dismissal never silences a new problem.

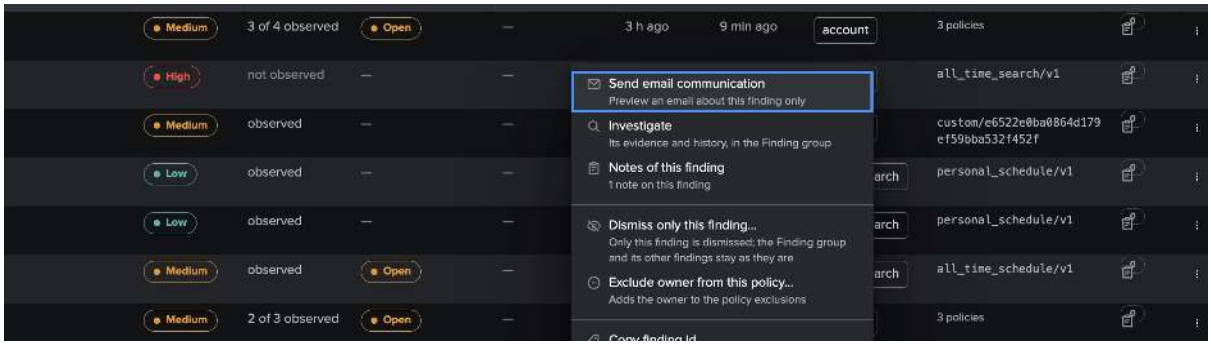
You can also group by hand — **Merge into one Finding group** (2 to 25 groups), **Add to the Finding group** (an existing manual group), **Remove from the Finding group**, **Ungroup** — for example to handle the accounts of one project together. The findings of one account always move together. See *Finding groups*.

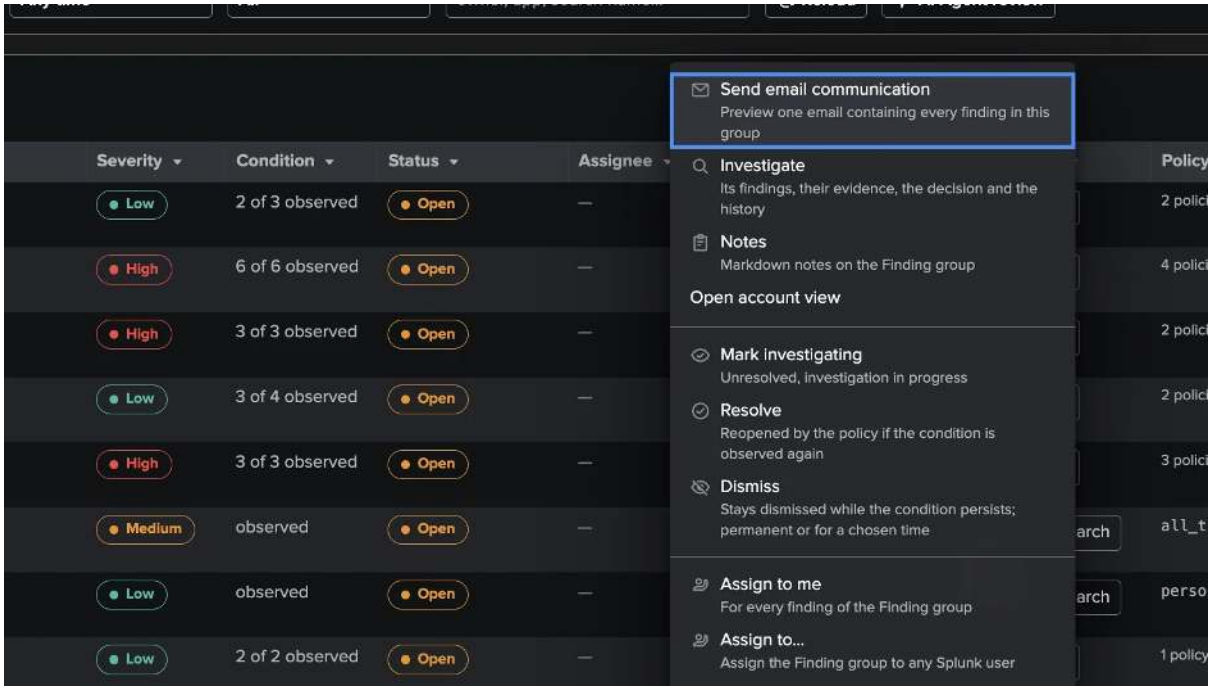
Decisions

From the row, group and member menus, the selection toolbar (**Bulk edit**) and the Triage panel of the Investigate view:

Decision	What it does
Mark investigating	Someone is on it.
Resolve	The investigation concluded. The policy reopens the finding if it observes the condition again. Nothing is changed on the saved search or the account.
Dismiss	No action needed: for 1, 7, 30 or 90 days, or **until reopened by hand** (permanent). A dismissed finding stays listed under the *Dismissed* tile.
Reopen	Back to open; on a dismissed finding, ends the dismissal now.
Assign to me, **Assign to…**, **Unassign**	To a Splunk user or an assignment group. The assignee is kept when the finding reopens.
Snooze notifications	Holds the operator emails (from 1 hour to 30 days) without changing the status; **Clear notification snooze** resumes them. It does not hold the notifications to users.
Set severity	Overrides the policy severity (Low, Medium, High, Critical); the override survives later runs. Choosing the derived severity again (**Members** on a Finding group, **Use policy severity** on a single finding) restores it.
Notes	Markdown notes on a finding or a Finding group, permanent or expiring, plus a short investigation note recorded with each decision.
Exclude owner from this policy	Adds the finding's owner to the exclusions of its built-in policy (custom policies write their exclusions in their SPL).

The row menu also has **Send email communication**, **Send webhook…** and your own actions under **Custom actions**, **Ask the AI Agent…**, the account view and saved search view, **Copy finding ID** and *Open record / events in search*.





Every decision is recorded in the history of the finding and of its Finding group, with the actor, the time and the note. Decisions are protected against concurrent changes: if the Finding group changed since you opened it, your decision is refused and the table reloads.

Assignment and findings actions

Assignment groups — "Platform team", "Security operations" — and ordered **routing rules** (by policy, severity, subject type, owner or account, app, environment) are set in *Settings → Assignment*; routing assigns new Finding groups automatically, never overriding an existing owner, and can email the assigned team. See *Assignment and automatic routing*.

Findings actions (*Settings → Finding actions*) add **drilldown searches**, **links**, **webhooks** through system-wide Webhook accounts, and the **email** action to the finding menus, by hand or through per-policy **automation rules** (policy editor → *Automations*). UAM presets include *Audit activity of this account*, *Scheduler runs of this saved search* and *Finding transitions*. Tokens such as `$subject.account$`, `$subject.owner$`, `$subject.app$` and `$subject.name$` carry the finding's values. See *Findings actions* and *Webhook accounts*.

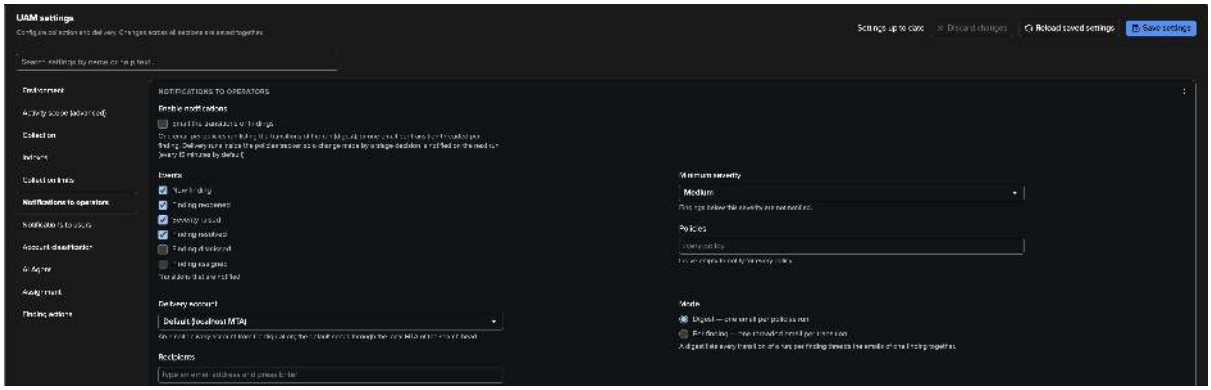
Every finding transition is also indexed in the tenant's notable index as `sourcetype=trackme:uam:finding`, ready for your own Splunk alerts, Enterprise Security correlations or SOAR playbooks.

Communicating with users

UAM can talk to two audiences: the **operators** who triage, and the **accounts** the findings are about. The two are independent.

Notifications to operators

Settings → Notifications to operators (off by default): the events that send an email (**New finding**, **Finding reopened**, **Severity raised**, **Finding resolved**, and optionally **Finding dismissed** and **Finding assigned**), the minimum severity (Medium), the policies (all), the email delivery account, up to 50 recipients, and the mode: a **Digest** (one email per policies run) or **Per finding** (one thread per finding). Emails are sent at the end of the next policies run, within about 15 minutes, and link to the finding. **Send test email** uses the saved settings, even while notifications are off.

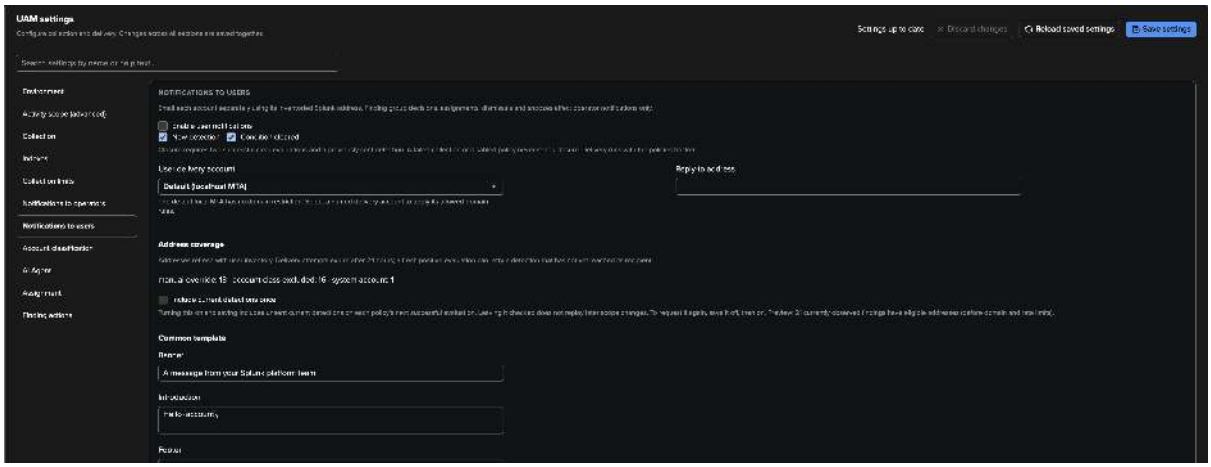


Notifications to users

Settings → Notifications to users (off by default) emails **the account a finding is about**, so that people fix their own searches:

- on a **New detection**, and when the **Condition cleared** (after two consecutive complete evaluations without it);
- addresses come from Splunk's user records on each tier, or from an administrator's **Override email address**; system accounts never receive mail, human accounts do by default, service and unclassified accounts only if you opt in;
- a common banner, introduction, footer, reply-to address and HTTPS support link, combined with each policy's own subject and body (*Policies → Edit → Email template*);
- limits: 50 emails per run, 3 per address per hour, 24 hours between repeat detections.

When you turn it on, UAM records the current findings as a baseline and only emails what is detected from then on, unless you choose **Include current detections once**. User emails follow the policy's detection, not your triage: resolving, dismissing or snoozing a finding does not stop them.



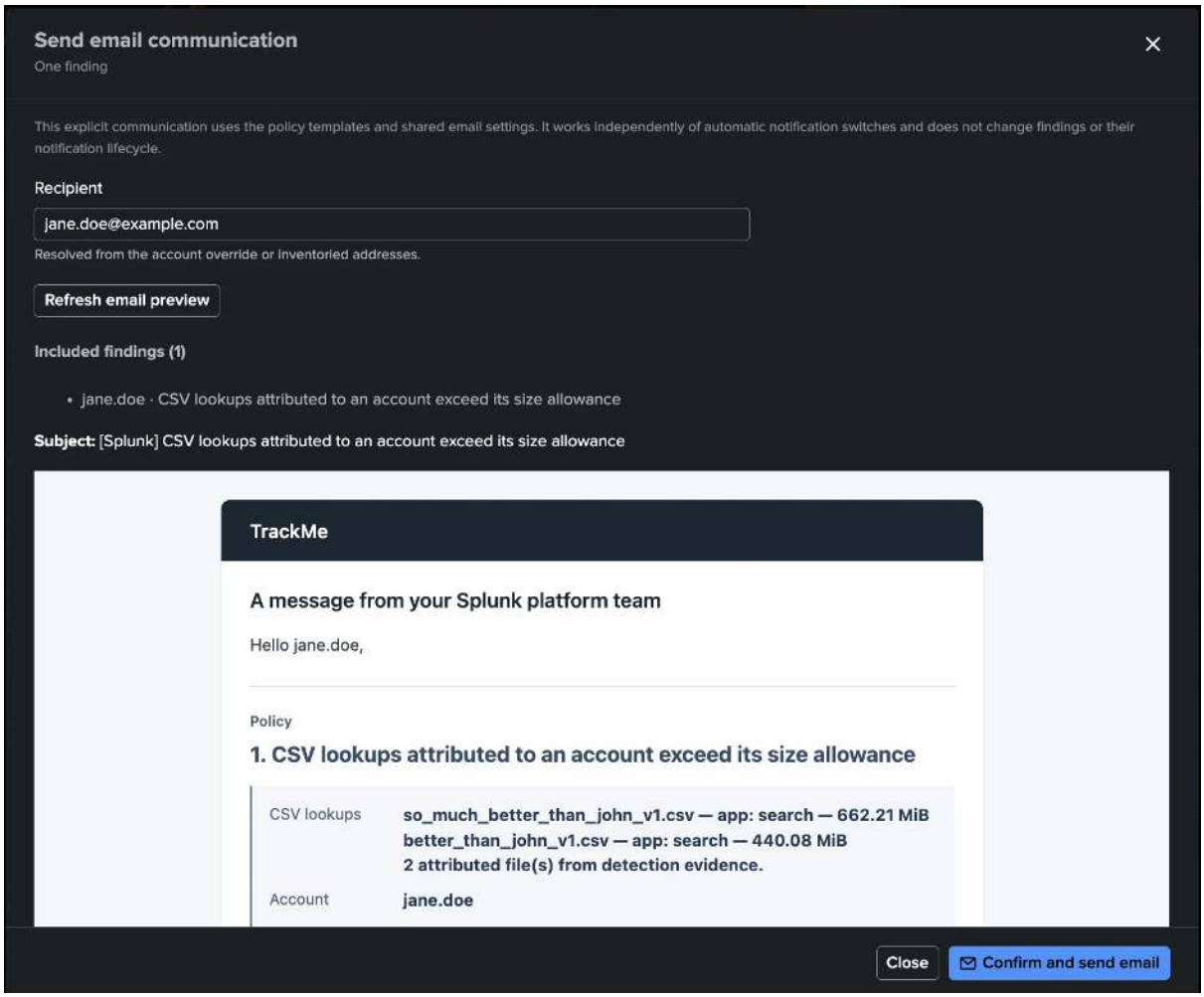
Important

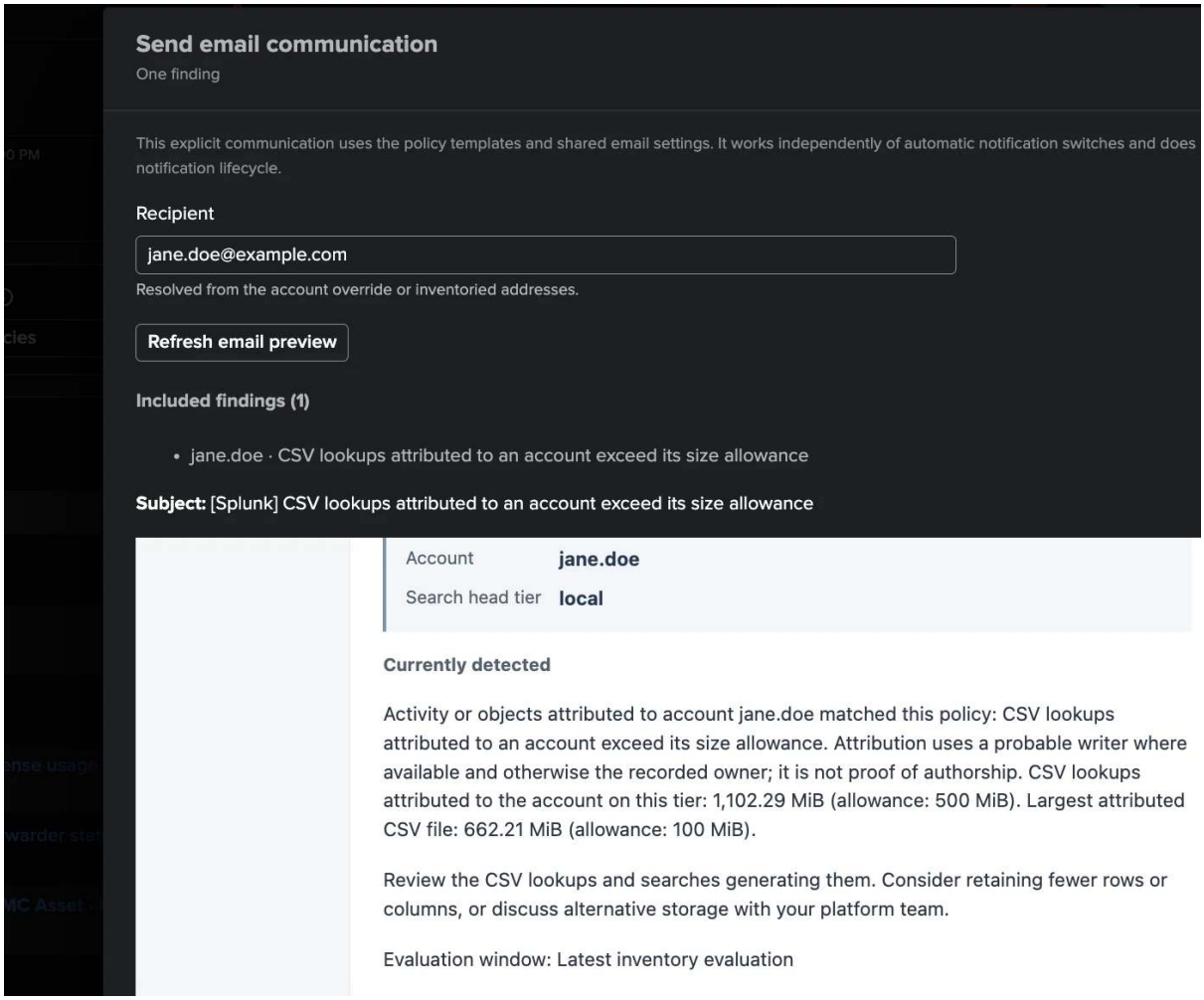
Before turning on notifications to users, check the **address coverage** shown in the section, set the reply-to address and the support link to your team, and preview the email template of each enabled policy. The people who receive these emails did not ask for them: make the first one clear and useful.

Send email communication

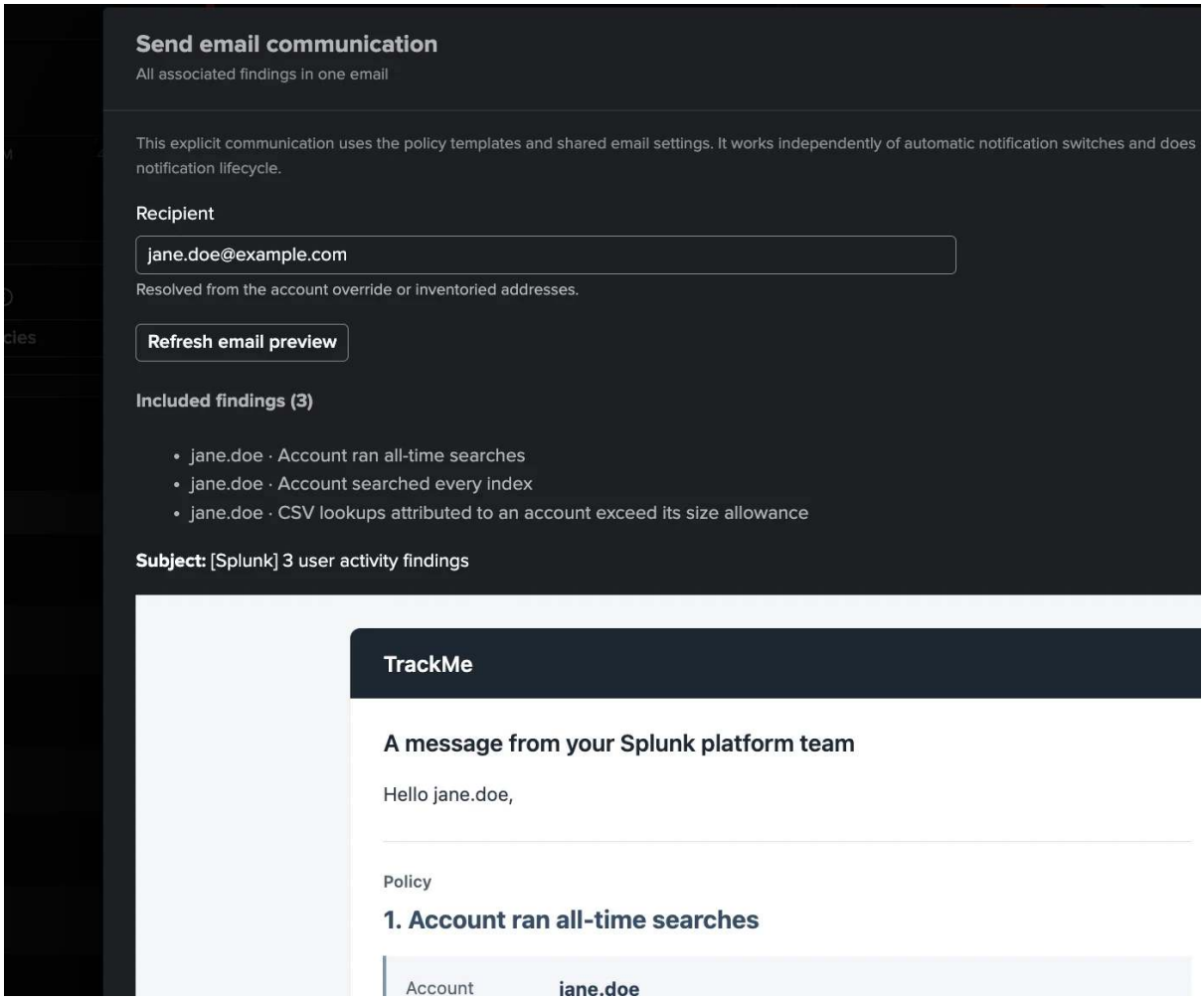
For a one-off message, **Send email communication** in a finding, member or Finding group menu prepares an email to the account concerned — one finding, or a whole group (up to 200 findings) in one email. You review the preview,

can change the recipient once, and confirm explicitly. The email is sent by a TrackMe search job, never by the browser, and a delivery that could not be confirmed is reported, never resent blindly.





From a Finding group menu, the same dialog prepares one email for every finding of the group: here the three findings of `jane.doe` — a built-in activity policy, a custom policy and the CSV lookup footprint — under the subject *3 user activity findings*.



The UAM AI Agent and the AI Assistant

The **UAM AI Agent** reviews findings the way an operator would: it reads the evidence TrackMe holds and writes its assessment as one current `[AI Agent]` note per finding.

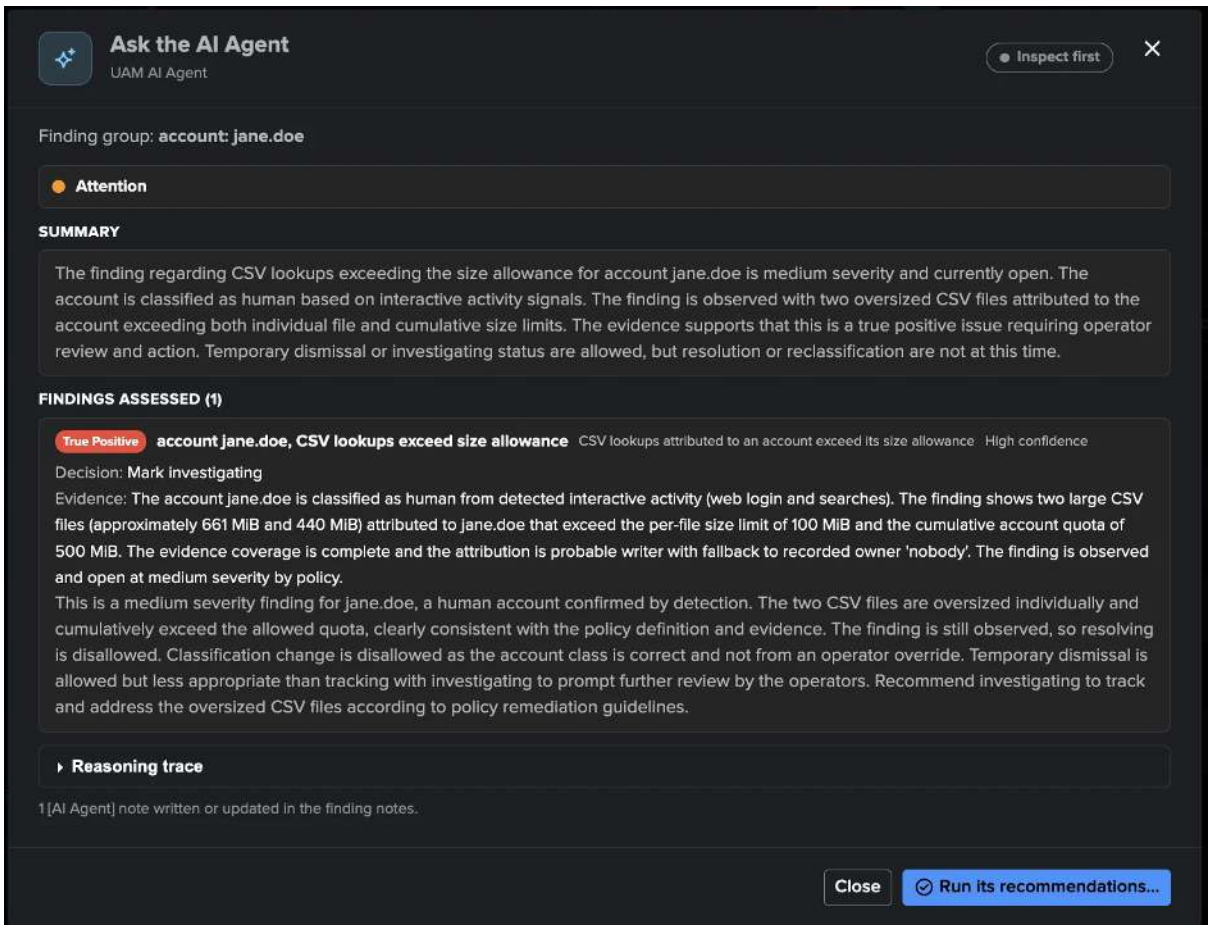
- **Inspect** changes nothing: assessment, confidence and recommendation.
- **Act** also applies the decisions the tenant allows: *mark investigating* and *dismiss for a time* (on by default, never for a critical or high finding still observed), *resolve* a finding its policy no longer observes and *classify* an unclassified account (off by default). It never dismisses permanently, assigns, or changes a severity, a policy or a setting.

Ask the AI Agent… in a finding or group menu inspects one Finding group on demand; you can then let it act on that inspection. **Scheduled triage** (*Settings → AI Agent*, off by default) reviews up to 10 findings every hour — by default only **critical and high** ones, one per Finding group — and never overrides a decision or an assignment made by a person. Every decision appears under the actor **AI Agent** in the history and in the **AI Agent** tab. The AI Agent needs the AI Assistant to be enabled with at least one AI provider; the scheduled triage also needs a tenant owner other than `nobody` or `splunk-system-user`. The UAM AI Agent replaces the UAM Findings Advisor of earlier releases. See *Findings AI Agents* and *AI Agents for findings*.

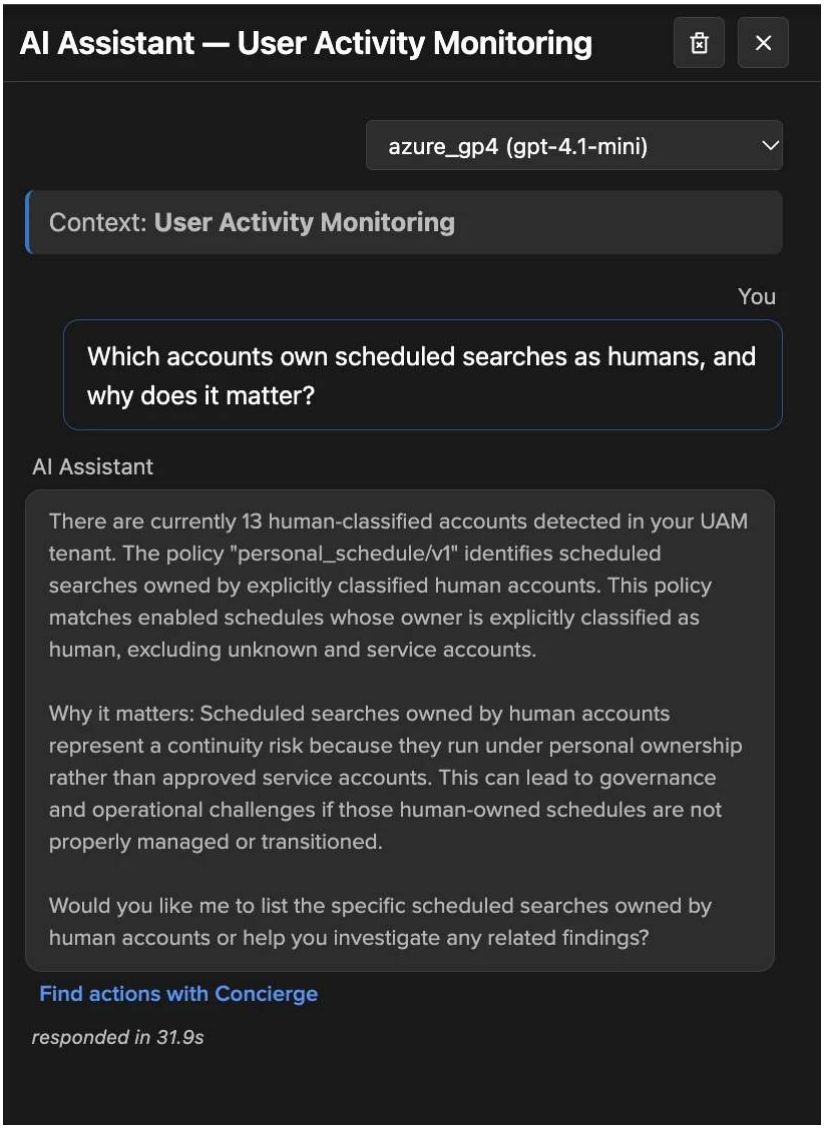
Custom instructions. The tenant administrators can give the agent instructions in plain English (*Settings → AI Agent → Custom instructions*, also in *Manage AI Agents automation* and in the *Configure tenant* dialog): at most 4,000 characters, line breaks kept. Every run of the agent follows them, asked or scheduled — for example, "The accounts named `svc_*` are service accounts." They never widen what the agent may do: the allowed decisions and TrackMe's checks before every decision still decide. The tenant's shared **Custom Instructions** of *Manage AI Agents automation* are different: they apply to scheduled runs only.

Evaluate… tries the unsaved draft before you save it: the agent runs in inspect mode on one open finding, or on

a review of the open findings (at most 10, the most severe first), optionally side by side with a run without custom instructions. An evaluation writes nothing — no note, no decision, no review record — and is recorded with the status `evaluation`. See *AI Agents for findings*.



The **AI Assistant** of a UAM tenant knows the workspace, the policies and the open findings: ask it to explain a finding, compare accounts or suggest what to tune, and it can propose an AI Agent review or a triage action for you to confirm. See *AI Assistant & Concierge*.



Tuning the policies

The built-in catalogue

Sixteen policies in four families. Seven are enabled on a new tenant; the four privileged policies are off until you turn them on.

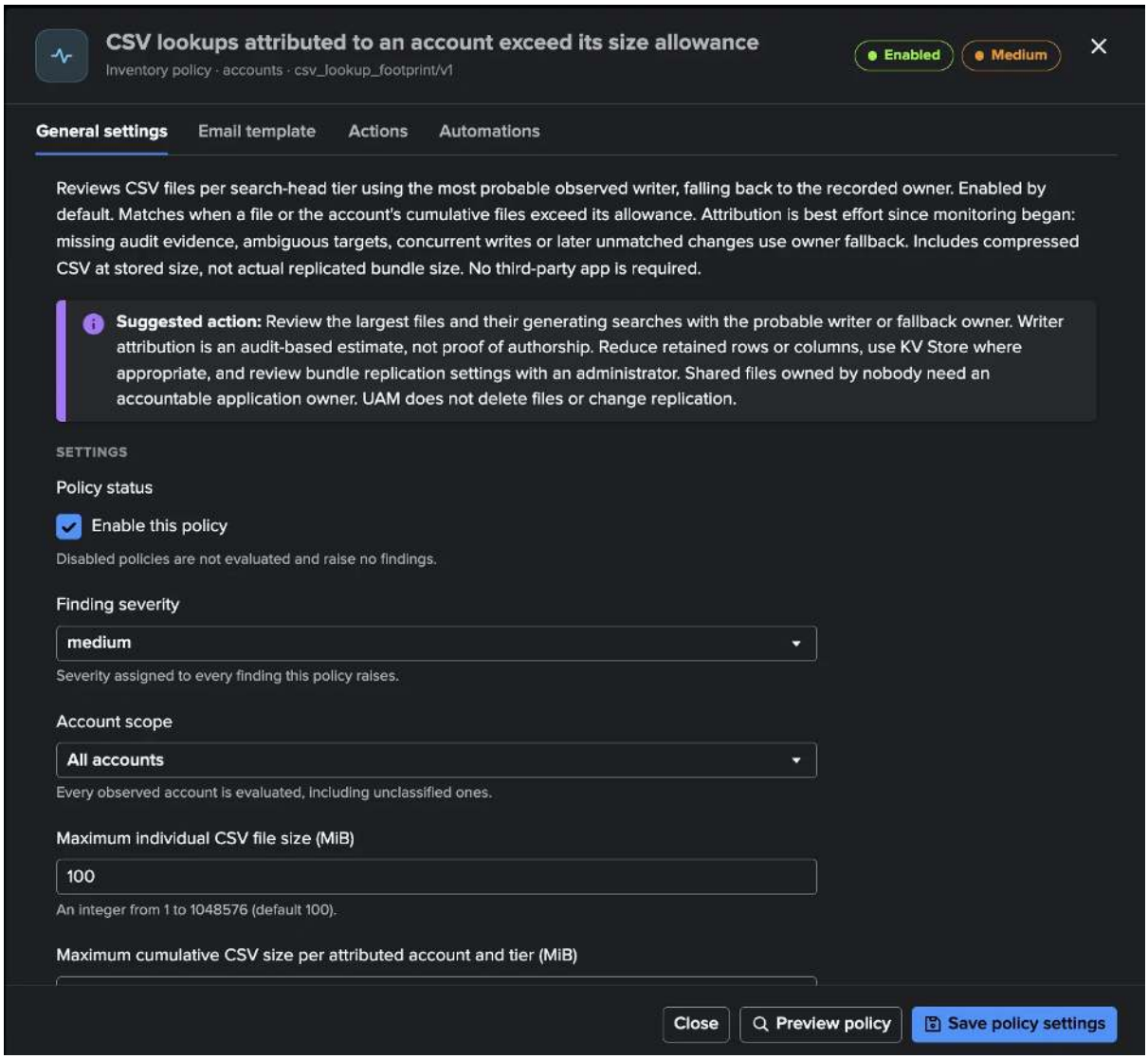
Policy	Family	Default	What it detects
Scheduled search owned by a human account (`personal_schedule/v1`)	inventory	on, Low	An enabled schedule owned by a person: it stops when that account goes.
Scheduled search owner is not classified (`unclassified_schedule/v1`)	inventory	off, Medium	An enabled schedule whose owner is neither human nor service.
Scheduled search exceeds the hourly schedule limit (`dense_schedule/v1`)	inventory	off, Medium	A cron with more minute slots per active hour than the limit (12).
Scheduled search has an all-time dispatch lower bound (`all_time_schedule/v1`)	inventory	on, Medium	A schedule whose earliest time is `0` or empty.
Scheduled search uses a real-time dispatch range (`realtime_schedule/v1`)	inventory	on, Medium	A real-time scheduled search.
CSV lookups attributed to an account exceed its size allowance (`csv_lookup_footprint/v1`)	inventory	on, Medium	Per tier, one CSV lookup over the file limit (100 MiB) or an account's CSV lookups over its total (500 MiB), attributed to the probable writer, else the recorded owner.
Account launched an unusual number of searches (`launch_burst/v1`)	activity	off, Medium	More distinct searches in the policy window than the limit (200).
Account ran repeated long searches (`costly_search/v1`)	activity	off, Medium	More than 3 searches of at least 300 seconds.
Account ran all-time searches (`all_time_search/v1`)	activity	on, **High**	Any all-time search (the maximum is 0).
Account's search resource consumption exceeded its allowance (`resource_consumption/v1`)	resource	on, Medium	CPU over the allowance in the window (10 % of the search capacity by default); the severity is raised one level when the account dominates the environment (40 %).
Account ran repeated resource-intensive searches (`resource_intensity/v1`)	resource	on, Medium	More than 3 searches heavy in CPU or memory.
Account's CPU consumption is unusual for its daily baseline (`resource_deviation/v1`)	resource	off, Medium	Today's CPU above a multiple (3) of the account's daily 95th percentile over 14 days. Ships in **Preview mode**: it shows what it would match, without raising findings or sending notifications, until you turn the mode off.
Account became privileged (`privileged_account_new/v1`)	privileged	off, **High**	An account that became privileged: created privileged, given a privileged role, or a role it holds now imports one. Reported once per change, with who made it when Splunk audited it.
Role gained a privileged capability (`privileged_role_capabil:v1`)	privileged	off, **High**	A role whose own capabilities or imported roles changed so that it grants more privileges, or a new role created privileged. One finding about the **role**, listing the roles that inherit it and the accounts newly privileged through them.
Privileged account is dormant (`privileged_account_dorma v1`)	privileged	off, Medium	A privileged account with no search, no Splunk Web login and no successful login reported by Splunk for 30 days.
Privileged accounts differ from the reference (`privileged_reference/v1`)	privileged	off, **High**	A privileged account your reference (typed names and lookup tables) does not declare; optionally a declared account that holds no privilege or no longer exists (Low).

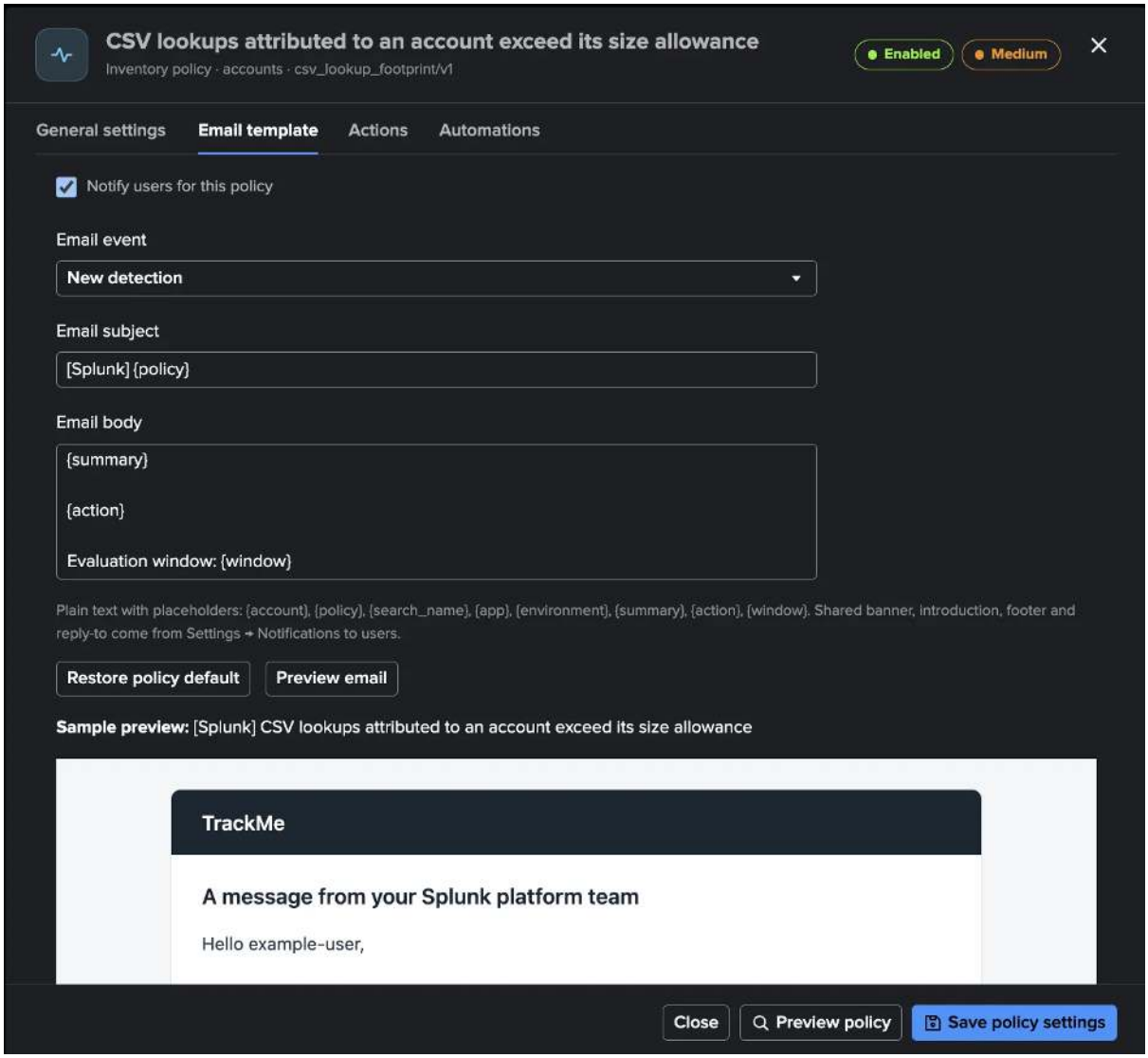
The activity and resource policies judge 15-minute windows of collected evidence by default. A window ends where the collected evidence ends, never at the current time, and consecutive windows always touch: no minute is skipped, and activity collected late is evaluated once it is collected. The inventory policies follow the catalog cadence (the full saved-search listing is daily by default). The resource policies stay **learning** — no finding, nothing cleared — until the tenant holds 24 hours of resource history and a measured capacity. The full catalogue, with every threshold and its bounds, is in *The built-in policy catalogue*.

Editing a policy

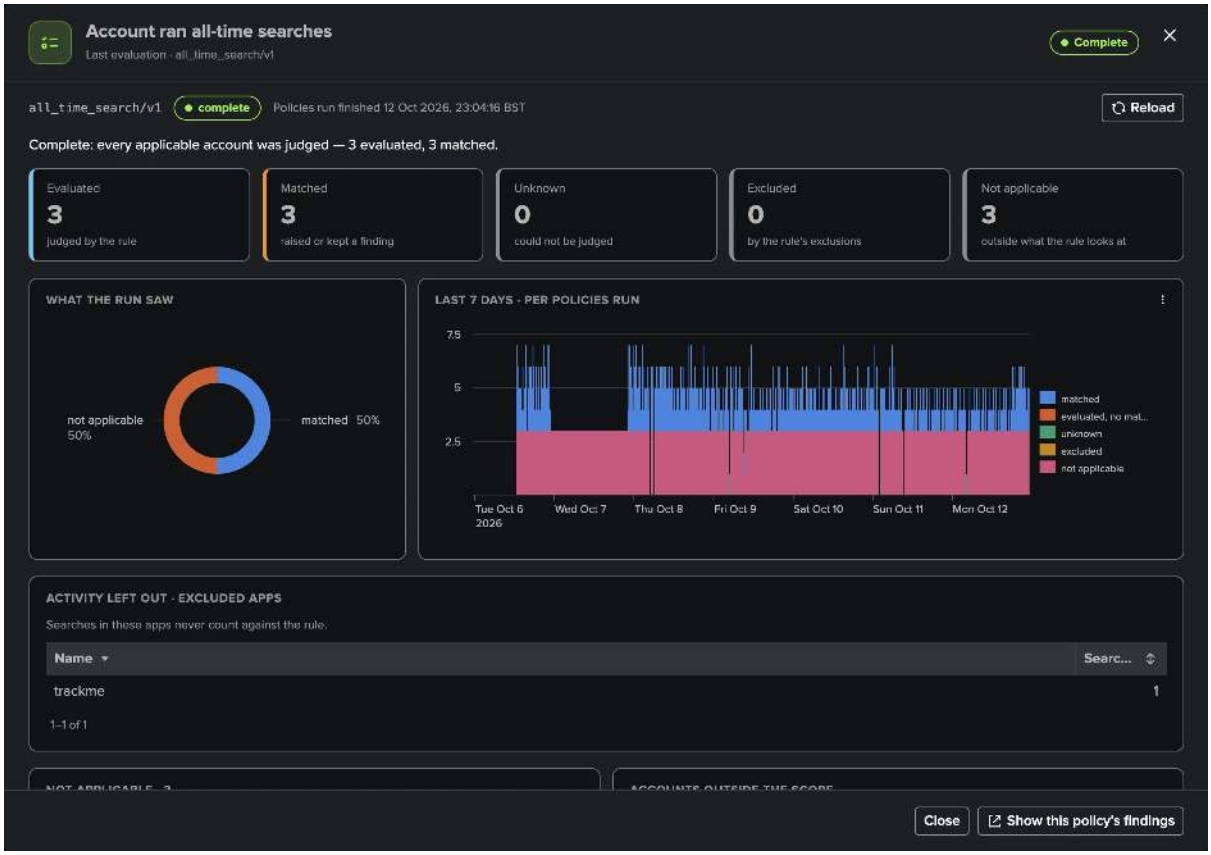
Edit opens the policy editor:

- **General settings** — status, **During bank holidays** (see *Bank holidays*), severity, account scope (human accounts only, classified accounts, all accounts), thresholds (the resource policies can derive them from the measured capacity, **auto thresholds**), excluded owners and excluded apps (exact names, up to 500), and **Preview policy** to see what the draft would raise against the current data before you save.
- **How it is evaluated** — the rule in plain words, then the exact searches it runs (built from the saved settings, with their time range, **Copy** and **Open in Search**), or, for an inventory policy, the records it judges.
- **Email template** — the subject and body of the user emails for this policy, for a new detection and for a cleared condition, with a preview and a test delivery; **Restore policy default** brings back the shipped text.
- **Actions** and **Automations** — the findings actions offered on this policy's findings, and the automation rules that run a webhook when they change.





The **policy insight** (from the last evaluation of a policy) explains a run: the verdict, what the run saw, what was matched, excluded, not applicable or could not be judged and why, the owners behind the unknowns (classifiable in place), the exclusions and a seven-day history, with the runs of those seven days that could not judge everything and their reasons (kept after the problem recovered). It is the first place to look when a policy raises nothing, or too much.



Privileged policies

Four policies watch who holds high privileges, from the assessment of the **Privileged users** tab (its rules: *Settings → Privileged users*). They are off by default: turn on the ones you need in *Policies*.

- **Account became privileged** reports, once, an account that became privileged on a tier: a new account created privileged, a privileged role assigned to it, or a role it holds that now imports a privileged role. When Splunk audited the change, the finding names who made it and when; a role mapped by a SAML or LDAP provider has no such record.
- **Role gained a privileged capability** reports, once, a role whose **own** definition changed so that it grants more: its own capabilities, the roles it imports, or a new role created privileged. Inheritance is transitive, so one change reaches every role that inherits it, directly or through other roles: the finding is about the role whose definition changed, and lists the roles that inherit it and the accounts newly privileged through them — one change, one finding.
- **Privileged account is dormant** reports a privileged account with no activity for 30 days (configurable): no search, no Splunk Web login, no successful login that Splunk reports on any tier. It judges an account only once it has been in the inventory and the tenant's evidence covers the whole period.
- **Privileged accounts differ from the reference** compares the privileged accounts with a reference you maintain — the accounts declared privileged, optionally per search head tier. It reports a privileged account the reference does not declare (High) and, if you keep *Report stale references* on, a declared account that holds no privilege or no longer exists (Low).

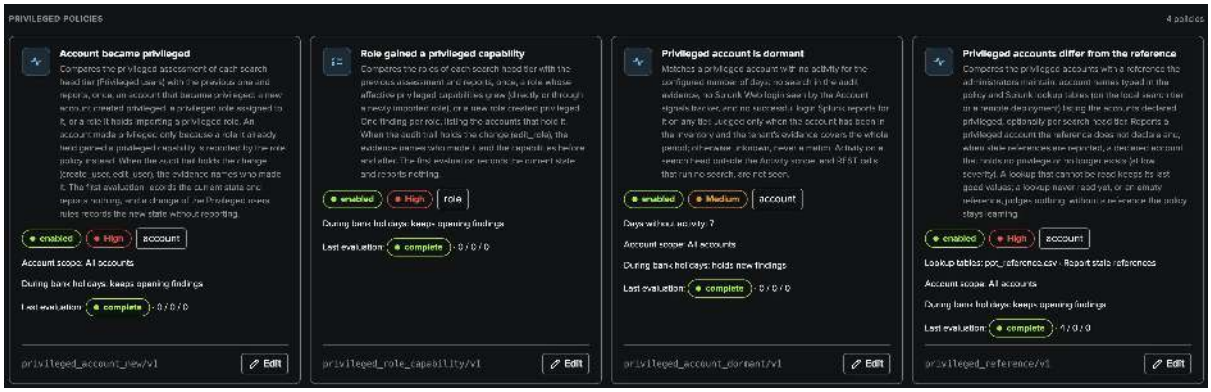
The reference is set in the policy editor, from two sources that add up, case-insensitive:

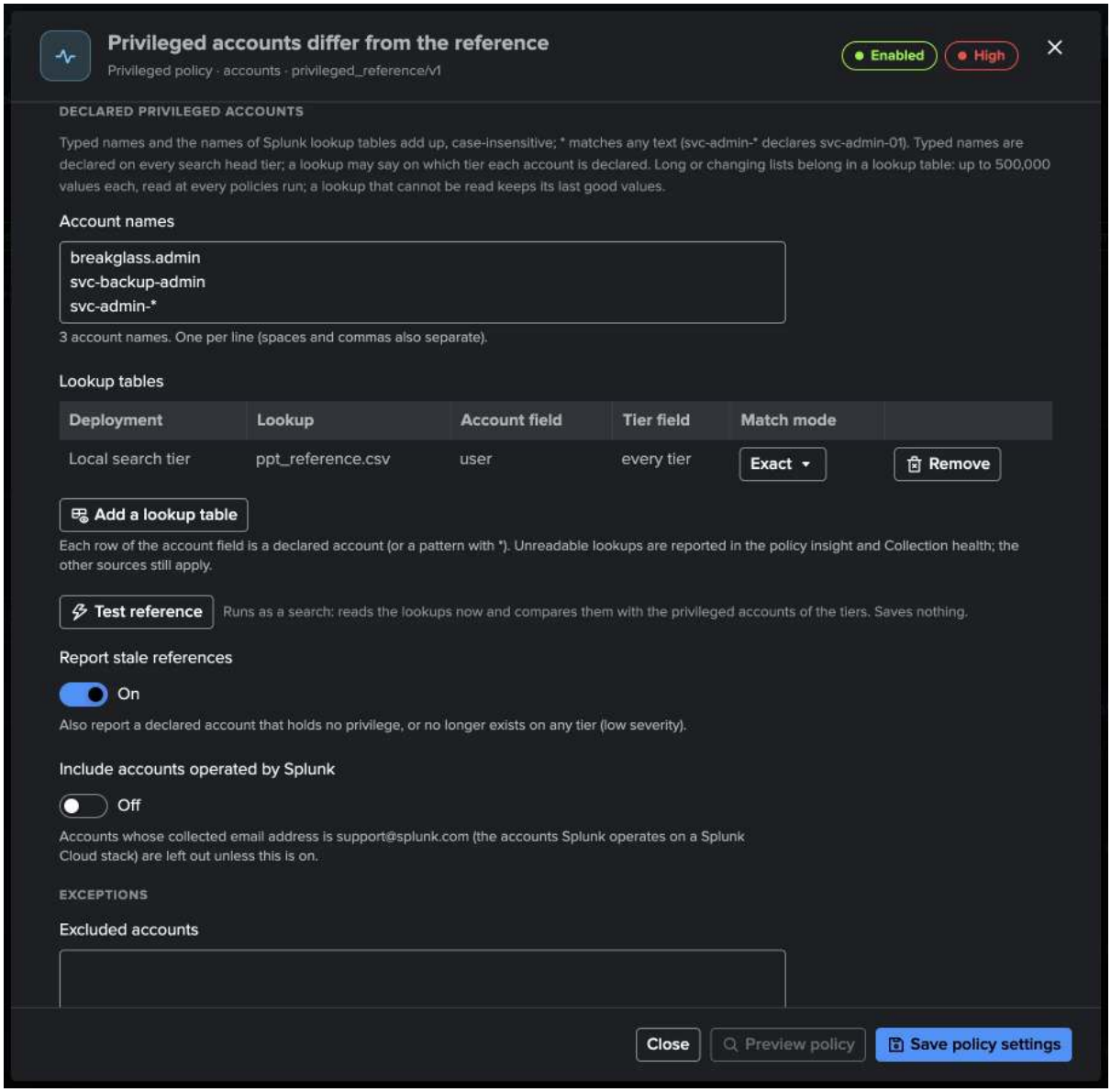
- **Account names** typed in the policy (up to 1,000), declared on every search head tier; `*` matches any text (`svc-admin-*`).
- **Lookup tables** (up to 20): **Add a lookup table** picks the Splunk deployment (the local search tier, or a remote deployment through a TrackMe remote account), the lookup (a CSV or KV store lookup shared with the TrackMe app, with a preview of its first rows), the **account field**, an optional **tier field** (the search head tier a row declares the account for: `local` or a remote account name; not set: every tier) and the match mode

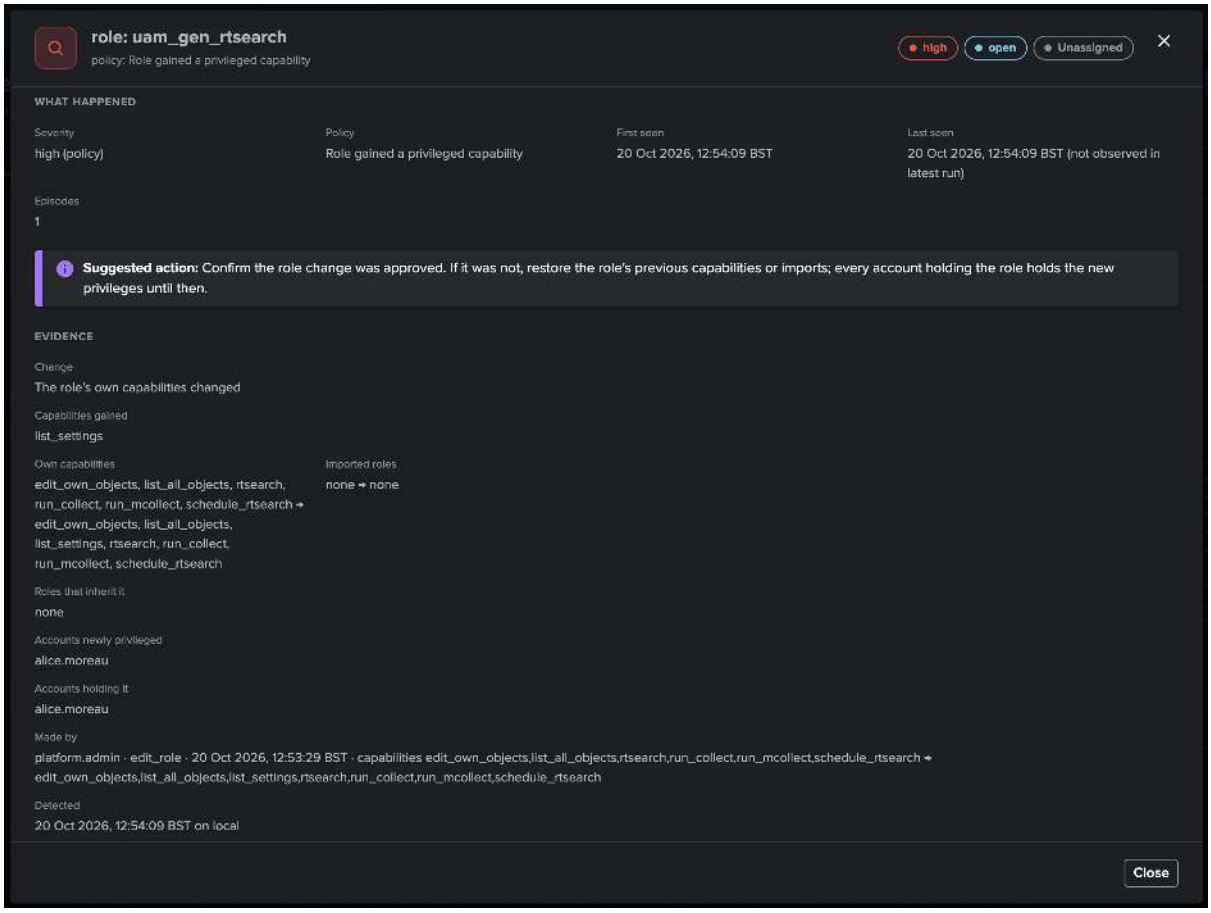
— **Exact** (a value holding `*` is ignored) or **Wildcard**. Each lookup holds up to 500,000 values and is read at every policies run; a lookup that cannot be read, or reads empty, keeps its last good values, and one never read yet makes the policy judge nothing until it can be read. TrackMe's own lookups are not offered.

Test reference reads every lookup now, as a search, and shows what each source declares, the privileged accounts the policy would report as undeclared, the stale references and the patterns that declare nothing. It saves nothing.

The first evaluation of a change policy records the current state and reports nothing, and a change of the Privileged users rules records the new state without reporting. The accounts Splunk operates on a Splunk Cloud stack (their email address is `support@splunk.com`) are left out unless you include them. These policies never email the account a finding is about; notifications to operators work as for every policy. A role finding has no account to write to: *Send email communication* is not offered on it.



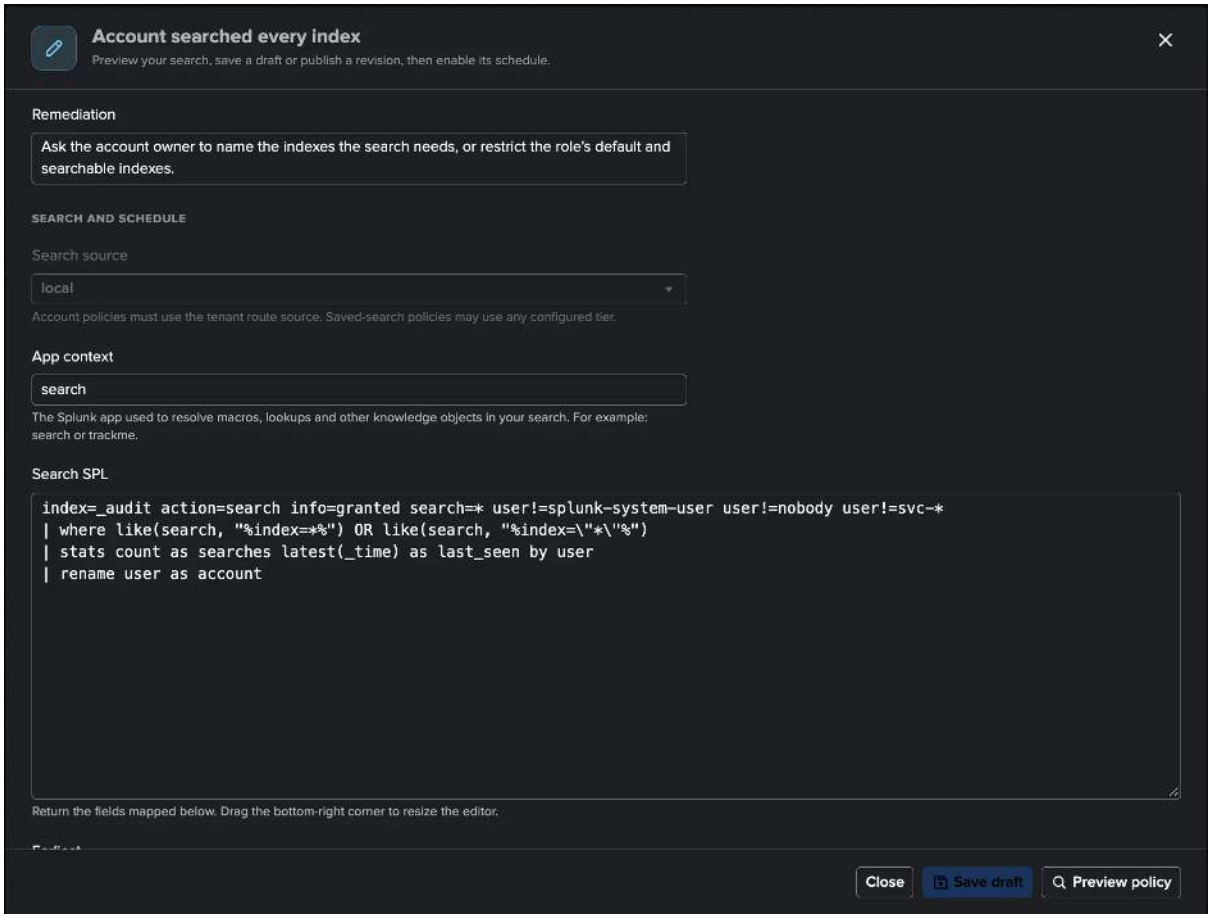

Privileged accounts differ from the reference
Privileged policy · accounts · privileged_reference/v1
Enabled
High
DECLARED PRIVILEGED ACCOUNTS
Typed names and the names of Splunk lookup tables add up, case-insensitive; * matches any text (svc-admin-* declares svc-admin-01). Typed names are declared on every search head tier; a lookup may say on which tier each account is declared. Long or changing lists belong in a lookup table: up to 500,000 values each, read at every policies run; a lookup that cannot be read keeps its last good values.
Account names
breakglass.admin
svc-backup-admin
svc-admin-*
3 account names. One per line (spaces and commas also separate).
Lookup tables
Deployment
Lookup
Account field
Tier field
Match mode
Local search tier
ppt_reference.csv
user
every tier
Exact
Remove
Add a lookup table
Each row of the account field is a declared account (or a pattern with *). Unreadable lookups are reported in the policy insight and Collection health; the other sources still apply.
Test reference
Runs as a search: reads the lookups now and compares them with the privileged accounts of the tiers. Saves nothing.
Report stale references
On
Also report a declared account that holds no privilege, or no longer exists on any tier (low severity).
Include accounts operated by Splunk
Off
Accounts whose collected email address is support@splunk.com (the accounts Splunk operates on a Splunk Cloud stack) are left out unless this is on.
EXCEPTIONS
Excluded accounts
Close
Preview policy
Save policy settings

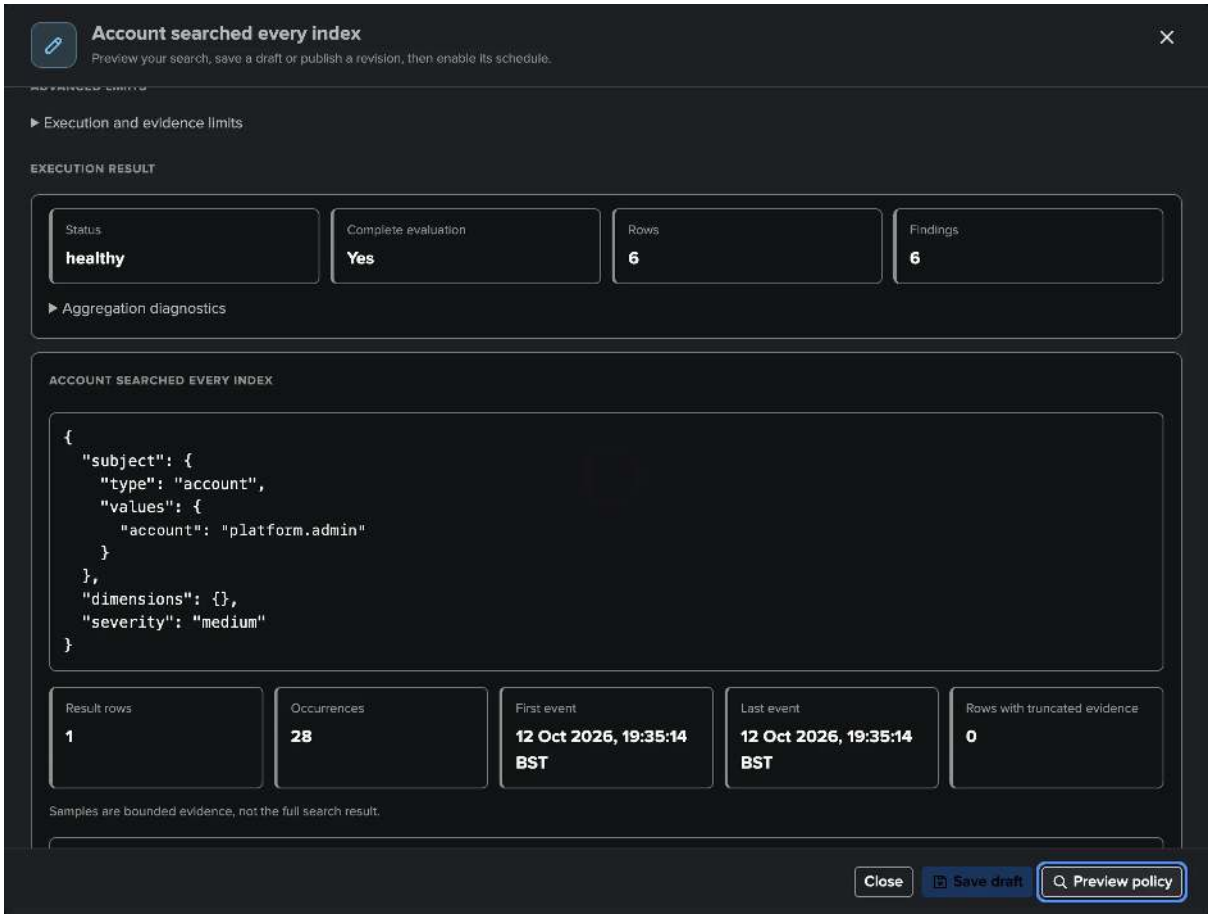


Custom policies

When the catalogue does not cover a rule of your own, write it in SPL: **Policies → Create policy** (TrackMe administrators of the tenant).

1. **Definition** — title, description, default severity and remediation.
2. **Search and schedule** — the search source (an account policy reads through the route account), the app context, the SPL (up to 64 KiB), a relative window ending now (at most 31 days) and a five-field cron.
3. **Map search results to findings** — the subject type, **Account** or **Saved search** (owner, app, name), and which result fields carry the identity, dimensions, evidence, count and time.
4. **Preview policy** — runs the draft as you and shows up to 20 candidate findings. A preview with no results cannot verify the mapping.
5. **Save draft**, **Publish** (an immutable revision), **Enable** (a scheduled search for the published revision). Version history lets you load, compare, restore or delete versions; **Disable**, **Archive** and **Delete permanently** end a policy (its findings, evidence and notes remain).


Account searched every index
Preview your search, save a draft or publish a revision, then enable its schedule.
Remediation
Ask the account owner to name the indexes the search needs, or restrict the role's default and searchable indexes.
SEARCH AND SCHEDULE
Search source
local
Account policies must use the tenant route source. Saved-search policies may use any configured tier.
App context
search
The Splunk app used to resolve macros, lookups and other knowledge objects in your search. For example: search or trackme.
Search SPL
index=_audit action=search info=granted search=* user!=splunk-system-user user!=nobody user!=svc-*
| where like(search, "%index=*%") OR like(search, "%index=\"*\"%")
| stats count as searches latest(_time) as last_seen by user
| rename user as account
Return the fields mapped below. Drag the bottom-right corner to resize the editor.
Close
Save draft
Preview policy


Account searched every index
Preview your search, save a draft or publish a revision, then enable its schedule.
Execution and evidence limits
EXECUTION RESULT
Status
healthy
Complete evaluation
Yes
Rows
6
Findings
6
Aggregation diagnostics
ACCOUNT SEARCHED EVERY INDEX
{
"subject": {
"type": "account",
"values": {
"account": "platform.admin"
}
},
"dimensions": {},
"severity": "medium"
}
Result rows
1
Occurrences
28
First event
12 Oct 2026, 19:35:14 BST
Last event
12 Oct 2026, 19:35:14 BST
Rows with truncated evidence
0
Samples are bounded evidence, not the full search result.
Close
Save draft
Preview policy

The definition also has a **Bank holidays** section (see *Bank holidays*). Custom findings appear with the built-in ones, join the Finding groups and are triaged the same way. In this release, operator and user notifications cover the built-in policies only, and a custom policy writes its exclusions in its SPL. A tenant holds up to 100 custom policies, 25 of them enabled; their execution health is in *Collection health → Trackers*, family *Custom policies*.

Bank holidays

A UAM tenant uses the *bank holiday calendars* like every tenant: the system default calendars, unless it selects its own or none (*Manage Bank Holidays → Tenants & settings*, or *Configure tenant → Bank Holidays*). One effect applies to it, **Suppress new alerts**, shown as **Hold new findings** and on by default: while a holiday of the tenant's calendars is active, the **policies set to hold** open no new finding. SLA, monitoring windows and ML training do not apply to a UAM tenant.

Each policy chooses in its editor, under **During bank holidays**: **Hold new findings** or **Keep opening findings**. The defaults:

Holds new findings	Keeps opening findings
Scheduled search owned by a human account, **Scheduled search owner is not classified**, **Scheduled search exceeds the hourly schedule limit**, **Scheduled search has an all-time dispatch lower bound**, **Scheduled search uses a real-time dispatch range**, **CSV lookups attributed to an account exceed its size allowance**, **Privileged account is dormant**	**Account launched an unusual number of searches**, **Account ran repeated long searches**, **Account ran all-time searches**, **Account's search resource consumption exceeded its allowance**, **Account ran repeated resource-intensive searches**, **Account's CPU consumption is unusual for its daily baseline**, **Account became privileged**, **Role gained a privileged capability**, **Privileged accounts differ from the reference**

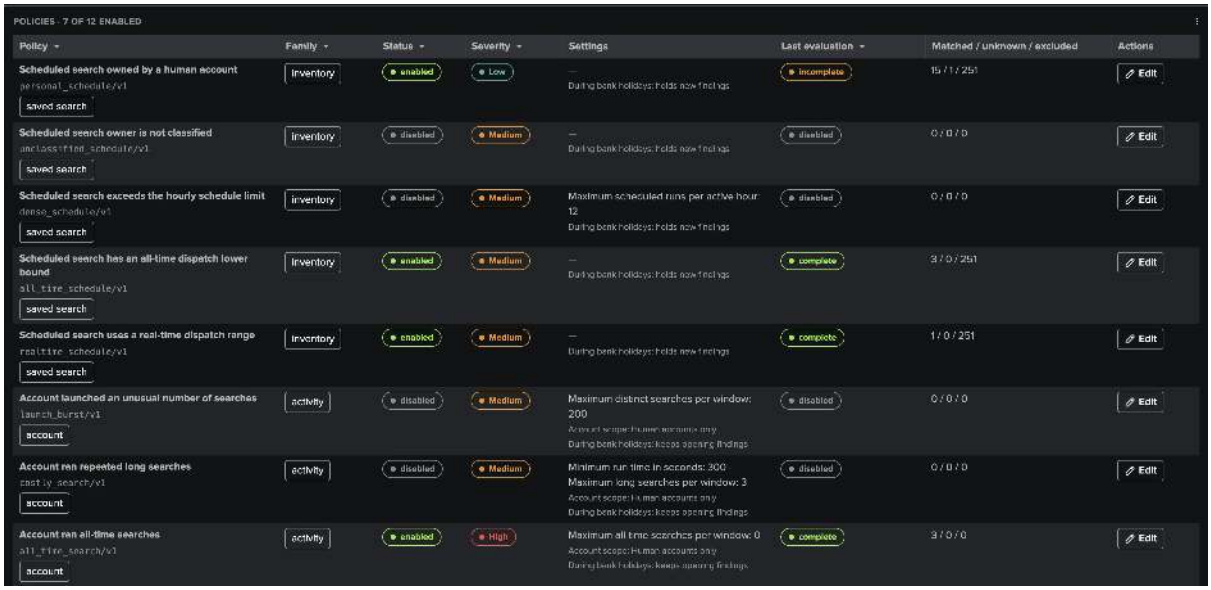
The inventory policies and the dormancy policy are reviews: nobody reviews them on a day off, and they are still there the next working day. A privilege granted on a day off is more telling, not less. The activity and resource policies keep opening findings, because unusual activity on a day off is more telling, not less. A configuration saved before TrackMe 2.4.19 reads each policy's default.

When a policy holds:

- An inventory policy holds while a holiday is active at the time of the run; an activity or resource policy holds when a holiday overlaps its activity window, so a burst on the last evening of a holiday is still a holiday burst.
- A held match writes nothing: no finding, no Finding group, no notification to operators or users, no findings action, and it uses none of the findings capacity. The first policies run after the holiday opens the finding if the condition is still there.
- A finding already observed is evaluated as on any other day: updated, escalated, or no longer observed. A held match does not reopen a resolved or dismissed finding.
- If the calendars cannot be read, nothing is held.

A **custom policy** keeps opening findings unless its definition turns the setting on (*Bank holidays* section of the custom policy editor; it takes effect when published). It holds while a holiday is active at the end of its search window.

Where to see it. Each policy on the Policies tab says *During bank holidays: holds new findings* or *keeps opening findings*. A policy that held matches shows *held N (bank holiday)* under its last evaluation, and its policy insight says *Held N (bank holiday: <names>)* with a **Bank holidays** line for the run. The policies run receipt records the state (holding, off or unavailable), the holiday names, the matches held and the enabled policies set to hold; a custom policy run records what it held the same way.



Upgrading to TrackMe 2.4.21

- **Three trackers for the indexing layer.** What the activity tracker did in one run is now **Audit activity**, **Resource usage** and **Account signals**, each with its own schedule, time budget, lock and health. The two new reports are created automatically within five minutes of the upgrade, with the tenant's owner; they continue where the earlier tracker stopped, and nothing is collected twice.
- **No row limits any more.** The seven record-count settings of *Collection limits* are removed: what a run cannot finish in its time continues at the next run. Stored values are ignored and disappear at the next save; an API client that still sends them keeps working.
- **New time budgets.** Every run budget defaults to its tracker's interval less 30 seconds. A budget still at its earlier default moves to the new one; a value you chose is kept. A scheduled run dispatched late ends 30 seconds before the next scheduled run, so the scheduler never skips one.
- **Correlated executions are no longer indexed.** They are correlated from the audit records when they are read. Your own searches, dashboards, alerts or custom policies on `sourcetype=trackme:uam:execution` return nothing for activity after the upgrade: move them to `sourcetype=trackme:uam:observation`. The UAM events also carry fewer fields (see *The evidence*).
- **Backfill.** A new tenant starts without backfill unless you choose a range, in 5-minute buckets. An existing tenant keeps its bucket size; set *Backfill bucket (s)* to 300 to move it to 5-minute buckets (the coverage already collected is kept).
- **Splunk restart.** Some index-time settings of the UAM events take effect after the splunkd restart of the upgrade, on the instances that parse the UAM log.
- **Privileged users and privileged policies.** The Privileged users tab and its settings appear with the release defaults (`*admin*`, `trackme_admin` excluded). The four privileged policies are off; when you turn on a change policy, its first evaluation records the current state and reports nothing. The users catalog now also collects the last successful login Splunk reports (never an inventory change).

Upgrading from TrackMe 2.4.18

- The first policies runs gather the existing findings into Finding groups; decisions, notes and history are kept.
- The inventory moves to the REST API catalogs: each tier gets its new knowledge-object reports, and its first full listing runs within the hour. The users and roles reports move from a 5-minute to a 15-minute schedule.
- The UAM Findings Advisor is replaced by the UAM AI Agent; an enabled advisor's settings are carried over once.

Where to go next

See also

- *User Activity Monitoring — reference* — the reference: tiers, trackers and schedules, evidence, classification, the full policy catalogue (*The built-in policy catalogue*), every setting (*Settings reference*), REST and troubleshooting (*Troubleshooting and known limits*).
- *Findings, Finding groups and findings actions* — findings, Finding groups, assignment, notifications, findings actions, Webhook accounts and the findings AI Agents, shared with the other dedicated tenant types.
- *Creating a splk-uam tenant* — the tenant creation wizard.
- *Who is doing what on your Splunk? User Activity Monitoring (UAM)* — the white paper.
- *User Activity Monitoring* — the hands-on deck.
- *Configuration Guardian* — the `uam_collection_health` check.
- *The AI Advisors* and *AI Agents automation* — the AI Agents and their automation.
- *Which components need a remote account?* — whether this component needs a remote account, and what it points at.

8.3.15 User Activity Monitoring — reference

This page is the reference companion of *User Activity Monitoring*: the tenant model and the search head tiers, every report and its schedule, the knowledge-object catalogs, the historical backfill, the indexed evidence, identity and classification, privileged users, the catalogue of built-in policies, custom policies, the findings lifecycle and Finding groups, notifications and communications, the UAM AI Agent, Collection health and the Configuration Guardian check, every setting, the roles and licence, the REST endpoints, the upgrades and the troubleshooting table.

For the task-oriented guide (creating a tenant, reading the workspace tab by tab, triaging a finding), read *User Activity Monitoring*. The concepts User Activity Monitoring shares with the other findings tenants — Finding groups, triage, assignment, notifications, findings actions, Webhook accounts and the AI Agents — are explained once in *Findings, Finding groups and findings actions*. For a narrated end-to-end walkthrough, read the *User Activity Monitoring white paper*.

The tenant model

User Activity Monitoring (UAM) is a **dedicated tenant type**, created from the *User Activity Monitoring* card of the tenant creation wizard:

- A tenant is either an entity tenant (DSM, DHM, MHM, FLX, FQM, WLK, VOL) or a UAM tenant, never both. Creation refuses a mix (`uam_tenant_must_be_dedicated`), the *Manage components* menu is disabled on a UAM card, replica trackers are refused and the add / remove component endpoints answer `uam_tenant_is_dedicated`.
- There are **no entities**: no decision maker, no impact score, no green / orange / red state, no stateful alerting, no priority / SLA / logical group. The unit of work is the **finding**, triaged as a **Finding group**.
- A UAM tenant watches **one indexing layer seen through one to ten search head tiers**: the accounts (human, service or system) and their search activity, their resource cost (CPU, peak memory, share of the environment) and their knowledge objects; the saved searches and the other knowledge objects of each tier; and the policies — sixteen built in, plus the custom policies you write — that raise findings about them.
- **Evidence lives in indexes, state in the KV store.** Collection runs in scheduled searches, never in a REST handler.
- The tenant identifier of a UAM tenant is short — lowercase letters, digits and hyphens, up to 20 characters — because it keys the per-tenant collections and reports.
- The Virtual Tenants card shows the open and high / critical findings and the collection problems; a double-click opens a read-only overview. Tenant Home opens the UAM workspace (tabs Findings, Users, Privileged users,

Activity, Scheduled searches, Policies, Audit changes, Collection health, AI Agent and Settings, described in *User Activity Monitoring*).

Note

What UAM claims, and what it does not. An account is the principal **reported by Splunk** in the audit records — not a verified person. Distinct search ids are a launch-count proxy. Collection is bounded by time only and resumable: nothing is dropped to fit a row count, and what a run cannot finish continues at the next run. Coverage still depends on what the sources retain and what the collection accounts can see: gaps and partial coverage are recorded and shown (receipts, *Collection health*), never hidden. There is no machine learning: the daily CPU deviation rule compares against a percentile baseline.

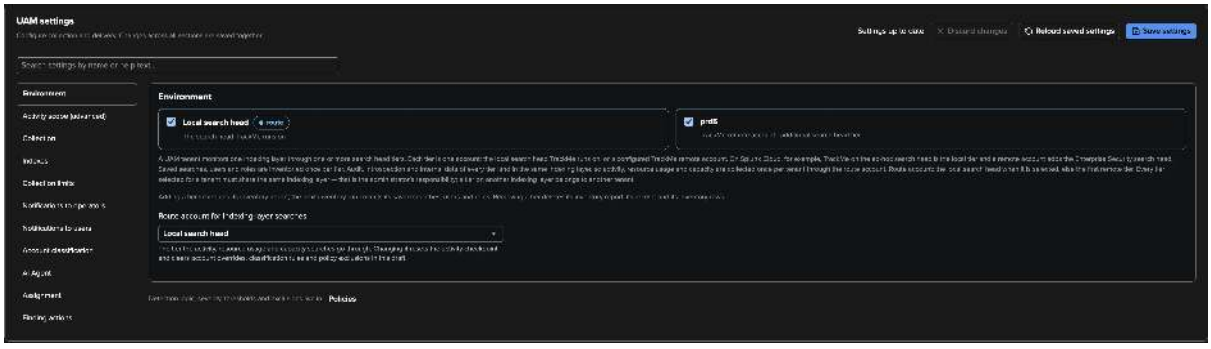
Per-tenant storage

KV collection	Content
`kv_trackme_uam_settings_tenant_<t`	The configuration, the checkpoints, the latest receipt and the lock of every job, the capacity, the detection records, the notification outbox, the user notification state and the private contact store.
`kv_trackme_uam_inventory_tenant_<`	The policy inventory: one row per saved search, user, role and CSV lookup file, per tier, with `present`, first / last seen and the owner's class.
`kv_trackme_uam_knowledge_tenant_<`	The published knowledge-object catalogs and their definitions (administrator access only, no lookup).
`kv_trackme_uam_findings_tenant_<t`	One row per finding, with its investigation state and its Finding group.
`kv_trackme_uam_finding_groups_ter`	One row per Finding group (automatic or manual).
`kv_trackme_uam_custom_policies_te` / `kv_trackme_uam_custom_policy_sta`	Custom policy definitions (drafts, published revisions) and their execution state.
`kv_trackme_uam_actions_tenant_<t:`	Findings actions definitions and automation rules.

The settings, inventory and findings collections each have an administrative lookup `trackme_uam_<kind>_tenant_<tid>`. Two global collections serve every UAM tenant: `kv_trackme_uam_agent_decisions` (the UAM AI Agent decision log, 90-day retention) and `kv_trackme_uam_custom_execution` (custom policy execution slots).

Search head tiers and the route account

One indexing layer, one to ten search head tiers:



Data	Lives on	Collected
Saved searches, users, roles, lookup files and the other knowledge objects	each search head tier	**once per tier**, from the Splunk REST API
`_audit` activity, `_introspection` resource usage, search completion lines	the indexing layer (every tier forwards its internal indexes there)	**once per tenant**, through the route account

- **Search head tiers** (`tiers`) — 1 to 10 unique names: `local` (the search head TrackMe runs on) and / or TrackMe remote accounts (see *Remote Splunk deployments*), one account per tier. Every selected tier must feed the **same indexing layer**; that invariant is the administrator's responsibility. The wizard tests every remote tier and waits until each is reachable.
- **Route account for indexing-layer searches** (`account`) — the tier through which activity, resources, capacity and the audit lookups are read: `local` when selected, else the first tier. It must be one of the tiers.
- A user present on several tiers is **one account** (roles united, tiers listed); a saved search belongs to its tier. Each tier has its own reports, locks and receipts, so a slow tier never starves another and a skipped tier is visible on its own.
- Typical shapes: on Splunk Cloud, TrackMe on the ad hoc search head (`local`) plus a remote account for a premium-app search head; on Splunk Enterprise, any number of remote accounts over the same indexers; for central monitoring, remote accounts only.
- A search head that keeps its internal indexes local (no forwarding of `_audit` / `_introspection`) hides its searches, logins and resource usage from the tenant — the forwarding prerequisite applies to every signal alike.

Activity scope (advanced)

The tiers select **REST connections** for the inventory; they do not restrict which events are read in a shared indexing layer, so a search head that is not a tier still contributes activity, classification evidence and resource cost. **Activity scope** (`search_head_scope`, wizard and *Settings*) restricts the activity to search head hostnames:

- `include` and `exclude` — up to 20 hostname patterns each (1 to 253 characters), matched case-insensitively on the whole hostname; `*` matches any characters, `?` one character, everything else is literal. No SPL, no regular expression.
- Both empty (the default): every origin. An empty include list means every origin except the exclusions. **Exclusions win.**
- Include every member of a cluster. Excluding only the cluster members does not exclude a management node that also runs searches; to isolate one search head tier completely, include its hostnames.
- The scope applies to the live collection and to the backfill: audit records, logins, automation evidence, resource records (indexer samples are mapped back to the search head of their search; an ambiguous mapping is excluded and reported as incomplete coverage), completion CPU and the capacity's search head part. A change starts a new scope.
- The scope is written into the **base searches**, so the indexers return only what can be in scope instead of every search head's events. The audit, login and automation reads filter on `host` (indexed metadata, matched case-insensitively); the resource and capacity reads keep the in-scope search heads' samples and the indexer processes of their searches (`remote_<search head>` origins), so the capacity lists the indexers that serve the scope. The base-search terms always select at least what the exact rule keeps. An exact name or a name ending in `*` is the filter itself. An include holding `?` or an inner `*` is written on its literal prefix (`team-*.example.net` → `team-*`; an include without one, such as `*.example.net`, leaves the includes out of the base search), an exclude of that kind is not written in the base search, and the exact rule is applied after the base search, so nothing in scope is ever skipped. Patterns written as exact names or with a trailing `*` filter the most at the source. A tenant without a scope runs the same searches as before.
- It is a monitoring rule, not an access-control boundary: nothing is deleted from the source indexes.

Reports and schedules

Provisioning creates every report of the tenant and schedules them at once. Each report runs `| trackmeuam-tracker tenant_id=<tid> job=<job>` (or its own command) as the **tenant owner**, with its own **lock**, its **receipt** (the outcome of the latest run, kept in KV and indexed as `trackme:uam:receipt`) and its entry in the tenant's operational status.

Report	Schedule	Does
`trackme_uam_activity_tracker_te` (**Audit activity**)	`*/5 * * * *`	The audit trail from a durable ingestion checkpoint through the route account, the delayed recounts and re-reads of its ranges; CSV writer hints.
`trackme_uam_resources_tracker_t` (**Resource usage**)	`2-59/5 * * * *`	The CPU and memory of every search, window by window from its own checkpoint, their re-reads, and the hourly search capacity. Unscheduled while resource collection is off.
`trackme_uam_signals_tracker_ten` (**Account signals**)	`9-59/15 * * * *`	The Splunk Web logins and the automated searches of the audit trail, up to where the audit activity has read, for the human and service account detection.
`trackme_uam_data_usage_tracker_` (**Data usage**)	`4-59/15 * * * *`	The indexes each audited search names, from its own checkpoint in the audit trail, resolved through the published catalogs of the search's tier; the indexers' confirmations on distributed deployments. See *Data usage*.
`trackme_uam_policies_tracker_te`	`3-59/15 * * * *`	Evaluates the built-in policies, writes findings and Finding groups, expires temporary dismissals, delivers operator and user notifications, runs the automation rules.
`trackme_uam_inventory_tracker_t`	`*/15 * * * *`	Summarises the published catalogs of the tier for the policies run (reads no source).
`trackme_uam_listing_tracker_ten`	hourly wake (`37 * * * *`)	Lists every saved search of the tier when due: every *Catalog collection every* hours (24) inside the off-peak window (01:00 for 5 hours, search head local time). A tier's first listing runs at the next hourly wake, whatever the window. A listing longer than its budget continues at the next wake its window allows.
`trackme_uam_kos_users_tracker_…` / `…kos_roles…`	15-minute wake	Users (with the email address for user notifications) and roles, read every *Users and roles every* minutes (60).
`trackme_uam_kos_lookup_files_tr`	15-minute wake	CSV lookup file metadata (size, owner, modification time; never the contents), read every 15 minutes.
`trackme_uam_kos_<family>_tracke` × 10	hourly wake, each at its own minute	The ten other knowledge-object families, on the catalog cadence, in the off-peak window.
`trackme_uam_backfill_tracker_te`	`17 * * * *`	The history before the live start, bucket by bucket; retires itself when done.
`trackme_uam_custom_<id>_tenant_`	the policy's own cron	One per enabled, published custom policy.
`trackme_uam_agent_tracker_tenan`	hourly, at a minute derived from the tenant	The UAM AI Agent's scheduled triage; created unscheduled, scheduled only when AI is available and the tenant enables it.

The per-tier catalog reports are named `trackme_uam_kos_<dataset>_tracker_tenant_<tid>_tier_<tier>`. The tracker schedules are fixed; the cadences inside them (*Catalog collection every*, *Users and roles every*, the listing window) are settings.

Run budgets

Time is the only bound of a run: what a run cannot read in its time continues at the next run through its checkpoint, never cut to fit a row count. Every budget defaults to its tracker's interval less 30 seconds, the margin covering the run's own writes before the next scheduled run:

Tracker	Setting (key)	Default
Audit activity	*Audit activity max seconds per run* (`activity_max_seconds`)	270 s (30–900)
Resource usage	*Resource usage max seconds per run* (`resources_max_seconds`)	270 s (30–900)
Account signals	*Account signals max seconds per run* (`signals_max_seconds`)	870 s (30–900)
Data usage	*Run budget* in *Settings → Data usage* (`data_usage_max_seconds`)	870 s (30–14,000)
Saved-search listing and catalogs	*Full listing budget* (`inventory_listing_max_seconds`)	3,570 s (300–14,000)
Historical backfill	*Backfill max seconds per run* (`backfill_max_seconds`)	3,570 s (600–3,570)

No skipped runs. The scheduler may dispatch a run up to five minutes late, and it skips a run whose previous run is still going. A scheduled run's budget is therefore capped to end 30 seconds before the tracker's next scheduled run; a run dispatched so late that less than 15 seconds would remain records a skip (`dispatched_after_schedule_cutoff`) instead of starting. The receipt records the budget applied (`run_budget`: the seconds, whether they were capped and by how much the dispatch was late), and the tracker's Duration in Collection health explains a cap.

Each indexing-layer tracker keeps a share of its own budget for its re-reads, recounts or capacity refresh only when some are due; nothing is shared between trackers, so a slow resource read never takes time from the audit trail, and the other way round.

Collect now, Run now and manual collection

- **Collect now** (workspace status strip) runs every tier's inventory summary, then the audit activity, the resource usage and the account signals, then the policies, and every tier's full saved-search listing. The other catalog families record a request their reports pick up at their next wake.
- **Run now / List now** (*Collection health*) dispatches one tracker.
- With **Scheduled trackers** off, nothing runs on a schedule: collection only happens on request. Scheduled collection requires a tenant owner (`scheduled_collection_requires_tenant_owner`).
- A scheduled run never blocks the workspace. A *Collect now* refused because a run holds the lock (`uam_job_running_or_recovery_required`) is explained, not reported as an error.

Locks and recovery

One run per tracker at a time. Locks carry the local server GUID and the search id and are **never stolen by age alone**:

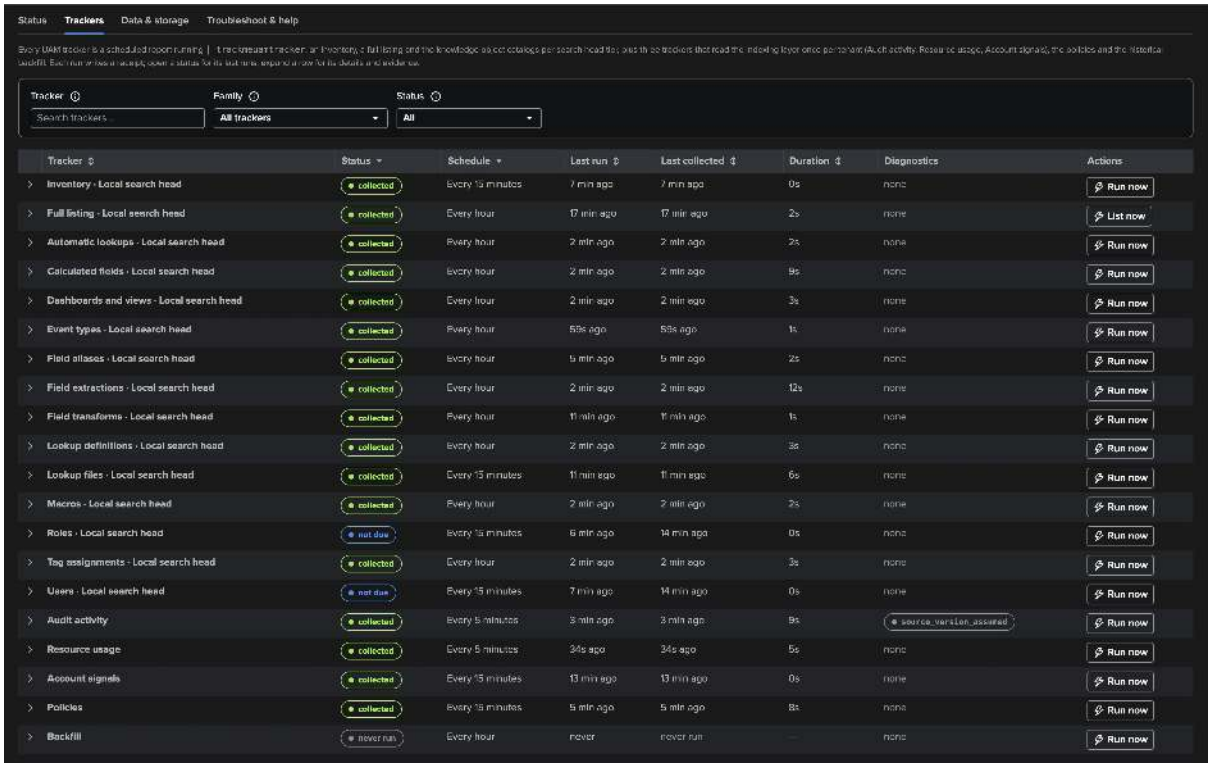
- A lock at least five minutes old whose search job has provably stopped **on the same search head** is recovered automatically at the next acquisition.
- **Recover stopped run** (*Collection health*, `POST recover`) does the same on demand.
- Locks held by another cluster member, locks without a search id and configuration locks without job ownership are never recovered automatically: verify the originating process on its search head.
- The tenant lifecycle (delete, enable, disable, RBAC) has its own guard bound to the process that holds it, so a request whose process died never blocks the tenant for good.

Knowledge-object catalogs

The inventory is read from the **Splunk REST API only** — no `_internal` or `_audit` search is used to discover objects or changes. Each dataset of each tier is its own scheduled traversal:

Family (account view tab)	Datasets	Cadence
Saved searches & alerts	Every saved search (reports and alerts), with owner, app, sharing, cron, dispatch bounds, enablement and the definition.	*Catalog collection every* (24 h), off-peak
Users and roles	Users (with their email address, kept in a private store) and roles.	*Users and roles every* (60 min)
Lookups	CSV lookup files (metadata only), lookup definitions, automatic lookups.	Files every 15 min; the others on the catalog cadence
Dashboards	Dashboards and views.	Catalog cadence
Macros	Macros.	Catalog cadence
Event types	Event types.	Catalog cadence
Fields	Field extractions, field transforms, field aliases, calculated fields.	Catalog cadence
Tags	Tag assignments.	Catalog cadence

- Every traversal is paged (*Inventory page size*) and bounded by time (*Full listing budget*), never by a record count, and **publishes a complete generation or nothing**. A traversal that does not finish within its budget continues at the next run of its tracker, from where it stopped, in a staging generation, and is published once complete; until then the previous catalog stays in use and keeps the policies complete. Collection health shows such a run as collected with *Listing continues next run*; *Listing made no progress*, *Listing restarted (access refused)* and *No page read in time* are the warnings, and a listing carried over for more than seven days starts over. A failed traversal keeps the last good catalog. Only one traversal runs at a time per tenant and tier; a busy report waits for its next wake.
- **Definitions** — saved search SPL, macro bodies, event type searches, dashboard source, field expressions, lookup configuration, tag assignments — are kept up to 256 KiB per object in the administrator-only knowledge collection, from an explicit attribute list (never credentials, never CSV contents). They are not executed, indexed or copied to the policy inventory. Older generations (other than the current and the previous one) are deleted after 30 minutes.
- **CSV lookup sizes before Splunk 10.4.** The lookup file listing gives no size before Splunk 10.4 (Splunk 10.4 and later report it, with no extra access); UAM then reads the size and modification time of each CSV lookup file listed without one through `admin/file-explorer`, one lookups directory at a time (the path only locates the file and is never stored). The tier's collection account must be allowed to read that endpoint, which by default only `admin` and `splunk-system-role` can: a least-privilege account cannot retrieve CSV lookup sizes before Splunk 10.4. A refusal is reported as `lookup_size_access_denied` (*Lookup sizes refused*), never as a partial listing.
- Each saved search, user, role and lookup file is diffed against its previous row: a new row is `added`, changed fields `changed` (with `changed_fields`), a classification change alone `reclassified`, a row seen again after removal `reappeared`; a row absent from a **complete** traversal is marked `present=false` — never deleted. A partial traversal marks nothing removed. Each change is indexed as `trackme:uam:inventory`.
- The catalogs reflect what the tier's account can see: coverage of private objects depends on its permissions. A traversal is not an atomic snapshot.
- Where they appear: the account view's owner catalogs (*Saved searches & alerts*, *Dashboards*, *Macros*, *Event types*, *Fields*, *Lookups*, *Tags*), **Search knowledge objects** on the Users tab (across every owner), the saved search view, and *Collection health → Trackers* (family *Knowledge object catalogs*). Tenant administrators only.



Activity

The **Audit activity** tracker reads the `_audit` trail from a **durable ingestion-time checkpoint**. Time is the only bound of a run: nothing is cut to fit a row count, and what a run cannot read in its time continues at the next run.

1. It loads the checkpoint, or initialises it at *now* − *window* on the first run or when the environment changed (`checkpoint_reset` on the receipt).
2. It reads ranges of up to *Activity window* (300 s) from the cursor, less the part of the *Activity overlap* still owed, until it catches up (*now* − *10 s*) or its budget (*Audit activity max seconds per run*, 270 s) is spent. Both audit formats are evidence: the legacy `Audit:[…]` key=value line and the structured JSON record. The rows are normalised and correlated by search id.
3. **A busy range is split, never cut.** A range holding more audit lines than one read returns, or that Splunk finalized, truncated or reported with dropped results, is halved, down to one second, and a one-second range is read whole, streamed in ingestion order.
4. **A range cut by the run's budget resumes** at the next run after the last line it wrote (`resume_next_run`); near the live edge it is read again from its first line instead, so a line indexed a moment late is not missed.
5. **A failing range is held and retried, never skipped** (`window_held`): it is retried at once within the run, then at the next runs, and halved after three failures to isolate the failing span. A range held for 15 minutes is a Collection health problem.
6. **Indexers that do not answer.** A range read while some search peers did not answer keeps what the others returned and is listed to be **read again** once every peer answers (`window_reread_pending`), with a back-off of 5 minutes up to an hour. The re-read writes only the lines missing the first time.
7. **Late lines are recounted.** About ten minutes after a range is closed, it is counted again with the same search; more lines than were written lists the range for a re-read (`window_recount_mismatch`).
8. The overlap and a ledger of what was written make sure **nothing is written twice**, resumed, retried and re-read ranges included. The checkpoint is persisted after **every** range.
9. Splunk's search messages are recorded, never interpreted: only a clear failure (a failed or cancelled job, a broken read) discards a search. Evidence-quality diagnostics (`normalization_gap`, `locator_conflict`, `invalid_event_envelope`, `unsupported_source_contract`) are recorded but never block.

The same run records the **CSV lookup writes** it saw (probable writer hints for the CSV policy). The resource usage, the capacity and the login and automation scans are trackers of their own (below).

Account signals

The **Account signals** tracker (every 15 minutes, *Account signals max seconds per run*, 870 s) scans the audit trail for **Splunk Web logins** and **automation** evidence (searches run by the scheduler, the REST API, acceleration), the signals behind the human and service detection (see *Data usage*). Each scan has its own cursor and trails the audit activity's cursor without ever holding it; the login scan takes up to half of the budget, the automation scan the rest. A scan range that cannot be read is listed to be read again, like an audit range. Before the first audit activity run, the tracker waits (`account_signals_wait_for_activity`).

Source contract. The audit source contract is the **format**, never the Splunk version: a line that parses as the legacy key=value grammar or the structured JSON is evidence. The qualified families are Splunk Enterprise 9.4, 10.4 and 10.6 and Splunk Cloud 10.5; any other version is read all the same and only adds the soft diagnostic `source_version_unqualified`. With *Source version assumption* on (the default), emitters whose version could not be verified are parsed under the connected head's version and labelled `assumed`.

Resource usage and CPU accounting

With *Collect search resource usage* on (the default), the **Resource usage** tracker (every 5 minutes, two minutes after the audit activity, *Resource usage max seconds per run*, 270 s) reads the resource records window by window from its own checkpoint and writes one `trackme:uam:resource` record per search and account: CPU seconds, the part spent on the indexers, peak memory, elapsed time, node count, and Splunk's own search type, provenance, mode and label. Its reads are paged and resumable like the audit's: every search of a window is kept, a window read while peers did not answer is collected again and replaces the earlier collection, and the policies never count a replaced collection. With resource collection off, the report is unscheduled and not assessed.

CPU follows Splunk's own accounting where it exists, decided per window and per tier (search head, indexers) from the data:

- Splunk Cloud 10.5 accounts CPU at search completion (the audit `cpu_time` on the search head, the indexers' completion lines). A tier with a completion figure counts its CPU once, at completion.
- A tier without one keeps the CPU integrated from the `_introspection` per-process samples (Splunk Enterprise 10.4).
- Every record carries `cpu_source` (`completion`, `head_completion`, `peer_completion` or `introspection`). Real-time searches always use sampled CPU.
- Peak memory reads **not measured** for a search no introspection sample caught (short searches in pooled processes on Splunk Cloud).

The **capacity** — the cores and memory of the *search hosts*, the nodes that ran a meaningful share of the search processes (a utility node running its own few searches is listed but kept out of the basis) — is refreshed at most hourly by the Resource usage tracker and drives the automatic budgets of the resource policies. With an activity scope, it counts the search heads in scope and the indexers that serve them. A failed refresh keeps the previous record.

Policies run

Every 15 minutes the policies job:

1. reads the present inventory rows of every tier under the inventory lock (a long inventory publication makes it wait up to 90 seconds, then fail with `inventory_publication_in_progress` and retry at its next schedule) and evaluates the **inventory** policies with the completeness of each tier's catalogs;
2. aggregates the indexed audit and resource records of the *Activity policy window* (15 minutes) and evaluates the **activity** and **resource** policies per account. The window ends where the collected evidence ends — the audit activity cursor, and the resource cursor when a resource policy is enabled — never at the current time, and never starts before the collected history. Consecutive windows always touch (with less than a whole window of new evidence, the window reaches back into what was already evaluated), so no minute is skipped; a run evaluates a further window only for a whole window of new evidence. Every aggregation is read whole; one Splunk reports as partial (missing peers, truncated output) is refused, and its window is evaluated again;

3. reconciles the findings and the Finding groups under the findings lock (see *Findings and Finding groups*), expires temporary dismissals, publishes the counts the card reads and writes one `trackme:uam:finding` event per transition;
4. delivers the operator notifications and the user notifications, and runs the automation rules (fail-open: a delivery failure never fails the run).

The receipt stores one **evaluation** per policy (enabled, state, evaluated / matched / excluded / unknown / not applicable counts, readiness, thresholds source) and its bounded **insight** (see *Policy insight and preview*). Completeness is per family: inventory findings are marked not observed only when every tier's catalogs are complete, audit findings when the activity window was complete, resource findings when the resource coverage is complete on top. A rule whose aggregation failed is `incomplete` and neither creates nor clears anything. There is no findings limit: every match opens a finding. A very large live investigation queue (more than 50,000 findings one run can change) raises the Collection health warning `findings_working_set_large`; nothing is refused.

Bank holidays. A UAM tenant takes one effect of the *bank holiday calendars*, **Suppress new alerts** (*Hold new findings*). While a holiday of the tenant's calendars applies and the effect is on, the policies set to hold (`policies.<policy id>.hold_on_bank_holidays`, defaults in *The built-in policy catalogue*) open no new finding:

- The run reads the holidays once, only when an enabled policy holds: the holidays that overlap the evaluated activity window, then the tenant's calendars and effect. The outcome is `holding`, `off` (a holiday applies but the tenant does not hold) or `unavailable` (the calendars could not be read: nothing is held, a warning is logged).
- An **inventory** policy holds while a holiday is active at the time of the run (it judges the configuration as it is now). An **activity** or **resource** policy holds when a holiday overlaps its activity window: a burst on the last evening of a holiday is a holiday burst, even when the run comes after midnight.
- A match of a held policy whose finding does not exist, or is not observed (resolved, dismissed, or not seen by the previous run), is skipped before the capacity check: no finding, no Finding group, no transition, no `trackme:uam:finding` event, no operator notification, no findings action, and no findings capacity used. The direct user notifications leave its state untouched: a held match is neither a detection nor a step towards a recovery.
- A finding already observed is evaluated as on any other day (updated, escalated, no longer observed); expiries and housekeeping are unchanged. A held match never makes the run incomplete, and the first run after the holiday opens what was held.

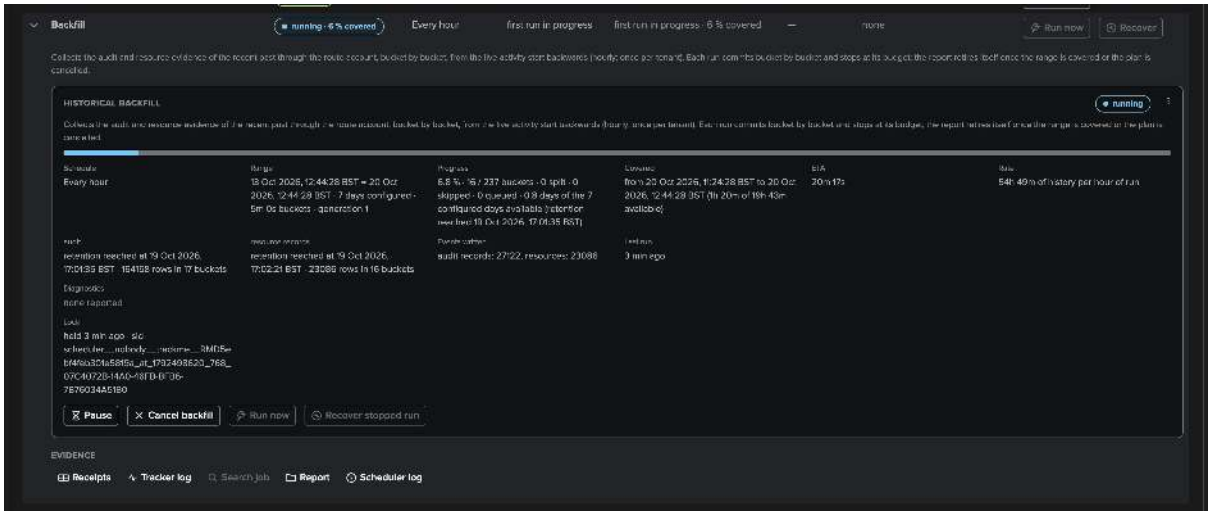
When the outcome is not "no holiday", the receipt carries `bank_holidays`: `state`, `names` (up to five holidays), `held` (the matches held) and `policies` (the enabled policies set to hold). Each evaluation that held carries its own `held` count, and the findings counters carry `held` while the run holds. The `policy_evaluation` endpoint returns the run's `bank_holidays`.

Historical backfill

The backfill report collects the audit and resource evidence of the configured range before the live activity start (*Backfill days*: none — the wizard's preset —, 7, 14, 30, 60 or 90 days in the wizard, 0–90 in *Settings*), through the route account. Evidence in an index cannot be taken back, so the backfill **never collects a span twice**, and nothing is skipped:

- The **plan** ends exactly where the live activity started and **waits** until the first audit activity and resource usage runs have set that point (`backfill_waits_for_first_activity_run`, `backfill_waits_for_first_resources_run`, normal receipts).
- **Fixed buckets** (*Backfill bucket*, 5 minutes by default) are walked **backwards**, newest history first, one audit read and one resource read each, with the live collection's reads: a busy bucket is split, down to a few seconds read whole.
- **A failing bucket is deferred, never skipped.** Before any bucket succeeded in a run, a failure stops the run (a source that is down costs one attempt per hour, and the run is reported failed). Once one succeeded, the deferred buckets get their next attempt and the run goes on past a failure; three failures in a row stop it (`backfill_source_failing`). A bucket that keeps failing is halved to isolate the failing span.

- **A bucket cut by the end of its run resumes** at the next run after the last line it wrote; a run stops at *Backfill max seconds per run* (3,570 s). A bucket read while some peers did not answer is listed to be read again, like a live range.
- **Retention edges** come from the sources: the plan stops at the oldest `_audit` and `_introspection` data (`retention_reached`). The introspection retention is usually shorter (about two weeks), so the resource history may end earlier than the audit history.
- **Re-plans carry the coverage forward**: a changed range or bucket size, or *Plan again*, starts a new plan where the previous one stopped; a range already covered completes at once. A complete or cancelled plan un-schedules the report; `backfill_days = 0` disables it.
- The backfill also scans each bucket for Splunk Web logins and automation evidence, so the people and services already active when the tenant was created are classified without waiting. It does not record CSV writer hints, and the inventory is never backfilled.
- Controls in *Collection health* → *Trackers*, on the expanded **Backfill** row: **Pause** (after the current bucket, progress kept), **Resume**, **Cancel** (retires the report; collected buckets stay indexed), **Plan again**, and **Run now**. The receipt carries the buckets done / split / retried / deferred / resumed, the events written, the progress (percent, estimated end from the measured rate) and the plan status. An existing tenant keeps its bucket size: set *Backfill bucket (s)* to 300 to move it to 5-minute buckets; the plan is rebuilt and the coverage already collected carried forward.



Receipts and diagnostics

A receipt is **value-free**: counts, statuses, diagnostic codes and window bounds — never an account name, a search text or a credential. *Collection health* renders the latest receipt of every job and the last seven days of runs from the indexed receipts, each diagnostic explained. A completed run with zero observations is a successful run: absence of activity is not a failure. A completed run is not a lossless-ingestion guarantee — the receipt says what was covered.

The evidence

Evidence events are written to the dedicated `trackme_uam_events.log` under `$SPLUNK_HOME/var/log/splunk` (UAM writes one line per search, so it never shares the state-events file of the other components) and routed by TrackMe's ingest evaluation to the target index of their family.

Sourcetype	Family	One record per	Main content
`trackme:uam:observati`	evidence	Search and window	The account and app, the search id, type and status, the run metrics, the time bounds (the `N/A` all-time marker included) and the grant.
`trackme:uam:resource`	evidence	Search and window	The search's resource usage: CPU seconds (search head and indexers) with their source, peak memory, elapsed time, nodes and samples.
`trackme:uam:inventory`	evidence	Inventory change	The object's fields, the change (added, changed, removed, reappeared, reclassified) and the fields that changed.
`trackme:uam:data_usage`	evidence	Index reference and batch	One fact per joint key (tier, account, app, origin, path, index, sourcetype, read or write, named or unresolved) and audit batch, with its counters and up to five sample search ids. See *Data usage*.
`trackme:uam:search_te`	search text	Reported search	The SPL, written only while *Store SPL search* is on.
`trackme:uam:finding`	findings	Finding transition	The finding with its evidence and investigation state.
`trackme:uam:receipt`	receipts	Tracker run	The outcome of the run.

- **Observation fields.** `account` and `account_source`, `app`, `sid`, `search_type`, `category`, `outcome`, `search_status`, `metrics` (`total_run_time`, `event_count`, `result_count`, `scan_count`, `exec_time`), `time_bounds` (`search_et` / `search_lt` as text) and `granted_time`; use `_time` for the event time. The same sourcetype also carries one `rest_access` summary per account and batch (REST job access counted per kind: `job_poll`, `job_lookup`, `job_dispatch`, `stream`, `remote`, `other`) and the `denied REST:` lines (security evidence).
- **Correlated executions** are not indexed: the Activity tab, the account views, the REST API (`activity` with `kind=execution`) and the AI Agent correlate each search from its audit records when they read it (start and completion observed, ambiguity, the accounts, apps and search types reported). The `trackme:uam:execution` events of earlier versions stay searchable until they age out, and are read too.
- **Resource fields.** `account`, `sid`, `app`, `search_type`, `provenance`, `mode`, `label`, `cpu_seconds`, `peer_cpu_seconds`, `cpu_source`, `peak_mem_mb`, `elapsed`, `nodes`, `samples`, `roles`, `window`.
- **Inventory fields.** Additions, reappearances and removals carry the whole object; a `changed` or `reclassified` event carries only the changed fields, new (`object.<field>`) and earlier (`previous.<field>`), plus the owner.
- **Data usage facts** carry `key` (the JSON list of their dimensions) and the counters `searches`, `index_searches`, `index_completed`, `late_searches` and `matched_events`, written as the indexed fields `uam_data_key`, `uam_data_refs`, `uam_data_n`, `uam_data_ndone`, `uam_data_late` and `uam_data_events`, plus `uam_data_index`, `uam_data_sourcetype` and `uam_data_label` (the saved search that ran). A map over 30 days is one `tstats` over these fields, never a read of raw events.
- **Finding transitions** carry the evidence only when the policies run changes it (`new`, `updated`, `reobserved`); every decision field stays.
- Every event carries `tenant_id` (an **indexed** field) and `environment_id`; the audit and resource records also carry the indexed dimensions `uam_kind`, `uam_environment`, `uam_account`, `uam_sid`, `uam_app` and `uam_status`, so the per-account and timeline reads run through `tstats`. These index-time settings take effect after the splunkd restart of the app installation, on the instances that parse the UAM log; until then, and for older evidence, the readers fall back to slower raw searches.
- **No field is written that nothing reads**, while the fields with investigative value stay.
- Default indexes: the tenant's **summary index** for evidence, search text and receipts, its **notable index** for findings. Each family can be pointed at another index in *Settings → Indexes*.

- **One compact record per search and window**: the grant folds into the completion (`granted_time`), REST job access is summarised per account rather than written line by line, null keys are dropped.
- Searches, dashboards, alerts or custom policies of your own that read `sourcetype=trackme:uam:execution` return nothing for activity collected since 2.4.21: rewrite them on `sourcetype=trackme:uam:observation` (one record per search and window, with `sid`, `outcome`, `granted_time`, `category`, `search_status` and the account fields).

Store SPL search and Fetch SPL

- **Store SPL search** (`store_search_text`) is **off by default**. When on, the SPL of every reported search is indexed as its own `trackme:uam:search_text` record in the search text index. Audit search text can carry literals and secrets: point the search text index at an index restricted to the people allowed to read it **before** you switch this on.
- **Fetch SPL** (account view, Activity tab, saved search view) shows the SPL of one search. It reads the stored `trackme:uam:search_text` record (or older evidence that carried the SPL inline) as the calling user; otherwise it reads the environment's `_audit` record live through the route account, **as the calling user**, and stores nothing. When the text cannot be shown, a reason says why: `not_in_audit`, `account_access_denied` (your roles are not granted the route's remote account), `access_denied` (the environment refused you), `account_resolution_failed`, `authentication_failed`, `transport_error`, `search_incomplete`, `deadline_exhausted` or `audit_lookup_failed`.

The saved search view

Opened from a saved search finding, the *Scheduled searches* tab or an account's owned searches, the **saved search view** shows the inventory definition of the saved search on its tier, its runs (CPU, run time, peak memory, a CPU chart and the 50 latest runs), its findings and its change history. **Fetch definition** reads the current definition live from the saved search's own tier, as the calling user (reasons such as `not_found`, `account_access_denied`, `access_denied` or `tier_not_configured` explain a refusal); nothing is stored. Runs are matched on name, app and owner (a `nobody`-owned saved search runs as `splunk-system-user`); a run of the same saved search by another account is left out.

Licence impact

The evidence is indexed data and counts against the Splunk licence like any TrackMe summary event. Its volume follows the search activity of the monitored environment — one compact record per search and window, one resource record per search and window, the inventory changes, one receipt per run. No row count caps it: UAM collects everything in scope. To reduce it, restrict the **Activity scope** to the search heads you need to monitor, switch off *Collect search resource usage* (the resource policies then stay learning), keep *Store SPL search* off, and size *Backfill days* with the retention of your sources in mind.

Data usage

The **Data usage** tab (*Data usage*) answers which indexes are read, by whom and through which path. This section describes the collection behind it and the rules its views follow.

The collection. The `trackme_uam_data_usage_tracker_tenant_<tid>` report (`4-59/15 * * * *`) has its own checkpoint in the audit trail, its own budget (870 s by default), lock, receipt and Collection health entry. It reads the audit trail through the same bounded, resumable reader as Audit activity, and writes no activity records. What a run cannot finish in its time continues at the next run; a batch whose extraction or writes failed is held and read again, never acknowledged.

- **Resolution.** Each search is parsed for the indexes and sourcetypes it names: literal `index=` and `sourcetype=` terms, `IN` lists, macros (with their arguments), event types, data models (their root constraints) and saved searches (`savedsearch` / `loadjob`), resolved through the **published catalogs of the search's own tier**, with Splunk's private / shared precedence. Wildcards stay patterns. A negated term never makes a read. A late field filter (after the first command) is a reference, not a source read. A `collect` destination is a write reference. So is the target index of a saved search's summary indexing or Log event alert action, read from the published saved search catalog (inherited `alert_actions.conf` defaults included; an empty target means

`summary` or `main`) and recorded on each run on the path `summary_index` or `log_event`. Custom alert actions are never inferred.

- **Unresolved scope.** What cannot be resolved is counted, not dropped: the search is recorded as *scope unresolved* with its reasons (a macro, event type, saved search or data model the catalog does not hold, a dynamic term, a wildcard role default, a truncated SPL, a resolution limit reached). For a search naming no index, the indexes its account's roles give it (the role search filter's indexes, else the default indexes the roles allow) are shown as context, never as reads.
- **Counted once.** A search is counted at its grant in the legacy `Audit:[…]` key=value format, or at its completion in the structured format. A canceled or failed completion counts; a search Splunk refused at its quota (`action=quota info=denied`) is not a read.
- **The indexers' confirmations.** On a distributed deployment, the search peers record in `remote_searches.log` the search they received after Splunk expanded its macros and saved searches. The Data usage run reads those lines from its own cursor and keeps them as a separate facet (*Indexers received*), never added to the search head's counts. A standalone search head has none.
- **Declarations.** Each catalog publication projects which knowledge objects name each index (saved searches, macros, event types, dashboards, data models). A declaration is not a read.
- **History.** When the tenant has a backfill, the Data usage history starts at the backfill depth, upgraded tenants included, but never before the oldest event of the audit trail: what `_audit` no longer holds cannot be read. Without a backfill, a new tenant's history starts where its Audit activity started, and an upgraded tenant's at its first Data usage run.

The rules of the views. Every case, with diagrams and worked examples, is in the *Data usage reference: how each claim is processed*.

- **One indexing layer.** Usage classes and review candidates are decided across every tier of the tenant: an index read on any tier is never a candidate. The indexes TrackMe writes to — the global index settings, every tenant's overrides and this tenant's own families — are never candidates.
- **No absence from too little evidence.** Until the collection covers the selected period, or seven days of it, no index is called unread and no knowledge object *not run*: the views say *so far* and give the date from which the claim can be made. A period with no Data usage run, an incomplete fact read or facts without the rollup key withhold every absence verdict. The first runs of a new tenant, before its first catalog listing, count as progress.
- **Excluded apps** (`trackme` and `Splunk_SA_CIM` by default) are shown, never counted: an index only they read is *Excluded apps only*.
- **Wildcards reach by index name.** `index=cribl_*` is matched against the indexes known for the tier with Splunk's index pattern rule (`*` never matches an internal index), then narrowed to the indexes its reader's roles allow; it is never narrowed by sourcetype. A wildcard reference never makes an index *read by name*.
- **Role search restrictions.** The roles of each reader are read from the published users and roles catalogs when the view is read: allowed indexes, minus the disallowed ones, narrowed by the index terms of the role search filters. A named index the roles do not allow is counted as *denied by roles*, never as a read. The scope is an upper bound, and the current roles apply to the whole period.
- **Not due.** A scheduled saved search that did not run, whose schedule leaves a longer gap than the period covers, is *Not due in this period*, never *Not run*.
- **Least accessed** ranks the indexes read by name with a daily volume above zero by reads per GB/day, and marks the lowest quarter. It needs at least four such indexes.

Volume and cost. A tier is linked to one VOL tenant (index volume) and up to 200 DSM tenants (incoming data), matched automatically whatever account they collect through unless set in *Settings → Data usage*. The cost per day comes from the VOL tenant's costs (its Global license usage configuration), when they are enabled; otherwise no cost is shown, never a guessed one. A link grants nothing: every read repeats the caller's visibility of the linked tenants. An index without a linked volume is counted, not weighed. Each index is weighed by its 30-day average of complete days; while VOL has no average for an index yet (its first day without a backfill), by its last 24 hours, and the views say so.

Reads without a time budget. The map, an index's detail, the knowledge objects and the access timelines run as the **caller's own search jobs**: the start request dispatches them and returns at once, the UI polls them until every

one is done (naming what is still being read), and cancels them when the selection changes or the view closes. Only the 600-second safety timeout applies; a read that reaches it is reported incomplete, never shown as empty data. The start answer fixes the period, and every poll reads the same one. The linked VOL and DSM reads page through every record, each request bounded by `splunkd_timeout`.

Limits. At most 50,000 fact keys and 10,000 catalog indexes per tier are read per map; a larger answer is flagged truncated and withholds every absence verdict. An access timeline has at most 120 clock-aligned buckets and 10,000 aggregated rows, ten lanes per page. Every limit reached is said in the view.

Identity and classification

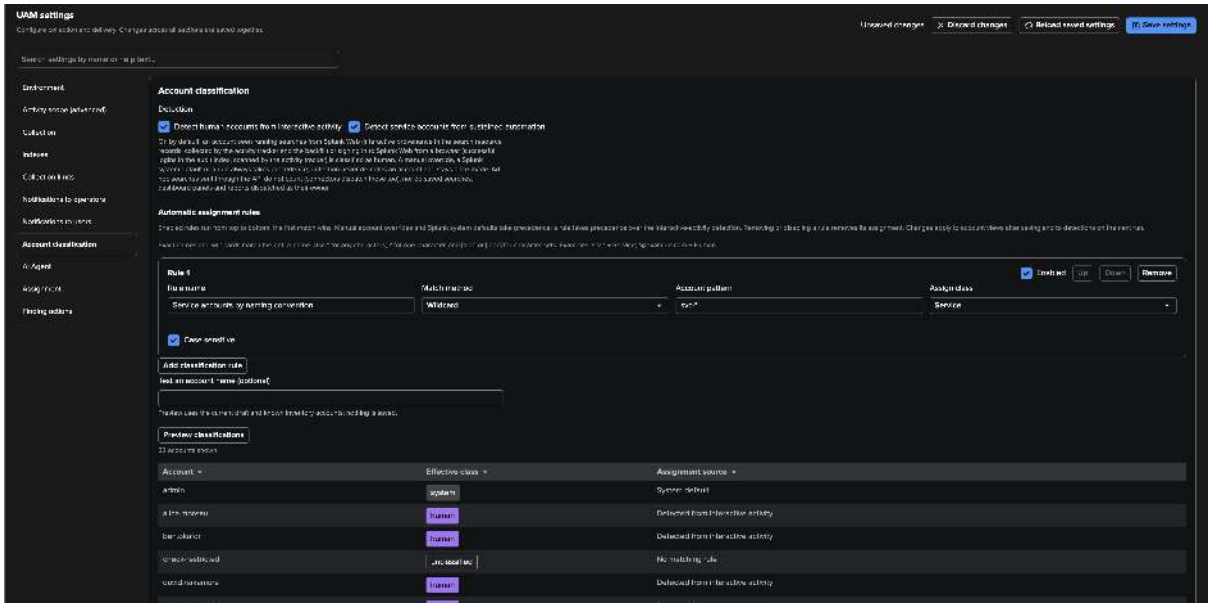
The account. Every observation carries `account` and `account_source` (`legacy_search_user`, `structured_actor` or `unknown`). The legacy `user` header wins when present: it names the caller, while the structured actor is often the system principal. It is a **labelled source claim**, not a verified initiator: UAM never claims that a person sat at the keyboard, and a scheduled search runs under its owner or the system account.

Classes and detection.

Class	Detected when	Limits
`human`	An **interactive search** — a resource record whose provenance starts with `UI:` (`UI:Search`, `UI:Dashboard:<name>`, `UI:Report`) and that carries no saved-search label — or a **successful browser login** to Splunk Web. One is enough; the detection is sticky and never demotes. Switch: *Detection* (human).	Ad hoc alone is not a signal (connectors dispatch ad hoc searches through the API); a labelled dispatch is not either (a dashboard set to run as its owner). Users working only through React apps (TrackMe's UI included) reach splunkd as `rest:jobs` — the login signal covers them.
`service`	**Sustained automation and nothing a person does**: automated searches (REST, scheduler, acceleration, internal) on at least **seven distinct days**, never a Splunk Web search, a splunkjs page, an MCP client or any other provenance a person might use, and not a detected human. Withdrawn as soon as such a search appears. Switch: *Detection* (service).	The automation days come from the legacy key=value `_audit` format (`provenance=` / `savedsearch_name=`): a JSON-only `_audit` gives no service detection, and the seven days must lie inside the audit retention. An account with no activity is never detected.
`system`	Seeded for `nobody`, `splunk-system-user` and the built-in `admin` whenever unset; never detected, never demoted by a detection.	Assign it by hand to the tenant owner and to the administrator accounts that run the platform's own workload.

Rules. Up to 100 ordered **classification rules**, each with a name, an enabled flag, a match type (`exact` or `wildcard` — `*`, `?`, `[abc]`, `[!abc]` on the whole name; no regular expression), a pattern, case sensitivity and the class. No naming heuristic runs unless an administrator saves a rule. **Manual classes** (up to 10,000, from *Settings*, the Users tab or its bulk classification) always win.

Precedence, in every reader (Users tab, account view, inventory, policies): manual class → system default → first enabled matching rule → human detection → service detection. Unmatched, undetected accounts stay **unclassified**. Removing a manual class returns the account to the automatic resolution; disabling a rule stops it at once; switching a detection off ignores its records without deleting them. *Settings → Account classification* previews the draft (up to 1,000 names, the effective class and its source) before saving; the Users tab shows a *detected* marker with the evidence next to the class. Changing a class never rewrites historical evidence nor resolves an investigation: the policies use it at their next run.



Privileged users

The **Privileged users** tab and *Settings → Privileged users* (`privileged` in the configuration) answer which accounts hold high privileges on the monitored tiers, and why.

Definition. An account is privileged when one of its roles, or a role that role imports (`imported_roles`, resolved transitively; a cycle is tolerated), is a **privileged role** or grants a **privileged capability**.

Inheritance. Splunk grants roles and capabilities through direct membership and through inheritance, and a role inherits from roles that inherit further: the assessment resolves the import chain breadth first. An inherited reason keeps its chain (`path`: the role held first, the privileged role last), an inherited capability names the role that grants it itself (`granted_by`, the nearest one; none when only Splunk's `imported_capabilities` knows it), and each role keeps its own definition (its own capabilities and the roles it imports itself) apart from what it inherits. The tab words them: *team_lead imports admin*, *team_lead inherits admin through ops*, *leaf inherits list_settings from base through mid*.

Rule (key)	Default	Behaviour
Role name patterns (`role_patterns`)	`*admin*`	Live, case-insensitive globs (`*`, `?`): a role created later is privileged as soon as its name matches.
Roles added (`roles_include`)	none	A role no pattern matches, made privileged by name.
Roles excluded (`roles_exclude`)	`trackme_admin`	Overrides a pattern. It removes only the name rule: a privileged capability the role grants still counts.
Privileged capabilities (`capabilities`)	`admin_all_objects`, `change_authentication`, `delete_by_keyword`, `edit_roles`, `edit_tokens_all`, `edit_user`, `list_settings`, `list_storage_passwords`, `rest_apps_management`	Granted directly (`capabilities`) or through imported roles (`imported_capabilities` and the capabilities of every role imported).

Up to 500 entries per list. A role name holds letters, digits and `_ . -`; a pattern also `*` and `?`; a capability also `:`; a role cannot be both added and excluded.

Data. No new collection and no call to Splunk: the assessment reads the published users and roles catalogs of every tier (each role's `capabilities`, `imported_roles` and `imported_capabilities`, each user's `roles`), so it

follows *Users and roles every* (60 minutes by default). An account present on several tiers is one account, and each of its reasons names the tier that grants it.

What the tab shows. Tiles for the privileged accounts (and their share of all accounts), the privileged roles, the privileged capabilities in use and the privileged accounts active in the last 24 hours; pies of why accounts are privileged (role, capability or both) and of their classes; a bar of the accounts per privileged capability; the privileged accounts with their reasons in words, last seen and executions over 24 hours; the roles of every tier with their rule, inherited roles and privileged capabilities. The activity is one `tstats` over the audit records, read as you, naming the privileged accounts (up to 1,000; above, it reads every account and keeps the privileged ones); when it cannot be read, the tab says the activity is unavailable, never that the accounts were inactive.

Settings. The patterns in a text area, then two dual lists filled from what the tiers hold: roles (moving a role a pattern matches out of the privileged list adds it to the exclusions; moving another role in adds it by name) and capabilities. The rules apply from the next read of the tab.

The tab describes; the four privileged policies (see *Privileged policies*) raise findings from the same assessment.

The built-in policy catalogue

A UAM tenant evaluates **sixteen built-in policies** in four families. Every policy has:

- **Enabled** — on or off.
- **Severity** — `low`, `medium`, `high` or `critical` (UAM has no informational level). A manual severity set on a Finding group survives later runs; choosing *Members* (the highest severity of its findings) restores the derived one.
- **Excluded owners / accounts** and **Excluded apps** — up to 500 exact, case-sensitive names each (no wildcard). Every policy except the CSV lookup footprint starts with the apps `Splunk_SA_CIM` and `trackme` excluded, so TrackMe's own trackers and data-model acceleration do not flag the monitoring itself. *Exclude owner from this policy* in a finding menu adds the owner to this list.
- **Account scope** (account policies) — `human` (explicitly classified human), `classified` (human or service, never system) or `all`. Accounts outside the scope count as *not applicable*.
- Its **thresholds**, each compared **strictly**: a finding needs *more than* the limit.
- **During bank holidays** — *Hold new findings* or *Keep opening findings* (see *Policies run*). The six inventory policies hold by default: configuration reviews nobody handles on a day off, still there the next working day. The activity and resource policies keep opening findings by default: unusual activity on a day off is more telling, not less. A configuration stored before 2.4.19 reads each policy's default. Changing it never resets the direct user notifications.

The policy editor (*Policies → Edit*) has four tabs: **General settings** (status, *During bank holidays*, severity, scope, thresholds, automatic budgets, exclusions, *Preview policy*), **Email template** (the direct user emails of this policy, see *Notifications and communications*), **Actions** and **Automations** (findings actions, see *Findings actions*).

Inventory: saved searches

Policy	What it detects	Default	Severity
`personal_schedule/v1`	An enabled schedule owned by an account classified human.	On	Low
`unclassified_schedule/v1`	An enabled schedule whose owner has no human or service class.	Off	Medium
`dense_schedule/v1`	More scheduled runs per active hour than the limit.	Off	Medium
`all_time_schedule/v1`	A schedule with no lower time bound (all time).	On	Medium
`realtime_schedule/v1`	A schedule with a real-time dispatch range.	On	Medium

Inventory: accounts

Policy	What it detects	Default	Severity
`csv_lookup_footprint/v1`	A CSV lookup file too large, or an account's CSV lookups too large in total.	On	Medium

Activity: accounts

Policy	What it detects	Default	Severity
`launch_burst/v1`	An account launching an unusual number of searches.	Off	Medium
`costly_search/v1`	An account running repeated long searches.	Off	Medium
`all_time_search/v1`	An account running all-time searches.	On	High

Resource: accounts

Policy	What it detects	Default	Severity
`resource_consumption/v1`	An account's search CPU above its allowance.	On	Medium → High
`resource_intensity/v1`	An account running repeated searches heavy in CPU or memory.	On	Medium
`resource_deviation/v1`	An account's CPU today unusual for its daily baseline.	Off	Medium

Privileged: accounts and roles

Policy	What it detects	Default	Severity
`privileged_account_new/v1`	An account that became privileged (reported once per change).	Off	High
`privileged_role_capabi/v1`	A role whose own definition now grants more privileges (reported once, about the role).	Off	High
`privileged_account_doi/v1`	A privileged account without activity for the period.	Off	Medium
`privileged_reference/v1`	A privileged account the reference does not declare, or a stale reference.	Off	High (stale: Low)

Default settings

What each policy applies to, its account scope, what it does on a bank holiday, and its thresholds with their defaults.

Policy	Applies to	Bank holidays	Thresholds and notes (defaults)
`personal_schedule/v1`	Saved search	Hold	No threshold. Common where administrators own every schedule, hence **low**.
`unclassified_schedu v1`	Saved search	Hold	No threshold. A governance gap, not evidence of misuse.
`dense_schedule/v1`	Saved search	Hold	Maximum scheduled runs per active hour 12 (1 to 60). Configured frequency, not observed launches.
`all_time_schedule/v1`	Saved search	Hold	No threshold. `dispatch.earliest_time` of `0` or empty.
`realtime_schedule/v1`	Saved search	Hold	No threshold. An explicit `rt` bound.
`csv_lookup_footprin v1`	Account, scope `all`	Hold	100 MiB per file, 500 MiB per account and tier. No app excluded by default.
`launch_burst/v1`	Account, scope `human`	Keep opening	Maximum distinct searches per window 200.
`costly_search/v1`	Account, scope `human`	Keep opening	Minimum run time 300 s, maximum long searches 3.
`all_time_search/v1`	Account, scope `human`	Keep opening	Maximum all-time searches **0**.
`resource_consumptio v1`	Account, scope `classified`	Keep opening	Allowance 10 % of the search CPU capacity (else 3,600 s). Raised one severity level above a dominant share of 40 %.
`resource_intensity/v1`	Account, scope `classified`	Keep opening	A heavy search uses 2 % of the search CPU or 10 % of the smallest search host memory (else 300 s or 2,048 MB); maximum heavy searches 3.
`resource_deviation/v1`	Account, scope `classified`	Keep opening	3 × the daily p95 over 14 days, floor 3,600 s. Off, in **preview mode**: matches are recorded, no finding and no notification.
`privileged_account_ v1`	Account, scope `all`	Keep opening	No threshold. Splunk-operated accounts left out (*Include accounts operated by Splunk*, off).
`privileged_role_cap v1`	Role	Keep opening	No threshold. Exclusions name roles.
`privileged_account_ v1`	Account, scope `all`	Hold	Days without activity 30 (7 to 365). Splunk-operated accounts left out.
`privileged_referenc v1`	Account, scope `all`	Keep opening	No declared account and no lookup table (the policy stays learning), *Report stale references* on. Splunk-operated accounts left out.

The server supplies the titles, descriptions, bounds and remediation of every policy; the sections below give the full detail.

Inventory policies — saved searches

The subject is one saved search of one tier; the finding keeps the same identity across runs. Explicitly unscheduled or disabled searches are not applicable; a missing flag or an unreadable bound is *unknown*, never assumed. Evidence: the owner and its class (with where the class comes from), the app, the tier, the schedule and the bounds read. UAM never edits nor dispatches the observed searches — the remediation is advisory. These policies follow the catalog cadence: the saved-search listing is daily by default.

Policy	Details
`personal_schedule/v1` *Scheduled search owned by a human account*	• **Counts:** enabled schedules whose owner is explicitly classified **human** (unknown and service owners do not match). • **Thresholds:** none. Common where administrators own every schedule, hence the **low** severity. • **Suggested action:** review whether this schedule should run under an approved service account, or record an owner exception.
`unclassified_schedule/v1` *Scheduled search owner is not classified*	• **Counts:** enabled schedules whose owner has no human or service class. A governance gap, not evidence of misuse. • **Thresholds:** none. • **Suggested action:** classify the owner in Settings after confirming the account's purpose.
`dense_schedule/v1` *Scheduled search exceeds the hourly schedule limit*	• **Counts:** distinct **configured** minute slots in an active clock hour of a standard five-field cron (wildcards, steps, ranges and lists; duplicates count once; named days, macros and extended forms are unknown). Configured frequency, not observed launches. • **Thresholds:** maximum scheduled runs per active hour, 1 to 60 (default 12). • **Suggested action:** review the business need and reduce the frequency, or add a justified exception.
`all_time_schedule/v1` *Scheduled search has an all-time dispatch lower bound*	• **Counts:** enabled schedules whose `dispatch.earliest_time` is `0` or **empty** (Splunk's default: no lower bound). Inline SPL time modifiers may still restrict the search. • **Thresholds:** none. • **Suggested action:** configure a bounded dispatch time range where appropriate.
`realtime_schedule/v1` *Scheduled search uses a real-time dispatch range*	• **Counts:** enabled schedules with an explicit real-time bound (`rt`, `rt-5m@m`, `rt@w0`…). An unsupported `rt` form is unknown, never historical. • **Thresholds:** none. • **Suggested action:** review whether a scheduled historical search can meet the requirement.

CSV lookup footprint

`csv_lookup_footprint/v1` — *CSV lookups attributed to an account exceed its size allowance* — is an inventory policy whose subject is an **account**:

- **What it detects**, separately on every search head tier: **one CSV lookup file** larger than *Maximum individual CSV file size* (default **100 MiB**), or the **cumulative size** of the CSV lookups attributed to one account larger than *Maximum cumulative CSV size per attributed account and tier* (default **500 MiB**). Both 1–1,048,576 MiB.
- **On by default**, severity medium, account scope `all`, no app excluded by default.
- **Attribution: probable writer, recorded owner as fallback.** The Audit activity tracker records, from the audit evidence it already reads, which account last wrote each lookup file. A file is attributed to that **probable writer** when the evidence supports it, else to its **recorded owner** (a shared file owned by `nobody` falls back to its recorded owner unless audit evidence names a writer). The evidence keeps the basis of the attribution and both identities. Attribution is best effort since monitoring began, not proof of authorship: missing audit evidence, ambiguous targets, concurrent writes or later unmatched changes use the owner fallback, and the backfill does not provide writer hints.
- **Size** is the stored file size (a compressed CSV counts at its stored size), not the replicated bundle size. Only the metadata is read — never the contents — and no third-party app is required.
- Only tiers whose lookup-file catalog is **complete and less than an hour old** are evaluated; a missing, partial or stale catalog can never clear a finding.
- *Suggested action*: review the largest files and their generating searches with the probable writer or the owner; reduce retained rows or columns, use the KV store where appropriate, review bundle replication settings with

an administrator. UAM never deletes a file nor changes replication. Direct user emails of this policy list up to 20 of the largest attributed lookups.

Activity policies — accounts

Aggregated from the `trackme:uam:observation` records of the evaluated *Activity policy window* (15 minutes of collected evidence by default, see *Policies run*), one row per search first, then per account and app. The counts are **distinct search ids** the Audit activity tracker observed — a launch-count proxy — labelled with the coverage (`complete` / `partial`). One finding per policy, environment and account: a repeated burst updates the same finding. Evidence: the window, the count, the thresholds, the account class and scope, the top five apps and, for `costly_search/v1`, the maximum and total run time. Excluded apps are removed from the account's aggregate before the threshold is compared.

Policy	Details
`launch_burst/v1` *Account launched an unusual number of searches*	• **Counts:** distinct searches by one account in the window. • **Thresholds:** maximum distinct searches per window, 1 to 100,000 (default 200). • **Suggested action:** confirm whether the burst is expected automation; raise the limit or record an account exception if it is.
`costly_search/v1` *Account ran repeated long searches*	• **Counts:** distinct searches whose reported run time reached the minimum. Run time is the audit completion figure; CPU is not estimated. • **Thresholds:** minimum run time 1 to 86,400 s (default 300); maximum long searches 0 to 100,000 (default 3). • **Suggested action:** review the long searches (run time, scan count); work on time ranges, scheduling or acceleration.
`all_time_search/v1` *Account ran all-time searches*	• **Counts:** distinct completed searches whose reported earliest bound is explicitly unbounded (legacy `search_et=N/A`, structured `0`). A missing bound is unknown. • **Thresholds:** maximum all-time searches, 0 to 100,000 (default **0**). • **Suggested action:** agree on bounded time ranges, or an account exception for approved investigations.

Resource policies — accounts

Read from the `trackme:uam:resource` records (search heads and indexers, ad hoc and scheduled), through one bounded rollup per policies run: CPU adds up across windows, memory keeps its peak. The `system` class is outside the default `classified` scope (data-model acceleration never becomes a finding by default) but its CPU stays in the share denominator: it is real load.

Automatic budgets. With *automatic thresholds* on (the default), the absolute limits are derived from the capacity — the core sum of the search hosts and the memory of the smallest one — and clamped to each threshold's own range:

Policy	Percentage (default)	Derives
`resource_consumption/v1`	Allowance as a share of the search CPU capacity (10 %)	Maximum CPU seconds = search cores × window seconds × percentage
`resource_intensity/v1`	Heavy search CPU as a share of the search CPU capacity (2 %)	Minimum CPU seconds of a heavy search = search cores × window seconds × percentage
`resource_intensity/v1`	Heavy search memory as a share of the smallest search host (10 %)	Minimum peak memory of a heavy search = smallest search host memory × percentage

With automatic thresholds off, the stored absolute values apply. With them on but no usable capacity, the stored values apply and the evaluation says `manual_fallback`. The evidence carries the thresholds source (`auto` / `manual` / `manual_fallback`) and the capacity basis; the policy editor shows the effective values.

Learning period. The resource rules are **learning** — nothing evaluated, no finding created, nothing marked not

observed — until the tenant holds *Resource rules: minimum history* (24 hours by default) of resource history (the backfill counts) and, for automatic budgets, a known capacity. The Policies tab shows the reason.

Policy	Details
`resource_consumption/v1` *Account's search resource consumption exceeded its allowance*	• **Counts:** the CPU seconds of the account's searches in the window, against its allowance. Its share of the environment's search CPU is always in the evidence; strictly above the **dominant share** the severity is raised **one level**. The share never opens a finding on its own. • **Thresholds:** maximum CPU seconds 1 to 10,000,000 (default 3,600, derived when automatic); allowance 1 to 100 % (default 10); dominant share 1 to 100 % (default 40). • **Suggested action:** open the account's most expensive searches; work with the owner on time ranges, acceleration, scheduling or workload rules; raise the allowance or record an exception for approved workloads.
`resource_intensity/v1` *Account ran repeated resource-intensive searches*	• **Counts:** distinct searches whose CPU seconds **or** peak memory reached the per-search minimum (a search spanning several windows is summed; memory keeps its peak). • **Thresholds:** minimum CPU seconds of a heavy search 1 to 1,000,000 (default 300, derived); minimum peak memory 64 to 1,000,000 MB (default 2,048, derived); maximum heavy searches 0 to 100,000 (default 3); automatic percentages 2 % CPU and 10 % memory. • **Suggested action:** review the heavy searches (type, provenance, label); agree on bounded time ranges, acceleration or scheduling.
`resource_deviation/v1` *Account's CPU consumption is unusual for its daily baseline*	• **Counts:** today's accumulated CPU (UTC day) against `max(multiplier × historical daily p95, floor)`, after at least the minimum number of **fully covered** prior days since the account was first seen. Missing coverage is unknown, never zero; today is never extrapolated. • **Thresholds:** historical lookback 7 to 30 days (default 14); minimum complete days 7 to 30 (default 7, at most the lookback); multiplier 1 to 20 (default 3); minimum daily CPU seconds 1 to 100,000,000 (default 3,600). In **preview mode** by default: matches are recorded in the evaluation, no finding and no notification. • **Suggested action:** review the daily baseline, the coverage and the expensive searches; confirm workload changes before adjusting the multiplier.

Privileged policies

The four privileged policies read the privileged assessment (see *Privileged users*) of every tier, from the published users and roles catalogs, in the policies run. They have no excluded apps (they judge roles and accounts, not searches) and never email the account a finding is about.

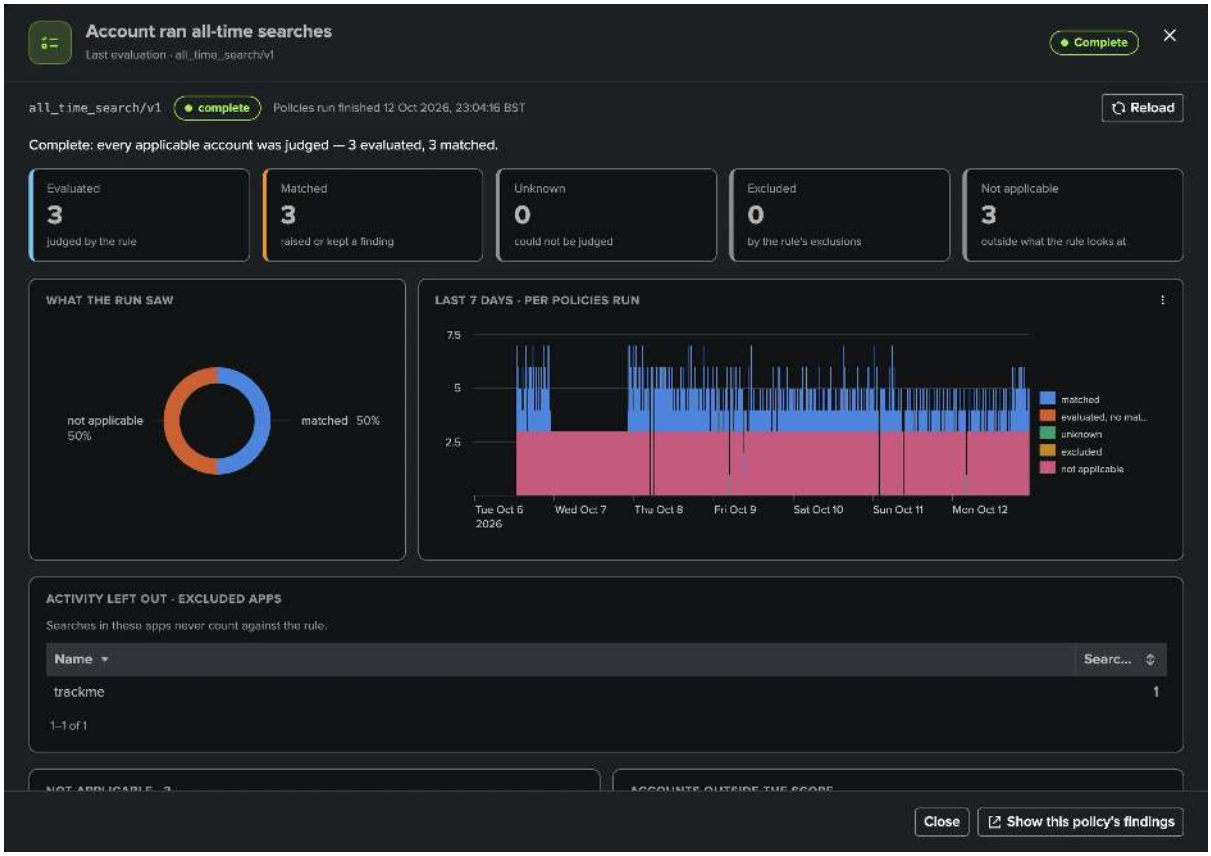
Changes, reported once. The two change policies compare the current assessment with the previous one, which the policies run keeps (a snapshot, per tier). A change is matched by the run that finds it and opened as a finding; the next complete run no longer matches it, so a resolve sticks and a later change opens a new episode. A change held by a bank holiday stays pending and is opened by the first run after it. The first evaluation, a change of the Privileged users rules, and a change policy turned on record the current state without reporting. A partial users or roles catalog never moves the snapshot, never matches and never clears.

Policy	Details
`privileged_account_ne` `v1` *Account became privileged*	• **Matches**: an account privileged now that was not at the previous assessment of its tier, whose own roles changed (`roles_changed`) or that is new (`account_created`). An account made privileged with its own roles unchanged is the role policy's finding; with the role policy off, or the originating role excluded from it, it is this policy's (`privilege_inherited`, naming the roles whose definition changed). • **Evidence**: the roles before and after, the reasons with their chains, and, when the audit trail holds the change (`create_user` / `edit_user`), who made it and when. • **Suggested action**: confirm the grant was approved; otherwise remove the privileged role and review the account's activity since.
`privileged_role_capak` `v1` *Role gained a privileged capability*	• **Matches**: a role whose **own** definition changed — its own capabilities (`capabilities_gained`), the roles it imports itself (`imports_changed`) — so that it grants more privileged capabilities or inherits a privileged role, or a new role created privileged (`role_created`). The roles inheriting it are not reported again: the finding lists them (`descendant_roles_reached`) with every account newly privileged through any of them. A role whose inheritance grew while no role in its chain changed its own definition in the catalog is reported on itself (`inheritance_changed`). • **Subject**: the role, on its tier (a Finding group of its own); findings actions read its name as `$subject.name$`. • **Evidence**: the capabilities gained and the role granting each, the own capabilities and imports before and after, and, from `edit_role`, who made it with the capabilities before and after. • **Suggested action**: confirm the change was approved; otherwise restore the role's capabilities or imports.
`privileged_account_dc` `v1` *Privileged account is dormant*	• **Matches**: a privileged account with no search in the audit evidence (one `tstats` over the period, naming the privileged accounts), no Splunk Web login or interactive search seen by the Account signals tracker, and no successful login Splunk reports on any tier (`last_successful_login`, collected by the users catalog without making it an inventory change), for *Days without activity*. • **Never a guess**: an account in the inventory for less than the period is not applicable; while the tenant's audit evidence (the live history, plus a completed historical backfill) does not cover the period, the account is unknown. • **Not seen**: activity on a search head outside the Activity scope, and REST calls that run no search.
`privileged_reference/` `v1` *Privileged accounts differ from the reference*	• **Reference**: the account names typed in the policy (every tier) and each lookup table, read with `| inputlookup <lookup>` on its deployment — the local search head, or a remote deployment through its TrackMe remote account — from the TrackMe app: the account field, and the tier field when set (an empty tier declares every tier). Matching is case-insensitive; `*` matches any text in typed names and in a lookup set to Wildcard (Exact, the default, ignores a value holding `*`). Up to 20 lookups of 500,000 values each; TrackMe's own lookups are refused. • **Matches**: `undeclared` — privileged on a tier the reference does not declare it for; `stale_reference` (with *Report stale references*, at low severity) — an exact declared name holding no privilege, or absent from every tier (a pattern is never stale). The evidence names the sources that declare the account. • **Never a guess**: a lookup that cannot be read, reads empty, is too large or is not read in time keeps its last good values (diagnostic `privileged_reference_unreadable`, note *reference_lookup_cached*); a lookup with no good values yet, or a reference that declares no account, judges nothing; without a source, the policy is learning. A tier field with no value in any row is treated as unreadable (a renamed column).

Accounts whose collected email address is `support@splunk.com` — the accounts Splunk operates on a Splunk Cloud stack — are left out of the account policies unless *Include accounts operated by Splunk* is on. *How it is evaluated* shows the records judged and, for the dormancy and reference policies and the change author read, the exact searches.

Policy insight and preview

- **Policy insight** — the *Last evaluation* and the count cells of the Policies tab open what the last run saw: a verdict, the population, why records were unknown, excluded or not applicable (`owner_unclassified`, `missing_<field>`, `cron_unreadable`, `excluded_owner` / `excluded_app`, `unscheduled` / `disabled`, `out_of_scope_<class>`…), the owners and apps behind each bucket (owners can be classified in place), up to 25 samples, the run (window, readiness, thresholds source, inventory per tier) and a seven-day history per policies run.
- **Preview policy** (*General settings*) evaluates the draft against the latest stored inventory of the tenant without writing anything and shows the totals and up to 20 matching examples. It needs a collected inventory.



Custom policies

Tenant administrators write their own policies in SPL (*Policies → Create policy*). A custom policy runs on its own schedule and its findings appear with the built-in ones, in the same Finding groups.

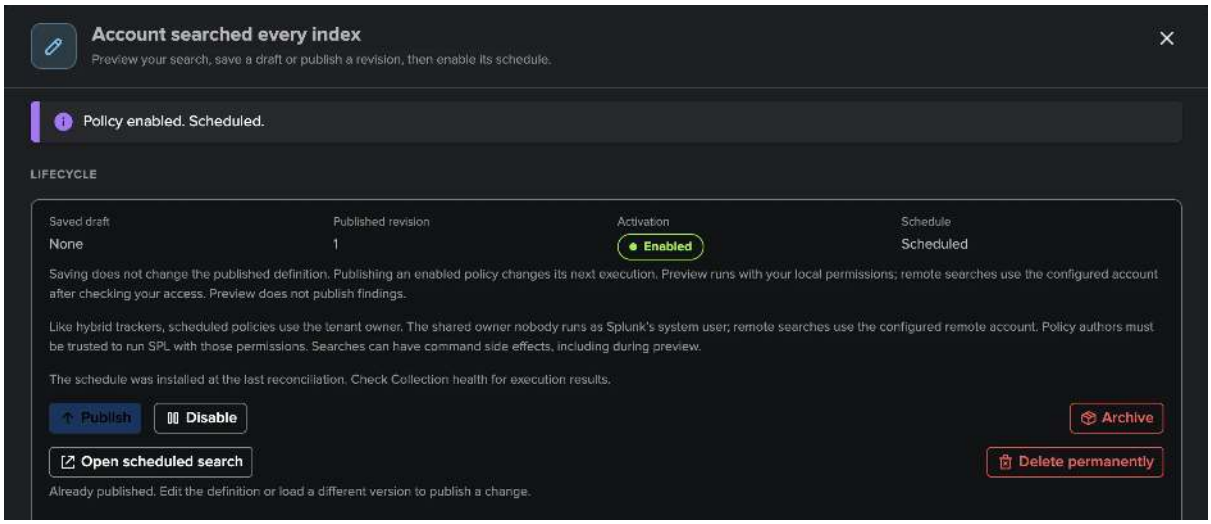
Definition.

- Title, description, default severity and remediation (shown on its findings).
- **Source tier** — a policy about **accounts** reads the route account (the indexing layer); a policy about **saved searches** reads the selected tier. **App context** — the search namespace.
- **SPL** — up to 64 KiB. **Time range** — relative, from `-<n><s|m|h|d|w>` to `now` or an earlier relative bound, at most 31 days in the past; no all-time, real-time, future bound or calendar snap. **Schedule** — a numeric five-field cron.

Map search results to findings.

- **Subject type**: **account** (the result field holding the account) or **saved search** (the fields holding the owner, the app and the name — three distinct fields; alias them in the SPL). The subject places the finding in the right Finding group.
- **Dimensions** (up to eight fields) — distinguish several findings about one subject.
- **Evidence fields** (up to eight), a **count field** (occurrences per row, else one per row) and a **time field** (epoch seconds).
- **Advanced limits**, for example the result rows read (10,000 by default, at most 100,000), the findings per run (1,000, at most 10,000), **25 findings per subject**, the samples and evidence bytes kept per finding.
- Changing the subject mapping or the source requires a higher **identity version** before publication: a new identity, so earlier findings are not reused for a different meaning.
- **Bank holidays** — *Hold new findings* or *Keep opening findings* (`hold_on_bank_holidays`, off when absent). It is not part of the identity, so turning it on needs no new identity version; it takes effect when published. A custom policy holds while a holiday of the tenant's calendars is active at its **anchor**, the end of its search window: a holiday that ended earlier inside a longer window no longer holds. Held results publish no finding and use no findings capacity; a finding already observed is still evaluated. The run's commit records `held` and, when a holiday applied, `bank_holidays` (state and names), shown as *Held N (bank holiday: <names>)* on the run.

Lifecycle.



Step	What it does
Draft	Save keeps a **saved version** of the draft (the 30 most recent are kept). Saving never publishes nor enables.
Preview	Runs the current form — saved or not — as **you**, asynchronously and cancellable (60 seconds, 1,000 result rows), and shows completeness, diagnostics, counts and at most 20 candidate cards. Nothing is published; a zero-row result cannot verify a mapping.
Publish	Freezes an immutable **published revision** (up to 100 per policy). Draft edits never change what is published.
Enable / Disable	Schedules or unschedules the policy's report `trackme_uam_custom_<id>_tenant_<tid>`, owned by the tenant owner.
Archive	Irreversible: the policy stops, its definition and findings stay readable and it can be duplicated.
Delete permanently	Removes the definition, its versions, its scheduled search and its execution history; findings, evidence and notes remain.
Version history	Load, compare, restore (as a new draft) or delete a version; the published revision is protected.

Limits. 100 definitions per tenant (archived ones included) and 25 enabled. A scheduled run has 300 seconds and

10,000 result rows; at most 1,000 findings are kept per custom policy and 10,000 per tenant, separately from the built-in findings limit. Four custom policy runs at a time per installation, two per tenant, one per policy (previews included).

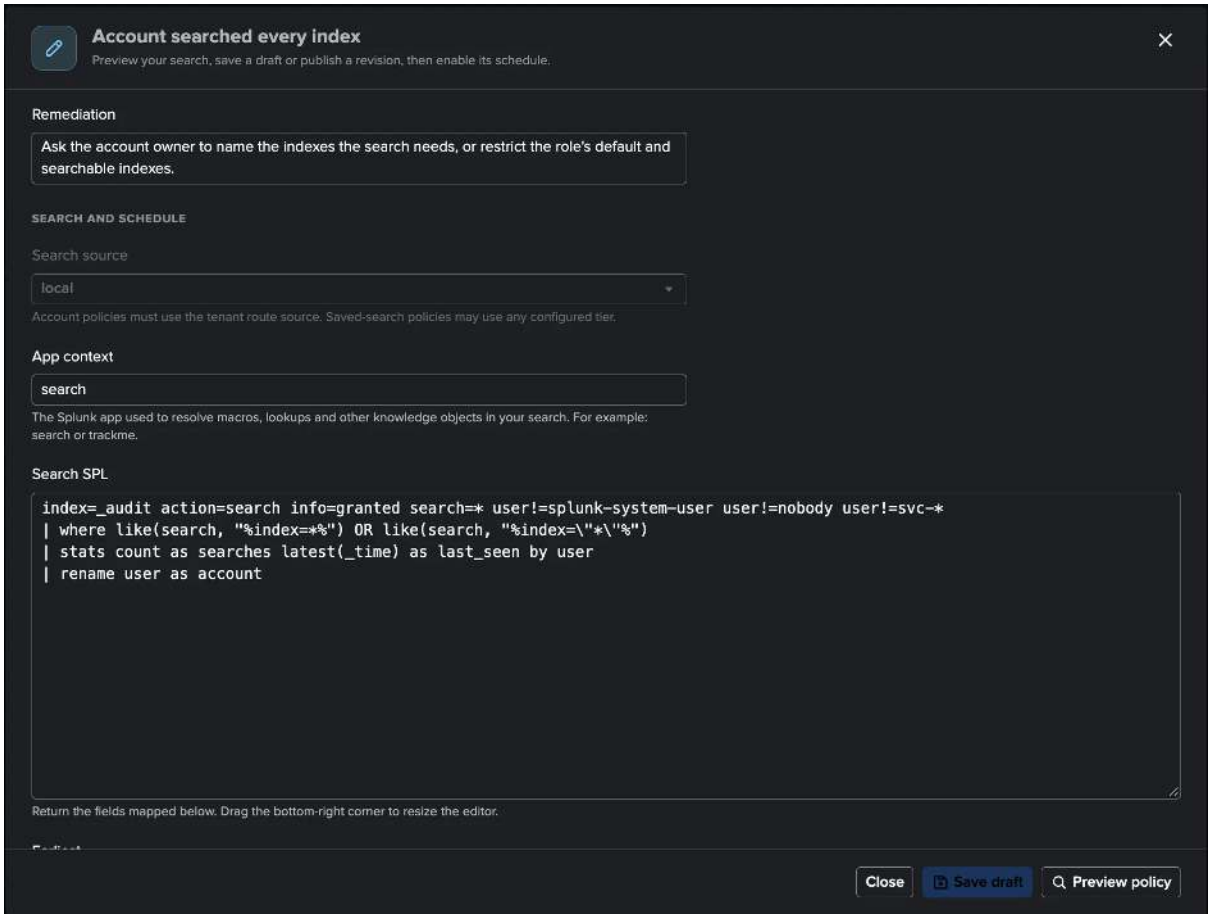
Execution health. *Collection health → Trackers*, family *Custom policies*, shows, per policy, the requested and actual schedule, the cron and next run, the last complete evaluation, the failure count, the recent runs and any running search; states are disabled, never run, healthy, healthy (no match), incomplete, failed, cancelled, capacity deferred, schedule drift, missed schedule and recovery required. *Run now* runs the published revision (findings may be published) and **Recover** frees a run whose searches are verified stopped. The Configuration Guardian reports the problems (see *Configuration Guardian*).

What differs from built-in policies.

- Custom findings are **not** notified: the operator and user notification policy selectors list built-in policies only.
- There is no *Exclude owner from this policy* action: write exclusions in the SPL.
- A custom policy can only map to accounts and saved searches.

Warning

The SPL of a custom policy is arbitrary SPL run on a schedule as the tenant owner (with the `nobody` owner, as Splunk's internal scheduler identity), and a preview runs it as its author: a command or macro with side effects has them, preview included. Only trusted tenant administrators can author policies; give remote accounts least-privilege credentials.



Findings and Finding groups

The shared lifecycle, triage and grouping concepts are explained in *A finding and its lifecycle* and *Finding groups*; this section gives the UAM specifics.

The finding

A finding is one row per policy, environment and subject (a saved search, an account, or a role for the role policy), with a stable id, its `status` (`open`, `investigating`, `resolved`, `dismissed`), its severity (policy or manual), its assignee, a short investigation note, `observed`, first / last seen, `episodes`, `reopened_count`, the dismissal deadline, a bounded history (the 50 most recent records) and the evidence.

Transition	Meaning
`new`	First match: the finding opens.
`updated`	Severity or evidence changed while observed (a threshold edit updates the evidence without changing the identity).
`reobserved`	Matched again after an absence: a new episode; open and investigating findings only gain the history record.
`unobserved`	Not matched by a **complete** evaluation: the finding stays, its *Condition* reads **not observed**. It is never resolved for you; a partial evaluation never marks a finding not observed.
`reopened`	A **resolved** finding observed again returns to `open`.
`expired`	A **temporary dismissal** ended: the finding returns to `open` at the next policies run, observed or not.
`triage`	An operator decision; the policies never overwrite it.
`retired`	The policy no longer exists and its replacement evaluated fully: resolved by the system, not notified.

Decisions

Decisions are made on the **Finding group** (one group, or up to 500 groups at a time), fenced on the group's version (a decision captured on an older version is refused, `finding_group_version_changed`):

- **Status —** *Investigate*, *Resolve*, *Dismiss*, *Reopen*. **Resolve** says the investigation concluded: the policy **reopens** the finding the next time it observes the condition. **Dismiss** says the finding is accepted: it stays dismissed while observed. Dismiss permanently ("until reopened by hand") or for **1, 7, 30 or 90 days** (API: one hour to 366 days); an ended dismissal returns the finding to open. Any status change clears a deadline; a new duration replaces it.
- **Snooze notifications —** 1 hour, 1 day, 7 days or 30 days (API: 60 seconds to 90 days): the operator emails about the group are held; the status does not change. Transitions during the snooze are not sent later.
- **Severity —** a manual severity; *Members* restores the severity derived from the group's findings.
- **Assign —** to a user or an assignment group (see *Assignment and automatic routing*).
- **Notes —** the shared TrackMe Notes (Markdown, permanent or expiring) on the group or on a finding, separate from the short investigation note recorded in the history.

Every change leaves a history record with its author (an operator, the policy, the system or the AI Agent). Decisions on findings and groups (`triage`, `group_members`) require a TrackMe administrator of the tenant; a power user of the tenant can run findings actions, send an email action and set an assignment.

Finding groups

Automatic grouping. All findings about **one account and the saved searches it owns** form one Finding group — for example the two personal schedules of `analyst-a` and its all-time-search finding. Ownership defines the grouping; classification does not. A saved search without an owner keeps its own group. Every tier of the indexing layer shares the account's group; different tenants never do. A role finding (the privileged role policy) keeps a group of its own. Each finding keeps its own subject, tier, policy, evidence and id. An automatic group holds up to **5,000** findings.

The group is the triage unit. A decision applies to every member except a member **dismissed on its own** (*Dismiss only this finding*): that member keeps its own dismissal until it ends or is reopened, and the group's decisions leave it alone. A new kind of problem on a dismissed group reopens the group and keeps the earlier members dismissed on their own: a dismissal never silences a new problem. The group's severity is the highest of its observed members unless an operator set one.

Manual grouping. Select groups on the Findings tab:

- **Merge into one Finding group** — 2 to 25 groups become one manual group (optional title). The most open status wins; a member whose decision differs keeps it on its own.
- **Add to a manual group**, **Remove from the Finding group** (up to 25 findings at a time) and **Ungroup**.
- An object's findings always move together: removing a saved-search finding removes every finding of that object; removing an account returns it with its owned searches.
- A manual group holds up to **50** findings; later findings of a full manual group go back to their automatic group. Automatic grouping never undoes a manual grouping.

Assignment and findings actions

- **Assignment** (*Settings → Assignment*): up to 100 assignment groups (name, description, optional team email) and up to 100 ordered automatic assignment rules (by policy, severity, subject type, owner, app, environment), with optional emails to the assigned group. See *Assignment and automatic routing*.
- **Findings actions** (*Settings → Finding actions*): up to 50 administrator-defined drilldown searches, links, webhooks (through system-wide *Webhook accounts*) and the email action, run by hand from a finding or group menu, and up to 100 automation rules that run them for a policy (*Policies → Edit → Automations*). See *Findings actions*.

Finding events

Every finding transition is indexed as `trackme:uam:finding` in the findings index (the tenant's notable index by default): build your own Splunk alerts, Enterprise Security correlation searches or SOAR playbooks on them.

Notifications and communications

UAM has three email channels, all independent of each other. The shared principles are in *Notifications*.

Notifications to operators

Settings → Notifications to operators (`notifications`), off by default:

- **Events**: *New finding*, *Finding reopened*, *Severity raised*, *Finding resolved* (on by default), *Finding dismissed*, *Finding assigned*. Being observed again or not observed never notifies.
- **Minimum severity** (medium), **Policies** (empty = all built-in policies), **Delivery account** (a TrackMe email account, see *Configuration*; empty = the search head's local mail transport), **Recipients** (up to 50, filtered by the account's allowed domains), **Mode**: **Digest** (one email per policies run, grouped by event then severity) or **Per finding** (one email per transition, threaded per finding in the mail client).
- Transitions are delivered at the end of the next policies run (within about 15 minutes); a failed send is retried at the next runs, and transitions older than 24 hours expire. A snoozed group's transitions are not sent.
- **Send test email** sends a synthetic message with the **saved** settings, even when notifications are off.
- Every email links to the finding (`TenantHome?tenant_id=<tid>&component=uam&tab=findings&finding=<id>` opens its Investigate view).

Notifications to users

Settings → Notifications to users (`user_notifications`), off by default: one email to the **account a finding is about**, on **detection** and when the condition is **no longer detected**.

- **Addresses** come from Splunk's user records, collected per tier by the users catalog and kept in a private store (never in the inventory, the evidence or the findings), or from an administrator's **Override email address** (Users tab). A saved-search or CSV finding uses its own tier; an account finding needs one agreed address across the inventoried tiers, or the **authoritative identity tier** you choose. Missing, stale or conflicting addresses defer the email (at most 24 hours).
- **Who receives**: never a system account; human accounts by default; service and unclassified accounts only when you opt in.
- **Detection and clearing**: a new match opens an episode and sends one detection email. **Two consecutive complete evaluations** without the condition close the episode and send the "no longer detected" email (only after a sent detection, to the same recipient). Partial, unknown or learning evaluations never clear. A recurrence within *repeat hours* (24) does not send a new detection.
- **Baseline**: the first enabled evaluation records the current matches without sending them, unless **Include current detections once** is on.
- **Independent of operator triage**: resolving, dismissing, snoozing, grouping or assigning a finding does not stop or trigger user emails.
- **Content**: a common banner, introduction, footer, reply-to address and an HTTPS support link, plus per-policy subjects and bodies (*Policies → Edit → Email template*; every built-in policy is on except `unclassified_schedule/v1`; the privileged policies never email the account). Plain-text templates with the placeholders `{account}`, `{policy}`, `{search_name}`, `{app}`, `{environment}`, `{summary}`, `{action}` and `{window}`; subjects up to 256 characters, bodies up to 10,000.
- **Limits**: 50 emails per policies run (1–200), 3 per address per hour (1–20), 24 hours between repeat detections (1–168). One recipient per email.
- **Preview** and **Send test email** (to a test recipient, for one policy) use synthetic data and work while the channel is off.

Send email communication

Send email communication (finding and group menus, tenant administrators and power users) prepares a reviewed one-off email about a finding, a group member or a whole Finding group (up to 200 findings, one section per finding): the modal shows the findings, the recipient (which you may change for this email only), a full preview, and asks for explicit confirmation. It is sent once from a search process, never retried blindly: when the outcome is uncertain, check the inbox before sending again. It does not change the findings or the automatic notification state.

The UAM AI Agent

The **UAM AI Agent** reviews UAM findings with the evidence UAM collected; it replaces the UAM Findings Advisor of 2.4.18. Its modes, notes and decisions are those of every findings AI Agent: see *Findings AI Agents*.

- **Inspect** writes one current `[AI Agent]` note per finding (assessment, confidence, recommendation) and changes nothing. **Act** also applies the decisions the tenant allows.
- **Ask the AI Agent** (one finding, from its menu or the Investigate view), **Run a review now** (AI Agent tab) and the **scheduled triage** (hourly, off by default).
- The scheduled triage reviews at most *findings per run* findings, **one per Finding group**, skips a group reviewed within the review interval unless it changed, and never overrides an operator decision. Its selection uses the shared AI filters: by default only **critical and high** findings.
- The AI Agent tab shows the scheduled triage status and the decision log (kept 90 days).
- Requirements: AI enabled with a provider configured; for the scheduled triage, a tenant owner that is not `nobody` or `splunk-system-user` (Guardian `ai_automation_owner_is_system_user`). Launching,

deciding and reviewing need a TrackMe administrator of the tenant; tenant power users can read the settings and the decision log.

Setting (`vtenant_account`)	Default	Meaning
`ai_uam_agent_enabled`	0	Scheduled triage on or off.
`ai_uam_agent_mode`	inspect	Mode of the scheduled triage (`inspect` or `act`).
`ai_uam_agent_provider_name`	empty	AI provider; empty = the first available.
`ai_uam_agent_min_days_betw`	1	Days before a reviewed Finding group is reviewed again (unless it changed).
`ai_uam_agent_max_findings_`	10	Findings reviewed per scheduled run, one per group (1–50).
`ai_uam_agent_max_runtime_s`	3000	Time budget of one hourly run, in seconds.
`ai_uam_agent_allow_investi`	1	Act may mark a group investigating.
`ai_uam_agent_allow_dismiss`	1	Act may dismiss for a time — never a critical or high finding still observed.
`ai_uam_agent_dismiss_max_d`	7	Longest dismissal, in days (1–90).
`ai_uam_agent_allow_resolve`	0	Act may resolve a finding its policy no longer observes.
`ai_uam_agent_allow_classif`	0	Act may set the class (human or service) of an account without an operator class.
`ai_uam_agent_custom_instru`	empty	Plain-English instructions of the tenant administrators, followed by every run of the agent, asked or scheduled (at most 4,000 characters, line breaks kept).
`ai_automated_priority_filt`	critical,high	Shared: the finding severities the scheduled triage reviews.
`ai_automated_filter_expres`	empty	Shared: a filter expression on the finding (`policy_id`, `owner`, `app`, `account`, `subject_type`, `status`…).

These settings are edited in *Settings → AI Agent* of the workspace; *Manage AI Agents* shows the UAM AI Agent of each UAM tenant and edits its schedule, mode, provider, review interval, runtime and custom instructions. A permanent dismissal, an assignment, a severity change and any policy or configuration change are never available to the agent. See also *The AI Advisors*.

Custom instructions and Evaluate. The agent's custom instructions (*Settings → AI Agent → Custom instructions*, also in *Manage AI Agents automation* and the *Configure tenant* dialog) come last in its prompt, after the provider's custom prompt and, for scheduled runs, after the tenant's shared custom instructions (`ai_automated_custom_instructions`, scheduled runs only). They never widen what the agent may do: the allowed decisions and the checks TrackMe makes before every decision still decide. **Evaluate…** runs the agent in inspect mode on the unsaved draft (empty = without instructions), on one open finding or a review of the open findings (at most 10, the most severe first), optionally side by side with a run without custom instructions. It writes nothing — no note, no decision, no review record — and is recorded with the status `evaluation`, so a later act run never starts from it. See *AI Agents for findings*.

Collection health and Configuration Guardian

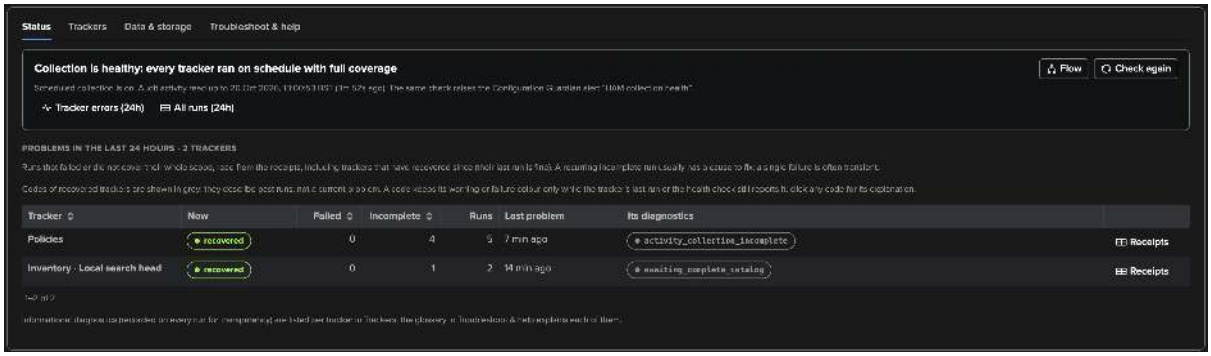
Collection health

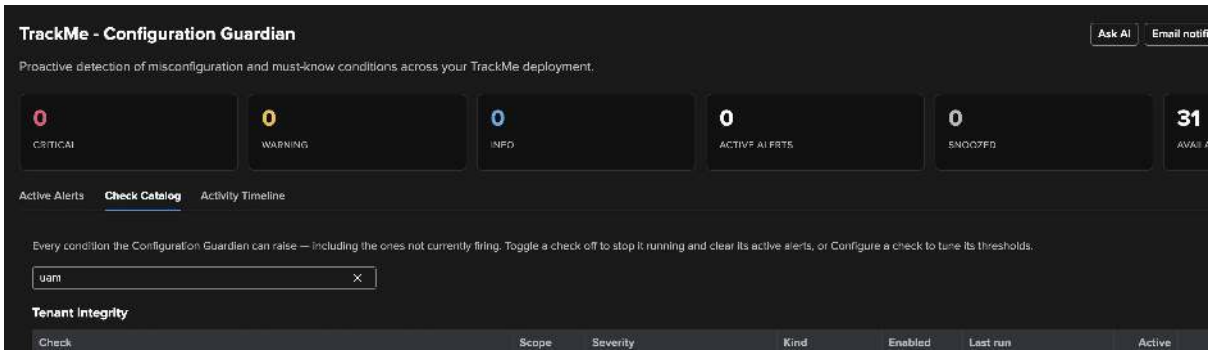
The *Collection health* tab is the first stop when something looks wrong. Everything that says something is wrong comes with its explanation, its evidence and its action, and every artefact UAM writes is named, located and one click away. Four views:

- **Status** — one row per tracker that needs attention, from the same assessment as the `uam_collection_health` Guardian check (so the page and the alert never disagree), plus the tracker's last receipt. Expanded: every reason with its meaning and where it was reported (run, dataset, tier, window), **What to do**, the facts of the last run (status, finish, consecutive failures, search job, report and its live schedule, lock), the policies a policies run could not complete (**Show why** opens the policy insight), the evidence shortcuts (**Receipts**, **Tracker log**, **Search job**, **Report**, **Scheduler log**) and **Run now / Recover**. Below: **Problems in the last 24 hours**, the failed and incomplete runs of each tracker from the indexed receipts, recovered or not (the backfill only while a plan is active or a problem is current). **Flow** opens a read-only topology of the collection pipeline — the tiers and the indexing layer, the trackers, the KV collections and indexes they write, the policies, findings and their consumers — coloured by health.

- **Trackers** — every tracker in one table: status (opens the last-run view), live schedule (read from Splunk, not the defaults), last run, last collected, duration (with the budget applied, see *Run budgets*), diagnostics, actions. Filters: name, family, status (needs attention, failed, running, not scheduled), tier. Expanded: the activity checkpoint and the windows of the last run with their outcome (acknowledged, split, resumed next run, retried) for Audit activity; the resource windows, re-reads and capacity for Resource usage; the scans and their re-reads for Account signals; the datasets of a catalog; the backfill plan and its controls. **Searches of the last run** (Audit activity, Resource usage, Account signals, Policies) lists every search the run sent, exactly as sent — audit windows with the clause that resumes them, recounts, re-reads, resource windows, scan chunks, each policy aggregation per evaluated window — with its time range, **Copy** and **Open in Search**; only searches the run really sent are listed, with a note when the settings changed since or the run read through a remote account. Custom policies have their execution health under the *Custom policies* family.
- **Data & storage** — what UAM writes for the tenant: the events per index and sourcetype (volume over 1 hour to 7 days, latest event, search), the KV collections (records, size, lookup definition, a record viewer with a KV query, what large collections are made of), the logs and every scheduled report with its live state, owner and next run.
- **Troubleshoot & help** — the **diagnostic glossary**: every code UAM can report, with *What it means*, *Where it comes from*, *Possible causes* and *What to do*; searchable (spaces match underscores), the codes the tenant reports now first, codes of earlier versions kept and marked. Then ready-made searches: tracker errors, failed or incomplete runs, every run, skipped scheduled reports, REST calls, who changed UAM.

The view and its filters, and the expanded row, survive a tab switch and the auto-refresh. A diagnostic code is a chip: its short meaning on hover, its full explanation on click; it keeps its warning or failure colour only while its condition is current, and a code of a recovered past run is drawn neutral and names the run it describes. The Status verdict answers in seconds; the 24-hour history and the record counts follow in a second request.





Configuration Guardian

The `uam_collection_health` check (tenant scope, **warning**, see *Configuration Guardian*) runs at every health cycle of each UAM tenant and on Guardian *Run now*. It is skipped for a disabled tenant, an unconfigured workspace or manual collection (which clears a stale alert). The alert names the tenant, lists its problems in the metadata and points at *Collection health*; it clears itself once the conditions are gone. Unreadable inputs leave an existing alert untouched.

Problem code	Condition
`report_missing` / `report_not_scheduled`	A report the tenant expects is absent, or not scheduled.
`no_run_recorded`	A report has no recorded run.
`tracker_stale`	The last run is older than three missed schedule opportunities (plus the listing budget for the catalog reports).
`repeated_failed_runs`	Three or more consecutive failed runs of one report.
`notification_delivery_failed`	Operator notifications are on and the last policies run failed to deliver some.
`user_notification_delivery_`	User notifications are on and the last policies run failed to deliver some.
`listing_overdue`	A catalog was last published more than twice its cadence plus a day ago.
`listing_incomplete`	The last listing or catalog traversal failed or did not complete.
`listing_queued_repeatedly`	A catalog of a tier waited for another listing of the same tier on three due runs in a row and listed nothing.
`checkpoint_stalled`	A checkpoint did not move for three schedules plus one window — the audit activity cursor, the resource usage cursor or an account signals scan. A checkpoint catching up, however far behind, is not stalled. The alert names the cause when one is recorded: a range being written over several runs, or a held range.
`activity_window_held`	An audit range has been held (its search keeps failing) for 15 minutes.
`window_reread_overdue`	A window read while some search peers did not answer has waited more than a day to be read again.
`window_recount_overdue`	A recount of closed ranges has waited more than a day past its time.
`findings_working_set_large`	One policies run can change more than 50,000 findings. Every finding is still evaluated.
`dead_job_lock`	A lock is held by a run whose search has stopped (use *Recover stopped run*).
`lock_owner_unverifiable`	A lock's owner cannot be verified (another member, or no search id).
`definition_capacity`	More custom policies are enabled than the tenant allows.
`schedule_drift`	A custom policy's scheduled report is missing, unscheduled or differs from its published revision.
`repeated_incomplete_runs`	Three or more consecutive incomplete runs of a custom policy.
`recovery_required`	A custom policy run outlived its time limit and needs a verified recovery.
`missed_schedule`	A custom policy missed its scheduled runs.

Tenant lifecycle

Disabling the tenant un-schedules its reports; enabling it reconciles them. Deleting a UAM tenant cancels a running backfill, disables and deletes its reports, lookups and collections, and refuses to run while a job lock is held (a provably stopped lock is recovered first; wait for or recover a live one, then retry). When provisioning is incomplete, the wizard result and the workspace offer **Repair UAM workspace**, which re-provisions the collections, lookups and reports idempotently.

Settings reference

The UAM configuration is one record (`uam_config` at creation, `config` of `POST uam/configure` afterwards), edited from the workspace *Settings* tab and saved at once with a revision check (an older draft is refused). Every limit is a tunable with a validated range, not a product cap, and every limit of a run is a time: there is no row limit. The seven record-count settings of earlier versions (`activity_max_records`, `activity_max_window_retries`, `resources_max_records`, `inventory_max_records`, `findings_max_records`, `backfill_max_records`, `backfill_max_retries`) are ignored: stored values disappear at the next save, and an API client that still sends them keeps working.

Section	Setting (key)	Default	Bounds and meaning
Environment	Search head tiers (`tiers`)	`["local"]`	1–10: `local` and / or remote account names, one indexing layer.
	Route account for indexing-layer searches (`account`)	`local`	One of the tiers; `local` when selected, else the first tier.
Activity scope	Include / exclude (`search_head_scope.include` / `.exclude`)	empty / empty	Up to 20 hostname patterns each (`*`, `?`); exclusions win.
Collection	Scheduled trackers (`scheduled`)	On	Off = collection only on request. Requires a tenant owner.
	Store SPL search (`store_search_text`)	**Off**	Index the SPL of reported searches in the search text index.
	Collect search resource usage (`resources_enabled`)	On	Required by the resource policies.
	Source version assumption (`source_version_assumptic`	On	Parse unverified emitters under the head's version, labelled assumed.
Indexes	`indexes.evidence` / `.findings` / `.receipts` / `.search_text`	empty	Empty = the tenant's summary index (evidence, receipts, search text) or notable index (findings).
Limits · activity	Activity window (`activity_window_seconds`)	300	60–3,600 s: the largest span one audit or resource window reads (a busier window is split and read in full).
	Activity overlap (`activity_overlap_seconds`	60	0–600 s, less than the window.
	Audit activity max seconds per run (`activity_max_seconds`)	270	30–900: the budget of one Audit activity run.
	Resource usage max seconds per run (`resources_max_seconds`)	270	30–900: the budget of one Resource usage run, capacity refresh included.
	Account signals max seconds per run (`signals_max_seconds`)	870	30–900: the budget of one Account signals run (half at most for the login scan).
	Data usage run budget (`data_usage_max_seconds`)	870	30–14,000: the budget of one Data usage run; a scheduled run ends before the next 15-minute slot, a manual run can use the whole budget.
	Activity policy window (`activity_policy_window_s`	900	300–86,400 s: the window of collected evidence the activity and resource policies evaluate.
Limits · inventory	Inventory page size (`inventory_page_size`)	500	50–500 records per REST page.
	Catalog collection every (h) (`inventory_reconcile_hour`	24	1–168: saved-search listing and the ten other families.
	Users and roles every (min) (`inventory_identity_minut`	60	15–1,440.
	Full listing budget (s) (`inventory_listing_max_se`	3,570	300–14,000 per run of a traversal; a longer traversal continues at the next run.
	Listing window start / length (`inventory_listing_start_` / `inventory_listing_window_hour`	1 / 5	0–23 / 1–24 (24 = any time), search head local time.
Limits · policies	Resource rules: minimum history (h) (`activity_min_history_hou`	24	0–720; the learning period of the resource policies.

continues on next page

Table 13 – continued from previous page

Section	Setting (key)	Default	Bounds and meaning
Limits · backfill	Backfill days (`backfill_days`)	30 (the wizard proposes none)	0–90 (0 = no backfill). The wizard always sends its choice; 30 is what a stored configuration without the setting reads.
	Backfill bucket (`backfill_bucket_seconds`)	300	300–3,600 s.
	Backfill max seconds per run (`backfill_max_seconds`)	3,570	600–3,570.
Account classification	Detection (`classification_detect_i` / `_service`)	On / On	Human and service detection.
	Rules (`classification_rules`)	none	Up to 100 ordered exact / wildcard rules.
	Manual classes (`account_classes`)	`nobody`, `splunk-syste` `admin` → system	Up to 10,000 accounts.
Data usage	Supply links (`data_usage.links`)	automatic	Per tier: one VOL tenant (`vol_tenant`) and up to 20 DSM tenants (`dsm_tenants`). A tier without an entry is matched automatically; an explicit entry, even empty, stays as set.
	Excluded apps (`data_usage.exclude_apps`)	`trackme`, `Splunk_SA_CI`	Up to 10 apps whose searches are shown, never counted as reads.
	Internal indexes (`data_usage.include_internal`)	Off	Show the _-prefixed indexes in the views by default.
	Account and sourcetype grain (`data_usage.per_account` / `.sourcetype_facts`)	On / On	Write facts per account and per sourcetype. Changes apply to new facts only.
	Resolution limits (`data_usage.limits`)	65,536 / 128 / 256 / 64	SPL excerpt characters (1,024–262,144), reference depth (1–256), resolution attempts per search (1–4,096) and nested subsearch levels (1–128). A limit reached leaves the search unresolved, never dropped.
Privileged users	`privileged.role_patterns` / `.roles_include` / `.roles_exclude` / `.capabilities`	`*admin*` / none / `trackme_admi` / nine capabilities	Up to 500 entries each; see *Privileged users*.
Notifications to operators	`notifications`	Off; new, reopened, severity raised, resolved; medium; digest	Up to 50 recipients.
Notifications to users	`user_notifications`	Off; human accounts; baseline	50 per run (1–200), 3 per address per hour (1–20), 24 h between repeat detections (1–168).

continues on next page

Table 13 – continued from previous page

Section	Setting (key)	Default	Bounds and meaning
Policies	`policies.<policy id>`	see the catalogue	Enabled, severity, exclusions, scope, thresholds, automatic thresholds, preview mode, bank holidays hold (`hold_on_bank_holidays`); the privileged policies' settings (`inactive_days`, `reference_accounts`, `reference_lookups`, `report_stale`, `include_splunk_operated`).

Stored elsewhere:

- **AI Agent —** the `ai_uam_agent_*` fields of the tenant (see *The UAM AI Agent*).
- **Assignment —** up to 100 assignment groups and 100 routing rules; group emails off by default.
- **Finding actions —** up to 50 definitions and 100 automation rules.
- **Custom policies —** their own definitions (see *Custom policies*).
- **Per viewer —** *Exclude TrackMe activity* on the Users tab (browser storage, off), the Policies view mode (Cards, Compact cards, List), and the three Tenant Home display preferences (Collection monitors, Single stats, Charts).

The tracker schedules, the one-day retention of old catalog generations and the 90-day retention of the AI Agent decision log are fixed.

Roles and licence

Role	Access
TrackMe administrator of the tenant (`trackmeadminoperations` and one of the tenant's admin roles; Splunk `admin_all_objects` also passes)	Everything: the workspace in Tenant Home, triage and grouping, configuration, policies and custom policies, search text and live definitions, the knowledge-object catalogs, collection control, the AI Agent.
Power user of the tenant (`trackmepoweroperations` and one of the tenant's power roles)	**REST API access only** (Tenant Home shows the UAM workspace to tenant administrators): the workspace, findings, Finding groups, counts, activity, accounts, privileged users, inventory, saved search view, policy evaluation and history, tracker history, AI Agent settings and decision log; **Send email communication**; findings actions (run an action, email action, set an assignment).
Read-only user (`trackme_user` only)	No UAM operation.

Licence: any licence beyond the Foundation edition — Enterprise, Unlimited, the trial and Developer licences. Foundation and the Foundation Trial are refused (`uam_not_available_in_foundation`). On an expired licence the workspace stays readable and changes are refused; scheduled collection and custom policy runs continue through the read-only and escalation stages of the licence enforcement ladder and stop with the tenants at the stop stage.

REST endpoints

Resource group `uam`: `/trackme/v2/uam/<operation>` (browse it in **TrackMe REST API Reference**). Every operation is a POST with `tenant_id` in the body, except `GET workspace` and `GET agent_settings`, and answers `describe=true`. "Power" marks the operations also open to tenant power users; the others need a TrackMe administrator of the tenant. Every `earliest` / `latest` accepts what the Splunk time picker produces.

Area	Operations
Workspace reads (power)	`workspace`, `finding_counts`, `activity`, `timeline`, `accounts`, `account`, `privileged` (the privileged users, with `activity: true` for their activity), `inventory`, `saved_search`, `tracker_history`
Data usage (power)	`data_usage` (`view`: `map`, the default, with the value view, or `timeline` for an index or an account), `data_usage_index` (one index), `data_usage_objects` (the knowledge objects), `search_jobs_cancel`; `data_usage_links` (the supply candidates, administrators). The reads accept `search_jobs: "start"` and then the returned jobs to poll; without it they answer synchronously within the safety timeout
Findings and groups	`findings`, `finding_groups`, `finding_group` (power); `triage` (decisions on up to 500 Finding groups, `expected_versions`), `group_members` (`merge`, `add`, `remove`, `ungroup`), `classify` (manual classes of 1 to 200 accounts)
Policies	`policy_evaluation`, `policy_history` (power); `preview` (evaluate a draft configuration without writing)
Custom policies	`custom_policies`, `custom_policy`, `custom_policy_health`, `custom_policy_write` (create, save, restore, publish, enable, disable, archive, delete, delete a version), `custom_policy_execution` (preview or run: dispatch, status, cancel, recover)
Notifications	`notification_test` (operators or users channel); `user_notification_preview` (power); `user_contact_override`; `communication_preview`, `communication_send` (power)
Inventory and catalogs	`saved_search_definition` (Fetch definition), `knowledge_objects`, `search_text` (Fetch SPL)
Collection	`configure` (full configuration with its `revision`), `collect` (`job`: `all` or one job, optional `tier`), `backfill` (`pause`, `resume`, `cancel`, `restart`), `recover`, `collection_health` (optional `parts`: `status` and / or `history`), `storage_counts`, `searches` (the exact searches of a tracker's last run, `job` = `activity`, `resources`, `signals` or `policies`, or of one policy, `policy_id`)
AI Agent	`agent_settings` (GET power, POST saves), `agent_decisions` (power), `agent_findings`, `agent_evidence`, `agent_decide`, `agent_review`, `agent_run`; job control in `/trackme/v2/ai_findings_agent/{agent,agent_status,agent_cancel}`
Findings actions	`/trackme/v2/findings_actions/*` with `component=uam`

The tenant is created with `POST /trackme/v2/vtenants/admin/add_tenant` carrying `tenant_uam_enabled=true` and a `uam_config` object (the wizard sends `tiers`, `scheduled`, `backfill_days` and `search_head_scope`); see *Creating a splk-uam tenant*.

Upgrading to 2.4.22

- **Data usage** arrives with its own report, `trackme_uam_data_usage_tracker_tenant_<tid>`, created on an upgraded tenant by the health tracker's schedule reconciliation, without a configuration save. Its history starts at its first run: the searches before the upgrade are not read. Until the collection covers the selected period, or seven days of it, the views say *so far* and call no index unread.
- The facts' index-time fields (`uam_data_*`) are extracted by the same TrackMe ingest rule as the other UAM dimensions: on a distributed deployment, the instances that parse the UAM log need the upgraded app and its restart. Until then the map falls back to a slower raw read.
- Searches the UI starts on a UAM tenant run as the caller's own search jobs, polled until done: a long range or a heavy account is read completely instead of being cut after 45 to 60 seconds.

Upgrading to 2.4.21

- **Indexing-layer trackers**: the health tracker's report check (every 5 minutes) creates the Resource usage and Account signals reports of an existing tenant, with its owner, ACL and schedule, before the Guardian check of the same cycle (a report it just created is never reported missing). The Activity report keeps its definition and runs the audit only. Resource usage continues from the existing resource cursor, the scans from where the earlier runs stopped; until the first Resource usage run, the policies read the resource usage of the last activity receipt. No schema migration, no settings save.
- **Budgets**: a stored budget still at its earlier default (200 s for the activity, 3,600 s for the full listing, 3,000 s for the backfill) moves to the new default (270 s, 3,570 s, 3,570 s); a value an administrator chose is kept. The report check also refreshes a tracker description left by an earlier build.
- **Retired limits**: the seven record-count settings are ignored and dropped at the next save.
- **Evidence**: correlated executions are no longer indexed, and the UAM events carry fewer fields (see *The evidence*). Rewrite your own searches on `trackme:uam:execution`. The index-time `uam_*` fields of audit and resource records now follow their sourcetype: the rule takes effect at the splunkd restart, and the earlier rule still applies until then.
- **Backfill**: the wizard proposes no backfill and new configurations use 5-minute buckets; an existing tenant keeps its bucket size until you change it.
- **Catalogs**: older catalog generations are deleted after 30 minutes (a day before), which removes many KV records on a deployment with many users. The users catalog also collects `last_successful_login`, kept outside the compared fields (a login is never an inventory change).
- **Privileged users**: the tab and *Settings → Privileged users* appear with the release defaults; the four privileged policies are off, and a change policy turned on starts from a baseline.

Upgrading from 2.4.18

- **Finding groups**: the first policies runs after the upgrade consolidate the existing findings into Finding groups (up to 5,000 findings per run; an account is never split). Triage decisions, notes and history are kept.
- **Inventory**: each tier gets its `listing` and `kos_*` reports. The first saved-search listing runs at the next hourly wake, then on the catalog cadence. The users and roles reports move from 5-minute to 15-minute wakes unless an administrator changed their schedule. *Inventory max seconds*, the fallback listing cadence and the delta object limit of 2.4.18 have no effect any more.
- **UAM AI Agent**: the UAM Findings Advisor report is removed; an enabled advisor's settings (enabled, mode, provider, review interval, runtime capped at 3,000 seconds) are carried to the UAM AI Agent once.
- The indexed `uam_*` dimensions take effect after the splunkd restart of the upgrade.

Troubleshooting and known limits

Start with **Collection health** (each tracker's last run, diagnostics and recovery) and the Configuration Guardian check `uam_collection_health`. The technical logs carry outcomes only, with `tenant_id` index-time extracted: `index=_internal sourcetype=trackme:custom_commands:trackmeuam* tenant_id=<tid>` (`trackme_uam_tracker.log`, `trackme_uam_inventory.log`, `trackme_uam_activity.log`) and `index=_internal sourcetype=trackme:rest_api tenant_id=<tid>` (`trackme_rest_api_uam.log`). `trackme_uam_events.log` is the evidence, not a troubleshooting log.

Symptom	What to check
The inventory is complete but the Activity tab stays empty	The route account cannot read `_audit` on the indexing layer (roles on the remote deployment), the search heads keep their internal indexes local, or the activity scope excludes them. Check the activity receipt in *Collection health* and run `index=_audit action=search` as the route account.
Data usage calls nothing unread, and says *so far* with a date	The collection is younger than the selected period (or than seven days): an absence of reads is claimed from that date. The coverage card of the view gives it.
Data usage shows *Partial coverage* or no volume	The coverage card names the cause: a collection run that did not complete (see the Data usage tracker in *Collection health*), an index catalog that could not be read, facts without the rollup key (the parsing tier lacks the upgraded app), or no VOL / DSM tenant linked to the tier (*Settings → Data usage*).
Most audit lines are unparsed (`unsupported_source_contract`	The emitters' version could not be verified and *Source version assumption* is off: turn it on.
No service account is ever detected	The `_audit` trail is JSON-only (the automation scan reads the legacy key=value format), or its retention is under seven days. Classify the accounts by rule or by hand.
The resource policies say *learning* for days	Less than 24 hours of resource history (*Collect search resource usage* off, or `_introspection` not forwarded), or no capacity record (see the activity receipt).
A policy opened a finding during a bank holiday	The policy keeps opening findings (*During bank holidays*), the tenant does not hold new findings (*Manage Bank Holidays → Tenants & settings*), the holiday is not in the tenant's calendars, or the finding was already observed. The policy insight and the policies receipt say what the run held, or that the holidays could not be read.
Peak memory reads *not measured*	No introspection sample caught the search (short pooled searches on Splunk Cloud). CPU still comes from Splunk's completion accounting.
Fetch SPL shows a reason instead of the text	`not_in_audit`: the record aged out of `_audit`; `account_access_denied`: your roles are not granted the route's remote account; `access_denied`: the environment refused your search. *Store SPL search* keeps the text in an index you control.
A finding keeps coming back after *Resolve*	Expected: the condition persists and the policy reopens a resolved finding. Fix the cause, add an owner / account or app exclusion on the policy, or **dismiss** it.
A CSV finding names the recorded owner, not the writer	No supported audit evidence of the write since monitoring began (or the tier's members could not be discovered: see the policy evaluation's writer diagnostics). The owner fallback is by design.
The Users tab shows a class the settings do not	The class is *detected*; a manual class or a rule always wins, and a detection can be switched off.
Collect now answers that a run is in progress	A scheduled run holds the lock; wait for its receipt. If no run is active, *Recover stopped run* (after five minutes, same search head only).
The backfill reports *waiting for the first activity run*	Normal until the Audit activity and Resource usage trackers set the live start; the backfill plans itself at its next hourly slot.
A custom policy does not run	Check *Collection health → Trackers*, family *Custom policies*: published and enabled, *Scheduled trackers* on, a tenant owner set; the Guardian names `schedule_drift`, `missed_schedule` or `recovery_required`.
The activity checkpoint lag keeps growing	The audit trail grows faster than one run reads it within its budget. Restrict the **Activity scope** to the search heads you monitor (it is applied in the base searches); *Searches of the last run* on the Audit activity tracker shows each window and its outcome. Nothing is dropped meanwhile: the next runs continue.
Audit activity says *Range held*	The search of one range keeps failing: it is retried, then halved to isolate the failing span, never skipped. Open the search job and the tracker log from the Status row; a range held for 15 minutes raises `activity_window_held`.
Windows to read again (informational)	Some search peers did not answer while a window was read. The window is read again once every peer answers, writing only what was missing; after a day of waiting, `window_reread_overdue` is raised: check the peers' state.
The Duration of a run says its budget was capped	The scheduler dispatched the run late (a busy scheduler): the run ended 30 seconds before the next scheduled run so that none is skipped, and the next runs continue through the checkpoint. Frequent caps point at scheduler load on the search head.

Known limits.

- One indexing layer per tenant: keeping the tiers on one layer is the administrator's responsibility.
- A saved search with the same name, app and owner on two tiers shares one set of run figures in the saved search view (resource records carry no tier).
- The knowledge-object catalogs reflect what the collection account can see, and a traversal is not an atomic snapshot.
- Several tenants monitoring the same tier each pay their own collection cost.
- Activity policies judge 15-minute windows of collected evidence by default; inventory policies follow the catalog cadence (a daily saved-search listing by default).
- Custom policies map to accounts and saved searches only, and their findings are not notified.
- A line that becomes searchable more than about ten minutes after its range closed, with no Splunk message, is not recovered; the login and automation scans, the resource usage and the backfill are not recounted. Rows recovered by a re-read after their policy window was evaluated are not evaluated again by the activity policies.
- A one-second range holding 500,000 audit events or more is held.
- A single Splunk call that blocks on a saturated instance is not interrupted inside a run, so such a run can last longer than its budget.

See also

- *User Activity Monitoring* — the task-oriented guide.
- *Findings, Finding groups and findings actions* — findings, Finding groups, assignment, notifications, findings actions and the AI Agents.
- *Who is doing what on your Splunk? User Activity Monitoring (UAM)* — the end-to-end walkthrough.
- *Creating a splk-uam tenant* — the creation wizard and the REST flags.
- *Configuration Guardian* — the Configuration Guardian.
- *Remote Splunk deployments* — remote deployment accounts (the remote tiers).
- *The AI Advisors* — AI Advisors and Agents.

8.3.16 Data usage reference: how each claim is processed

This page is the reference for the **Data usage** tab of a User Activity Monitoring tenant (see *the Data usage section of the UAM guide*). For each kind of claim a search can make about an index, it shows how Data usage records the claim, how the map interprets it when you read the view, worked examples and an explicit list of what it cannot establish.

The same text and diagrams are in the product: every **How this works** link of the Data usage tab opens them, and the **Explain** list selects the topic. The collection behind the tab (its report, facts, limits and REST operations) is described in *Data usage in the UAM in-depth reference*.

To see the facts behind a view, use **Open in Search** in the three-dots menu of any tile, chart or table of the Overview, Value & cost, Knowledge objects and Evidence views. It opens the search over the recorded facts that view reads, with a note on what only the map adds when you read the view: the catalog expansion of wildcards, the role scopes and the VOL volumes.

The examples use generic index and account names (`web`, `fw_*`, `hr_payroll`, `analyst`). Splunk's role search behaviour described here was measured on Splunk Enterprise 10.6 and Splunk Cloud 10.7.

How Data usage works

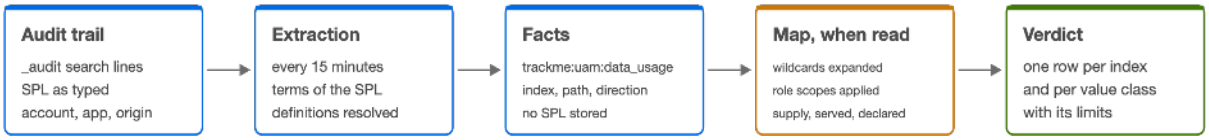
What it means. This view connects known indexes to searches UAM observed. It helps you investigate usage and identify data ingestion worth reviewing.

Where it comes from. The Inventory tracker lists indexes for each monitored search head tier. The dedicated Data usage collector reads audit observations and extracts index references from search text, resolving supported macros, event types, saved searches and data models from collected definitions. Explicitly linked VOL and DSM tenants contribute positive ingestion observations. The map compares these with retained search facts.

What it cannot establish. A reference does not prove that an index returned events, that results were consumed, or that an index has business value. Counts cover the selected period and scope, not all time. Sourcetypes are best effort. The map relies on the UAM contract: every search head tier of a tenant reads the same indexing layer, so one index name is one index. Watch a separate Splunk deployment with its own tenant.

What to do next. Start with Overview, investigate an index in Index activity, or use Value & cost to weigh each index's volume and cost against its reads: indexes no one accessed, indexes only wildcards reach, and the least read for their volume. Check Evidence and Collection health before acting on an absence.

From an audit line to a verdict



Splunk writes an audit record for every search it runs, with the account, the app, the dispatch origin (provenance) and the search text exactly as it was typed. It does not record which indexes the search read. Data usage therefore works from what the search asked for: its index claims.

A dedicated Data usage collector reads the search lines of the audit trail on its own schedule (every 15 minutes by default). It reads the SPL of each search, splits it into its pipeline stages and subsearches, and extracts the index and sourcetype terms of each stage. Macros, event types, saved searches and data models are resolved from the knowledge object catalogs that UAM publishes for the same search head tier. The SPL itself is never stored in the facts; it is held in memory while the batch is extracted.

A search is counted once, by rule: at its grant on the legacy audit format (Splunk writes it when the search starts), otherwise at its structured completion. A canceled or failed search still counts: it ran and read its sources. A search Splunk refused after granting it (a concurrency or disk quota) is not counted. Within one audit batch, each index named by a search receives one unit of weight, whatever the number of paths or sourcetypes that name it.

The collector writes facts (sourcetype trackme:uam:data_usage) to the tenant's evidence index: per window, tier, emitter, account, app, origin, path, index, sourcetype, direction and scope, with their counters as index-time fields. The map reads these facts through one tstats over those fields, so a long period never reads raw events for its counts.

Everything that depends on the current state of the deployment is applied when the map is read, not when the facts are written: wildcard patterns are expanded against the index catalog, role scopes come from the current users and roles catalogs, and incoming data, volume and declarations come from their latest publication. This is why a change of roles or a new index changes the view of past facts.

The map has one row per index for the whole indexing layer: every search head tier of a tenant searches the same indexers, so the same index name on two tiers is one index. A tier filter scopes the facts and the catalog shown; the verdict that decides a review candidate is taken on every tier together.

Limits

- An index claim is what the search asked for, never proof that the index returned events, that the results were looked at, or that the data has value.
- The figures are index claims, not distinct searches: one search that names three indexes counts once on each of them.
- A collector batch retried after a write failure cannot be told apart from the first write by the indexed counts; its receipt reports it.

- Older facts keep the interpretation they were written with: a change of the extraction or of the catalogs does not rewrite them, except for what the map applies when it is read.

Every kind of claim at a glance

Each line of the table below is a kind of claim a search can make about an index, with what Data usage records for it and what it does to the index's verdict. Only two kinds establish that an index was searched: a named source read on the search head side, and a named term in the search the indexers received. Everything else is context: it is shown, never counted as use.

Claim	Example	Recorded as	Verdict effect
Named read	`index=web`	A read of web, path literal	Searched · Read by name
Wildcard reference	`index=fw_* · index=*`	The pattern; expanded when the map is read, to the known indexes its reader could search	Indirect only · Wildcard only
Definition	`` `fw_idx` · eventtype=x · | savedsearch r · | tstats … from datamodel=M ``	What the definition names, path macro:… / eventtype:… / savedsearch:… / datamodel:…	As the indexes it yields: named or wildcard
Denied by roles	`index=hr` when the roles do not allow hr	Counted apart as a denied search	None: never a read
Index-less search	`sourcetype=nginx status=500`	Unresolved, path role_default; the role scope shown as context	None: never attributed
Late filter	`… | stats count by index | search index=web`	A late reference of web	None
Negation	`NOT index=web · index!=web`	Nothing counted for web	None
Write	`| collect index=summary_web`	A write reference of summary_web	None
Writing alert action	A saved search's summary indexing or Log event action targets summary_web	A write reference of summary_web on each run	None
Indexer confirmation	The search the indexers received names web	A served fact, never added to the search head counts	Searched when the search head side has no named read
Declaration	A saved search definition names web	A declared reference of web	None: a definition is not a run
Incoming data	VOL or DSM saw web receive data	Receiving, with volume and cost from VOL	None: decides which indexes are candidates
Excluded app read	A search of the trackme app names web	Shown, never counted	Excluded apps only, when nothing else reads it

Limits

- The table describes the default view: internal indexes hidden and the configured apps (TrackMe and the CIM acceleration app by default) excluded.

Named reads

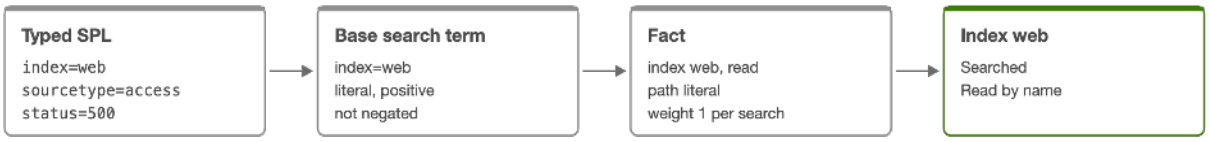
What it means. Source reads count observed search claims that name the index as a source, directly or through a supported definition. They describe the search request, not bytes read or returned events.

Where it comes from. Audit activity extracts source terms such as `index=example_web`. Index totals are deduplicated within each audit batch; path and sourcetype rows describe joint references and may overlap.

What it cannot establish. Do not add path or sourcetype counts to obtain a unique-search total. Older partial claims and later completions can remain in separate collection windows. Last searched is the latest named source claim in the selected evidence.

What to do next. Open Detail to see the account, app, source path and origin together. Fetch SPL examples to inspect the actual retained search text.

A literal index term



A named read is a positive, literal index term in the base search of a search or of one of its subsearches. Every form Splunk accepts is read: `index=web`, `index="web"`, `index::web`, index = web and `index IN (web, auth)`, whose members are one read each. Index names are compared lower-cased, as Splunk stores them.

The commands whose terms select data are the search itself and search, where, tstats, mstats, mcatalog, metadata, dbinspect and eventcount. Their index terms count, so | tstats count where `index=web` and | metadata type=hosts `index=web` are named reads of web.

The extraction reads syntax; it does not evaluate Boolean logic. `index=fw_emea` OR `index=web` records a read of each index. A term inside a subsearch (append, join, a […] argument) is a source term of that subsearch and counts as a read too.

When a search names exactly one literal index in its base search and its extraction is complete, the sourcetypes it matched are paired with that index from the completion's per-sourcetype event counts. With several indexes, a wildcard or a subsearch, no index and sourcetype pair is invented.

Last searched is the latest time a named read of the index was recorded in the selected evidence. Searches counts named reads; Completed counts the reads whose completion was recorded.

Worked examples

Search or case	What Data usage records
`index=web source-type=access_combined status=500`	One named read of web. The completion's event count for access_combined is paired with web.
`index IN (fw_emea, fw_amer) ac-tion=blocked`	One named read of fw_emea and one of fw_amer.
`| tstats count where index=web by sourcetype`	One named read of web: tstats selects data with its where clause.
`index=web [ search index=auth ac-tion=failure | fields user ]`	Named reads of web and of auth (a subsearch's base search is a source).
`index=fw_emea OR index=web`	Named reads of both: the extraction does not simplify Boolean SPL.

Limits

- A named read is a claim from the SPL. A search for an index that held no event in its time range still counts; so do metadata, dbinspect and eventcount, which read no raw event.
- A search whose retained SPL was truncated never yields a named read: its missing suffix could negate a term it kept. It is counted as unresolved.
- A named index the reader's roles do not allow is not a read: see Role search restrictions.
- Counts are per audit batch and per index, not a deployment-wide count of distinct searches.

Wildcard references

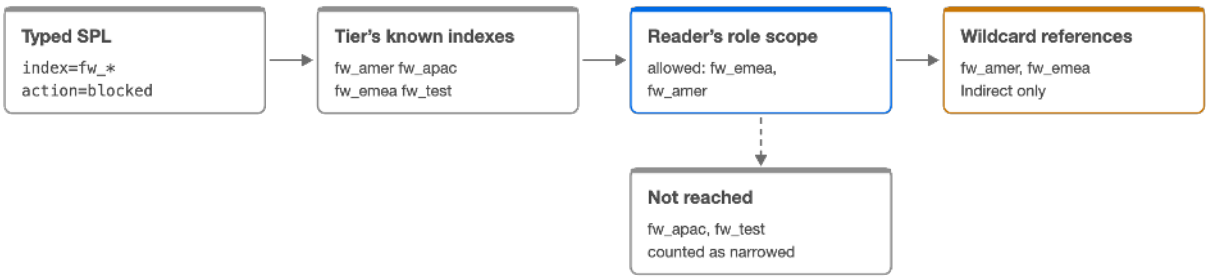
What it means. The index matches a wildcard source term, but UAM has no named source read for it in this selection.

Where it comes from. For example, `index=example_*` is expanded against the collected index catalog for the same search head tier, then narrowed to the indexes the search's account could search: its roles' allowed indexes and the index terms of their search filters. Matching indexes receive indirect references.

What it cannot establish. A broad `index=*` search can reference many indexes without proving that each supplied events or useful results. A field search filter (for example `sourcetype=…`) can still empty an allowed index. Indirect references are not named reads and are not a non-use verdict.

What to do next. Review the original SPL and its owner before judging value. Compare with ingestion volume and operational requirements outside this view.

Expansion, then narrowing to the reader



A wildcard term (`index=fw_*`, `index=*`, `index IN (fw_*, web)`) is kept in the facts as the pattern itself. The collector never knows which indexes matched it, and does not try.

When the map is read, the pattern is matched against the indexes known for the same search head tier: the tier's index catalog, listed by the Inventory tracker, plus any index a linked VOL or DSM tenant or a published knowledge object names for that tier. A new index is matched as soon as one of these sources names it: an index only the catalog will know is matched after the next catalog listing, while one a linked tenant or a published object already names is matched right away.

The matches are then narrowed to the reader's role scope, per account: the same pattern typed by two accounts can reach different indexes. Every remaining match receives one wildcard reference (the Indirect column); the matches the reader could not search are left out and counted in Evidence as wildcards narrowed.

A wildcard is a reference, not a read, because Splunk does not report which of the matched indexes held events in the searched range. Splunk searches every allowed index that matches; whether one of them supplied data is not in the audit.

Every pattern, typed or from a role (allowed, disallowed, filter and default indexes), is matched with Splunk's index pattern rule: case-insensitive, * as the only wildcard, and * never matching an internal index (a name starting with _) unless the pattern itself starts with _. Internal indexes are hidden from the view unless Include internal is on; with it on, `index=*` still does not reach them, while `index=_*` does.

When the source tier of a fact is unknown (its emitter does not map to a single tier), the pattern is kept as recorded, unexpanded, and never assigned to another tier's indexes.

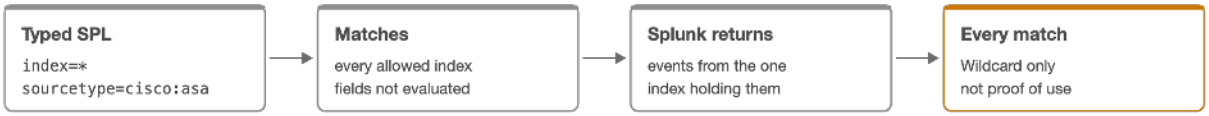
Worked examples

Search or case	What Data usage records
`index=fw_* action=blocked`	By an account allowed fw_emea and fw_amer: wildcard references on fw_emea and fw_amer only. fw_apac and fw_test are not reached (one wildcard narrowed, two matches removed).
`index=* error`	By an account whose roles allow *: a wildcard reference on every non-internal index of the tier.
`index=fw_* | search index=fw_emea`	Wildcard references from fw_*; fw_emea is also a late reference, which is not a read.
`index=_* | stats count by index`	With Include internal on, by an account whose roles allow _*: wildcard references on the internal indexes. `index=*` would reach none of them.

Limits

- An account whose role scope cannot be established keeps every match in the catalog (see Role search restrictions).
- The expansion is bounded per request (3 CPU seconds of the reading thread, 200,000 joint rows). Past the bound the answer is partial (expansion_limit) and candidates are withheld.

Why Wildcard only is never a verdict of non-use



An index reached only by wildcard references has the verdict Indirect only and the value class Wildcard only. It is not proven used: a broad pattern reaches it whether or not it held events. It is not proven unused either: the search may have returned its events. So Data usage keeps it as a review candidate, in a group of its own, and never folds it into No reader.

The role scope is an upper bound. A field term of the search or of the role search filter (`sourcetype=…`, `host=…`) can leave an allowed index without any event, and only the events would show it. Data usage never evaluates field terms per index.

Worked examples

Search or case	What Data usage records
`index=* sourcetype=cisco:asa`	A wildcard reference on every allowed index, although only the indexes holding cisco:asa events returned anything.

Limits

- Accesses (the Overview weight) add wildcard references to named reads: one `index=*` search adds one access to every index it reaches. Read the Wildcard only share before judging a much-accessed index.

Role search restrictions

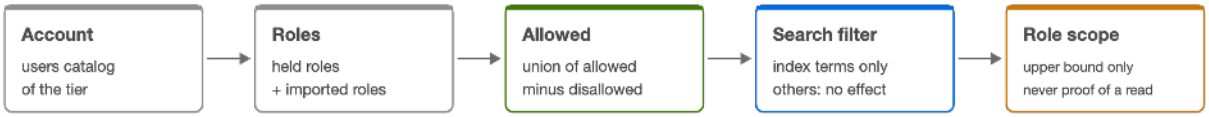
What it means. Splunk searches only what the account's roles allow. A wildcard reaches only the indexes its account could search, and a search naming an index its roles do not allow returned nothing: it is counted as denied, never as a read. A search naming no index can reach the indexes its role search filter names, else its default indexes, within what its roles allow: shown as context on the unresolved searches, never attributed (an index-free command reaches none).

Where it comes from. The published users and roles catalogs of each search head tier: the roles an account holds and imports, their allowed and disallowed indexes, and the index terms of their search filters. Splunk enforces these silently: the audit keeps the search as typed.

What it cannot establish. This is what the roles allowed, an upper bound: a field filter such as `sourcetype=…` can still empty an allowed index. The current roles apply to the whole period. An account or role missing from the catalogs, or a tier whose listing saw only its own account, keeps every wildcard match.

What to do next. Check Evidence for how many accounts were resolved. Before acting on an index reached only by wildcards, compare its readers' roles with the index; a denied read is an index the account asked for and could not read, even when the same search read other, allowed indexes.

How an account's role scope is built



Splunk restricts every search to its account's roles and records none of it: the audit keeps the SPL as typed, the search head appends the roles' search filter to the search it sends the indexers, and the indexers apply the allowed indexes without a message or an error. Data usage rebuilds the restriction from the users and roles catalogs UAM publishes for each search head tier.

The scope of an account is: the union of the allowed indexes of its roles and of every role they import (followed transitively); minus every disallowed index, which wins over any allowance; narrowed by the index terms of the roles' search filters.

Search filters combine as Splunk combines them. An empty filter is ignored, and a role without a filter does not open what another role's filter closes. Filters of several roles are joined with OR. Only positive index terms of a filter narrow the indexes (`index=…`, `index::…`, `index IN (…)`); a negation, a field term or a filter that cannot be read restricts no index.

Patterns are matched as Splunk matches index patterns: case-insensitive, * only, and * never matches an internal index.

The scope is computed when the map is read, from the catalogs as they are now, and applies to every fact of the period. Nothing is written on the facts.

Worked examples

Search or case	What Data usage records
Roles: analyst (allowed fw_*, disallowed fw_test) imports base (allowed web)	Scope: fw_* and web, without fw_test.
Roles: analyst (allowed *) with search filter `index=fw_emea` OR `index=fw_amer`	Scope: fw_emea and fw_amer.
Roles: analyst (allowed *, no filter) and auditor (allowed *, filter `index=audit_*`)	Scope: audit_* only: the unfiltered role does not lift the other role's filter.

Limits

- An upper bound: a field term of the filter (`sourcetype=nginx:*`) can still empty an allowed index.
- Splunk's roles endpoint does not return srchFilterSelecting, so every filter is read as selecting (joined with OR). An eliminating filter (joined with AND in authorize.conf) is widened, never narrowed.
- No scope is established, and every wildcard match is kept, when: the audit did not report the account; the account is not in the users catalog; one of its roles is missing from the roles catalog or was listed before this version; the tier's users and roles listing saw only its own account; the catalogs are not published yet; or the roles catalog holds more than 5,000 roles.

- The current roles apply to the whole period: a role changed last week rewrites the view of the weeks before. A role change also takes a few minutes to reach the indexers (knowledge bundle replication), during which Splunk applies the previous definition.

Denied by roles



A literal index that the reader's role scope does not allow is counted as a denied search on that index (the Denied by roles tile of the index detail), with the accounts that asked. It is never a read and changes no verdict: Splunk served nothing.

Only reads are checked. A write (collect) and a late filter keep their own meaning. An indexer confirmation of a denied index is not counted either.

Worked examples

Search or case	What Data usage records
`index=hr_payroll`	By an account whose roles allow fw_*: one denied search on hr_payroll, no read.
`index=hr_payroll OR index=fw_emea`	By the same account: a denied search on hr_payroll and a named read of fw_emea.

Limits

- A denial follows the roles' definition. It is not corroborated by the search's completion (a zero scan count is also what an empty time range gives).
- A denied search means that the denied index term returned nothing for its account, not that the whole search failed: the same search can read other, allowed indexes (the second example). It is a hygiene signal, never a use of the denied index.

Role filter terms in what the indexers received

Splunk appends the role search filter to the search the indexers receive. When the filter names indexes, the indexers' log shows index terms the account never typed. A served index term equal to one of the reader's own filter patterns is dropped as the role's, not the search's, and counted in Evidence as served filter terms.

Worked examples

Search or case	What Data usage records
Typed: error · role filter: `index=fw_*` · indexers received: litsearch error (`index=fw_*`)	The served fw_* is the role's filter: dropped. The typed search stays index-less.

Limits

- The match is exact, pattern for pattern, after lower-casing.

Macros, event types, saved searches and data models

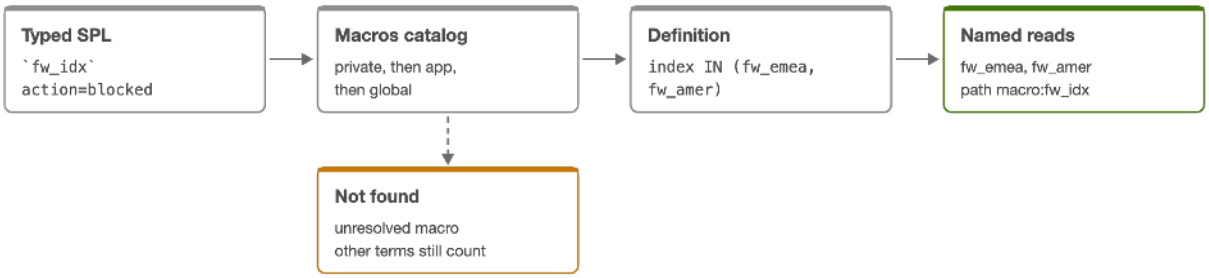
What it means. A search can name its indexes through a knowledge object. Data usage resolves the object from the catalogs of the search head tier and records the indexes its definition names, with the object as their path.

Where it comes from. The macros, event types, saved searches and data models catalogs UAM publishes for each tier. The collector pins the published catalog generations at the start of a run, so a run reads one consistent version of them.

What it cannot establish. An object the catalog cannot see (a private object of another owner, an app not listed) stays unresolved. Tags and custom commands are not expanded. The definition is the one published when the facts were written.

What to do next. Open the index detail's paths to see which objects lead to an index, and the Knowledge objects view for what each object declares and ran.

Resolution from the catalogs



A macro call (`` `name` `` or `name(arg1, arg2)`) is looked up by name and argument count, with Splunk's namespace order: the search owner's private object in the search's app, then an object shared in that app, then objects shared globally from other apps. Two candidates of the same rank leave the macro unresolved. Arguments replace their exact arg tokens. The definition is then extracted like typed SPL, recursively.

`eventtype=name` is resolved to the event type's search. | savedsearch name, | loadjob saved-search="owner:app:name" and | from savedsearch:name are resolved to the saved search's SPL; the loadjob owner syntax does not open another owner's private object.

A data model reference (| tstats … from datamodel=Model.Object, | datamodel Model Object search, | from data-model:Model, | pivot Model Object) is resolved to the root constraints of the model: the constraint of the object's own root dataset (BaseEvent, BaseSearch or BaseTransaction), or of every root when no object is named. Child constraints never widen the scope.

The indexes found are named reads or wildcard references as they would be typed. Their path is the first object of the chain: a macro calling another macro is recorded as macro:<first>.

A macro used inside a command whose arguments are not a search (macro_context), a missing object, a cycle, or a resolution beyond its limits is recorded as an unresolved part of the search; the other index terms of the same search still count.

Worked examples

Search or case	What Data usage records
`` `fw_idx` action=blocked (definition: index IN (fw_emea, fw_amer)) ``	Named reads of fw_emea and fw_amer, path macro:fw_idx.
`eventtype=web_errors (search: index=web status>=500)`	A named read of web, path eventtype:web_errors.
`| tstats count from datamodel=Web.Errors (root constraint: index=web sourcetype=access_*)`	A named read of web, path datamodel:Web.
`` `undefined_macro` index=web ``	A named read of web, and an unresolved part macro:undefined_macro.

Limits

- Resolution limits (Settings → Data usage): depth 128 (1 to 256), 256 resolution attempts per search (1 to 4,096), 64 nested subsearches (1 to 128), and 65,536 characters of SPL (1,024 to 262,144). Reaching one records an explicit diagnostic.
- Tags (`tag=…`) are not resolved; a data model constraint that holds only tags names no index.

- map, foreach, dashboard tokens ($token$) and eval-built index values are dynamic: never resolved.
- Custom commands that run their own searches are not expanded; their inner searches are audited and extracted separately.

Index-less and unresolved searches

What it means. UAM observed searches with an unresolved part: missing source scope, a definition it could not expand, incomplete text or conflicting context. The same search can still have identified source indexes, counted separately. Unknown search head tier is a separate attribution problem.

Where it comes from. Examples include searches with no explicit source index, dynamic expressions or unsupported syntax. Role search defaults use the latest retained snapshot for each reference group, as context only. When the account's roles are resolved, a search naming no index shows what they could let it reach instead: the indexes its role search filter names, else its default indexes its roles allow.

What it cannot establish. An unresolved search may have used an index in your candidate list. Defaults do not tell us which index actually supplied data.

What to do next. Investigate the account and search syntax, and review collection diagnostics. Do not interpret an unresolved search as proof that an index was unused.

Searches that name no index



A search whose base search reads data but names no positive index term (`sourcetype=nginx` status=500, error OR fail*, | tstats count where `sourcetype=x`) is index-less: its scope is unresolved and its path is role_default. Splunk searched the indexes the account's roles give it, and the audit does not say which.

When the account's role scope is established, the unresolved accounts' Role scope column shows what an index-less search could reach: the indexes its role search filter names, when the filter names some (Splunk then searches those, not the default indexes), else its default indexes, always within what its roles allow, and only among the tier's catalog indexes. The table lists three names and counts the rest; the investigation dialog lists up to 50 with the total.

When the scope cannot be established, the default indexes recorded with the facts are shown instead (the latest snapshot per account and app). A default * is flagged as a wildcard default.

This is context only. An index-less search is never attributed to an index, so it never makes an index Searched and never removes one from the review candidates. A searched verdict is always provable from a named term.

A search that starts with a command reading no index (| rest, | inputlookup, | makeresults) makes no index claim at all and is not counted here.

Worked examples

Search or case	What Data usage records
`sourcetype=nginx status=500`	Unresolved, index-less. Role scope: the role filter's indexes, else the default indexes the roles allow.
`sourcetype=nginx [ search index=web | fields host ]`	A named read of web from the subsearch; the base search stays index-less (a subsearch cannot supply its scope).
`| rest /services/server/info`	No index claim: not counted.

Limits

- An index-less search may have read an index in your candidate list: the unresolved count is a reason to investigate before acting on an absence.
- The indexers never see the default indexes: an indexer confirmation cannot resolve an index-less search.

Other unresolved parts

A search is also unresolved, wholly or in part, when its SPL builds index values dynamically (map, foreach, $token$, eval `index=…`), names a macro, event type, saved search or data model the catalogs cannot resolve, holds a cycle or exceeds a resolution limit, has unbalanced syntax, or was truncated. Its path is dynamic. Index terms identified in the same search still count, separately.

A search whose extraction failed (extraction_failed) or whose text the audit did not carry (search_text_unavailable) is counted as unresolved, never dropped.

Each unresolved account keeps up to five sample searches with the reason of each part, so Fetch SPL examples shows what could not be resolved.

Limits

- The reasons are counted per reference group, not per distinct search.

Late filters, subsearches and negations

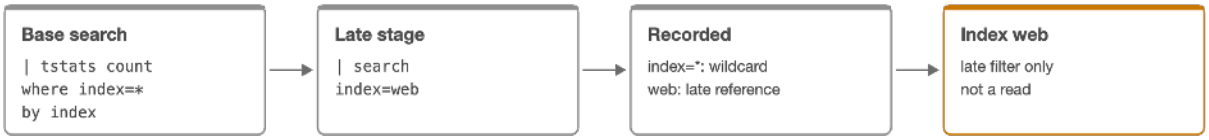
What it means. Where a term appears in the pipeline decides what it means. Only base searches select data; later index terms filter results, and negated terms exclude.

Where it comes from. The extraction splits each search into its stages and subsearches. The first stage of a pipeline is its base; every later stage is late; a subsearch has its own base.

What it cannot establish. Syntax is read, never executed: Boolean expressions are not simplified, and a late term after a stats command filters a field, not a dataset.

What to do next. Use the late references of an index detail to find searches that filter on an index without reading it, and Fetch SPL examples to check the pipeline.

Late filters



An index term after the first pipe (| search `index=web`, | where `index="web"`) filters results that the base search already read. It is recorded as a late reference: shown in the index detail, never counted as a read, never a verdict.

An index counts as late only when every term naming it in that search is late. `index=web` | search `index=web` is a named read.

A late filter never closes an unresolved base: an index-less base followed by | search `index=web` stays index-less.

Worked examples

Search or case	What Data usage records
`| tstats count where index=* by index | search index=web`	Wildcard references from `index=*`; web is a late reference only.
`sourcetype=nginx | search index=web`	Index-less (unresolved) and a late reference of web.

Limits

- A late reference can be the strongest hint of interest in an index, but the base search decided what was read.

Negations

NOT `index=web`, `index!=web` and an index term inside a NOT (…) group are negated: never counted as use. A negated term of an index also cancels a positive term of the same index in the same branch of the search, since the extraction does not simplify Boolean SPL.

Worked examples

Search or case	What Data usage records
`index=* NOT index=web`	Wildcard references from `index=*`; nothing counted for web from the negation.
`index=web NOT index=web`	No read of web.

Limits

- The expansion of `index=*` does not subtract the negated index: an expansion is a reference, not a computed set.

Write references

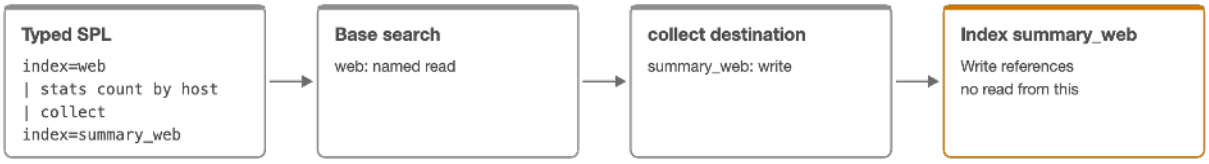
What it means. The search writes to an index: a collect destination in its SPL, for example | collect `index=summary`, or a saved search whose summary indexing or Log event alert action writes its output to an index.

Where it comes from. UAM extracts collect destinations from observed SPL, and reads the summary indexing and Log event targets from the saved search's published definition. Each is recorded with the direction write, separately from source reads.

What it cannot establish. This is evidence of a write reference, not confirmation that events were successfully indexed. A destination can have writes and no observed reads.

What to do next. Inspect SPL examples to understand the producer. Use Splunk ingestion evidence to confirm successful indexing and assess its volume.

collect and mcollect destinations



The index argument of | collect and | mcollect is recorded as a write reference of that index, in the Write references column. A write is never a read: an index only written to has no read from it and stays No reader (or Wildcard only) unless something else reads it.

collect's sourcetype argument labels its output; it is not a read filter and is not recorded. A macro used as collect's arguments keeps its meaning: its index term stays a write.

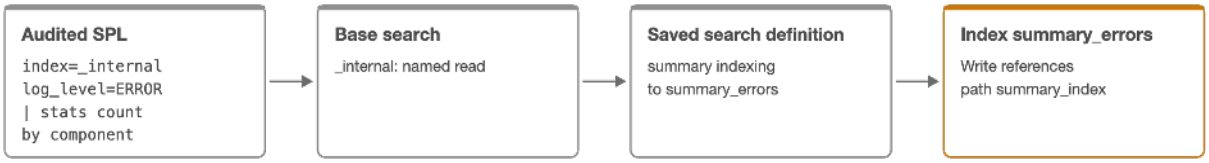
Worked examples

Search or case	What Data usage records
`index=web | stats count by host | collect index=summary_web`	A named read of web and a write reference of summary_web.
`| mstats avg(cpu) where index=metrics span=5m | mcollect index=metrics_rollup`	A named read of metrics (metric) and a write reference of metrics_rollup.

Limits

- outputlookup and outputcsv write lookups and files, not indexes, and tscollect writes namespaces: none is a write reference. Splunk's summary indexing and Log event actions are covered in the next case.
- A write reference is what the SPL asked for, not proof that events were indexed.

Summary indexing and the Log event alert action



A saved search can write its output to an index without any collect in its SPL: with summary indexing enabled (action.summary_index, target action.summary_index._name) or with Splunk's built-in Log event alert action (action.logevent, target action.logevent.param.index). The audit keeps only the saved search's SPL, so neither write appears in the search text.

UAM reads both settings from the saved search's published definition. Each run of that saved search records a write reference of the target index, on the summary_index or log_event path, and the saved search declares the same write in Knowledge objects. Splunk reports the effective target, including a default inherited from alert_actions.conf; when it reports none, the shipped defaults apply: summary for summary indexing, main for Log event.

Worked examples

Search or case	What Data usage records
Saved search "Hourly errors": `index=_internal` log_level=ERROR | stats count by component, summary indexing to summary_errors	Each run: a read of _internal and a write reference of summary_errors (path summary_index).
Alert "Failed logins": `index=auth` action=failure | stats count by user, Log event with an empty index	Each run: a read of auth and a write reference of main (path log_event).

Limits

- A run records the write the action is configured to make. Splunk writes only when the action fires: a summary search with no result or an alert whose condition is not met writes nothing.
- The target is the one Splunk reports for the saved search at the catalog's last listing, including a default inherited from alert_actions.conf. Only when Splunk reports none do the shipped defaults (summary, main) apply.
- Custom alert actions can write anywhere and are not predictable: they are never recorded.

Indexer confirmations

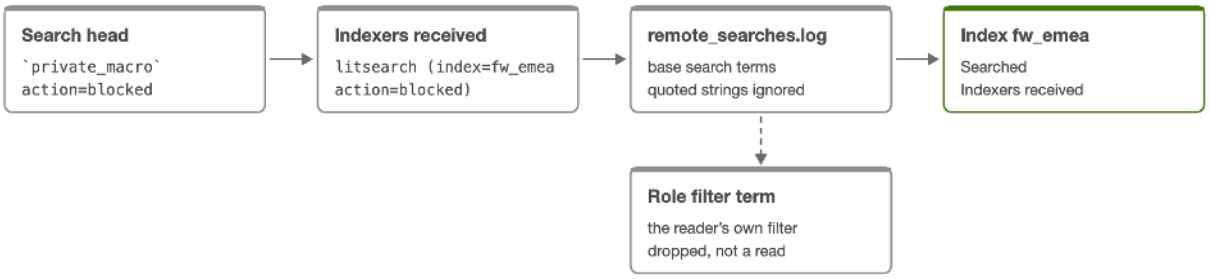
What it means. The search the indexers received, after Splunk expanded its macros, event types and saved searches. Its named index terms confirm reads the search head side could not resolve.

Where it comes from. Each search peer logs Streamed search connection closed in remote_searches.log, with the search it received. The Data usage run reads these lines after its audit windows, up to the audit cursor.

What it cannot establish. Only distributed deployments log them. The indexers never see a role's default indexes, so an index-less search stays unresolved. A served fact never adds to the search head counts.

What to do next. Use the Indexers received column to find indexes read through private macros, apps the catalogs cannot see or dynamic SPL.

What the indexers received



Splunk expands a search's macros, event types and saved-search references on the search head and sends the indexers the expanded search. Each indexer logs it when the connection closes, with the search head, the origin search id, the account, the app and the origin.

Only the base search counts (the text before the first pipe): a later search `index=…` filters results. A quoted string is never an index term. One fact per search head and origin search id.

If the received text holds a wildcard index term, every explicit index term of that text may be Splunk's own expansion of it, so only the patterns are recorded, as wildcard confirmations. A negated index term (NOT `index=…`, a term in a NOT (…) group, or a NOT group nested deeper than three levels) voids the terms of that search.

A served named term makes an index Searched when the search head side has no named read of it, with the verdict text "Named in the search the indexers received". It removes the index from the review candidates: it is the backstop for private macros, apps the catalogs cannot see and dynamic SPL. It is never added to the search head counts.

Served facts follow the reader's role scope like the others, and a served term that is the reader's own role filter is dropped.

Worked examples

Search or case	What Data usage records
`` `private_macro` action=blocked (indexers received: litsearch (index=fw_emea action=blocked)) ``	Search head side: unresolved macro. Served: a named confirmation of fw_emea, which makes it Searched.
`Received: preeventcount index=* index=_* index=web …`	A wildcard text: only `index=*` and `index=_*` are recorded; web is not confirmed by name.

Limits

- A standalone search head usually has no indexers logging what they received.
- A received line cut before its end is marked truncated; the terms of the text it kept stand.
- A search whose indexers log in two adjacent read ranges counts once per range.
- The access timeline shows search head accesses only: served facts are not in it.

Declared references

What it means. The published knowledge objects that name an index: saved searches, macros, event types, dashboards and data models. A declaration says an object names the index, not that anybody ran it.

Where it comes from. Each time a tier's knowledge object catalogs are published, their definitions are extracted with the same grammar as typed SPL and reduced per index: objects, writes and unresolved objects.

What it cannot establish. Declarations never make an index Searched and never remove it from the candidates. Wildcard declarations are not expanded onto index rows. Unscheduled and disabled objects remain declarations.

What to do next. Use the declared objects of an index detail, and the Knowledge objects view, to find who to ask before acting on an index nobody reads.

Definitions that name an index



The definitions read are: the search of saved searches and event types, the definition of macros, the search and query bodies of dashboards (Simple XML and Dashboard Studio), and the root constraints of data models. A dashboard query with tokens, or a chained search, is dynamic.

Each index row carries its declared objects (objects naming it, objects writing to it, unresolved objects); the index detail lists up to 50 sampled objects per tier with their app, owner, sharing, schedule and disabled state.

The Knowledge objects view compares what each object declares with what it ran in the period.

Worked examples

Search or case	What Data usage records
Saved search "Daily firewall report": `index=fw_*` | stats count by action	A declared wildcard reference fw_* (not expanded onto index rows).
Macro web_idx: `index=web`	A declared reference of web, one object.

Limits

- Private objects the catalog cannot see are not declarations.
- A collect in a saved search's SPL, and its summary indexing or Log event action, are declared writes.
- A declaration is the definition at the last publication, not at the time of a run.

Incoming data and review candidates

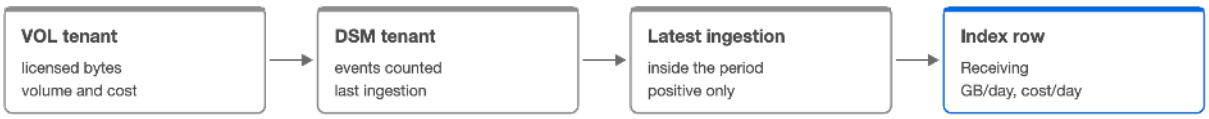
What it means. By default, candidates have positive linked VOL or DSM ingestion evidence in the selected period, but no named source reads. Wildcard-only references remain candidates because they do not establish that each matching index supplied data.

Where it comes from. Value & cost compares the linked incoming-data observations with the retained search references of every monitored tier. Account, app, class and path filters from Index activity do not apply to it.

What it cannot establish. This does not mean never used or safe to delete. Retention, incomplete collection, unresolved searches and excluded activity can hide usage. Write references may still exist. Volume and cost come from the linked VOL tenant: without one, no volume or cost is shown.

What to do next. Choose a representative period, check coverage and unresolved searches, then consult owners and ingestion or storage measurements. Treat the list as candidates for investigation, not an automatic deletion recommendation.

Incoming data from VOL and DSM



Each search head tier can be linked to one VOL tenant and to DSM tenants (Settings → Data usage). Every enabled VOL and DSM tenant you can read is listed, whichever account it collects through. Without an explicit link, the map selects them automatically when it is read: the first VOL tenant by tenant identifier (more than one is not expected) and every DSM tenant.

An index is receiving when its latest ingestion observation lies in the selected period: for VOL, the last time its licensed volume was positive; for DSM, an entity with a positive event count and a last ingestion time. Incoming data is evidence that data arrived, never that it was read.

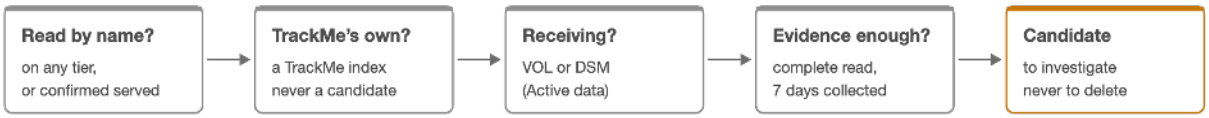
Volume is VOL's 30-day average of complete days (or the last 24 hours while the index has no complete day), with the cost of VOL's license view. The same index seen through two links is one volume: the larger.

Links grant no access: every read rechecks that you can read the linked tenant, and that it is enabled.

Limits

- These are latest observations: an index that stopped receiving before the period does not count as receiving, and an older period cannot be reconstructed.
- A linked tenant that could not be read completely leaves the candidates in place but withholds their volume and cost totals.
- Indexes are matched by name, so a tenant that measures other indexers can add incoming data or volume to an index of the same name. Unselect any tenant that measures other indexers than the ones this UAM tenant searches.

How a review candidate is decided



An index is a review candidate when no search head tier reads it by name and no indexer confirmation names it, it is not one of TrackMe's own indexes and, with Active data on (the default), it receives data. An index read on another tier is never a candidate.

Wildcard only and Excluded apps only indexes remain candidates and are shown apart: a broad pattern, or a TrackMe tracker reading an index, is not use of its data.

TrackMe's own indexes are never candidates: the global index settings, every tenant's index overrides and this UAM tenant's own indexes. They are written and read by TrackMe.

Candidates, and their volume and cost, are withheld while the evidence cannot support an absence of reads: when the fact read was incomplete, when no Data usage run read the audit trail in the period, or while the collection covers less than the period and less than seven days of it (Collecting, with the date the list will be available).

Limits

- A candidate is an index to investigate, never a recommendation to delete it: retention, collection gaps, index-less searches and excluded apps can hide reads.

Verdicts and value classes

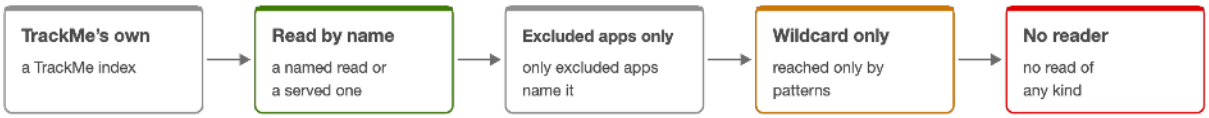
What it means. Each index has a verdict for the selected tier (Searched, Indirect only, No observed reads) and a value class on the indexing layer (Read by name, Wildcard only, No reader, Excluded apps only, TrackMe's own).

Where it comes from. The verdict comes from the named and wildcard reads, on the search head side or served. The value class adds the excluded apps' reads and TrackMe's own indexes, and is taken on every tier together.

What it cannot establish. No verdict claims an absence of use: No observed reads and No reader describe the evidence held for the period, nothing more.

What to do next. Use the verdict to investigate a tier, the value class to weigh volume and cost against reads.

Verdicts and value classes, in order



Verdicts. Searched: at least one named read on the search head side, or a named term in the search the indexers received. Indirect only: reached by wildcard references only. No observed reads: neither. Late filters, writes, denied searches, index-less searches and declarations never decide a verdict.

Value classes, decided in this order. TrackMe's own: an index TrackMe writes to for a tenant, whatever reads it. Read by name: Searched on any tier. Excluded apps only: only the excluded apps read it by name (TrackMe and the CIM acceleration app by default), even if wildcards also reach it. Wildcard only: reached only by wildcard references. No reader: no read of any kind.

Reads by excluded apps are shown, never counted: the verdict text says "Read only by excluded apps: … (N searches)". Only their literal reads count there (no wildcard, write or served fact).

Accesses, the Overview weight, are an index's named reads (or its served named reads when the search head side has none) plus its wildcard references. Low use marks the quarter of the Read by name indexes with a volume read the least per GB/day, when there are at least four of them.

Limits

- With a tier selected, a verdict describes that tier; an index read on another tier says so in its detail.
- Before seven days of collection, the counts describe the evidence collected so far, dated "since collection began".

Paths, origins and source types

What it means. Path explains how a dataset was referenced. Origin reports where the search was dispatched. Sourcetype is the source type named in the observed search, when it can be established.

Where it comes from. Literal means a direct source term; macro, eventtype, savedsearch and datamodel identify a definition UAM resolved. A wildcard is a kind of term, matched against the catalog when the map is read. UI:Search and UI:Dashboard identify dispatch provenance reported by Splunk.

What it cannot establish. A later SPL field filter is only a reference, not a source read. Role defaults describe possible scope and never count as observed usage. Missing app, origin or sourcetype values are unknown, not inferred.

What to do next. Use the joint detail rows rather than combining separate totals. Open the account to investigate its activity, and fetch SPL examples to inspect the syntax.

The values you can meet

Paths: literal (a term typed in the search, a wildcard pattern included), macro:<name>, eventtype:<name>, savedsearch:<name>, datamodel:<name> (the first object of a resolution chain), role_default (an index-less search), dynamic (any other unresolved part) and served (received by the indexers).

Origins: UI:Search, UI:Dashboard:<view>, scheduler, rest, rest:jobs, summary_director, subsearch:…; other UI origins are grouped as UI:*, and a missing one is N/A.

Index totals give one unit per search to each index. Path and sourcetype rows count references: one search through two paths counts once on each path, so their sum is not a number of searches.

Limits

- A sourcetype is recorded only for a search whose base names one literal index and whose extraction is complete: the sourcetypes its completion counted, else the sourcetype terms of its base search. Any other search records none.

Coverage and uncertainty

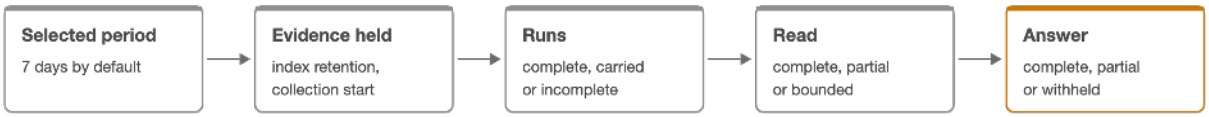
What it means. The selected range bounds the retained facts used for this answer. Retained facts dates show the evidence returned, not proof that every moment or index was observed.

Where it comes from. The map reads tenant-scoped evidence, collection receipts and index catalogs using your access. The catalog describes the monitored search head tier; it is not proof of which indexer served a search.

What it cannot establish. Missing or stale catalogs, collection gaps and read limits can make an answer partial. Even a complete response cannot establish lifetime non-use. An active Exclude TrackMe setting also narrows usage evidence.

What to do next. Check Collection health and the selected tier and period before interpreting missing reads. Data usage stays still until you change the selection or use the global Refresh button; the workspace health indicator can update independently.

What bounds an answer



Retention: the facts live in the tenant's evidence index. A period longer than its retention holds only what remains. A new tenant reads the audit trail from where its collection began (and, with a backfill, from the backfill depth, never before the oldest audit event held).

Collection start: until the collection covers the period, or seven days of it, the counts are dated "since collection began" and the candidates are withheld (Collecting).

Runs: a run that stopped on its time budget carries its range over to the next run (carried, nothing lost). A run that could not extract (incomplete) makes the period partial (extraction_coverage).

Catalogs: a tier's index catalog older than 48 hours is stale; a missing one leaves wildcards unexpanded. A catalog still being listed on a new tenant is pending, not a fault.

Reads: at most 50,000 fact keys and 10,000 catalog indexes per tier are read. Past a bound, or when a search failed or was cut, the answer is partial and the candidates are withheld. Facts written before the rollup key existed are reported and not counted.

Roles: the current users and roles catalogs apply to the whole period.

Limits

- Even a complete answer covers the evidence held for the period, never a proof of lifetime non-use.
- Exclusions (Settings → Data usage, and the Exclude TrackMe switch) narrow the evidence counted, never collection.

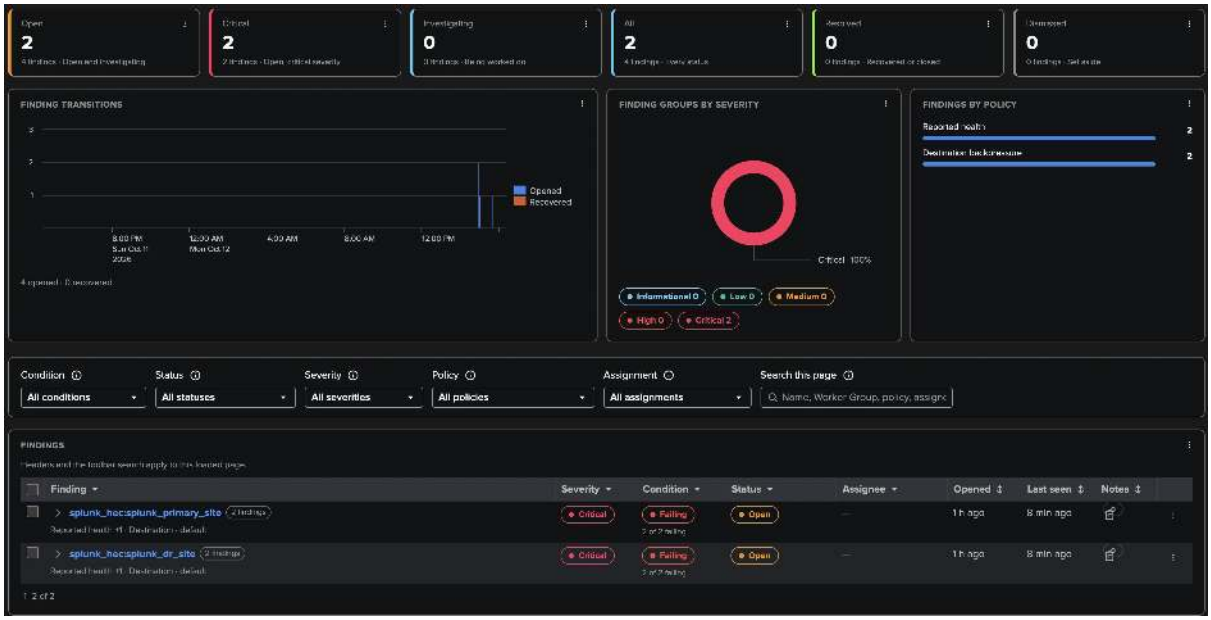
8.3.17 Findings, Finding groups and findings actions

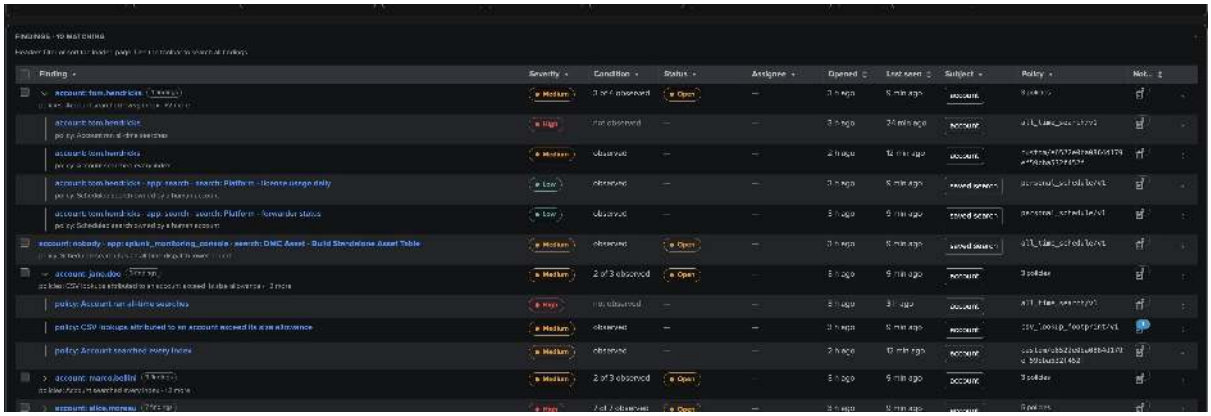
Seven tenant types in TrackMe do not track entities with a state colour: **User Activity Monitoring (UAM)**, **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet**, **Splunk Forwarder Collectors**, **Cribl Stream** and **Cribl Edge**. They are **dedicated findings tenants**. Each one collects what it monitors on a schedule, evaluates a catalogue of **policies** against it, and raises a **finding** whenever a policy detects a condition on one subject: an account running all-time searches, a Worker Group under backpressure, an Edge Node that stopped reporting, a forwarder input that went quiet. A healthy object has no finding at all; the work queue only ever contains what needs a look.

This page explains the concepts the seven tenant types share: what a finding is and how it lives, how findings are triaged together as **Finding groups**, how they are assigned and routed to teams, notified, acted upon with **findings actions** and **Webhook accounts**, and reviewed by the **AI Agents**. The pages of each tenant type document what they monitor, their policies and their own differences:

- *User Activity Monitoring* — User Activity Monitoring

- *Cribl Stream* and *Policy catalogue* — Cribl Stream
- *Cribl Edge* and *Built-in policies* — Cribl Edge
- *Splunk HEC* and *Built-in policies* — Splunk HEC
- *Splunk S2S* and *Built-in policies* — Splunk S2S
- *Splunk UF Fleet* and *Built-in policies* — Splunk UF Fleet
- *Splunk Forwarder Collectors* and *Built-in policies* — Splunk Forwarder Collectors





From policies to findings

The flow is the same in the seven tenant types:

1. **Collection jobs** (scheduled reports run by TrackMe search jobs) build the inventory of the monitored objects and read their activity or their measurements.
2. **Policies** — built in, each one enabled or not, with a default severity, a scope and its own settings — are evaluated against what was collected. You can add your own **custom policies** in SPL.
3. Every condition a policy detects on one subject is a **finding**. The same condition on the same subject is always the same finding: it keeps its identity, its evidence, its notes and its history across the collection runs.
4. Findings about the same object are triaged together as one **Finding group**.
5. Around that core: email notifications, assignment to users or teams, findings actions (drilldown searches, links, webhooks), the AI Agent, indexed finding events for your own Splunk alerts, Enterprise Security or SOAR, and a **Collection health** tab that tells you whether collection runs as configured.

Bank holidays. A findings tenant uses the *bank holiday calendars* like every tenant. While a holiday of its calendars is active, the policies **set to hold** open no new finding; each policy chooses it in its editor (*During bank holidays*: **Hold new findings** or **Keep opening findings**), with a default that suits it: the configuration reviews of UAM, Traffic outliers on Cribl Stream, Fleet connectivity and Fleet coverage on Cribl Edge, the traffic policies and Token silence on Splunk HEC, Pool traffic drop and Pool traffic outlier on Splunk S2S, Agent disconnected, Mass disconnect, Fleet coverage and Phonehome drop on Splunk UF Fleet, Collected volume drop, Collected volume outlier and Mass silence on Splunk Forwarder Collectors hold; every other policy keeps opening findings, so a real failure on a holiday still opens its finding. A finding already open keeps being evaluated and can recover and close, a resolved or dismissed finding is not reopened by a held condition, and a calendar that cannot be read holds nothing. After the holiday, a condition that is still there opens its finding as on any other day.

How a findings tenant differs from an entity tenant

	Entity tenants (DSM, DHM, MHM, FLX, FQM, WLK, VOL)	Findings tenants (UAM, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge)
What you look at	Entities that always exist and always have a state	Findings: they exist because a policy detected something
Health	A state colour (green, orange, red, blue) driven by thresholds, outliers and an impact score	A finding has a **severity** and an investigation **status**; there is no score and no state colour
Alerting	Stateful alerts (opened / updated / closed emails), notable events, acknowledgements	Email notifications from the finding transitions, finding events, webhook automation rules
Configuration	Thresholds per tenant and per entity, priority / tag / SLA policies	A policy catalogue with severities, scopes and settings
Good at	The continuous health of many similar things, at a glance, with SLA and KPI history	Sparse, varied conditions that need an investigation, evidence, an owner and a recorded decision

A tenant is either an entity tenant or one of the seven findings tenant types, never a mix: each findings tenant type is created from its own card in the tenant creation wizard.

Which tenant types

	User Activity Monitoring	Cribl Stream	Cribl Edge
Findings are about	Accounts and the saved searches they own (and other knowledge objects)	The Leader, Worker Groups, Workers and Worker processes, Sources, Destinations, Pipelines, Routes	The Leader, Fleets, Sources and Destinations, Edge Nodes
A finding is	One policy on one subject (an account or a saved search)	One policy and one measurement on one object	One policy and one measurement on one object
When the condition goes away	The finding reads **not observed**; it is never resolved for you	A measured recovery **resolves** the finding automatically	Same as Cribl Stream
Automatic Finding group	One per account: its findings plus those of the saved searches it owns	One per object (a Worker machine with its processes)	One per object (a Fleet, a node, a Source or Destination)
Severities shown	Low, Medium, High, Critical	Informational, Low, Medium, High, Critical	Informational, Low, Medium, High, Critical
Dismiss until it recovers	—	Yes (the default)	Yes (the default)
Email action, direct user notifications	Yes	—	—
AI Agent	UAM AI Agent	Cribl Stream AI Agent	Cribl Edge AI Agent
Bank holidays (effects that apply)	Suppress new alerts	Suppress new alerts, Exclude from ML training	Suppress new alerts
Licence	Any edition except Foundation	Enterprise, Unlimited, trial and Developer licences	Same as Cribl Stream

A **Splunk HEC** tenant (*Splunk HEC*) reads like the Cribl Edge column: its findings are about HEC endpoints, pools and tokens, one policy and one measurement on one object; a measured recovery resolves the finding; each object has one automatic Finding group; it has its own AI Agent, the Splunk HEC AI Agent; its bank holidays suppress new alerts and are excluded from its traffic ML models; and it needs the same licences as Cribl Stream.

Splunk S2S (*Splunk S2S*) and **Splunk UF Fleet** (*Splunk UF Fleet*) tenants read the same way: their findings are about receivers, intermediate forwarders, pools and destinations, or about agents, groups, the fleet, Deployment Servers and destinations; a mass event is one cohort finding holding the findings of its members; each has its own AI Agent, and they need the same licences as Cribl Stream. **Splunk Forwarder Collectors** (*Splunk Forwarder Collectors*) tenants read the same way too: their findings are about collectors, their inputs, groups, the fleet and destinations. Bank holidays suppress new alerts on all three, and are also excluded from the traffic ML models of Splunk S2S and from the collected-volume models of Splunk Forwarder Collectors (Splunk UF Fleet has no ML model).

Note

Every finding transition is also written as an indexed **finding event** in the tenant's notable index — `sourcetype=trackme:uam:finding` for UAM, `sourcetype=trackme:cribl:finding` (and `trackme:cribl:finding_group` for the Finding group transitions) for Cribl Stream, Cribl Edge, Splunk HEC, Splunk S2S, Splunk UF Fleet and Splunk Forwarder Collectors, with `object_category=cribl_edge` on Edge, `splunk_hec` on Splunk HEC, `splunk_s2s` on Splunk S2S, `splunk_uf` on Splunk UF Fleet and `splunk_collector` on Splunk Forwarder Collectors. Build your own Splunk alerts, Enterprise Security correlations or SOAR playbooks on them.

A finding and its lifecycle

Identity

A finding is identified by **its policy and its subject**. On UAM the subject is an account or a saved search of the monitored environment; on the other dedicated tenant types it is an object and the measurement the policy watches (a Worker process and its CPU, a Destination and its backpressure, an Edge Node and its connectivity, a HEC endpoint and its refusals, a forwarding agent and its connection, a collector's input and its delivery). The identity is stable: when the same condition comes back on the same subject, it is the same finding again, with its earlier notes and history.

Each finding carries:

- its **evidence**: what the policy measured or saw, and when (first seen, last seen);
- a **severity** and an investigation **status**;
- an optional **assignee**, **notes** (Markdown) and a full **history** of what changed, who changed it and why — an operator, the policy itself, TrackMe or the AI Agent.

Episodes: opens, recovers, reopens

A finding has two independent axes: whether the **policy still sees the condition**, and the **status** an operator gave it. How the first axis behaves follows the nature of the data, and that is the main difference between the tenant types.

Tenant type	What happens when the condition goes away and comes back
User Activity Monitoring	A finding is **observed** while the policy sees the condition. After a complete policy evaluation that no longer sees it, the finding reads **not observed** — it is never resolved for you, because a configuration or a behaviour that disappears from one evaluation deserves a human decision. A resolved finding that is observed again is **reopened**. An evaluation that could not cover everything may add findings, never clear them.
Splunk HEC, **Splunk S2S**, **Splunk UF Fleet**, **Splunk Forwarder Collectors**, **Cribl Stream** and **Cribl Edge**	The **condition** of a finding is *failing*, *recovered*, *unknown* or *ended*, and each failure is a numbered **episode**. A measured recovery **resolves** an open or investigating finding; a new failure **reopens** it as a new episode. An unknown measurement never counts as a recovery. TrackMe also closes findings automatically, always with a reason: the object was not evaluated for 24 hours, was removed from Cribl or from the tenant's directory or is no longer listed, monitoring was turned off, the source of the tenant changed.

Severity

The dedicated tenant types use one severity scale. Every policy has a default severity, which you can change in the policy settings, and an operator can override the severity of a finding or a Finding group by hand (**Set severity**). **Use policy severity** (on a Finding group: **Use member severity**) removes the override.

Severity	Meaning and use
Critical	Act now: an outage, a component that stopped, a change with an immediate impact.
High	Act soon: a degradation or a risk that is likely to become an outage.
Medium	Investigate: a deviation or a misconfiguration. The default severity of most policies and the default minimum severity of notifications.
Low	Worth knowing: a small drift, a hygiene topic.
Informational	Every dedicated tenant type except UAM: how a finding (or a Finding group) is **shown** once it recovered, its episode ended or it was resolved. The finding keeps the severity it reached, which the tooltip of the Informational pill names, so you can still sort and filter on the real level.

How each tenant type uses the scale:

- **UAM** shows Low, Medium, High and Critical. Most built-in policies default to Medium; a personal schedule defaults to Low, an all-time search to High, and the resource consumption policies raise the severity with the share of the environment an account consumes. A Finding group carries the highest severity of its observed findings.
- **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet**, **Splunk Forwarder Collectors**, **Cribl Stream** and **Cribl Edge** add the Informational display for settled findings. Each policy has its own default severity, listed in the policy reference of its tenant type (on Cribl Stream, for example, Leader availability defaults to Critical and most availability and connectivity policies to High), and some policies raise a finding to Critical while the condition is severe. A Finding group carries the highest severity of its failing findings.

In the example below, a Worker that went down and came back is shown **Informational**, **Recovered** and **Resolved**; hovering the severity names the severity it reached while it failed (*Recovered or resolved; it reached High*).



Statuses and triage

The status is the **operator axis**: what a person (or the AI Agent) decided. The same four statuses exist in every dedicated tenant type.

Status	Meaning
Open	New, or back for another look. Every finding starts open.
Investigating	Someone is on it.
Resolved	The problem is fixed. On every dedicated tenant type except UAM, a measured recovery resolves the finding automatically.
Dismissed	No action is needed, even while the condition holds — for a time, until it recovers, or permanently (see below).

Triage decisions, available from the finding menu, the Finding group menu, a selection of rows and the investigation view:

Dismiss
: Choose how long the dismissal holds:

  - **Until it recovers** (every dedicated tenant type except UAM, the default): the dismissal holds while the object keeps failing; if it fails again after recovering, the policy reopens the finding.

- **For a time**: 1, 7, 30 or 90 days. When the time is up the finding returns to open, whether or not the condition still holds, so you get another look at it.
- **Until reopened by hand (permanent)**: no later detection reopens it.

Snooze
Holds the emails without changing the status. On UAM, **Snooze notifications** pauses the email notifications for 1 hour, 6 hours, 1 day, 7 days or 30 days. On the other dedicated tenant types, **Snooze** (1 hour to 30 days) also holds the automation rules, and a snoozed finding no longer colours its object in Topology Studio: the object reads blue when nothing else is wrong.

Resolve, Mark investigating, Reopen
The status decisions above, with an optional note recorded in the history.

Assign
To a Splunk user or an assignment group — see *Assignment and automatic routing*.

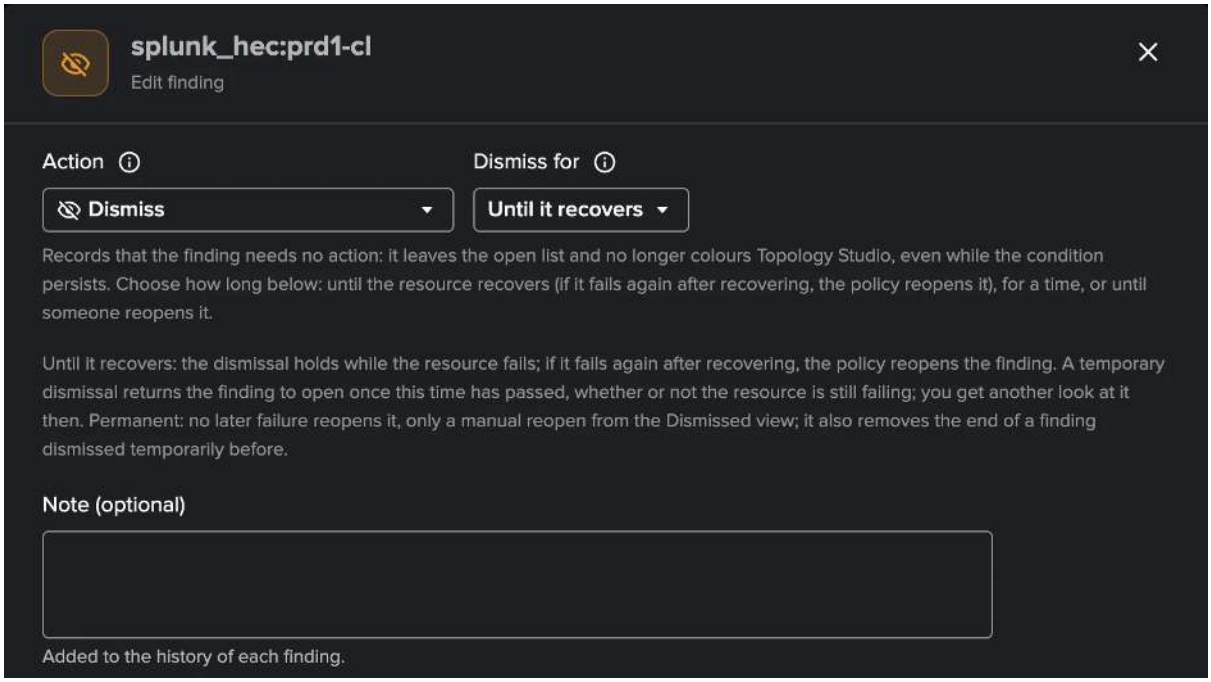
Set severity
A manual override that survives the policy runs.

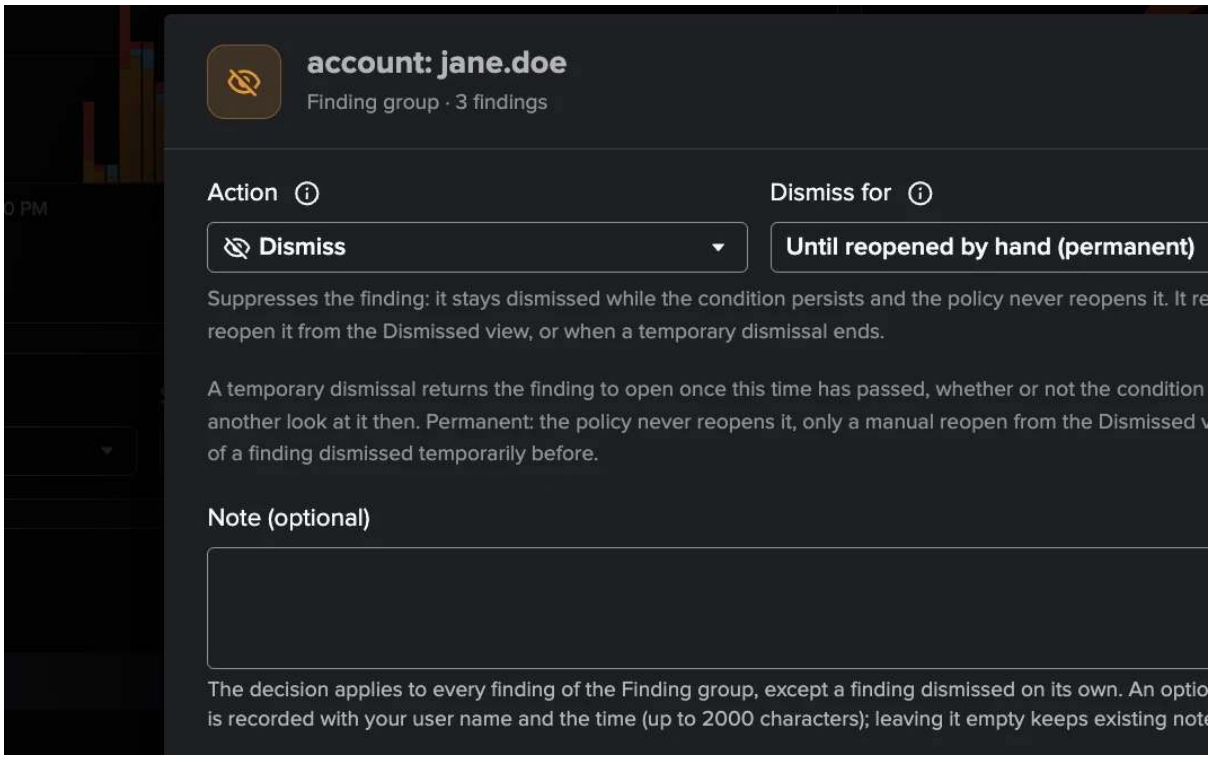
Notes
Markdown notes on the finding or the Finding group, visible to everyone who can read the tenant. The AI Agent keeps one current note of its own per finding, signed `[AI Agent]`.

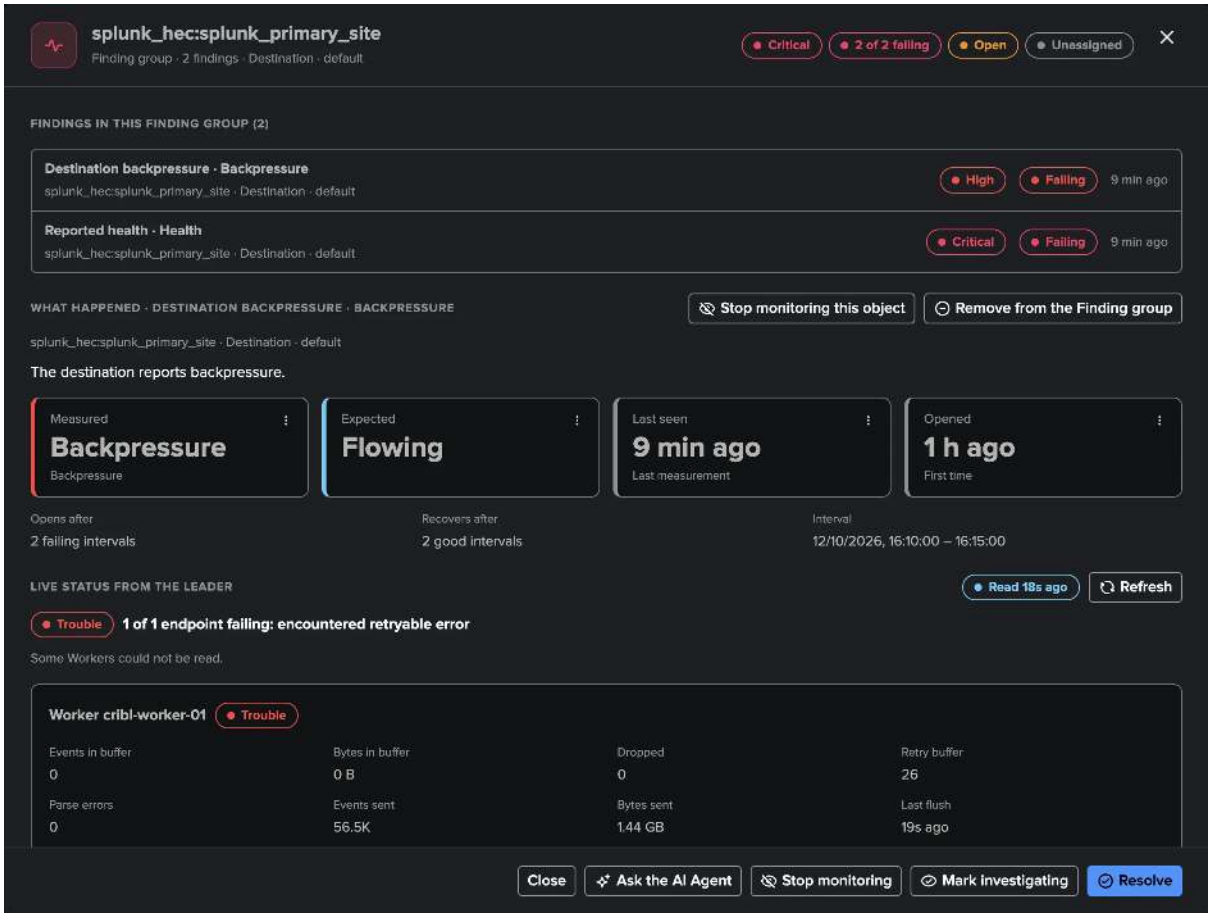
Some decisions belong to one tenant type:

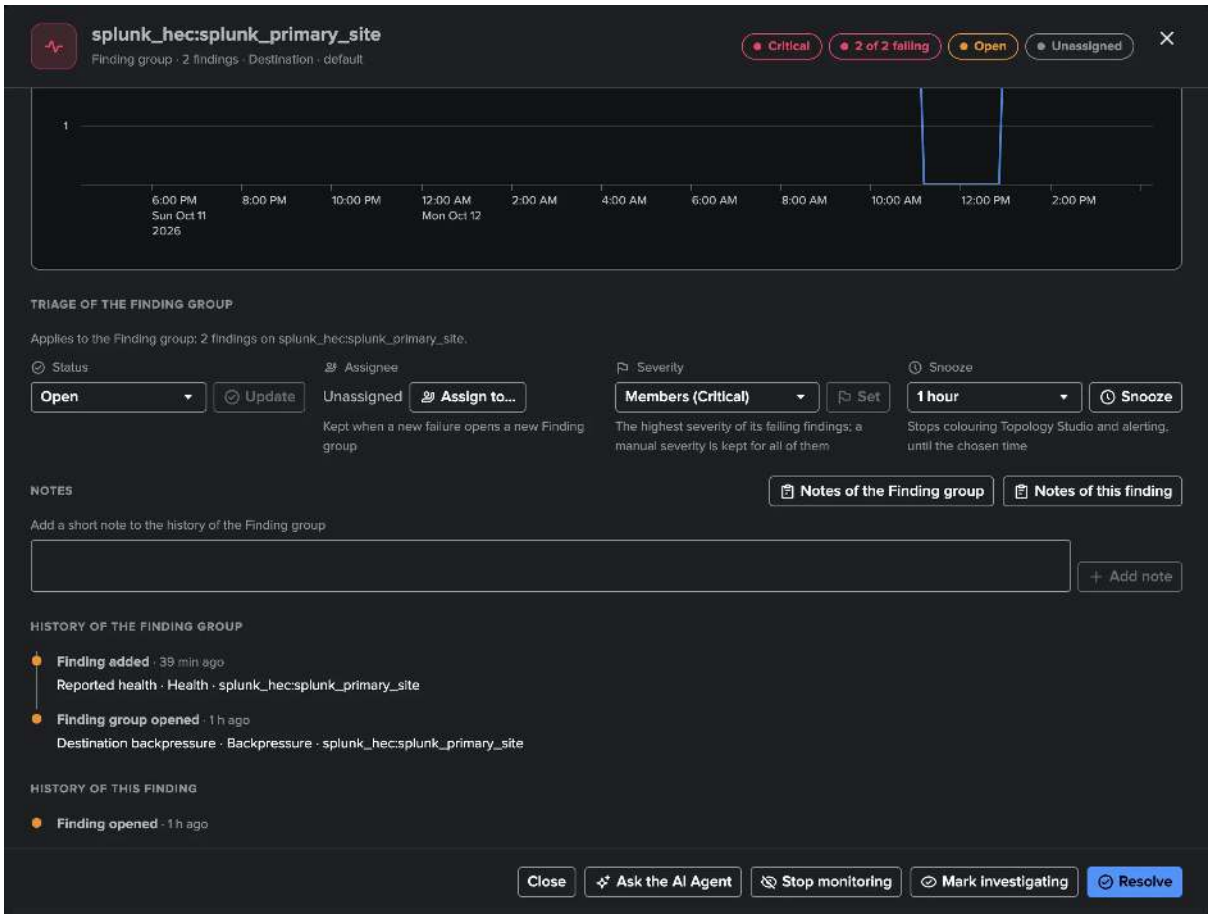
Tenant type	Its own decisions
User Activity Monitoring	**Set class** of an account (human, service, system) and **Exclude owner from this policy**; **Dismiss only this finding** inside a Finding group (see *Finding groups*).
Cribl Stream	**Not an anomaly** (a traffic outlier finding: its model relearns without that period), **Stop monitoring this object**, **Forget this object** (an object removed from Cribl).
Cribl Edge and **Splunk UF Fleet**	None beyond the shared decisions.
Splunk HEC and **Splunk S2S**	**Not an anomaly** on a traffic outlier finding (a HEC pool or token, an S2S pool): its model relearns without that period.
Splunk Forwarder Collectors	**Not an anomaly** on a collector volume outlier, and **Exclude this input from this policy** (one input out of one input policy; its other policies keep judging it).

Every decision is recorded in the history of the finding (and of its Finding group) with the actor, the time and the note. Decisions are protected against concurrent changes: if someone changed the Finding group after you opened it, your decision is refused and you are asked to reload rather than overwriting their work.


splunk_hec:prd1-cl
Edit finding
Action
Dismiss for
Dismiss
Until it recovers
Records that the finding needs no action: it leaves the open list and no longer colours Topology Studio, even while the condition persists. Choose how long below: until the resource recovers (if it fails again after recovering, the policy reopens it), for a time, or until someone reopens it.
Until it recovers: the dismissal holds while the resource fails; if it fails again after recovering, the policy reopens the finding. A temporary dismissal returns the finding to open once this time has passed, whether or not the resource is still failing; you get another look at it then. Permanent: no later failure reopens it, only a manual reopen from the Dismissed view; it also removes the end of a finding dismissed temporarily before.
Note (optional)
Added to the history of each finding.


account: jane.doe
Finding group · 3 findings
Action
Dismiss for
Dismiss
Until reopened by hand (permanent)
Suppresses the finding: it stays dismissed while the condition persists and the policy never reopens it. It re
reopen it from the Dismissed view, or when a temporary dismissal ends.
A temporary dismissal returns the finding to open once this time has passed, whether or not the condition
another look at it then. Permanent: the policy never reopens it, only a manual reopen from the Dismissed v
of a finding dismissed temporarily before.
Note (optional)
The decision applies to every finding of the Finding group, except a finding dismissed on its own. An optio
is recorded with your user name and the time (up to 2000 characters); leaving it empty keeps existing note


splunk_hec:splunk_primary_site
Finding group · 2 findings · Destination · default
Critical
2 of 2 failing
Open
Unassigned
FINDINGS IN THIS FINDING GROUP (2)
Destination backpressure · Backpressure
splunk_hec:splunk_primary_site · Destination · default
High
Failing
9 min ago
Reported health · Health
splunk_hec:splunk_primary_site · Destination · default
Critical
Failing
9 min ago
WHAT HAPPENED · DESTINATION BACKPRESSURE · BACKPRESSURE
Stop monitoring this object
Remove from the Finding group
splunk_hec:splunk_primary_site · Destination · default
The destination reports backpressure.
Measured
Backpressure
Backpressure
Expected
Flowing
Last seen
9 min ago
Last measurement
Opened
1 h ago
First time
Opens after
2 failing intervals
Recovers after
2 good intervals
Interval
12/10/2026, 16:10:00 – 16:15:00
LIVE STATUS FROM THE LEADER
Read 18s ago
Refresh
Trouble
1 of 1 endpoint failing: encountered retryable error
Some Workers could not be read.
Worker cribl-worker-01
Trouble
Events in buffer
0
Bytes in buffer
0 B
Dropped
0
Retry buffer
26
Parse errors
0
Events sent
56.5K
Bytes sent
1.44 GB
Last flush
19s ago
Close
Ask the AI Agent
Stop monitoring
Mark investigating
Resolve


splunk_hec:splunk_primary_site
Finding group · 2 findings · Destination · default
Critical
2 of 2 failing
Open
Unassigned
1
6:00 PM
Sun Oct 11
2026
8:00 PM
10:00 PM
12:00 AM
Mon Oct 12
2:00 AM
4:00 AM
6:00 AM
8:00 AM
10:00 AM
12:00 PM
2:00 PM
TRIAGE OF THE FINDING GROUP
Applies to the Finding group: 2 findings on splunk_hec:splunk_primary_site.
Status
Open
Update
Assignee
Unassigned
Assign to...
Kept when a new failure opens a new Finding group
Severity
Members (Critical)
Set
The highest severity of its failing findings; a manual severity is kept for all of them
Snooze
1 hour
Snooze
Stops colouring Topology Studio and alerting, until the chosen time
NOTES
Notes of the Finding group
Notes of this finding
Add a short note to the history of the Finding group
Add note
HISTORY OF THE FINDING GROUP
Finding added · 39 min ago
Reported health · Health · splunk_hec:splunk_primary_site
Finding group opened · 1 h ago
Destination backpressure · Backpressure · splunk_hec:splunk_primary_site
HISTORY OF THIS FINDING
Finding opened · 1 h ago
Close
Ask the AI Agent
Stop monitoring
Mark investigating
Resolve

Finding groups

One problem, one decision

A single problem rarely raises a single finding. A Worker that runs out of memory may raise CPU, memory and availability findings on the machine and on each of its Worker processes; an employee who leaves may own a dozen saved searches, each raising a personal-schedule finding. Deciding on each finding separately is slow and error-prone. A **Finding group** gathers the findings of one object so that you investigate it once, decide once and assign it once.

The **Finding group carries the triage decision**: its status, dismissal, assignee, snooze and severity override apply to every finding in it, and a finding that joins the group later takes the group's state.

Automatic grouping

Finding groups form automatically; you do not need to do anything.

Tenant type	What one automatic Finding group holds
User Activity Monitoring	**An account and the saved searches it owns**: every finding about the account (all-time searches, resource consumption) together with the findings about its saved searches (personal schedules, real-time schedules). A saved search without an owner keeps its own group. An automatic group can hold up to 5,000 findings.
Cribl Stream	**One object**: a Source, a Destination, a Worker Group, the Leader — and a Worker machine together with its Worker processes. Up to 50 findings.
Cribl Edge	**One object**: a Fleet, an Edge Node, a Source or Destination of a Fleet, the Leader. Up to 50 findings.
Splunk HEC	**One object**: an endpoint, a pool or a token. Up to 50 findings.
Splunk S2S	**One object**: a receiver, an intermediate forwarder or a pool. Up to 50 findings.
Splunk UF Fleet	**One object**: an agent, a group of agents, the fleet or a Deployment Server. A mass event is one cohort finding on the group or the fleet. Up to 50 findings.
Splunk Forwarder Collectors	**One object**: a collector, an input, a group of collectors or the fleet. Up to 50 findings.

On every dedicated tenant type except UAM, a new failure on a resolved Finding group reopens the group. On UAM, a new kind of finding on a dismissed group returns the group to open with the new finding, while the findings the dismissal covered stay dismissed: a dismissal never silences a new kind of problem.

A finding keeping its own decision (UAM)

On UAM, an account group may mix an accepted situation and a real problem — for example a personal schedule you approve and a real-time schedule that must be fixed. **Dismiss only this finding** on a member sets that one finding aside, dismissed on its own (for a time or permanently), while the rest of the group stays open. The expanded row says so ("Dismissed on its own until …"); later decisions on the group leave it untouched. **Reopen this finding** brings it back into the group's state. On the other dedicated tenant types, remove the finding from the Finding group instead.

Manual grouping

Sometimes you know better than the automatic grouping: two Destinations failing because of the same upstream outage, or several accounts belonging to one project. The Finding group menus and the selection toolbar offer:

Action	What it does
Merge into one Finding group	Merges 2 to 25 Finding groups into one manual group (up to 50 findings).
Add to the Finding group	Adds findings or groups to an existing manual group.
Remove from the Finding group	Takes up to 25 findings out of a group; they keep the decisions of the group. On the other dedicated tenant types each removed finding gets a group of its own; on UAM the findings return to their account's automatic group, and the findings of one account always move together.
Ungroup	Dissolves a manual group; every finding returns to its automatic group.

Manual groups stay as you made them: automatic grouping never undoes a deliberate merge. On UAM, manual grouping is reserved to the tenant administrators; on the other dedicated tenant types, to the roles that can triage (the tenant's power and admin roles).

How the Findings tab lists Finding groups

The **Findings** tab lists **one row per Finding group**, with the finding, its severity, its condition or observation, its status, its assignee and when it was opened and last seen. A group of more than one finding has an expander that shows its members, each with its own policy, severity and condition. The tiles above the table count Finding groups and drive the filters. Filters, sorting and selections act on **whole Finding groups**: a filter that matches one member returns its group, and a member hidden by a filter can never become the target of a bulk decision.

Assignment and automatic routing

A Finding group is assigned to **one destination**: a Splunk user, or an **assignment group**. Assignment lives in the workspace **Settings → Assignment**.

Important

An **assignment group** is a team destination — "Platform team", "Security operations" — not to be confused with a **Finding group**, which gathers related findings. An assignment group has a name, an optional description and an optional **team email** address. It has no members and grants no access: being "in" a team is not a TrackMe concept, and an assignment never changes who can read the tenant.

Assignment groups

Create up to 100 assignment groups per tenant. A group can be disabled (it then receives no new assignment and no email) but never deleted, so the history of past assignments stays readable.

Routing rules

Routing rules assign new Finding groups automatically. They are **ordered**: TrackMe evaluates them from the top and the **first enabled rule that matches** wins. Within a criterion, the values are alternatives (OR); the criteria of a rule must all match (AND), on the same finding of the group. An empty criterion matches everything; text criteria accept case-sensitive `*` and `?` patterns.

Tenant type	Routing criteria
Every tenant type	Policy, severity, subject type
User Activity Monitoring	Owner or account, app, environment
Cribl Stream	Worker Group, object name
Cribl Edge	Fleet, object name (a Leader finding has no Fleet; its name is *Cribl Leader*)
Splunk HEC and **Splunk S2S**	Pool, object name
Splunk UF Fleet and **Splunk Forwarder Collectors**	Group, object name

Automatic routing runs in the tenant's own collection job, a few Finding groups at a time (25 per run, in rotation). It only assigns Finding groups that are **unassigned and under automatic assignment**; it never changes an existing owner. Two buttons help you tune the rules before relying on them:

- **Preview matches** shows which open Finding groups the draft rules would assign, without saving anything;
- **Apply to selected groups** applies the saved rules to up to 25 Finding groups now.

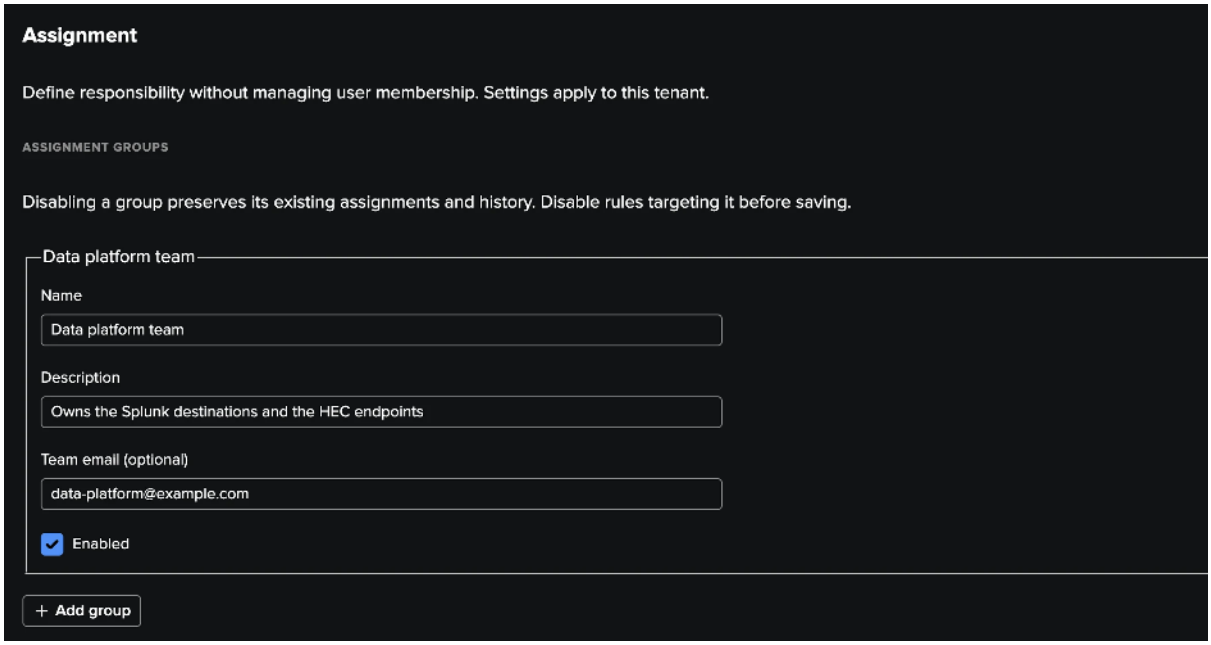
Assigning by hand

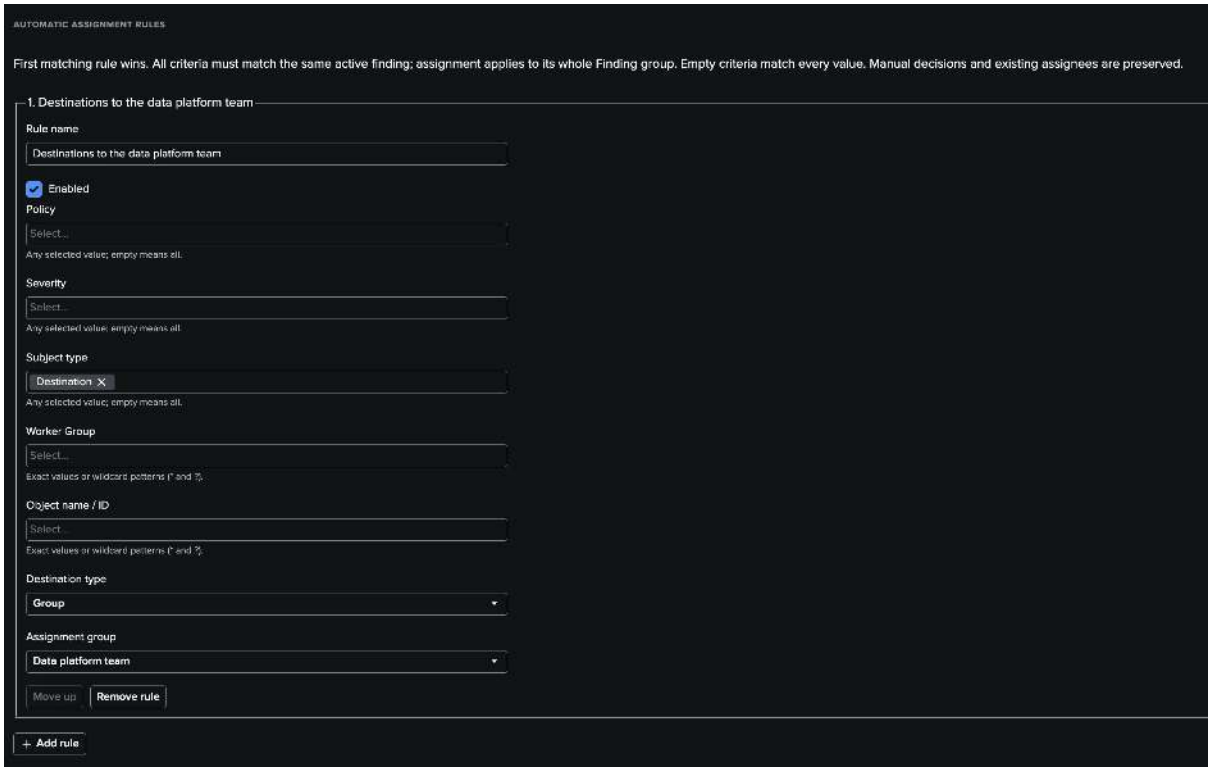
Assign to me, **Assign to…** and **Unassign** are in the finding and Finding group menus, the selection toolbar and the investigation. A manual assignment always wins over the rules; an explicit unassignment also holds until you choose **Use automatic assignment**. On UAM the user picker searches the accounts of the monitored environment; on the other dedicated tenant types you type the Splunk username. A group assignment shows the group name; an automatic one names its rule in a tooltip; the AI Agent keeps its purple **AI Agent** pill. The findings filters tell user and group assignments apart, including **Assigned to me**.

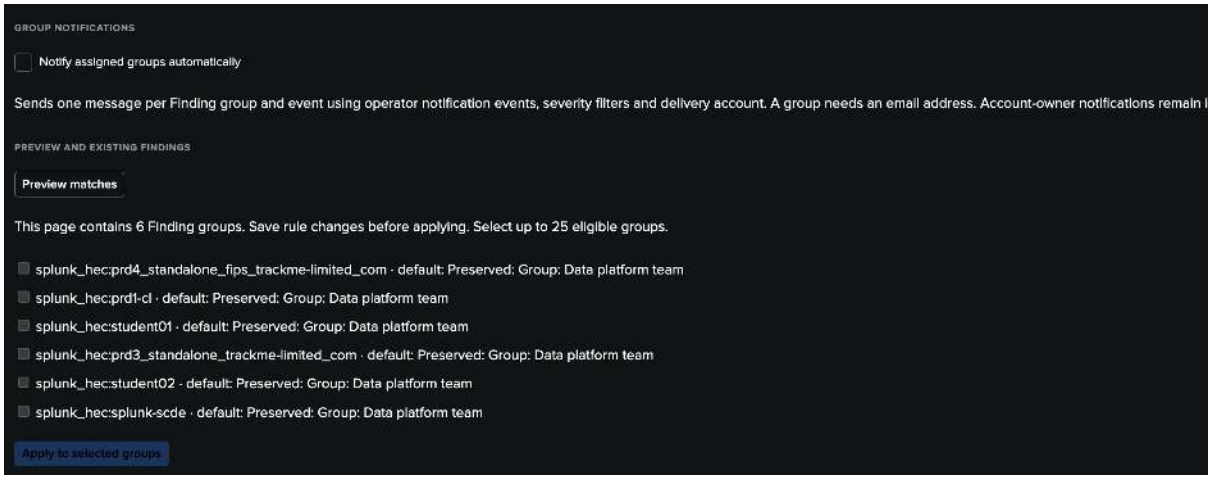
Team email

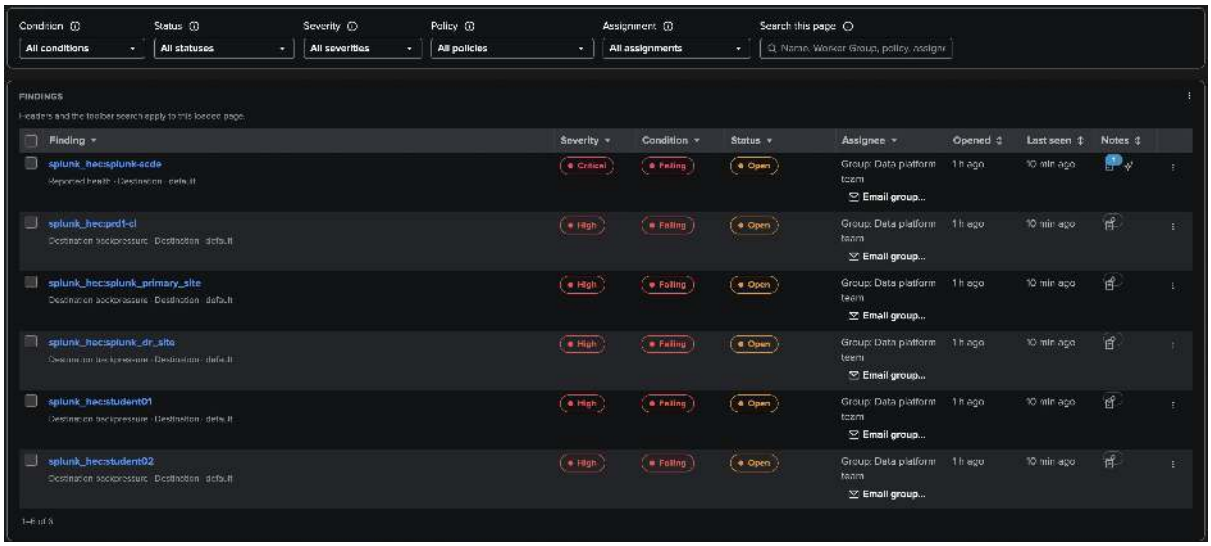
When **team email** is turned on, each Finding group event (with the event, severity and policy filters of the tenant's notifications) is mailed to the address of the assignment group it is assigned to — one message per Finding group and event, through the tenant's email delivery account. Snoozed Finding groups receive no automatic mail. You can also send a team email by hand: **Email group…** opens an HTML preview of the message and the findings it includes; **Confirm and send email** sends it. An email whose delivery could not be confirmed is reported as such and never resent automatically.

Who may do what: the assignment settings, the rule preview, **Apply to selected groups** and team email are for the TrackMe administrators of the tenant. Assigning a Finding group needs the triage permission on the other dedicated tenant types (the tenant's power and admin roles) and the administrator permission on UAM. Users who can only read the tenant see the group names, never their email addresses or the rules.




AUTOMATIC ASSIGNMENT RULES
First matching rule wins. All criteria must match the same active finding; assignment applies to its whole Finding group. Empty criteria match every value. Manual decisions and existing assignees are preserved.
1. Destinations to the data platform team
Rule name
Destinations to the data platform team
Enabled
Policy
Select...
Any selected value; empty means all.
Severity
Select...
Any selected value; empty means all.
Subject type
Destination
Any selected value; empty means all.
Worker Group
Select...
Exact values or wildcard patterns (* and %).
Object name / ID
Select...
Exact values or wildcard patterns (* and %).
Destination type
Group
Assignment group
Data platform team
Move up
Remove rule
+ Add rule


GROUP NOTIFICATIONS
Notify assigned groups automatically
Sends one message per Finding group and event using operator notification events, severity filters and delivery account. A group needs an email address. Account-owner notifications remain
PREVIEW AND EXISTING FINDINGS
Preview matches
This page contains 6 Finding groups. Save rule changes before applying. Select up to 25 eligible groups.
splunk_hec:prd4_standalone_fips_trackme-limited_com · default: Preserved: Group: Data platform team
splunk_hec:prd1-cl · default: Preserved: Group: Data platform team
splunk_hec:student01 · default: Preserved: Group: Data platform team
splunk_hec:prd3_standalone_trackme-limited_com · default: Preserved: Group: Data platform team
splunk_hec:student02 · default: Preserved: Group: Data platform team
splunk_hec:splunk-scde · default: Preserved: Group: Data platform team
Apply to selected groups


Condition
Status
Severity
Policy
Assignment
Search this page
All conditions
All statuses
All severities
All policies
All assignments
FINDINGS
Finding
Severity
Condition
Status
Assignee
Opened
Last seen
Notes
splunk_hec:splunk-scde
splunk_hec:prd1-cl
splunk_hec:splunk_primary_site
splunk_hec:splunk_dr_site
splunk_hec:student01
splunk_hec:student02
Group: Data platform team
Email group...
1 h ago
10 min ago

Notifications

Email notifications to operators

Each findings tenant can email its operators when findings change. Configure it in the workspace **Settings → Notifications** (UAM: **Notifications to operators**); it is off by default.

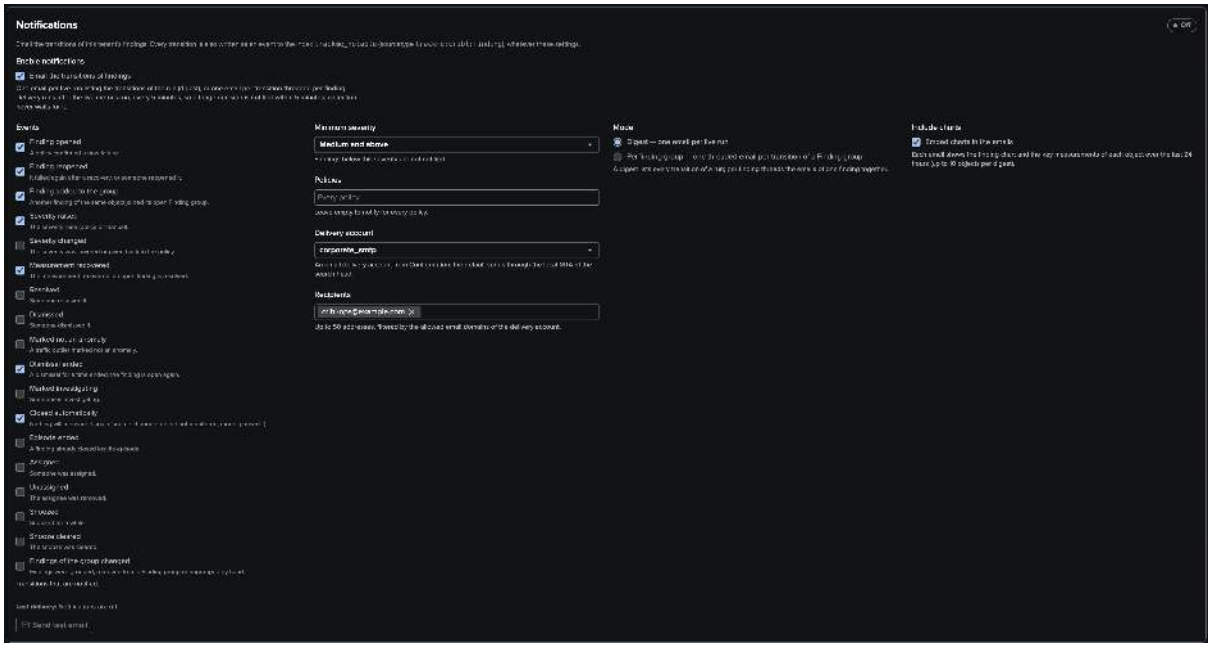
- **Events**: choose the transitions that send an email.
  - UAM: New finding, Finding reopened, Severity raised, Finding resolved, Finding dismissed, Finding assigned.
  - The other dedicated tenant types: Finding opened, Finding reopened, Finding added to the group, Severity raised, Measurement recovered, Dismissal ended, Closed automatically, and the operator decisions.
- **Minimum severity** (Medium by default) and **policies** (all by default).
- **Delivery account**: one of the TrackMe email delivery accounts (Configuration → Manage Email delivery accounts), and up to 50 **recipients**.
- **Mode**: a **digest** (one email per run) or one threaded email **per finding** (UAM) or **per Finding group** (every other dedicated tenant type, with the finding chart and the key measurements of the last 24 hours as inline images). In per Finding group mode, when more than 10 Finding groups have updates to send at once (a fleet event such as a patch wave or a site recovering, or a delivery catching up), they are sent as one digest that says so, never one email each.
- **Send test email** sends a test message with the saved settings, even while notifications are off.

Emails are sent by the tenant's collection job, never by the browser: within about 15 minutes on UAM (the policies run), after each collection run on the other dedicated tenant types. They link straight to the finding in the workspace. A delivery that keeps failing shows up in the **Collection health** tab.

Direct notifications to users (UAM)

UAM can also email **the account a finding is about**: on a new detection and when the condition cleared, with per-policy subjects and bodies (Policies → Edit → **Email template**), reply-to and support details, and rate limits. Human accounts are notified by default, service accounts on opt-in, system accounts never. **Send email communication** sends a reviewed, one-off message about a finding or a Finding group, with a preview and an explicit confirmation. See *User Activity Monitoring* for the details.

The other dedicated tenant types have no direct user notifications: each tenant type's own page describes its notification settings.



Findings actions

Findings actions turn a finding into the next step of your workflow: open the right search, jump to your runbook or CMDB, or call an external system. A TrackMe administrator of the tenant defines them once; everyone who works the findings then finds them in the menus, already filled in with the finding's values.

Type	What it does	Available on
Drilldown search	Opens Splunk Search with a search built from the finding (as the clicking user), with a time range around the finding, a fixed relative range, the workspace range or all time. Optional preview of the rendered search.	Every findings tenant
Link	Opens an external page (`https://`) or a Splunk page in a new tab: a runbook, a CMDB record, a Cribl UI page.	Every findings tenant
Webhook	Sends an HTTPS request through a *Webhook account*: to a ticketing system, a chat channel, an on-call service, a SOAR or another Splunk. Sent by hand, or automatically by automation rules.	Every findings tenant
Email	Sends the policy's Email template to the account a finding is about, after a preview and confirmation (by hand only).	UAM

Defining actions

Where the editors live

Tenant type	Settings → **Finding actions**	Policy editor tabs **Actions** and **Automations**
User Activity Monitoring	Every definition and every automation rule	Yes, for that policy
Cribl Stream	Every definition and every automation rule	Yes, for that policy
Cribl Edge, Splunk HEC	Every definition and every automation rule	Yes, for that policy
Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors	Every definition and every automation rule	— (use Settings → Finding actions; a rule still names its policies)

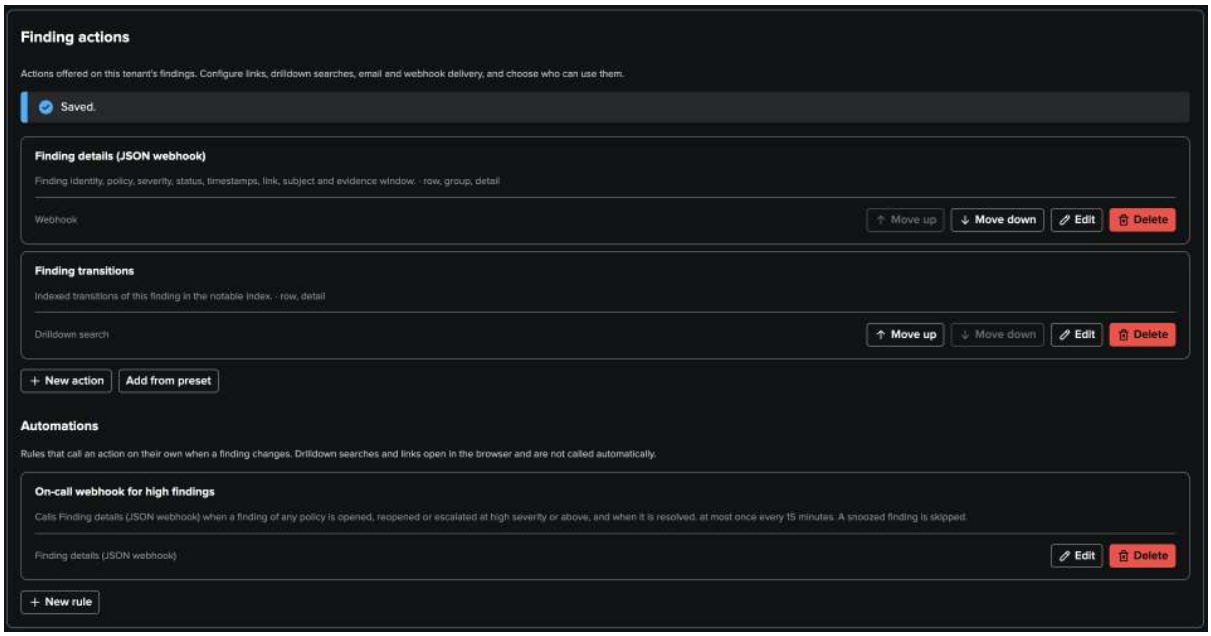
Add from preset inserts a ready-made definition, **disabled** until you review and enable it:

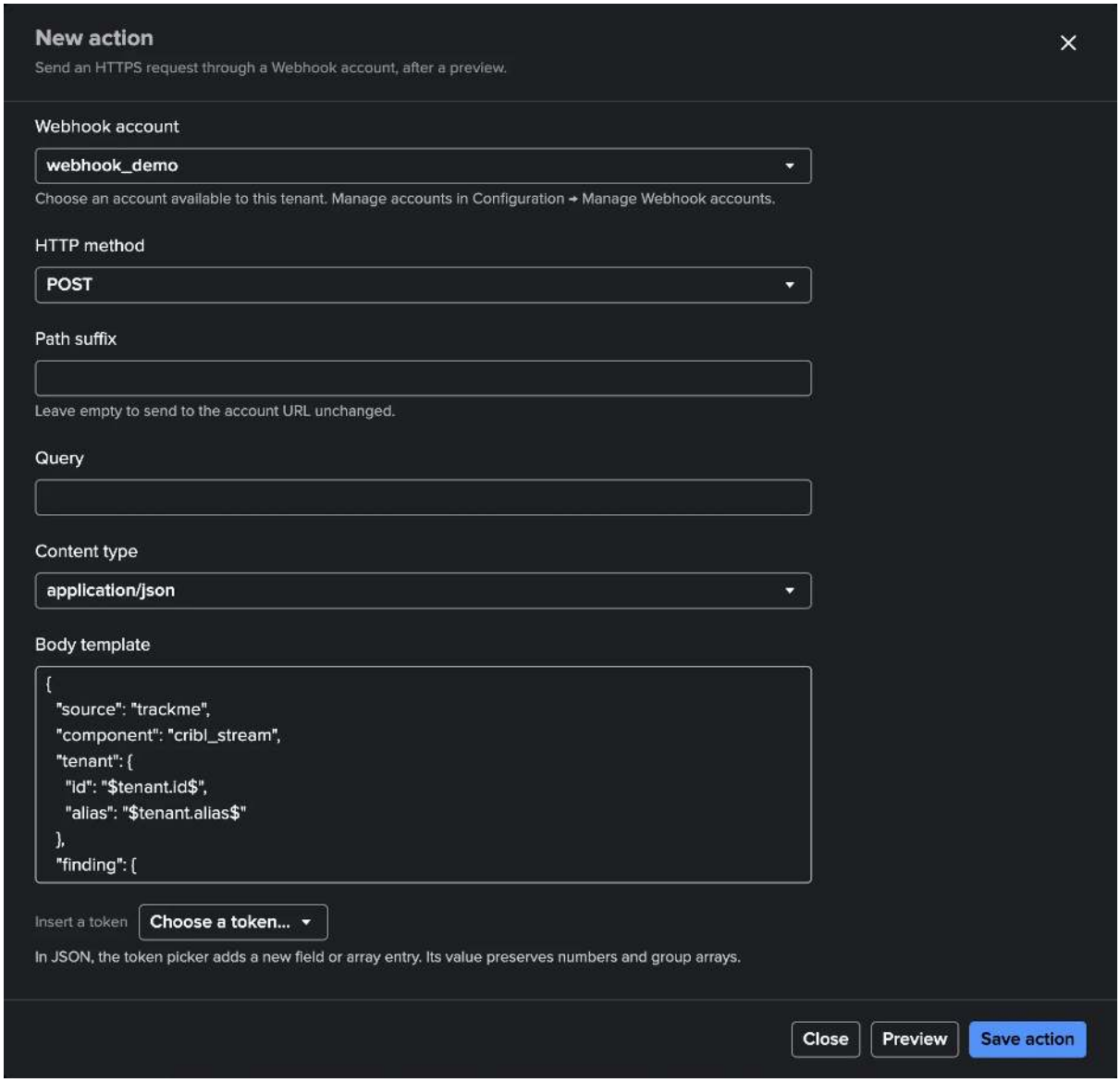
- UAM: *Audit activity of this account*, *Scheduler runs of this saved search*, *Finding transitions*;
- Cribl Stream: *Metrics of this object* (set your Cribl metrics index first), *Finding transitions*;
- Cribl Edge: *Finding transitions*, *Findings of this Fleet*;
- Splunk HEC and Splunk S2S: *Finding transitions*, *Findings of this pool*;
- Splunk UF Fleet and Splunk Forwarder Collectors: *Finding transitions*, *Findings of this group*;
- every tenant type: *Finding details (JSON webhook)* and *Finding summary (text webhook)*.

Every definition has a label, an optional description, and:

- a **scope** — the policies, subject types, severities and statuses it applies to (empty = all), so a "Open the Cribl Destination page" link is only offered on Destination findings;
- **where it appears** — the finding menu, the Finding group menu, the selection toolbar, and up to three buttons on the investigation;
- a **minimum role** — drilldowns and links are offered to everyone who can read the tenant, webhooks and emails to the roles that can triage; a definition can raise its minimum role, never lower it.

Up to 50 definitions and 100 automation rules per tenant. Drilldown searches refuse commands that write or send data (`delete`, `outputlookup`, `collect`, `sendemail`, `map` and similar) unless the definition explicitly allows them.





Tokens

A template refers to the finding with **tokens** written `$namespace.field$` — `$finding.title$`, `$subject.account$`, `$tenant.id$`. Write `$$` for a literal dollar sign. The editor's **Choose a token…** picker lists the tokens of the tenant type, and an unknown token is refused when you save.

Tokens	Available on
`tenant.id`, `tenant.alias`, `finding.id`, `finding.title`, `finding.policy_id`, `finding.policy_title`, `finding.severity`, `finding.status`, `finding.first_seen`, `finding.last_seen` (and `_iso` variants), `finding.group_id`, `finding.link` (the workspace link), `finding.assignee`, `finding.assignment_type` (`user` or `group`), `actor.user`, `action.label`, `action.request_id`, `now`	Every tenant type
`subject.type`, `subject.account`, `subject.owner`, `subject.app`, `subject.name`, `subject.environment`, `evidence.window_start`, `evidence.window_end` and the numeric evidence of the policy	User Activity Monitoring
`subject.worker_group`, `subject.entity_kind`, `subject.entity_id`, `subject.label`, `subject.kpi`, `finding.condition`, `connection.name`, `evidence.value`, `evidence.unit`	Cribl Stream
`subject.fleet` (instead of `subject.worker_group`), `subject.entity_kind`, `subject.label`, `subject.kpi`, `finding.condition`	Cribl Edge
`subject.pool` (instead of `subject.worker_group`), `subject.family`, `subject.entity_kind`, `subject.entity_id`, `subject.label`, `subject.kpi`,	Splunk HEC, Splunk S2S

Search text and secrets are never tokens.

Encoding is automatic. A finding's values come from names you do not control — a saved search title, a Cribl object name — so inserting them raw would let a crafted name inject SPL, break a URL or a JSON document. TrackMe encodes every token **for the place it sits in**: SPL string, URL path or query, HTTP header, JSON, plain text or form. You cannot turn this off. A few **filters** tell TrackMe how you want the value:

Filter	Meaning
`|s`	In SPL: the value as a quoted string, quotes included.
`|in`	In SPL: a list of values in parentheses, for `field IN …` on a Finding group or a selection.
`|n`	In SPL: the raw value, accepted only for numbers, times, identifiers and fixed lists of values.
`|json`	In a JSON body: the whole JSON value (a number stays a number, several values become an array).
`|default:<text>`	A fallback when the value is empty.

In SPL a string token must sit **inside double quotes**, or use `|s`. On a Finding group or a selection, a value the findings share stays one value; a value that differs becomes the list of the distinct values — use `|in` in SPL, an array in JSON; URLs and headers refuse it.

Example 1 — a drilldown search on a UAM account (the token inside quotes):

```
index=_audit action=search user="$subject.account$"
```

Example 2 — a drilldown on a whole Finding group, every finding of the group at once:

```
sourcetype="trackme:cribl:finding" tenant_id=$tenant.id|s$ finding_id IN
↪$finding.id|in$
```

Example 3 — a JSON webhook body with typed values and a fallback (the `"$…|json$"` string is replaced by the JSON value itself):

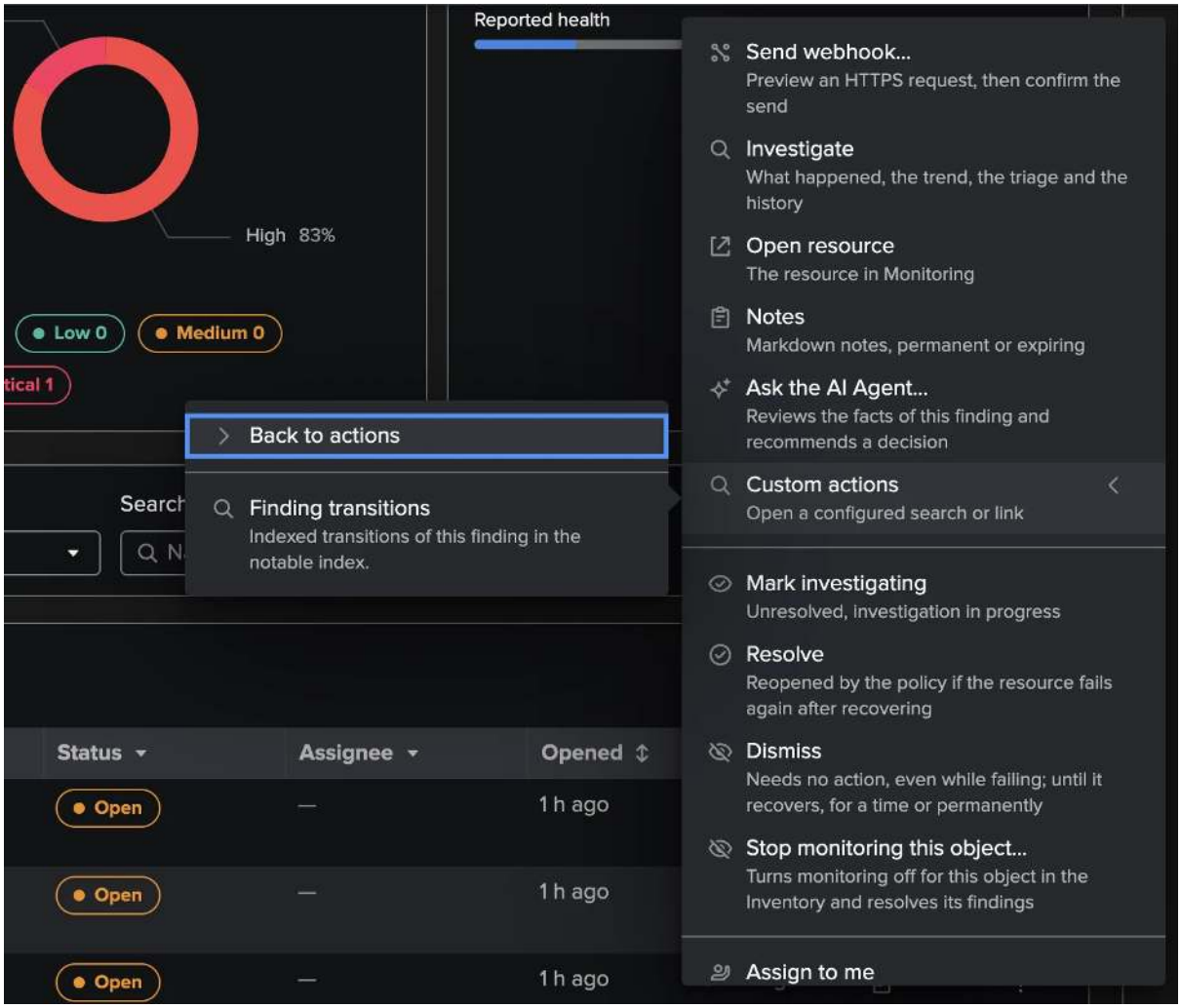
```
{
  "summary": "[TrackMe] $finding.title$",
  "severity": "$finding.severity|json$",
  "first_seen": "$finding.first_seen|json$",
  "owner": "$finding.assignee|default:unassigned$",
  "link": "$finding.link|json$"
}
```

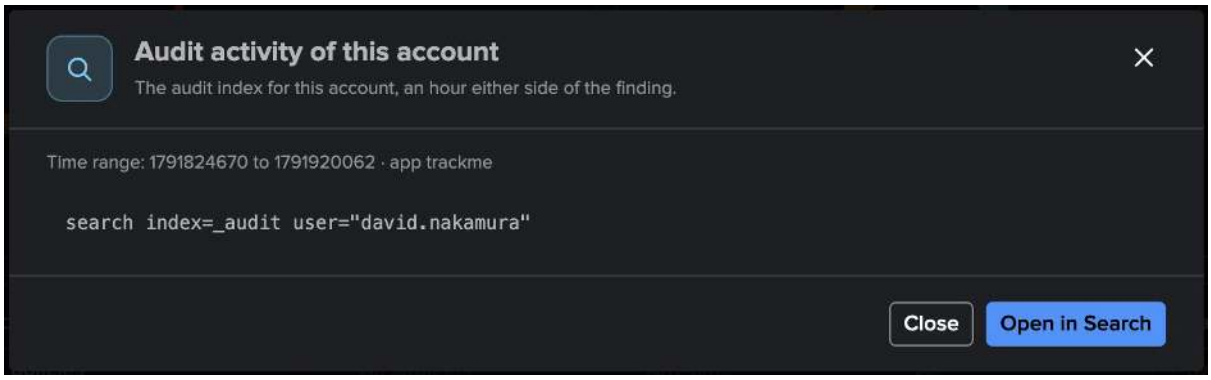
Running actions by hand

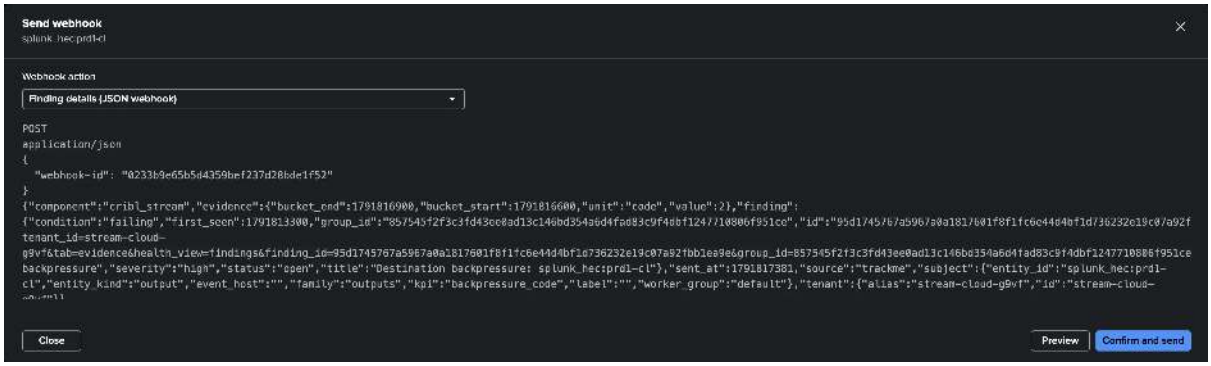
The finding menu and the Finding group menu carry a **Custom actions** submenu listing the actions whose scope matches and whose minimum role you hold. The investigation shows up to three drilldown and link buttons. A selection of rows offers **Run action** for drilldowns (up to 200 findings).

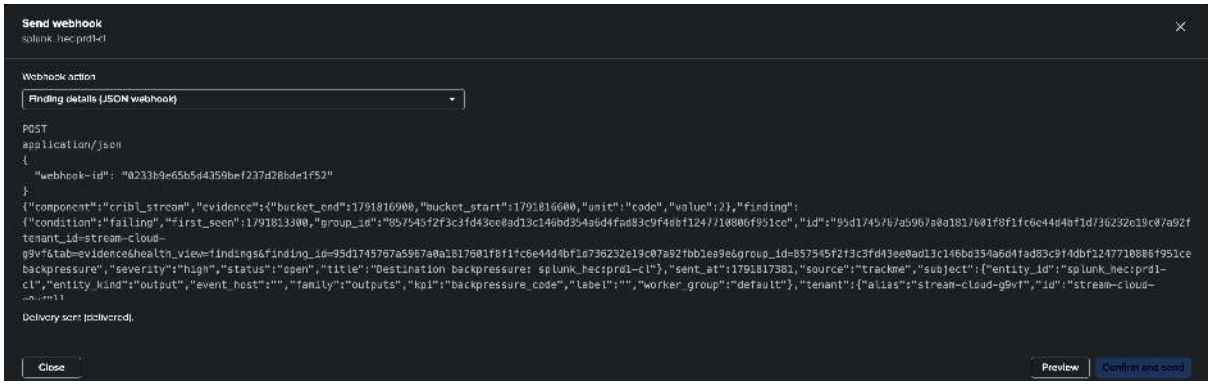
- **Drilldown search**: with preview on, the rendered search and its time range are shown, then **Open in Search**.
- **Link**: opens in a new tab.
- **Send webhook…**: shows the request as it will be sent (secrets masked), then sends it after you confirm, and follows its status: *sent*, *failed* or *unconfirmed* (the receiver may have received it; a manual send is never repeated automatically). On a Finding group, one request carries the group's values.
- **Email** (UAM): a preview of the message, an optional one-time recipient change, then confirmation.

Sending is done by a TrackMe search job, never by the browser or a REST request.


Reported health
High 83%
Low 0
Medium 0
Back to actions
Finding transitions
Indexed transitions of this finding in the notable index.
Send webhook...
Preview an HTTPS request, then confirm the send
Investigate
What happened, the trend, the triage and the history
Open resource
The resource in Monitoring
Notes
Markdown notes, permanent or expiring
Ask the AI Agent...
Reviews the facts of this finding and recommends a decision
Custom actions
Open a configured search or link
Mark investigating
Unresolved, investigation in progress
Resolve
Reopened by the policy if the resource fails again after recovering
Dismiss
Needs no action, even while failing; until it recovers, for a time or permanently
Stop monitoring this object...
Turns monitoring off for this object in the Inventory and resolves its findings
Assign to me
Status
Assignee
Opened
Open
1 h ago


Audit activity of this account
The audit index for this account, an hour either side of the finding.
Time range: 1791824670 to 1791920062 · app trackme
search index=_audit user="david.nakamura"
Close
Open in Search


Send webhook
Webhook action
Finding details (JSON webhook)
POST
application/json
Close
Preview
Confirm and send



Automation rules

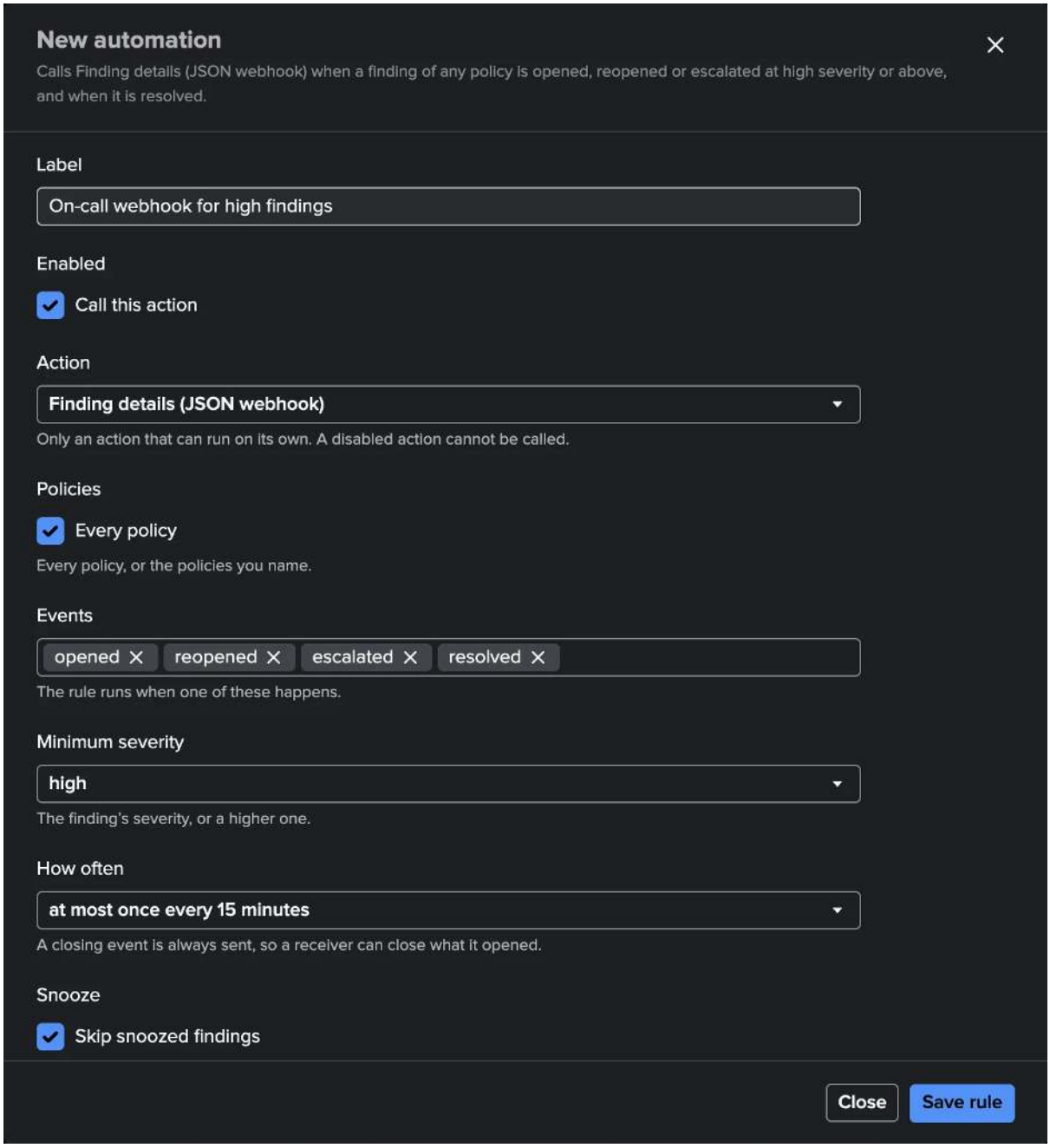
An **automation rule** sends a webhook definition automatically when a finding changes.

Field	Meaning
Which policies	The policies the rule applies to, or every policy.
Events	The transitions that trigger it (see below).
Minimum severity	Low (default) to Critical.
Action	The webhook definition it sends (only webhooks can be automated).
How often	The throttle per finding: from 0 to 86,400 seconds, 900 by default.
Snooze	Respect snooze (default): a snoozed finding sends nothing.

Tenant type	Events
User Activity Monitoring	new, reopened, escalated, resolved, dismissed, assigned
Every other dedicated tenant type	opened, reopened, member added, escalated, recovered, resolved, dismissed, dismissal expired, closed

Throttling. A rule sends at most once per finding per throttle period. **Closing events** (resolved, dismissed — and, outside UAM, recovered and closed) are never throttled, so the receiver can always close what it opened.

The queue. Matching transitions are queued in the tenant (at most 5,000 deliveries) and sent by the tenant's own collection job: the policies run on UAM (about every 15 minutes), the collection runs on the other dedicated tenant types (every 5 minutes). Each run sends at most 200 deliveries; a failed delivery is retried up to 6 times with an increasing delay (30 seconds to 20 minutes) and dropped after 24 hours. Every automatic send carries a stable `webhook-id` header, so the receiver can recognise a retried request it already processed. Disabling or deleting a definition disables the rules that use it.



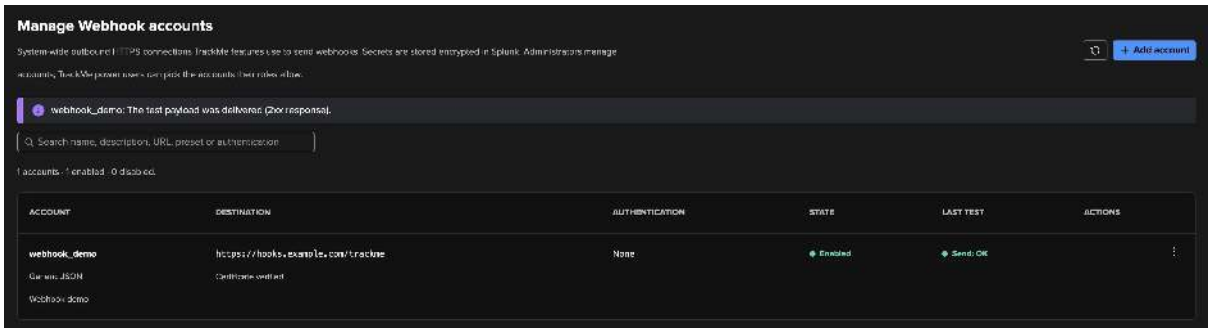
Checking deliveries

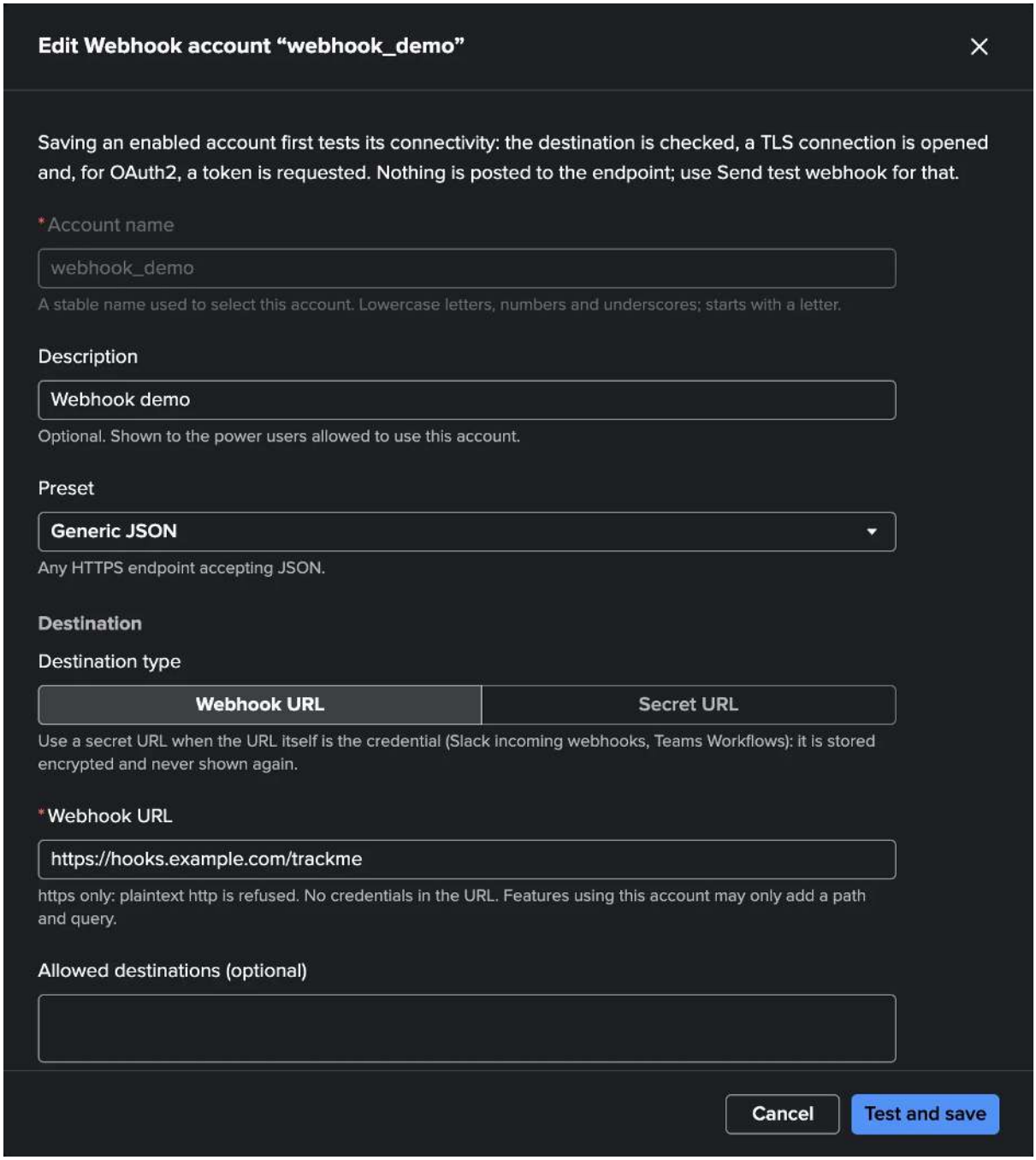
- The Configuration Guardian check **findings_action_delivery_failing** warns when a queued delivery of a tenant was attempted three times or waited more than 30 minutes — see *Configuration Guardian*.
- Manual sends are logged in `index=_internal sourcetype=trackme:custom_commands:trackmefindingsaction`.
- On UAM, the receipt of each policies run (`sourcetype=trackme:uam:receipt`) carries the counts of the automatic sends.
- Changes to definitions, rules and assignment settings are audited in the TrackMe audit index.

Webhook accounts

A **Webhook account** is a reusable, **system-wide** outbound HTTPS connection: where to send, how to authenticate, which certificates to trust, through which proxy. Webhook definitions only name the account and add a path, a query, headers and a body — so the destination and its credentials are managed once, by an administrator, and never copied into each tenant.

Manage them in **Configuration → Manage Webhook accounts**.





A receiver the stack cannot reach

From Splunk Cloud, a webhook receiver on your own network cannot be reached directly. A *TrackMe Broker* solves this: an add-on on a Heavy Forwarder on your side carries the deliveries. Every connection starts on your side, so you open nothing inbound.

Set up a broker, and on its forwarder declare the receiver as a target. A target's URL can hold a path, for example `https://hook.example.com:8443/api/hooks`. If the account uses OAuth2, a target must also cover its token endpoint. Then set **Connectivity** on the account to **Through broker** and that broker's name.

The list of accounts shows **Broker** and its name under **Connection type**.

Fields

Field	Notes
Account name, **Description**, **Account state**	The name is how definitions select the account (lowercase letters, digits and underscores). A disabled account is never used and saves without a connectivity test.
Preset	Generic JSON, Slack, Microsoft Teams Workflows, PagerDuty Events v2, ServiceNow, Splunk HEC, Splunk SOAR, Cribl HTTP source. A preset pre-fills the empty fields with the receiver's usual settings; it is informational afterwards. The message itself (a Slack text, a Teams card, a PagerDuty event) is written in the webhook definition, starting from the JSON preset.
Connectivity	**Direct**, or a *TrackMe Broker*, for a receiver this instance cannot reach. Through a broker, the broker's target decides what is reached and how its certificate is verified, and the account's own certificate, proxy and allowed-destination options are hidden.
Webhook URL or **Secret URL**	HTTPS only: `http://` is refused, and so are credentials in the URL. Use a **Secret URL** when the URL itself is the credential (Slack, Teams): it is encrypted and never displayed again. The host is fixed: a definition may only append a path and a query.
Allowed destinations	Optional: host names, `*.example.com`, IP addresses and CIDRs. Addresses in private, loopback, link-local and other special-purpose ranges are refused unless an allowed IP or CIDR covers them; cloud metadata, multicast and reserved addresses are always refused, and so are the Splunk management ports of non-public addresses.
Authentication type	None, Basic, Bearer token, API key header (header name and an optional one-word prefix such as `Splunk` for HEC), HMAC signature (Standard Webhooks or GitHub SHA-256), OAuth2 client credentials (token URL, client ID, client secret, optional scope and audience). Secrets are stored in Splunk's encrypted storage.
Verify SSL certificate, **CA certificate chain**	On by default. For a private CA or a self-signed certificate, keep verification on and add the CA chain (paste it or **Upload PEM file**). Turning verification off keeps the traffic encrypted but stops checking the destination's identity.
Proxy URL, user name, password	Optional, `http://` or `https://`. The request stays TLS to the destination inside the proxy tunnel. Environment proxies are never used.
Default headers	A JSON object of non-secret headers sent with every request; reserved and credential headers are refused.
Timeouts, **Maximum retries**, **Retried HTTP statuses**, **Maximum body size**	Connect 5 s (up to 30), read 15 s (up to 60), 2 retries (up to 5) on 429, 502, 503 and 504 by default, bodies up to 256 KiB (up to 1 MiB). Redirects are never followed.
Allowed roles, **Allowed tenants**	Who may use the account: comma-separated roles whose TrackMe power users may select it (empty: every TrackMe power user; TrackMe administrators always may), and the tenants allowed to use it (empty: every tenant). Checked when a definition is saved and again before every send.

Every address a destination resolves to is checked before **every** attempt, and the connection is pinned to the address that was checked. Through a proxy the proxy resolves the name, so the name is checked but the address cannot be pinned.

Testing an account

- **Saving** an enabled account runs a connectivity test — destination checks, the TLS handshake and, for OAuth2, the token request — and **sends nothing** to the receiver (a test message in a Slack channel or a PagerDuty incident would be noise). A failure rejects the save with its reason; save the account **disabled** to keep it while you fix the destination.
- **Test connectivity** in the row menu runs the same test again.
- **Send test webhook** (with a confirmation) posts one `trackme.webhook.test` message, so you can see what the receiver gets. **View last result** shows the outcome.

Who manages and who uses them

- TrackMe **administrators** (`trackmeadminoperations`, Splunk `admin` or `sc_admin`) create, edit, test and delete Webhook accounts.
- TrackMe **power users** only see the list of accounts their roles and tenant may use, without any secret, when they edit or send a webhook definition.
- Other users see nothing.

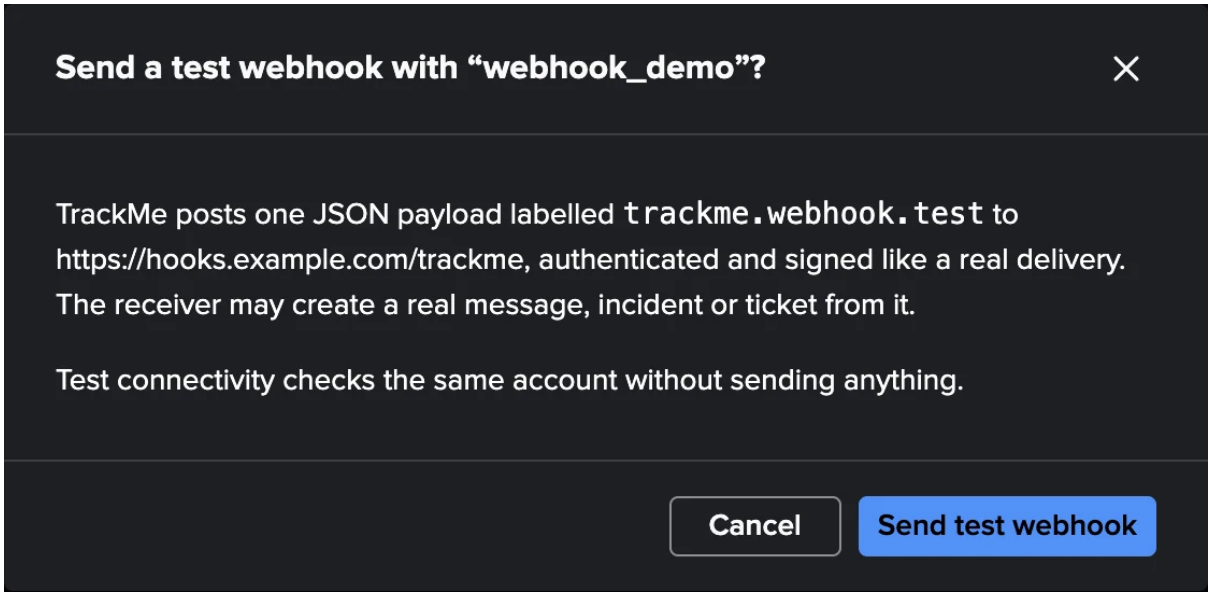
Warning

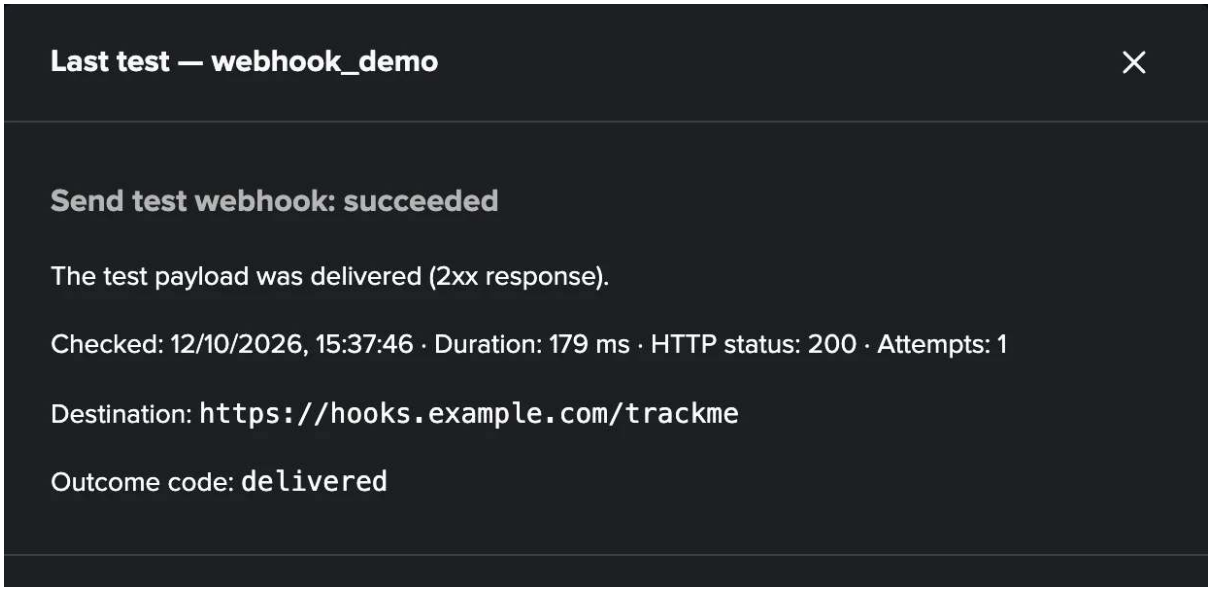
With **Verify SSL certificate** turned off on an account, traffic stays encrypted, but TrackMe does not check the destination's identity. Turning verification off is allowed on every platform, Splunk Cloud included, and TrackMe raises no alert for this choice. To keep verification on with a private CA, add the CA chain of the destination.

Worked example: from an account to an automatic webhook

This example uses a test endpoint such as a request-inspection service, which shows every request it receives — the quickest way to see exactly what TrackMe sends before you point it at a real receiver.

1. **Create the account.** In Configuration → Manage Webhook accounts, **Add account**: name `webhook_demo`, preset **Generic JSON**, the HTTPS address of your test endpoint as **Webhook URL**, authentication **None**, **Verify SSL certificate** on. Save: the connectivity test runs, and the row reads *Connectivity: OK*.
2. **Check what the receiver gets.** Row menu → **Send test webhook** → confirm. The test endpoint shows a `trackme.webhook.test` JSON message.
3. **Define the action.** In a Cribl Stream workspace, Settings → **Finding actions** → **Add from preset** → *Finding details (JSON webhook)*. Select the Webhook account `webhook_demo`, review the body template, enable the definition and save.
4. **Send by hand.** On a finding, open **Custom actions** → **Send webhook…**, review the request preview and confirm. The test endpoint shows the finding as JSON: its id, title, policy, severity, status, subject, condition and the link back to the workspace.
5. **Automate.** Open a policy → **Automations** → **New rule**: events *opened* and *recovered*, minimum severity Medium, the action you just created, the default throttle. The next time the policy opens or recovers a finding, the live collection run sends it.
6. **Watch the deliveries.** If the endpoint stops answering, the deliveries are retried and the Guardian check **findings_action_delivery_failing** warns after three attempts or 30 minutes.





Findings AI Agents

An **AI Agent** reviews findings the way an operator would: it reads the evidence TrackMe collected, writes an assessment as a note, and — when you allow it — applies the decisions the tenant permits. Seven agents, one per dedicated tenant type, share one framework:

Tenant type	AI Agent
User Activity Monitoring	**UAM AI Agent** — it replaces the UAM Findings Advisor of TrackMe 2.4.18; on upgrade, an enabled advisor's settings are carried over to the agent.
Splunk HEC	**Splunk HEC AI Agent**
Splunk S2S	**Splunk S2S AI Agent**
Splunk UF Fleet	**Splunk UF Fleet AI Agent**
Splunk Forwarder Collectors	**Splunk Forwarder Collectors AI Agent**
Cribl Stream	**Cribl Stream AI Agent**
Cribl Edge	**Cribl Edge AI Agent**

Inspect and act

Mode	What it does
Inspect	Reads the evidence and writes **one current note per finding**, signed `[AI Agent]`, with its assessment, confidence and recommendation. It changes nothing. A later review refreshes the note when the assessment is unchanged and replaces it otherwise.
Act	Inspects, then applies the decisions the evidence supports **and** the tenant settings allow. Act must be chosen explicitly.

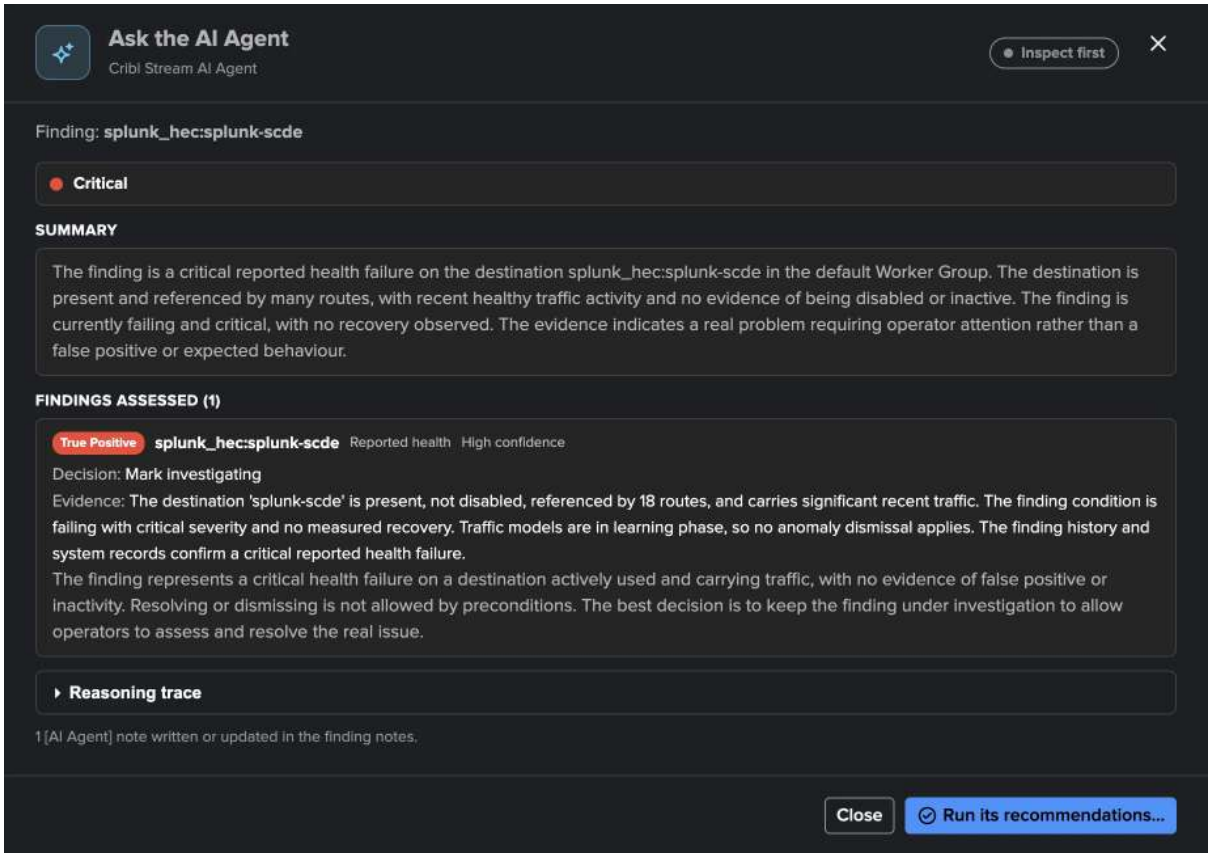
What act may do, depending on the settings:

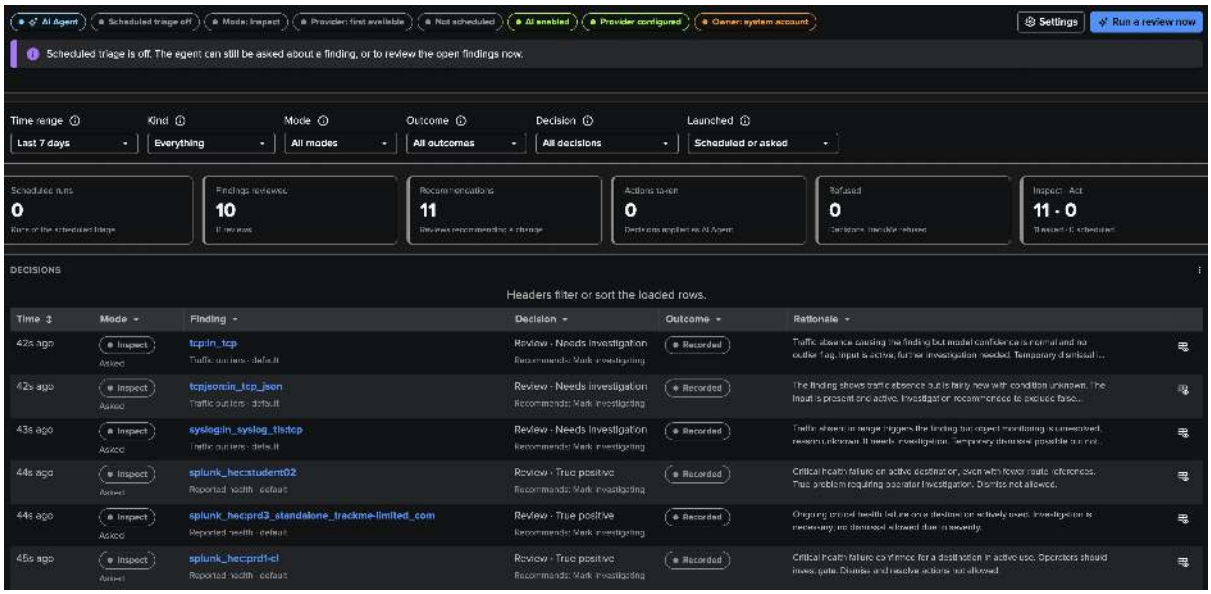
Decision	Default	UAM	Cribl Stream	Cribl Edge	Splunk HEC, S2S, UF Fleet, Forwarder Collectors
Mark investigating	On	Yes	Yes	Yes	Yes
Dismiss for a time (up to 7 days by default, at most 90)	On	Yes, never a critical or high finding still observed	Yes, a critical failing finding only with proof that the object is disabled, missing or inactive	Yes, a critical or high finding still failing only when its node or Fleet is gone from the node directory	Yes, a critical or high finding still failing only when its object is gone from the directory or the inventory
Not an anomaly (a traffic outlier)	On	—	Yes	—	—
Resolve	Off	Only a finding its policy no longer observes	Only after a measured recovery	Only after a measured recovery	Only after a measured recovery, with no other member of its group failing
Classify an account	Off	Only an account without an operator class	—	—	—
Stop monitoring an object	Off	—	Only with proof of inactivity (7 days by default)	—	—

The agent **never** dismisses permanently, assigns or unassigns, changes a severity, or changes a policy or a setting. Every decision it applies is recorded under the actor **AI Agent** (a purple pill in the history), never under the person who asked; the decision log also records who asked, the AI provider and whether the run was scheduled.

Asking the AI Agent

Ask the AI Agent… is in the finding and Finding group menus, the investigation and the **AI Agent** tab. The agent first inspects; you read its assessment and can then let it **act** on that inspection within 30 minutes. The AI Assistant of the workspace can also propose an AI Agent review in the chat. Asking needs AI to be enabled with at least one AI provider; on UAM it is reserved to the TrackMe administrators of the tenant, on the other dedicated tenant types to the roles that can triage.





Scheduled reviews

The agent can also review the open and investigating findings every hour on its own. This is **off by default**. A scheduled run only starts when:

- the AI Assistant is enabled and at least one **AI provider** is configured;
- the tenant is enabled and its AI Agent is enabled (Settings → **AI Agent**);
- the tenant owner is a real account, not `nobody` or `splunk-system-user` — the Guardian check **ai_automation_owner_is_system_user** warns otherwise. On Cribl Stream, Cribl Edge and Splunk HEC the owner of the scheduled report must also be a Splunk administrator.

Each run reviews up to 10 findings (one per Finding group, most severe first), skips the findings it reviewed recently, and **never overrides a person**: a finding or a Finding group that an operator decided on, or that has an assignee, is left to the operators. By default the scheduled selection only includes **critical and high** findings: the shared filters of the automated AI reviews apply, with the finding severity as priority.

Settings

In the workspace **Settings → AI Agent** (each agent is also in **Manage AI Agents automation**, for enabling, mode, provider, review interval, runtime and custom instructions):

Setting	Default	Notes
Enabled	Off	Scheduled reviews only; asking the agent needs only AI to be available.
Mode	Inspect	Inspect or act, for scheduled runs.
AI provider	First available	
Review interval (days)	1	A finding is not reviewed again within this interval unless it changed.
Maximum runtime (seconds)	3000	Fits the hourly cadence.
Findings per run	10	1 to 50.
Custom instructions	Empty	Plain English the agent follows on every run (at most 4000 characters).
Allowed decisions and longest dismissal	See above	Per decision.

Custom instructions tell the agent what to treat as expected, what to look at first and how to word its recommendations, for example "The accounts named `svc_*` are service accounts." They never widen what it may do: the allowed decisions and the checks TrackMe makes before every decision still decide. **Evaluate…**, next to the field, runs the agent in inspect mode with an unsaved draft, on one open finding or a short review, optionally side by side with a run without instructions; an evaluation writes nothing (see *AI Agents for findings*).

Everything the agents do is in the **AI Agent** tab, the finding history, the `[AI Agent]` notes and the **Review AI Advisor activity** and **Review Scheduled AI Advisor activity** dashboards. The AI side of the agents — how a review works, the evidence, the data sent to the provider and the audit — is in *AI Agents for findings*. See also *The AI Advisors* and *AI Assistant & Concierge*.

The shared workspace conventions

The seven findings workspaces look and behave alike, so what you learn on one applies to the others.

Table header filters and sorting
: Text and category columns have a filter in their header: search the values, select exact values, or type a "contains" match. Numeric and date columns sort, with empty values always last. On the Findings tab these filters act on whole Finding groups. On tables loaded page by page (Cribl Edge nodes, UAM catalogs), header controls act on the loaded page, while the toolbar filters search everything. Header controls never change data or permissions.

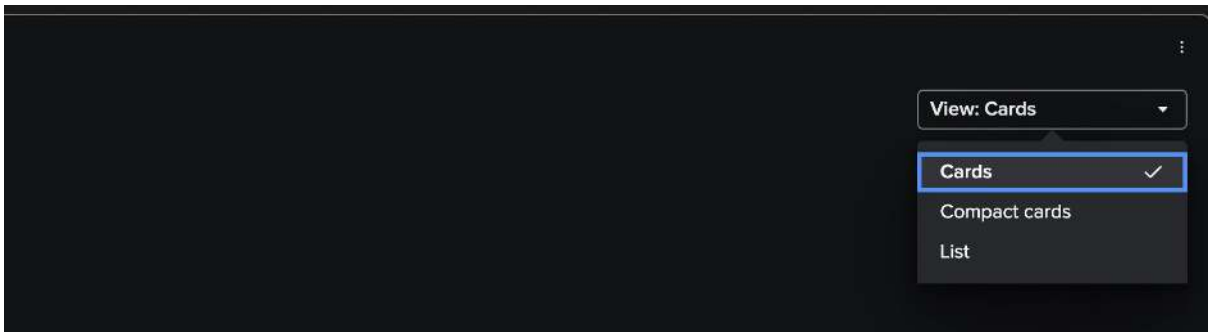
Policy view modes
: The Policies tab shows **Cards** (default), **Compact cards** or a sortable **List**. The choice is remembered per user and per tenant type (Cribl Stream and Cribl Edge share it) and writes no configuration.

Menus
: Finding, Finding group and selection menus stay within the screen and scroll. **Set severity** and **Custom actions** are submenus (Right Arrow opens, Left Arrow returns). An entry you cannot use stays listed, disabled, with its reason.

One Save in Settings
: Every Settings section — Notifications, Assignment, Finding actions, AI Agent and the tenant's own sections — is saved with one **Save**, **Discard** or **Reload**.

Deep links
: Notification emails and the `finding.link` token open the finding (or its Finding group) in place in the workspace; Cribl Edge links can also open a node or a Fleet.





8.3.18 Cribl Stream and Cribl Edge

TrackMe monitors Cribl with two **dedicated tenant types**, generally available since TrackMe 2.4.19:

- a **Cribl Stream** tenant follows one Cribl Stream deployment end to end: the Leader, the Worker Groups, the Workers and their machines, and the Sources, Destinations, Pipelines and Routes, with traffic machine learning models on every data object;
- a **Cribl Edge** tenant follows the Fleets of a Cribl Edge Leader: connectivity, version and configuration drift, the health of the Sources and Destinations reported by the nodes, and the disk, CPU and memory of every Edge Node, at any fleet size.

Like User Activity Monitoring, a Cribl tenant has no entities: its policies raise **findings**, which you triage, group, assign, act on and notify — see *Findings, Finding groups and findings actions*.

The Cribl documentation lives with the third party integrations:

Cribl Stream & Cribl Edge Which Cribl tenant to use, how they work, and the Cribl connections.

Cribl Stream & Cribl Edge Cribl Stream Create a Cribl Stream tenant and explore its workspace.

Cribl Stream Cribl Edge Create a Cribl Edge tenant and explore its workspace.

Cribl Edge To monitor the data sources and hosts whose logs pass through Cribl, with the DSM and DHM components, see *Monitor data sources and hosts with Cribl signals*.

A self-managed Leader that this instance cannot reach, such as from Splunk Cloud, can go *through a TrackMe Broker*.

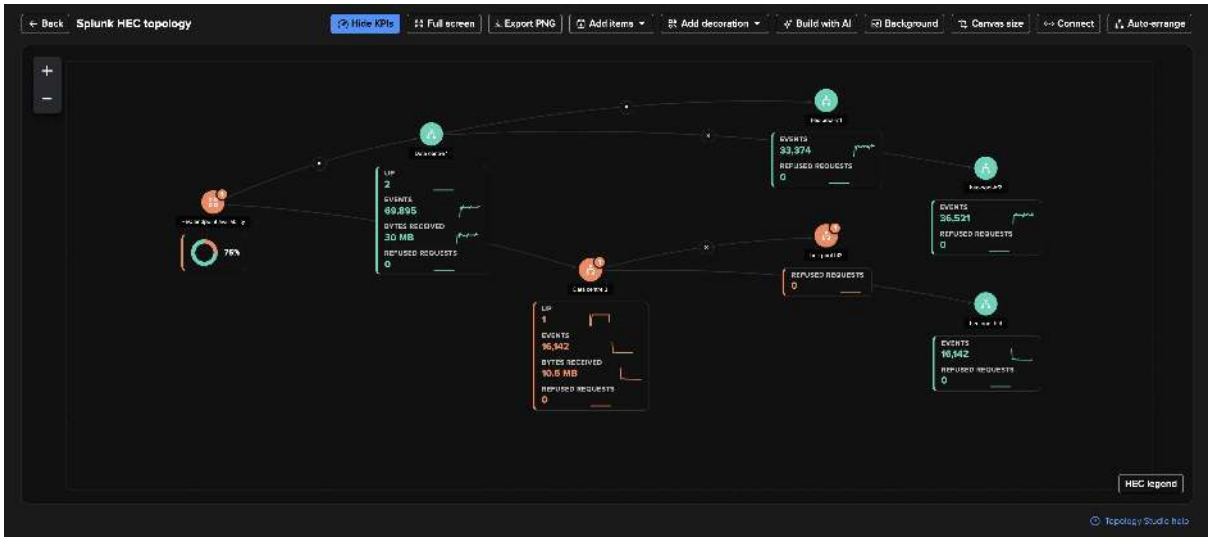
8.3.19 Splunk HEC

The **Splunk HEC tenant** monitors the **HTTP Event Collector** (HEC) endpoints of one Splunk deployment: the heavy forwarders, indexers or Splunk Cloud indexers that receive HEC traffic, the **pools** they form behind a load balancer, and the **tokens** senders use. It answers the questions a Splunk administrator asks about HEC: *are all my HEC endpoints up and receiving? are senders being refused, and why? where is the bottleneck when HEC answers "Server is busy"? which tokens send data Splunk rejects? did a business-critical token stop sending? is my load balancer spreading the traffic over every member of the pool?* The answers are **findings**, raised by built-in policies, with the same triage, Finding groups, notifications, assignment and findings actions as the other dedicated tenants, and a findings AI Agent of its own, the Splunk HEC AI Agent.

A Splunk HEC tenant is a **dedicated tenant type**: it has no entities and no state colour, and one tenant follows the HEC endpoints of one Splunk deployment. It reads the telemetry Splunk **already writes** about HEC — the HEC metrics in `_introspection` and the HTTP input, queue and output messages in `_internal` — on the local

search tier or through a TrackMe remote account. Nothing is installed on the endpoints, no new data is ingested, and TrackMe never changes a HEC setting, a token or anything else on Splunk: the tenant only reads.

The Splunk HEC tenant is generally available since TrackMe 2.4.21.



The view above is the **HEC topology** of a deployment with two pools of two heavy forwarders, generated from the workspace in one click and saved to Topology Studio (see *Generated topologies*). One heavy forwarder of *Data centre 2* was stopped: the endpoint and its pool turned orange with the findings they opened, and the endpoint availability fell to 75 %.

For a guided tour on a live deployment, see the hands-on walkthrough *Splunk HEC*.

What it monitors

Object	What TrackMe watches
Endpoints	Every Splunk instance whose HEC heartbeat reaches the search tier. **Availability** (an endpoint writes its HEC metrics every minute, traffic or not: no heartbeat means it is down), **refused requests** (server busy, refused tokens, rejected data), **back pressure** along its ingestion pipeline and the stage that cannot keep up, its **output** to the indexers, **indexer acknowledgment** under pressure or never polled, and optionally its host's **CPU and memory**.
Pools	The endpoints behind one load balancer or with one role. **Capacity** (members up against the expected size), **imbalance** (a member up but starved of traffic), a **traffic drop** against the same hour of the previous days, **traffic outliers** against a traffic model, and, between the members, **token consistency** and **configuration drift** (Splunk versions, global HEC settings).
Tokens	The HEC inputs by name, whatever pool receives them. **Rejected data** (invalid format, incorrect index, missing or blank event, indexed fields, oversized events), **silence** of the tokens you select, and **traffic outliers** against each busy token's traffic model.

A token is always known by its **name**: TrackMe never reads, stores, logs or shows a token value.

How it works

Splunk's own telemetry is the source. Every HEC instance writes, every metrics interval (60 seconds by default, `limits.conf [http_input] metrics_report_interval`), its HEC metrics to `_introspection` (`sourcetype=http_event_collector_metrics`): an instance series, written even when the endpoint is idle (a heartbeat), and a series per token while the token receives data. It logs each refused request to `splunkd.log` (`component=HttpInputDataHandler`, with the token name, the client address and the HEC reply code), and its queues, thread activity and forwarding to `metrics.log`. Splunk forwards its internal indexes by default, so TrackMe sees every HEC endpoint with **no agent, no network path to the endpoint and no credential to it**.

Bounded searches, whatever the fleet size. The collection runs a fixed number of searches per run, aggregated by host and token, over what Splunk **indexed** since the previous run (by index time, in settled 5-minute windows): its cost does not grow with one search per endpoint.

Job (Collection health label)	What it does
HEC activity (every 5 minutes)	Reads the last settled 5-minute window: the HEC metrics of every endpoint and token, the refused requests by reply code, token and client, the ingestion queues and their threads, the output and its receivers, and the host's CPU and memory. It publishes the window (endpoints, pools, tokens), writes the history points to the tenant's TrackMe metric index, then evaluates the policies (**Monitoring policies**) and sends the notifications.
Endpoint directory (every 10 minutes; rebuilt about once an hour)	The endpoints (those seen in the last 24 hours, kept while they reported in the last 7 days) and the token names seen in the last 24 hours, their identity (Splunk version, OS, CPUs, memory) and the pool each one joins. With remote accounts bound, it also reads the optional splunkd API (see *The splunkd API enrichment*).
Traffic models (twice an hour)	Learns the incoming traffic of each pool and token per hour of the week, from the HEC metrics (see *Traffic models*).
Monitoring policies (after every HEC activity run; on demand)	Evaluates the policies on the newest window. **Run now** in Collection health evaluates them again.

Endpoint states. In each window an endpoint is **up** (a heartbeat in the window), **late** (no heartbeat this window, but one in the last 5 minutes), **stalled** (no heartbeat, and its output was in trouble: its own telemetry waits behind the same blocked output) or **silent**. An endpoint silent for longer than the dormancy period of Endpoint availability (7 days by default) is **dormant**, a retired instance: it opens nothing. The directory keeps the endpoints that reported in the last 7 days.

Why "Server is busy". HEC writes every request into one input queue, shared by every token, then into the ingestion pipeline (parsing, merging, typing, indexing) and, on a heavy forwarder, the output to the indexers. When a stage cannot keep up its queue fills, then the queues before it, until HEC's own queue is full and HEC refuses requests with HTTP 503. The tenant reads every queue of that path and names the **bottleneck**: the stage after the furthest-downstream saturated queue, or the **output** when the backup reaches the indexing queue and an output queue is saturated too, and whether that stage's thread was busy (working at its limit) or waiting on the next stage.

Before you start

A licence. A valid **Enterprise**, **Unlimited**, Unlimited trial or **Developer** licence. The Splunk HEC tenant is not available with the Foundation edition or the Foundation trial.

A source. The search tier that sees the HEC endpoints' internal indexes:

- **local**, this search tier, when it searches the indexers that hold the endpoints' `_internal` and `_introspection`;
- or a **TrackMe remote account** (*Remote Splunk deployments*) to the search tier of another deployment, a Splunk Cloud stack included. The tenant's scheduled searches run as the tenant owner, so the **tenant owner must hold one of the remote account's roles**: the wizard and the Settings refuse the account otherwise (`remote_account_refused`).

Which components need a remote account? shows, for every component, when a remote account is needed and where it points.

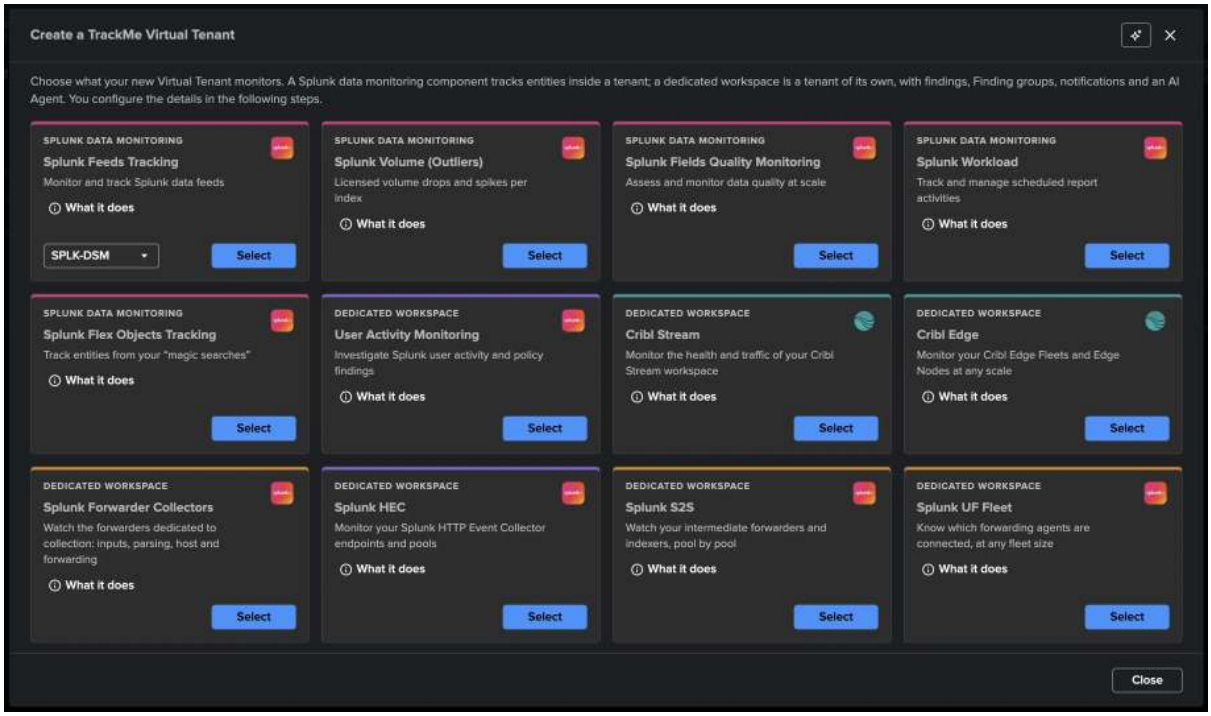
HEC telemetry that reaches the indexers. The endpoints forward their `_internal` and `_introspection` indexes (the default forwarding of internal indexes does it). On Splunk Cloud, `_introspection` and `_internal` are searchable and HEC runs on the indexers; the tenant works the same way.

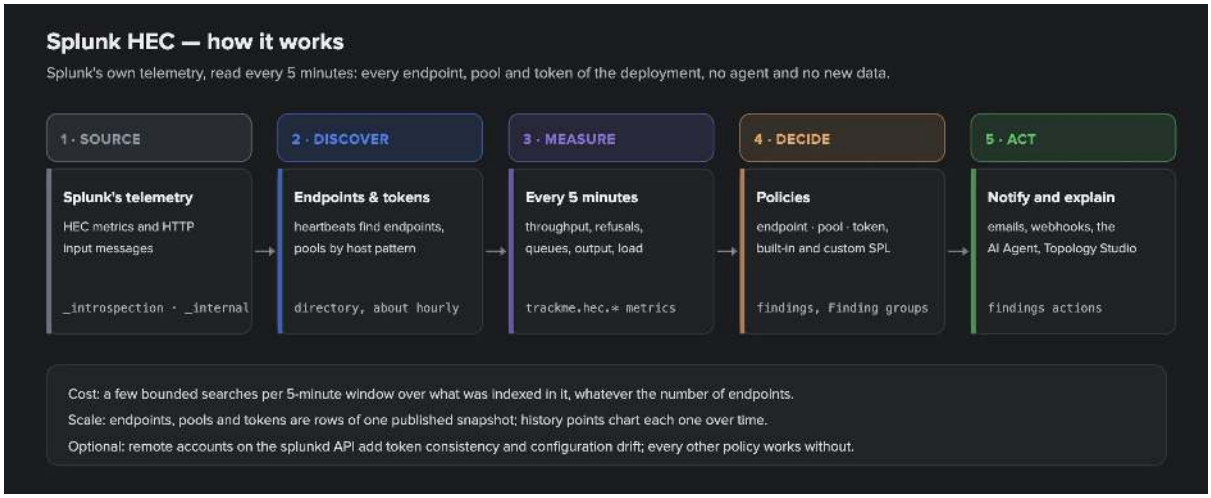
A tenant owner: `nobody`, or an account (typically a service account) holding the capabilities the creation wizard lists, so that the tenant's scheduled reports run. The scheduled triage of the Splunk HEC AI Agent needs a Splunk administrator account as owner (see *The Splunk HEC AI Agent*).

Optionally, remote accounts on the endpoints' splunkd API, for the two policies that compare the configuration of pool members (see *The splunkd API enrichment*). Every other policy works without them.

Creating the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Splunk HEC** card ("Monitor your Splunk HTTP Event Collector endpoints and pools"). Its first screen explains what the tenant does, with a **How it works** diagram on demand. The wizard has four steps.





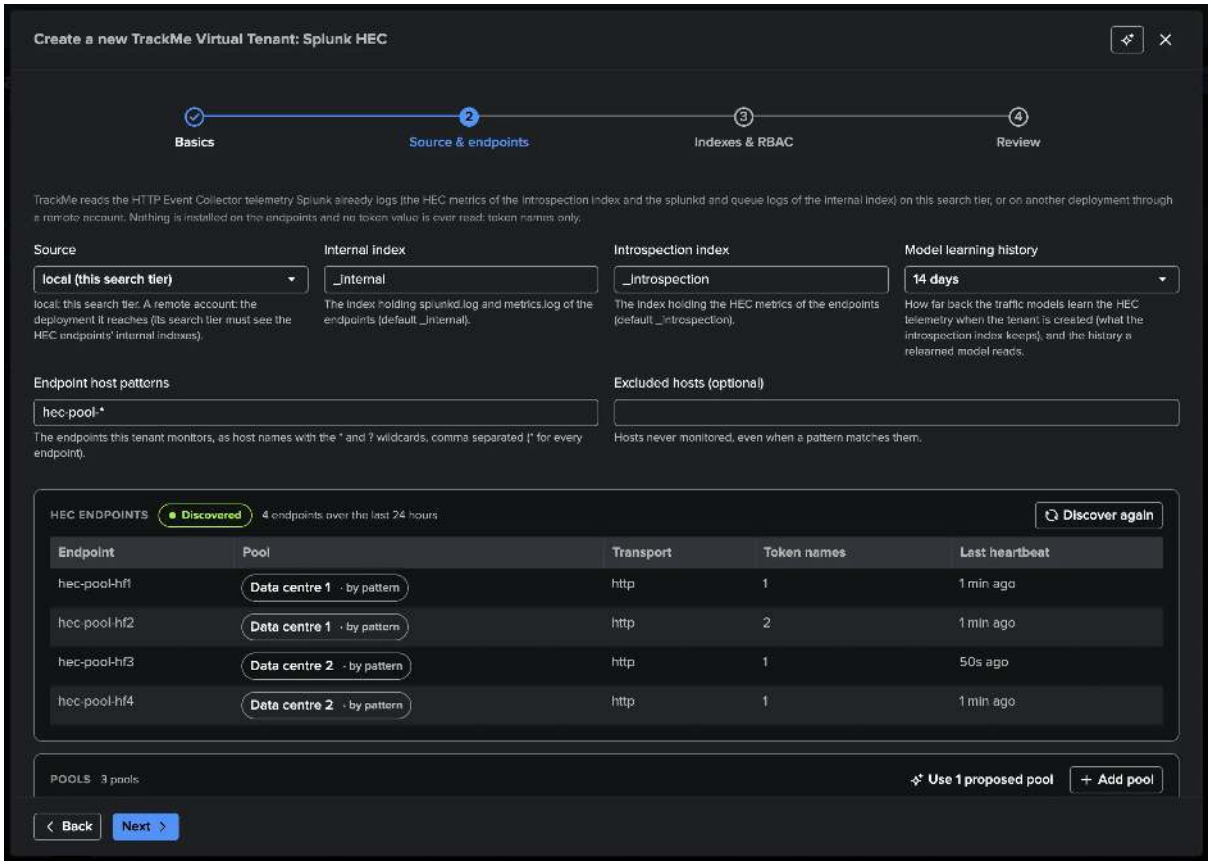
1. Basics. The **tenant identifier** (lowercase letters, digits and hyphens, up to 20 characters, for example `example-hec`; it cannot be changed afterwards), an optional **alias** and **description**.

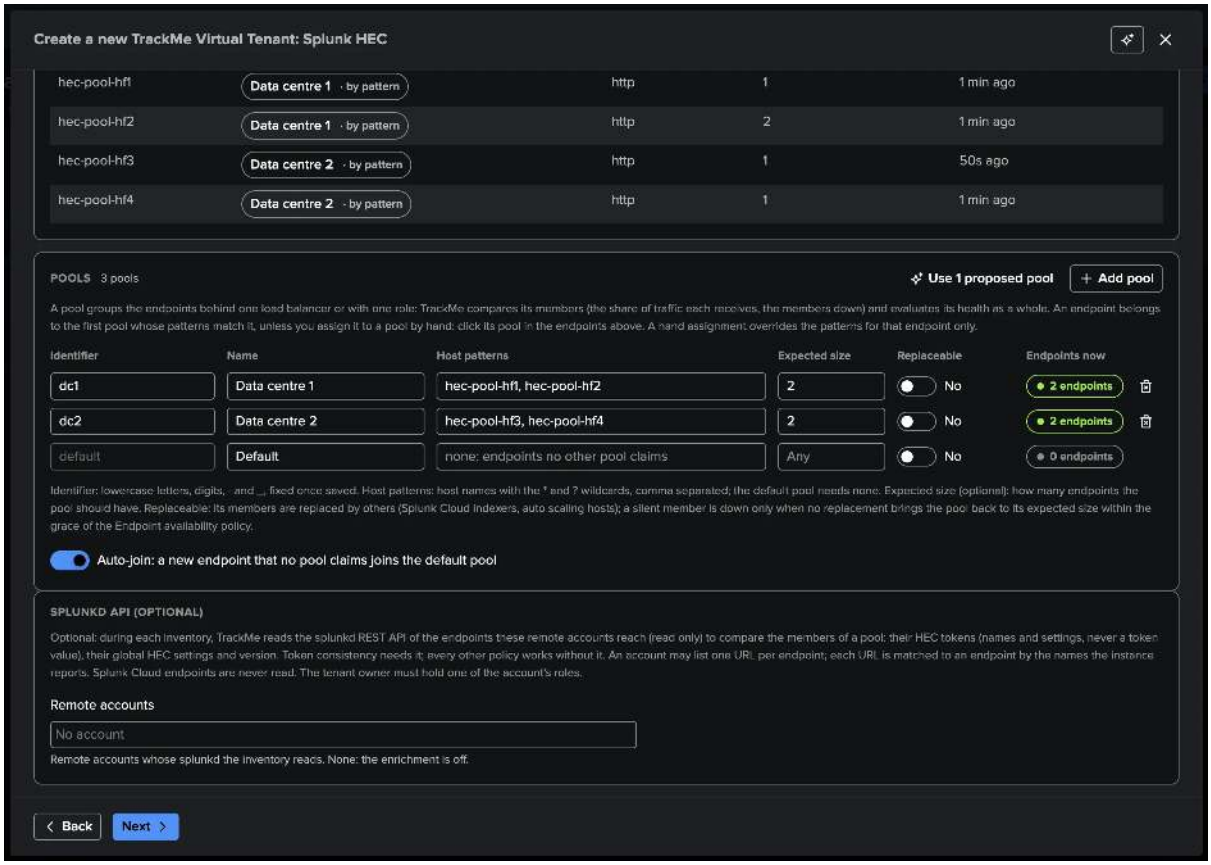
2. Source & endpoints.

- **Source**: `local` (this search tier) or a remote account. It is locked once the tenant exists: another deployment takes another tenant.
- **Internal index** (default `_internal`) and **Introspection index** (default `_introspection`): where the endpoints' `splunkd.log` / `metrics.log` and their HEC metrics are. Locked once the tenant exists.
- **Model learning history** (14 days by default; *from today on*, 7, 14, 30, 60 or 90 days): how far back the traffic

models learn the HEC telemetry when the tenant is created, and what a relearned model reads. `_introspection` must keep that much (Splunk's default retention is 14 days). Changed later in *Settings → Collection*.

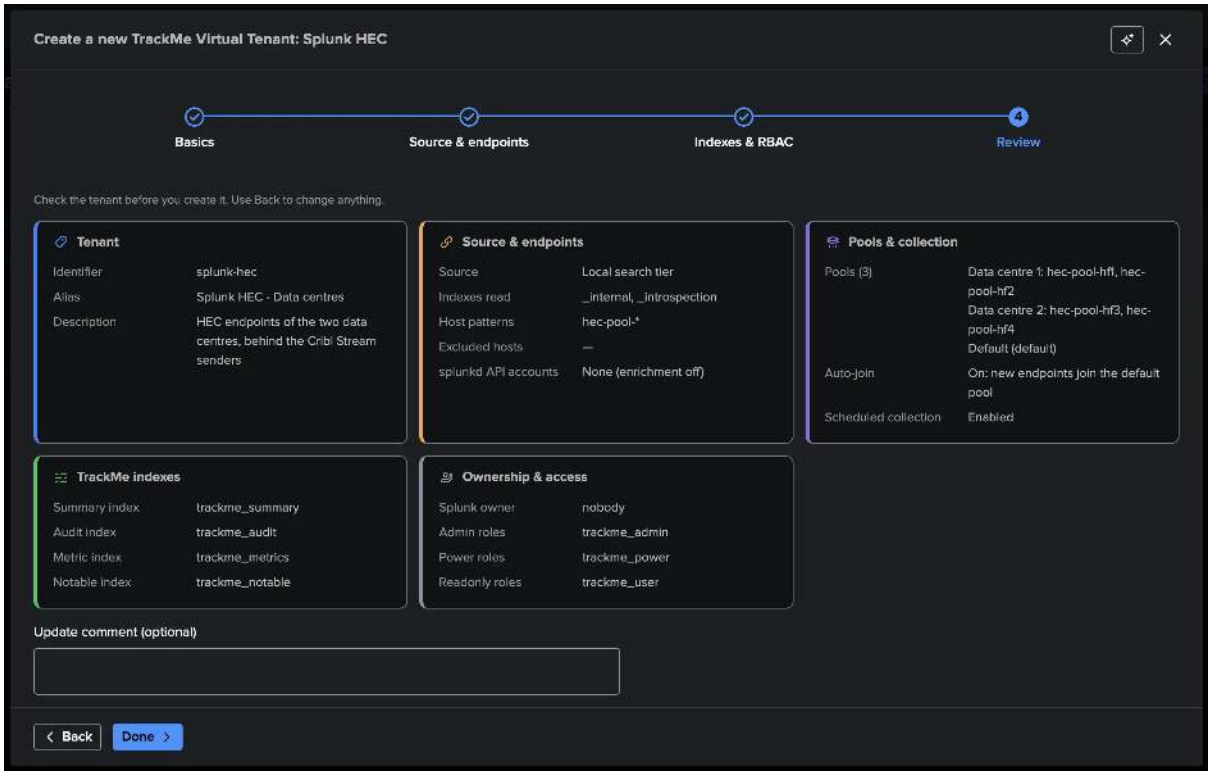
- **Endpoint host patterns**: the endpoints this tenant monitors, as host names with the `*` and `?` wildcards (`*` for every endpoint, the default), and **Excluded hosts**, never monitored even when a pattern matches them.
- **HEC endpoints**: TrackMe searches the HEC telemetry of the source and lists every endpoint that reported in the last 15 minutes (an endpoint reports a heartbeat every minute, traffic or not), with its transport, the token names active in those 15 minutes and its last heartbeat: two searches of 5 minutes at most. **Discover again** repeats the search. When nothing reported, the tenant can still be created: endpoints join it as soon as they report.
- **Pools**: **Use the proposed pools** applies the pools TrackMe proposes from the host names (see *Pools*), or **Add pool** defines your own. Each endpoint's pool is shown; click it to assign the endpoint to another pool by hand.
- **Splunk Cloud**: when the discovery finds Splunk Cloud indexers (`*.splunkcloud.com`), the wizard adds the **Splunk Cloud indexers** pool on its own, before the default pool: one per stack, matching `idx-*.<stack>.splunkcloud.com`, with **replaceable** members, because Splunk Cloud replaces its indexers under new host names. A card in the Pools section says so. The other endpoints of the deployment stay in their pools, and a pool of yours that already claims those indexers is left alone. Remove the pool or turn Replaceable off and the wizard keeps your choice; the card then warns that every replaced indexer would be reported down, with a button that adds the pool back.
- **Auto-join** (on by default): a new endpoint that no pool claims joins the default pool. Off, it waits as *discovered*: listed, but not evaluated by the pool policies.
- **splunkd API (optional)**: the remote accounts whose splunkd the directory reads (none: the enrichment is off).

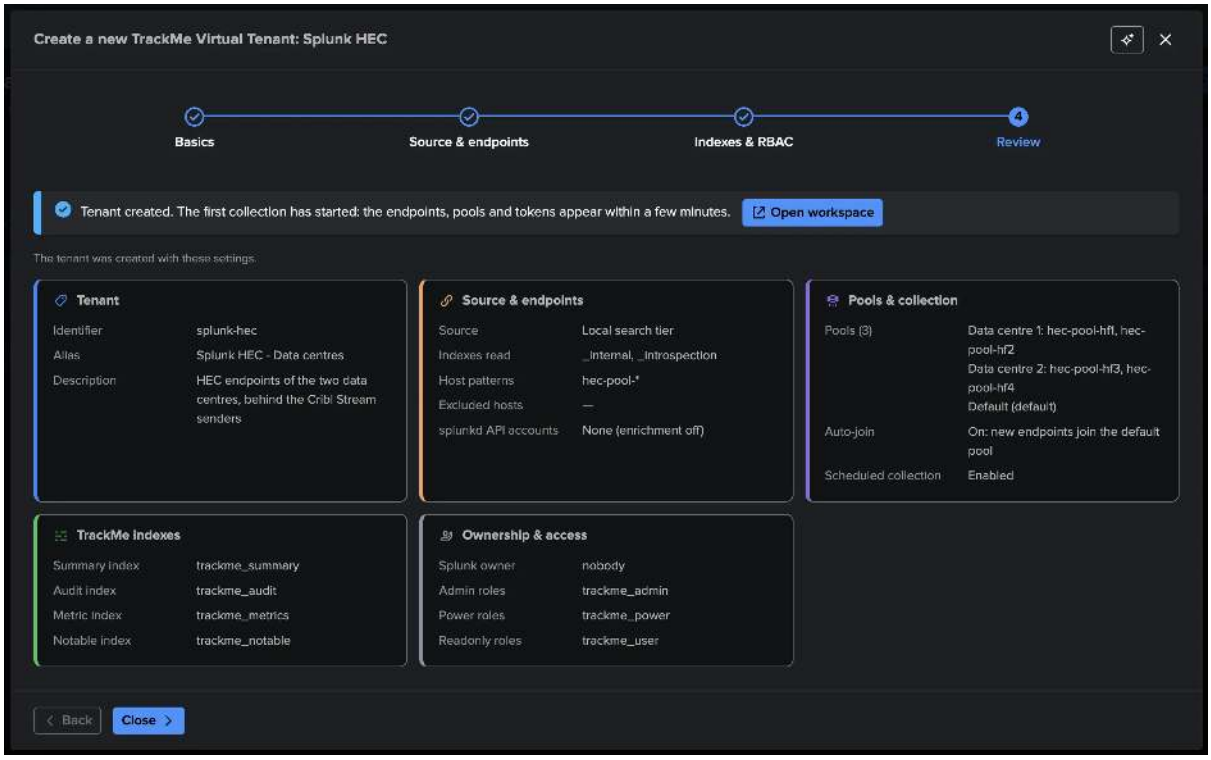




3. Indexes & RBAC. The tenant's indexes (summary, audit, metric, notable), its admin, power and user roles, and its owner, as for any tenant.

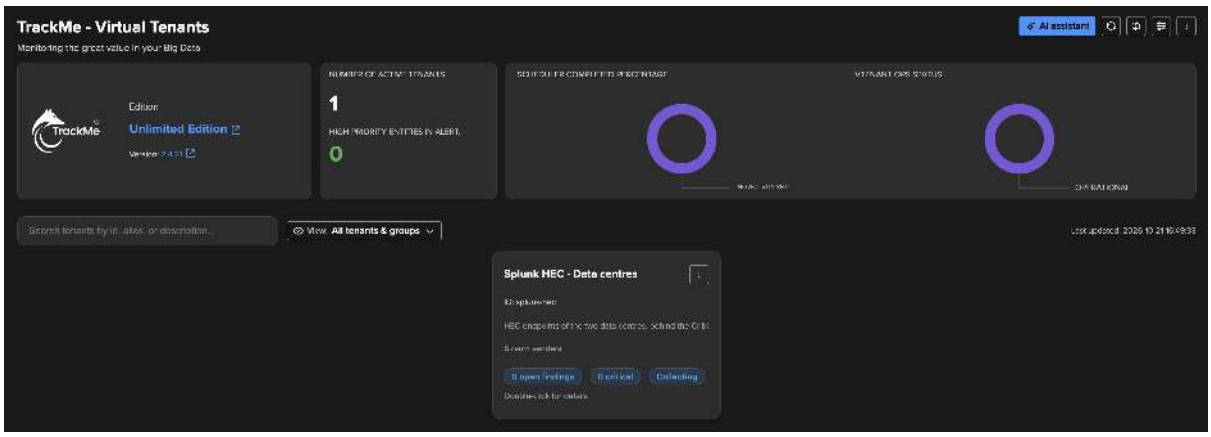
4. Review. Check the summary — source, indexes read, host patterns, pools, auto-join, splunkd API accounts, indexes and roles — and create the tenant. TrackMe creates the tenant's collections and reports and starts the first collection: the endpoints, pools and tokens appear in the workspace within a few minutes (**Open workspace**).





Collection is scheduled from the start: the HEC activity is read and the policies evaluated every 5 minutes, the endpoint directory rebuilt about once an hour, the traffic models trained twice an hour. To pause it, turn off **Scheduled collection** in **Settings → Collection**.

The workspace



Open the tenant from Virtual Tenants: Tenant Home shows the **Splunk HEC** workspace, built with the same components as the Cribl workspaces. **The status strip** shows the source, the last run of **HEC activity**, **Endpoint directory** and **Traffic models**, **Refresh** and, for the TrackMe administrators of the tenant, **Collect now**.



Tab	What it is for
Findings	The open findings and Finding groups of the tenant, their triage, assignment and findings actions.
Overview	The tiles of the deployment, the traffic over time, the pools, the endpoints needing attention, the discovered endpoints and the busiest tokens.
Endpoints	Every endpoint, by pool, filtered and paged by the server. Opens the endpoint.
Tokens	Every token name seen in the last 24 hours. Opens the token.
Traffic models	The traffic model of each pool and token, its state and expected range; opens a model and tunes one or several (see *Traffic models*).
Policies	The built-in policies and the custom policies of the tenant.
Collection health	Whether collection runs as configured, job by job, the source and the traffic models.
AI Agent	The Splunk HEC AI Agent: whether it can run, its totals, **Run a review now** and its decision log.
Settings	Source & pools, Collection, Notifications, AI Agent, Assignment, Finding actions.

The last five minutes

The states, findings and history come from the collected windows, which are settled on purpose (a window is read once its telemetry is indexed). The figures you look at are more recent: when the workspace opens, and at every refresh (automatic or manual), TrackMe also reads the **statistics of the last five minutes up to now** from Splunk and lays them over the collected rows. The badge says **Last 5 minutes · as of HH:MM:SS**; when that read fails, the workspace shows the collected window with the reason (**Collected window**). This read is an ordinary historical search over fixed bounds, run with your own rights: it never changes a state, a finding or the history. It runs at most two searches at a time, and a read still running is never started twice.

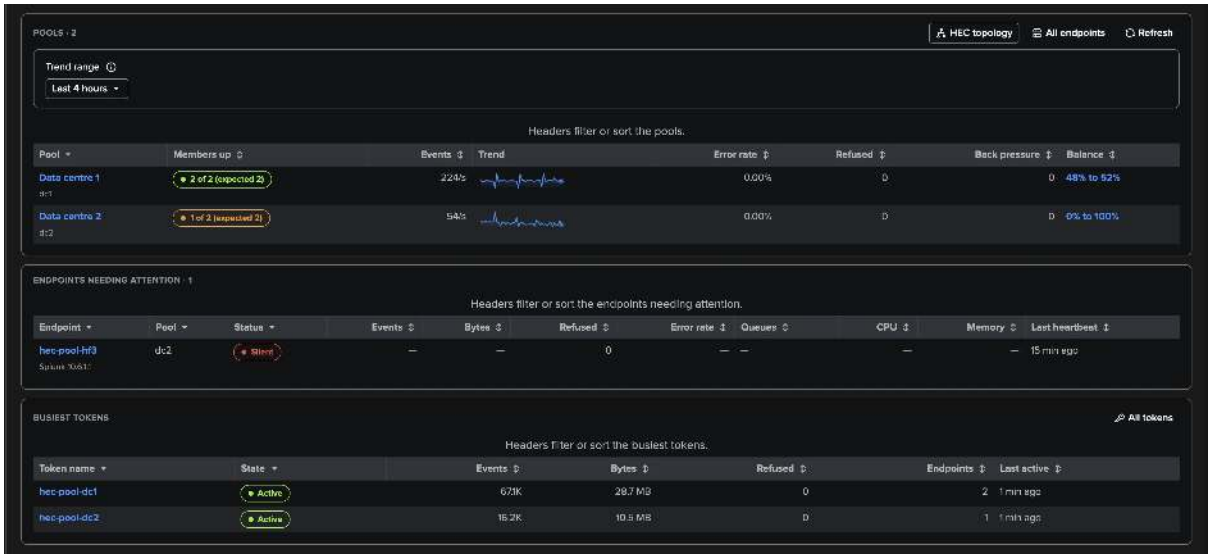
The large charts (the Overview traffic, the endpoint, pool and token drilldowns, a finding's history) also read their **latest buckets** from Splunk when the last collected window is behind, and say so under the chart.

Overview

The tiles answer the HEC questions at a glance: **Endpoints up**, **Events** and **Bytes** per second, **Active tokens**, **Refused requests** (load balancer probes excluded), **Server busy** (HTTP 503: HEC could not queue the request), **Back pressure** (endpoints whose queues backed up), and when it happens **Stalled** endpoints and **Oversized events**. Refused requests, server busy, back pressure and stalled open the Endpoints tab sorted or filtered on them.

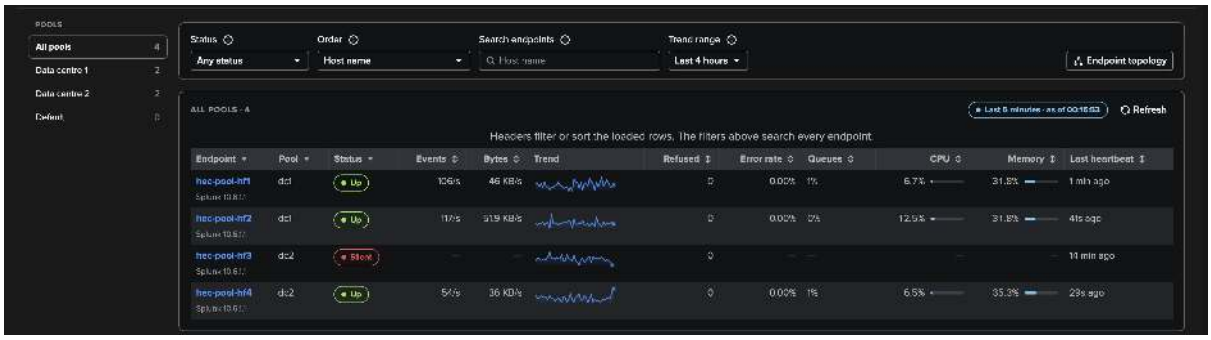


Below them: **Traffic over time** (events and bytes per second, refused requests and server busy replies per bucket, over a range you choose), the **Pools** (members up, traffic, refusals, back pressure, a trend, and the pool's traffic distribution), **Endpoints needing attention** (not up, refusing, backed up, dropping oversized events, failing destinations), the **Discovered endpoints** waiting for a pool, and the **Busiest tokens**. **HEC topology**, on the Pools panel, draws every pool and its endpoints with their live status (see *Generated topologies*).



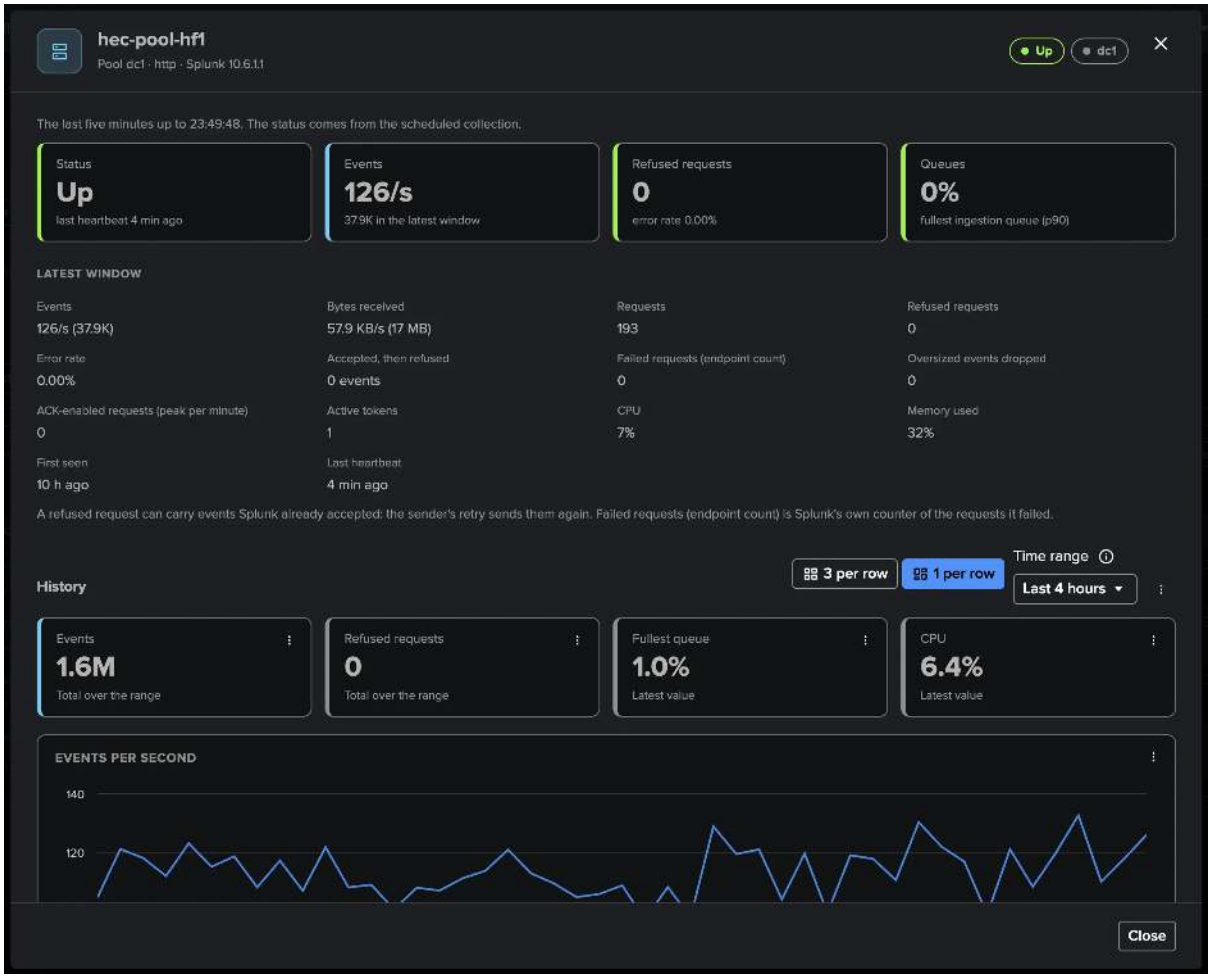
Endpoints

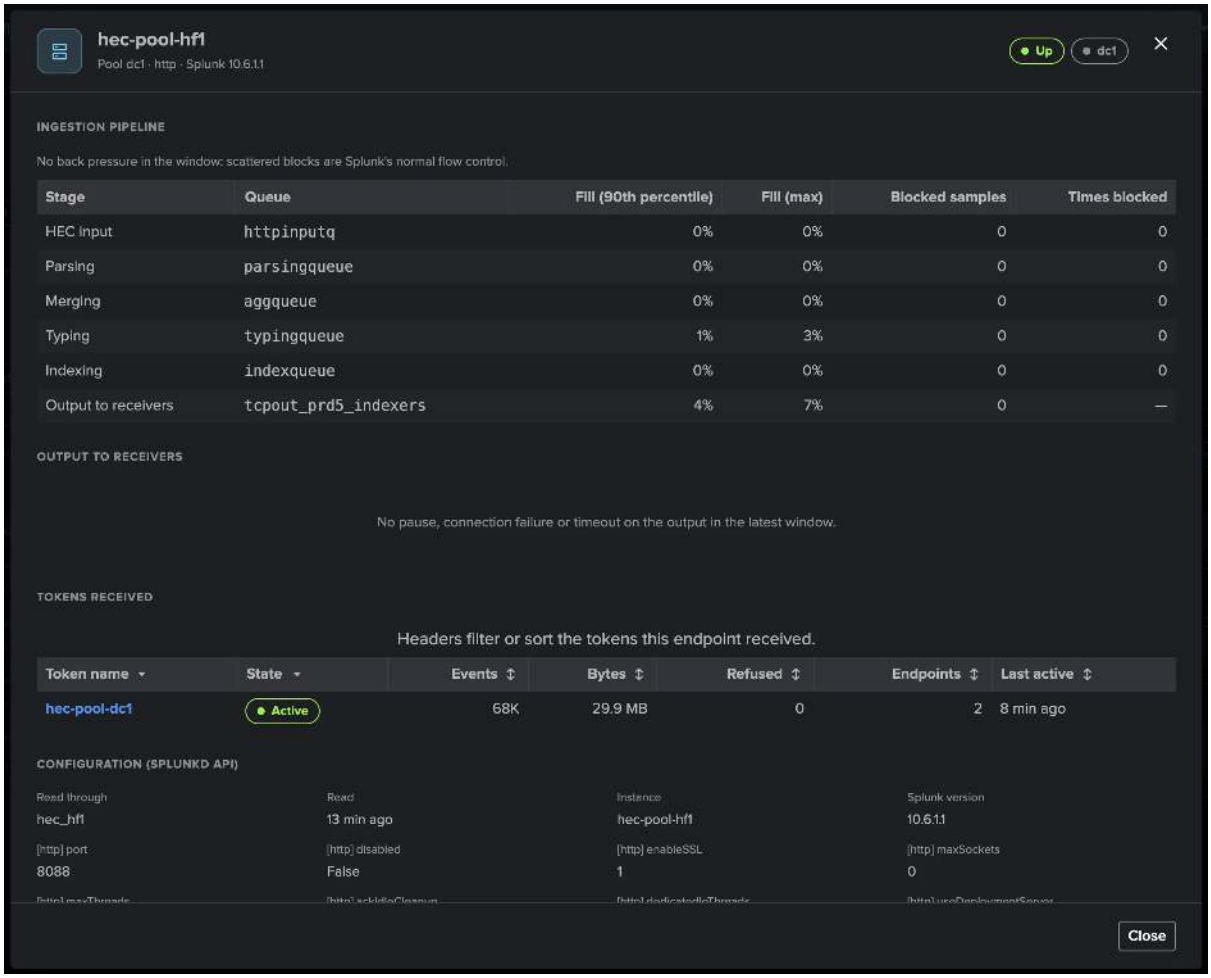
A sidebar selects **All pools**, one pool (with its members in the latest window), or the **Discovered** endpoints. The table, paged by the server, filters by **status** (up, stalled, late, silent) and host name, and sorts by host name, most events, most refused requests, most server busy replies, highest error rate, most failed requests or back pressure first. Each row shows the pool, the status, the traffic, the refused requests, the fullest ingestion queue (or the bottleneck), a trend over the **Trend range**, and the host's **CPU** (user and system, average over the window) and **Memory** used (the peak over the window), as a percentage with a bar that turns amber from 75 % and red from 90 %. Both are host-wide (the introspection `Hostwide` data: every process on the endpoint's host); a dash means the window holds no host resource telemetry for the endpoint, as for a silent one. **Endpoint topology** draws the endpoints the tab selects (see *Generated topologies*).



The endpoint drilldown shows its status and last heartbeat, events per second, refused requests and its queues, then:

- **Latest window**: events, bytes received, requests, refused requests (with the server busy share), the error rate (excluding the empty probe requests of a load balancer), the events **accepted, then refused** (a request stops at its first error: the events before it are indexed, and the sender's retry duplicates them), oversized events dropped, active tokens, CPU, memory, first seen and last heartbeat;
- **Refused requests** by HEC reply code;
- **Ingestion pipeline**: each queue of the HEC path with its stage, its fill (90th percentile and maximum), its blocked samples, the bottleneck and how busy its thread was;
- **Output to receivers**: each destination with its longest pause, connection failures and timeouts, and, when their own logs are searchable from the source, the **Receivers** and whether they stopped listening;
- **History** (events per second, refused requests, fullest queue, CPU) and, when bound, the **Configuration (splunkd API)**.


hec-pool-hf1
Pool dc1 · http · Splunk 10.6.1.1
Up
dc1
The last five minutes up to 23:49:48. The status comes from the scheduled collection.
Status
Up
last heartbeat 4 min ago
Events
126/s
37.9K in the latest window
Refused requests
0
error rate 0.00%
Queues
0%
fullest ingestion queue (p90)
LATEST WINDOW
Events
126/s (37.9K)
Bytes received
57.9 KB/s (17 MB)
Requests
193
Refused requests
0
Error rate
0.00%
Accepted, then refused
0 events
Failed requests (endpoint count)
0
Oversized events dropped
0
ACK-enabled requests (peak per minute)
0
Active tokens
1
CPU
7%
Memory used
32%
First seen
10 h ago
Last heartbeat
4 min ago
A refused request can carry events Splunk already accepted; the sender's retry sends them again. Failed requests (endpoint count) is Splunk's own counter of the requests it failed.
History
3 per row
1 per row
Time range
Last 4 hours
Events
1.6M
Total over the range
Refused requests
0
Total over the range
Fullest queue
1.0%
Latest value
CPU
6.4%
Latest value
EVENTS PER SECOND
140
120
Close



Why it was flagged. A refused-requests or back-pressure figure in the tables, the tiles of the drilldown or a pool row opens a screen that shows *why*, from Splunk's own logs, over the last hour, 4 hours or 24 hours: the summary (the refused total with its top reply codes and clients, or the saturated queues and the stage after the furthest-downstream one, with the output pauses that caused it), the timeline, the breakdown and the raw lines (at most 50, token-shaped values masked), each with **Open in Search**.

Pools

A **pool** groups the endpoints behind one load balancer or with one role. TrackMe compares its members — the members up, the share of the traffic each one receives, their versions and settings — and judges its capacity and its traffic as a whole. A tenant has up to 200 pools, and always a **default** pool.

Membership. An endpoint belongs to the pool it is **assigned** to by hand, else the first pool whose **host patterns** match it (case-insensitive `*` and `?` wildcards, for example `hf-*.example.com`), else the default pool's own patterns, else the default pool when **auto-join** is on. Otherwise it is listed as **discovered**: shown, evaluated by the endpoint policies, but by no pool policy. An endpoint excluded or outside the tenant's host patterns is in no pool, whatever its assignment.

- **Assignments** (up to 500) pin one endpoint, by its exact host name, to a pool; **By pattern** removes the assignment. An assigned endpoint the discovery did not report stays listed as *Not seen in 24 hours*.
- **Proposed pools**: the discovery proposes one pool per host-name family (the host names with their trailing digits or cloud instance id removed; a family of one host joins the default pool) and one **Splunk Cloud indexers** pool per Splunk Cloud stack.
- **Expected size** (optional, 1 to 10,000): how many endpoints the pool should have. Without it, Pool capacity learns the size from the pool's own history.
- **Replaceable**: the pool's members are replaced by others with new host names (Splunk Cloud indexers, auto scaling groups, containers). A silent member is then judged through its pool: while the pool is back at its

expected size, it was **replaced** and opens nothing; when no replacement arrives within the **replacement grace** of Endpoint availability (30 minutes by default), it is down as in any pool. A replaced member's open findings close as *endpoint replaced* (never as recovered), it leaves the directory at its next rebuild, and the pool drilldown lists the last replaced members. The discovery turns it on for the Splunk Cloud pools it proposes, and the creation wizard adds those pools on its own (see *Creating the tenant*). A replaceable pool takes no hand assignment.

Pools are configuration: a pool change applies at the next HEC activity run. Moving an endpoint moves its findings: those under the old pool close and open under the new one if the condition holds; its history is kept by host.

The pool drilldown (from a pool's name or **Balance** in the Overview, the Endpoints tab or a pool finding) shows the latest window, then the **Traffic distribution** over a range: **Balance** (100 for an even split over the members up, 0 when one member takes everything), the members carrying traffic, the smallest share against the fair share, the members up that received nothing, the share of each member over time, the balance over time and the members table, then the pool history and its replaced members.

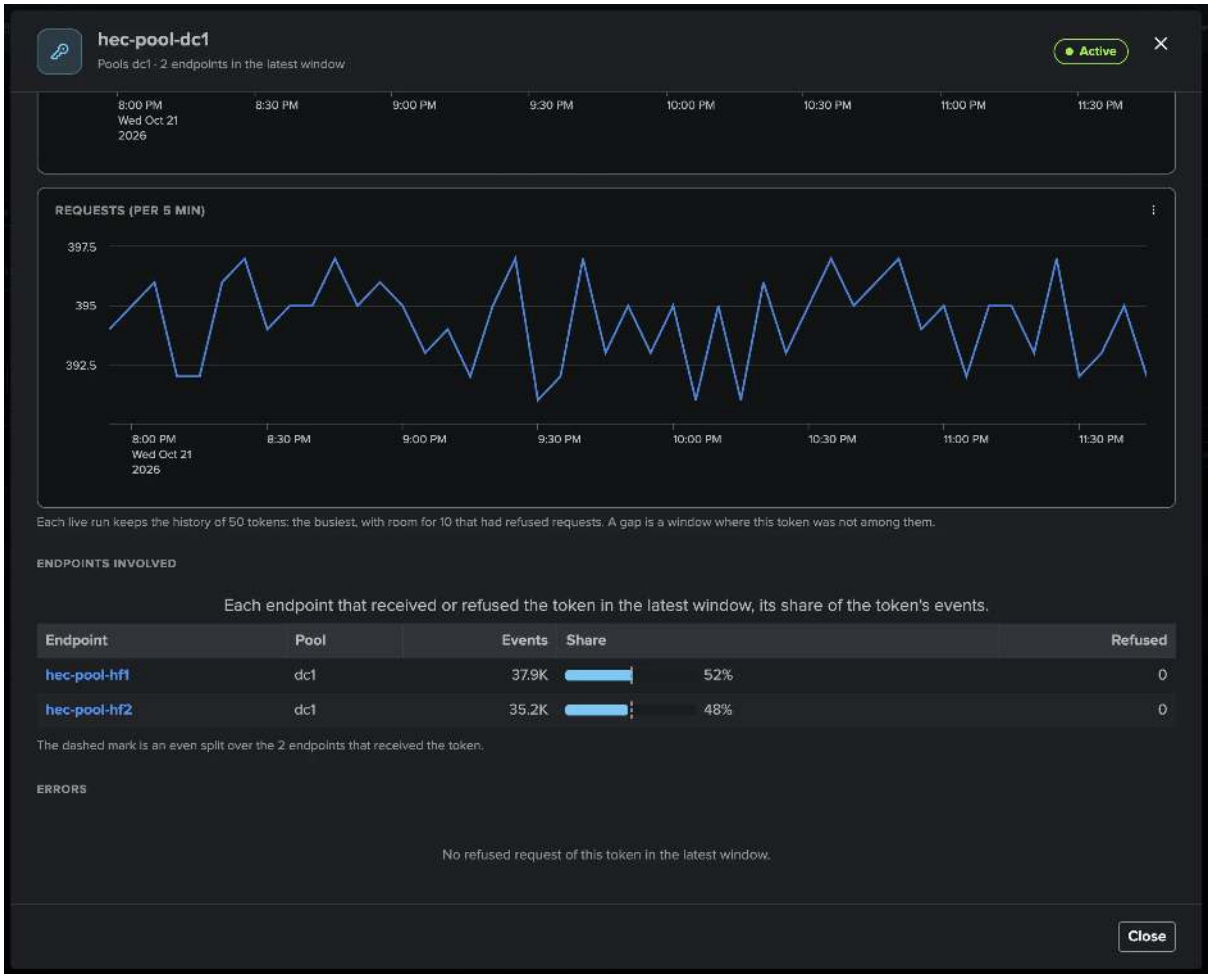


Tokens

The **Tokens** tab lists every token name seen in the last 24 hours: the pools and endpoints that received it, its last activity, its events, bytes and refused requests. A token name in this tab, the Overview's busiest tokens, an endpoint or a token finding opens the **token drilldown**: its state in the latest window (active or idle), events per second, refused requests and error rate, its last 24 hours, an alert naming the index when Splunk refused its events for an **incorrect index**, its **Traffic over time**, the **Endpoints involved** (each endpoint that received or refused it, its share of the token's events against an even split), and the **Errors** (classes, HTTP statuses, reply codes in plain words, indexes refused, client addresses). **Token topology** draws the tokens the tab lists under the pools that received them (see *Generated topologies*).


Search token names
Token topology
TOKEN NAMES · 2
Refresh
Headers filter or sort the loaded rows. The search above applies to every token name.
Token name
State
Events
Bytes
Refused
Endpoints
Last active
hec-pool-dc1
Active
hec-pool-dc2
Active

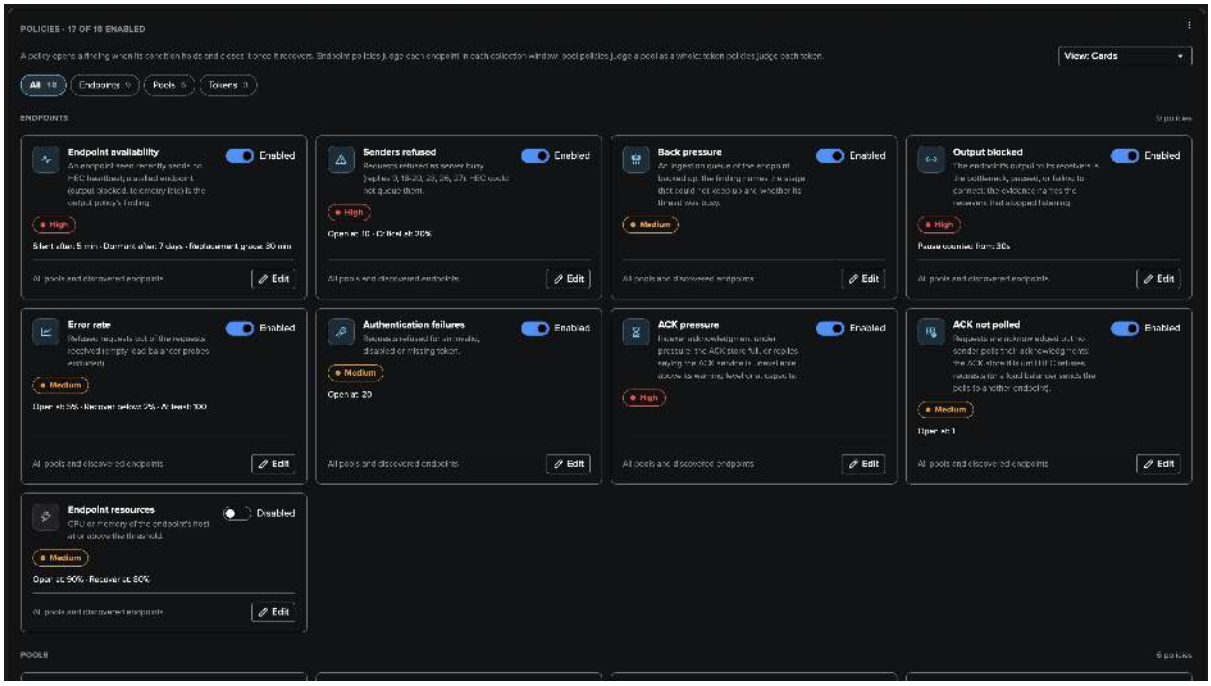

hec-pool-dc1
Pools dc1 · 2 endpoints in the latest window
Active
State
Active
last active 44s ago
Events
244/s
69.1K in the latest window
Refused requests
0
error rate 0.00%
Last 24 hours
6.9M
events · 2.95 GB · 65 failures
The last five minutes up to 23:54:18. The status comes from the scheduled collection.
LAST 5 MINUTES
Events
244/s (69.1K)
Bytes received
30.6 MB
Requests
390
Refused requests
0
Error rate
0.00%
Accepted, then refused
0 events
Oversized events dropped
0
ACK-enabled requests (peak per minute)
0
Last active
44s ago
A refused request can carry events Splunk already accepted: the sender's retry sends them again.
Traffic over time
3 per row
1 per row
Time range
Last 4 hours
Events
3.2M
Total over the range
Bytes received
1.35 GB
Total over the range
Refused requests
0
Total over the range
Requests
18.5K
Total over the range
EVENTS PER SECOND
300
250
200
Close



The traffic history of a token is written for the 50 busiest tokens of each window, with 10 places kept for tokens with refused requests: a window where a token was outside them is a gap in its chart, not a zero.

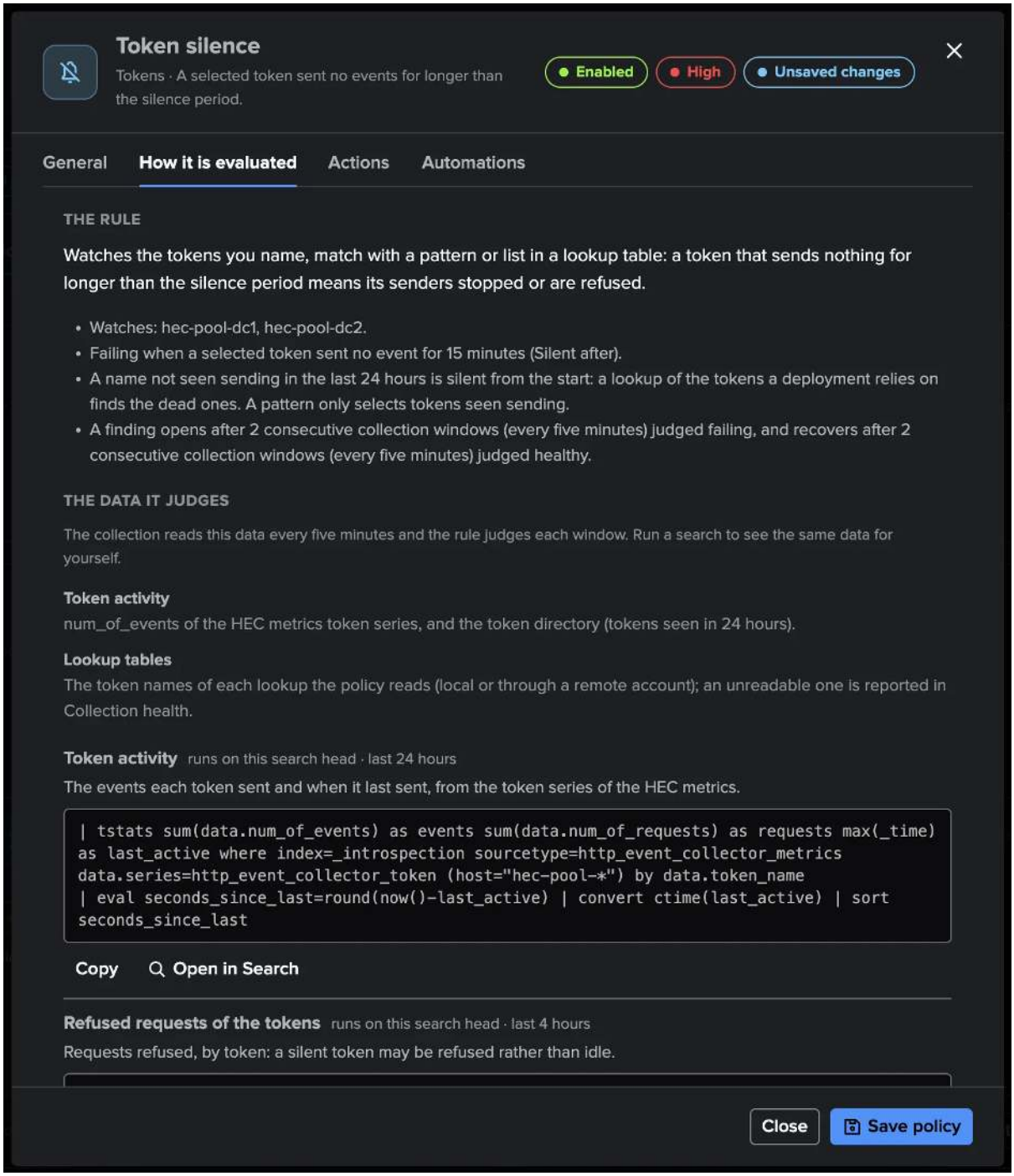
Policies

The catalogue of the **eighteen** built-in policies, in three categories: **Endpoints**, **Pools** and **Tokens**. Seventeen are on out of the box; **Endpoint resources** is off (it watches the host, not HEC). **Token silence** is on but decides nothing until you select tokens, and **Token consistency** needs the splunkd API.



Show them as cards, compact cards or a list. A TrackMe administrator of the tenant turns a policy on or off and opens its editor:

- **General**: on / off, severity, the **checks to open** (consecutive failing 5-minute windows) and **to recover**, the policy's own conditions with their help, **During bank holidays** (see *Bank holidays*) and the **Pools** it applies to (empty: every pool; the endpoint policies also cover the discovered endpoints). A token policy has no pool scope: a token goes to every pool that accepts it.
- **How it is evaluated**: what the policy does and how it decides **with its current settings**, the data it reads, **ready-to-run searches** built from the tenant's own source (its indexes, host patterns and remote account), each with **Open in Search** and **Copy**, how to investigate, the causes and resolutions, the related policies and notes. The same help opens from a finding's investigation, with the searches narrowed to its endpoint or token.
- **Actions** and **Automations**: the findings actions offered on this policy's findings and the rules that call them.



Every policy, its settings and defaults are in *Built-in policies*.

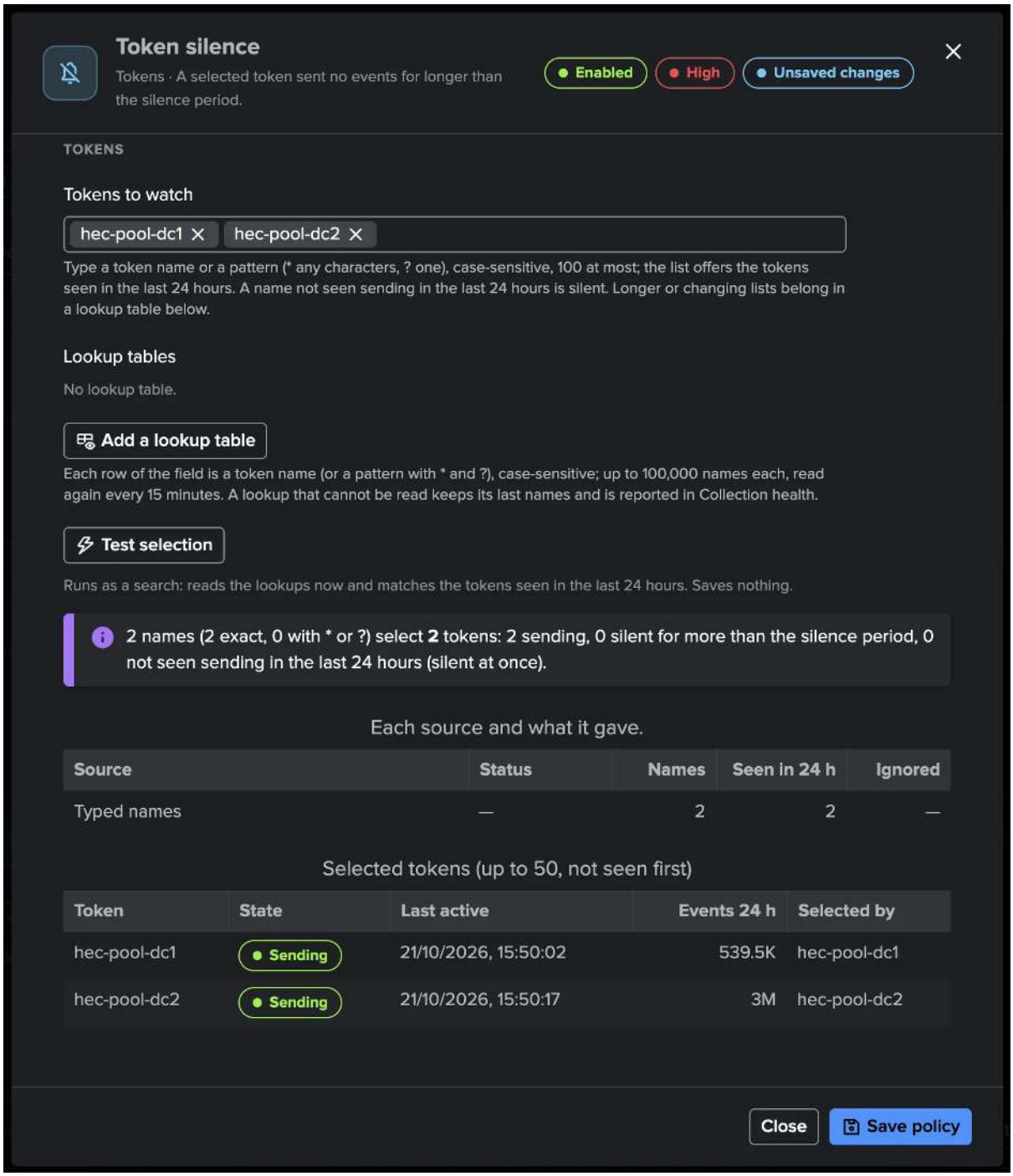
Tokens that must keep sending

Token silence opens a finding when a token you select sends no event for longer than the silence period (15 minutes by default). Its editor, **Tokens to watch**, selects them from two sources that add up:

- **Token names** typed in the policy (the list offers the tokens seen in the last 24 hours), up to 100 names or patterns, case-sensitive, with `*` (any characters) and `?` (one character), for example `app.prod` or `payments-*`;
- **Lookup tables** from Splunk, for long or changing lists such as the tokens a deployment relies on: up to 20 lookups, on the local search tier or on a remote deployment through a TrackMe remote account, each with the field holding the token names and a **match mode**, **Exact** (the default) or **Wildcard**. A lookup holds up to 100,000 distinct names and is read again every 15 minutes; one that cannot be read keeps its last good names, and Collection health says so (`token_lookups_failing`).

A name selected exactly that was **not seen sending in the last 24 hours is silent at once**: this is how a lookup of the tokens you rely on finds the dead ones. A pattern selects only the tokens seen sending.

Test selection checks the selection before you save: it runs as a search, reads every lookup now and matches the latest token directory and window. It shows the tokens selected by state (sending, silent, not seen in 24 hours), up to 50 examples with the entry that selected each, what each source gave, and the patterns that select nothing. It saves nothing.



Custom policies

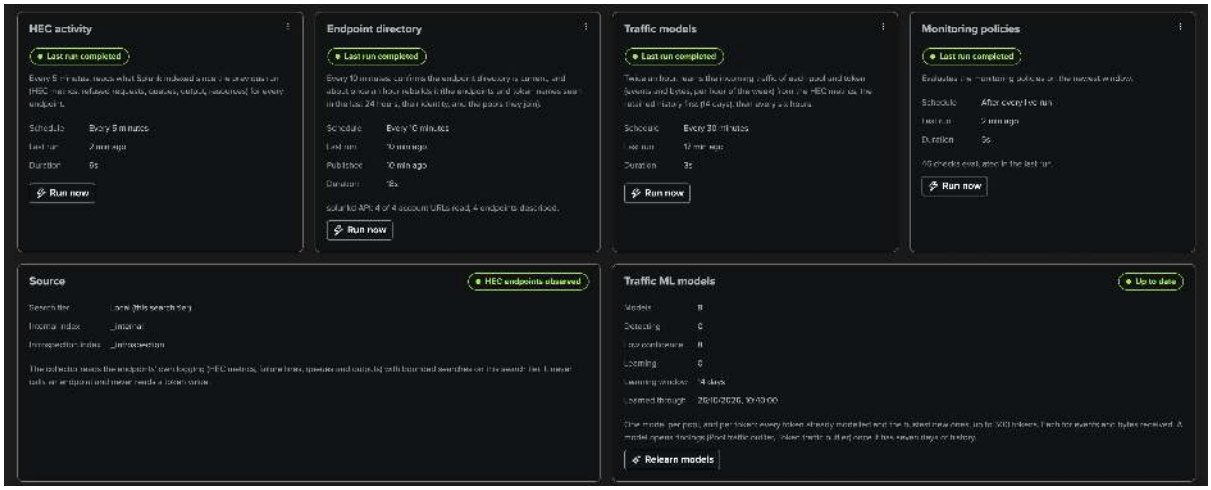
When no built-in policy covers a rule of your own, write it as an SPL search on the tenant's Splunk data: **Policies → Custom policies → Create policy**. On a Splunk HEC tenant each result row names an **endpoint** (a host name within the tenant's host patterns), a **pool** (a pool id of the tenant) or a **token** (a token name). A custom finding joins the Finding group of its object and is triaged, notified, offered findings actions and reviewed by the AI Agent like a

built-in one. A host outside the tenant's host patterns, or a pool it does not define, is not monitored: the row is skipped and its earlier findings recover.

The editor, the preview, the version history and the lifecycle are the same as on a Cribl Stream tenant (see *Custom policies*). Only the tenant's TrackMe administrators see and edit custom policies, because they hold SPL; their execution health is in *Collection health → Custom policies*.

Collection health

One card per job — **HEC activity**, **Endpoint directory**, **Traffic models** and **Monitoring policies** — with its schedule, its last run and its outcome, the next run and **Run now**. A run that Splunk completed short, or whose searches carried Splunk messages, says so: those messages are recorded, never a failure. The **Endpoint directory** card also says what the splunkd API enrichment read. The **Source** panel shows the search tier and indexes the tenant reads and whether HEC endpoints were observed; the **Traffic ML models** panel shows the models, their confidence and the learning history.



When collection does not run as configured, the tab lists the problems in plain words, and the Configuration Guardian raises **Splunk HEC collection needs attention** (`cribl_collection_health`). Every condition is in *Splunk HEC reference*.

Settings

One Save covers every section:

- **Source & pools**: the source, the two indexes (locked once configured), the host patterns and excluded hosts, the pools with their patterns, assignments, expected size and replaceable members, auto-join, and the **splunkd API** accounts. **Changing the host patterns or excluded hosts starts a new source revision**. **Repair workspace** checks the tenant's collections and reports and keeps findings and settings.
- **Collection**: **Scheduled collection** on or off, and the **Model learning history**.
- **Traffic ML models**: the defaults every traffic model of the tenant follows (see *Traffic models*).
- **Notifications**: finding emails (events, minimum severity, policies, delivery account, recipients, digest or one email per Finding group, charts, test email).
- **AI Agent**: the Splunk HEC AI Agent (see *The Splunk HEC AI Agent*).
- **Assignment**: assignment groups, their team email, ordered routing rules and manual overrides.
- **Finding actions**: drilldown searches, links and webhooks offered on findings, and the **Automations** that call an action automatically.

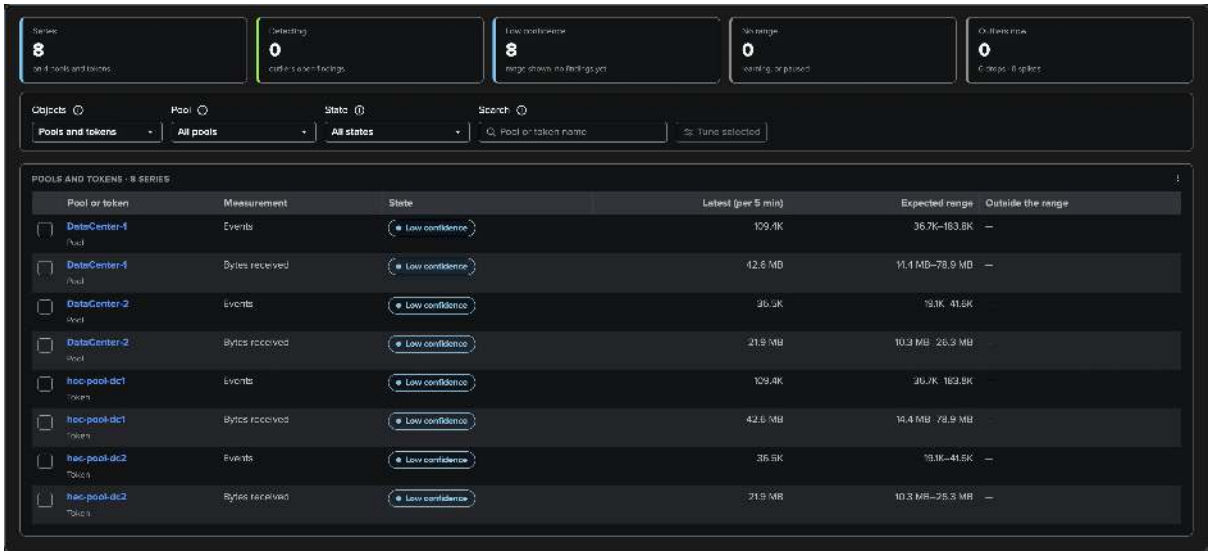
A TrackMe administrator of the tenant manages these settings; other roles read them.

Traffic models

The Splunk HEC tenant learns what each **pool** and each **token** usually receives — its **events** and its **bytes received** — per hour of the week, with the model arithmetic of the Cribl Stream tenant's traffic models (*Traffic ML models and Expected traffic*): recent weeks count more, the hour of the day first, then the weekday and hour once four weeks are learned, and an expected range of the usual value plus or minus three standard deviations, never tighter than 10 % of it. Endpoints get no model: load balancing moves traffic between the members of a pool, and Pool imbalance already watches their shares.

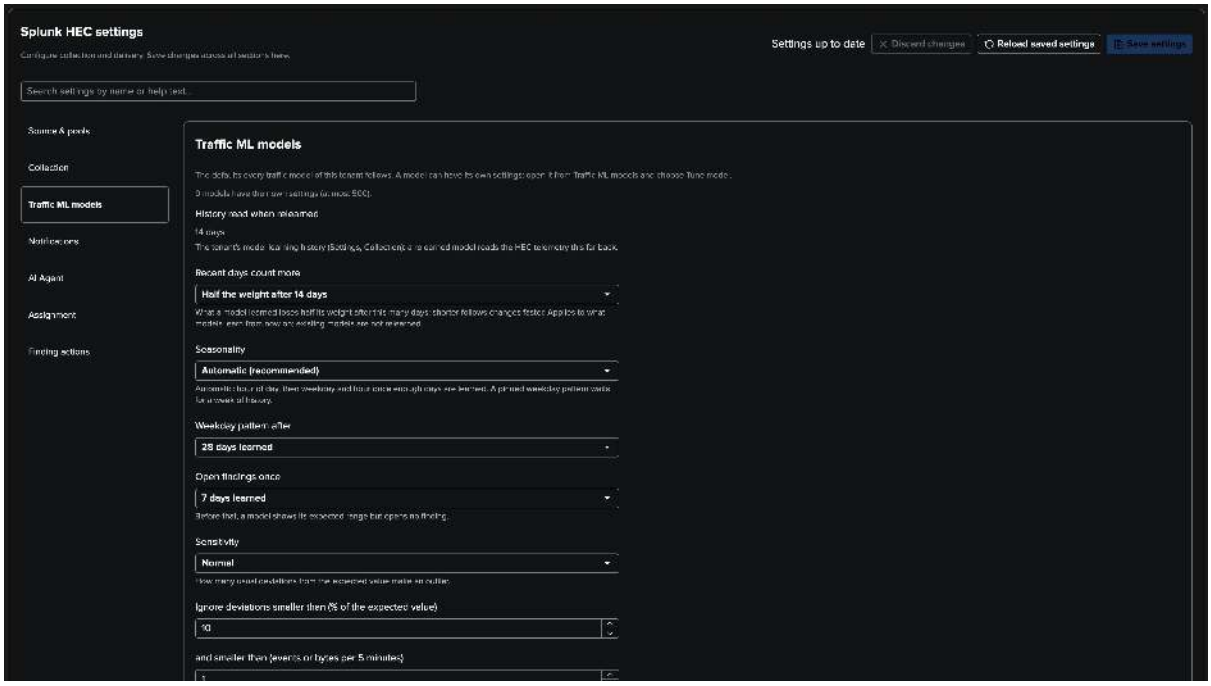
- **Learned from the HEC telemetry itself**, not from the history TrackMe writes: on its first runs, a new tenant learns the history the source keeps in `_introspection`, as far back as its **Model learning history** (14 days by default, chosen in the wizard; see *Creating the tenant*), so its models can reach normal confidence within the first runs. Then the models learn new traffic every six hours.
- **One model per pool, and per token for the 500 busiest tokens**, admitted busiest first. A modelled token that sends nothing is learned at zero; a series without traffic for 14 days is retired.
- **Confidence**: *learning* (less than a day: no expected range), *low confidence* (1 to 7 days: the range is drawn, no finding), *detecting* (7 days and more, normal confidence).
- **Bank holidays**: the training leaves out the hours of the tenant's bank holidays when the *Exclude from ML training* effect applies to the tenant (on by default; see *Bank holidays*), as for Cribl Stream.

The Traffic models tab counts the series, how many detect, how many have a range but low confidence, how many have no range yet (learning or paused) and the outliers now (drops and spikes). Its table lists every model — pool or token, measurement, state, the latest value per five minutes, the expected range and how far outside it the value is — filtered by object, pool, state or name. A name opens the **model view**: its state, the days learned, its seasonality, the expected range now, and the measured traffic against the expected band.



The expected range on the charts. The pool and token histories, and the trend of a Pool or Token traffic outlier finding, draw the measured traffic (events, bytes received) against the range its model expects, with the spikes and drops marked, a **Traffic ML model** pill with the model's state and the number of buckets outside the range. The **Expected range** toggle shows or hides the band (remembered per browser, as on Cribl Stream), and the **Traffic ML model** button opens the model.

Tuning. A TrackMe administrator of the tenant sets the defaults every model follows in *Settings → Traffic ML models* (how much more recent days count, seasonality, when the weekday pattern starts, when findings open, sensitivity, the smallest deviation that counts), and gives one model, or several of one measurement (**Tune selected** in the Traffic models tab), **its own settings** with **Tune model**: the same settings plus the training days, the direction (drops, spikes or both), a pause and **excluded periods** the model must not learn from. The tuning screen simulates the current and the proposed range over the last 24 hours, 7 or 30 days, with the drops and spikes each would have raised, before anything is saved. A paused model keeps learning but gives no range; pausing the measurement a finding is judged on closes that finding.



Not an anomaly. When a Pool or Token traffic outlier finding was expected — a maintenance, a load test, a migration of senders — a power or admin role of the tenant marks it **Not an anomaly**, from the finding or from a chart interval of the model view: the period is excluded from what the model learns, the model relearns without it at the next training run, and the finding is dismissed with your reason as its note.

Relearn. *Collection → Relearn models* forgets every model and learns the history again on the next training run; **Open models in search** lists them. A model relearned on 14 days of history goes back to hour-of-day seasonality until it learns four weeks again.

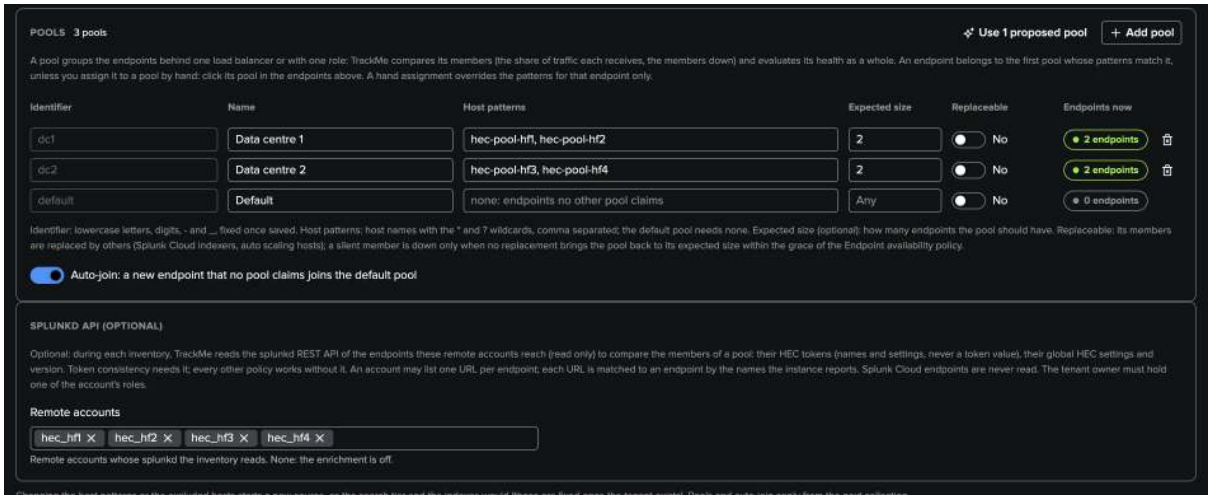
Two policies use the models: **Pool traffic outlier** and **Token traffic outlier** open a finding when the window's events (or bytes) fall below or rise above the model's range for that hour, once the model reaches normal confidence (or low confidence, if you choose), and only when the window or the model reaches the **At least** amount per five minutes (100 by default). A modelled token absent from the window is judged at zero: the drop that matters most.

Pool traffic drop is the simpler check that needs no model: the pool's rate against the median of the same hour of the day over the previous 7 days.

The splunkd API enrichment

The telemetry tells what HEC does, not how it is configured. Two policies compare the configuration of the members of a pool, and need the **splunkd REST API** of the endpoints: **Token consistency** (a token missing on some members, or with a different status, index, allowed indexes or ACK) and the HEC settings part of **Configuration drift** (the Splunk versions are compared without the API). The enrichment is **optional** and **read only**.

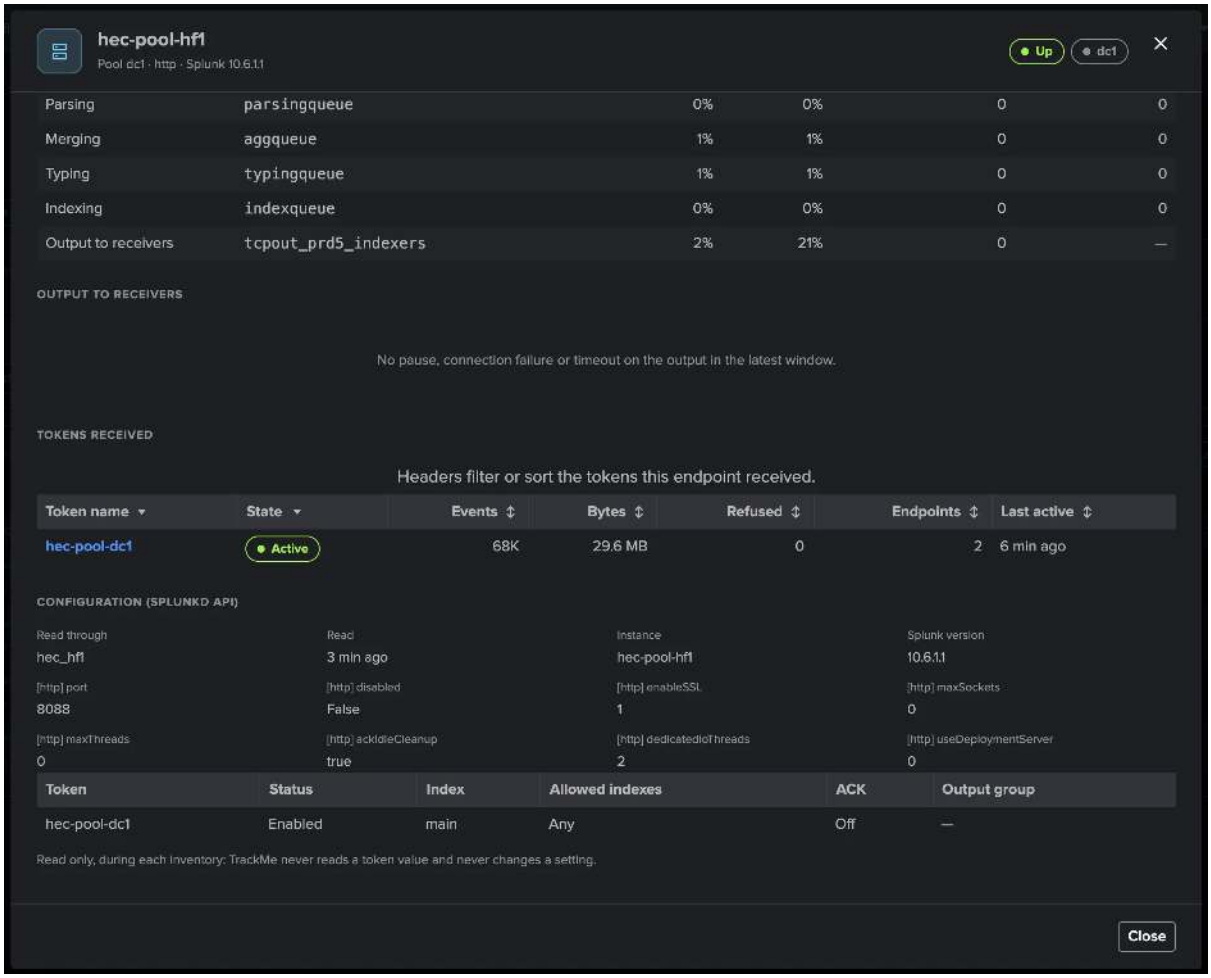
Setting it up. In *Settings → Source & pools → splunkd API (optional)*, choose up to 20 TrackMe remote accounts (*Remote Splunk deployments*) whose URLs reach the endpoints' management port. An account may list one URL per endpoint (up to 50 HTTPS URLs). The tenant owner must hold one of each account's roles.



What is read. At each rebuild of the endpoint directory, for every URL: `/services/server/info` (the instance's names and version), `/services/data/inputs/http/http` (the global `[http]` settings) and `/services/data/inputs/http` (the HEC tokens by name, up to 200 per endpoint: disabled, index, allowed indexes, ACK, output group). **The token value is dropped while the answer is parsed**: it is never stored, logged or returned. Each URL is matched to an endpoint of the directory by the names its instance reports; two URLs matching one endpoint bind neither.

Splunk Cloud is never read: an account URL on `*.splunkcloud.com` is skipped and a Splunk Cloud endpoint is never described or compared (Splunk Cloud indexers have no customer-reachable management port). The rest of the tenant works on Splunk Cloud from `_introspection` and `_internal`.

Failures never fail the directory. A URL that cannot be read is counted, and its endpoint keeps its earlier facts while they are fresh (3 hours). The **Endpoint directory** card in Collection health says what was read and why the rest was not, per account: for example *splunkd API: 3 of 4 account URLs read, 3 endpoints described, 1 could not be read (api_hf: connection failed)*. The reasons are *connection failed*, *timed out*, *access denied (check the account token)*, *HTTP error*, *answer too large*, *unexpected answer*, *out of time this run* and *read failed*; the card also counts URLs that matched no endpoint, Splunk Cloud URLs skipped, accounts the tenant owner may not use and accounts unavailable. The endpoint drilldown shows the facts in **Configuration (splunkd API)**.



Note

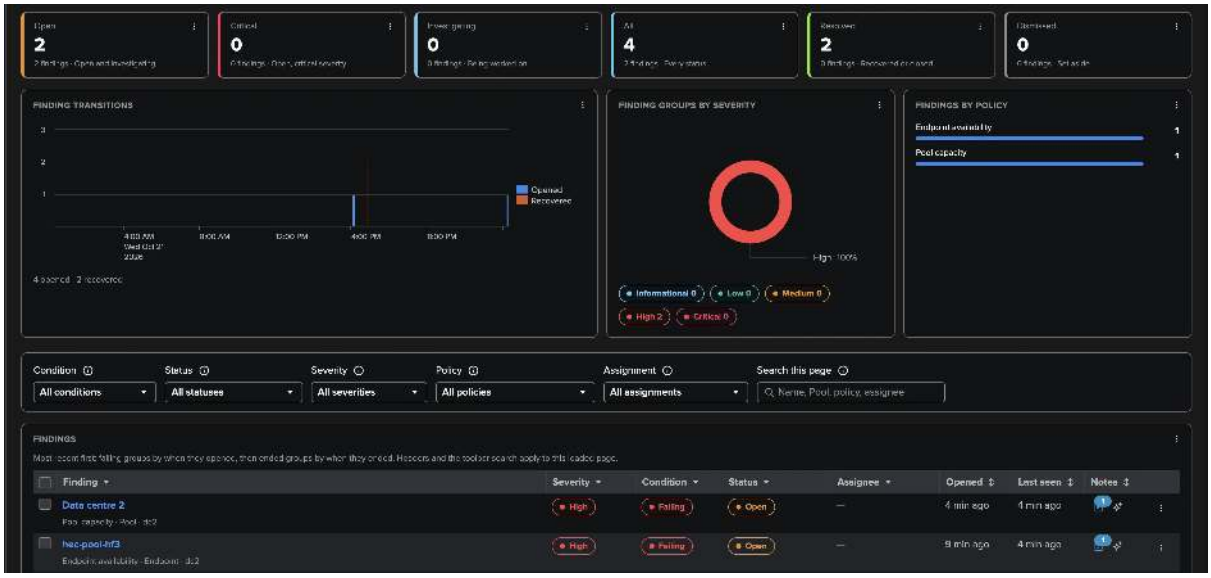
The reads follow the trust model of every TrackMe remote-account read: the remote account has no certificate verification setting. Bind accounts whose management port is reached over a trusted network.

Findings in practice

A Splunk HEC finding is about one object and one policy:

The finding is about	Raised by
An endpoint	Endpoint availability, Senders refused, Back pressure, Output blocked, Error rate, Authentication failures, ACK pressure, ACK not polled, Endpoint resources.
A pool	Pool capacity, Pool imbalance, Pool traffic drop, Pool traffic outlier, Token consistency, Configuration drift.
A token	Rejected data, Token silence, Token traffic outlier.

Findings work as on Cribl Stream and Cribl Edge. Each object gathers its findings in one automatic **Finding group**, the unit you triage and get notified about. A finding **recovers by itself** when the measured condition clears, and closes by itself when its object leaves the tenant's scope, when its endpoint was replaced, or when nothing evaluated it for 24 hours. Triage, notes, snooze, dismissal, the severity override and assignment are the same as for every findings tenant; a Pool or Token traffic outlier finding can also be marked **Not an anomaly** (see *Traffic models*). The investigation of a HEC finding gives the decision in plain words, the measurement and what it was compared with, the recorded facts, the trend of the KPIs its policy judges, and **How it is evaluated**; **Open resource** opens the endpoint, pool or token drilldown in place.



The shared concepts are described once:

- *Findings, Finding groups and findings actions* — findings, statuses, triage, severity, notifications;
- *Finding groups* — Finding groups, automatic and manual;
- *Findings actions* — drilldowns, links, webhooks and automation rules;
- *Webhook accounts* — the accounts a webhook action posts with;
- *Assignment and automatic routing* — assignment groups, routing rules and team emails.

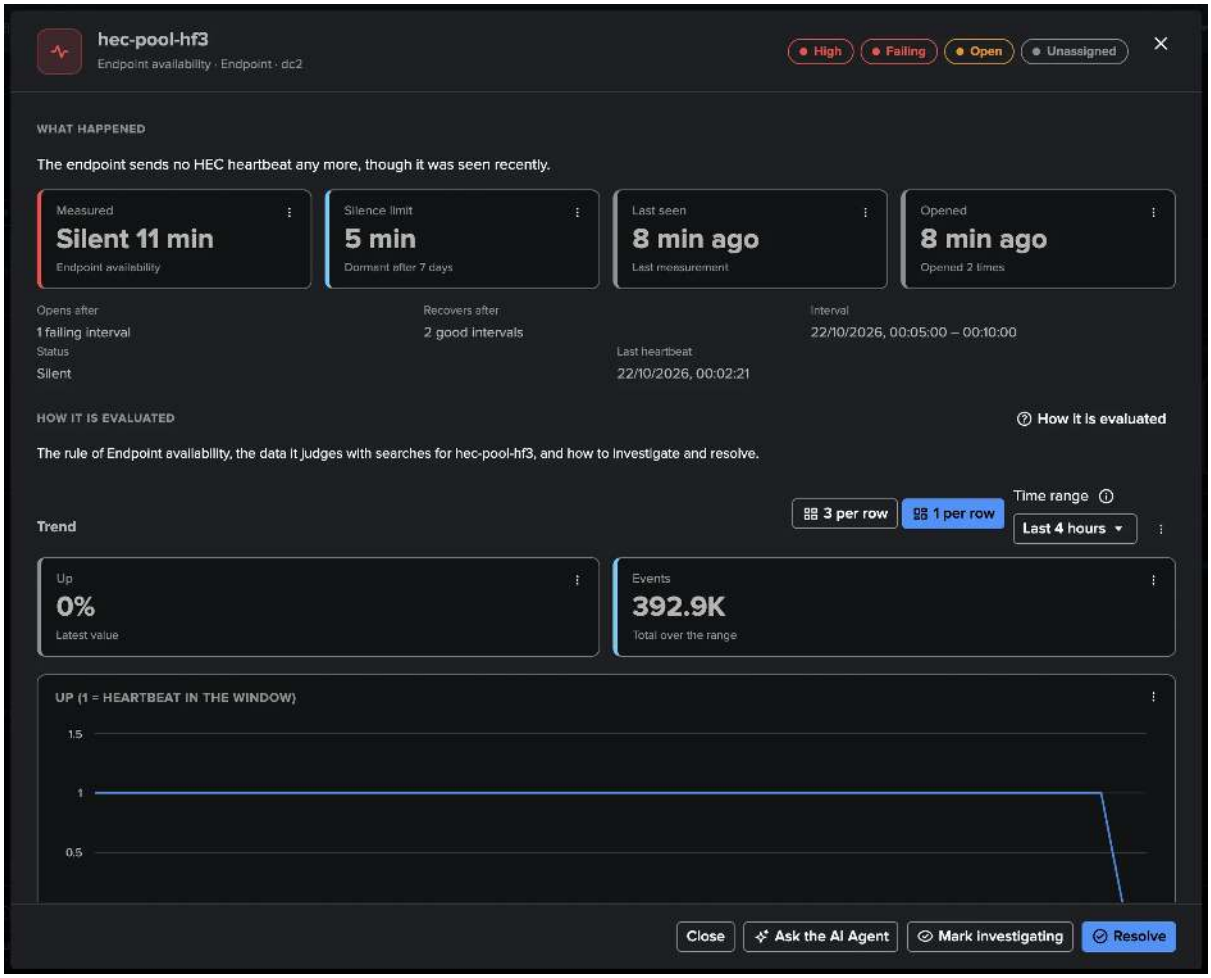
Links into the workspace. An email, an action or a dashboard can open the workspace on an object: `TenantHome?tenant_id=<tenant>&tab=endpoints&host=<host>` opens an endpoint, `tab=overview&pool=<pool id>` a pool and `tab=tokens&token=<token name>` a token.

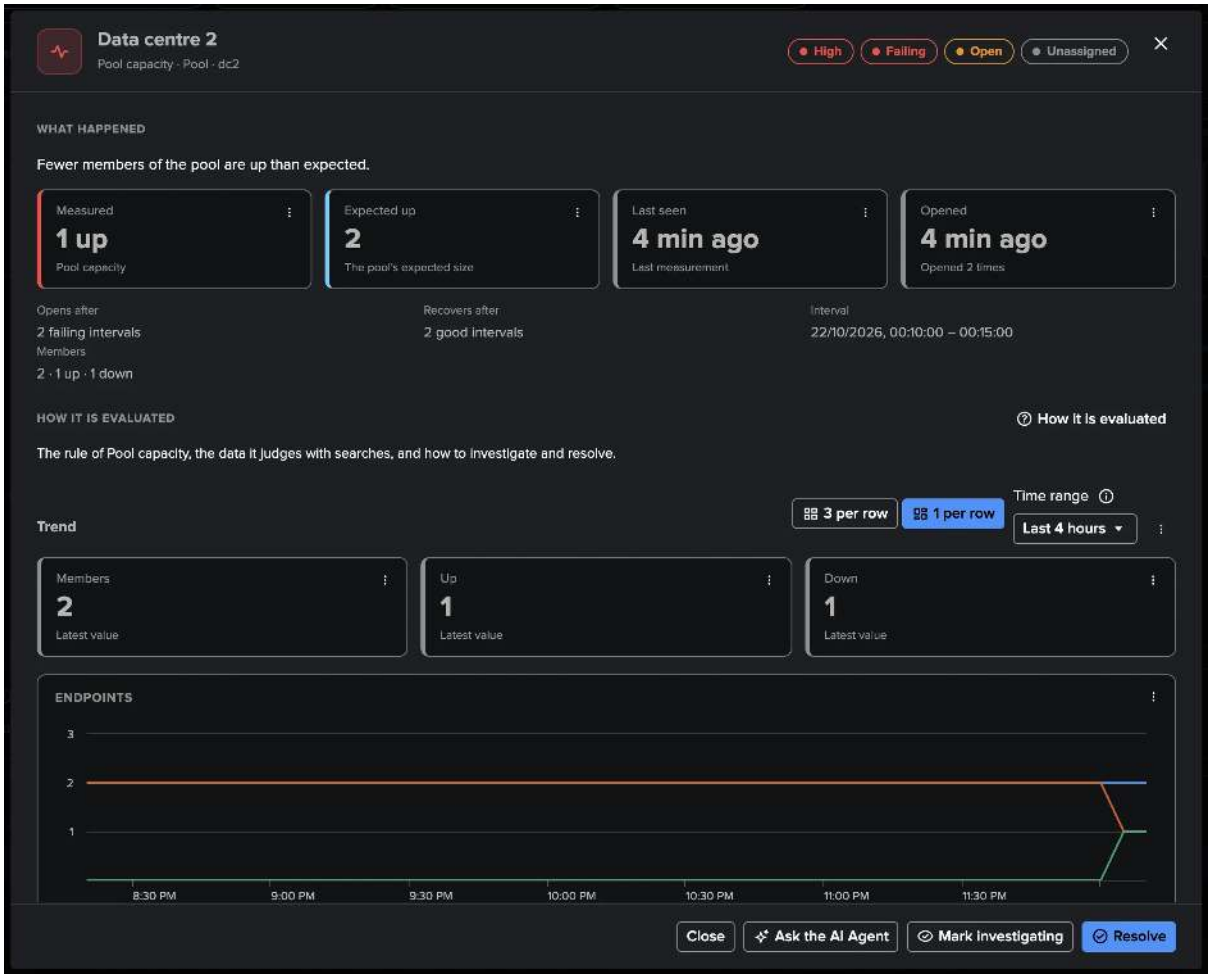
An endpoint that stops: what you see

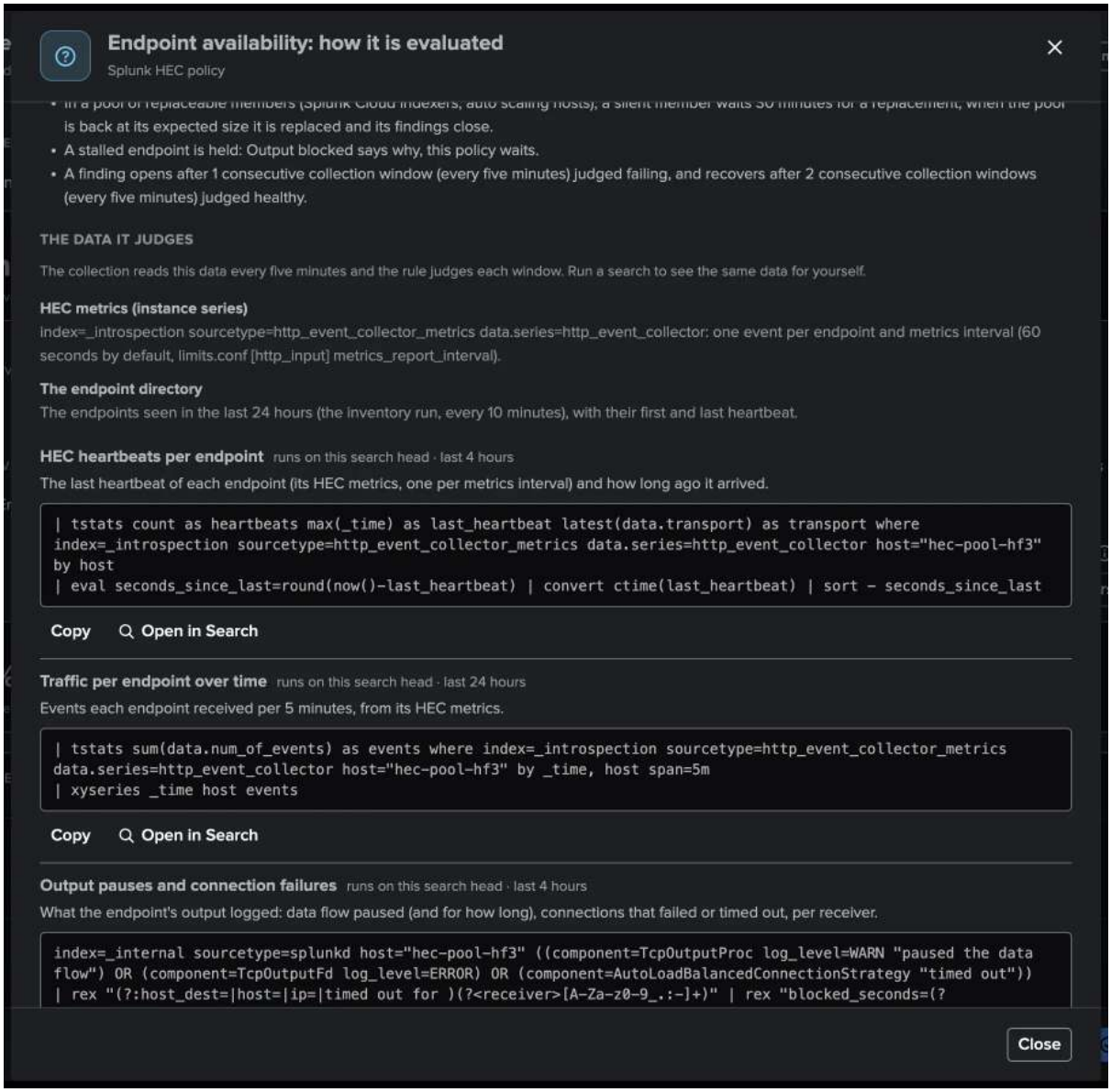
Take a pool of four heavy forwarders, `hf-1.example.com` to `hf-4.example.com`, behind a load balancer, and stop splunkd on `hf-2`:

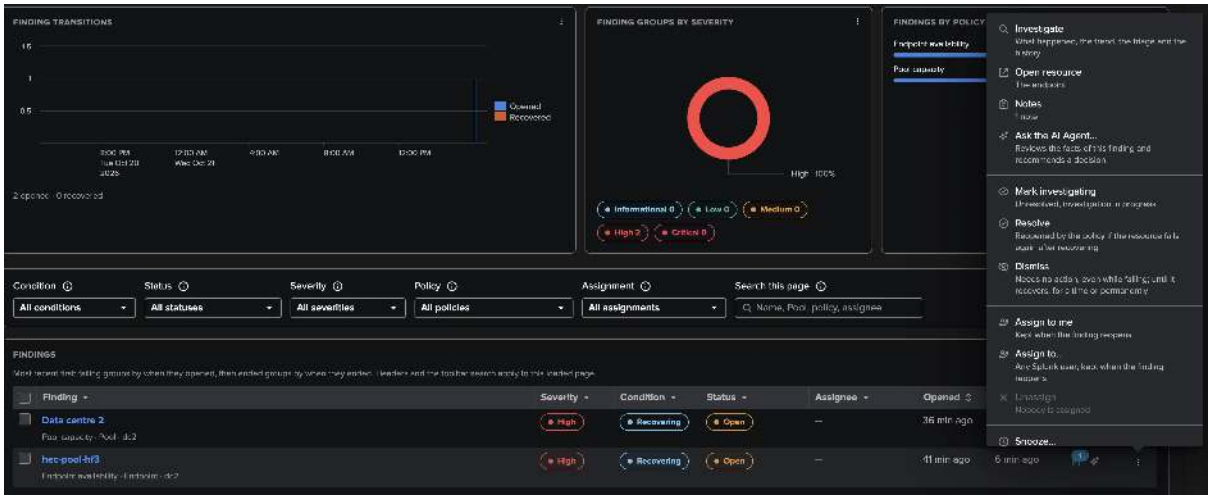
1. In the next window, `hf-2` is **late** (no heartbeat this window, one in the last 5 minutes): no finding yet, and Topology Studio draws it green with a *Late* cue.
2. Once it has sent no heartbeat for 5 minutes (**Silent after**, 300 seconds), it is **silent**: **Endpoint availability** opens a **High** finding on it at the first silent check. Had every member of the pool stopped, the finding would be **Critical**.
3. **Pool capacity** compares the members up (3) with the pool's expected size (4, set or learned) and opens a **High** finding on the pool after two checks (**Critical** below half the expected size).
4. The other members now take its share: Pool imbalance stays quiet, since no member up is starved.
5. When `hf-2` reports again, both findings recover after two healthy checks.

The screens below come from a deployment with two data centres, a pool of two heavy forwarders each, where `hec-pool-hf3` of *Data centre 2* was stopped: **Endpoint availability** opened on it at the first silent check, then **Pool capacity** on its pool, 1 of 2 members up, after two checks. On the generated HEC topology, both turn orange with their finding badge, and the endpoint availability falls to 75 % (the view at the top of this page).


hec-pool-hf3
Endpoint availability · Endpoint · dc2
High
Failing
Open
Unassigned
WHAT HAPPENED
The endpoint sends no HEC heartbeat any more, though it was seen recently.
Measured
Silent 11 min
Endpoint availability
Silence limit
5 min
Dormant after 7 days
Last seen
8 min ago
Last measurement
Opened
8 min ago
Opened 2 times
Opens after
1 failing interval
Recovers after
2 good intervals
Interval
22/10/2026, 00:05:00 – 00:10:00
Status
Silent
Last heartbeat
22/10/2026, 00:02:21
HOW IT IS EVALUATED
How it is evaluated
The rule of Endpoint availability, the data it judges with searches for hec-pool-hf3, and how to investigate and resolve.
Trend
3 per row
1 per row
Time range
Last 4 hours
Up
0%
Latest value
Events
392.9K
Total over the range
UP (1 = HEARTBEAT IN THE WINDOW)
1.5
1
0.5
Close
Ask the AI Agent
Mark investigating
Resolve


Data centre 2
Pool capacity · Pool · dc2
High
Failing
Open
Unassigned
WHAT HAPPENED
Fewer members of the pool are up than expected.
Measured
1 up
Pool capacity
Expected up
2
The pool's expected size
Last seen
4 min ago
Last measurement
Opened
4 min ago
Opened 2 times
Opens after
2 failing intervals
Recovers after
2 good intervals
Interval
22/10/2026, 00:10:00 – 00:15:00
Members
2 · 1 up · 1 down
HOW IT IS EVALUATED
How it is evaluated
The rule of Pool capacity, the data it judges with searches, and how to investigate and resolve.
Trend
3 per row
1 per row
Time range
Last 4 hours
Members
2
Latest value
Up
1
Latest value
Down
1
Latest value
ENDPOINTS
3
2
1
8:30 PM
9:00 PM
9:30 PM
10:00 PM
10:30 PM
11:00 PM
11:30 PM
Close
Ask the AI Agent
Mark investigating
Resolve





Had the forwarder's **output** been blocked instead (indexers not accepting data), its own telemetry would wait behind the same output and arrive late: the endpoint is **stalled**, Endpoint availability holds, and **Output blocked** opens a finding naming the destinations, their pauses and, when their logs are searchable, the receivers that stopped listening.

Notifications

Finding notifications work as for the Cribl tenants (*Findings, Finding groups and findings actions*): turn them on in **Settings → Notifications**, pick the events, a minimum severity, the policies, a delivery account and recipients, and choose a **digest** or **one email per Finding group**, with charts. Emails are sent after the policy evaluation and link to the finding in the workspace. **Send test email** checks the delivery.

Every transition is also written as an indexed event (`trackme:cribl:finding` and `trackme:cribl:finding_group`, `object_category=splunk_hec`), whatever the notification settings, for your own Splunk alerts, Enterprise Security or SOAR: see *Splunk HEC reference*.

Bank holidays

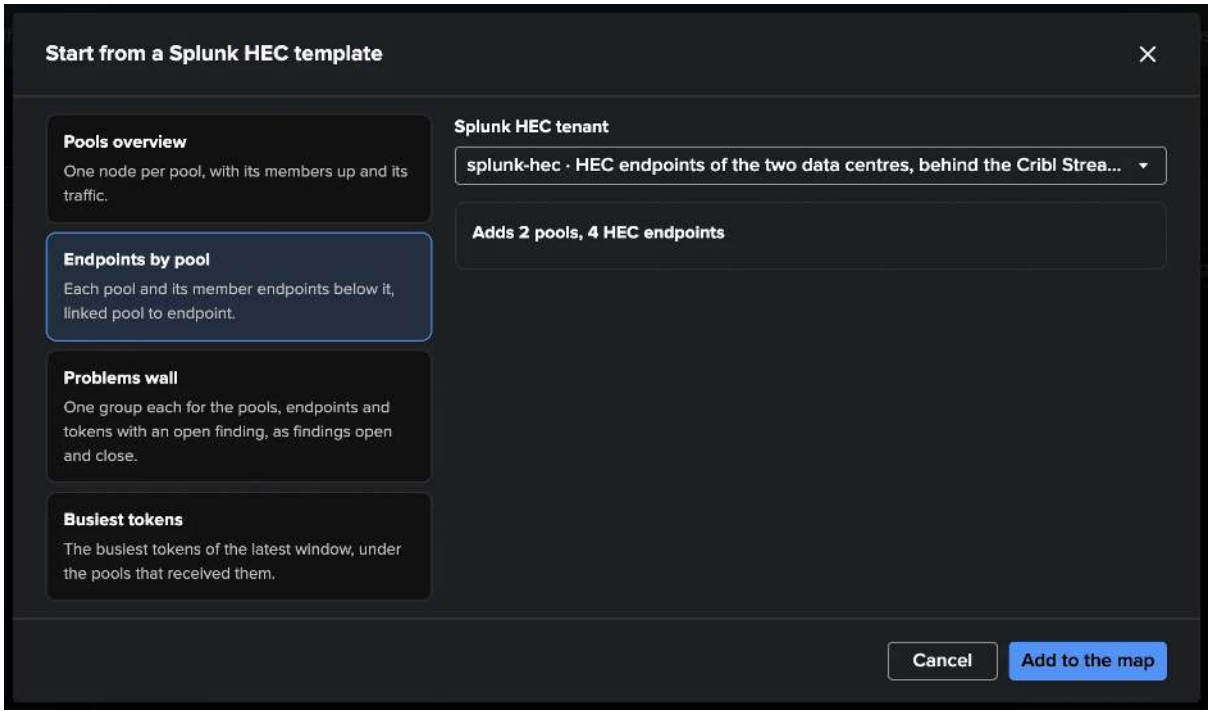
A Splunk HEC tenant uses the *bank holiday calendars* of its scope: the system default calendars, unless it selects its own or none. While a holiday of its calendars is active, the policies **set to hold** open no new finding. Each policy decides it in its editor (*During bank holidays*: **Hold new findings** or **Keep opening findings**). By default the policies that judge **traffic** hold — **Pool traffic drop**, **Pool traffic outlier**, **Token traffic outlier** and **Token silence** — since traffic is expected to change on a holiday; every other policy keeps opening findings, so an endpoint down on a holiday still opens its finding.

While a holiday holds a policy, a failure opens no new finding and sends no notification; a finding already open is still evaluated and can recover and close; a resolved or dismissed finding is not reopened by a held failure. When the calendars cannot be read, nothing is held.

Topology Studio

Splunk HEC objects are Topology Studio objects (*Topology Studio*): the **pools**, **endpoints** and **tokens** of a HEC tenant.

- **Add entities**: pick pools, endpoints (by pool, searchable, the discovered ones included) and tokens, each with its live status.
- **Groups** (*Add an aggregate node*, **Follow a selection**): a group of pools, endpoints or tokens, narrowed by pool (the *discovered* endpoints included), by a name that contains a text, and optionally to the objects with a finding. Start from **Endpoints of a pool**, **Every token** or **Anything failing**. A group follows its objects as they come and go (up to 2,000 objects per kind) and shows its health and healthy share; groups carry no summed KPI.
- **Templates**, offered when you can read a Splunk HEC tenant (on the empty canvas and in the *Add items* menu): **Pools overview** (one node per pool), **Endpoints by pool** (each pool over its endpoints, up to 10 pools of 12 endpoints), **Problems wall** (three groups: pools, endpoints and tokens with open findings) and **Busiest tokens** (the 12 busiest tokens under the pools that received them).
- **Status**: findings first (red or orange; blue when only snoozed). Otherwise, while the collected window and the policy evaluation are current: an endpoint up is green, a **late** endpoint green with a *Late* cue (Endpoint availability has not decided yet), a silent, dormant or stalled endpoint without a finding **quiet grey**; a pool at or above its expected size green, below it without a finding quiet; a token with events green, an idle one quiet. Quiet objects are disclosed and stay outside a group's healthy share.
- **KPIs**: for an endpoint, up, events, bytes, rates, requests, refused, error rate, fullest queue, ACK-enabled requests, CPU, memory; for a pool, members up, members, down, stalled, back pressure, traffic, refused, error rate, fullest queue; for a token, events, bytes, requests, refused, error rate. Endpoints, pools and tokens have 24-hour sparklines.
- **The integrity linter** reports a pool removed from the tenant's configuration or an endpoint no longer in the directory; a token is never reported removed.
- **Topology Alerts** (*Topology Alerts*) work on HEC objects and groups.
- **Build with AI** (*Build with AI — a topology from a description*) lists HEC tenants ("· Splunk HEC") with three starting points: **HEC endpoints by pool**, **HEC problems wall** and **Busiest HEC tokens**. The builder knows the pools, endpoints, busiest tokens and open findings by name, never a token value.
- Opening an object from a view opens the HEC workspace on it.

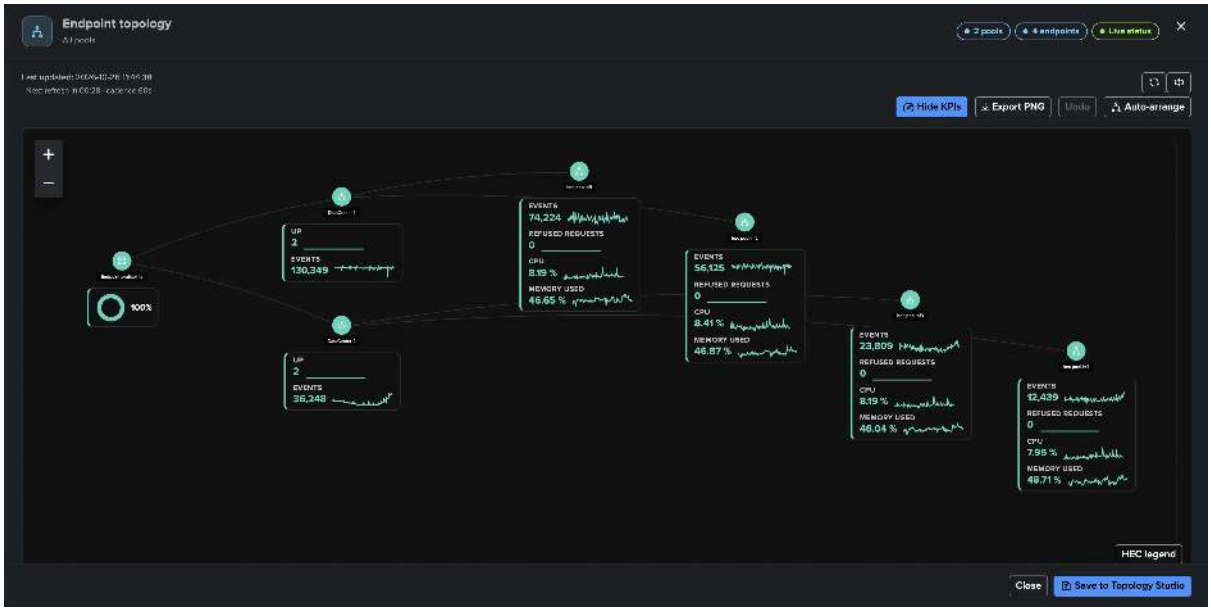


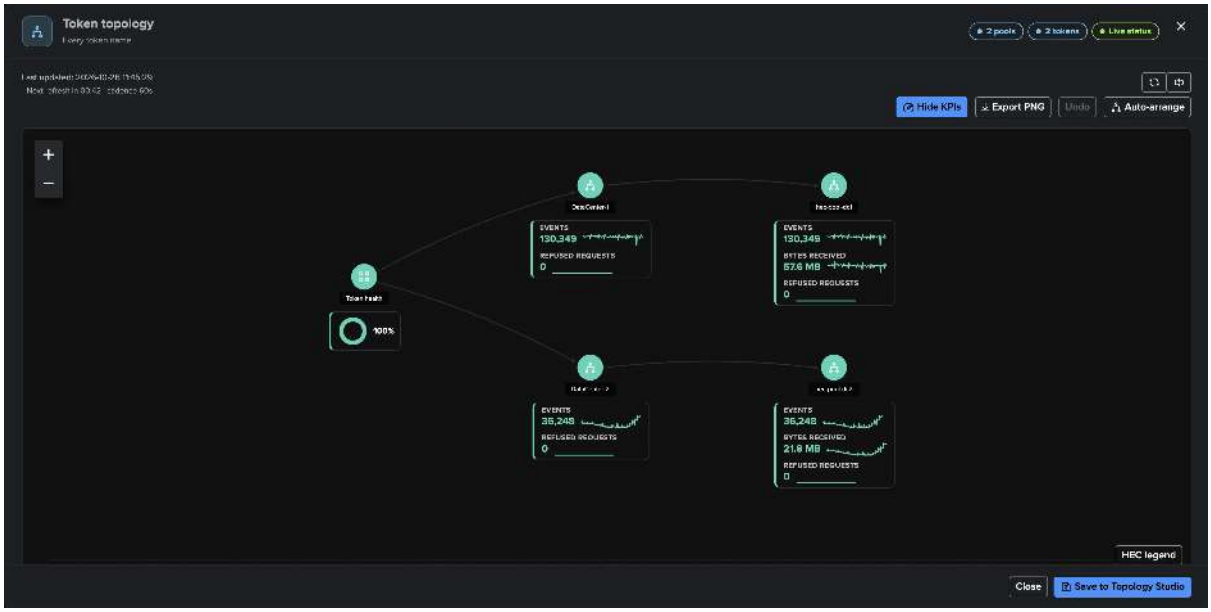
Generated topologies

Each workspace tab draws its own map on the Topology Studio canvas, the way the Cribl Stream Worker and Flow topologies do. **HEC topology** is on the Overview's Pools panel, **Endpoint topology** in the Endpoints toolbar and **Token topology** in the Tokens toolbar.

- **HEC topology**: the endpoint availability (a donut over every endpoint), each pool with its members up, events, bytes received and refused requests, and each endpoint below its pool with its events and refused requests.
- **Endpoint topology**: the endpoints the tab selects (its pool, status, search and order) under their pool, each with its events, refused requests, **CPU** and **memory**. Endpoints no pool claims sit under a *Discovered endpoints* group.
- **Token topology**: the tokens the tab lists under the pools that received them (a token sent to several pools is linked from each, one no pool received sits under a *No pool* group), with their events, bytes received and refused requests.

Every node shows its live status and its 24-hour sparklines, refreshed every minute; the canvas keeps the positions you give it, and **Auto-arrange**, **Hide KPIs** and **Export PNG** work as in Topology Studio. A topology draws up to 50 endpoints or tokens: when the tab selects more, it draws the first 50 and says how to narrow the selection. **Save to Topology Studio** saves it as a view, proposing its name and description, with the roles that can view and edit it. From then on it is a view like any other: edit it, share it, add Topology Alerts.





The Splunk HEC AI Agent

The **Splunk HEC AI Agent** reviews HEC findings the way an operator would, from facts TrackMe already holds. It is its own agent, built on the same framework as the other findings AI Agents (*AI Agents for findings*). It assesses each finding — **true positive**, **likely false positive**, **expected behaviour**, **endpoint or pool not in use** or **needs investigation** — recommends a decision, and keeps one current **[AI Agent]** note on the finding.

- **Inspect** only recommends: nothing changes but the note.
- **Act** applies the recommendation, but only what the tenant allows.

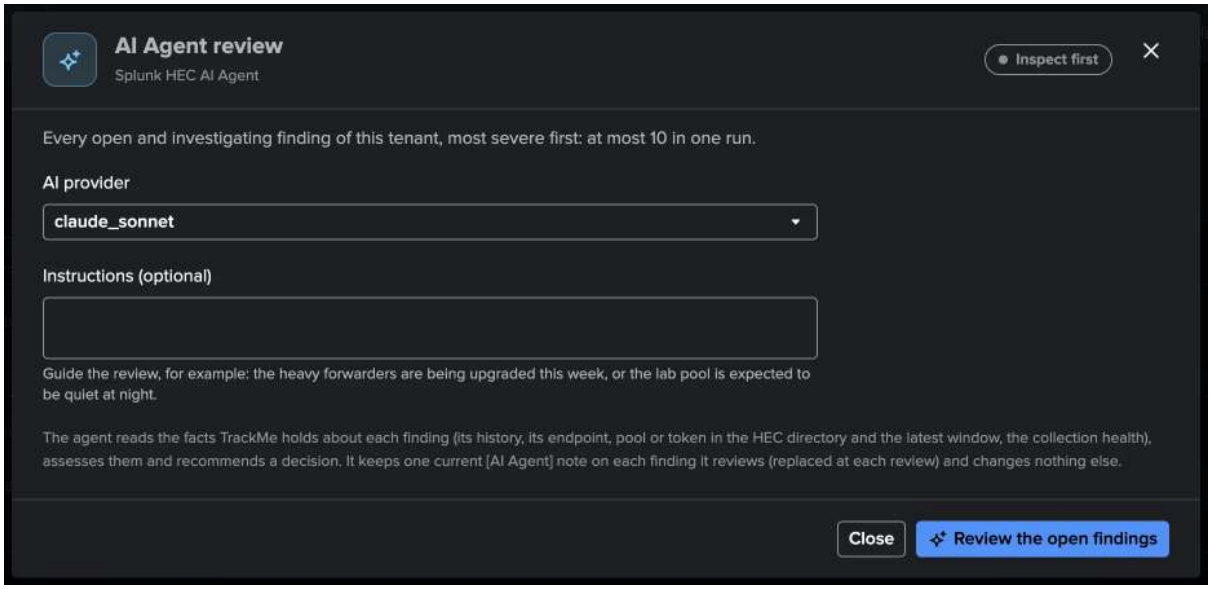
What it reads. The finding (its policy, settings, evidence, history, episodes and Finding group), and its object in the published directory and latest window: an **endpoint**'s status, last heartbeat, refusals, queues and bottleneck, output, version, resources and splunkd API status; a **pool**'s definition (expected size, replaceable) and its members; a **token**'s latest window, last 24 hours and endpoints, by name only. It also reads whether the endpoint or pool is still there (only a current directory can prove an endpoint gone, once the finding has not been evaluated for an hour; a token is never gone), the tenant's Collection health and its own past decisions. It runs no search.

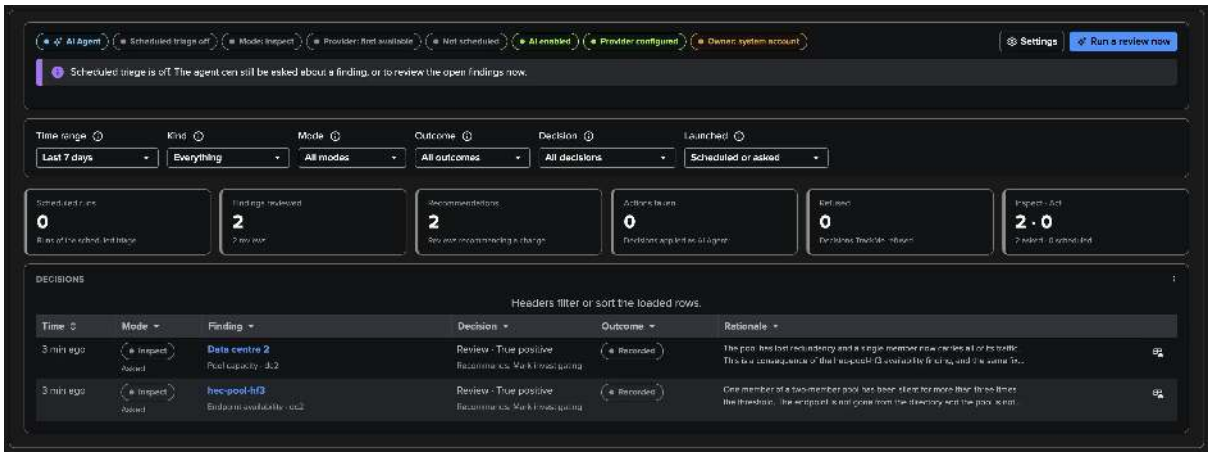
What it may decide (act mode, each decision with its own setting, checked again by TrackMe before it is applied):

Decision	Default	Condition checked by TrackMe
Mark investigating	Allowed	The finding is open.
Dismiss for a time	Allowed, 7 days at most (up to 90)	A **critical** or **high** finding that is still failing only when its endpoint is no longer in the directory (retired, dormant or replaced) or its pool is no longer defined.
Resolve	Not allowed	Only after a measured recovery.

The agent never dismisses permanently, assigns, or changes a severity, a policy, the pools or the source. A scheduled run never overrides an operator decision, and the agent makes at most one decision per Finding group in a run.

Asking and scheduling. Ask the AI Agent… is in the finding and Finding group menus and the investigation view, and **Run a review now** on the **AI Agent** tab reviews the open findings: the agent inspects first, then **Run its recommendations…** lets it act on that inspection. Scheduled triage is **off by default**: turned on in *Settings → AI Agent*, it reviews up to 10 findings an hour (one per Finding group, the most severe first, by default only critical and high), skips a finding reviewed in the last day unless it moved, and needs the AI Assistant enabled, an AI provider, the tenant enabled and a tenant owner that is a Splunk administrator account (not `nobody`). **Custom instructions** and **Evaluate…** work as for the other findings agents (*AI Agents for findings*). The settings and their defaults are in *Splunk HEC reference*; the **Manage AI Agents automation** page (*AI Agents automation*) lists the tenant with its agent.





AI Assistant

On a Splunk HEC tenant's Tenant Home, the AI Assistant (*AI Assistant & Concierge*) knows what the tenant is, its configuration, its policies and their settings, the pools, the endpoints needing attention, the busiest tokens and the open findings: ask it why an endpoint refuses requests, what a policy checks, or what to do about a finding. When a finding or the open findings would benefit from a review, it proposes a **Splunk HEC AI Agent** review as a consent card: nothing runs until you confirm. It proposes no Concierge action.

Troubleshooting

Symptom	What to check
The wizard lists no HEC endpoint	No endpoint reported HEC metrics on this source in the last 24 hours. Check that HEC is enabled, that the endpoints forward `_internal` and `_introspection` to indexers the source searches, the index names, and the host patterns.
The source is refused: `remote_account_refused`	The tenant owner holds none of the remote account's roles: its scheduled searches would be refused. Add one of the owner's roles to the account, or choose another owner.
The workspace shows no endpoint	The first collection has not published yet: **Collect now**, then **Collection health**.
Collection health: `no_hec_endpoints`	No endpoint for an hour after configuration: HEC disabled, internal indexes not searchable from the source, or host patterns matching nothing.
An endpoint is **late** or **stalled** but no finding opened	Expected: a late endpoint has reported in the last 5 minutes; a stalled one's telemetry waits behind its blocked output, and **Output blocked** says why.
Error rate is high but nothing is busy	Open the refused requests (*Why it was flagged*): the reply classes say whether senders use refused tokens (Authentication failures) or send data Splunk rejects (Rejected data). Empty load balancer probes (reply 5) are never counted.
Rejected data: incorrect index (reply 7)	The sender writes to an index the token may not use or that does not exist: create the index, add it to the token's allowed indexes, or fix the sender. The token drilldown names the index refused.
ACK not polled	A sender uses indexer acknowledgment but never polls `/services/collector/ack`, or a load balancer sends the polls to another endpoint: make it poll, disable `useACK` on its token, or make the load balancer sticky per channel.
Token consistency or the HEC settings of Configuration drift decide nothing	They need splunkd API accounts bound in *Settings → Source & pools*, with fresh facts from two members up of the pool. Splunk Cloud members are never compared. The **Endpoint directory** card says what the API read and why the rest was not.
Token silence decides nothing	No token is selected: type token names or add a lookup table, then use **Test selection**.
Collection health: `token_lookups_failing`	A Token silence lookup table could not be read, is too large or is empty. Check that the lookup exists, is shared with the TrackMe app, that the owner can read it and, on a remote deployment, that its remote account connects.
Pool or Token traffic outlier opens nothing	The models need 7 days of history (normal confidence) by default, and the traffic must reach the **At least** amount per five minutes.
Splunk Cloud indexers keep opening Endpoint availability findings	Mark their pool **replaceable** (the discovery proposes it for Splunk Cloud pools): a replaced indexer then opens nothing while the pool is at its expected size.
"Last 5 minutes" falls back to **Collected window**	The read of the last five minutes failed (the badge gives the reason). The collected data always shows; the next refresh tries again.
A run failed and the reason is unclear	The collector log, `index=_internal sourcetype=trackme:custom_commands:trackmecriblcollect` (file `trackme_cribl_collect.log`): one summary line per run, with the Splunk messages of a failed search.

What the Splunk HEC tenant does not do

- **No writes to Splunk**: TrackMe never changes a HEC setting or a token, and never restarts anything.
- **No token values and no event content**: tokens are known by name; the raw lines shown as evidence have token-shaped values masked.
- **No per-request tracing**: Splunk logs refused requests, not successful ones.
- **No health probe of the endpoints**: TrackMe never calls an endpoint's HEC port; the splunkd API enrichment is optional and never reads Splunk Cloud.
- **No Stop monitoring this object** on findings: narrow the host patterns or exclude the host instead.
- **No Concierge actions**: the AI Assistant answers questions and can propose a Splunk HEC AI Agent review.

Next: the reference

Every policy with its settings and defaults, the jobs and their schedules, the limits, the settings, the AI Agent, the indexed finding events, the permissions, the Configuration Guardian conditions and the run reasons: *Splunk HEC reference*.

8.3.20 Splunk HEC reference

The reference of the Splunk HEC tenant: the built-in policies and their defaults, the HEC reply codes, the collection jobs, the limits, the history TrackMe keeps, the settings, the AI Agent, the indexed finding events, the permissions, the Configuration Guardian conditions, the run reasons and the REST endpoints. The user guide is *Splunk HEC*; the findings model shared by every dedicated tenant type is *Findings, Finding groups and findings actions*.

Built-in policies

A Splunk HEC tenant has **eighteen** built-in policies. **Seventeen are on** out of the box; **Endpoint resources** is off (it watches the host, not HEC). **Token silence** is on but decides nothing until tokens are selected, and **Token consistency** decides only with the optional splunkd API. Your own SPL policies on endpoints, pools and tokens are described in *Custom policies*.

Endpoints

Policy	What it detects	Default	Severity
Endpoint availability	An endpoint seen recently sends no HEC heartbeat.	On	High → Critical
Senders refused	Requests refused as server busy (replies 9, 18–20, 23, 26, 27): HEC could not queue them.	On	High → Critical
Back pressure	An ingestion queue of the endpoint backed up; the finding names the stage that could not keep up and whether its thread was busy.	On	Medium → High
Output blocked	The endpoint's output to its receivers is the bottleneck, paused, or failing to connect.	On	High
Error rate	Refused requests out of the requests received (empty load-balancer probes excluded).	On	Medium
Authentication failures	Requests refused for an invalid, disabled or missing token.	On	Medium
ACK pressure	Indexer acknowledgment under pressure: the ACK store full, or replies saying the ACK service is unavailable, above its warning level or at capacity.	On	High
ACK not polled	Requests are acknowledged but no sender polls their acknowledgments.	On	Medium
Endpoint resources	CPU or memory of the endpoint's host at or above the threshold.	Off	Medium

Pools

Policy	What it detects	Default	Severity
Pool capacity	Fewer pool members up than expected.	On	High → Critical
Pool imbalance	A pool member up but receiving a very small share of the pool's events.	On	Low
Pool traffic drop	The pool receives far fewer events than it usually does at this hour of the day.	On	High → Critical
Pool traffic outlier	The pool's events (or bytes) outside what its traffic model expects at this hour of the week.	On	Medium
Token consistency	A token configured on some pool members only, or with a different status, index, allowed indexes or ACK from one member to another (splunkd API).	On	High
Configuration drift	Pool members running different Splunk versions or, with the splunkd API, different HEC settings, for longer than the grace period.	On	Medium

Tokens

Policy	What it detects	Default	Severity
Rejected data	Requests of a token refused for their content, or events dropped as oversized.	On	Medium → High
Token silence	A selected token sent no events for longer than the silence period.	On	High
Token traffic outlier	A token's events (or bytes) outside what its traffic model expects at this hour of the week.	On	Medium

Default settings

What each policy applies to, its checks to open and to recover (one check per collected 5-minute window), and its own settings with their defaults and ranges.

Policy (id)	One finding per	Open / recover	Settings and notes (defaults)
Endpoint availability (hec_endpoint_av	Endpoint	1 / 2	**Silent after** 300 s (300–86,400): no heartbeat for that long. **Dormant after** 7 days (1, 7 or 30): a longer silence opens nothing. **Replacement grace** 1,800 s (300–86,400): in a pool of replaceable members, a silent member waits that long for a replacement (see *Splunk HEC*, Pools). A late endpoint (a heartbeat in the last 5 minutes) and a stalled one (its output blocked) are held. Critical when every member of the pool is down.
Senders refused (hec_server_busy	Endpoint	2 / 3	**Open at** 10 busy refusals in a window (1–1,000,000,000). **Critical at** 20 % of the requests (1–100). Busy replies 9, 18–20, 23, 26, 27, and the HEC queue insertion failures when the endpoint reports them. Decided for an endpoint up in the window.
Back pressure (hec_queue_satur	Endpoint	3 / 3	No setting. A queue of the HEC path is saturated when its 90th percentile fill reaches 80 % or it was reported blocked in 30 % of its samples or more; back pressure is a saturated queue or busy refusals. High when the endpoint also refuses requests as busy.
Output blocked (hec_forwarding_	Endpoint	2 / 2	**Pause counted from** 30 s (1–3,600): an output pause that long in the window; a failing destination or the output as the bottleneck counts too, and so does a stalled endpoint.
Error rate (hec_error_rate)	Endpoint	3 / 3	**Open at** 5 % refused (0.01–100), **Recover below** 2 % (below the open threshold), **At least** 100 requests in the window (1–1,000,000,000). Load balancer probes (reply 5) excluded.
Authentication failures (hec_auth_failur	Endpoint	2 / 3	**Open at** 20 token refusals in a window (1–1,000,000,000): replies 1, 2, 3, 4, 21, 22. The detail names the reply codes and the clients refused most.
ACK pressure (hec_ack_pressur	Endpoint	3 / 3	No setting. The ACK service logged its store full (`HttpInputAckService`), or requests were refused with replies 19, 20, 25 or 27.
ACK not polled (hec_ack_unpolle	Endpoint	6 / 2	**Open at** 1 request acknowledged (1–1,000,000,000) in a window that received no ACK poll; after 6 checks (30 minutes), so a sender polling every few minutes never opens it. Decided where Splunk reports both counters (9.4 and 10.x); on 10.x the detail names the tokens acknowledged with no poll.
Endpoint resources (hec_endpoint_re	Endpoint	3 / 3	Off by default. **Open at** 90 % CPU or memory (0–100), **Recover at** 80 % (below the open threshold).
Pool capacity (hec_pool_capaci	Pool	2 / 2	**Critical below** 50 % of the expected members (1–100), or with none up. Expected size: the pool's own, else learned (the second highest daily peak of members up over 7 days). Nothing is decided while the size is being learned.
Pool imbalance (hec_pool_imbala	Pool	6 / 3	**Smallest share** 5 % of the pool's events (0.01–50), **At least** 10,000 events in the window (1–1,000,000,000,000), at least two members up. A member first seen less than 15 minutes ago is skipped (warm-up).
Pool traffic drop (hec_pool_traffi	Pool	3 / 2	**Open below** 10 % of the usual traffic (1–90): the median rate of the same hour of the day over the last 7 days, from 3 days at least. **At least** 1,000 usual events an hour (1–1,000,000,000,000). Critical when the pool received no event. Holds on bank holidays by default.
Pool traffic outlier (hec_pool_traffi	Pool	2 / 2	**Traffic judged**: events (default) or bytes received. **Direction**: drops and spikes (default), drops only, spikes only. **At least** 100 per five minutes (0 or more): below it in both the window and the model, nothing is judged. **Open from**: normal confidence (7 days learned, default) or low confidence (1 day). Holds on bank holidays by default.
Token consistency (hec_token_consi	Pool	2 / 2	No setting. Compares the members up of a pool with fresh splunkd API facts (two at least, else nothing is decided): a token missing on some members (only when no member's token list was cut at 200), or differing in status, index, allowed indexes or ACK.
Configuration drift	Pool	2 / 2	**Grace** 60 minutes (5–10,080): a difference lasting that long opens a finding. Splunk versions (from the telemetry) and, with

Settings every policy has. On / off; **severity** (Low, Medium, High, Critical; the policies above raise it in the cases listed); **checks to open** and **checks to recover** (1 to 12 consecutive windows, one every 5 minutes); **During bank holidays** (**Hold new findings** or **Keep opening findings**; see *Bank holidays*); and for the endpoint and pool policies, a **pool scope** (empty: every pool; the endpoint policies also take the *discovered* endpoints). A token policy has no pool scope.

How a check works. A check is one collected 5-minute window: the policies run after every **HEC activity** run, on the window it published. A window older than 20 minutes is not judged (`hec_live_stale`). The counter policies judge only the endpoints measured in the window.

How findings close by themselves. A finding recovers when the measured condition clears for the checks to recover. It closes automatically when its object leaves the tenant's scope (a host pattern or pool changed, a policy turned off or out of scope), when its endpoint was replaced in a replaceable pool (closing reason *endpoint replaced*, never a recovery), or when nothing evaluated it for 24 hours. A condition TrackMe cannot decide (*unknown*, for example a pool still learning its size) is never written as a recovery.

HEC reply codes

The reply code Splunk logs for each refused request (`reply=` in `HttpInputDataHandler`), and how the tenant counts it:

Class	Reply codes	Counted by
Busy	9, 18, 19, 20, 23, 26, 27	Senders refused, Back pressure, Error rate
Token refused	1, 2, 3, 4, 21, 22	Authentication failures, Error rate
Rejected data	6, 7, 12, 13, 15	Rejected data, Error rate
ACK	19, 20, 25, 27	ACK pressure
Channel, configuration	10, 11, 14, 16	Error rate
No data (load balancer probes)	5	Counted and shown, **never** a sender error
Capacity warning	24, 25	Shown (only with Splunk's `warn_at_80` back pressure state)

`num_of_errors` of the HEC metrics is collected but never used: it does not count refusals. The error rate is the refused requests (every class but the probes) over `num_of_requests`, which includes the refused requests.

Bank holidays

A Splunk HEC tenant uses the bank holiday calendars of its scope (*Bank holidays*). While a holiday is active, the policies set to hold open no new finding; a finding already open is still evaluated and can recover and close.

Policy	Default
Pool traffic drop, Pool traffic outlier, Token traffic outlier, Token silence	Hold new findings
Every other policy	Keep opening findings

A rule saved before this setting existed reads its default. When the calendars cannot be read, nothing is held. The traffic models leave the hours of the tenant's holidays out of their training when the *Exclude from ML training* effect applies to the tenant; a run that cannot read the holidays learns nothing and fails with `ml_holidays_unavailable` (the next run retries).

Custom policies

A custom policy is an SPL search on the tenant's Splunk data whose result rows each name a HEC object. How to write one: *Splunk HEC*, section Custom policies.

Object type	Resolution
Endpoint	A host name within the tenant's host patterns; its pool is the one the configuration assigns (*discovered* when none).
Pool	A pool id of the tenant.
Token	A token name.

- A custom finding takes the subject of the built-in findings on the same object, so it joins that object's Finding group.
- A host outside the tenant's host patterns or a pool the tenant does not define is **not monitored**: skipped, counted in the run's receipt, and its earlier findings recover. Only a value that cannot name an object is unresolved; a run with an unresolved row recovers nothing.
- **Runs, lifecycle and limits**: the same as on a Cribl Stream tenant, see *Custom policies*.
- **Health**: *Collection health → Custom policies*, and the `custom_*` problems of the Configuration Guardian check `cribl_collection_health`.

Collection jobs

Each Splunk HEC tenant has its collection reports, named `trackme_splunk_hec_<job>_<generation>_tenant_<tenant_id>` run as the tenant owner, and the AI Agent report (below). Each tenant's schedule runs at a minute offset of its own.

Job (Collection health label)	Schedule	What it does
`live` (**HEC activity**)	Every 5 minutes	Reads the last settled 5-minute window by index time, in two waves of searches dispatched together (wave 1: the instance and token series of the HEC metrics; wave 2: the refused requests, the queues and thread duty cycles, the output, the host resources), then the receivers of the endpoints whose output was in trouble. Publishes the window, writes the history points, evaluates the policies, sends the notifications.
`inventory` (**Endpoint directory**)	Every 10 minutes; rebuilt about once an hour	Rebuilds the endpoint and token directory (the endpoints seen in the last 24 hours, kept while they reported in the last 7 days; the token names seen in the last 24 hours; the identity from `Host-wide`) and, with accounts bound, reads the splunkd API. Between rebuilds, confirms the directory is current. A manual run always rebuilds.
`mltrain` (**Traffic models**)	Twice an hour	Learns the traffic models of the pools and tokens from the HEC metrics, then projects the expected ranges of the next 24 hours.
`policies` (**Monitoring policies**)	After every live run; on demand	Evaluates the policies on the newest window (**Run now**).
`trackme_hec_agent_` (AI Agent)	Hourly, when enabled	Scheduled triage by the Splunk HEC AI Agent. Off by default: the report stays unscheduled until the triage is turned on.

Run budgets. Each job runs at most its interval less 30 seconds (the live and policies jobs 270 s, the directory 570 s, the traffic models 1,770 s), and a scheduled run ends 30 seconds before its next scheduled run, however late the scheduler started it: a late run reads less and the next one carries on. A run started with less than 15 seconds left does nothing and says so (`collection_time_budget`). The tenant's reports carry no scheduler skew (`allow_skew = 0`, set again at every schedule reconciliation).

Windows. A live run reads what became searchable since the previous run, up to the last 5-minute boundary at least 120 seconds old (the settle delay, never less than 60 seconds), and scans 30 minutes of event time to find it: telemetry more than 30 minutes late is not counted. After missed runs, at most three windows are read (one history point each); older windows stay gaps. A window already counted is never read again. Rates are per second of measured activity.

Searches. A search is judged by its job state: Splunk messages attached to a search are recorded on the run (never a failure), and a search Splunk completed short is used and flagged. Only a failed job, a time limit or a broken read fails the run. The searches name the directory's endpoints (up to 500, else the host patterns).

Scheduled collection off (*Settings → Collection*) unschedules the reports; **Collect now** and **Run now** still work. A live run never builds the directory: a new tenant's first live run answers `hec_directory_pending` and starts the directory job beside it (2.4.23). A search may use what is left of its job's budget: there is no fixed per-search timeout. The reports run `| trackmesplunkcollect` (log `trackme_splunk_collect.log`).

Limits and history

Bound	Value
Endpoints per run	5,000 (beyond, counted as truncated; Collection health says so)
Tokens per run	20,000
Refused-request rows, output rows, receiver rows per run	20,000 each
Queue rows per run	50,000
Receivers read per run	200
Client addresses kept per endpoint	20
Indexes refused kept per token	10
Endpoints listed per token	200
Pools per tenant	200 (host patterns per pool: 50)
Tenant host patterns	100
Hand assignments	500
Expected size of a pool	1 to 10,000
splunkd API accounts / URLs per account / tokens read per endpoint	20 / 50 / 200
Token silence: typed names / lookup tables / names per lookup	100 / 20 / 100,000
Token silence: tokens selected	20,000; at most 250 new silence findings started per run (the others start at the next runs)
Token traffic models	500 tokens (busiest first)
Traffic model history learned on the first runs (**Model learning history**)	14 days by default; from today on, 7, 14, 30, 60 or 90 days (at most what `_introspection` keeps: 14 days by default)
Traffic models with their own settings	500
Endpoints, pools or tokens in one Topology Studio group	2,000 per kind (narrow the group beyond)
HEC references per tenant in one Topology Studio view	100 (use groups for more)
Rows per page of the Endpoints and Tokens tabs	Up to 100 (the viewer's preference)

History kept by TrackMe. At every live run, in the tenant's TrackMe metric index (through `trackme_hec_metrics.log`):

- `trackme.hec.endpoint.*` per endpoint (dimension `endpoint`, the host): `up`, `requests`, `events`, `bytes_received`, `errors`, `parser_errors`, `auth_failures`, `busy_replies`, `refused`, `failures`, `probe_noise`, `waiting_ack`, `oversized_events`, `worst_queue_fill`, `queue_insertion_failures`, `back_pressure`, `output_paused_seconds`, `cpu_pct`, `mem_used_pct`;
- `trackme.hec.pool.*` per pool (dimension `pool_id`): `members`, `up`, `down`, `stalled`, `requests`, `events`, `bytes_received`, `auth_failures`, `busy_replies`, `refused`, `failures`, `oversized_events`, `waiting_ack`, `worst_queue_fill`, `back_pressure`;
- `trackme.hec.token.*` (dimension `token_id`, a digest of the name, never the name): `events`, `bytes_received`, `requests`, `refused`, for the 50 busiest tokens of the window with 10 places kept for tokens with refused requests.

Every point carries `tenant_id` and the end of its window. They draw the trends of the workspace, the drilldown histories and the Topology Studio sparklines. `waiting_ack` is the ACK-enabled requests of the busiest minute, a count per interval, not a backlog.

Settings reference

Setting (where)	Default	Meaning
Source (wizard; Settings → Source & pools)	`local`	This search tier, or a TrackMe remote account to the search tier that sees the endpoints' internal indexes. The tenant owner must hold one of the account's roles. Locked once configured.
Internal index, **Introspection index** (same)	`_internal`, `_introspection`	Where the endpoints' `splunkd.log` / `metrics.log` and HEC metrics are. Locked once configured.
Endpoint host patterns, **Excluded hosts** (same)	`*`, none	The endpoints in scope (`*` and `?` wildcards). Changing them starts a new source revision; an excluded host is never monitored.
Pools (same)	A `default` pool	Identifier (lowercase letters, digits, - and _, up to 40 characters, fixed once saved; never `discovered` or `tokens`), name (up to 80 characters), host patterns (the default pool needs none), expected size, replaceable.
Assignments (same, the endpoint's pool pill)	None	An endpoint pinned to a pool by its host name; never to a replaceable pool.
Auto-join (same)	On	A new endpoint no pool claims joins the default pool; off, it waits as discovered.
splunkd API accounts (same)	None	Up to 20 remote accounts (never `local`) whose splunkd the directory reads. See *Splunk HEC*, The splunkd API enrichment.
Scheduled collection (Settings → Collection)	On	Runs the live job every 5 minutes, the directory about hourly, the traffic models twice an hour.
Model learning history (wizard; Settings → Collection)	14 days	How far back the traffic models learn on the first runs, and what a relearned model reads: from today on, 7, 14, 30, 60 or 90 days.
Traffic ML models (Settings → Traffic ML models; a model's **Tune model**)	Recent days half the weight after 14 days, automatic seasonality, weekday pattern after 28 days, findings once 7 days are learned, normal sensitivity, deviations under 10 % ignored	The defaults every traffic model follows; a model's own settings add the training days, the direction, a pause and excluded periods. See *Splunk HEC*, Traffic models.
Bank holidays (Manage Bank Holidays; Configure tenant → Bank Holidays)	The system default calendars	The tenant's calendars. See *Bank holidays*.
Notifications (Settings → Notifications)	Off	Finding emails: events, minimum severity, policies, delivery account and recipients, digest, charts.
Assignment (Settings → Assignment)	None	Assignment groups, team email, ordered routing rules, manual overrides. See *Assignment and automatic routing*.
Finding actions (Settings → Finding actions; a policy's Actions and Automations)	None	Drilldown, link and webhook actions, and the Automations that call them. See *Findings actions*.
AI Agent (Settings → AI Agent)	Scheduled triage off	See *AI Agent*.
Policies (Policies tab)	See *Built-in policies*	Saved for the tenant with their own revision.

The span of a window (300 seconds) and the settle delay (120 seconds, up to 600) are part of the source configuration.

The tenant's indexes, roles and owner are set in the wizard and edited like any tenant's.

Token silence selection

Token silence watches the tokens that must keep sending, from two sources that add up:

Source	Rules
Typed token names	Up to 100 names or patterns, case-sensitive (a token name is the HEC input's name), 1 to 256 characters without a control character, a quote or a backslash. `*` matches any characters and `?` one character.
Splunk lookup tables	Up to 20 per policy, each a lookup shared with the TrackMe app, on the local search tier or on a remote deployment through a TrackMe remote account (never a TrackMe KV store lookup), with the field holding the token names. Up to 100,000 distinct names per lookup; a larger one is refused (`too_large`), never truncated. Values are trimmed; values that cannot be a token name are counted as ignored.
Match mode (per lookup)	**Exact** (default): whole token names; a value holding `*` or `?` is ignored. **Wildcard**: `*` and `?` match.

Decisions. Every name selected exactly — typed or from an Exact lookup — is selected whether seen or not: a name not seen sending in the last 24 hours is silent at once. The patterns select the known names they match (the latest window, the 24-hour directory, the tokens already followed). At most 20,000 tokens are selected, exact names first.

Refresh. The policies run reads each lookup again every 15 minutes, the oldest first, and keeps its names between reads. A lookup that cannot be read keeps its last good names; a lookup never read successfully selects nothing. Collection health reports `token_lookups_failing` when a lookup is unreadable, too large or empty.

Test selection runs as a search: it reads every lookup now and matches the published token directory and the latest window, then shows the tokens selected by state (sending within the draft's silence period, silent, not seen in 24 hours), up to 50 examples (not seen first) with the entry that selected each, each source's names, and the patterns selecting none. It saves nothing. Token names are selectors: they are not copied into the evidence of the findings.

AI Agent

The Splunk HEC AI Agent triages the tenant's findings from what TrackMe holds: the finding and its HEC detail, history, episodes and Finding group; its endpoint, pool or token in the published directory and latest window; whether the endpoint or pool is still there; Collection health; and its own past decisions. It runs no search. The user guide is in *Splunk HEC*; the shared model in *AI Agents for findings*.

Assessments: true positive, likely false positive, expected behaviour, endpoint or pool not in use, needs investigation.

Settings (*Settings → AI Agent*; the core ones also in **Manage AI Agents automation**, *AI Agents automation*, and in *Configure tenant*), stored in the tenant record:

Setting	Field	Default	Range and notes
Scheduled triage	ai_hec_agent_enabled	Off (0)	Asking the agent needs only AI to be available.
Mode	ai_hec_agent_mode	Inspect	inspect (recommend only) or act (apply allowed decisions); scheduled runs.
AI provider	ai_hec_agent_provider_name	First available	An AI provider account name.
Review interval (days)	ai_hec_agent_min_days_betwe	1	0 reviews every run; a new episode is reviewed at once.
Findings per run	ai_hec_agent_max_findings_pe	10	1 to 50, most severe first.
Time budget per run (seconds)	ai_hec_agent_max_runtime_se	3,000	60 to 86,400.
Custom instructions	ai_hec_agent_custom_instruct	Empty	At most 4,000 characters, line breaks kept; followed by every run; never widen what the agent may do.
Allow Mark investigating	ai_hec_agent_allow_investig	Allowed (1)	An open finding only.
Allow Dismiss for a time	ai_hec_agent_allow_dismiss_t	Allowed (1)	A critical or high finding still failing only when its endpoint is no longer in the directory (retired, dormant or replaced) or its pool is no longer defined.
Longest dismissal (days)	ai_hec_agent_dismiss_max_da	7	1 to 90.
Allow Resolve	ai_hec_agent_allow_resolve	Not allowed (0)	Only after a measured recovery.

The shared filters of the automated AI reviews, ai_automated_priority_filter (by default critical and high findings) and ai_automated_filter_expression, apply to the scheduled selection with the finding severity as the priority.

Gone from the directory. An endpoint is gone only when a **current** directory no longer lists it and the finding has not been evaluated for an hour; a pool is gone when the tenant no longer defines it. A token is never gone.

Scheduled triage runs the report trackme_hec_agent_tracker_tenant_<tenant_id> hourly when all of these hold: the AI Assistant is enabled, an AI provider is configured, the tenant is enabled, and scheduled triage is on. The report runs as the tenant owner, which must be a Splunk administrator account (a member of admin or sc_admin), not nobody: the run records itself in a store only they can write. Saving the settings schedules or unschedules it, and the tenant health tracker checks the same conditions every five minutes.

Audit. Every run is indexed to the tenant summary index with the sourcetype trackme:ai_agent:hec_agent:inspect or trackme:ai_agent:hec_agent:act; decisions appear in the finding's history under the actor **AI Agent**, and the reviewed finding carries one current [AI Agent] note.

```
index=trackme_summary sourcetype="trackme:ai_agent:hec_agent:*"␣
↪tenant_id="example-hec"
| table _time, sourcetype, launched_by, automated, status
```

Indexed finding events

Every transition of a finding and of a Finding group is written as an event, whatever the notification settings:

Item	Value
Index	The tenant's notable index (`trackme_notable` by default)
Sourcetypes	`trackme:cribl:finding` (a finding) and `trackme:cribl:finding_group` (a Finding group), shared with the Cribl tenants
`object_category`	`splunk_hec` (with `tenant_id`)

Example: the HEC findings opened or reopened in the last day.

```
index=trackme_notable sourcetype=trackme:cribl:finding object_category=splunk_hec␣
↪tenant_id="example-hec"
    transition IN ("opened", "reopened")
| table _time, policy_name, entity_name, severity, reason, finding_id
```

Permissions

Each tier needs the TrackMe capability **and** one of the tenant's matching roles; a Splunk administrator (`admin`, `sc_admin`) holds every tier on every tenant.

Action	Who
Read every tab, the drilldowns, the last five minutes and *Why it was flagged*; run drilldown and link actions	Any role of the tenant (`trackmeuseroperations`). The reads run with your own rights: a remote source also checks the remote account's roles.
Triage findings and Finding groups (investigate, assign, snooze, dismiss, resolve, change the severity, group by hand), notes; ask the AI Agent and let it act on its inspection	The tenant's power and admin roles (`trackmepoweroperations`)
Policies (on / off, severity, conditions, scope, bank holidays), the Token silence lookups and Test selection, custom policies	A TrackMe administrator of the tenant (`trackmeadminoperations` and a tenant admin role)
Source & pools, Collection, **Collect now** / **Run now**, Repair workspace, Notifications, AI Agent settings, Assignment settings, Finding actions and Automations	A TrackMe administrator of the tenant
Create or delete the tenant; manage remote accounts and Webhook accounts	TrackMe administrators (see *Remote Splunk deployments* and *Webhook accounts*)

The **scheduled** AI Agent needs a tenant owner that is a Splunk administrator account (not `nobody`). A remote account used as the source or for the splunkd API must be usable by the tenant owner (one of its roles).

Configuration Guardian

The tenant check `cribl_collection_health` raises **Splunk HEC collection needs attention** (warning) when collection does not run as configured, and clears by itself on the first run without a problem. The Collection health tab shows the same conditions.

Condition	In plain words
`live_never_ran`, `live_not_running`	No live HEC run since the source was configured, or none for 20 minutes. Check that the reports exist, are scheduled and that their owner can run searches.
`live_failing`, `inventory_failing`, `policies_failing`	A job keeps failing (30 minutes for live and policies, 90 minutes for the directory). Search the collector log, and check that the tenant's remote account connects.
`inventory_never_ran`, `inventory_not_running`	No endpoint directory since the source was configured, or none for 3 hours.
`no_hec_endpoints`	No endpoint for an hour after configuration: HEC disabled on the endpoints, the internal and introspection indexes not searchable from the source, or host patterns matching nothing.
`capacity_reached`	A collection limit was reached (more endpoints or rows than a run keeps). Narrow the host patterns or exclude hosts.
`token_lookups_failing`	A Token silence lookup table could not be read, holds more than 100,000 names, or is empty, at the last policies run. A lookup that cannot be read keeps its last names. Check the lookup, its sharing with the TrackMe app, and the remote account when it is on a remote deployment.
`custom_schedule_drift`, `custom_policy_failing`, `custom_missed_schedule`, `custom_recovery_required`, `custom_health_unavailable`	A custom policy's report or runs need attention: the same conditions as on a Cribl Stream tenant (see *Custom policies*).
`notification_delivery_failing`	Notifications are on and the emails have failed for 30 minutes. Collection never waits for delivery.

Each alert carries its recommended actions. See *Configuration Guardian*. Also relevant: `ai_automation_owner_is_system_user` warns when the AI Agent's scheduled triage is on and the tenant owner is `nobody` or a system account.

Run reasons

The outcome each job records, shown on its Collection health card:

Reason	Meaning
`live_published` / `window_current`	The latest HEC window was collected / was already collected.
`directory_published` / `directory_current`	The endpoint directory was rebuilt / is up to date.
`hec_policies_evaluated`	The policies were evaluated on the latest window.
`hec_live_not_published`	No HEC window published yet.
`hec_live_stale`	The latest window is more than 20 minutes old: it is not judged.
`hec_policy_time_budget`	The policies ran out of time.
`hec_scope_changed`, `source_revision_changed`	The source changed during the run; the next run reads the current settings.
`hec_snapshot_changed`	A new window was published while reading; the next run evaluates it.
`hec_policies_unavailable`	The policies could not be evaluated.
`remote_account_refused`	The remote account refused the tenant owner. Add one of the owner's roles to the account, or change the owner.
`search_failed`, `search_timeout`	A collection search failed or exceeded its time limit. The collector log carries the Splunk messages of the failed search.
`models_current`, `training_continues`	The traffic models are up to date / are still learning the history (the next run continues).
`ml_holidays_unavailable`	The tenant's bank holidays could not be read: the training waited.
`job_lock_busy`	Another run of the same job was running; logged as information.

The collector log is `index=_internal sourcetype=trackme:custom_commands:trackmecriblcollect` (file `trackme_cribl_collect.log`).

REST endpoints

The workspace uses the `/trackme/v2/cribl/tenant/*` endpoints shared with the Cribl tenants (`workspace`, `configure`, `collect`, `collect_result`, `repair`, `policies`, `policy_save`, `findings`, `finding`, `triage`, `finding_groups`, `group_triage`, `group_members`, `notifications`, `notification_test`, the custom policy and AI Agent operations) plus the HEC reads below, which answer `splunk_hec_tenant_required` on another tenant. An operation a HEC tenant does not have answers `not_available_for_splunk_hec`. Every endpoint documents itself with `describe=true` and is listed in the REST API Reference page.

Endpoint	Purpose
`hec_overview`	The summary of the latest window, the pools, the endpoints needing attention, the discovered endpoints and the busiest tokens
`hec_endpoints`	A page of endpoints (up to 200), filters host, pool and status (`up`, `stalled`, `late`, `silent`), sorts by host, events, failures, error rate, busy, queue or refused
`hec_endpoint`	One endpoint, with its pipeline, output and receivers, and its splunkd API facts
`hec_tokens`	A page of token names
`hec_token`	One token by `token_name`: its latest window, its last 24 hours, its endpoints, its errors, the indexes refused and the client addresses
`hec_lookups`, `hec_lookup_preview`, `hec_token_selection_prev` `hec_token_selection_stat`	The Token silence editor: the lookups of a deployment, one lookup's preview, and Test selection (start, then poll)

Example:

```
| trackme url="/services/trackme/v2/cribl/tenant/workspace" mode="get"␣
↪body="{\"tenant_id\":\"example-hec\"}"
```

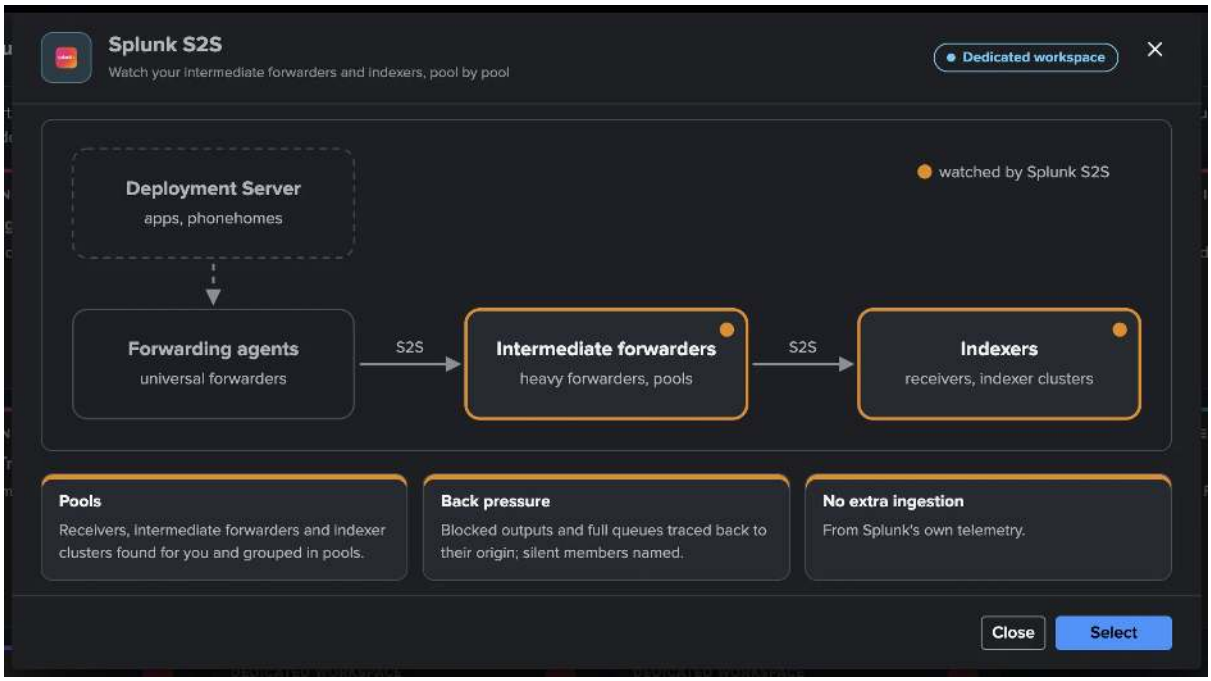
8.3.21 Splunk S2S

The **Splunk S2S tenant** watches the **Splunk-to-Splunk receiving side** of one Splunk deployment: the **receivers** (indexers, standalone instances, Splunk Cloud indexers), the **intermediate forwarders** (heavy or universal forwarders that receive S2S and forward it) and the **pools** they form, and the data path between them. It answers the questions a Splunk administrator asks about forwarding: *is every receiver accepting and keeping up? is my intermediate tier blocked, backed up, silent, or load balancing over fewer destinations than usual? when senders are held back, which tier is the bottleneck, and is the cause downstream, local or an upstream surge? how much flows into each pool, from how many senders of which type and version, with which TLS and acknowledgement posture?* The answers are **findings**, raised by built-in policies, with the same triage, Finding groups, notifications, assignment and findings actions as the other dedicated tenants, **traffic ML models** on every receiver pool, and a findings AI Agent of its own, the Splunk S2S AI Agent.

A Splunk S2S tenant is a **dedicated tenant type**: it has no entities and no state colour, and one tenant follows the S2S receiving side of one Splunk deployment. It reads the telemetry Splunk **already writes** — the receivers' and intermediates' `metrics.log` connection lines, queues and throughput, their `splunkd.log` input and output messages, their health report and `_introspection` — on the local search tier or through a TrackMe remote account. Nothing is installed on any Splunk instance, no new data is ingested, no forwarder is ever called: the tenant only reads.

The **senders** (universal forwarders, heavy forwarders, search heads, Deployment Servers) are context here: counted, typed and versioned per receiver and pool, never findings. The forwarding agents and their Deployment Servers have their own tenant type, *Splunk UF Fleet*.

The Splunk S2S tenant is generally available since TrackMe 2.4.22.



For a guided tour on a live deployment, see the hands-on walkthrough *Splunk S2S*. Every policy, setting, job and command is described in the *Splunk S2S reference*.

What it monitors

Object	What TrackMe watches
Receivers	Every host that receives S2S and indexes it. Whether it **accepts** its senders (a receiver whose queues stayed blocked stops its listening ports), whether it is **silent** while the other members of its pool report, **back pressure** along its ingestion path and the stage that cannot keep up, its share of the pool's senders (**imbalance**), **connection churn**, and the **TLS and acknowledgement posture** and **Splunk versions** of its senders.
Intermediate forwarders	Every host that receives S2S and forwards it: a heavy forwarder tier, or a universal forwarder used as a relay. Its **output paused** (the senders behind it held back), **back pressure** and its origin (downstream, local or an upstream surge), **silence**, failing destinations, a **single destination** where it usually load balances over several, what it receives against what it forwards (**in / out anomaly**), the TLS posture of its senders, and a universal forwarder **relaying** universal forwarders.
Pools	The receivers of an indexer cluster, one pool per site of a multisite cluster (found without configuration), the intermediate forwarders grouped by host-name family (also automatic), or receivers and intermediates grouped by host patterns, output groups or hosts you assign. **Capacity** (members up against the expected size), **not receiving** (every member silent), a **traffic drop** against the same hour of the previous days, **traffic outliers** against a traffic ML model, and a **back pressure cascade**: one finding naming the receivers that back up and the intermediates paused towards them.
Destinations	A receiver that **many senders fail to reach** without a break: one finding on the destination listing the senders, never one per sender.

Indexer clusters are found without configuration. Each indexer cluster the source search head is attached to becomes a receiver pool, its members and expected size from the cluster. A **multisite** cluster forms **one pool per site**, named `<cluster>:<site>` (`idxc1:site1`, `idxc1:site2`): each site is judged on its own peers, so a site that loses its indexers is a capacity finding even while the other site carries the load. A single-site cluster, or one whose peers' sites are not all known, stays one cluster-wide pool. An optional remote account on the cluster manager adds its view of **planned work**: while a cluster is in maintenance mode or a rolling restart, the availability, capacity, balance, churn and traffic findings of its pools wait. A cluster's own health (peers, replication and search factors, buckets) is shown in the workspace, never judged here.

The intermediate tier is grouped for you too. The intermediate forwarders no pool claims form one **automatic pool per host-name family** (`hec-pool-hf1` to `hec-pool-hf4` form `hec-pool-hf`). **Configure** an automatic pool to rename it, give it its output groups and an expected size: it becomes a configured pool and keeps its id, its findings and its history.

How it works

Roles are read from signatures, never configured. Over the last 24 hours, a host that receives S2S (`group=tcpin_connections` lines) and indexes is a **receiver**; a host that receives and forwards (`group=tcpout_connections` or cooked output) is an **intermediate forwarder**. An intermediate's type comes from how its own receivers see it: `fwdType=full` is a heavy forwarder, `uf` a universal forwarder. Host patterns and excluded hosts bound which hosts may be receivers or intermediates; they never apply to senders.

Bounded reads, whatever the deployment size. Every read is an aggregation over indexed terms or metric dimensions (`tstats` with `TERM()` and `PREFIX()`, `mstats` on the `_metrics` mirror), never a search per host. The live job reads index-time windows of five minutes with catch-up (never the same window twice), and every view of a window is published under one id, so two views never mix two windows.

Job (Collection health label)	What it does
Inventory (every 10 minutes; the directory is rebuilt about hourly, always on a manual run)	The roles over a day, the intermediates' type, the indexer clusters (the search head's view every run, the manager's through its account), the pools, and a probe that chooses the telemetry form. Publishes the directory.
Live (every 5 minutes)	Per receiver and intermediate: the inbound connections (senders by type and version, TLS and acknowledgement mix, volume), an intermediate's output groups and destinations, output pauses and connection failures; assembles the latest facts of the jobs below. Per pool: the sums, the members up and the balance. Writes the series to the tenant's metric index, then **evaluates every policy**. A live run never builds the directory: before the first inventory it waits for it.
Pipeline, **Health report**, **Destinations** (every 5 minutes each)	Dedicated jobs, each with its own report, schedule, time budget and Collection health line (2.4.23): the pipeline queues and their threads, stopped listening ports, the intermediates' output trouble, churn, CPU and memory; the health report, with the detail for the flagged hosts only; the destinations the senders fail to reach and the cluster peers restarting. A job that misses a slot makes the policies hold, never count twice.
Traffic ML models (twice an hour)	Learns each receiver pool's models when due (see *Traffic ML models*).
Policies (on demand)	Evaluates the policies again on the latest live window (**Run now** in Collection health).

Host statuses. In each window a host is **not accepting** (it stopped its listening ports), **silent** (no connection line in the live window), **output paused** (an intermediate), **back pressure**, or **up** — the first that applies.

Why senders are held back. A receiver writes every S2S connection into its input queue, then into the ingestion pipeline (`splunktcpin` → `parsingqueue` → `aggqueue` → `typingqueue` → `rulesetqueue` for parsed data → `indexqueue`); a heavy intermediate adds its output queues (`tcpout_<group>`). When a stage cannot keep up, its queue fills, then the queues before it, until the receiver pauses its senders or stops its listening ports. The tenant reads every queue of that path, names the **bottleneck** — the stage after the furthest-downstream saturated queue — and whether that stage's thread was busy (working at its limit) or waiting on the next stage. On an intermediate it names the **origin**: *downstream* (its output paused, or a receiver pool it feeds backs up), *upstream surge* (inbound above 1.5 times its learned volume) or *local*.

Note

An intermediate's own `tcpout` count is **compressed bytes on the wire**, not comparable with what its receivers count (uncompressed). The workspace labels it **Sent on the wire (compressed)** and shows the receivers' side as the relayed volume: never compare the two.

Before you start

A licence. A valid **Enterprise**, **Unlimited**, Unlimited trial or **Developer** licence. The Splunk S2S tenant is not available with the Foundation edition or the Foundation trial.

A source. The search tier that sees the receivers' and intermediates' internal indexes:

- **local**, this search tier, when it searches the indexers that hold their `_internal` (and `_metrics` / `_introspection` when used);
- or a **TrackMe remote account** (*Remote Splunk deployments*) to the search tier of another deployment. The tenant's scheduled searches run as the tenant owner, so the **tenant owner must be allowed to use every remote account the tenant reads**: the wizard and the Settings check it before saving and name the fix.

The three index names are configurable (defaults `_internal`, `_metrics`, `_introspection`).

Reading the indexer clusters. The cluster reads (`/services/cluster/searchhead/generation` and each peer's `/services/cluster/peer/info`, through `| rest`) need the **read-only** `list_indexer_cluster` capability for the reader: the tenant owner on the local search tier, or the source account's user. Without it Splunk

answers nothing, and Collection health reports `cluster_read_refused`. A deployment with no indexer cluster can set **Indexer clusters as pools** to Off.

Optionally, a manager account per indexer cluster. A TrackMe remote account on the cluster manager's management port lets the tenant read the manager's maintenance mode and rolling restarts, which **Hold during planned work** then follows. Give it a **read-only** role:

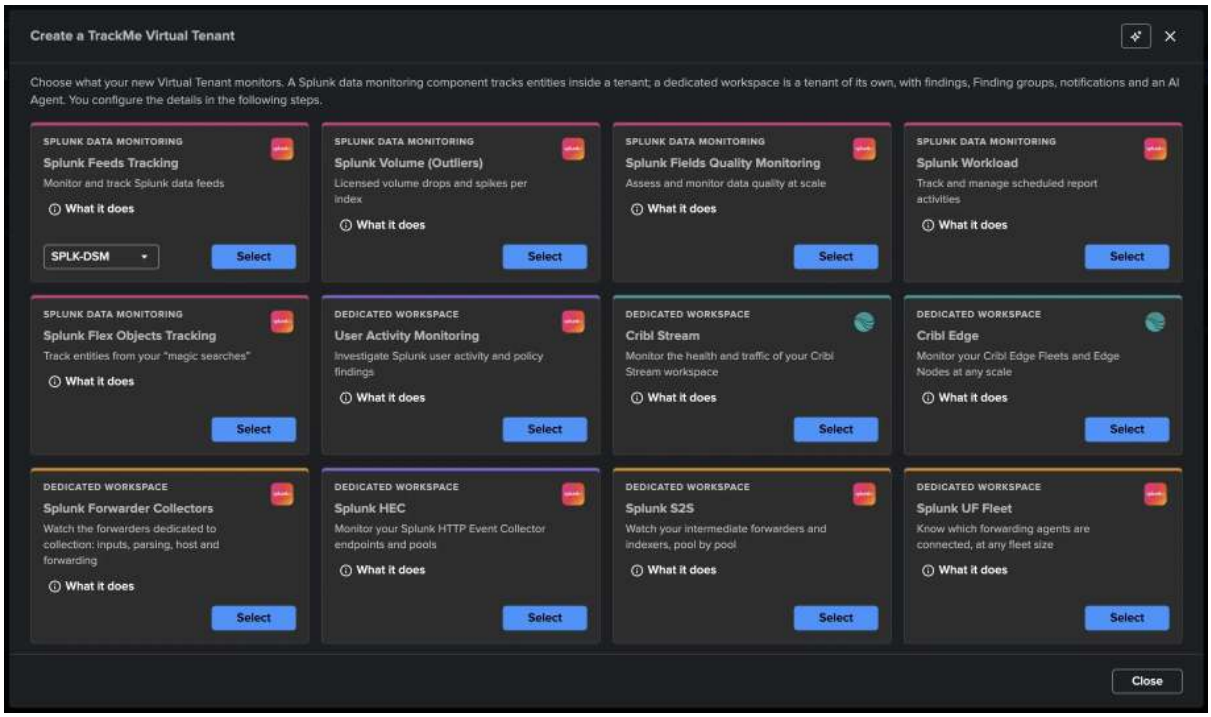
```
[role_trackme_cluster_reader]
importRoles =
search = enabled
rest_properties_get = enabled
list_indexer_cluster = enabled
```

The account is checked as soon as it is chosen, and an account that could change the cluster gets an advisory: TrackMe only reads, a read-only role is recommended.

A tenant owner: `nobody`, or an account (typically a least-privilege service account) holding the capabilities the wizard lists (`trackmeuseroperations`, `trackmepoweroperations`, `trackmeadminoperations`, `schedule_search`, `list_settings`, `list_storage_passwords`, `admin_all_objects`). The scheduled triage of the Splunk S2S AI Agent needs an owner that is not `nobody` (see *The Splunk S2S AI Agent*).

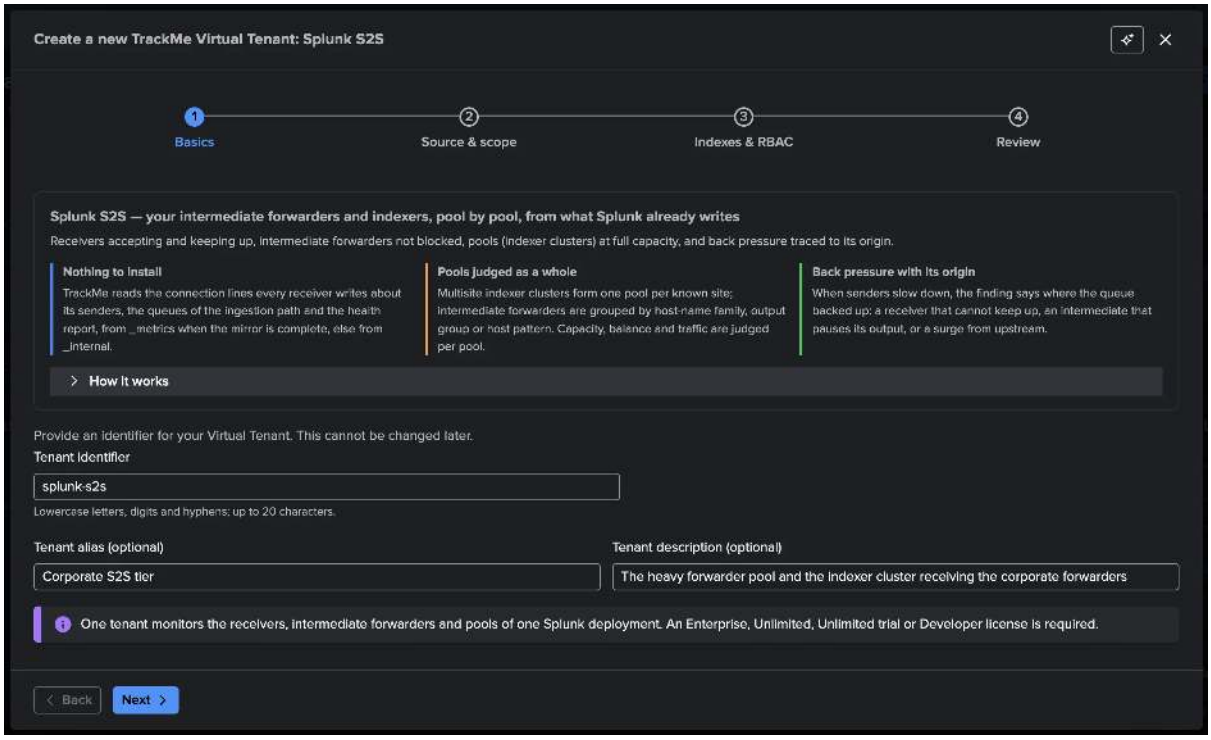
Creating the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Splunk S2S** card ("Watch your intermediate forwarders and indexers, pool by pool"). **What it does** opens the dialog shown at the top of this page.

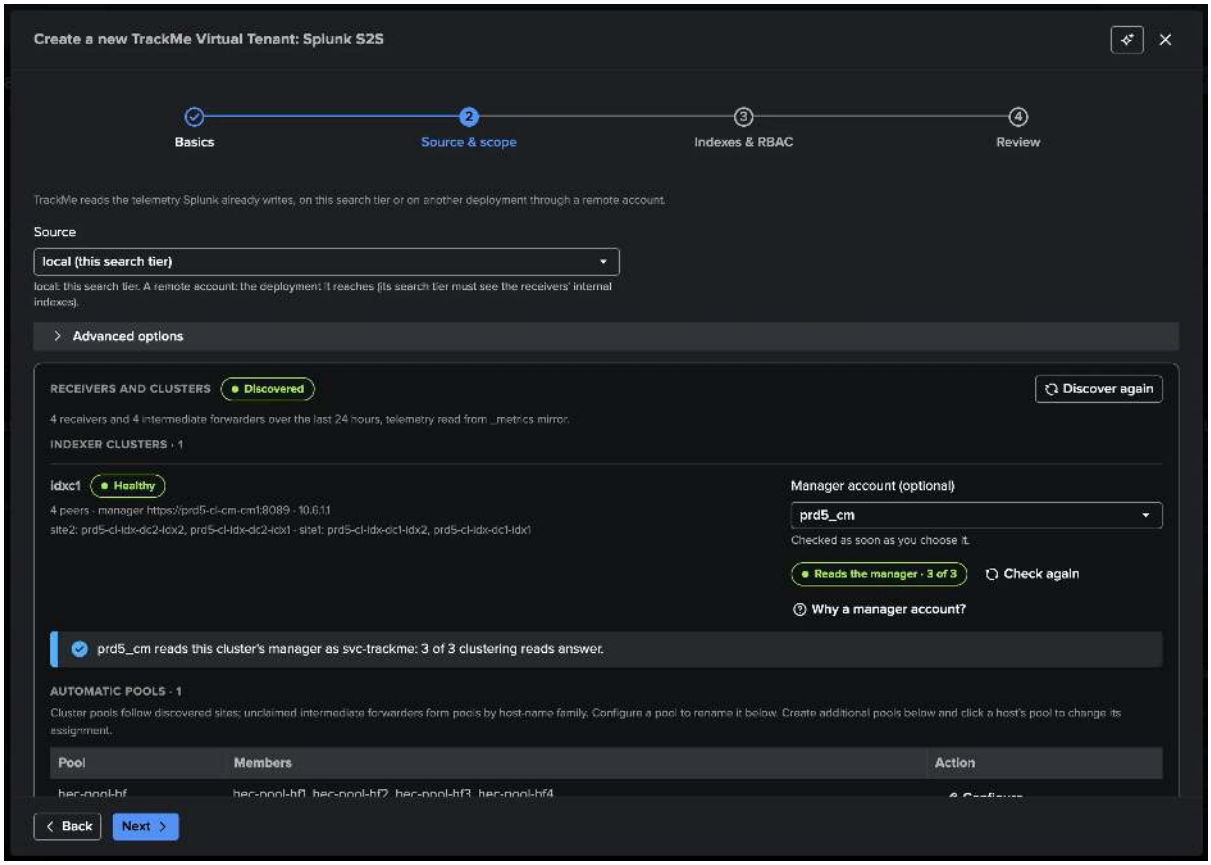


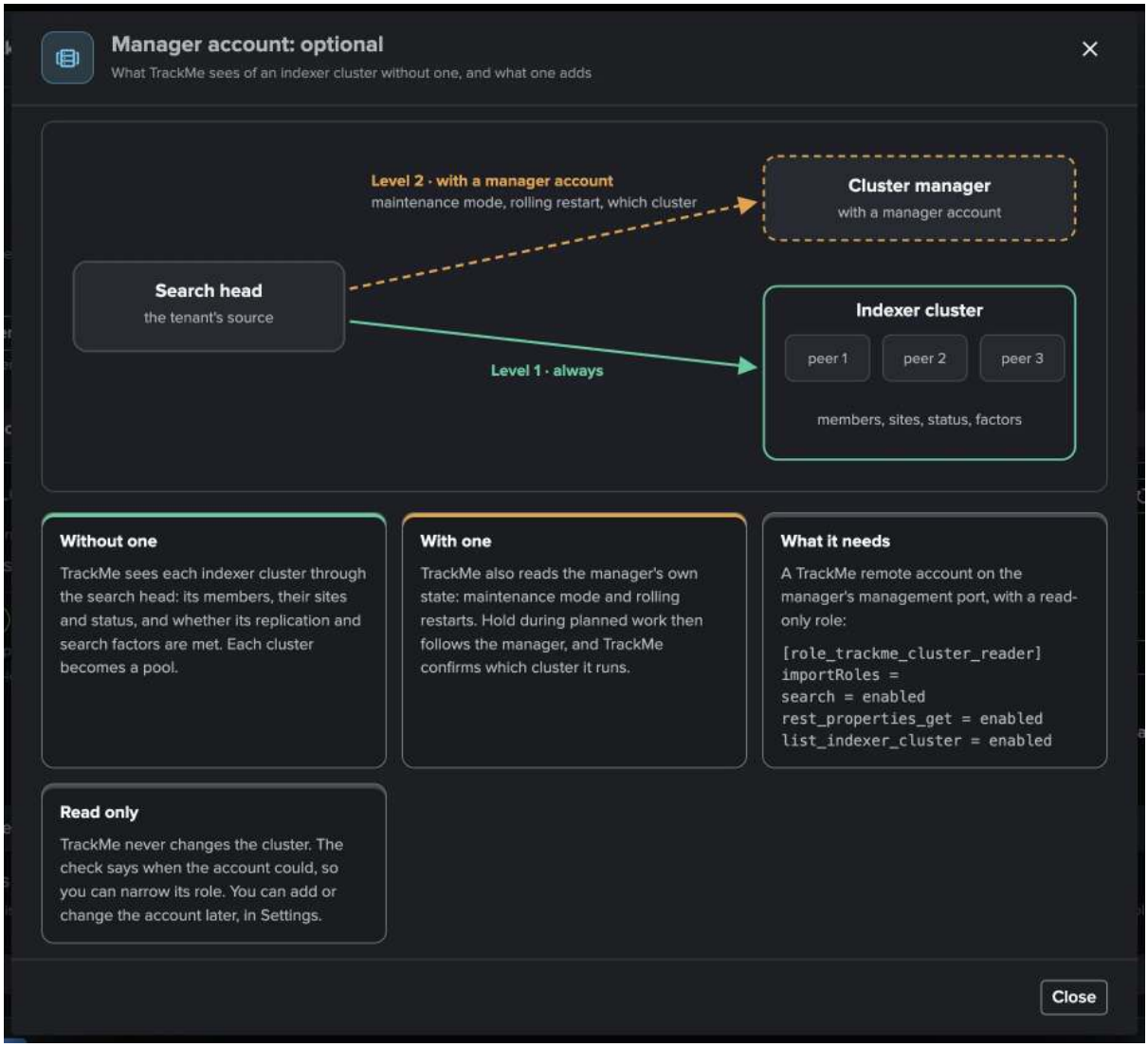
The wizard has four steps.

1. Basics — an introduction to what a receiver, an intermediate and a pool are, then the tenant identifier (lowercase letters, digits and hyphens, up to 20 characters, fixed once created), an alias and a description. One tenant monitors the receivers, intermediate forwarders and pools of one Splunk deployment.

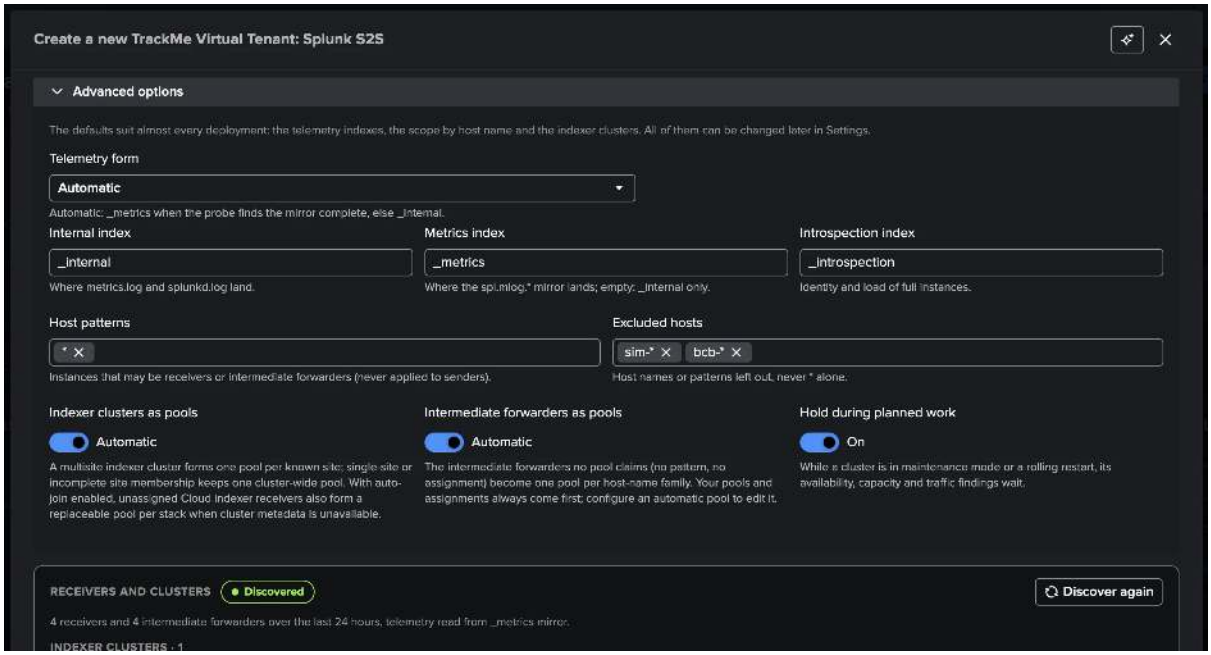


2. Source & scope — the **Source** (this search tier, or a remote account), then **Receivers and clusters**, a short discovery of the receivers and intermediate forwarders found over the last 15 minutes (at most two searches of 5 minutes: one row per receiver with its role, its senders counted on the indexers and its inbound volume, then the indexer clusters): each indexer cluster with its sites, its health and its peers, and beside it the cluster's optional **Manager account**. The account is checked as soon as you choose it ("Reads the manager · 3 of 3"); each refused read says what TrackMe cannot read, what it costs and the role to create, and an account that could change the cluster gets an advisory recommending a read-only role. **Why a manager account?** explains the two levels.

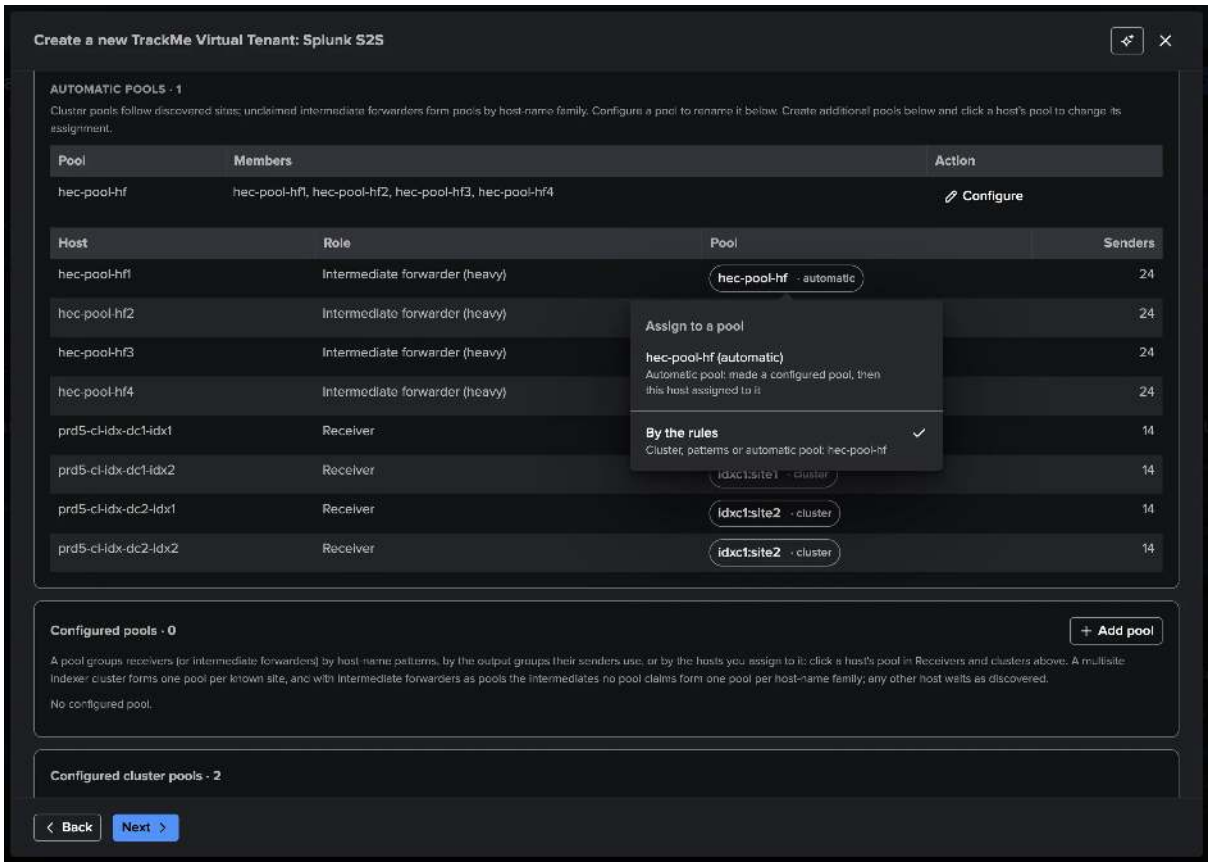




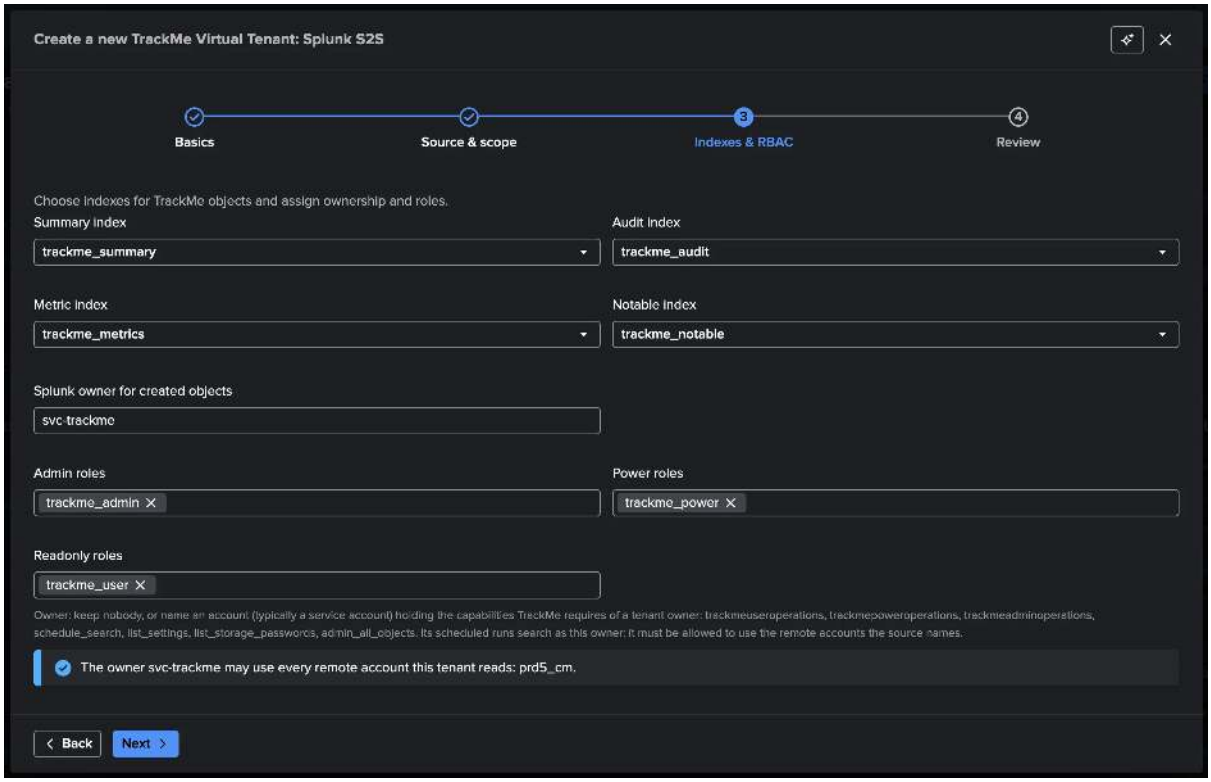
Advanced options (folded; opened when one of its settings needs a fix) hold the telemetry form (*Automatic*: the `_metrics` mirror when the probe finds it complete, else `_internal`; or forced), the three index names, **Host patterns** and **Excluded hosts** (type a host name or a pattern and press Enter: each becomes a token; a pasted comma-separated list is split), **Indexer clusters as pools** and **Intermediate forwarders as pools** (Automatic / Off) and **Hold during planned work** (On / Off). The defaults suit almost every deployment. Set an exclusion here rather than later: the first backfill (14 days) never reads the hosts it leaves out. Below, two families of test receivers sharing the same indexes are left out: `sim-*` and `bcb-*`.



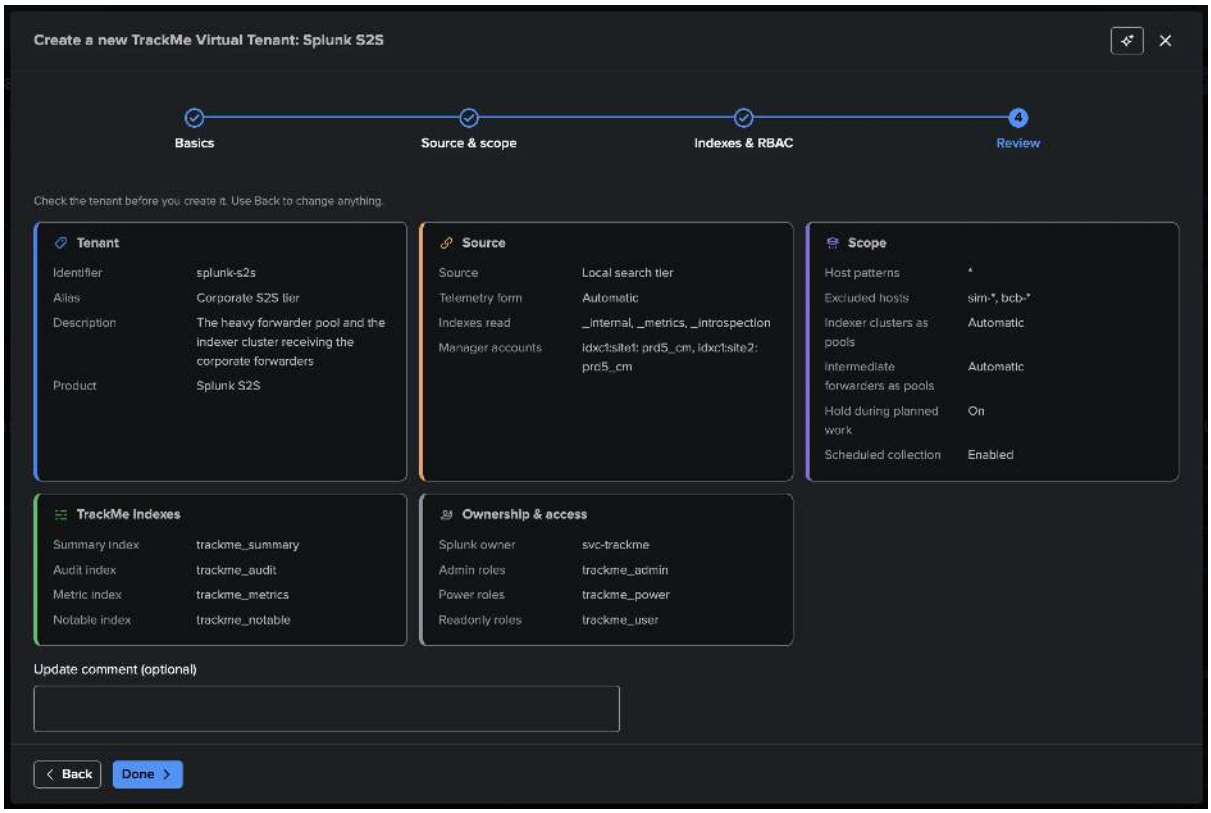
Below the clusters, the step shows the pools the tenant will start with, the same editor as **Settings → Pools**: the **automatic pools** (each site of a multisite cluster, and one pool per host-name family of the intermediates no pool claims, each with **Configure**), every host with its **pool pill**, and the **configured pools**. Click a host's pool pill to **assign it to a pool**, or choose **By the rules** to give it back to the cluster, the patterns or its automatic pool. Nothing needs configuring here: the indexer cluster and the heavy forwarder tier below are pools from the first run.

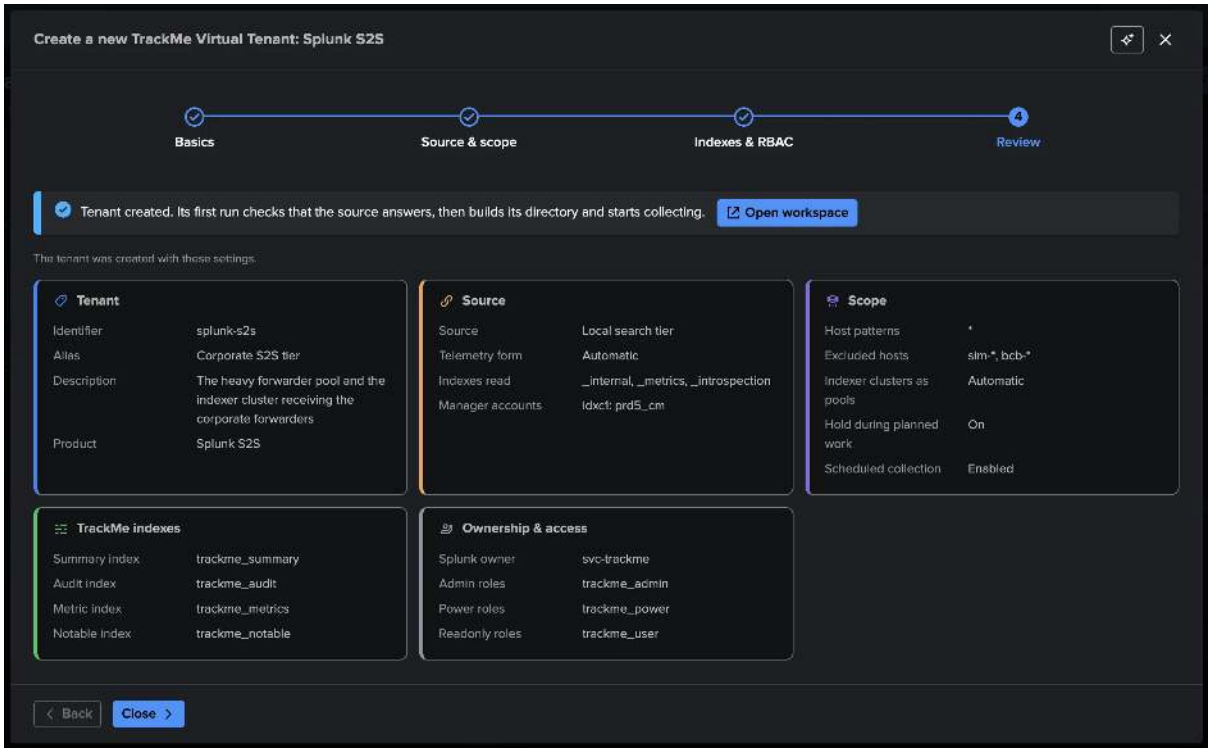


3. Indexes & RBAC — the tenant's TrackMe indexes (summary, audit, metric and notable), the owner and the admin, power and read-only roles. The wizard checks that the owner may use every remote account the tenant reads, the manager accounts included.



4. Review — the tenant, its source with the manager account of each site pool, its scope and its pool settings, then **Done**. The first live run starts at once: it checks that the source answers, then builds the directory and starts collecting. **Open workspace** opens the tenant.

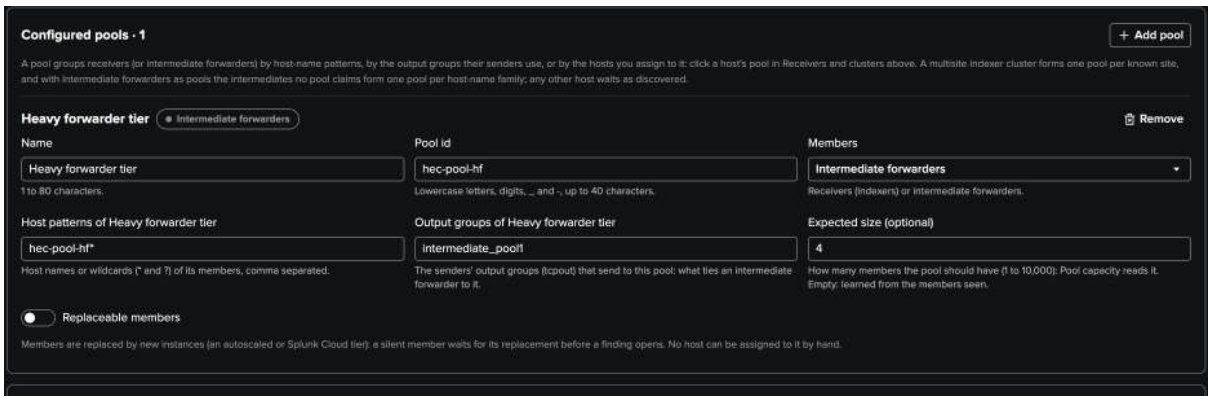




Configuring pools

An indexer cluster (one pool per site of a multisite cluster) and the intermediate tier (one automatic pool per host-name family) are pools from the first run. **Settings → Pools** shows the same editor as the wizard: the **automatic pools**, every host with its **pool pill**, and the **configured pools** (up to 200 pools in all). Make a pool your own when it needs a name, its output groups or an expected size; group receivers no cluster claims (standalone indexers, Splunk Cloud indexers) in a configured pool of their own.

Configure on an automatic pool turns it into a configured pool with the **same id**, so its findings and history continue; on the lab tenant, the automatic pool `hec-pool-hf` became the **Heavy forwarder tier**. A configured pool has a name, an id (fixed once saved: its findings and history follow it), its **members** (receivers or intermediate forwarders), **host patterns**, the **output groups** that send to it, an optional **expected size**, and **Replaceable members** for a tier whose members are replaced by new instances (an autoscaled tier, Splunk Cloud indexers): a silent member then waits for its replacement before a finding opens. A site pool of an indexer cluster can be renamed too (its members always come from the cluster); removing a configured pool gives its hosts back to the automatic pools.



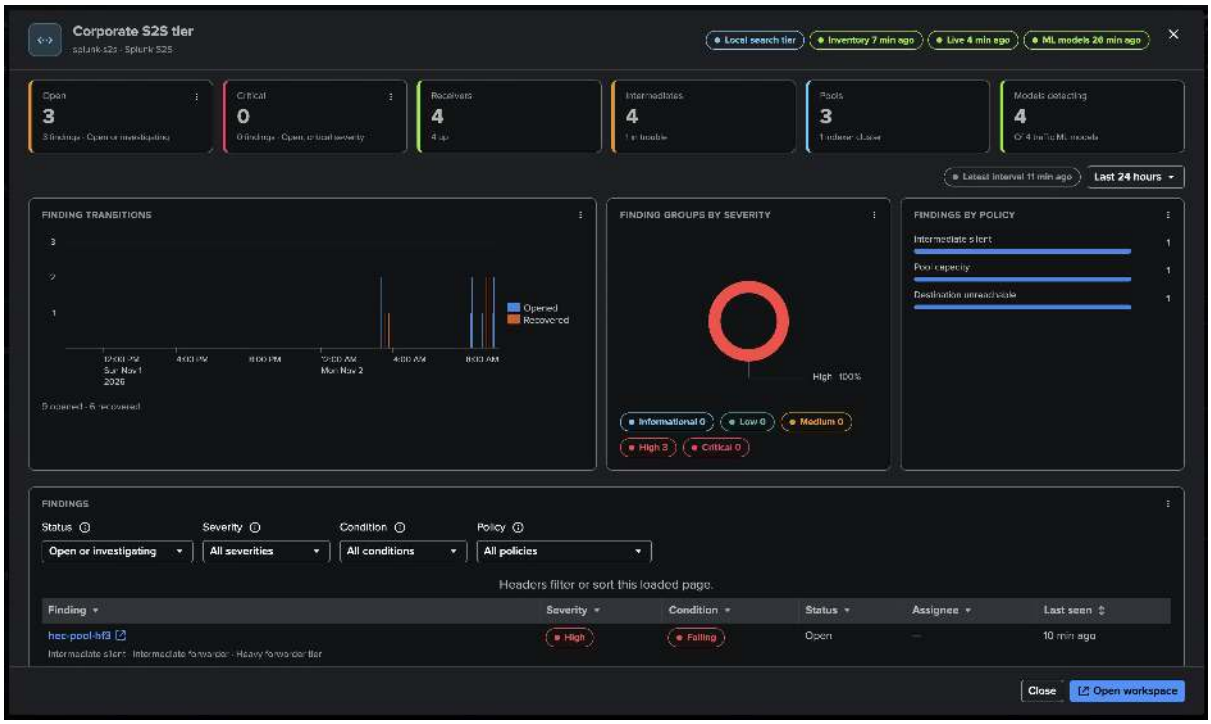
Output groups tie an intermediate tier to what it feeds. The output groups are the senders' `tcpout` group names that send to the pool (here the forwarders' `intermediate_pool1`). They are how an intermediate is tied to the receiver pool it feeds, and how a sender's traffic is placed on the data path; an intermediate is never tied by guessing from a destination address.

A host is placed by the first rule that applies: a host **assigned by hand** (from its pool pill), then the **indexer cluster**

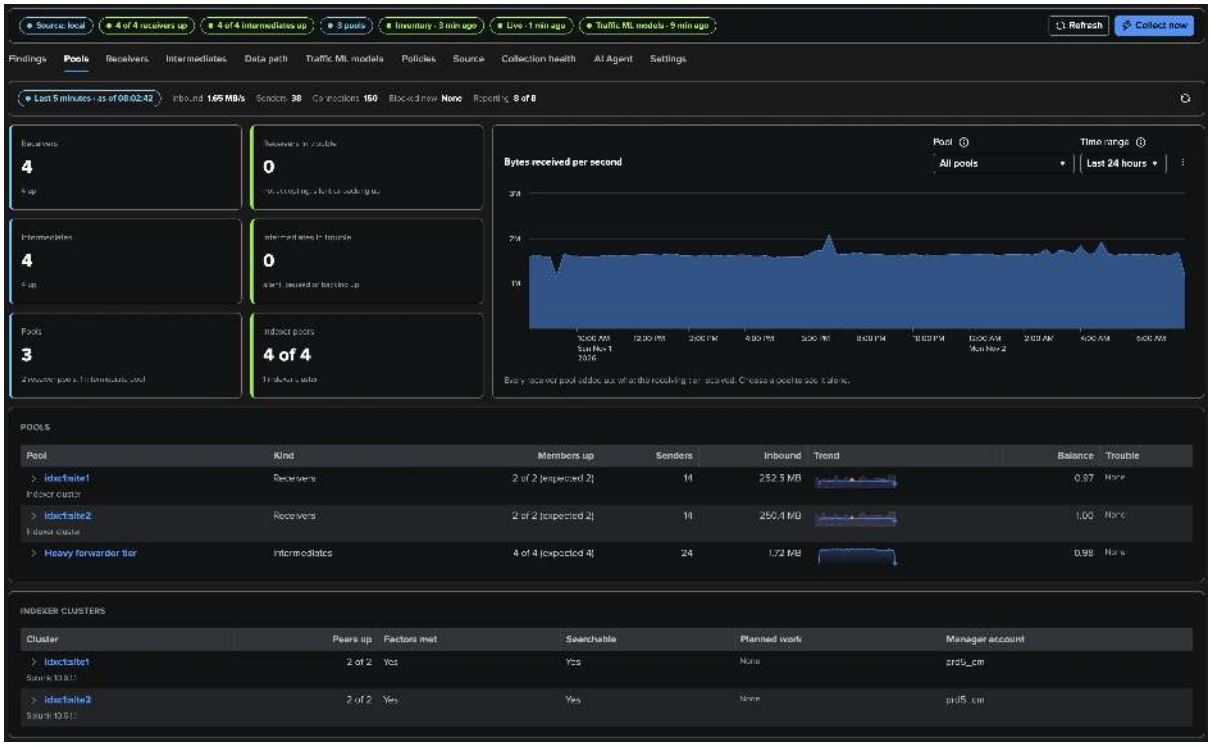
it is a peer of (its site's pool on a multisite cluster), then a configured pool whose patterns match (when **Auto-join** is on), then the **automatic pool** of its host-name family (an intermediate), else **discovered** (no pool). An intermediate never joins a cluster pool, and your pools and assignments always come before the automatic ones. A pool may hold hand-picked hosts only, with no pattern. A pool change applies from the next inventory; to see it at once, **Run now** the inventory in Collection health, then **Collect now**.

The workspace

The preview. On the *Virtual Tenants* page, a double-click on the tenant card opens its preview before the workspace: the source and the collection pills, the **Open** and **Critical** figures, **Receivers** and **Intermediates** (with how many are in trouble), **Pools** and **Models detecting**, the finding transitions and severities of the last 24 hours, the findings by policy and the findings table, read-only. A finding opens in the workspace; **Open workspace** opens the tenant. Before the first live run, the figures read as dashes.

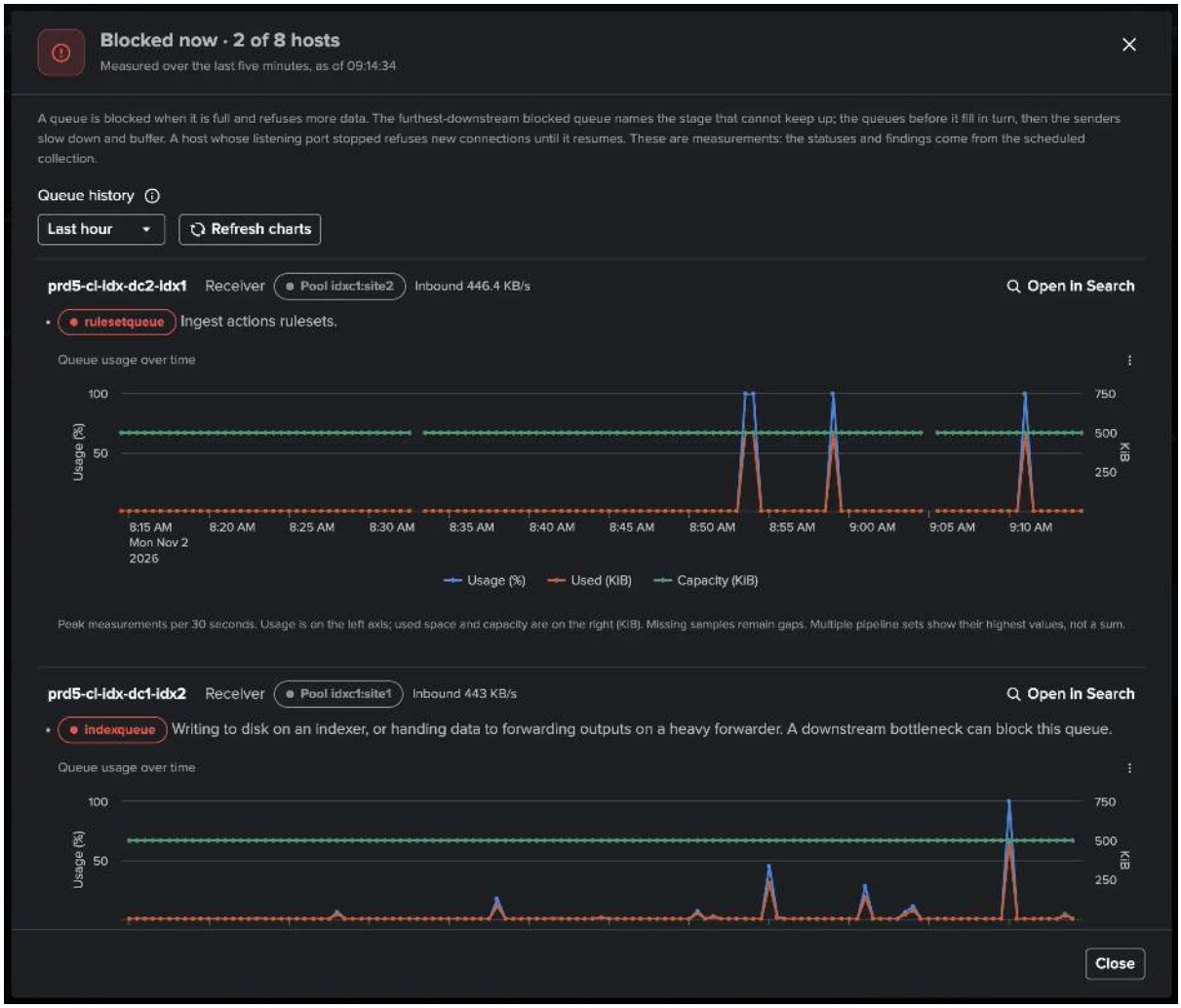


The workspace is the Tenant Home of the tenant. A **status strip** sits above the tabs: the source, "N of M receivers up", "N of M intermediates up", the pools, and the **Inventory**, **Live** and **Models** job pills (each opens Collection health), with **Refresh** and **Collect now** (runs the live job).

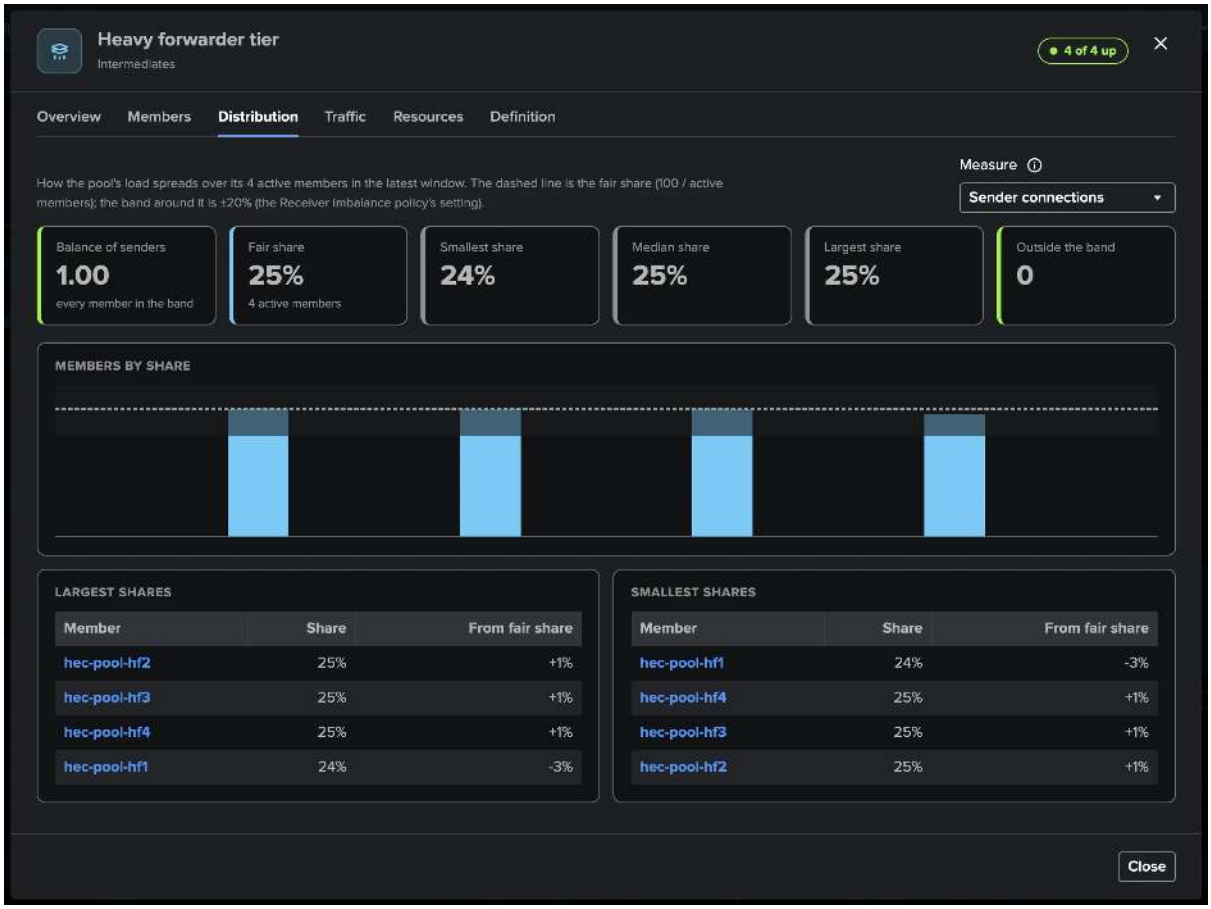


Tab	What it shows
Findings	The findings centre shared by every dedicated tenant (see *Findings, Finding groups and findings actions*). Families: Receivers, Intermediate forwarders, Pools, Destinations. The findings of one object (a host, a pool, a destination) form one Finding group; the pool is the policy scope (`discovered` for a host in no pool).
Pools	The **Last 5 minutes** strip (inbound per second, senders, connections, **Blocked now**, hosts reporting), host tiles beside **Bytes received per second** (every receiver pool added up, or one pool alone, over the chosen time range), the pools table (each row expands to its members, an indexer cluster pool to its peers), the indexer clusters, one row per site pool (peers up, factors met, searchable, planned work, manager account), the Splunk versions of the hosts, the intermediate types, and the hosts needing attention.
Receivers, **Intermediates**	Every host, paged on the server, with filters on status, pool and name. **Signal pills** (back pressure and its bottleneck, destinations paused, origin, red or yellow health, without TLS) open the facts and their evidence. A host opens its modal.
Data path	The generated three-column topology: sender types → intermediates → receivers, a pool as one node, edge width by volume, with a Flows table.
Traffic models	Every receiver pool's models, their state, the model view, tuning and relearning.
Policies	The policy catalogue, custom policies, and **How it is evaluated** for every policy.
Source	The search tier, the qualification, the telemetry form, and the collection runs with their outcome and receipt.
Collection health, **AI Agent**, **Settings**	See below.

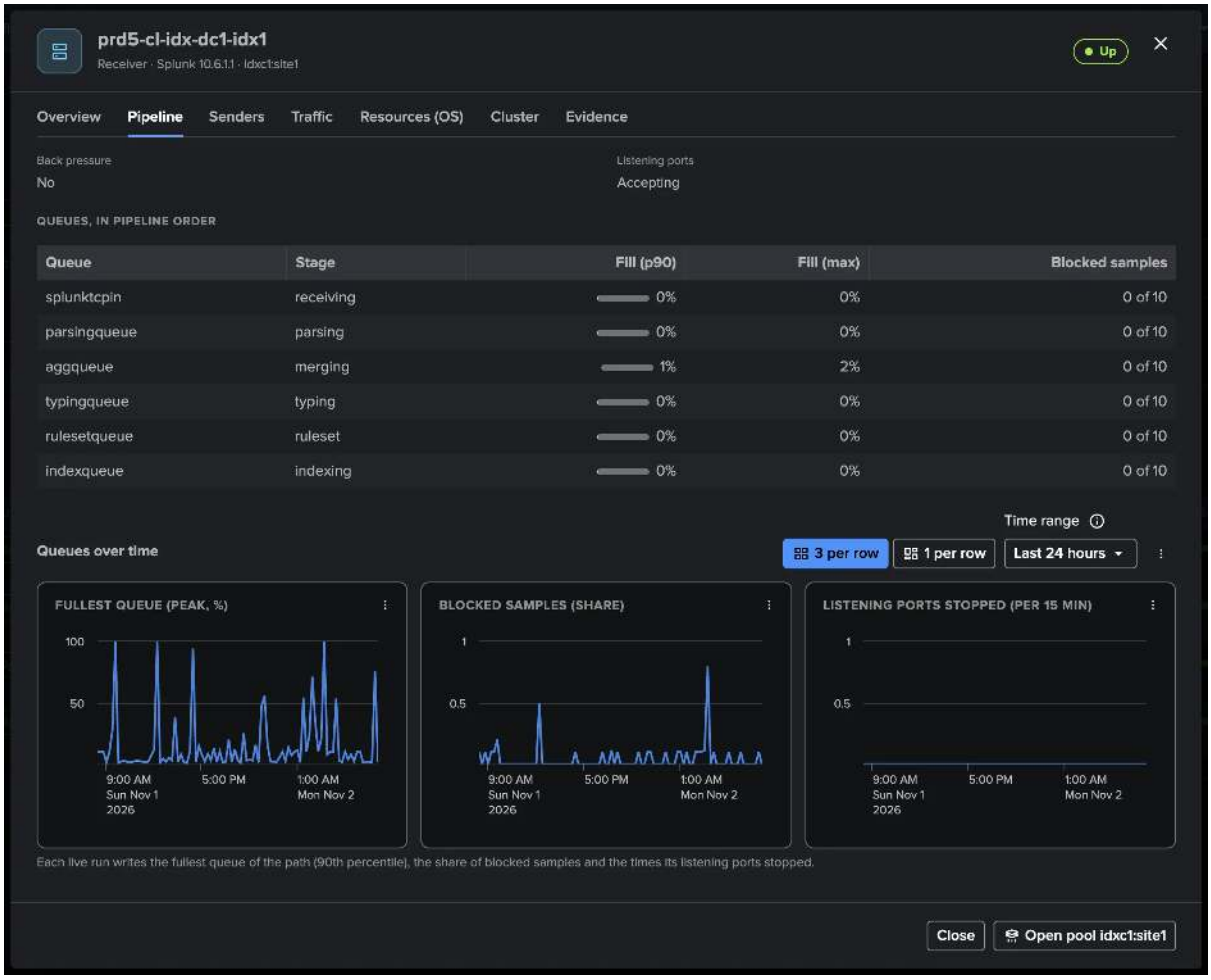
Blocked now. When a host of the last five minutes has a blocked queue, the strip's **Blocked now · N of M hosts** opens what is blocked: each host with its role, its pool and its inbound, every blocked queue with what that stage of the pipeline does, a stopped listening port, **Open in Search** on the host's queues, and the **queue history** of each blocked queue (usage, used and capacity, last 15 minutes to 24 hours, two hosts per page). These are measurements: the host statuses and the findings come from the scheduled collection, so a queue blocked for a few samples shows here without any finding opening.



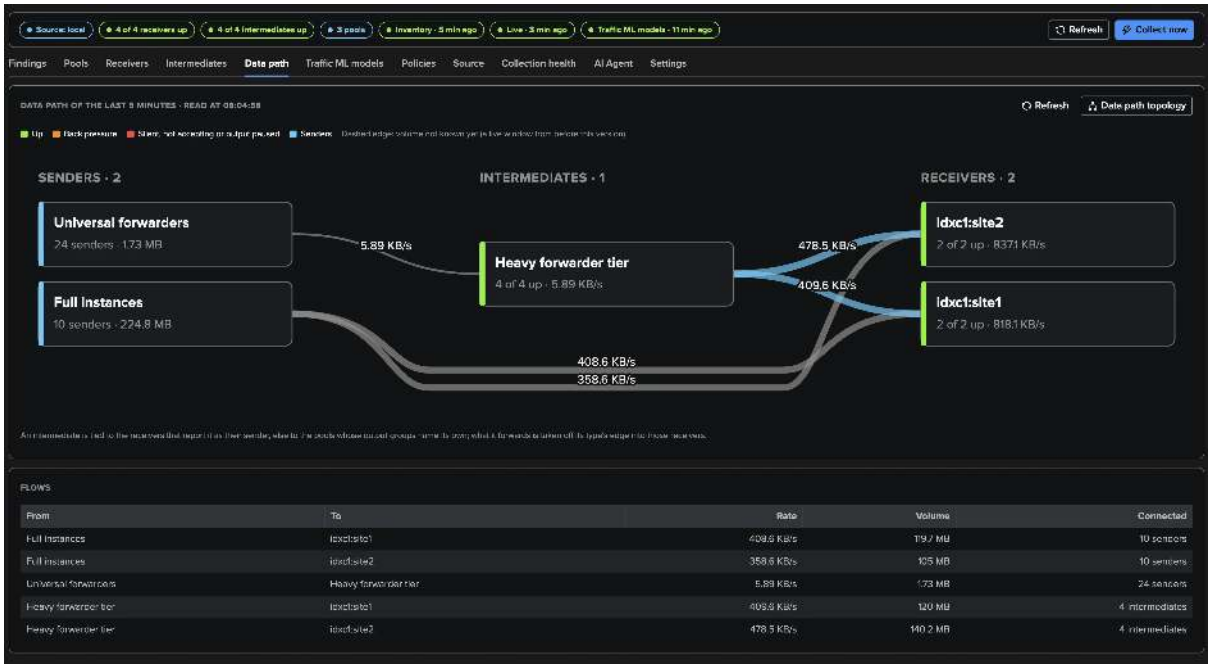
The pool modal has tabs Overview, Members, **Distribution** (one bar per active member's share of the pool, the fair-share line and the imbalance band of the Receiver imbalance setting; switch between sender connections and volume; Jain's fairness index), Traffic (with the traffic model's band on a receiver pool), Cluster (an indexer cluster pool), Resources (the members' CPU and memory) and Definition.



The host modal has tabs Overview (status, senders, inbound, worst queue, the last 5 minutes, and what needs attention with its open findings), **Pipeline** (the queues in pipeline order with fill bars and over time, the listening ports, an intermediate's output groups and destinations), Senders (posture tiles, senders by version, the senders table), Traffic, Resources, Cluster (an indexer peer) and Evidence.



The Data path tab draws the forwarding path of the last five minutes from what the receivers report: the sender types, the intermediates and the receivers, edges sized by volume, and a Flows table that states every edge in words. An intermediate is tied to its receivers by its output group, never by guessing from a destination address; **Data path topology** saves it to Topology Studio.



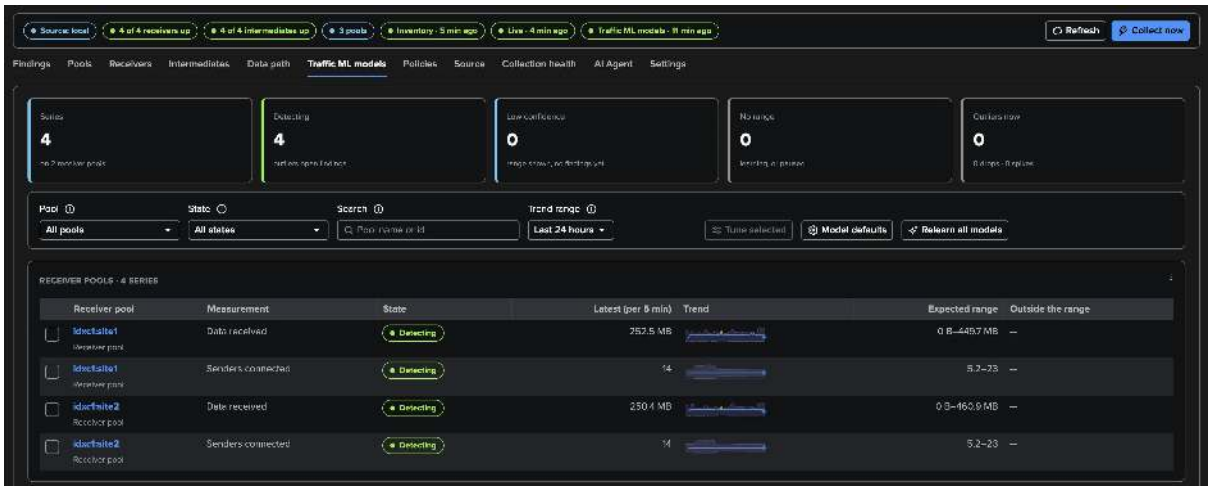
What the figures mean

Figure	Meaning
Inbound	The KB of the host's (or pool members') connection lines in the live window, by index time.
Senders	Distinct sender GUIDs, exact: a sender load balancing over N members counts once for the pool, never as a sum of per-member counts.
Connections	Sender connections: a sender connected to N receivers counts N times.
Worst queue	The highest 90th-percentile fill of the host's path queues (the output queue excluded).
Back pressure	A queue saturated: 90th-percentile fill of 80 % or more, or blocked in 30 % or more of its samples.
Balance	Jain's fairness index of the volume of every member up (1 is an even split).
Expected size	A cluster's registered peers (a site pool: the peers of its site), else the configured size, else the second-highest daily peak of members up over 7 days.

Traffic ML models

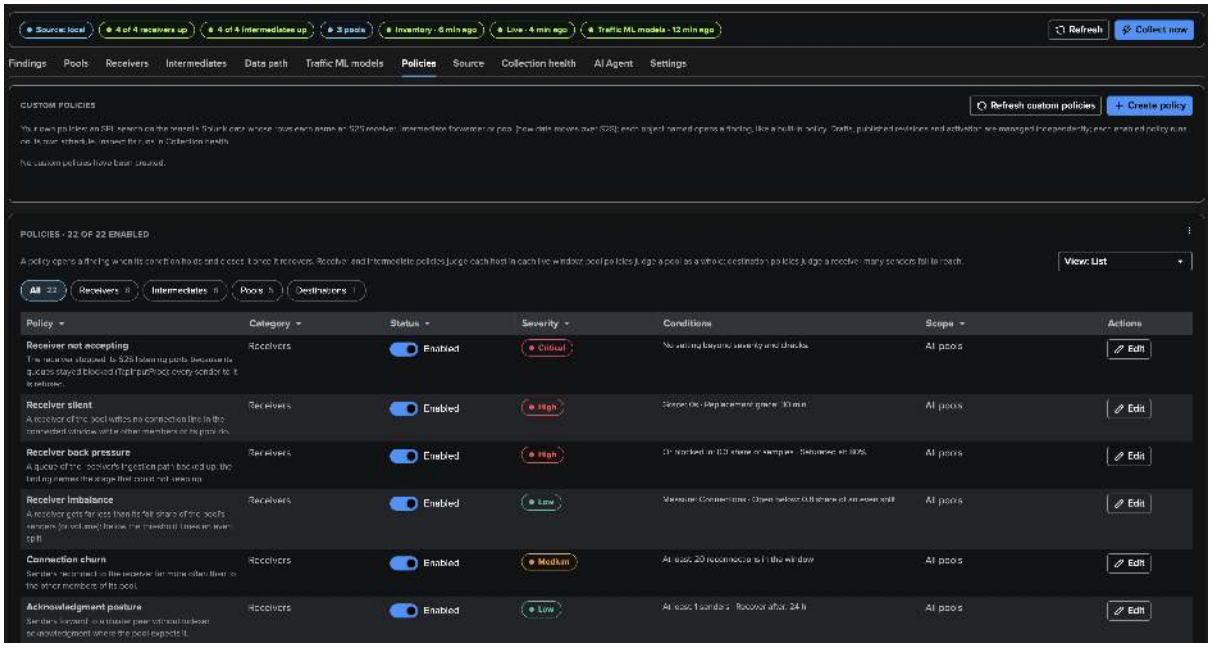
Every receiver pool (each site pool of a multisite cluster) has two traffic models: **KB received** (per five minutes) and **connected senders** (the distinct senders of each member, summed over the members). They are learned per weekday and hour from the members' connection lines over the tenant's model learning history (14 days by default; 0, 7, 14, 30, 60 or 90 days), refreshed every half hour. A five-minute bucket in which a member reported and the pool received nothing counts as zero.

The **Traffic models** tab lists every model with its state, opens the model view (the expected range against the observed traffic), **Tune model** with a simulation of past windows, **Not an anomaly** on a finding, and, for TrackMe administrators, **Model defaults** and **Relearn all models**. These are the traffic model features of the Splunk HEC and Cribl Stream tenants: see *Traffic ML models and Expected traffic*. The **Pool traffic outlier** policy uses them; **Pool traffic drop** is a plain statistic (the same hour's median over 7 days) and needs no model. Intermediate pools, discovered hosts and empty pools are not modelled. Bank holidays are not learned.



Policies

Twenty-two built-in policies cover the receivers, the intermediate forwarders, the pools and the destinations; every policy is evaluated. Each policy has its severity, the runs it needs to open and to recover, a scope (**Pools**: empty means every pool and the hosts no pool claims), a bank holiday hold, and a **How it is evaluated** tab that shows the rule as configured, the data it reads with ready-to-run searches on the tenant's own source, and how to investigate and resolve.



The most useful policies to know first:

- **Receiver not accepting** (critical) — the receiver stopped its listening ports because its queues stayed blocked: every sender to it is refused.
- **Intermediate output blocked** (critical) — an intermediate paused its output: every sender behind it is held back. The finding counts the senders held and names the destinations.
- **Back pressure cascade** (critical) — in one window, a receiver pool backs up while intermediates are paused towards it: one finding names the origin and the intermediates held.
- **Pool capacity** (high, critical when no member is up) — fewer members up than expected.
- **Receiver back pressure** and **Intermediate back pressure** (high) — a queue of the path is saturated; the finding names the bottleneck, and on an intermediate whether the cause is downstream, local or an upstream surge.
- **Destination unreachable** (high) — many senders fail to reach one receiver for 15 minutes: one finding on the receiver, the senders listed in its evidence.

Every policy, its defaults and its settings are in *Built-in policies*.

Planned work is not an incident. With **Hold during planned work** on (the default), Receiver silent, Pool capacity, Pool not receiving, Pool traffic drop, Receiver imbalance, Connection churn and Pool traffic outlier hold on a cluster pool in maintenance mode or a rolling restart: an open finding stays open, a new one waits until the planned work ends.

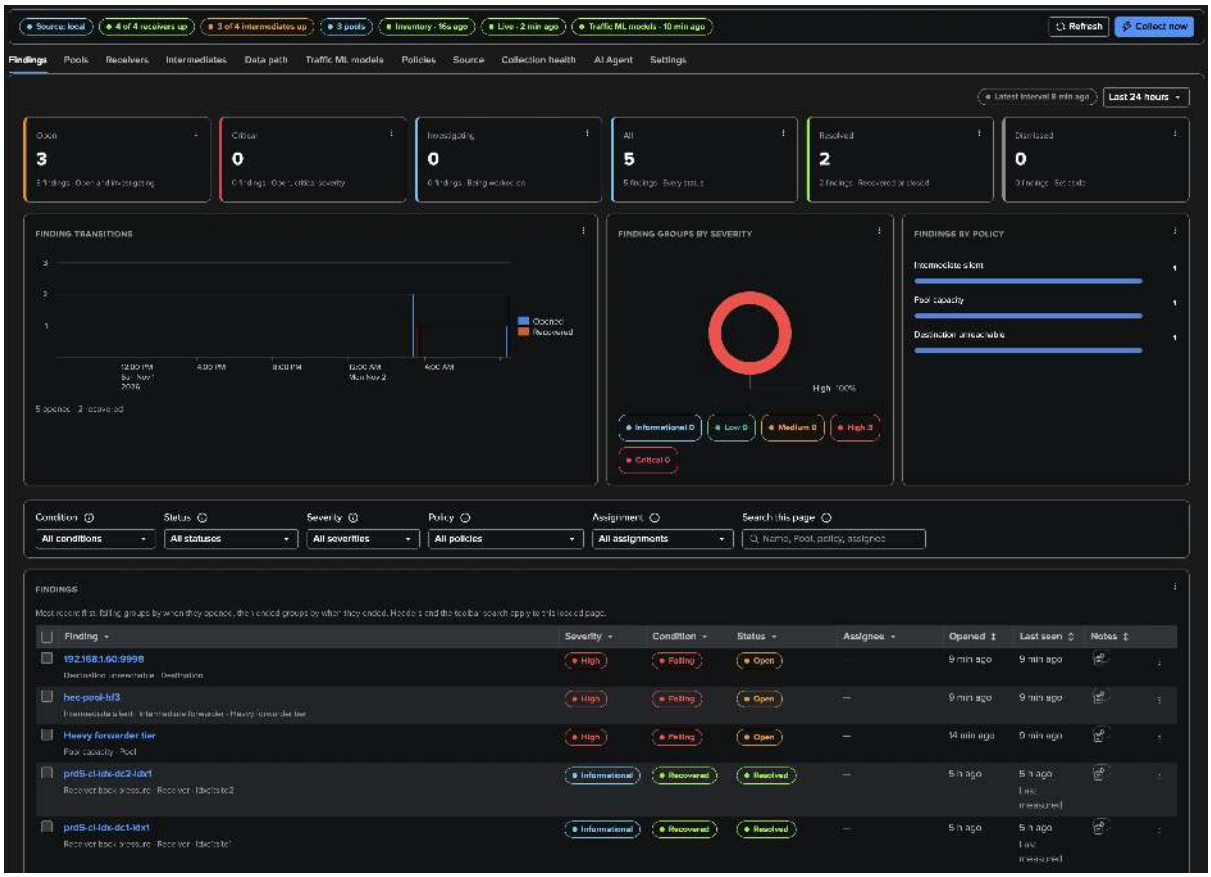
Custom policies. Above the catalogue, TrackMe administrators of the tenant write their own SPL: each result row names a receiver (`receiver`), an intermediate forwarder (`intermediate`) or a pool (`pool`), and opens a finding like a built-in policy. See *Custom policies*.

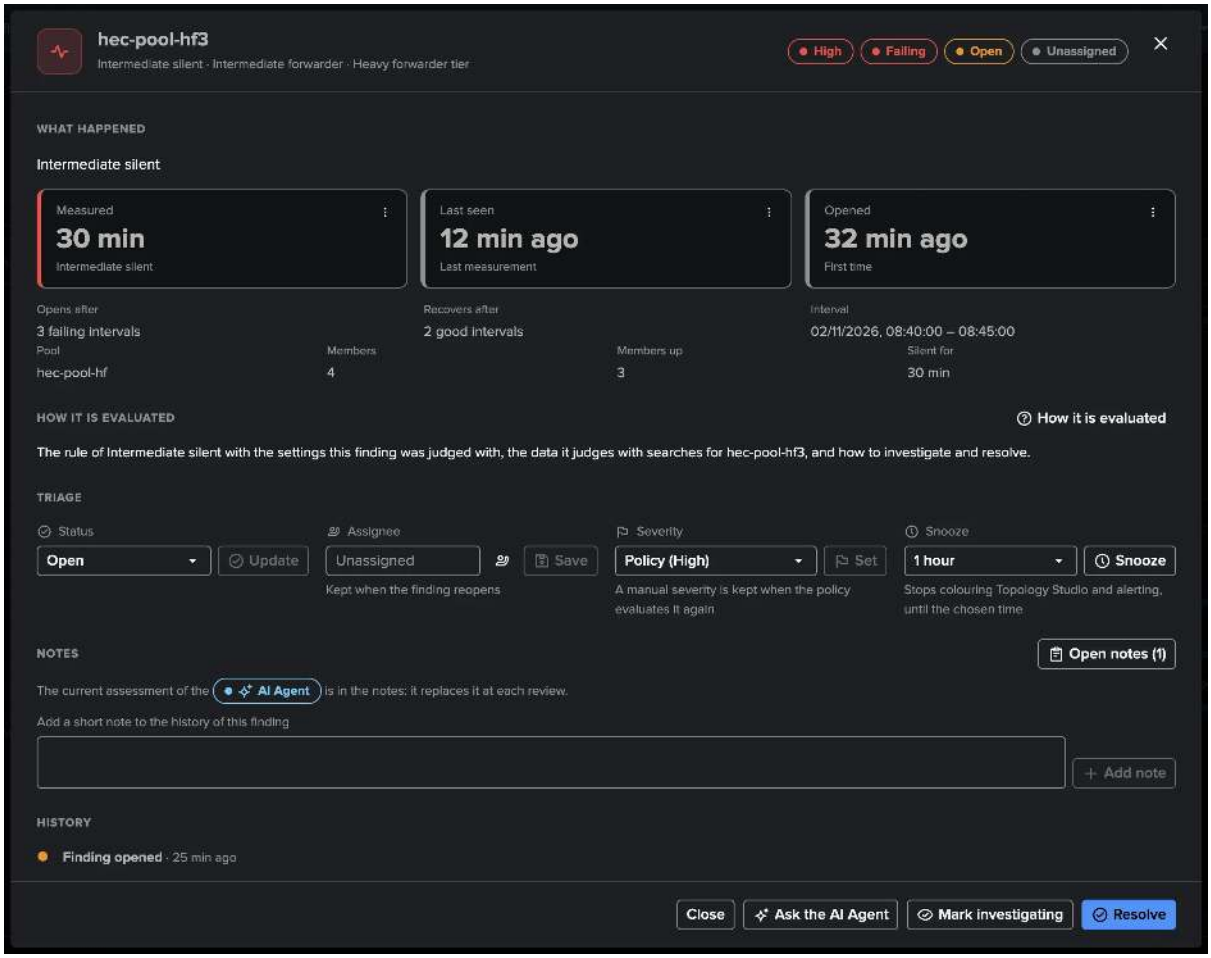
Findings in practice

A finding of a Splunk S2S tenant reads like every finding of a dedicated tenant (see *Findings, Finding groups and findings actions*): one row per Finding group, the decision history, the measurements of the policy, triage (investigate, snooze, dismiss, resolve), assignment, notes, findings actions and **Ask the AI Agent**. **Open resource** opens the host or the pool modal in place. The findings of one object form one Finding group: a stopped heavy forwarder's own findings read together, the pool's capacity finding on the pool, and the senders that cannot reach it on the destination. Merge them by hand into one group when one incident is worked as one.

Stopping one heavy forwarder of the tier opened three findings: **Pool capacity** on the Heavy forwarder tier first (three members up against four expected), then **Intermediate silent** on the heavy forwarder and **Destination unreachable** on its receiving port, measured on the 23 universal forwarders that failed to reach it for 15 minutes or more (24 of

24 failing). The forwarders load balanced to the other members, so no sender was held and no back pressure cascade opened.

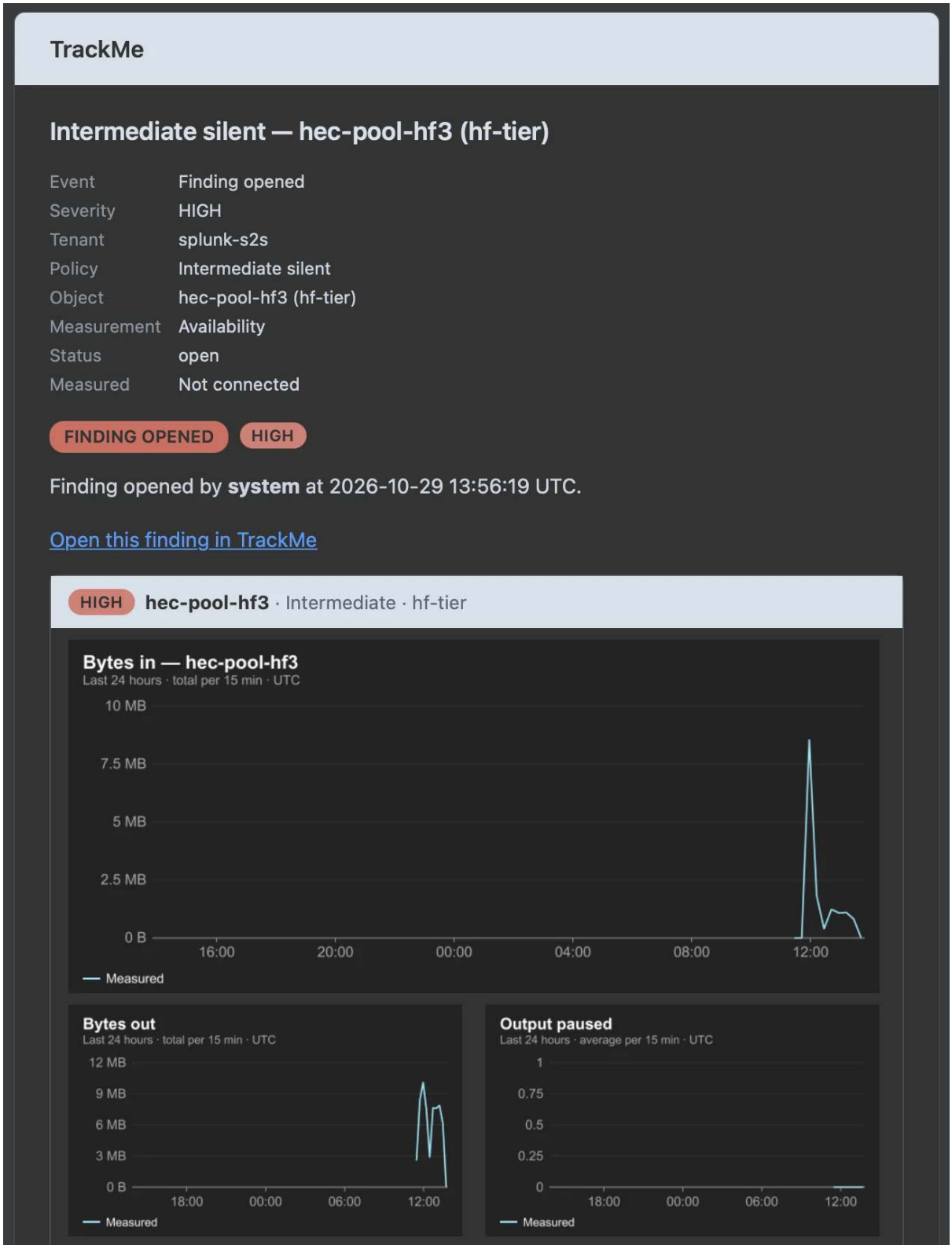




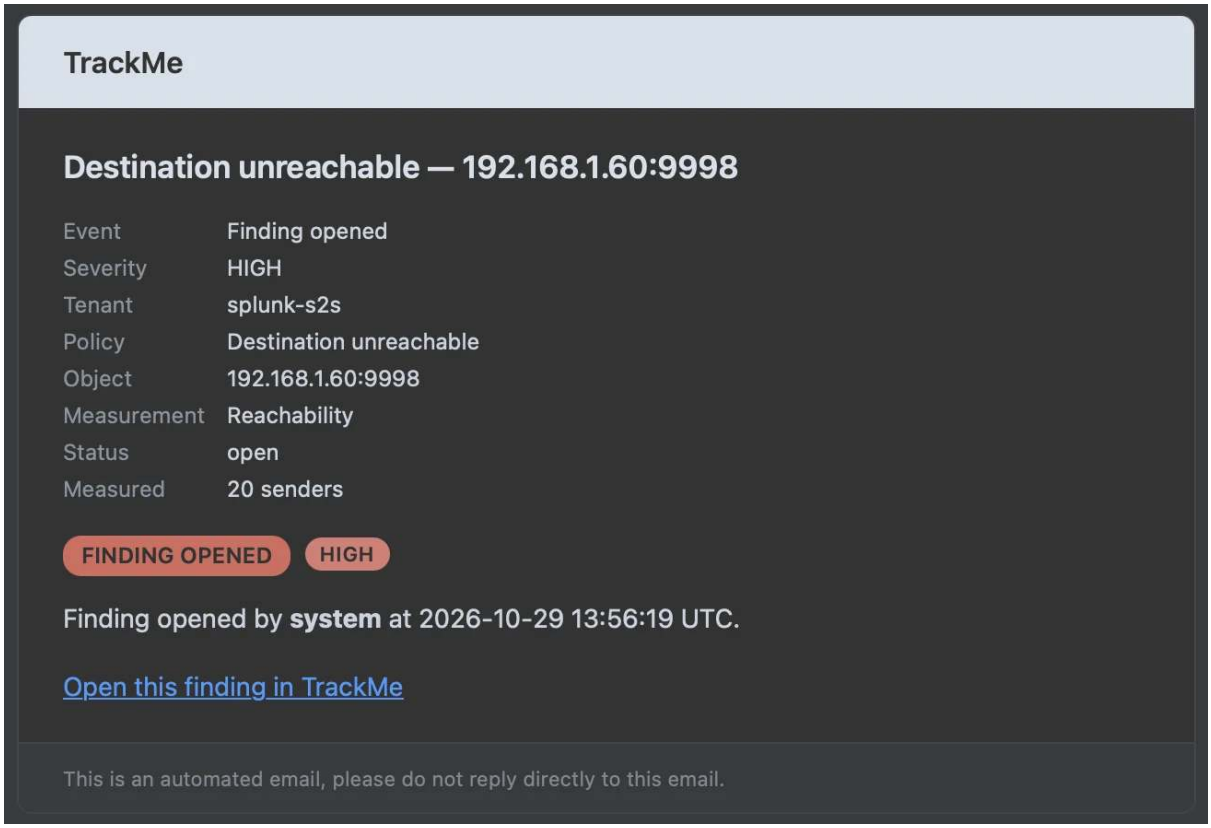
Notifications

Settings → Notifications emails the transitions of the tenant's findings: one digest per live run, or one threaded email per transition of a Finding group, from a minimum severity, for every policy or chosen ones, through an email delivery account, with the finding's chart embedded (in per Finding group mode, more than 10 Finding groups with updates to send at once go as one digest). Every transition is also written as an event to the tenant's notable index (`sourcetype= trackme:cribl:finding`), whatever the notification settings. See *Findings, Finding groups and findings actions*.

An intermediate forwarder of the pool went silent: the email names the policy, the object and its measurement, links back to the finding, and embeds the intermediate's last 24 hours (bytes in, bytes out, output paused).



The same stop made the intermediate unreachable for the universal forwarders that send to it: one **Destination unreachable** finding on its receiving port, measured on the senders that failed to reach it (here, in an earlier run of the same exercise, twenty).



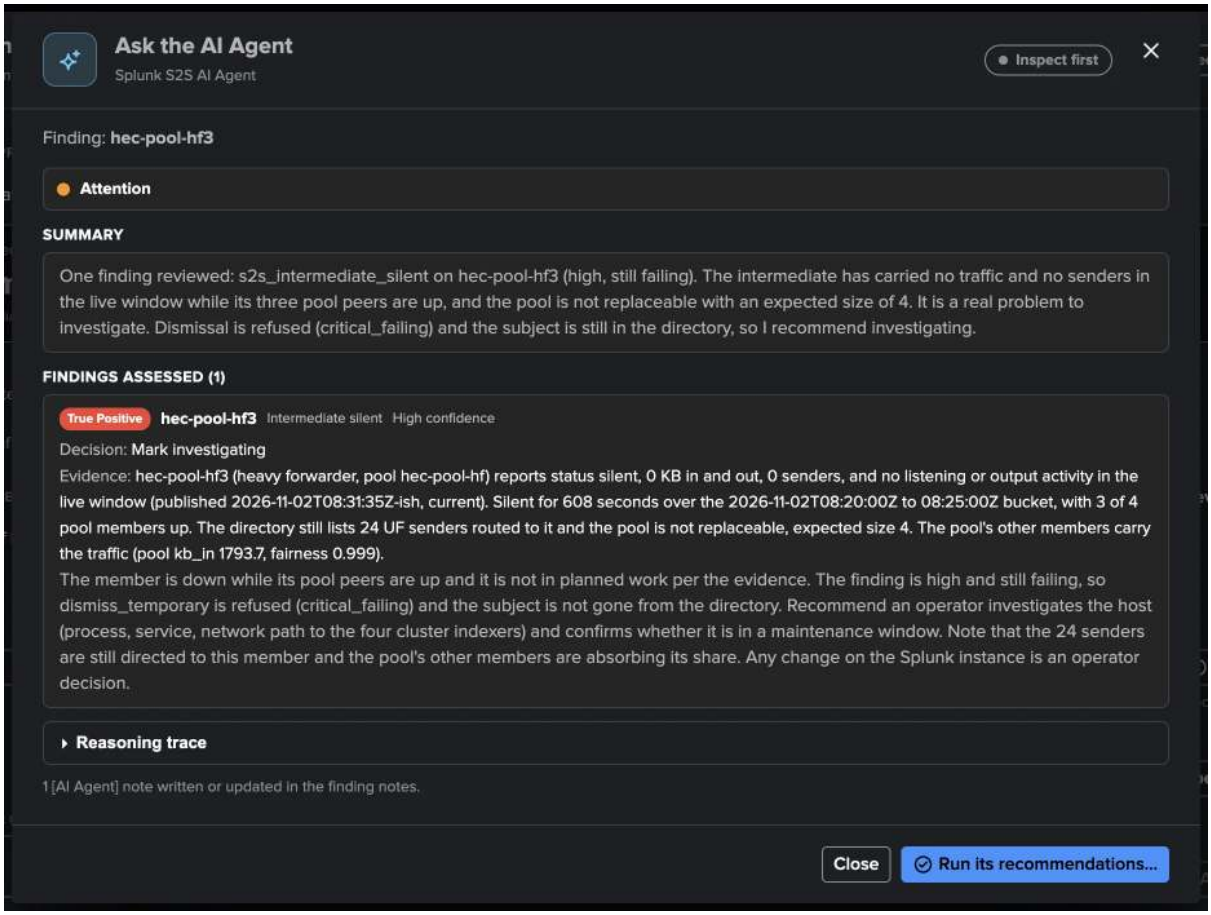
Topology Studio

Splunk S2S objects are first-class in Topology Studio: receivers, intermediates and pools in the entity picker and as dynamic groups (by status, the receivers of a pool, back pressure, open findings). Their status comes from their findings first; without one, a measured host is green and a silent one quiet. The templates **Forwarding path** (sender types → intermediates → receivers), **Receivers by pool**, **Problems wall** and **Intermediates** build a view in one click, and the AI Topology Builder knows the tenant ("Draw my forwarding path"). When a Splunk UF Fleet tenant reads the same source, its fleet is drawn linked to these receiving nodes. The workspace's **Data path** tab saves its generated topology to Topology Studio on request. See *Topology Studio*.

The Splunk S2S AI Agent

The **Splunk S2S AI Agent** reviews findings on the shared findings AI Agent framework (see *AI Agents for findings*). It reads what TrackMe holds about the finding — the host's live row (pipeline, listening ports, output trouble, origin, senders' mix, TLS, acknowledgement, resources, churn, health), its pool with the members needing attention and the cluster summary, collection health, the Finding group and past decisions — and may read further through its tools (the overview, a host, a pool, the senders, the data path, the traffic models, log evidence, trends, the last 5 minutes). It writes one `[AI Agent]` note per review; in act mode it applies only what the tenant allows: marking a finding investigating (allowed by default), a temporary dismissal of at most 7 days (allowed), and resolving after a measured recovery (not allowed by default). It never changes anything on a Splunk instance.

Ask the AI Agent from a finding or a Finding group runs it on demand. **Scheduled triage** is off by default (**Settings → AI Agent**, or *Manage AI Agents*): hourly, up to 10 findings per run, a finding reviewed again after a day unless it moved. The agent needs the AI Assistant enabled and an LLM provider; a scheduled run works with a least-privilege tenant owner, not with `nobody`.



The Tenant Home **AI Assistant** knows the workspace, its policies and its open findings: ask "Which pools have members down or backing up, and where is the bottleneck?" or "Which senders forward without TLS or acknowledgement?".

Collection health

The **Collection health** tab shows whether the tenant collects as configured: the **Status** view lists every problem with its meaning, its source, its causes and the steps to fix it, and the runs that did not complete in the last 24 hours; **Trackers** shows each job's last run with the searches it sent and a **Run now**; **Data & storage**, **Troubleshoot & help** and **Custom policies** (the runs of each custom policy) complete it. The same assessment raises the Configuration Guardian check `cribl_collection_health`. Every problem code is listed in *Splunk S2S reference*.



Troubleshooting

My indexer cluster shows as two pools. It is a multisite cluster: each site is a pool, judged on its own peers (capacity, balance, traffic). A single-site cluster, or one whose peers' sites are not all known, stays one pool.

My heavy forwarders are in a pool I never created. That is the automatic pool of their host-name family (Intermediate forwarders as pools). **Configure** it in Settings → Pools to rename it and give it its output groups; set the setting to Off to leave the intermediates no pool claims as discovered.

The cluster pools are empty and Collection health says `cluster_read_refused`. The reader (the tenant owner, or the source account's user) lacks `list_indexer_cluster`. Add it (read-only), or set **Indexer clusters as pools** to Off.

My intermediate's output is far smaller than what its receivers count. Its own count is compressed bytes on the wire; the receivers count uncompressed data. The workspace never compares them.

A pool's Senders is lower than the sum of its members' senders. A sender load balancing over N members is one sender for the pool: the counts are exact distinct counts.

Connection churn on every receiver? Splunk's automatic load balancing alone reconnects each sender many times an hour. The policy compares each receiver with the median of its pool's other members, never with an absolute count.

Findings stopped during a rolling restart. That is **Hold during planned work**: an open finding stays open, a new one waits until the cluster's planned work ends.

The manager account check fails, but the account searches fine. The clustering endpoints need `list_indexer_cluster`; an account that cannot serve a `| rest` read answers no row. The check names the missing capability and the role to create.

What the Splunk S2S tenant does not do

- It does not follow the forwarding agents themselves: that is *Splunk UF Fleet*.
- It does not judge the inputs of the collection points: that is *Splunk Forwarder Collectors*.
- It does not judge an indexer cluster's own health (peers, factors, buckets) nor a host's CPU and memory: both are shown, never findings.
- A sender behind a non-Splunk hop (Cribl Stream, for example) appears as the hop's workers; senders using S2S over HTTP are not counted.
- TLS and acknowledgement posture are read from the connection lines' `ssl` and `ack` flags.
- Several ingestion pipeline sets are merged, the worst queue of each kind winning.
- It never changes anything on a Splunk instance.

Next: the reference

Every policy with its defaults and settings, the custom policy subjects, the jobs, the settings, the Collection health codes, the commands and the permissions are in *Splunk S2S reference*.

8.3.22 Splunk S2S reference

The reference of the Splunk S2S tenant: every built-in policy with its defaults and settings, custom policies, the collection jobs, the settings, Collection health, the AI Agent, the indexed finding events, the permissions and the commands. The guide is *Splunk S2S*.

Built-in policies

Twenty-two policies in four categories. Every policy is evaluated on each live window (five minutes). Common settings of every policy:

- **Enabled** and **Severity** (Informational, Low, Medium, High, Critical; a few policies raise it, as their row says).
- **Checks to open** and **Checks to recover**: consecutive failing, then healthy, live windows.
- **Scope** (**Pools**): empty means every pool of the tenant and the hosts no pool claims.
- **During bank holidays**: keep opening findings, or hold (Pool traffic drop and Pool traffic outlier hold by default; see *Bank holidays*).

General rules:

- A **partial window** (a capped read, search peers that did not answer, a failed second read) may open a finding, never close one.
- The pipeline, output and resources policies decide only when those reads succeeded.
- A finding not evaluated for 24 hours closes as not observed.
- **Hold during planned work** (Settings → Pools, on by default) turns a failing decision of Receiver silent, Pool capacity, Pool not receiving, Pool traffic drop, Receiver imbalance, Connection churn and Pool traffic outlier into a hold on a cluster pool in maintenance mode or a rolling restart.

Receivers

Policy	What it detects	Default	Severity
Receiver not accepting	The receiver logs `TcpInputProc` "Stopping all listening ports" (its queues stayed blocked) with no "Started listening" since. Recovers on the restart line.	On	Critical
Receiver silent	No connection line in the live window while other members of its pool have lines (a whole pool silent is Pool not receiving, its members held). A replaceable pool waits for a replacement first.	On	High
Receiver back pressure	A queue of the receiver path saturated (90th-percentile fill ≥ the saturation, or blocked in ≥ the blocked share of its samples). The finding names the bottleneck and whether its thread was busy.	On	High
Receiver imbalance	In a pool of at least 3 active members, a member's share of the pool's senders (or volume) is below the fairness threshold times an even split.	On	Low
Connection churn	Reconnections per sender per hour at least 3 times the median of the pool's other members, and at least the minimum in the window. Recovers below 1.5 times.	On	Medium
Acknowledgment posture	A cluster peer receives senders with `ack=false` where acknowledgement is required (Settings → Posture: Required, or Automatic and most senders of the pool use it).	On	Low
TLS posture	Senders connect to a receiver with `ssl=false` where TLS is required (Required, or Automatic and the pool's majority is TLS).	On	Low
Version best practice	A sender of the day runs a newer major.minor version than the receiver (Splunk's matrix allows it; its best practice is receivers at least as new as every forwarder). Unknown while the receiver's version is unknown.	On	Low

Intermediates

Policy	What it detects	Default	Severity
Intermediate output blocked	The intermediate's `TcpOutputProc` paused the data flow for at least the blocked seconds in the window. Counts the senders held and names the destinations; an output queue still at 80 % or more keeps it open. Held while 30 % or more of the intermediates (and at least 2) are paused together: the next tier is the problem, and the Back pressure cascade and Destination unreachable findings say which.	On	Critical
Intermediate back pressure	A queue of the intermediate path saturated. Names the origin: **downstream** (its output paused, or a receiver pool it feeds backs up), **upstream surge** (inbound above 1.5 times its learned volume), else **local**.	On	High
Intermediate silent	As Receiver silent, within the intermediate pool.	On	High
Intermediate destinations failing	The intermediate fails to reach one of its destinations without a break for 15 minutes while the other senders reach it (a destination many senders fail on is that destination's finding).	On	Medium
Intermediate single destination	An output group names one destination for the minutes set while it named two or more in the last 24 hours.	On	Medium
Intermediate in / out anomaly	What it receives and what it forwards drift beyond the factor (or below its inverse) of their learned ratio (the median of up to 7 daily medians, 3 days learned), without an output pause.	On	Medium
Intermediate TLS posture	Senders connect to the intermediate without TLS where its pool expects it (the forwarders sending to a heavy forwarder tier).	On	Low
Universal forwarder relay	A universal forwarder acting as intermediate receives universal forwarders (Splunk advises against it: partial events, blocked output queues).	On	Low

Pools

Policy	What it detects	Default	Severity
Pool not receiving	Every member of a receiver pool is silent.	On	Critical
Pool capacity	Fewer members up than expected (the cluster's peers, else the configured size, else the size learned over 7 days). Nothing is decided while nothing is learned; a replaceable pool waits for replacements.	On	High → Critical when no member is up
Pool traffic drop	The pool's rate is below the drop percentage of the same hour's median over 7 days (3 days learned at least). Recovers at 80 %.	On	High → Critical when nothing is received
Pool traffic outlier	KB received (or connected senders) outside the traffic model's range for this hour of the week, from the model confidence set, when the window or the model reaches the minimum.	On (needs a model)	Medium
Back pressure cascade	In one window, a receiver pool has a member backing up or not accepting **and** an intermediate is paused towards it (an intermediate tied to no pool counts towards every receiver pool). One finding names the origin and the intermediates held.	On	Critical

Destinations

Policy	What it detects	Default	Severity
Destination unreachable	At least the shared minimum of senders, and the shared share of the destination's senders, fail to reach one receiver without a break for the open delay (longer while its cluster peer restarts). One finding on the destination listing the senders; the senders' own destination findings are held.	On	High

Default settings

Each policy's checks to open and to recover (one check per live window, every five minutes) and its own settings with their defaults.

Policy (id)	Open / recover	Settings and notes (defaults)
Receiver not accepting (s2s_receiver_not_accep	1 / 2	No setting.
Receiver silent (s2s_receiver_silent)	3 / 2	Grace 0 s; replacement grace 1,800 s
Receiver back pressure (s2s_receiver_back_pres	2 / 2	Saturation 80 %; blocked share 0.3
Receiver imbalance (s2s_receiver_imbalance	3 / 2	Fairness 0.8; measure connections or volume
Connection churn (s2s_receiver_connectic	2 / 12	At least 20 reconnections
Acknowledgment posture (s2s_receiver_ack_postu	1 / 2	At least 1 sender; recover after 24 h
TLS posture (s2s_receiver_tls_postu	1 / 2	Recover after 24 h
Version best practice (s2s_version_best_pract	1 / 2	Recover after 24 h
Intermediate output blocked (s2s_intermediate_outpu	3 / 2	Blocked 30 s; shared from 30 % and 2 intermediates
Intermediate back pressure (s2s_intermediate_back_	2 / 2	Saturation 80 %; blocked share 0.3
Intermediate silent (s2s_intermediate_silen	3 / 2	Grace 0 s; replacement grace 1,800 s
Intermediate destinations failing (s2s_intermediate_desti	1 / 2	Open after 900 s
Intermediate single destination (s2s_intermediate_singl	1 / 1	15 minutes
Intermediate in / out anomaly (s2s_intermediate_fan_i	3 / 2	Ratio factor 2.0
Intermediate TLS posture (s2s_intermediate_tls_p	1 / 2	Recover after 24 h
Universal forwarder relay (s2s_uif_relay)	1 / 2	Recover after 24 h
Pool not receiving (s2s_pool_not_receiving	1 / 1	No setting.
Pool capacity (s2s_pool_capacity)	2 / 2	Replacement grace 1,800 s
Pool traffic drop (s2s_pool_traffic_drop)	2 / 2	Drop 50 %
Pool traffic outlier (s2s_pool_traffic_outli	2 / 2	Measure (KB received or connected senders); direction; at least 100; open from normal confidence
Back pressure cascade (s2s_backpressure_casca	3 / 2	No setting.
Destination unreachable (s2s_destination_unreac	1 / 2	Open after 900 s (1,800 s while the peer restarts); shared from 30 % and 3 senders

Custom policies

TrackMe administrators of the tenant write their own policies in the **Policies** tab, above the catalogue: an SPL search run on this search head as the tenant owner, on its own schedule. Each result row names one subject, which opens a finding like a built-in policy and joins the Finding group of its object.

Subject type	Result field	Resolves to
Receiver	`receiver` (host name)	The host in the live publication (else the directory), with its pool. A host the tenant does not follow as a receiver is not monitored.
Intermediate forwarder	`intermediate` (host name)	The same, as an intermediate.
Pool	`pool` (pool id or name)	A configured pool, a cluster or cluster site pool, an automatic intermediate pool or `discovered`; a name two pools carry is ambiguous.

A row whose subject is not monitored is skipped and the run's other findings recover; an unresolved row (ambiguous, or before the first publication) keeps every finding of the run as it is. A remote source is searched through `| splunkremotesearch` in the SPL. The examples offered include *Receivers logging S2S input errors* and *Intermediate forwarders failing TLS*. Drafts, published revisions and activation are managed independently, Splunk validates the SPL before every preview and run, and each policy's runs are in Collection health. See *Findings, Finding groups and findings actions*.

Collection jobs

Job	Schedule	What it does
Inventory	Every 10 minutes	Roles over a day, the intermediates' type, the indexer clusters (the search head's view every run, the manager through its account), the pools, the telemetry probe; publishes the directory (rebuilt about hourly, always on a manual run).
Live	Every 5 minutes	Index-time windows with catch-up: per host the inbound connections, senders, posture and output; assembles the latest facts of the jobs below; per pool the sums, members up and balance. Writes the series, then evaluates every policy. Never builds the directory: a run before the first inventory waits for it (`s2s_directory_pending`).
Pipeline	Every 5 minutes	Each host's queues and listening ports, the intermediates' output trouble, the resources and the connection churn, over strict index-time windows of its own.
Health report	Every 5 minutes	The hosts' health report indicators, with the detail for the flagged hosts only.
Destinations	Every 5 minutes, just before the live run	The destinations the senders fail to reach and the cluster peers restarting.
Traffic ML models	Twice an hour	Learns each receiver pool's models when due.
Policies	On demand	Evaluates the policies on the latest live window.

Schedules are offset per tenant. **Collect now** (the status strip) runs the live job; **Run now** in Collection health runs a job; **Scheduled collection** (Settings → Collection) is on by default, off means manual collection only. The reports are `trackme_splunk_s2s_<job>_<generation>_tenant_<tenant>`, and the AI Agent report `trackme_s2s_agent_tracker_tenant_<tenant>` (scheduled only once the scheduled triage is enabled).

Each job has its own report, time budget (its interval less 30 seconds), lock, receipt and Collection health line (2.4.23), and a search may use what is left of its job's budget: there is no fixed per-search timeout. The fact jobs run in the minutes before the next live run; a job that misses a slot makes the policies hold, never count twice, and facts older than three runs read as not read. With **Scheduled collection** off, a **Collect now** starts the other jobs beside the live run,

and it takes a second **Collect now** before the policies judge anything. The reports run `| trackmesplunkcollect` (log `trackme_splunk_collect.log`); up to five jobs of a tenant can overlap under the tenant owner, so a least-privilege owner may need a higher search quota.

Settings reference

Settings is one browser with a single **Save settings**. A tenant role that is not a TrackMe administrator of the tenant sees it read-only. Changing a field marked *starts a new source* restarts collection, and the findings of the previous source close. Every policy threshold is a policy setting, never a tenant setting.

Section	Setting	Default	Notes
Source	Source (search tier)	`local`	Fixed once the tenant exists: create another tenant for another deployment.
	Internal, Metrics, Introspection index	`_internal`, `_metrics`, `_introspection`	Fixed once the tenant exists; an empty Metrics index means `_internal` only.
	Telemetry form	Automatic	Automatic, `_metrics` mirror, or `_internal`; starts a new source.
	Host patterns, Excluded hosts	`*`, none	Which hosts may be receivers or intermediates, one token per host name or pattern (type it and press Enter); never `*` alone in the exclusions; starts a new source.
	Repair workspace	—	Recreates the tenant's collections and reports; findings and settings are kept.
Pools	Indexer clusters as pools	Automatic	One pool per indexer cluster, or per site of a multisite cluster (`<cluster>:<site>`, when every peer has a site and at least two sites are known). Off: clusters ignored, configured pools only.
	Intermediate forwarders as pools	Automatic	The intermediates no pool claims (no assignment, no configured pattern) form one automatic pool per host-name family. Off: they wait as discovered.
	Hold during planned work	On	The maintenance hold of the cluster pools.
	Manager accounts	None	Per indexer cluster (every site pool of the cluster uses it), checked as chosen.
	Configured pools	None	Up to 200 pools in all (automatic ones included), with up to 50 host patterns and 50 output groups each. **Configure** on an automatic pool makes it a configured pool with the same id.
	Auto-join, hosts assigned by hand	On, none	A newly seen host joins the pool whose patterns match; up to 500 hosts assigned by hand from each host's pool pill, never to a replaceable pool.
Posture	TLS, Indexer acknowledgement	Automatic	Automatic (the pool's majority), Required, Not required.
Collection	Scheduled collection	On	
Traffic ML models	Model learning history, model defaults	14 days	0, 7, 14, 30, 60 or 90 days.
Notifications, AI Agent, Assignment, Finding actions	Shared settings of the findings tenants	Off / none	See *Findings, Finding groups and findings actions*; the assignment criterion is the **pool**.

Collection health codes

The Collection health tab and the Configuration Guardian check `cribl_collection_health` (warning) report:

Code	Severity	Meaning
`live_never_ran`, `live_not_running`, `live_failing`	Warning, Failure, Failure	No live run since configuration; none for 20 minutes; runs not completing for 30 minutes.
`inventory_never_ran`, `inventory_not_running`, `inventory_failing`	Warning, Failure, Failure	No inventory run; none for 3 hours; not completing for 90 minutes.
`job_never_ran`, `job_not_running`, `job_failing`	Warning, Failure, Failure	A dedicated job (2.4.23) never ran 15 minutes after the first assembled snapshot; none for 15 minutes; runs not completing for 30 minutes. Until its next run, the policies reading its facts hold.
`policies_failing`	Failure	Policy evaluation not completing for 30 minutes.
`models_failing`	Failure	Traffic model training not completing for 90 minutes.
`no_forwarding_receivers`, `no_s2s_hosts`	Warning	An hour after configuration, the source shows no receiver of forwarded data; or the directory holds no receiver and no intermediate.
`cluster_read_failed`, `cluster_read_refused`, `cluster_peers_failed`	Warning	The cluster read failed (last membership kept); the reader lacks `list_indexer_cluster`; the peers' read failed (maintenance mode and restart state unknown).
`cluster_unreported`, `cluster_membership_unreadable`, `cluster_manager_unreadable`	Warning	A cluster pool the search head no longer reports; the search head cannot reach a cluster's manager; a pool's manager account fails (the planned-work hold relies on the search head).
`identity_read_failed`, `roles_unresolved`	Warning	The intermediates' type read failed; intermediates that could not be typed.
`pipeline_read_failed`, `pipeline_read_partial`	Warning	The live run's pipeline read failed or was partial: the pipeline policies wait.
`receivers_telemetry_late`	Warning	The receivers' own telemetry arrives later than the settle delay.
`capacity_reached`	Warning	A run reached one of its bounds (a capped read, the metric points' bound).
`notification_delivery_failing`	Warning	Notification emails failing.
`custom_*`	Warning	Custom policies: schedule drift, failing runs, missed schedules, recovery required.

AI Agent

The Splunk S2S AI Agent settings (**Settings → AI Agent**, or *Manage AI Agents*):

Setting	Default	Notes
Scheduled triage	Off	Hourly when on (`ai_s2s_agent_enabled`).
Mode	Inspect	Inspect writes assessments and notes; act also applies the decisions allowed below.
AI provider	First available	
Findings per run, time budget per run	10, 3,000 s	
Review interval	1 day	A finding reviewed by a scheduled run is skipped that long unless it moved.
Allow investigating, allow temporary dismissal, allow resolve	Yes, yes (up to 7 days), no	A scheduled run never overrides an operator decision.
Custom instructions	None	Added to the agent's instructions; **Evaluate…** tries them on a finding.

Runs are audited with the sourcetypes `trackme:ai_agent:s2s_agent:inspect` and `trackme:ai_agent:s2s_agent:act`. See *AI Agents for findings*.

Indexed finding events

Every finding transition is written to the tenant's notable index as `sourcetype= trackme:cribl:finding` (and Finding groups as `trackme:cribl:finding_group`), with `object_category=splunk_s2s`, whatever the notification settings: use them for Splunk alerts, Splunk Enterprise Security, SOAR or dashboards. The tenant's series are in its metric index under `trackme.s2s.receiver.*`, `trackme.s2s.intermediate.*` and `trackme.s2s.pool.*`.

Permissions

- Any role of the tenant reads the workspace.
- The tenant's power and admin roles triage findings.
- A TrackMe administrator of the tenant (`trackmeadminoperations` and an admin role of the tenant) edits Settings, policies, custom policies, model tuning and relearning.
- The scheduled runs search as the **tenant owner**: every remote account the tenant reads must be usable by the owner, and the indexer cluster reads need `list_indexer_cluster`.

Commands

Command	What it does
`| trackmes2sdiscover`	The wizard's discovery (read-only, as the caller); `manager_accounts=<a, b>` adds the manager account check, `mode=accounts` runs the check alone.
`| trackmes2strend`	TrackMe's S2S series of a receiver, an intermediate, a pool or the tenant (`kind=receiver|intermediate|pool|tenant`), with the model band on pools.
`| trackmes2srecent`	The Last 5 minutes figures of the tenant, a pool or a host (never a state, writes nothing).
`| trackmes2sevidence`	The raw lines behind a figure for a host or a pool's members (`kind=back_pressure|posture|churn|health`, `range=3600|14400|86400`): a timeline, a breakdown and at most 50 lines. `kind=queue_history` (a host only, `range=900` too) reads each queue's usage, used and capacity bytes and blocked samples: the charts of the **Blocked now** dialog.
`| trackmes2spath`	The Data path of the newest window, read with the caller's rights (nothing saved).

8.3.23 Splunk UF Fleet

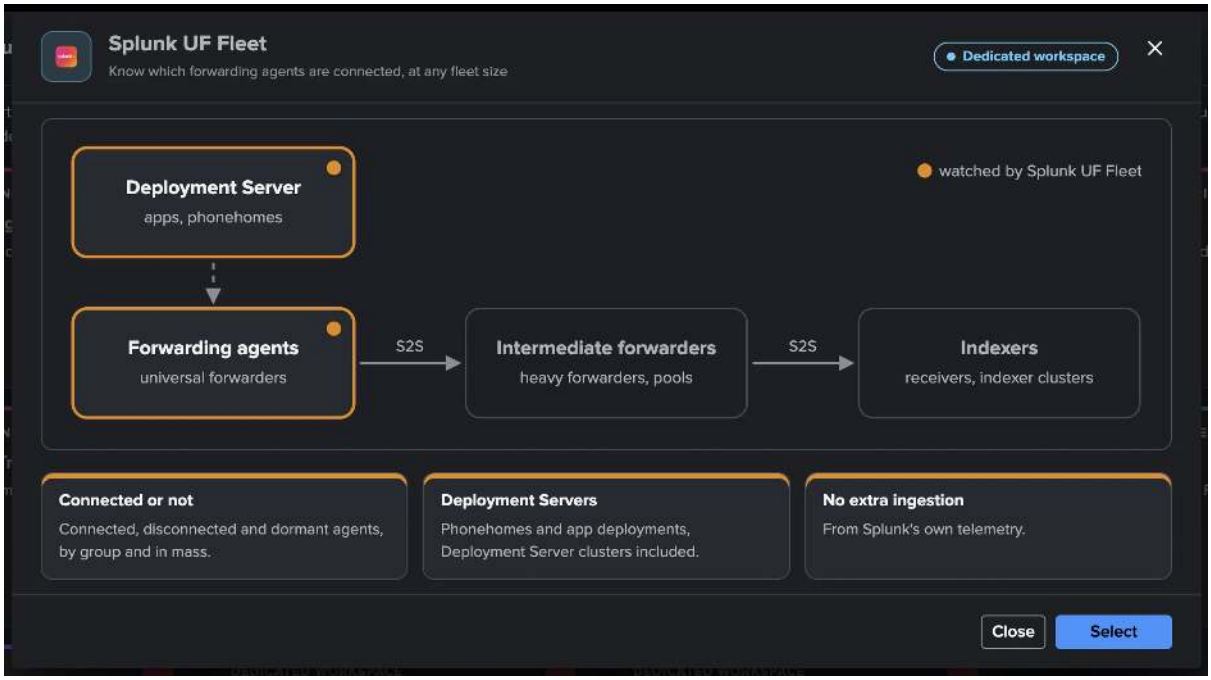
The **Splunk UF Fleet tenant** follows the **forwarding agents** of one Splunk deployment and the **Deployment Servers** that manage them. It answers the questions a Splunk administrator asks about a fleet of universal forwarders: *how many agents do I have, on which versions, platforms and types? which are connected, disconnected or dormant, and since when? which agents that must stay connected are not? which are blocked, throttled, failing their destinations, restarting in a loop, not phoning home or failing their app deployments? and per Deployment Server: its clients, its phonehome pace against its usual, its deployment failures, the restarts it asked for, the requests it refused and how fast it answers?* The answers are **findings**, raised by built-in policies, with the same triage, Finding groups, notifications, assignment and findings actions as the other dedicated tenants, and a findings AI Agent of its own, the Splunk UF Fleet AI Agent.

A Splunk UF Fleet tenant is a **dedicated tenant type**: it has no entities and no state colour, and one tenant follows the fleet of one Splunk deployment. It reads what Splunk **already writes**: the receivers' connection lines, the agents' own internal logs when they forward them (the default of a universal forwarder), and the Deployment Server indexes. TrackMe never polls an agent, never calls an agent's REST API and never installs anything on it: the tenant only reads.

Connected comes from the data path itself. An agent is connected when a Splunk receiver saw a connection from it within the tenant's connected window (15 minutes by default). The receivers describe the whole fleet, whatever its size, with no heartbeat to deploy.

Mass events are one finding. When many agents disconnect together — a network cut, a receiver tier down, a bad outputs push — one Mass disconnect finding on the fleet or on the group says it; the agents' own disconnection findings are held, never opened one by one.

The Splunk UF Fleet tenant is generally available since TrackMe 2.4.22. The receiving side of the same deployment (the intermediate forwarders, the indexers and their pools) has its own tenant type, *Splunk S2S*.



For a guided tour on a live fleet, see the hands-on walkthrough *Splunk UF Fleet*. Every policy, setting, job and command is described in the *Splunk UF Fleet reference*.

What it monitors

Object	What TrackMe watches
Agents	Every forwarding agent: its **connection** (connected, disconnected, dormant, removal candidate), whether it is one of the **expected agents** that must stay connected, its own troubles from its logs (**output blocked**, **destinations failing**, **throttled**, **ingestion latency**, **input errors**, **restart loop**), its **identity** (one GUID on several hosts, cloned images), its **version** (unsupported, newer than its receivers), and its **Deployment Server** client (not phoning home, handshake failing, app deployment failed).
Fleet and groups	The fleet as a whole and every **group** you define (by Deployment Server, server class, lookup table, enrichment value, host-name pattern or output group): **mass disconnect**, **coverage** against a minimum or the learned daily peak, **version drift**, and **cloned images** deployed without clearing their identity.
Deployment Servers	Every Deployment Server serving the fleet: **silent**, **phonehome drop** against its usual for the hour, **deployment failures**, **restart storms** (clients restarted again and again), the **errors** of its latest configuration load, **requests rejected** and **response slow**.
Destinations	A receiver that **many agents fail to reach** without a break: one finding on the receiver listing the agents, never one per agent.

How it works

An agent is a leaf forwarder. A sender seen on a receiver that is not itself a receiver, an intermediate forwarder or a Deployment Server. By default only **universal forwarders** are agents: a heavy forwarder cannot be told apart from a search head or a cluster manager without reading each host, so full instances are counted apart ("Left out: full instances"); Settings → Source can count them as agents.

Identity. The GUID first (the receivers' `guid`, the Deployment Server's `clientId`), the host name for display and as the key of an agent without a GUID. The receivers' connection lines (24 hours), the Deployment Server's phonehomes (a day) and its client records (30 days) are merged: an agent known only to a Deployment Server is listed too.

Agent states.

State	Meaning
Connected	A receiver saw a connection line from it within the connected window (15 minutes by default, 5 to 60).
Connected (heartbeat)	No receiver saw it, but its own internal logs arrive within the window: the agent sends behind a non-Splunk hop or over HTTP. Counted connected, shown with a cue. A heartbeat never creates an agent, and never keeps an older identity of a host connected: when the receivers see its host name under another GUID, the logs are that instance's (see **Forgotten**).
Disconnected	Neither, for less than the dormancy delay.
Dormant	No connection line for 7 days (1 to 90): it leaves every fleet count, and its disconnection finding closes as dormant, never as a recovery.
Removal candidate	No connection line for 30 days (1 to 365), counted from its last connection: opens Decommission candidate.
Forgotten	An operator **Forgot** it, or TrackMe did: a host rebuilt or reinstalled under its name comes back with a new GUID, and once that new identity is connected, its older identity is forgotten. Out of every count at once, and listed in the Agents tab only with **Show forgotten**; it comes back connected if it connects again.

Built for any fleet size. Every fleet-wide read is an aggregation over indexed terms or metric dimensions (`tstats` with `TERM()` and `PREFIX()`, `mstats`), never a search per agent. The agent directory is paged and written only when an agent's state or identity changes, and mass events are one cohort finding. No per-agent history is written: an agent's chart reads Splunk's own lines on demand.

Job (Collection health label)	What it does
Inventory (every 10 minutes; the directory is rebuilt about hourly, always on a manual run)	Builds the agent directory from the three identity sources and the Deployment Servers and their clients, with the server class and output group members of the groups and the enrichment lookups. The Deployment Server client records and phonehome baseline come from the Deployment Server history job, with no search of their own. The receivers' versions are read once a day, and the probe that chooses the telemetry form runs when its verdict is due.
Deployment Server history (hourly)	The client records indexed since the last read (the first read covers 30 days), the operating system each agent reports when it phones home, and once a day each Deployment Server's phonehome baseline (the last 7 complete days). The inventory applies them without a search; until the job's first run, or when it has not run for a few hours, the inventory reads the client records and the baseline itself.
Live (every 5 minutes)	The connected set over the connected window and the heartbeats; assembles the latest facts of the jobs below into the fleet snapshot, the groups and their counts; publishes the fleet summary, writes the series, then **evaluates every policy**. A live run never builds the directory: until the first inventory, it admits the connected agents and the policies judge nothing.
Agent trouble, **Destinations**, **Identity conflicts**, **Deployment Servers**, **Deployment Server detail** (every 5 minutes each)	Dedicated jobs, each with its own report, schedule, time budget and Collection health line (2.4.23): the agents' own troubles and health indicators (the day's input errors once an hour), the destinations the senders fail to reach, the identity conflicts of the connected window, the Deployment Servers' phonehomes, late and absent clients and deployments, their resources and access log (the deployment failures every 15 minutes, their own errors once an hour). A job that misses a slot makes the policies hold, never count twice; facts older than three runs read as not read.
Policies (on demand)	Evaluates the policies again on the latest live publication.

A Splunk UF Fleet tenant learns **no traffic model**: its baselines are plain statistics, the daily peak of connected agents and the Deployment Servers' phonehomes per hour of the day.

Before you start

A licence. A valid **Enterprise**, **Unlimited**, Unlimited trial or **Developer** licence. The Splunk UF Fleet tenant is not available with the Foundation edition or the Foundation trial.

A source. The search tier that sees the receivers' `_internal` (and `_metrics` / `_introspection` when used): **local**, or a **TrackMe remote account** (*Remote Splunk deployments*) to another deployment. The three index names are configurable.

The Deployment Server indexes (Splunk 9.2 and later on the Deployment Server): `_dsphonehome`, `_dsclient` and `_dsappevent`, searched on the **Deployment Server source** (local by default: a full instance forwards them to the indexers by default). On **Splunk Cloud** those indexes stay on the Deployment Server: point the Deployment Server source at a remote account on the Deployment Server itself, through a TrackMe Broker when the Deployment Server is on premises (see *TrackMe Broker*).

Optionally, a Deployment Server API account per Deployment Server. A TrackMe remote account on each Deployment Server gives the **exact members** of each server class group (its client listing). Its user needs only `list_deployment_server` and `rest_properties_get`, never an admin role. Without one, a server class group counts the clients with an app of the class deployed in the last 60 days; every other view and policy is the same. In a Deployment Server cluster one member's account lists every client.

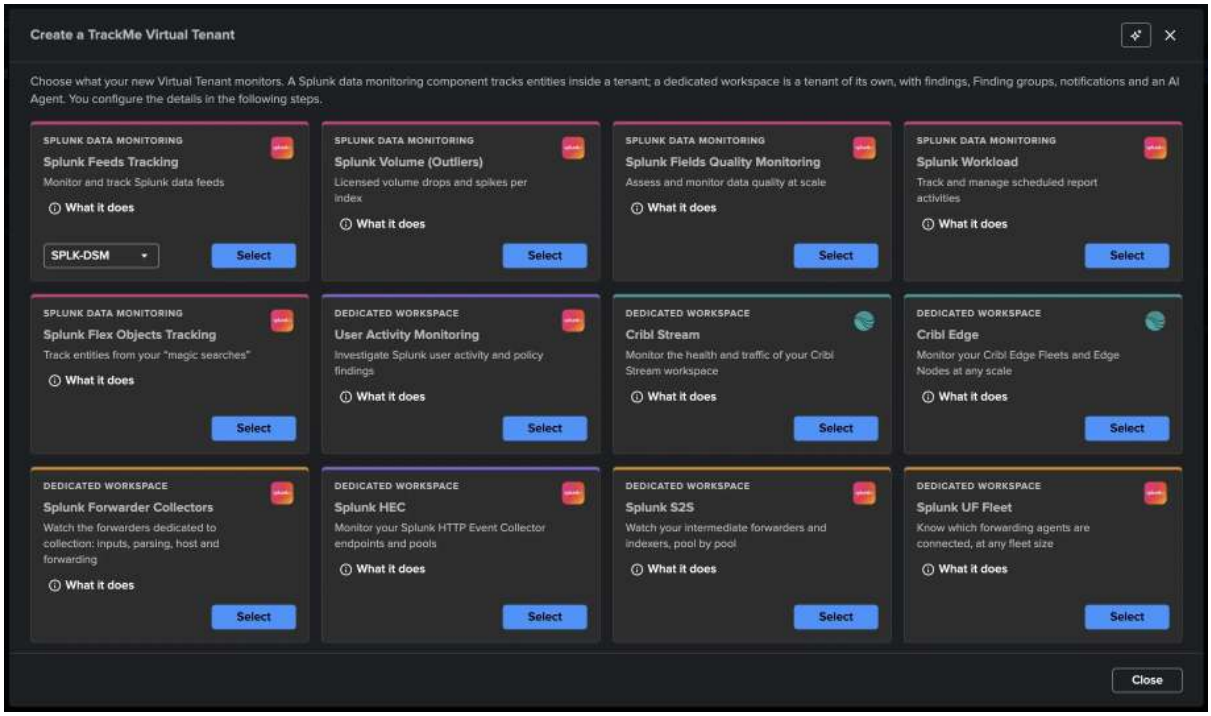
Least privilege. The tenant's scheduled searches run as the **tenant owner**: every remote account the tenant reads (the source, the Deployment Server source, the API accounts, the lookup tables) must be usable by the owner. The wizard and the Settings check this before saving and name the fix (add one of the owner's roles to the account in *Configuration → Remote accounts*, or choose another owner).

A tenant owner: `nobody`, or an account (typically a least-privilege service account) holding the capabilities the wizard lists. The scheduled triage of the Splunk UF Fleet AI Agent needs an owner that is not `nobody`.

The agents' own logs (optional). Health, output pauses, throttle, input errors, restarts and handshake retries come from each agent's own `splunkd.log`. A universal forwarder forwards its `_internal` by default; an agent whose internal logs are filtered out stays connected (from the receivers' lines) but has none of these signals.

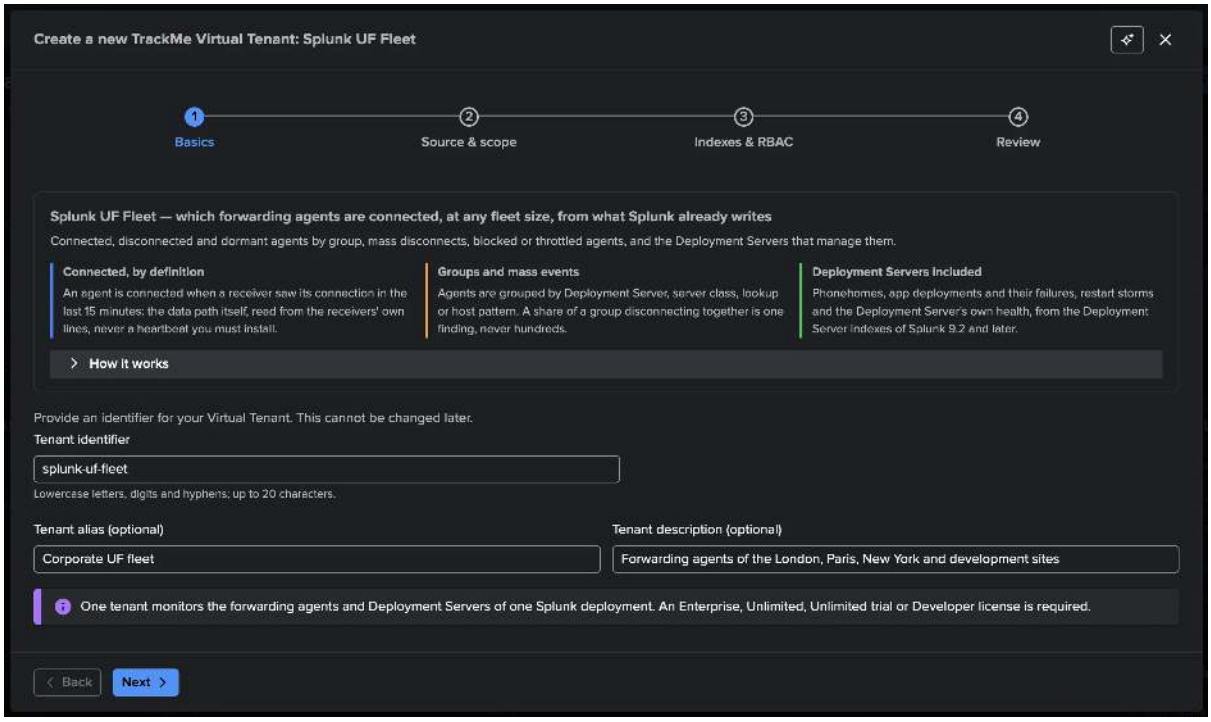
Creating the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Splunk UF Fleet** card ("Know which forwarding agents are connected, at any fleet size"). **What it does** opens the dialog shown at the top of this page.

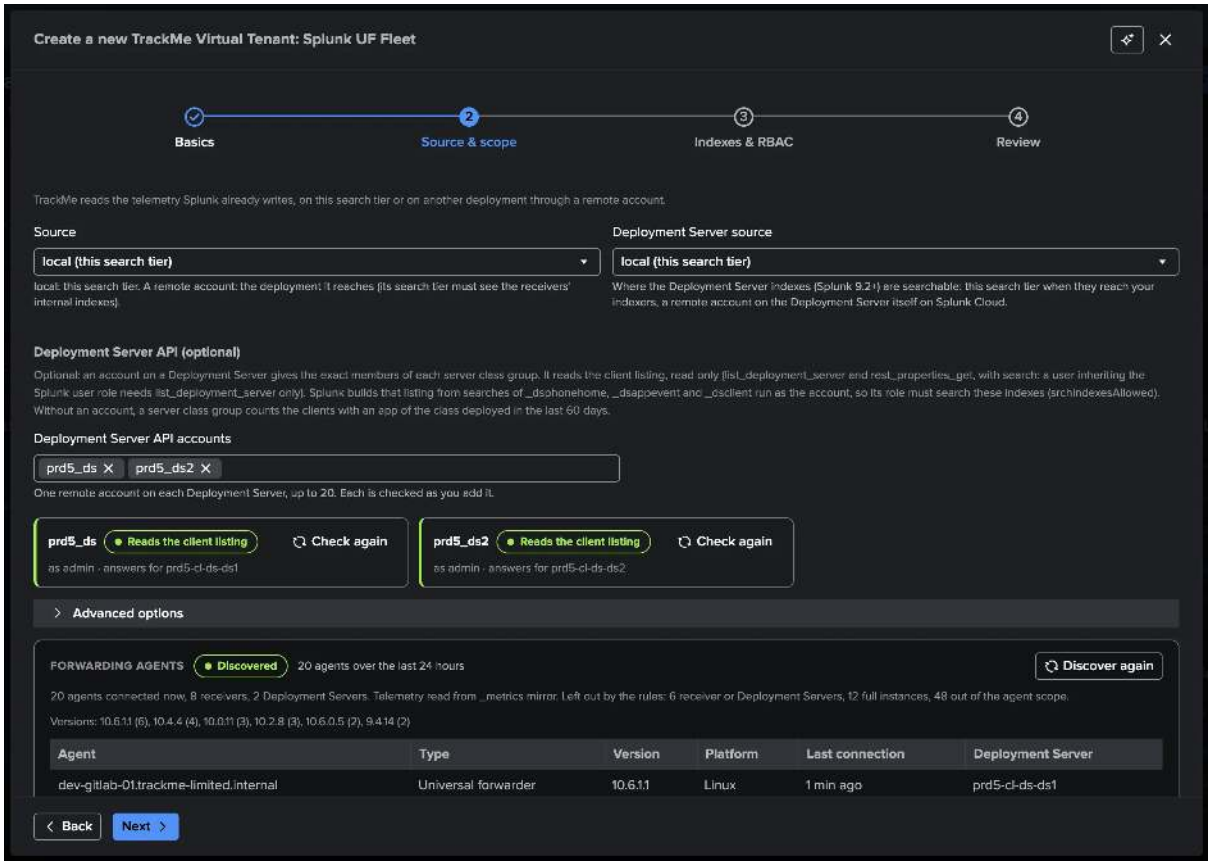


The wizard has four steps.

1. Basics — an introduction (connected by definition, groups and mass events, Deployment Servers included), then the tenant identifier (lowercase letters, digits and hyphens, up to 20 characters, fixed once created), an alias and a description. One tenant monitors the forwarding agents and Deployment Servers of one Splunk deployment.



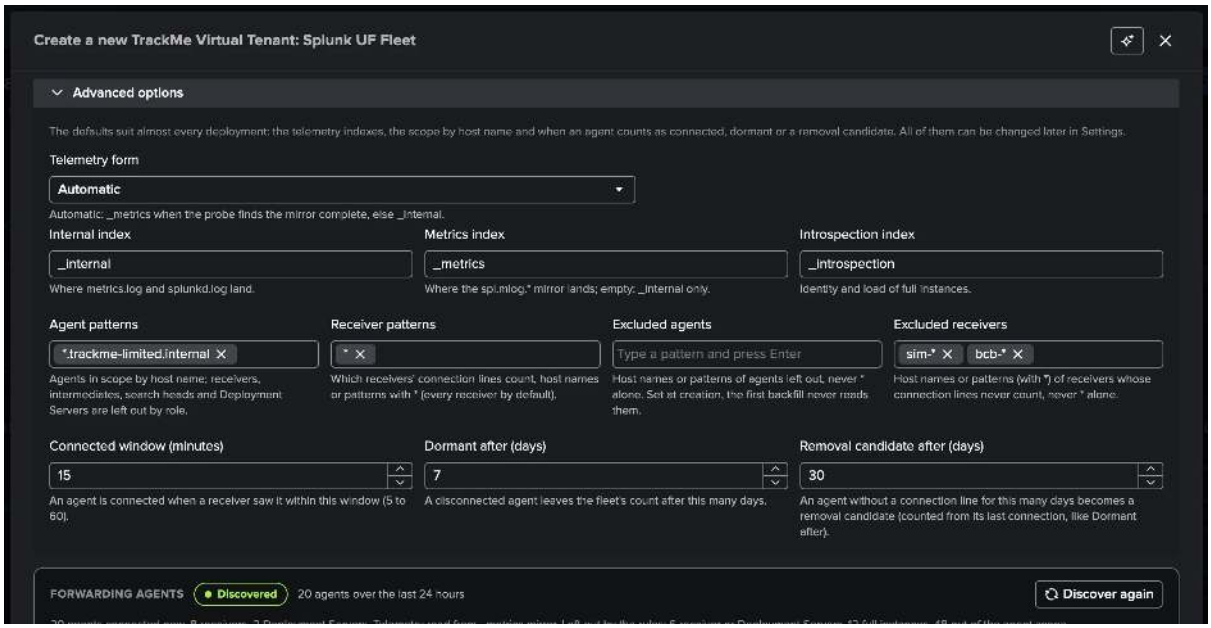
2. Source & scope — the **Source**, the **Deployment Server source** and the optional **Deployment Server API accounts**. Each account is checked as soon as you add it: who it is, whether it reads the client listing, and which Deployment Server answers ("Reads the client listing", "Lists no client", "Cannot search the DS indexes", "Listing refused", "Unreachable", with **What to do** and **Check again**). *Cannot search the DS indexes* means the discovery sees clients phoning home to that Deployment Server while the account sees none: Splunk builds the client listing from searches of `_dsphonehome`, `_dsappevent` and `_dsclient` run as the account, so its role must be allowed to search these indexes (`srchIndexesAllowed`).



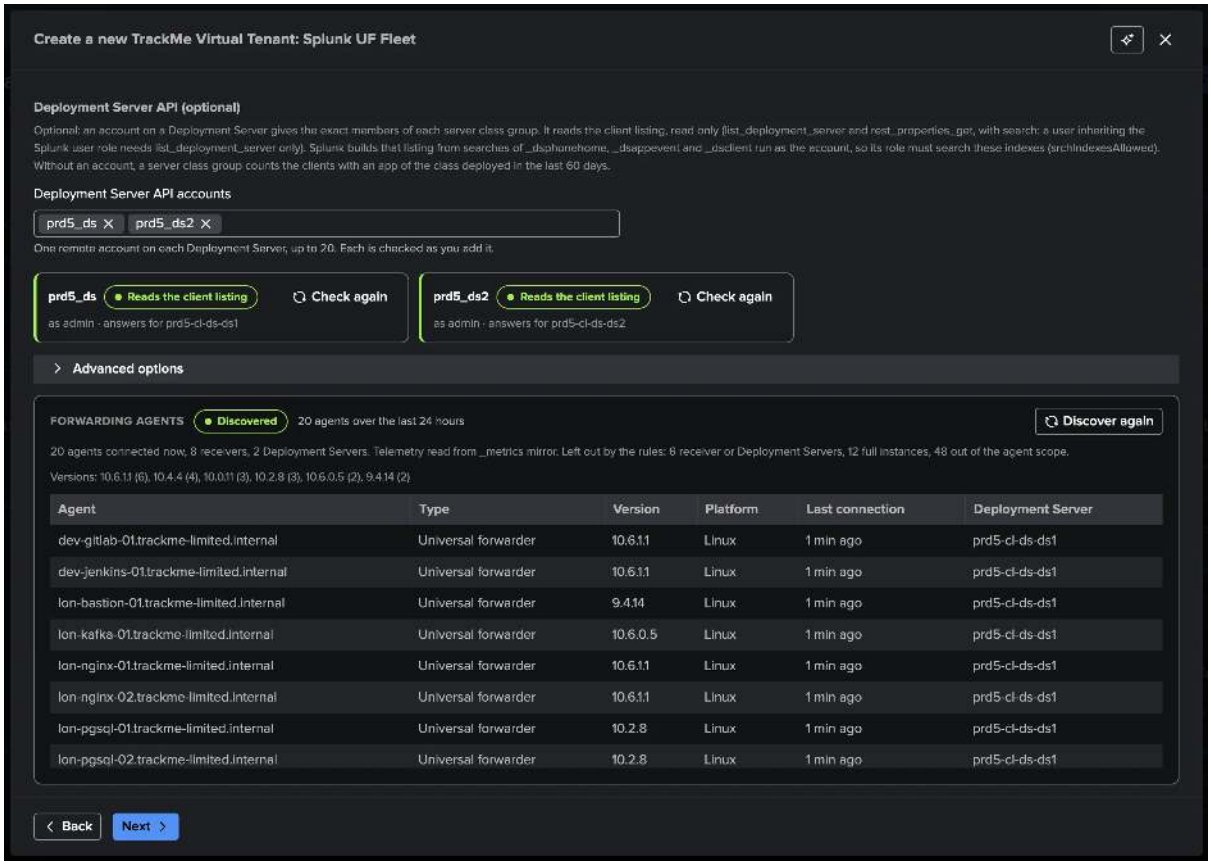
Here the two Deployment Servers form a cluster, one account on each member: each answers for its own Deployment Server. In a cluster whose members share their client listing, one member's account may list every client while the other's reads "Lists no client": that is expected too.

Advanced options (folded; opened when one of its settings needs a fix) hold the telemetry form, the three index names, the **Agent patterns** (which agents are in scope, by host name), the **Receiver patterns** (whose connection lines count), **Excluded agents** and **Excluded receivers**, the **connected window**, **Dormant after** and **Removal candidate after**. Each pattern is a token: type a host name or a pattern and press Enter (a pasted comma-separated list is split). Set the exclusions here rather than later: the first backfill (14 days) never reads what they leave out, so the new tenant's history starts clean. Receiver patterns and exclusions take `*` as their only wildcard.

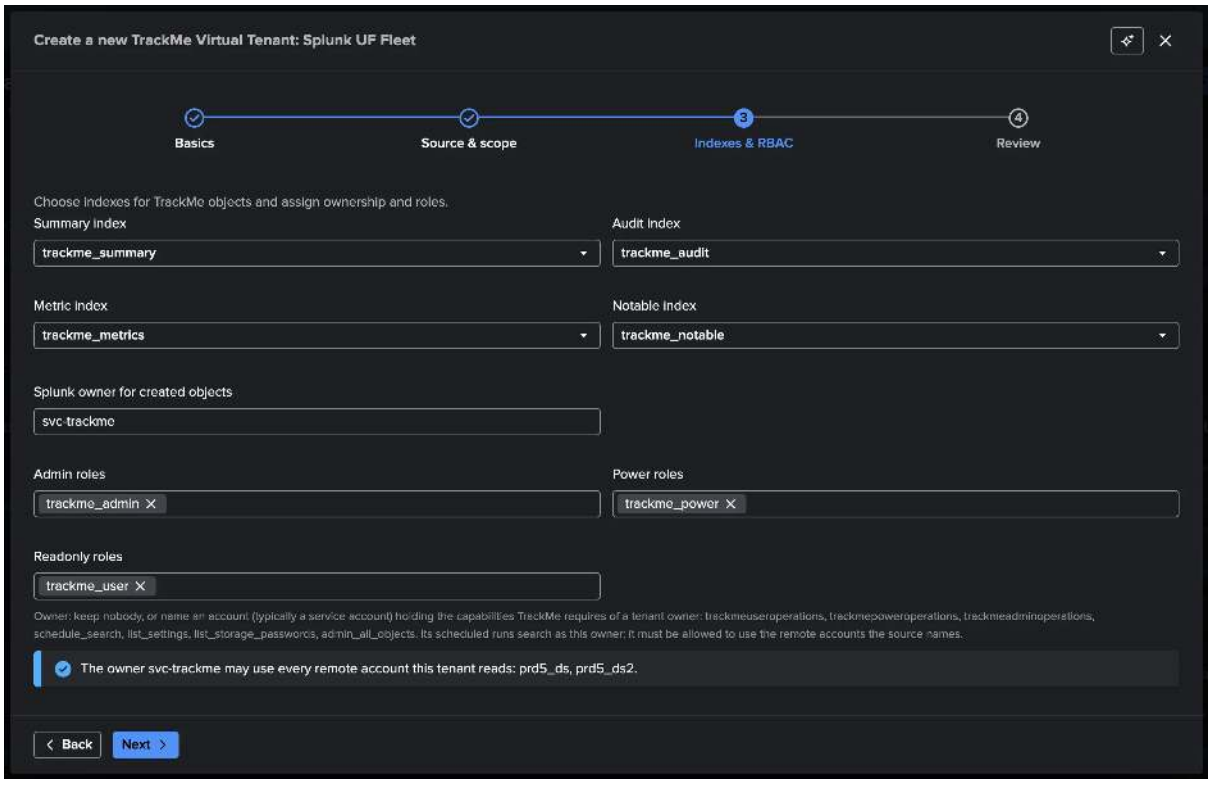
Below, the scope is the corporate domain, `*.trackme-limited.internal`, and two families of test receivers that share the same indexes are left out: `sim-*` and `bcb-*`.



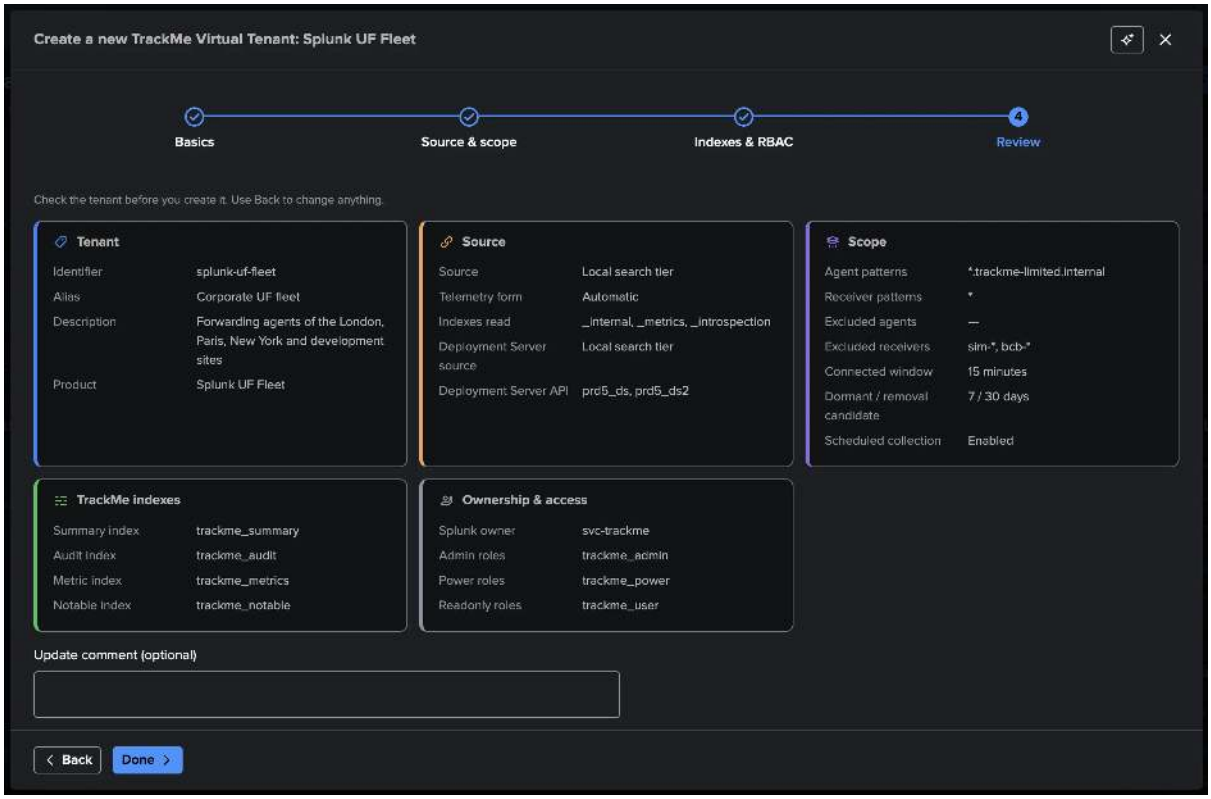
The **agent discovery** is a short preview of what the tenant would collect, with its own rules, over the last 15 minutes: how many agents sent data and to how many receivers, what the rules left out (infrastructure, full instances, out of scope), and the Deployment Servers with their clients. It runs at most two searches of 5 minutes each. It is informational and never blocks the step; the tenant's own collection reads further back and lists every agent, with its version, platform and last connection.

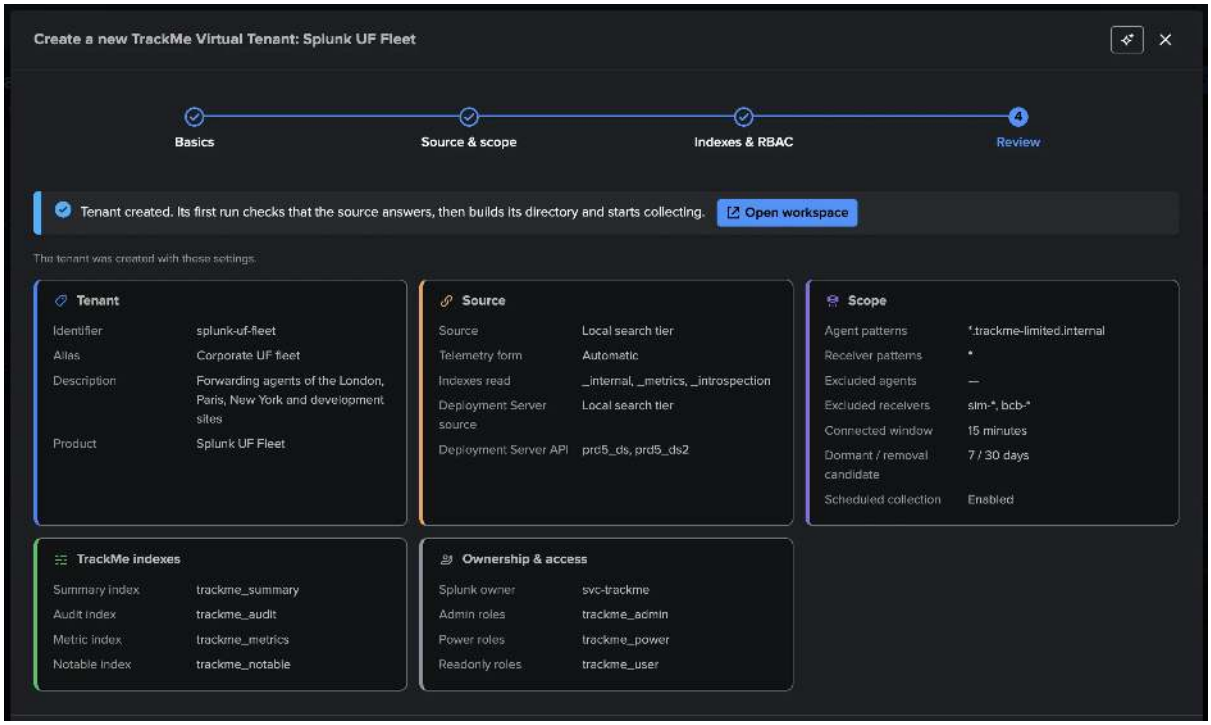


3. Indexes & RBAC — the tenant's TrackMe indexes, the owner and the roles. The wizard checks that the owner may use every remote account the tenant reads.



4. Review — then **Done**. The first live run starts at once: it checks that the source answers, then builds the directory and starts collecting. Expected agents and groups are set once the tenant exists (below).



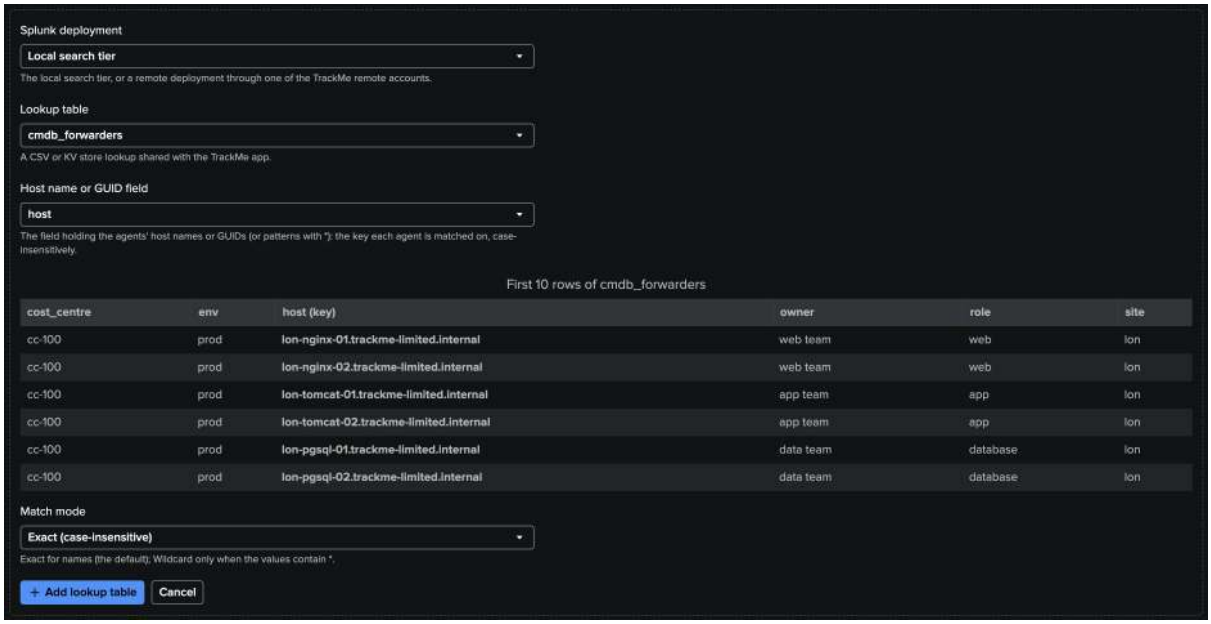


Configuring the tenant

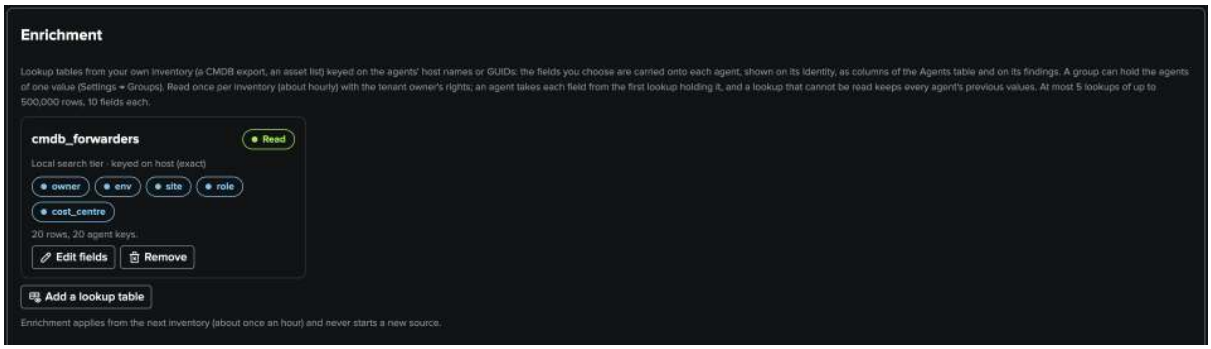
Three settings turn a list of agents into a fleet you can read: the **enrichment** from your own inventory, the **groups**, and the **expected agents** that must always stay connected. All of them are in the workspace once the tenant exists.

Enrichment

Settings → Enrichment carries fields from your own inventory (a CMDB export, an asset list, as a CSV or KV store lookup, local or through a remote account) onto each agent, keyed on the agents' host names or GUIDs (exact, or wildcard with *). Choose up to 5 lookups of up to 500,000 rows, and up to 10 fields each.



Then choose the **fields to carry**, in order: the first three are columns of the Agents table, and every field shows on the agent's Identity and on its findings (notifications, assignment and the AI Agent read them). Once the next inventory has read the lookup, its card says **Read** with its rows and the agent keys it holds.



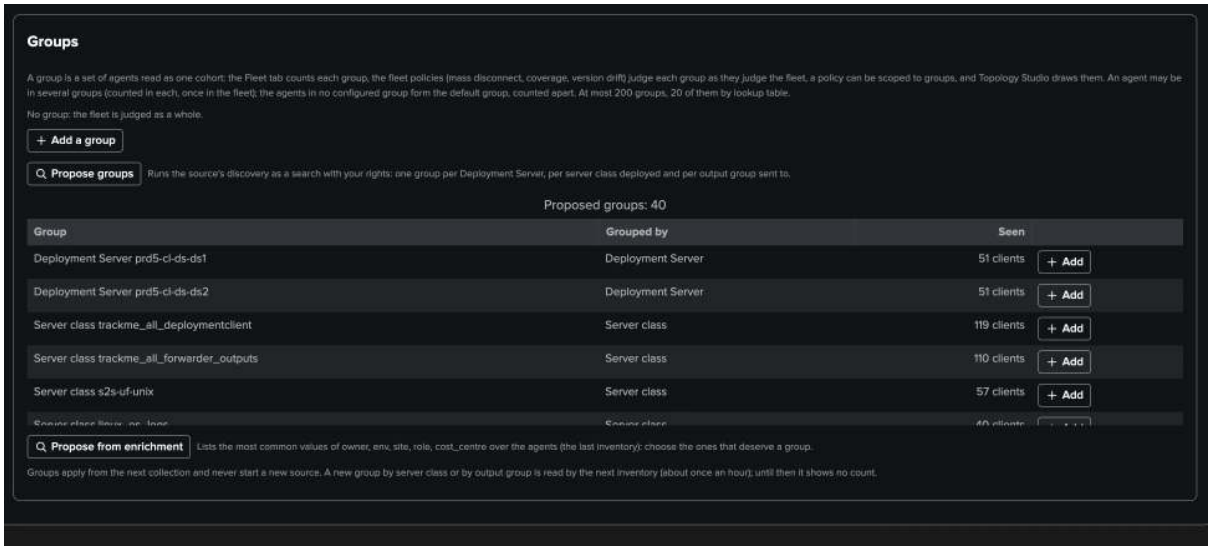
The enrichment is read once per inventory (about hourly) with the tenant owner's rights. A lookup that cannot be read, or holds more rows, keeps every agent's previous values, and its card says so. Enrichment never starts a new source.

Groups

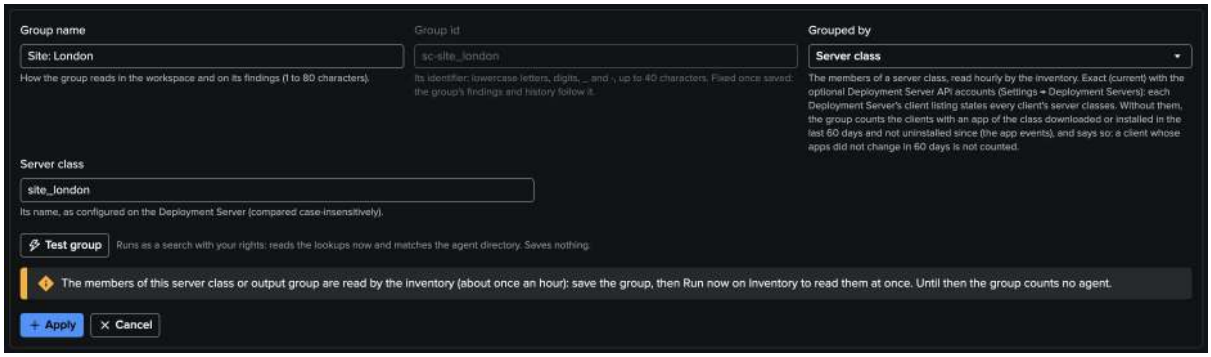
Settings → Groups reads a set of agents as one cohort: the Fleet tab counts each group, the fleet policies (Mass disconnect, Fleet coverage, Version drift) judge each group as they judge the fleet, a policy can be scoped to groups, and Topology Studio draws them. An agent may be in several groups (counted in each, once in the fleet); the agents in no configured group form the **default group**. Up to 200 groups, 20 of them by lookup table.

Group by	An agent is a member when
Deployment Server	The Deployment Server it phoned last is the value.
Server class	It is a member of the class: exact with the Deployment Server API accounts (membership *current*), else the clients with an app of the class deployed in the last 60 days (*app events*).
Lookup table	A field of a lookup holds its host name or GUID (exact, or wildcard with `*`).
Enrichment value	Its enrichment field holds the value.
Host-name pattern	Its host name matches one of the patterns (`*`, `?`).
Output group	It sent to the output group in the last hour.

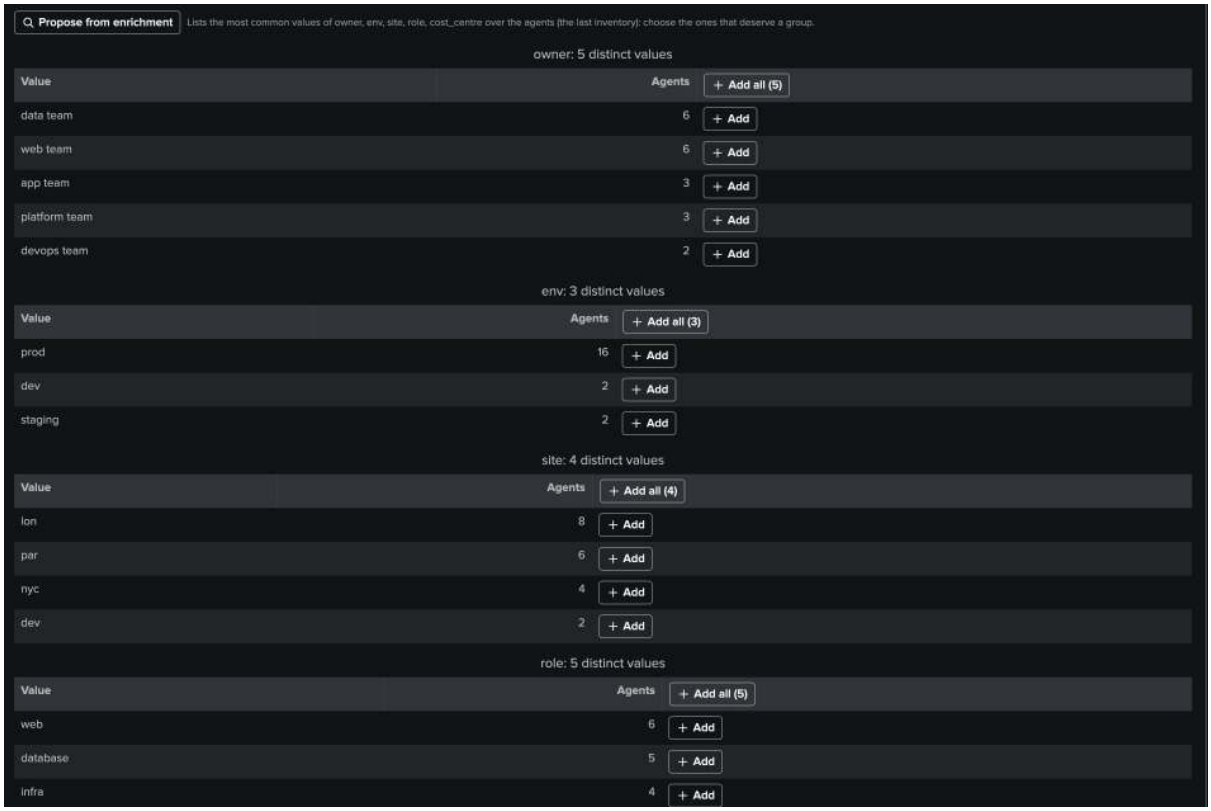
Propose groups runs its own reads with your rights (three searches: the Deployment Servers' clients, the server classes deployed in the last 60 days, the output groups sent to in the last 15 minutes) and proposes one group per Deployment Server, per server class and per output group: add the ones that deserve a group. Its counts are the Deployment Servers' own (every client of a class, whichever tenant follows it); **Test group** in the group editor shows the members this tenant counts.



A group opens in its editor: its name, its id (fixed once saved: its findings and history follow it), how it is grouped, and **Test group**, which matches the agent directory now without saving.



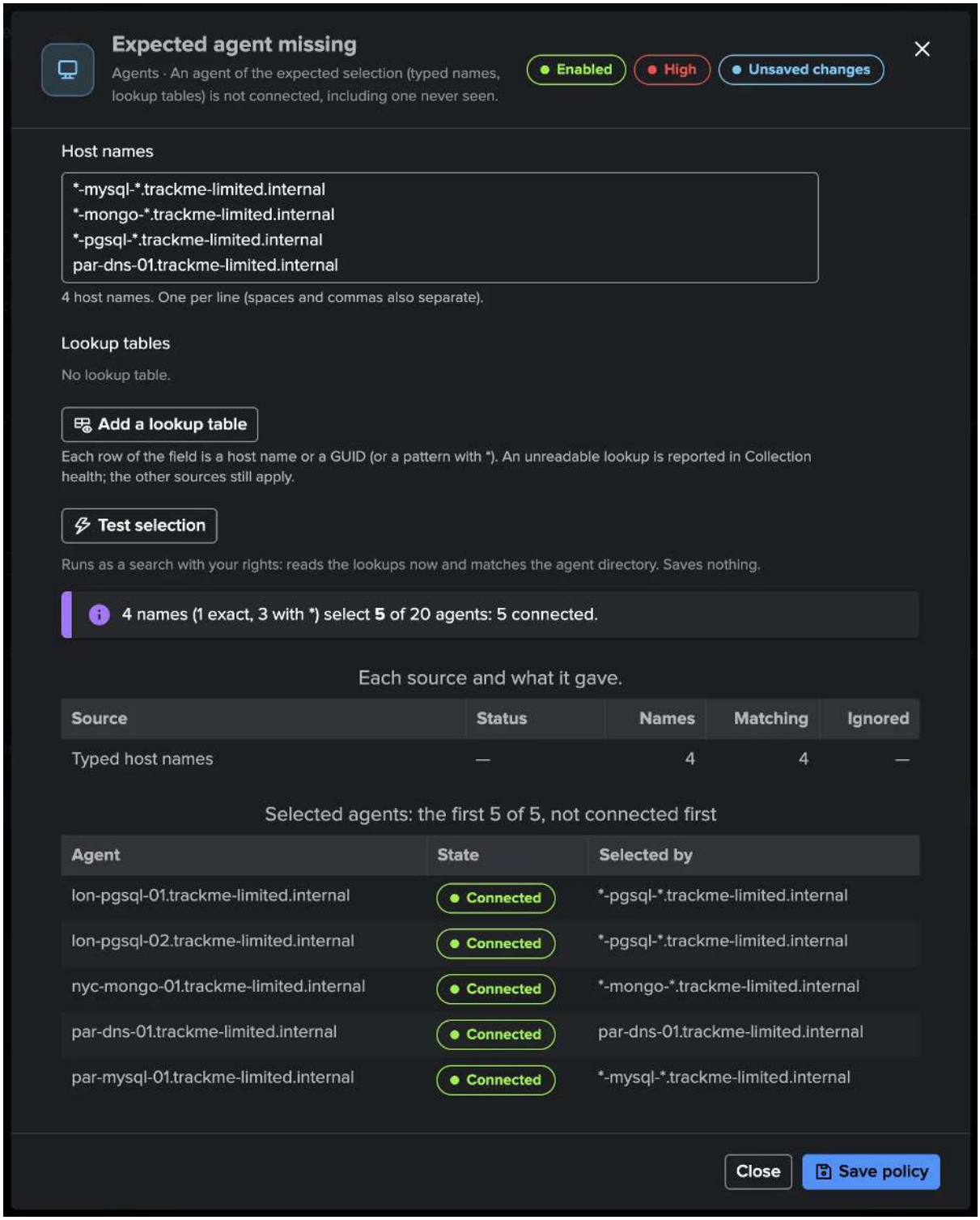
Propose from enrichment lists the most common values of each enrichment field with their agent counts: add a value, or every value of a field at once.



A new or edited group reads *pending* until the next collection computes it (a server class or output group group needs the next inventory). To see it at once, **Run now** the inventory in Collection health, then the live job. A group's membership state is **current**, **app events**, **pending** or **last known** (a read failed: the members of the last computation; such a group never recovers a finding).

Expected agents

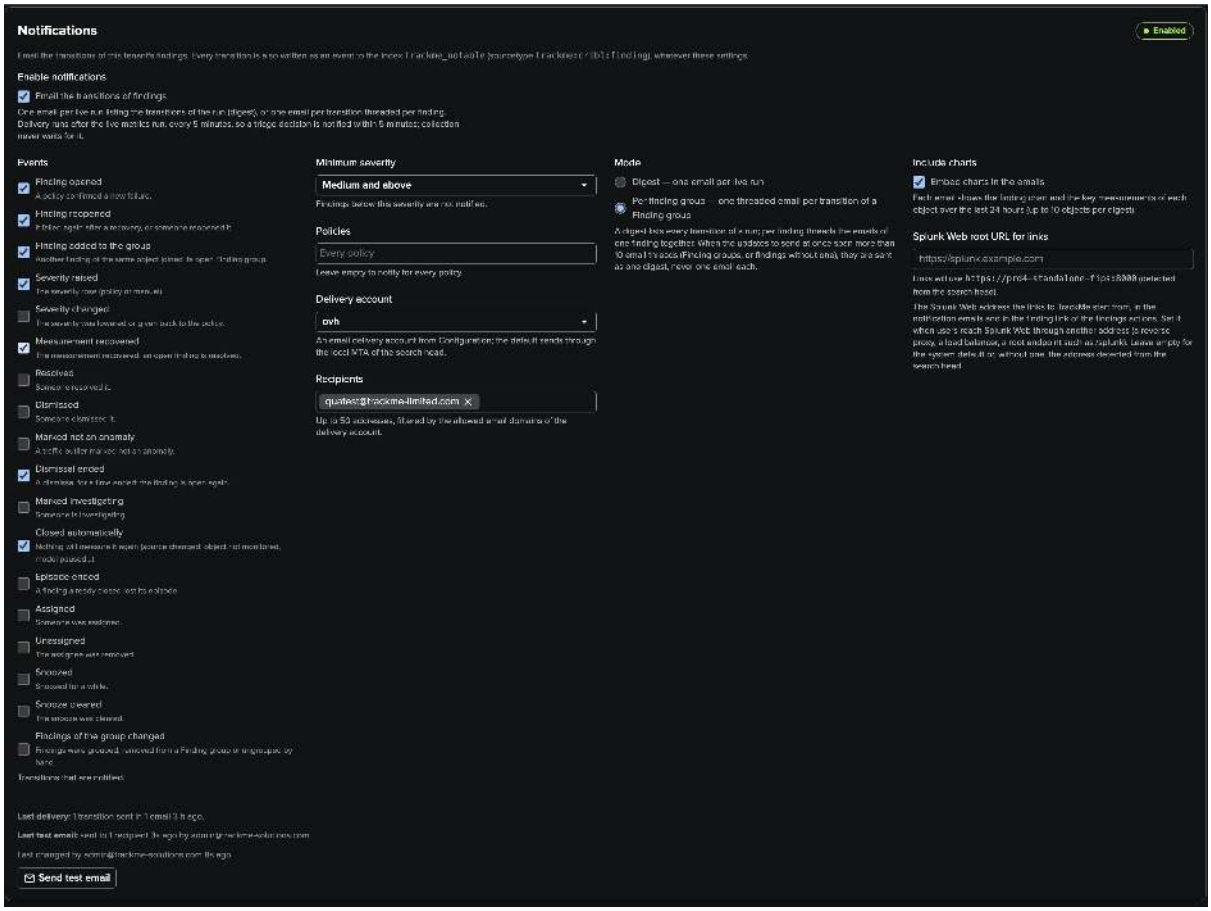
The agents that must always stay connected — the databases, the DNS, the bastions — are selected in the **Expected agent missing** policy (Policies tab): typed host names or GUIDs (`*` allowed) and up to 20 lookup tables. A name never seen is reported missing at once. **Test selection** runs as a search with your rights and shows the selected agents by state, the names never seen and the patterns selecting none; it saves nothing.



In the Agents tab, **Add to expected** adds the selected agents' host names; the agents the policy selects carry a pin. The policy's grace delays the finding when you need it.

Notifications

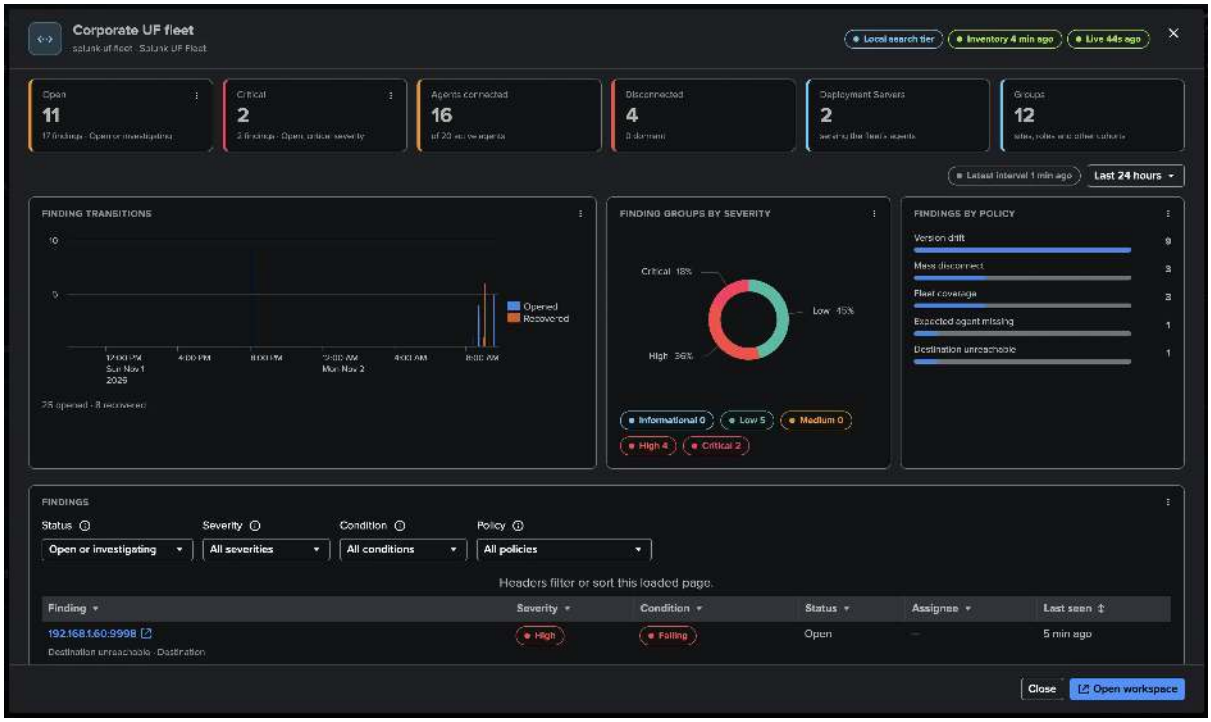
Settings → Notifications emails the transitions of the tenant's findings: the events notified, the minimum severity, the policies, the email delivery account, the recipients, and the mode — one digest per live run, or one threaded email per transition of a Finding group — with the finding's chart and key measurements embedded. In per Finding group mode, when more than 10 Finding groups have updates to send at once (agents of a whole site reconnecting, say), they go as one digest instead.



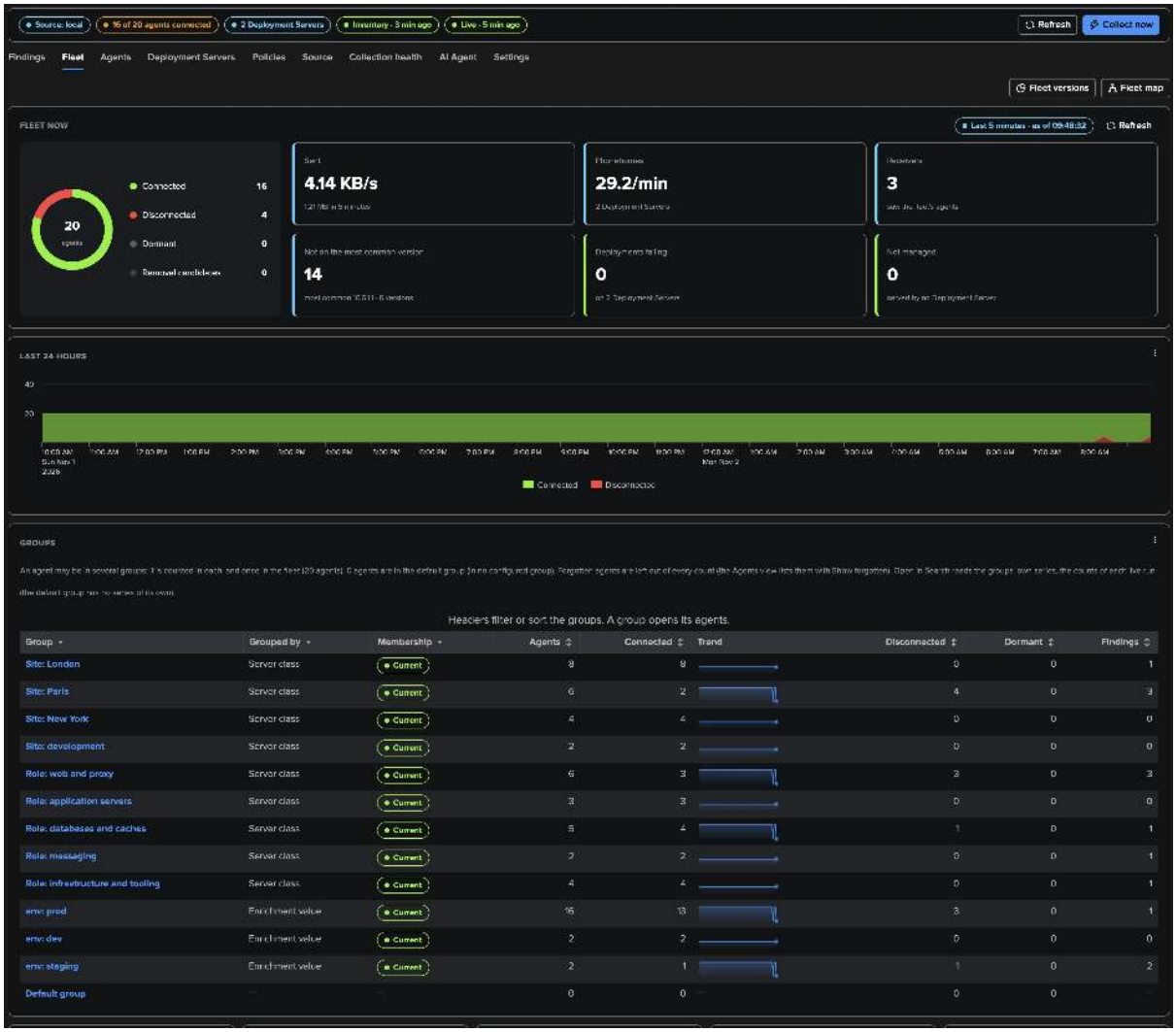
Every transition is also written as an event to the tenant's notable index, whatever these settings. See *Findings, Finding groups and findings actions*.

The workspace

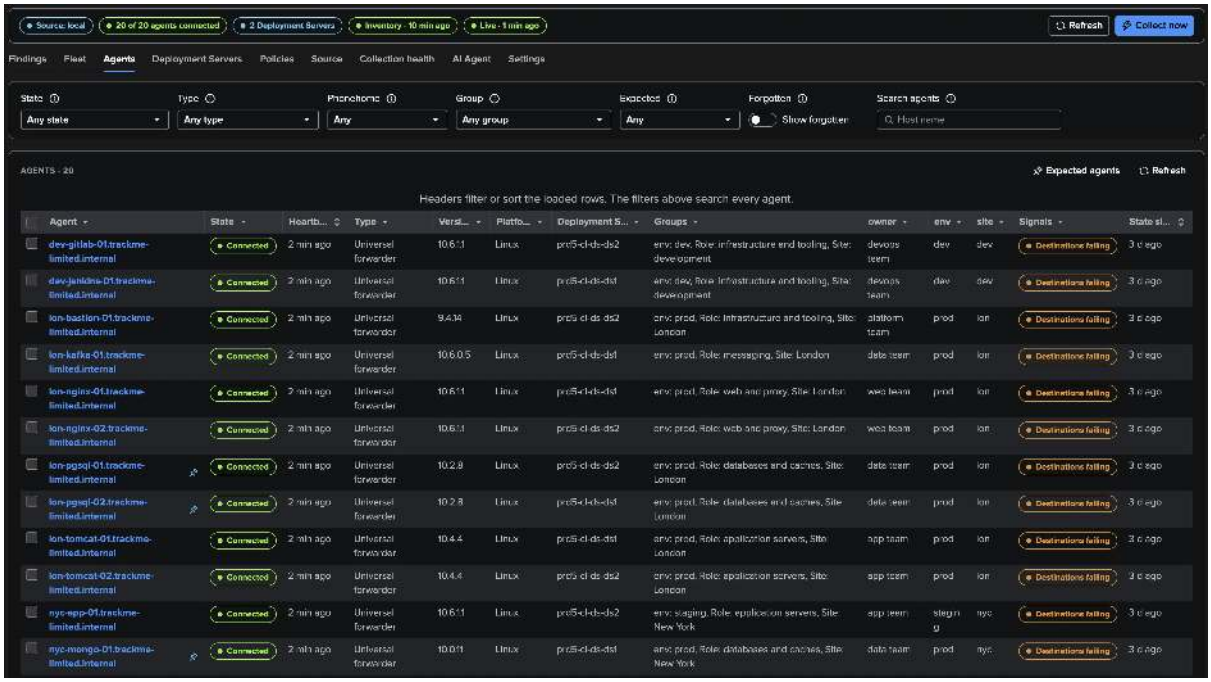
The preview. On the *Virtual Tenants* page, a double-click on the tenant card opens its preview before the workspace: the source and the collection pills, the **Open** and **Critical** figures, **Agents connected**, **Disconnected** (with the dormant agents), **Deployment Servers** and **Groups**, the finding transitions and severities of the last 24 hours, the findings by policy and the findings table, read-only. A finding opens in the workspace; **Open workspace** opens the tenant. Before the first live run, the figures read as dashes.



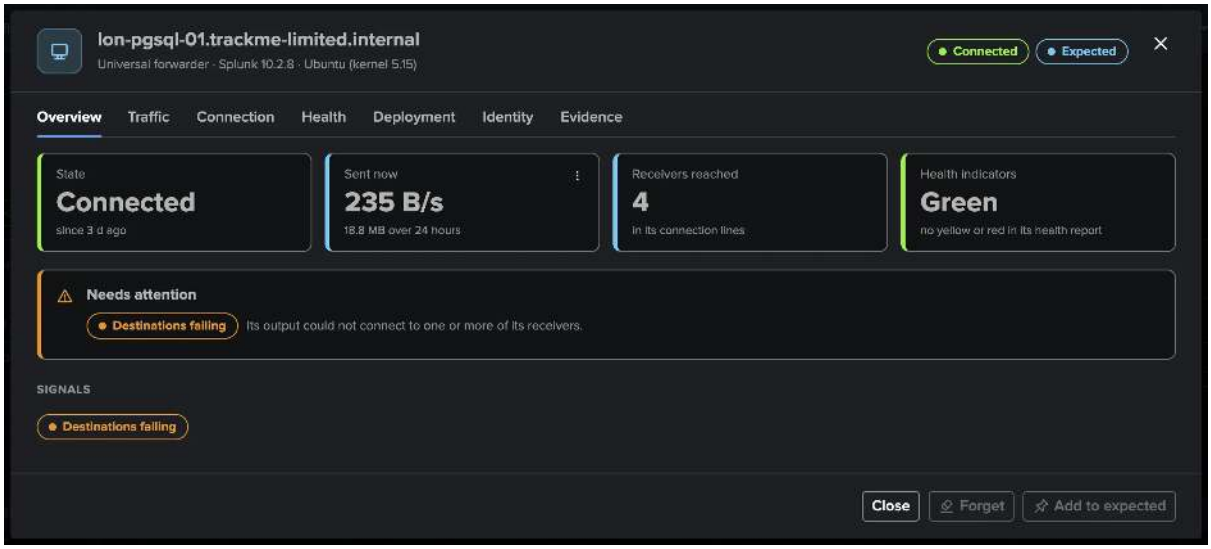
The workspace is the Tenant Home of the tenant. A **status strip** sits above the tabs: the source, "N of M agents connected" (opens the disconnected agents), "N dormant", "N Deployment Servers", the **Inventory** and **Live** job pills (each opens Collection health), **Refresh** and **Collect now** (runs the live job).

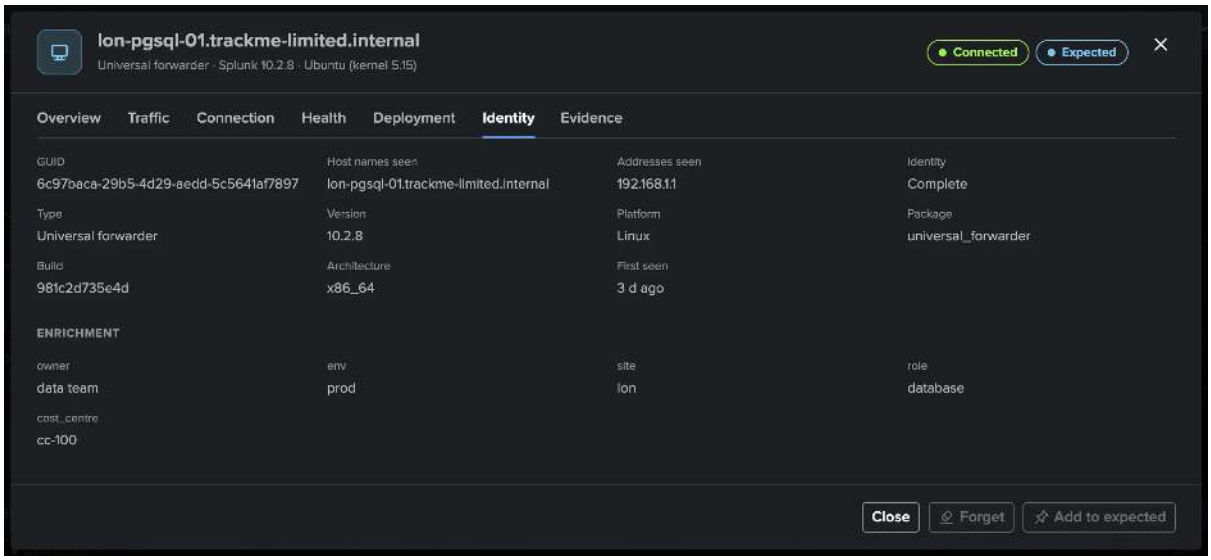

Source: local
16 of 20 agents connected
2 Deployment Servers
Inventory - 3 min ago
Live - 5 min ago
Refresh
Collect now
Findings
Fleet
Agents
Deployment Servers
Policies
Source
Collection health
AI Agent
Settings
Fleet versions
Fleet map
FLEET NOW
Refresh
20
Connected 16
Disconnected 4
Dormant 0
Removal candidates 0
4.14 KB/s
29.2/min
3
Not on the most common version
14
Deployments failing
0
Not managed
0
LAST 24 HOURS
Connected
Disconnected
GROUPS
Headers filter or sort the groups. A group opens its agents.
Group
Grouped by
Membership
Agents
Connected
Trend
Disconnected
Dormant
Findings
Site: London
Site: Paris
Site: New York
Site: development
Role: web and proxy
Role: application servers
Role: databases and caches
Role: messaging
Role: infrastructure and tooling
env: prod
env: dev
env: staging
Default group

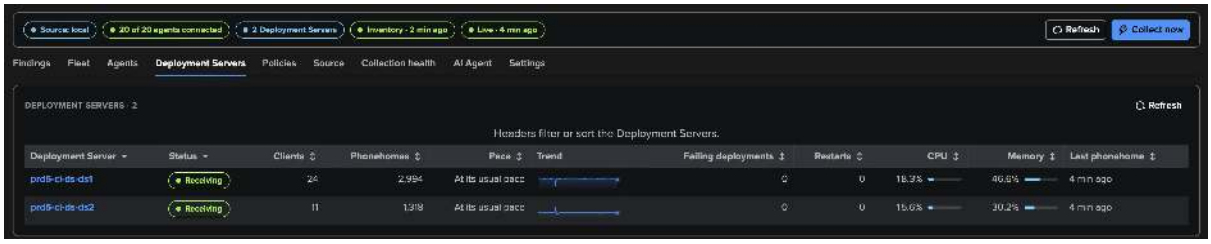
Tab	What it shows
Findings	The findings centre shared by every dedicated tenant (see *Findings, Finding groups and findings actions*). Families: Agents, Agent groups, Fleet, Deployment Servers, Destinations. **Open resource** opens the agent, its Deployment Server or the Fleet tab in place.
Fleet	**Fleet now**: the agents by state (Connected, Disconnected, Dormant, Removal candidates, each opening the Agents tab on that state) and the agents connected by heartbeat only, beside two rows of tiles: the **Last 5 minutes** (sent per second, phonehomes per minute, the receivers that saw the fleet), then the fleet's management (**Not on the most common version**, **Deployments failing** across the Deployment Servers, **Not managed**: served by no Deployment Server). **Last 24 hours**: the connected and disconnected agents over time; breakdowns (disconnection ages, versions, platforms, agent types, agents per Deployment Server, each opening the Agents tab filtered); the **Groups** panel (each group's membership state, counts, a 24-hour line and its open cohort findings); the recently disconnected agents; and the **Fleet map**, a generated topology of the fleet, its groups and Deployment Servers.
Agents	Every agent of the fleet, paged on the server, with filters on state, type, phonehome, group, expected and host name; forgotten agents are listed only with **Show forgotten** (off by default). Columns: agent, state, heartbeat, type, version, platform, Deployment Server, groups, the first three enrichment fields, signals and state since. **Signal pills** open the facts and their evidence. **Forget** and **Add to expected** act on the selection; the footer's **Rows per page** (25, 50 or 100) is your own preference for the tenant.
Deployment Servers	Every Deployment Server: clients, phonehomes, pace against its usual, failing deployments, restarts, CPU, memory, last phonehome.
Policies	The policy catalogue, custom policies, and **How it is evaluated** for every policy.
Source	The search tier, the qualification, the telemetry form, the receivers seen, what the rules left out, and the collection runs with their outcome and receipt.
Collection health, **AI Agent**, **Settings**	See below.



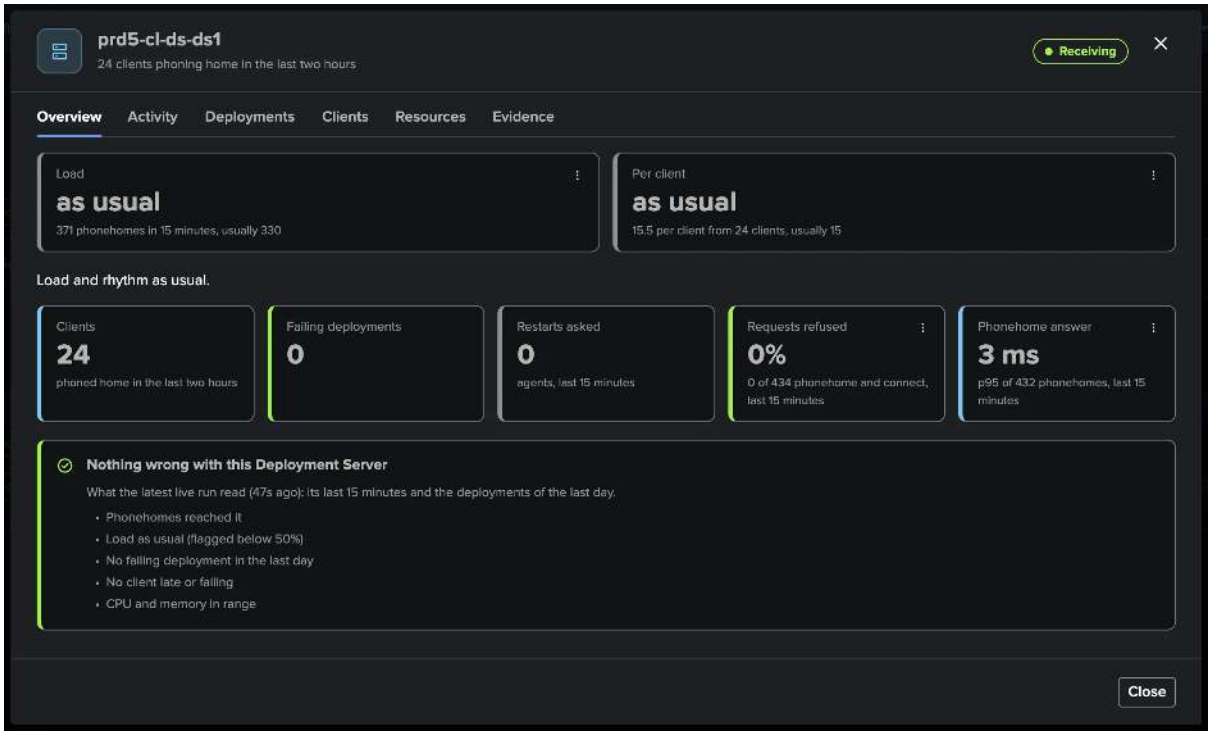
The agent modal has tabs Overview (state, sent now and over 24 hours, receivers reached, health, and what needs attention with its open findings), Traffic (data sent per second and the receivers reached, split per receiver), Connection, Health (its own logs' signals), Deployment (its Deployment Server, phonehome cadence, failing deployments, app events), **Identity** (with its **Enrichment**) and Evidence. **Forget** and **Add to expected** are in its footer, disabled with the reason when they do not apply: a connected agent cannot be forgotten, and an agent the Expected agent missing policy already selects shows *Expected* and needs no adding.







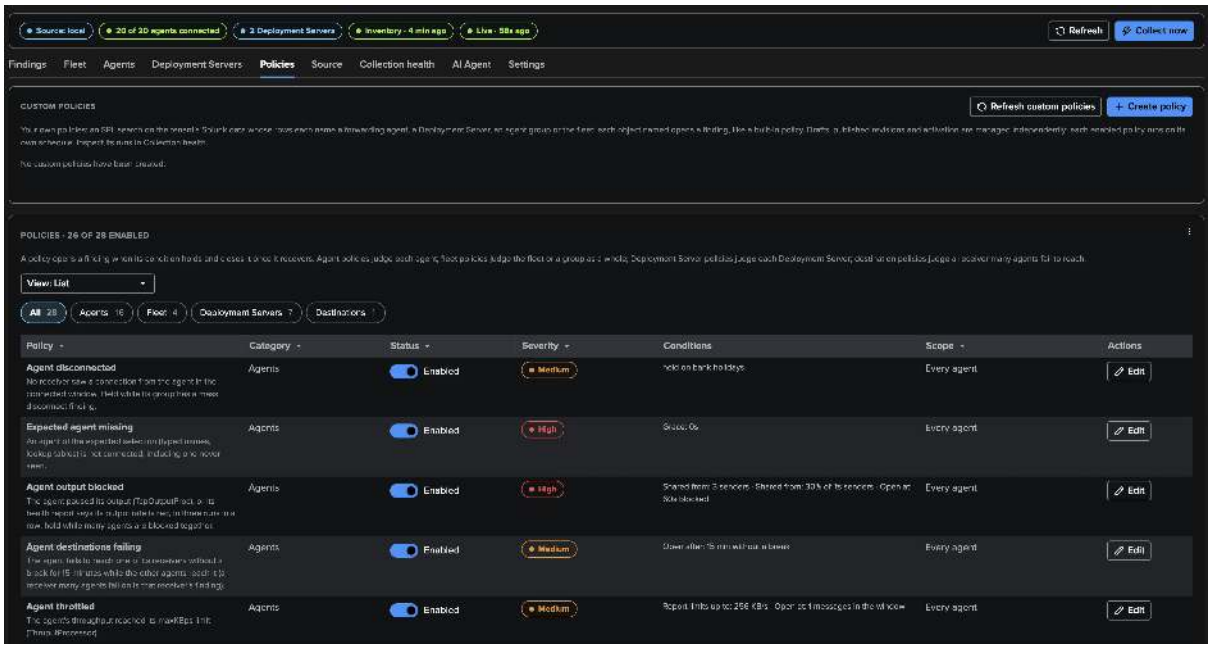
The Deployment Server modal has tabs Overview (clients, failing deployments, restarts asked, the **Load** and **Per client** readings, the refused requests and the 95th percentile of the phonehome answer), Activity (phonehomes per 15 minutes against the usual at that hour, the clients phoning home over time), Deployments, Clients, Resources and Evidence.

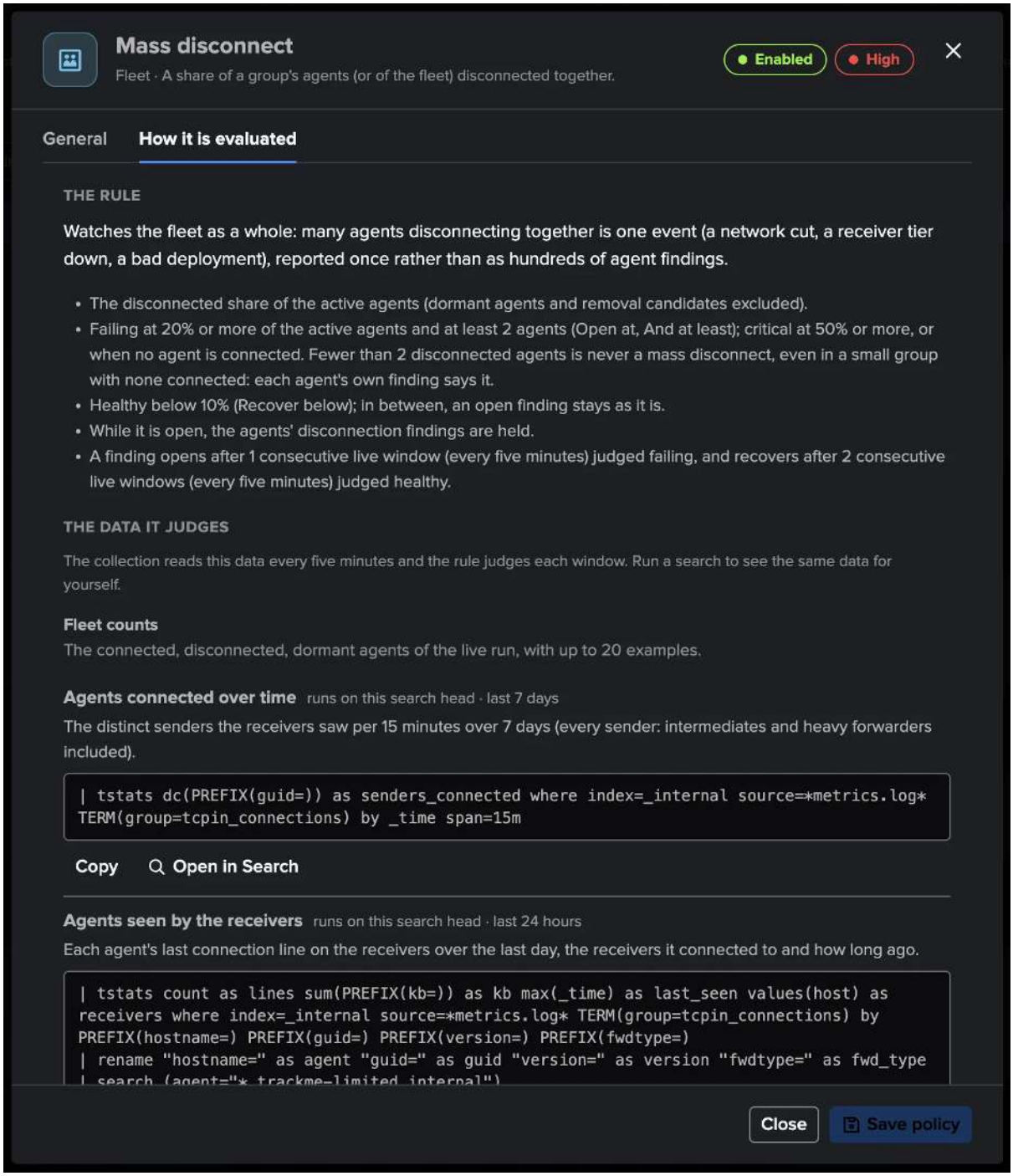


Load and Per client read together: *Load* compares the phonehomes with the usual for the hour (what the Deployment Server carries), *Per client* compares the phonehomes per client with the usual per client (the clients' rhythm). Load up with a stable rhythm means more clients; both up, clients phoning home more often; load down with a stable rhythm, clients that left.

Policies

Twenty-eight built-in policies cover the agents, the fleet and its groups, the Deployment Servers and the destinations; every policy is evaluated. Each has its severity, the runs it needs to open and to recover, a scope (**Groups**: empty means every agent, the fleet and every group; chosen groups judge those groups and the agents they hold, never the fleet), a bank holiday hold, and a **How it is evaluated** tab: the rule as configured, the data it reads with ready-to-run searches on the tenant's own source, how to investigate and how to resolve.





The most useful policies to know first:

- **Agent disconnected** (medium) — no receiver saw the agent in the connected window. Held while the fleet or one of its groups has a Mass disconnect; becoming dormant closes it as dormant.
- **Expected agent missing** (high) — an agent of the expected selection is not connected, a name never seen included. Dormancy never closes it.
- **Mass disconnect** (high, critical at 50 % or none connected) — 20 % or more of a group's (or the fleet's) active agents, and at least 2, disconnected together: one finding holding the agents' own findings.
- **Agent output blocked** (high) and **Agent destinations failing** (medium) — from the agent's own logs: its output paused, or one receiver unreachable for 15 minutes while the other agents reach it.
- **App deployment failed** (high) and **Deployment failures** (high) — an app deployed to an agent failed to download or install; app downloads or installs failing on a Deployment Server.

- **Deployment Server silent** (high) and **Phonehome drop** (high) — a Deployment Server with clients receives no phonehome, or far fewer than usual at this hour.
- **Agent version unsupported** (off by default) — the agent runs below the minimum versions you set; it decides nothing until a minimum is set.

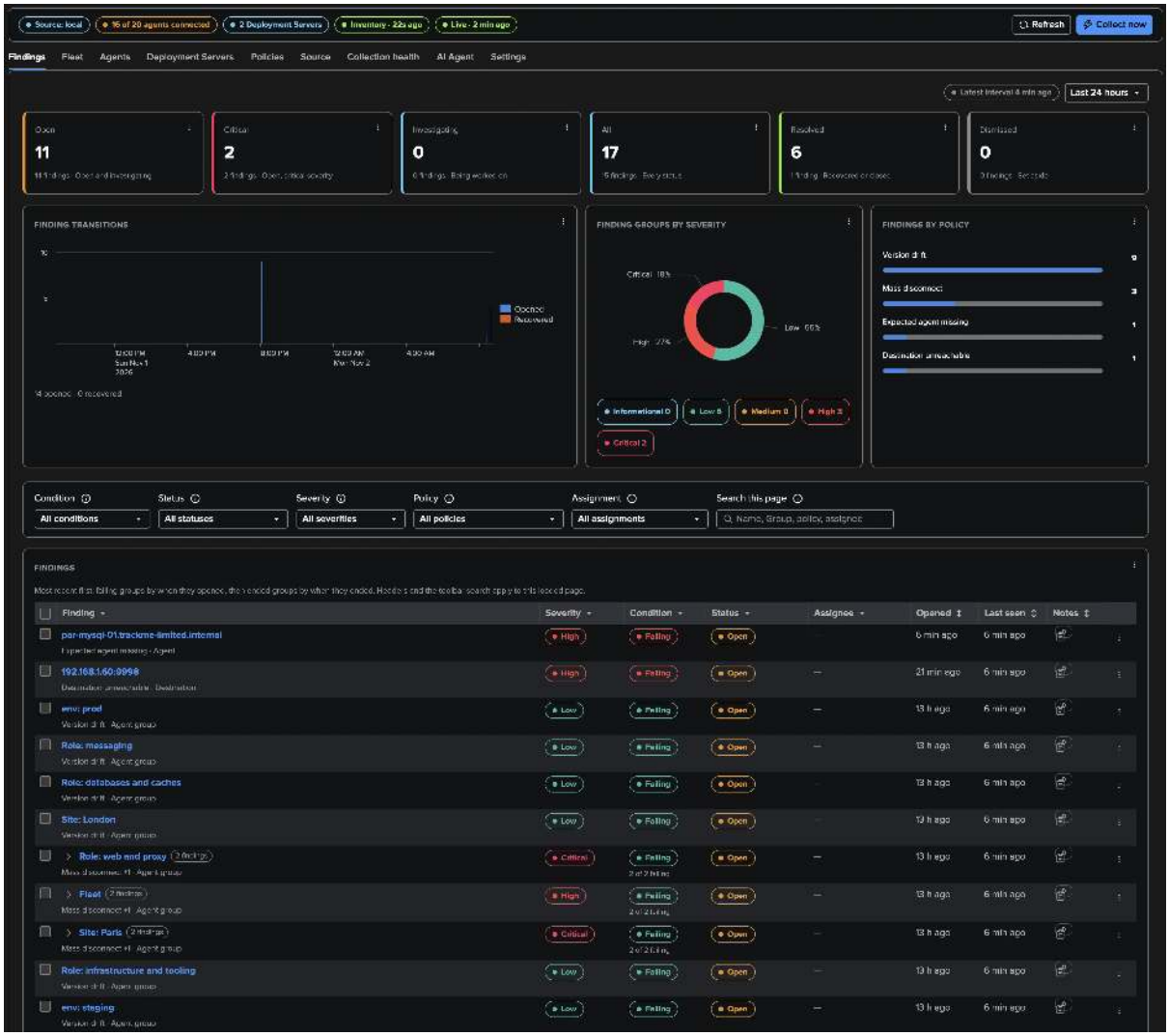
Every policy, its defaults and its settings are in *Built-in policies*.

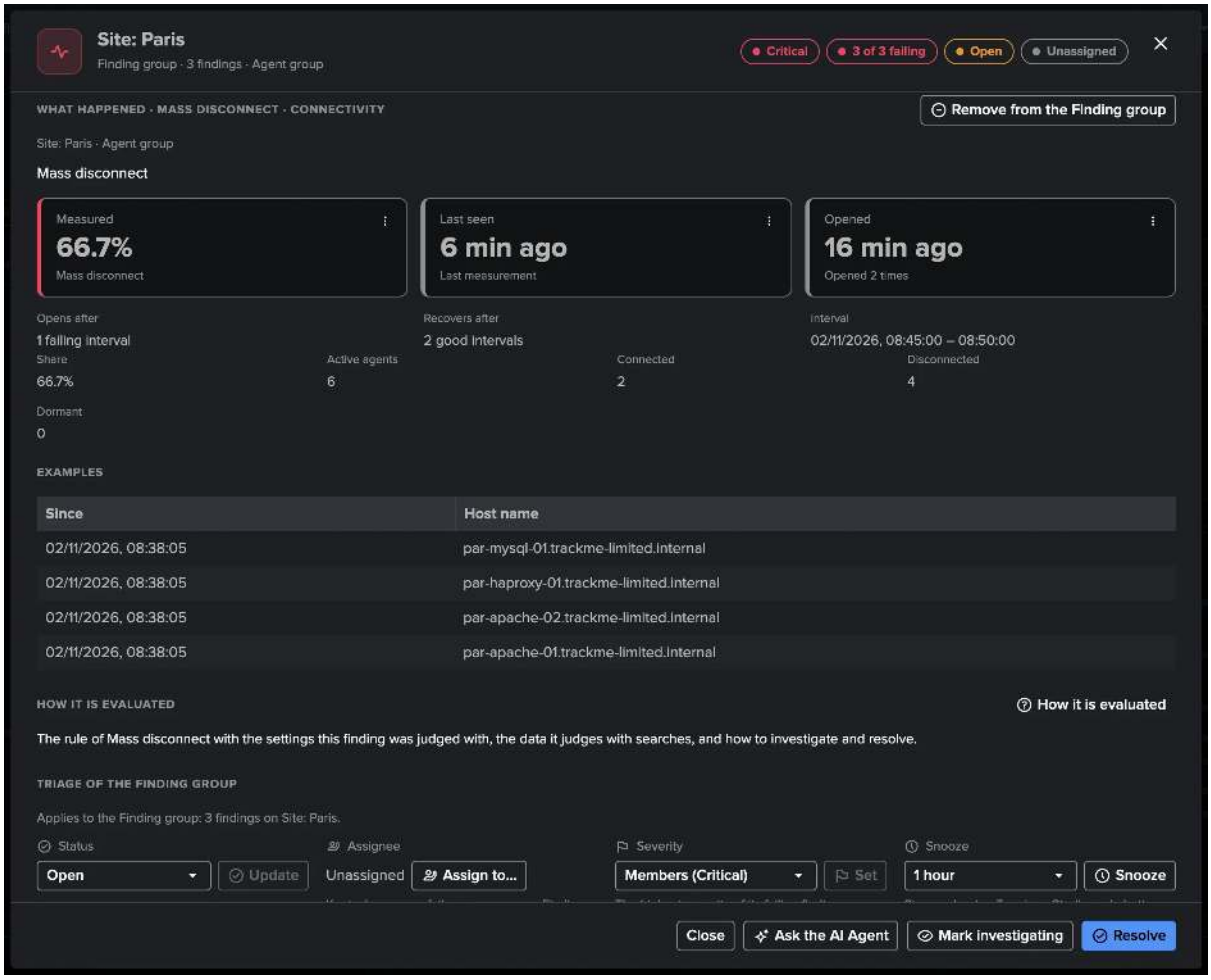
Custom policies. Above the catalogue, TrackMe administrators of the tenant write their own SPL: each result row names an agent (`agent`), a Deployment Server (`ds`), a group (`group`) or the fleet, and opens a finding like a built-in policy. See *Custom policies*.

Findings in practice

A finding of a Splunk UF Fleet tenant reads like every finding of a dedicated tenant (see *Findings, Finding groups and findings actions*): one row per Finding group, the decision history, the policy's measurements, triage (investigate, snooze, dismiss, resolve), assignment, notes, findings actions and **Ask the AI Agent**. **Open resource** opens the agent, its Deployment Server or the Fleet tab in place.

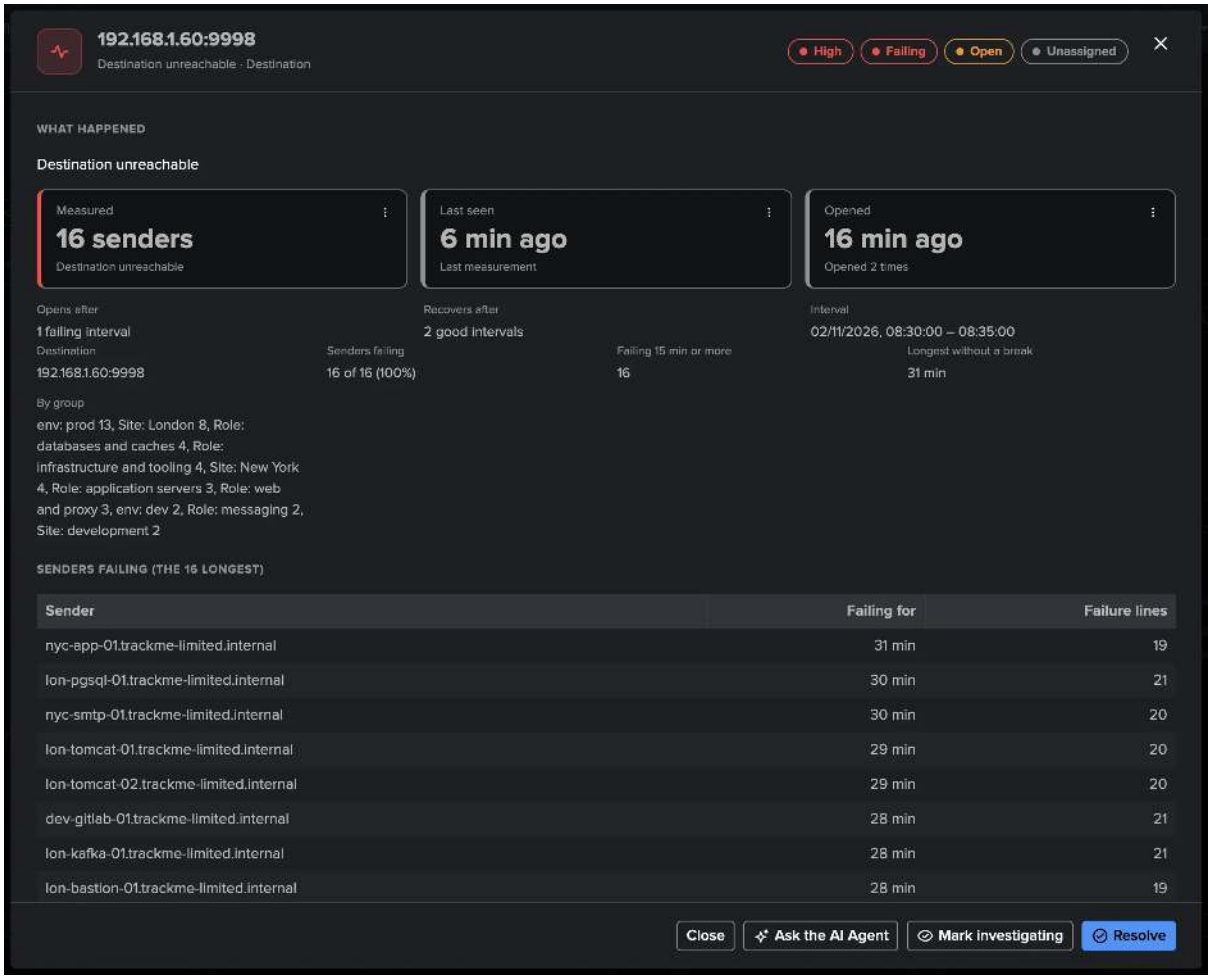
A site goes dark. Four of the six agents of the Paris site were stopped together. Mass disconnect opened on each cohort they are a large share of: the **Site: Paris** group (66.7 %, critical from 50 %), the **Role: web and proxy** group (critical) and the fleet (20 %, high). Each joined the Finding group of its cohort, beside the cohort's Version drift finding. Their own Agent disconnected findings were held, never opened one by one. par-mysql-01 is an expected agent: Expected agent missing opened for it whatever the cohort. At the same time a heavy forwarder of the receiving tier was stopped for the Splunk S2S walkthrough: the agents that send to it opened one **Destination unreachable** finding on its receiving port, never one per agent.





When the agents reconnect, a mass disconnect recovers once the share falls below its recovery percentage for two runs; an agent still disconnected then opens its own finding.

A destination many agents fail to reach is one finding on the receiver, never one per agent: its measurement is the number of agents failing to reach it for 15 minutes or more, with every agent failing, how long, and the groups they belong to.



Forget (Agents tab and agent modal, the tenant's power and admin roles) removes from the fleet an agent that is not connected — disconnected, dormant or a removal candidate — so an instance replaced or decommissioned is retired at once, without waiting for the dormancy delay: 1 to 100 agents per request, never an agent the enabled Expected agent missing policy selects. A connected agent, by its connection lines or its heartbeat, is refused: it would come back on its next line. Forget resolves the agent's open findings first, and the agent leaves the counts at once: the Fleet tab, the groups and the status strip no longer count it, without waiting for the next collection. Forgotten agents are listed in the Agents tab only with **Show forgotten**, without signals: the live facts of a host name belong to the instance that carries it now.

A rebuilt host's older identity is forgotten by TrackMe. A host rebuilt, re-imaged or reinstalled under its name comes back with a new GUID: once a receiver sees that new identity connected and the older one no longer connects, TrackMe forgets the older identity as Forget would. It leaves every count, its findings close, and **Show forgotten** lists it as *Replaced by a new identity*; its Identity tab names the new GUID. It comes back only if its own GUID connects again. The Expected agent missing policy follows the host by its name: the host is expected under its new identity. TrackMe never forgets an older identity the policy names by its own GUID, and decides only on a complete read of the receivers' connection lines.

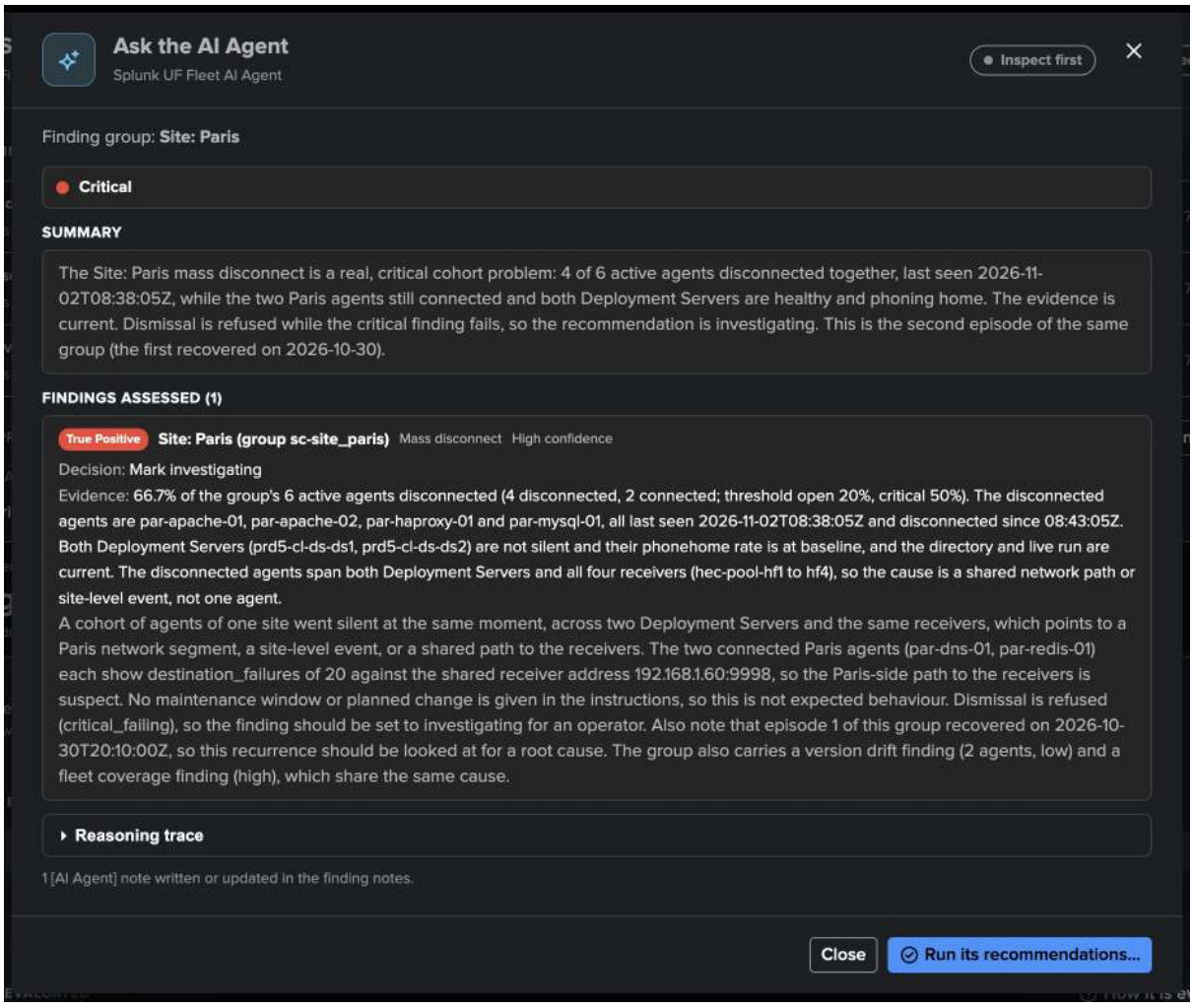
Topology Studio

Splunk UF Fleet objects are first-class in Topology Studio: the fleet, its groups, the Deployment Servers and the agents, in the entity picker and as dynamic groups (agents by state, by Deployment Server, by host name, with open findings); agents are drawn as groups past 50. Their status comes from their findings first; without one, a connected agent is green, a disconnected agent a quiet grey (an agent's colour comes from its own finding only), a Deployment Server measured green. The templates **Fleet by group**, **Deployment Servers** and **Problems wall** build a view in one click, and the AI Topology Builder knows the tenant ("Show my fleet by Deployment Server"). When a Splunk S2S tenant reads the same source, the Fleet map and the Fleet by group template draw its receiving nodes linked from the fleet. See *Topology Studio*.

The Splunk UF Fleet AI Agent

The **Splunk UF Fleet AI Agent** reviews findings on the shared findings AI Agent framework (see *AI Agents for findings*). It reads what TrackMe holds about the finding — the agent's directory row (state, identity, version, receivers, Deployment Server, troubles and deployment facts), its Deployment Server with its baseline and clients, the fleet summary, collection health, the Finding group and past decisions — and may read further through its tools (the fleet overview, agents, Deployment Servers, log evidence, trends, the last 5 minutes). It writes one `[AI Agent]` note per review and, in act mode, applies only what the tenant allows: marking a finding investigating (allowed by default), a temporary dismissal of at most 7 days (allowed), and resolving after a measured recovery (not allowed by default). It never Forgets an agent and never adds one to the expected agents: it recommends, an operator decides. It never changes anything on a Splunk instance.

Scheduled triage is off by default (**Settings → AI Agent**, or *Manage AI Agents*): hourly, up to 10 findings per run. It needs the AI Assistant enabled and an LLM provider; a scheduled run works with a least-privilege tenant owner, not with `nobody`.



The Tenant Home **AI Assistant** knows the workspace, its policies, the fleet summary and the open findings: ask "Which agents are disconnected or dormant, and do they share a group or a Deployment Server?".

Collection health

The **Collection health** tab shows whether the tenant collects as configured: the **Status** view lists every problem with its meaning, source, causes and steps; **Trackers** shows each job's last run with the searches it sent, and **Run now**; **Data & storage** and **Troubleshoot & help** complete it. The same assessment raises the Configuration Guardian check `cribl_collection_health`. Every problem code is listed in *Splunk UF Fleet reference*.

Troubleshooting

Why does a heavy forwarder not appear as an agent? Full instances are left out by default (Settings → Source, *Full instances*); the Source tab counts them under *Left out*.

Hundreds of agents are disconnected, but there is only one finding. A mass disconnection is one cohort finding on the fleet or the group; the agents' own findings are held while it is open, and open individually once it clears.

An agent stopped a week ago and its finding closed. It became dormant (7 days): the disconnection closes as dormant, not as a recovery. At 30 days without a connection it becomes a removal candidate (Decommission candidate). Expected agents never close on dormancy.

A server class group counts fewer agents than the Deployment Server shows. Without a Deployment Server API account, the group counts the clients with an app of the class deployed in the last 60 days only. Add one account per Deployment Server in Settings → Deployment Servers.

The wizard (or Settings) says the owner may not use a remote account. The tenant owner shares none of the account's roles: add one of the owner's roles to the account (*Configuration → Remote accounts*), or choose another owner.

Splunk Cloud: the Deployment Servers tab is empty. Point the Deployment Server source at a remote account on the Deployment Server itself, through a TrackMe Broker when the Deployment Server is on premises.

Phonehome drop reads three times the usual, but nothing is wrong. The usual is a 7-day median per hour; a fleet that grew during the week reads above its usual until the median catches up. Only a drop is judged.

What the Splunk UF Fleet tenant does not do

- It does not judge the receiving side (receivers, intermediates, pools): that is *Splunk S2S*.
- It does not judge the inputs of the forwarders used as collection points: that is *Splunk Forwarder Collectors*.
- It shows the CPU and memory of a Deployment Server, never judges them.
- Agents behind a non-Splunk hop (Cribl Stream, for example) are not seen by the receivers as themselves: with the heartbeat fallback they read *connected (heartbeat)*. Agents sending S2S over HTTP are not counted.
- An agent whose `inputs.conf` sets a host name the receivers never see gets no trouble facts.
- A health indicator turns green up to 35 minutes late (Splunk's report period).
- The phonehome baseline needs 3 days of history at that hour; `_dsphonehome` keeps 7 days by default.
- It never removes an agent on its own, never changes anything on an agent or a Deployment Server.

Next: the reference

Every policy with its defaults and settings, the custom policy subjects, the jobs, the settings, the Collection health codes, the commands and the permissions are in *Splunk UF Fleet reference*.

8.3.24 Splunk UF Fleet reference

The reference of the Splunk UF Fleet tenant: every built-in policy with its defaults and settings, custom policies, the collection jobs, the settings, Collection health, the AI Agent, the indexed finding events, the permissions and the commands. The guide is *Splunk UF Fleet*.

Built-in policies

Twenty-eight policies in four categories. Every policy is evaluated on each live window (five minutes). Common settings of every policy:

- **Enabled** and **Severity** (Informational, Low, Medium, High, Critical; a few policies raise it, as their row says).
- **Checks to open** and **Checks to recover**: consecutive failing, then healthy, live windows.
- **Scope** (**Groups**): empty means every agent, the fleet and every group; chosen groups judge those groups and the agents they hold, never the fleet (an expected agent never seen is judged whatever the scope).

- **During bank holidays**: keep opening findings, or hold (Agent disconnected, Mass disconnect, Fleet coverage and Phonehome drop hold by default; see *Bank holidays*). Expected agent missing keeps opening: those agents were selected as always connected.

General rules:

- An incomplete read may open a finding, never close one; absence of evidence never recovers.
- A finding not evaluated for 24 hours closes as not observed.
- At most 250 new agent findings open per run (the rest the next runs).
- The troubles read from the agents' own logs (output, destinations, throttle, latency, input errors, restarts, handshake) are decided for a **connected** agent only, on at most 500 agents per run (the worst first).

Agents

Policy	What it detects	Default	Severity
Agent disconnected	No receiver saw a connection from the agent in the connected window. Held while the fleet or one of its groups has a Mass disconnect; becoming dormant closes it as dormant.	On	Medium
Expected agent missing	An agent of the expected selection (typed names, lookup tables) is not connected, including a name never seen. Dormancy never closes it.	On	High
Agent output blocked	The agent paused its output (`TcpOutputProc`) for at least the blocked seconds in the last 15 minutes, or its health report says its output rate is red. Held while 30 % or more of the connected agents (and at least 3) are blocked together.	On	High
Agent destinations failing	The agent fails to reach one of its receivers without a break for the open delay while the other agents reach it (a receiver many agents fail on is that receiver's Destination unreachable finding).	On	Medium
Agent throttled	A `ThruputProcessor` "has reached maxKBps" line in 15 minutes, three live runs in a row, and the agent's limit is at most the reported limit (an agent rate-limited higher on purpose opens nothing).	On	Medium
Agent ingestion latency	The agent's health report says Ingestion Latency is red in its last report period.	On	Medium
Agent input errors	`TailReader` warnings and errors naming at least the minimum of distinct files in 15 minutes (permissions, checksum, read errors).	On	Low
Agent restart loop	At least the minimum of starts within an hour, on the larger of the agent's own starts ("Splunkd starting" lines, less the restarts its Deployment Server asked for) and the restarts its Deployment Server asked for: a deployment wave restarts an agent once, so fewer waves than the minimum never open it. A loop its Deployment Server causes is held while that server's **Restart storm** is measured failing or not yet recovered (one finding, never one per agent); a crash loop is never held.	On	Medium
Agent identity conflict	One GUID connected from several host names **at the same time** (latest connection lines within 3 minutes), or one host name under several GUIDs at the same time from different source addresses, or one GUID from several addresses on one receiver within 30 seconds, again and again (a clone that kept the golden host's name too). A rename, a replaced machine, NAT or several instances on one machine open nothing. Held while Cloned images is open.	On	High
Agent newer than receivers	The agent's release (major.minor) is newer than the newest release of every receiver it connected to over the day. A receiver of unknown version holds.	On	Low
Agent version unsupported	The agent's version is below the minimum, past the grace on its current version. The lowest listed version is the fleet's minimum; a listed version of a release raises it for that release	Off	Medium

Fleet

The fleet policies judge the fleet and every group with a known membership separately.

Policy	What it detects	Default	Severity
Mass disconnect	At least the open share of the active agents, and at least the minimum, disconnected together (fewer than 2 is never a mass disconnect). Recovers below the recover share; holds its agents' disconnection findings.	On	High → Critical at the critical share or none connected
Fleet coverage	Connected agents below the minimum, or below the second-highest daily peak of the last 7 complete days less the drop (3 days learned at least).	On	High
Version drift	Agents of the fleet or a group below the version baseline (Settings → Versions: the most common version, the highest, or a pinned one) after the rollout grace. One cohort finding listing the versions, never one per agent.	On	Low
Cloned images	At least the minimum of agents share GUIDs at the same time (a cloned image deployed without clearing its identity): one fleet finding, holding the clones' own identity findings.	On	High

Deployment Servers

Policy	What it detects	Default	Severity
Deployment Server silent	A Deployment Server with clients received no phonehome for 3 times its usual gap (15 minutes to 2 hours) and its `deploy-server` metric counted none.	On	High
Phonehome drop	The last 15 minutes below the drop percentage of a quarter of its usual hour (the median of the same UTC hour over 7 days; 3 days and 60 phonehomes an hour at least). Recovers at 80 %; held while silent.	On	High
Deployment failures	A deployment failing now, a failure in the last 15 minutes, or its failure counters rising; failures in the day but none now hold it.	On	High
Restart storm	At least the share of its clients, and the minimum of agents, restarted by it twice or more within an hour (a restart loop: an app redeployed again and again). One deployment wave opens nothing. While it is failing or recovering, its agents' Agent restart loop findings caused by it are held.	On	Medium
Deployment Server errors	An error line of the Deployment Server's own deployment components (server classes, deployment, apps served, client sessions) in its **latest configuration load** — a start or a reload, the only time it writes them. Recovers on the next load without them.	On	Medium
Requests rejected	At least the share of its phonehome and connect requests answered 4xx or 5xx in the last 15 minutes. Held under 20 requests or on a failed read.	On	Medium
Response slow	The exact 95th percentile of its phonehomes' response time at least the threshold over the same lines. Held under 20 phonehomes or on a failed read.	On	Low

Destinations

Policy	What it detects	Default	Severity
Destination unreachable	At least the shared minimum of agents, and the shared share of the receiver's senders, fail to reach it without a break for the open delay (longer while its cluster peer restarts). One finding on the receiver listing the agents and their count by group; the agents' own destination findings are held.	On	High

Default settings

Each policy's checks to open and to recover (one check per live window, every five minutes) and its own settings with their defaults.

Policy (id)	Open / recover	Settings and notes (defaults)
Agent disconnected (uf_agent_disconnected)	1 / 1	The tenant's connected window and dormancy
Expected agent missing (uf_expected_agent_miss	1 / 1	Host names or GUIDs (* allowed); up to 20 lookup tables; grace 0 s
Agent output blocked (uf_agent_output_blocke	3 / 2	Blocked 60 s; shared from 30 % and 3 agents
Agent destinations failing (uf_agent_destinations_	1 / 2	Open after 900 s
Agent throttled (uf_agent_throttled)	3 / 2	Lines 1; report limits up to 256 KB/s (the forwarder's default)
Agent ingestion latency (uf_agent_ingestion_lat	2 / 2	No setting.
Agent input errors (uf_agent_input_errors)	1 / 2	Files 3; recover after 24 h
Agent restart loop (uf_agent_restart_loop)	1 / 2	Starts 3; recover after 2 h
Agent identity conflict (uf_agent_identity_conf	3 / 2	Recover after 24 h
Agent newer than receivers (uf_agent_newer_than_re	1 / 1	No setting.
Agent version unsupported (uf_agent_version_unsup	2 / 2	Minimum versions (up to 20, one per line); grace 24 h
Agent not phoning home (uf_agent_not_phoning_h	2 / 2	Missing after 900 s (120 s to 7 days)
Deployment handshake failing (uf_agent_ds_handshake_	2 / 2	Retries 10
App deployment failed (uf_agent_deployment_fa	1 / 1	No setting.
Agent unmanaged (uf_agent_unmanaged)	1 / 1	No setting.
Decommission candidate (uf_agent_decommission_	1 / 1	The tenant's removal delay
Mass disconnect (uf_fleet_mass_disconne	1 / 2	Open 20 %; at least 2; critical 50 %; recover 10 %
Fleet coverage (uf_fleet_coverage)	2 / 2	At least 0; drop 20 %
Version drift (uf_version_drift)	1 / 1	Grace 24 h
Cloned images (uf_fleet_cloned_images	3 / 2	Agents 5; recover after 24 h
Deployment Server silent (uf_ds_silent)	1 / 2	No setting.
Phonehome drop (uf_ds_phonehome_drop)	2 / 2	Drop 50 %
Deployment failures (uf_ds_deployment_failu	1 / 2	Recover after 24 h
Restart storm (uf_ds_restart_storm)	1 / 3	Clients 20 %; at least 5
Deployment Server errors (uf_ds_config_errors)	1 / 1	Error lines 1; count warnings off; ignored messages (globs, up to 50)
Requests rejected (uf_ds_requests_rejecte	2 / 2	Rejected 5 %
Response slow (uf_ds_response_slow)	2 / 2	p95 1,000 ms
Destination unreachable (uf_destination_unreach	1 / 2	Open after 900 s (1,800 s while the peer restarts); shared from 30 % and 3 agents

Custom policies

TrackMe administrators of the tenant write their own policies in the **Policies** tab, above the catalogue: an SPL search run on this search head as the tenant owner, on its own schedule. Each result row names one subject, which opens a finding like a built-in policy and joins the Finding group of its object.

Subject type	Result field	Resolves to
Agent	`agent` (a GUID or a host name)	The agent of the directory: a GUID names its agent, a host name the agents reporting it. Several agents on one name (a cloned image) are ambiguous; an agent not listed or forgotten is not monitored.
Deployment Server	`ds` (a host name)	Named directly.
Group	`group` (a group id or name, or `fleet`)	The group of the tenant's settings.
Fleet	None	The fleet.

A row whose subject is not monitored is skipped and the run's other findings recover; an unresolved row (ambiguous, or before the first publication) opens what it resolved and recovers nothing. A remote source is searched through `| splunkremotesearch` in the SPL. The examples offered include *Agents dropping events* and *Deployment Servers logging errors*. Drafts, published revisions and activation are managed independently, Splunk validates the SPL before every preview and run, and each policy's runs are in Collection health. See *Findings, Finding groups and findings actions*.

Collection jobs

Job	Schedule	What it does
Inventory	Every 10 minutes	The agent directory from the receivers' lines (a day), the phonehomes (a day) and the agents' start lines; the Deployment Servers and their clients' phonehome state; the server class and output group members; the enrichment. The client records and the phonehome baseline come from the Deployment Server history job, with no search. The receivers' versions are read once a day (a receiver whose agents run a newer release, at once), and the telemetry probe runs when its verdict is due (none yet, a day old, no data, or a complete `_metrics` mirror, which is checked at every rebuild). Writes an agent row only when something about it changed; the directory is rebuilt about hourly, always on a manual run.
Deployment Server history	Hourly	The client records indexed since the last read (the first read, or a change of Deployment Servers, covers 30 days); what each agent reports when it phones home (the last hour); once a day, each Deployment Server's phonehome baseline (the last 7 complete days), and at once for a Deployment Server it does not cover. The inventory merges them without a search while they are fresh (client records about three hours, baseline about a day); otherwise it reads them itself.
Live	Every 5 minutes	The connected set and the heartbeats; assembles the latest facts of the jobs below into the fleet snapshot, the groups' membership and counts; publishes the fleet summary, writes the series, then evaluates every policy. Never builds the directory: until the first inventory, the connected agents are admitted and the policies judge nothing (`uf_directory_pending`).
Agent trouble	Every 5 minutes	The agents' own troubles (blocked outputs, throttling, input errors, restarts) and health indicators, the input files and throttle of the flagged agents only; the day's input errors once an hour, read last.
Destinations	Every 5 minutes, just before the live run	The destinations the senders fail to reach and the cluster peers restarting.
Identity conflicts	Every 5 minutes	The connected set and its source addresses: cloned GUIDs, host names under several GUIDs.
Deployment Servers	Every 5 minutes	The phonehomes, the late and absent clients, the received and recent deployments.
Deployment Server detail	Every 5 minutes	The Deployment Servers' resources and access log; the day's deployment failures every 15 minutes; a day of their own error lines once an hour.
Policies	On demand	Evaluates the policies on the latest live publication.

Schedules are offset per tenant. **Collect now** (the status strip) runs the live job; **Run now** in Collection health runs a job; **Scheduled collection** (Settings → Collection) is on by default. The reports are `trackme_splunk_uf_<job>_<generation>_tenant_<tenant>`, and the AI Agent report `trackme_uf_agent_tracker_tenant_<tenant>` (scheduled only once the scheduled triage is enabled).

Each job has its own report, time budget (its interval less 30 seconds), lock, receipt and Collection health line (2.4.23), and a search may use what is left of its job's budget: there is no fixed per-search timeout. The fact jobs run in the minutes before the next live run; a job that misses a slot makes the policies hold, never count twice, and facts older than three runs read as not read. With **Scheduled collection** off, a **Collect now** starts the other jobs beside the live run, and it takes a second **Collect now** before the policies judge anything. The reports run `| trackmesplunkcollect` (log `trackme_splunk_collect.log`); up to five jobs of a tenant can overlap under the tenant owner, so a least-privilege owner may need a higher search quota.

Settings reference

Settings is one browser with a single **Save settings**. A tenant role that is not a TrackMe administrator of the tenant sees it read-only. Changing a field marked *starts a new source* restarts collection, and the findings of the previous source close. Every policy threshold is a policy setting; the expected agents are set in their policy.

Section	Setting	Default	Notes
Source	Source (search tier)	`local`	Fixed once the tenant exists: create another tenant for another deployment.
	Internal, Metrics, Introspection index	`_internal`, `_metrics`, `_introspection`	Fixed once the tenant exists; an empty Metrics index means `_internal` only.
	Telemetry form	Automatic	Automatic, `_metrics` mirror, or `_internal`; starts a new source.
	Agent patterns	`*`	Host name patterns (`*` and `?`), matched case-insensitively. Every pattern setting of this section takes one token per host name or pattern (type it and press Enter; a pasted comma-separated list is split).
	Receiver patterns	`*`	The receivers whose connection lines count: host names or patterns with `*` (the only wildcard here). Every read of the receivers' lines takes it.
	Excluded agents, Excluded receivers	None	Never `*` alone; Excluded receivers take `*` as their only wildcard. Set at creation, the first backfill never reads what they leave out. A receiver left out stays infrastructure, never an agent.
	Full instances	Left out	*Counted as agents* makes every full-instance leaf sender an agent.
	Repair workspace	—	Recreates the tenant's collections and reports; findings and settings are kept.
Deployment Servers	Deployment Server source	`local`	Where the `_ds*` indexes are searched: a remote account on the Deployment Server on Splunk Cloud.
	Deployment Server API accounts	None	Optional, one per Deployment Server (up to 20), for the exact server class membership; never a new source. Its role needs `list_deployment_server`, and must be allowed to search `_dsphonehome`, `_dsappevent` and `_dsclient`: Splunk builds the client listing from these indexes, searched as the account (else the check reads *Cannot search the DS indexes*).
	Phonehome, Client, App event index	`_dsphonehome`, `_dsclient`, `_dsappevent`	Starts a new source.
Connectivity	Connected window	15 minutes	5 to 60.
	Heartbeat fallback	On	The heartbeat never creates an agent.
	Dormant after, Removal candidate after	7 days, 30 days	1 to 90; 1 to 365 and at least the dormancy delay, counted from the last connection.
Enrichment	Lookup tables and their fields	None	Up to 5 lookups of up to 500,000 rows, 10 fields each; never a new source.
Groups	Groups	None	Up to 200, 20 of them by lookup table.
Versions	Version baseline	The most common version	The most common, the highest, or a pinned version (Version drift).
Collection	Scheduled collection	On	
Notifications, AI Agent, Assignment, Finding actions	Shared settings of the findings tenants	Off / none	See *Findings, Finding groups and findings actions*; the assignment criterion is the agent **group**.

Collection health codes

The Collection health tab and the Configuration Guardian check `cribl_collection_health` (warning) report:

Code	Severity	Meaning
`live_never_ran`, `live_not_running`, `live_failing`	Warning, Failure, Failure	No live run since configuration; none for 20 minutes; runs not completing for 30 minutes.
`inventory_never_ran`, `inventory_not_running`, `inventory_failing`	Warning, Failure, Failure	No inventory run; none for 3 hours; not completing for 90 minutes.
`job_never_ran`, `job_not_running`, `job_failing`	Warning, Failure, Failure	A dedicated job (2.4.23) never ran 15 minutes after the first assembled snapshot; none for 15 minutes; runs not completing for 30 minutes. Until its next run, the policies reading its facts hold.
`policies_failing`	Failure	Policy evaluation not completing for 30 minutes.
`no_forwarding_receivers`, `no_agents`	Warning	An hour after configuration, the source shows no receiver of forwarded data; or the directory holds no agent.
`trouble_read_failed`, `trouble_read_partial`, `trouble_hosts_truncated`	Warning	The agents' own-log reads failed, came back partial, or more than 500 agents had something to report.
`ds_read_failed`, `ds_read_partial`, `ds_clients_truncated`	Warning	The Deployment Server indexes unreadable, partial, or more clients to report than kept.
`ds_baseline_failed`, `ds_recent_failed`, `ds_resources_failed`, `ds_requests_failed`, `ds_errors_failed`	Warning	The daily phonehome baseline, the last 15 minutes' phonehomes, the CPU and memory, the access log, or the configuration load errors could not be read.
`identity_addresses_failed`, `receiver_versions_failed`	Warning	The agents' source addresses (identity conflicts) or the receivers' versions (Agent newer than receivers) could not be read.
`group_lookups_failing`, `group_reads_failing`	Warning	A lookup group's lookup gave no value; the server class or output group membership read failed or was cut.
`group_serverclass_listing_ur`	Warning	Deployment Server API accounts are configured but a client listing could not be used, or no account answers for a Deployment Server of the fleet: server class groups count app events only.
`expected_lookups_failing`	Warning	An expected-agents lookup gave no host name (the last good values are in use).
`notification_delivery_failing`	Warning	Notification emails failing.
`custom_*`	Warning	Custom policies: schedule drift, failing runs, missed schedules, recovery required.

AI Agent

The Splunk UF Fleet AI Agent settings (**Settings → AI Agent**, or *Manage AI Agents*):

Setting	Default	Notes
Scheduled triage	Off	Hourly when on (`ai_uf_agent_enabled`).
Mode	Inspect	Inspect writes assessments and notes; act also applies the decisions allowed below.
AI provider	First available	
Findings per run, time budget per run	10, 3,000 s	
Review interval	1 day	A finding reviewed by a scheduled run is skipped that long unless it moved.
Allow investigating, allow temporary dismissal, allow resolve	Yes, yes (up to 7 days), no	A scheduled run never overrides an operator decision; it never Forgets an agent nor adds one to the expected agents.
Custom instructions	None	Added to the agent's instructions; **Evaluate…** tries them on a finding.

Runs are audited with the sourcetypes `trackme:ai_agent:uf_agent:inspect` and `trackme:ai_agent:uf_agent:act`. See *AI Agents for findings*.

Indexed finding events

Every finding transition is written to the tenant's notable index as `sourcetype= trackme:cribl:finding` (and Finding groups as `trackme:cribl:finding_group`), with `object_category=splunk_uf`, whatever the notification settings. The tenant's fleet, group and Deployment Server series are in its metric index under `trackme.uf.*`.

Permissions

- Any role of the tenant reads the workspace.
- The tenant's power and admin roles triage findings and **Forget** agents.
- A TrackMe administrator of the tenant (`trackmeadminoperations` and an admin role of the tenant) edits Settings, policies, groups, custom policies and the expected agents (**Add to expected** is a policy save).
- The scheduled runs search as the **tenant owner**: every remote account and lookup table the tenant reads must be usable by the owner.

Commands

Command	What it does
`| trackmeufdiscover`	The wizard's discovery (read-only, as the caller); `mode=ds_api ds_api_accounts=<a,b>` checks the Deployment Server API accounts.
`| trackmeuftrend`	24-hour series (`kind=fleet|group|ds|agent`): TrackMe's series for the fleet, a group or a Deployment Server; Splunk's own lines for one agent (at most 14 days).
`| trackmeufrecent`	The Last 5 minutes figures (never a state, writes nothing).
`| trackmeufevidence`	The raw lines behind a figure of an agent or a Deployment Server (`range=3600|14400|86400`): a timeline, a breakdown and at most 50 lines.

8.3.25 Splunk Forwarder Collectors

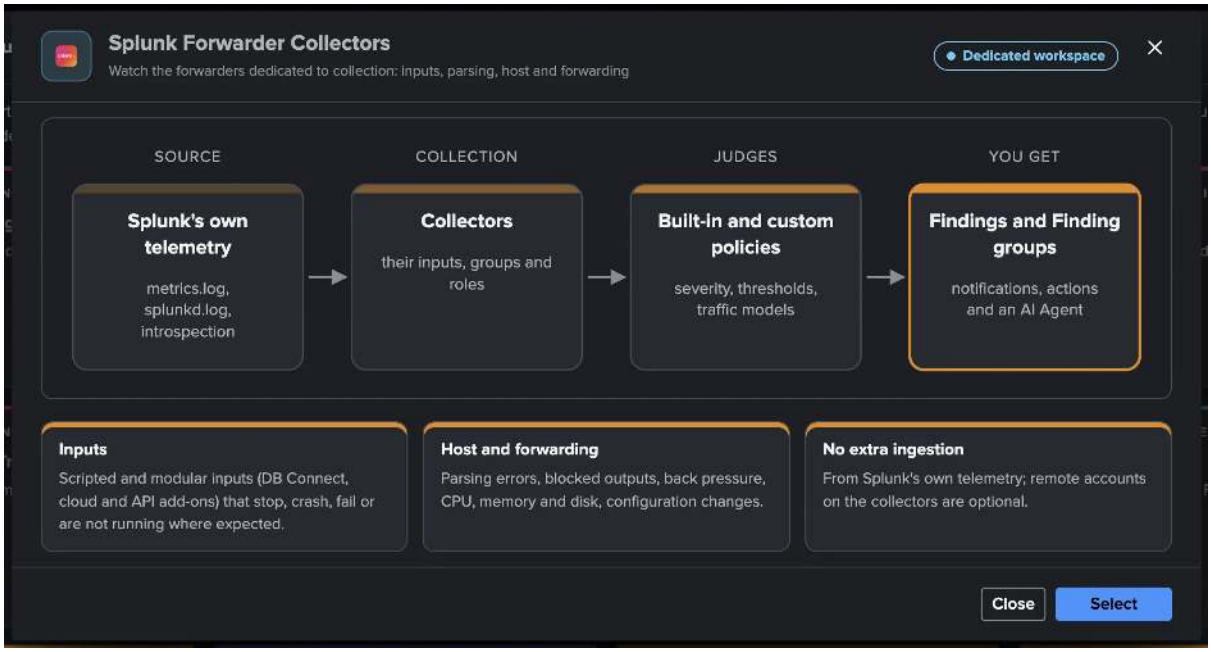
The **Splunk Forwarder Collectors tenant** follows the forwarders a Splunk deployment uses as **collection points**: the heavy forwarders that run DB Connect, cloud and API add-ons, scripted and modular inputs, and the forwarders, heavy or universal, that monitor the files rsyslog or syslog-ng write. It answers the questions a Splunk administrator asks about them: *is every input collecting? which inputs stopped, crash in a loop, write errors, went quiet, or were disabled when they must stay enabled? is the collector itself reporting, parsing, forwarding? is its output blocked, its pipeline saturated, its disk filling? and does each group of collectors keep enough healthy members?* The answers are **findings**, raised by built-in policies, with the same triage, Finding groups, notifications, assignment and findings actions as the other dedicated tenants, and a findings AI Agent of its own.

A Splunk Forwarder Collectors tenant is a **dedicated tenant type**: it has no entities and no state colour, and one tenant follows the collectors of one Splunk deployment. It reads what Splunk **already writes**: each collector's `metrics.log` and `splunkd.log` in `_internal`, its processes and disks in `_introspection`, and its configuration changes in `_configtracker`. Nothing is installed on a collector and no new data is ingested; remote accounts on the collectors are optional.

Every input as configured, for any add-on. TrackMe knows no add-on in particular: it lists each input an operator configured, on each collector, and gives two separate answers about it. *Does it run?* and *does it work?* A failing add-on may keep logging as usual, so running proves nothing about working: an input is judged by what it delivers, by the errors it writes, by a signal that went quiet, and by whether it stays enabled.

Syslog and file collectors are declared. Nothing in Splunk says that a forwarder collects other machines' data from files. You declare it on a group, and a universal forwarder is monitored only then.

The Splunk Forwarder Collectors tenant is generally available since TrackMe 2.4.24. The receiving side of the same deployment has its own tenant type, *Splunk S2S*, and the forwarding agents theirs, *Splunk UF Fleet*.



For a guided tour on a live lab, see the hands-on walkthrough *Splunk Forwarder Collectors*. Every policy, setting, job and command is described in the *Splunk Forwarder Collectors reference*.

What it monitors

Object	What TrackMe watches
Collectors	Every collector: whether its telemetry arrives (**silent**), what it collects and forwards against its usual (**volume drop**, **traffic model** outliers), its **output** (blocked, destinations failing, throttled), its **pipeline** (back pressure and the bottleneck queue), its **health report** (ingestion latency), **restarts**, **identity** (one GUID on several host names), **version**, **CPU, memory and disk**, **configuration changes**, and whether it is one of the **expected collectors** that must always report.
Inputs	Every input of every collector, for any add-on: an input that **must stay enabled** and is not, an input **process stopped** or **crashing in a loop**, **input errors** against the input's own usual level, an input that **went quiet**, and the resources a process uses. The file inputs too: **file input backlog**, **file input errors** and **parsing errors**.
Groups and fleet	Every **group** you define: **Group below minimum** when fewer of its members are healthy than the minimum you set, and **version drift** among its members; and the fleet as a whole: a **mass silence** of many collectors together is one finding, never one per collector.
Destinations	A receiver that **many collectors fail to reach**: one finding on the receiver listing the collectors, never one per collector.

How it works

A collector is a collection point. The inventory reads the Splunk instances of the host scope and keeps the full instances that forward and index nothing locally. Those that run inputs of their own are **collectors**; those that forward without collecting (licence managers, monitoring consoles, search heads that forward) are **utility nodes**, listed apart and left out unless you include them: all of them, or one by one with the **Include** box of the discovery. A universal forwarder is a collector only when you declare it (see below).

What is an input

Monitoring → **Inputs** lists what an operator configured, one row per input per collector; Monitoring → **Processes** lists what runs. The difference matters: one script runs every stanza of its modular input, so a process names no input, and what a process shows stays on its own row. Three generic sources name an input, whatever the add-on:

Source	Needs	What it gives
An API account on the collector	A TrackMe remote account	The current list of inputs, enabled or not, with their app, index, sourcetype and interval. An add-on may restrict its own configuration file to a least-privilege account.
The configuration changes (`_configtracker`)	Nothing (Splunk 9.0 or later)	Every stanza of a configuration file whose name holds `inputs`, its app, and whether it is disabled. A DB Connect input in `db_inputs.conf` is named this way.
What it delivers	Nothing	A modular input's default source is `scheme://stanza`: the source names the input. A file monitor delivers its files.

A row marked **Process** is a process with no input known behind it, for example an add-on that keeps its tasks in a file whose name does not hold `inputs`. Splunk's own inputs, what receives data from other forwarders (the Splunk S2S and Splunk HEC tenants watch those), the inputs your exclusions name (Settings → Inputs) and what an app only ships switched off are left out.

Input states. Each input shows the strongest evidence of the latest window:

State	Means
Disabled	Its configuration says so.
Delivering	Its own source took data in, or, on a syslog or file collector, its monitor stanza's files reached the indexes.
Logging	Its own log file took lines.
Running, **Writing**	Its process was sampled, or wrote to stderr (only when it is the only input of that process).
Its process runs	A process ran: a process names no input, so this says nothing more of this one input.
No activity evidence	Nothing in this window. **Never a failure on its own**: a daily input between two runs reads this way.

Syslog and file collectors

A typical Splunk deployment receives syslog on a few machines where rsyslog or syslog-ng write each device's messages to files, and a forwarder, heavy or universal, monitors those files. Nothing in Splunk says a forwarder does this, so **you declare it** with **Add collectors**, in the wizard, in Settings → Groups and in the Inventory. It lists the **universal forwarders seen sending** in the last 15 minutes, the busiest first, with their version, OS, receivers and volume: check the ones you know collect files, or type host names or patterns, and choose their group. The list is what the receivers saw, never a guess: TrackMe does not decide which forwarders collect files. A group's **Syslog or file collectors** switch declares its members the same way. A universal forwarder is monitored only when declared.

Once declared, on Splunk 9.4 and 10.x:

- its telemetry arriving, what it collects and forwards, its output and queues, its file backlog and file errors, and its Splunk version (from what its receivers write of it);
- the group's **minimum of healthy members**;
- each **monitor stanza** listed as an input: it reads **Delivering** when its files reached the indexes in the window, and **Input went quiet** opens, once per group, when a feed with a learned rhythm stops.

A stanza that names the host of its events (`host_segment`, `host_regex` or `host`) is read as a **feed**: its events carry each device's name, never the forwarder's. Any other stanza is read on the collector's own host name, since a path such as `/var/log/auth.log` exists on every machine. The tenant owner needs search access to the indexes the files go to: without it, a stanza stays *No activity evidence* and is never called quiet. Syslog **inputs** on a Splunk forwarder itself are out of scope (Splunk discourages them); only the forwarders monitoring the files written by a syslog daemon are in.

Collection

Twelve scheduled reports per tenant, each with its own schedule, time budget, lock, receipt and Collection health line: the **inventory**, the **registrations**, the **live** run (every 5 minutes, which assembles the facts and evaluates every policy), and nine fact jobs (**pipeline**, **resources**, **inputs**, **add-on signals**, **changes**, **trouble**, **destinations**, **enablement** and the **traffic ML models**). Twelve is the most of any product: on Splunk Cloud, the stack's search concurrency applies. Every read names the collectors it reads, so a large universal forwarder fleet in the same indexes never adds to the cost. A new tenant fills progressively: the directory first, then its registrations, then the configuration changes replayed back up to 90 days. See *Splunk Forwarder Collectors reference* for each job.

Before you start

A licence. A valid **Enterprise**, **Unlimited**, Unlimited trial or **Developer** licence.

The collectors' internal indexes. The collectors forward their `_internal`, `_introspection` and `_configtracker` to indexers the search tier can read: Splunk's default forwarding of internal indexes. `_configtracker` needs Splunk 9.0 or later on the collector. The index names are configurable.

A source. The search tier that sees those indexes: **local**, or a **TrackMe remote account** (*Remote Splunk deployments*) to another deployment.

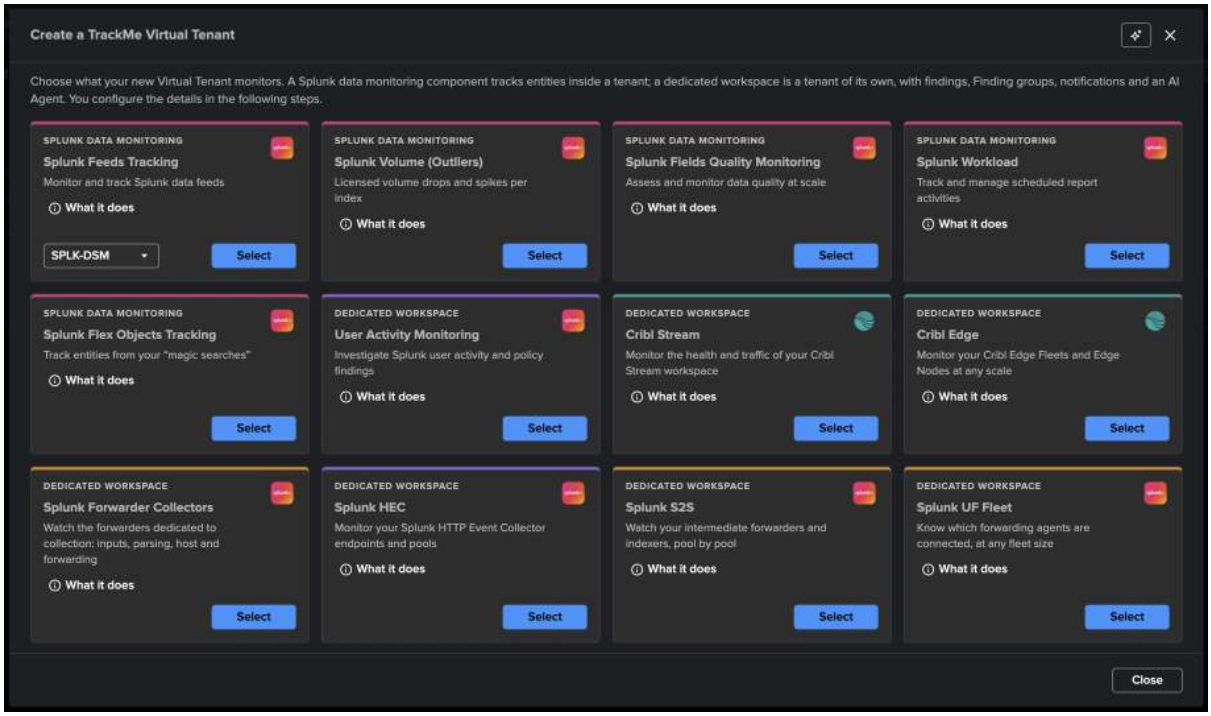
Optionally, an API account per collector. A TrackMe remote account on each collector gives the current list of its inputs and their state. It is never required: every policy works from the telemetry alone. From Splunk Cloud, an

account on an on-premises collector connects through a **TrackMe Broker** (the account's connectivity, see *TrackMe Broker*). *Which components need a remote account?* shows where each account points.

A tenant owner: `nobody`, or an account (typically a least-privilege service account) holding the capabilities the wizard lists, allowed to use every remote account the tenant reads and to search the indexes. The scheduled triage of the AI Agent needs an owner that is not `nobody`.

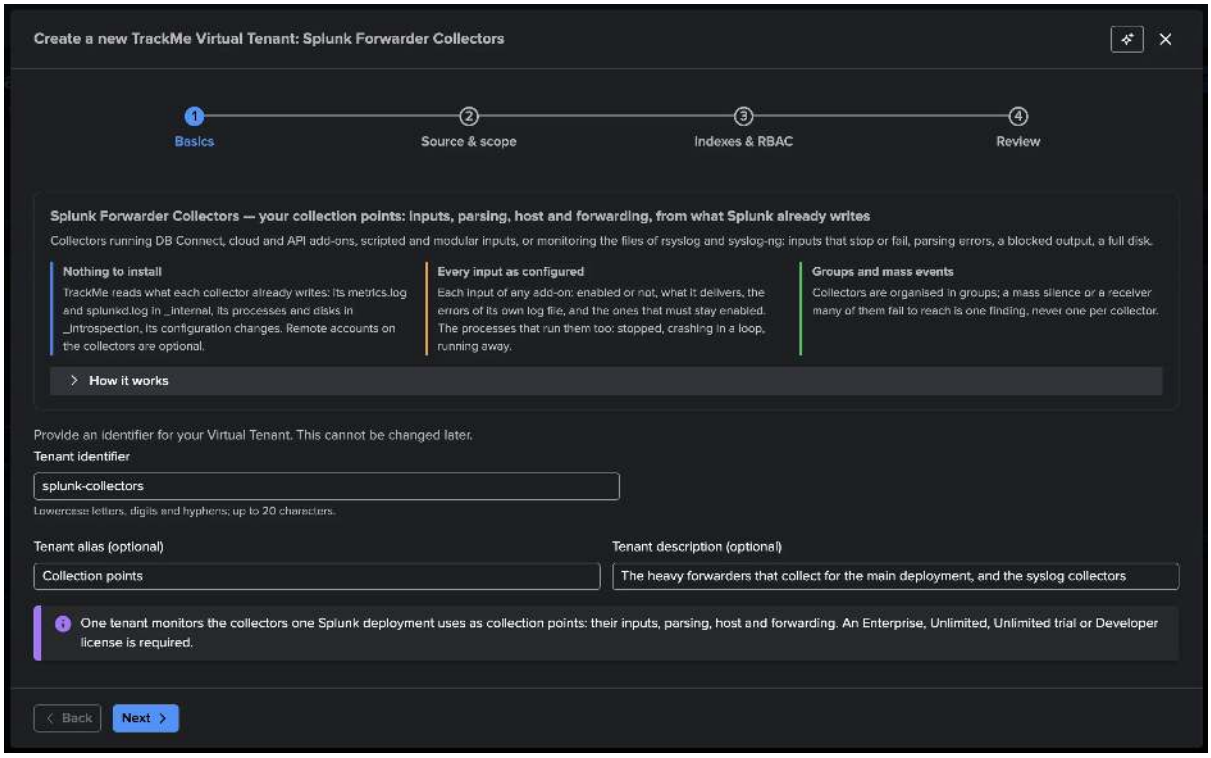
Creating the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Splunk Forwarder Collectors** card ("Watch the forwarders dedicated to collection: inputs, parsing, host and forwarding"). **What it does** opens the dialog shown at the top of this page.

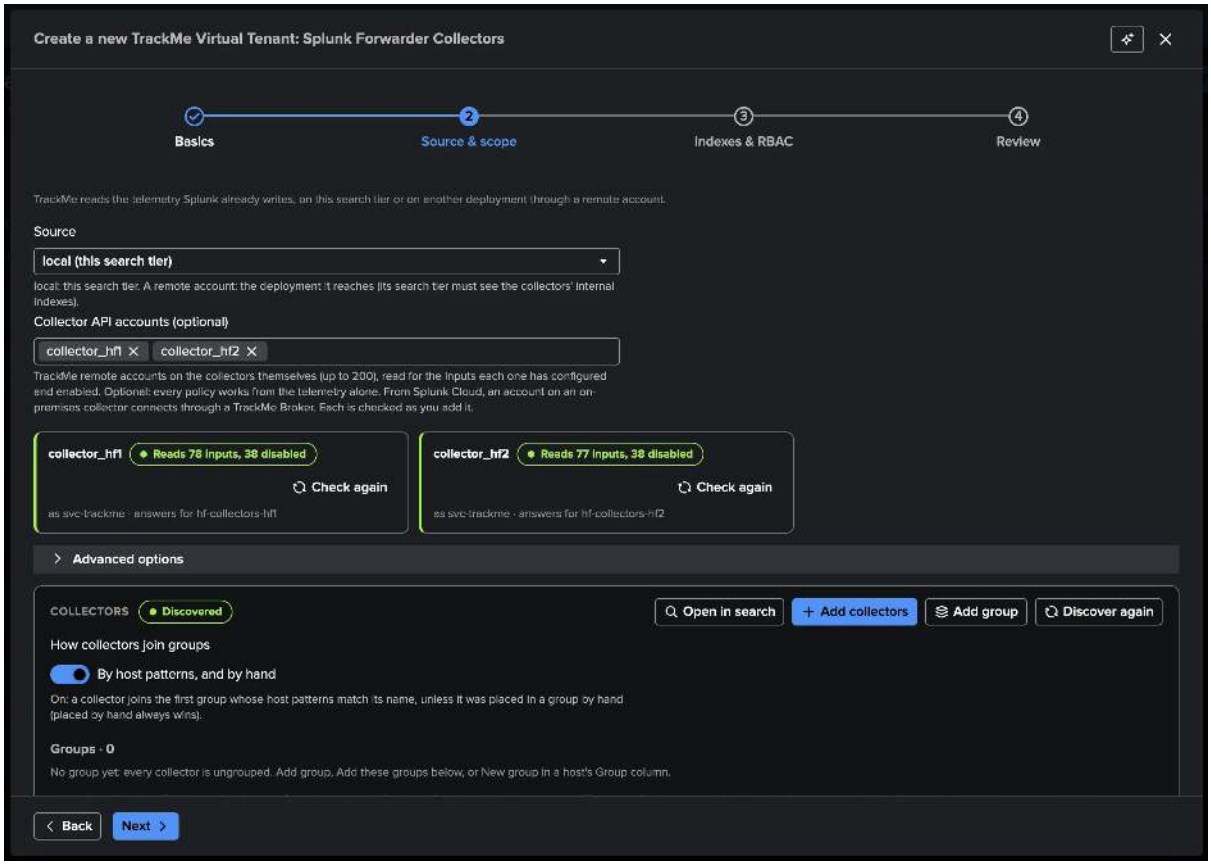


The wizard has four steps.

1. Basics — an introduction (nothing to install, every input as configured, groups and mass events), then the tenant identifier (lowercase letters, digits and hyphens, up to 20 characters, fixed once created), an alias and a description.

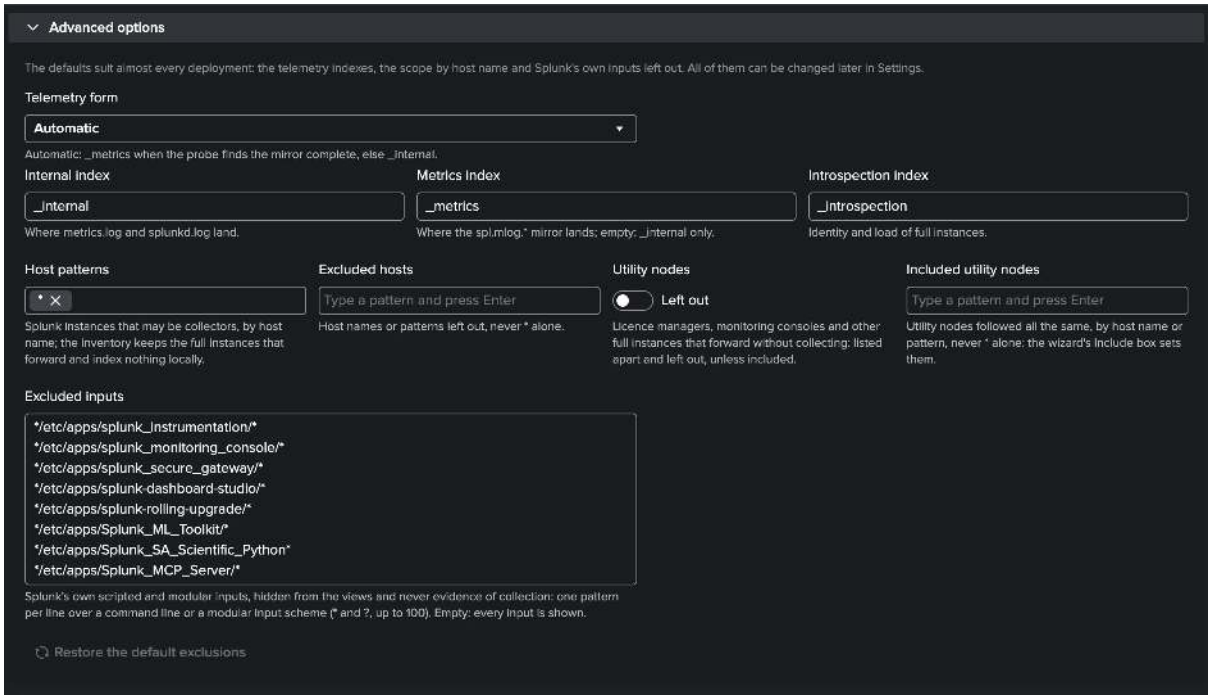


2. Source & scope — the **Source** and the optional **Collector API accounts**, then the **Collectors** card. Each API account is checked as you add it, one card each: who the account is on its collector, which collector answers, how many inputs it lists and how many are disabled. A read it cannot do is named with the read-only role that fixes it (`search`, `rest_properties_get` and `list_inputs`). A failing account can still be chosen: no policy depends on it. From Splunk Cloud, an account on an on-premises collector connects through a TrackMe Broker.



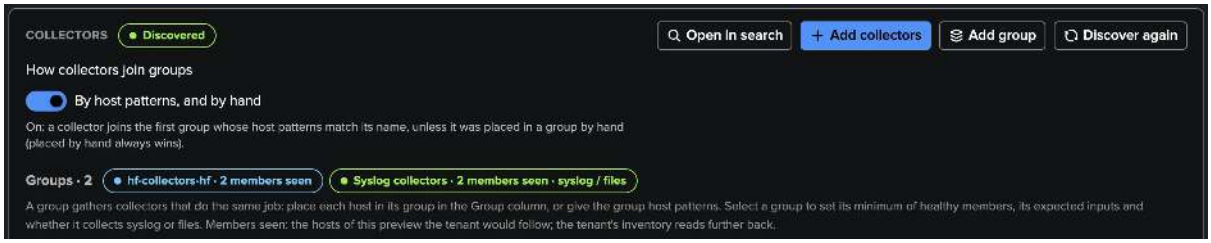
Advanced options (folded) hold the telemetry form, the three index names, the **Host patterns** (the instances that

may be collectors), **Excluded hosts**, whether **Utility nodes** count, **Included utility nodes** (utility nodes followed one by one, by name or pattern, never * alone) and the **Excluded inputs**: Splunk's own scripted and modular inputs, hidden from the views and never evidence of collection (one pattern per line, * and ?, up to 100). The defaults suit almost every deployment, and all of them can be changed later in Settings. The discovery's **Include** boxes fill Excluded hosts and Included utility nodes for you.



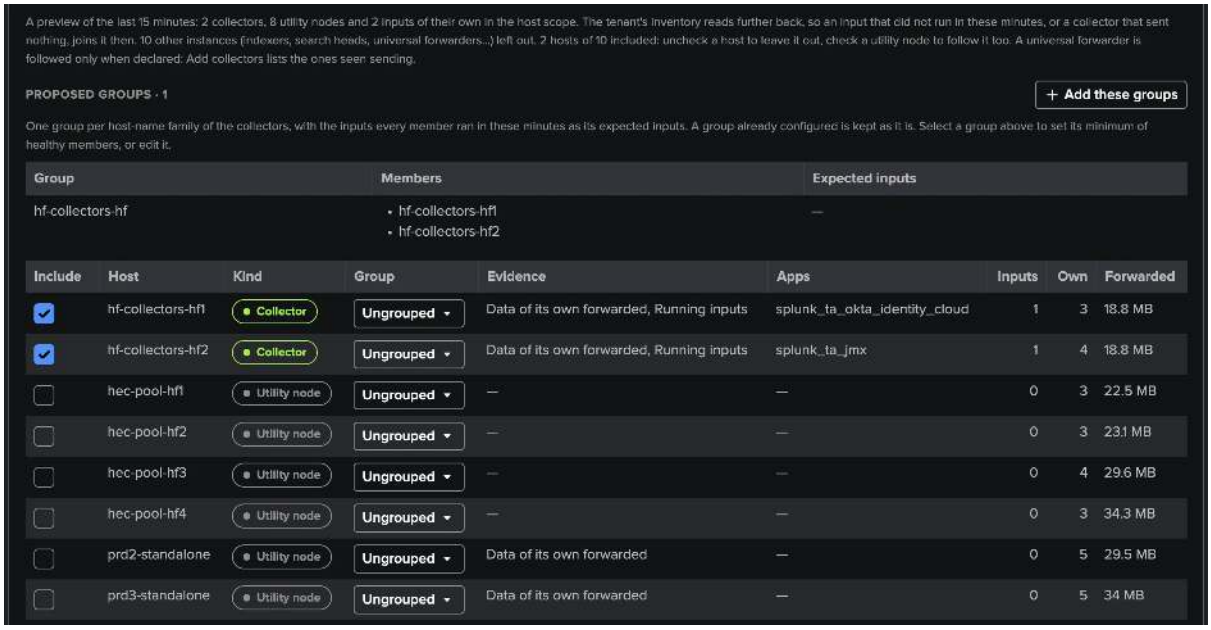
The **Collectors** card holds everything about the collectors and their groups:

- **How collectors join groups**: *By host patterns, and by hand* (a collector joins the first group whose host patterns match its name, unless it was placed in a group by hand) or *By hand only*.
- the **groups** of the draft, one pill each with its members seen; selecting a pill opens the group.
- **Add collectors**, **Add group**, **Open in search** (the discovery's searches) and **Discover again**.



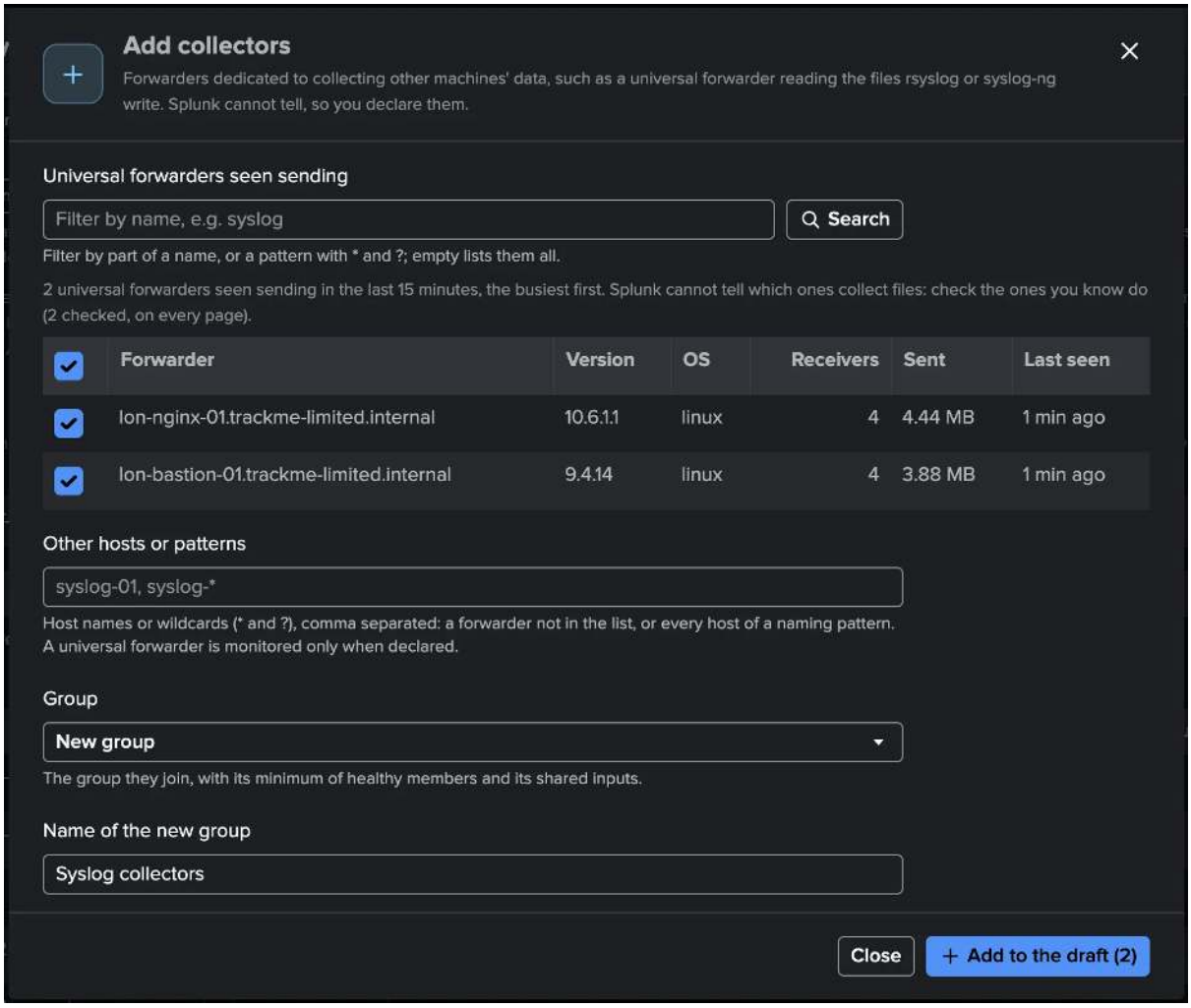
The **discovery** is a short preview of the last 15 minutes, with the tenant's own rules: the collectors, the utility nodes and the inputs they ran, and what was left out (indexers, search heads, universal forwarders not declared). It runs two short searches and never blocks the step; the tenant's inventory reads further back. It proposes **one group per host-name family** of the collectors, with the inputs every member ran as its expected inputs: **Add these groups** puts them in the draft. The table pages 50 hosts at a time, and says which ones the tenant will follow:

- **Include**: a collector is checked, and unchecking it adds its exact name to the excluded hosts (it stays listed, so you can check it again). A utility node is unchecked, and checking it adds it to the included utility nodes. A box is locked, with a tooltip saying why, when a pattern decides for the host: excluded by a host pattern, every utility node included, or a universal forwarder no longer declared.
- **Kind**: *Collector*, *Utility node*, or *Declared (UF)* for a universal forwarder you declared, from the draft rather than the last run.
- **Group**: places the host by hand in a group of the draft, or in a *New group*….

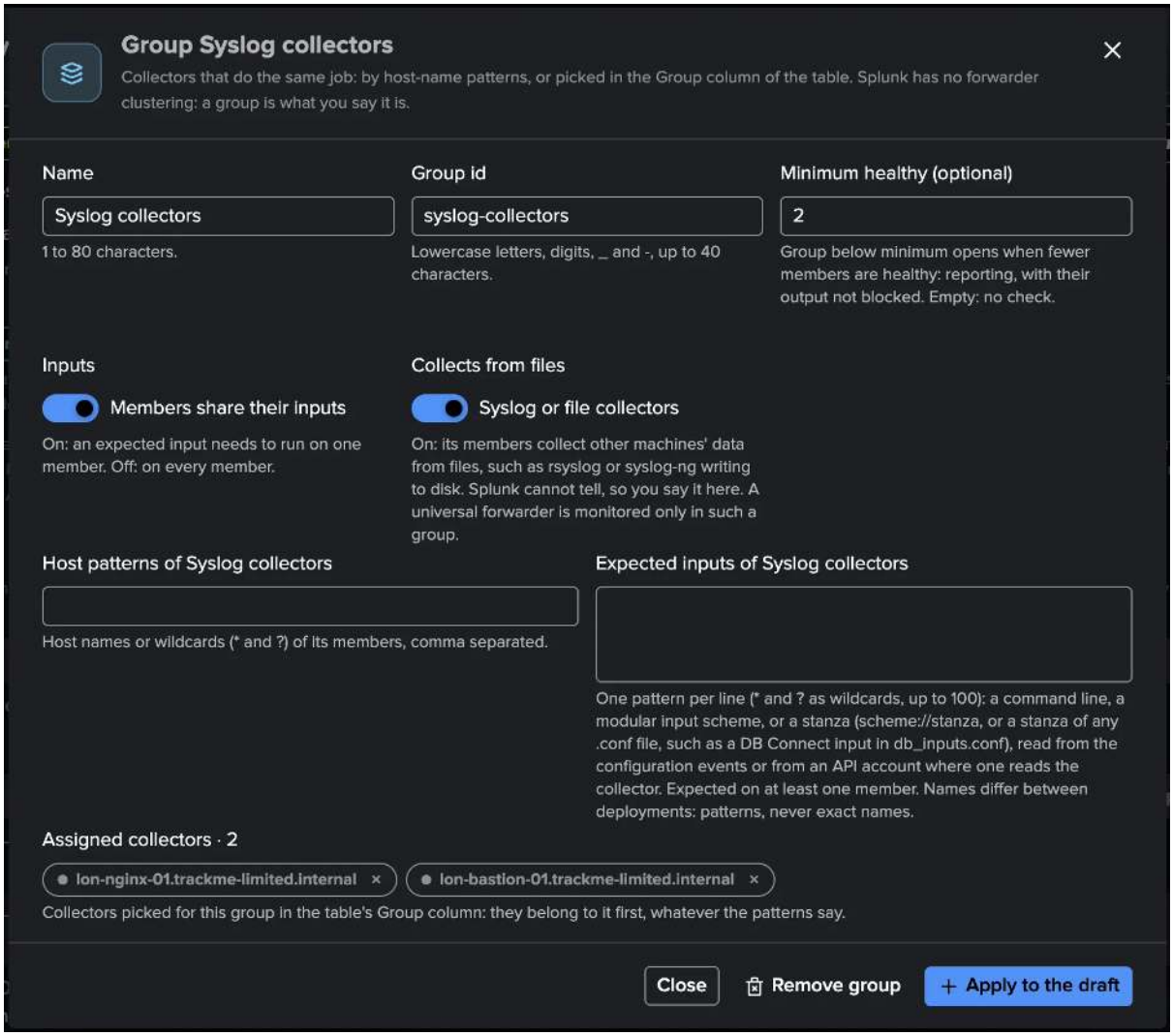


Groups gather collectors that do the same job, by host-name patterns or by hand. Splunk has no forwarder clustering, so a group is what you say it is. Each group has a name, an id (fixed once saved: its findings and history follow it), an optional **minimum of healthy members**, **Members share their inputs** (a standby pair, or an add-on with its own high availability: an expected input then needs to run on one member only), **Syslog or file collectors**, its host patterns, its **expected inputs** and its **assigned collectors** (the hosts placed in it by hand).

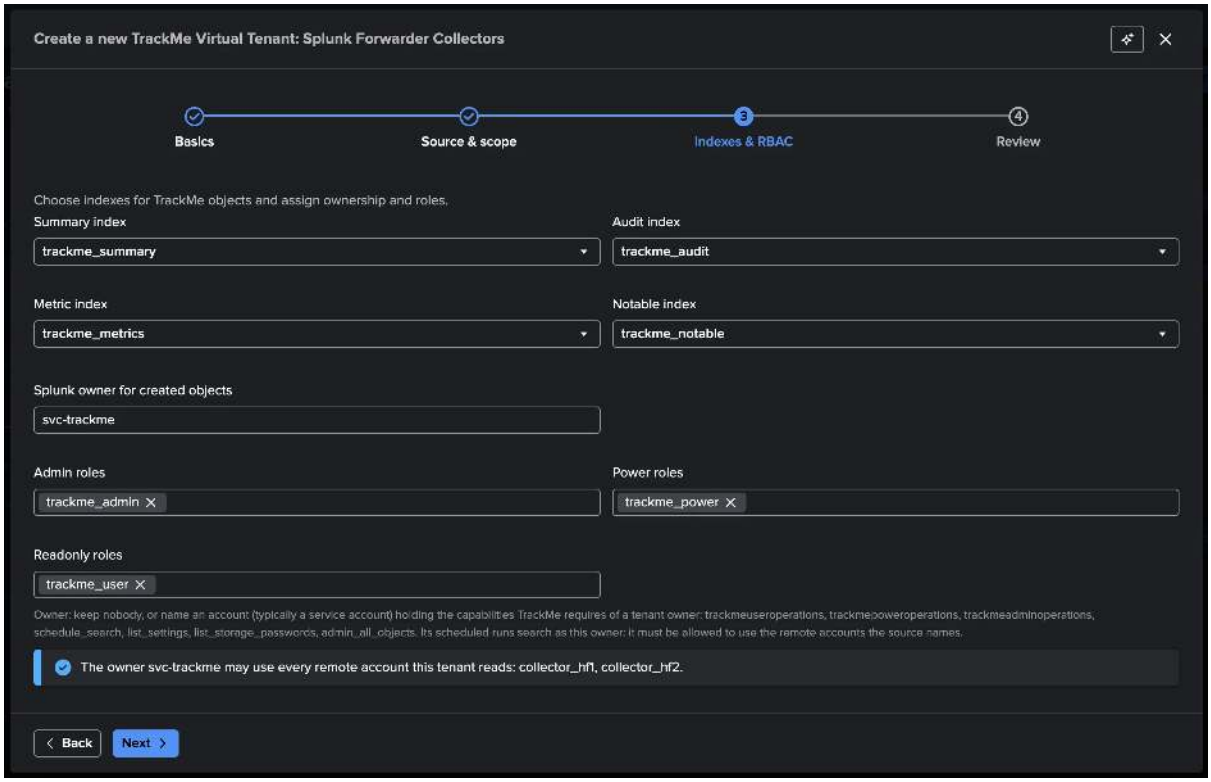
Add collectors declares forwarders that collect other machines' data from files. It lists the universal forwarders the receivers saw sending in the last 15 minutes, the busiest first, 50 a page, with a name filter (part of a name, or a pattern with `*` and `?`); picks are kept across pages and searches. **Other hosts or patterns** declares a forwarder not in the list, or every host of a naming pattern. Here the two syslog forwarders join a new group, **Syslog collectors**:



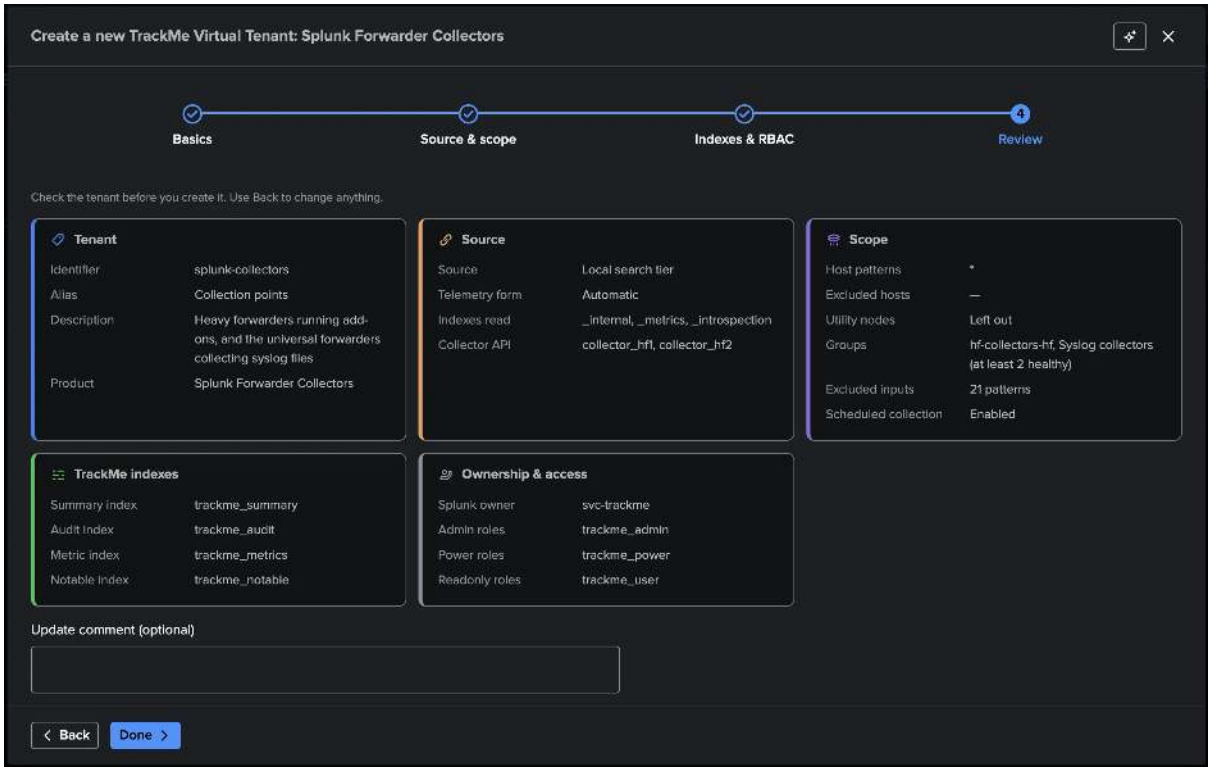
The picked forwarders are assigned to the group, which becomes a syslog or file collectors group. Selecting its pill opens it: below, it also gets a minimum of 2 healthy members and shares its inputs, since both forwarders monitor the same files and one of them is enough to run an expected input. **Remove group** deletes it from the draft.

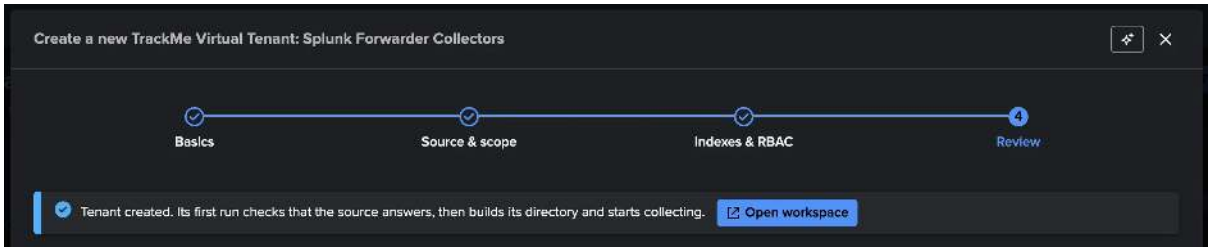


3. Indexes & RBAC — the tenant's TrackMe indexes, the owner and the roles. The wizard checks that the owner may use every remote account the tenant reads.



4. Review — then **Done**. The first run checks that the source answers, then builds the directory and starts collecting.

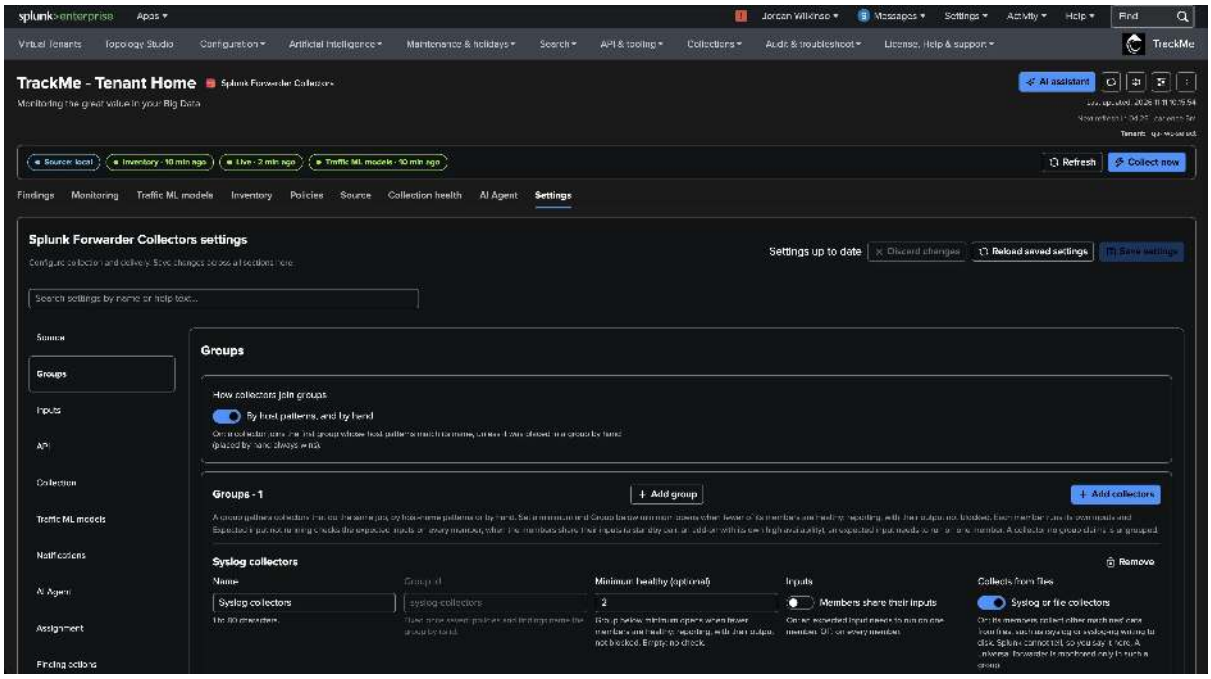




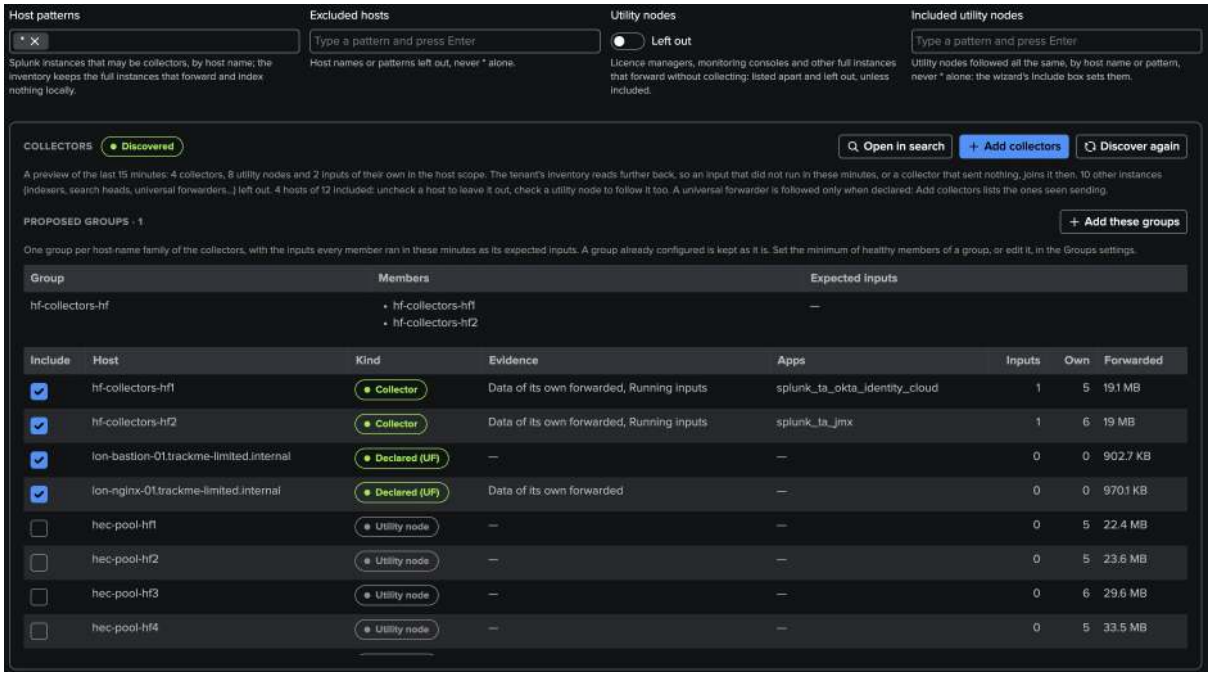
Configuring the tenant

Groups

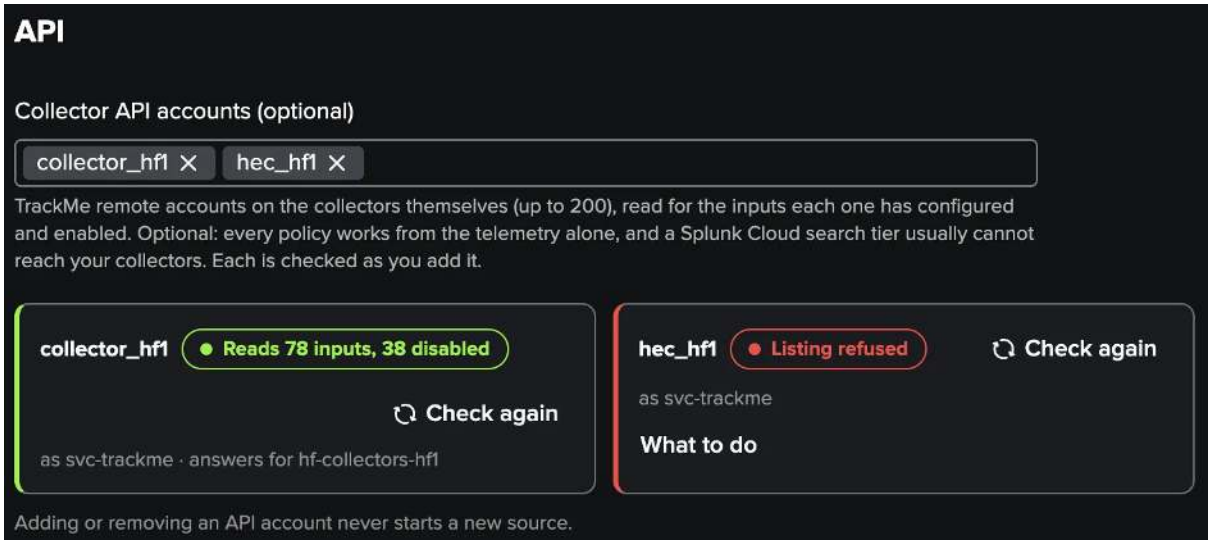
Settings → Groups holds the groups of the wizard: **How collectors join groups** at the top, then each group in full, with **Add group** and **Add collectors** (the same dialog as the wizard's). A group's minimum, its shared inputs, its expected inputs and its syslog or file collectors switch can be changed at any time; its id cannot.



Settings → Source runs the same discovery as the wizard, with its **Include** boxes: the universal forwarders you declared read *Declared (UF)*, and checking a utility node follows it. Changing the telemetry form, the host patterns, the excluded hosts or the utility nodes followed starts a new source, with its own findings and history.



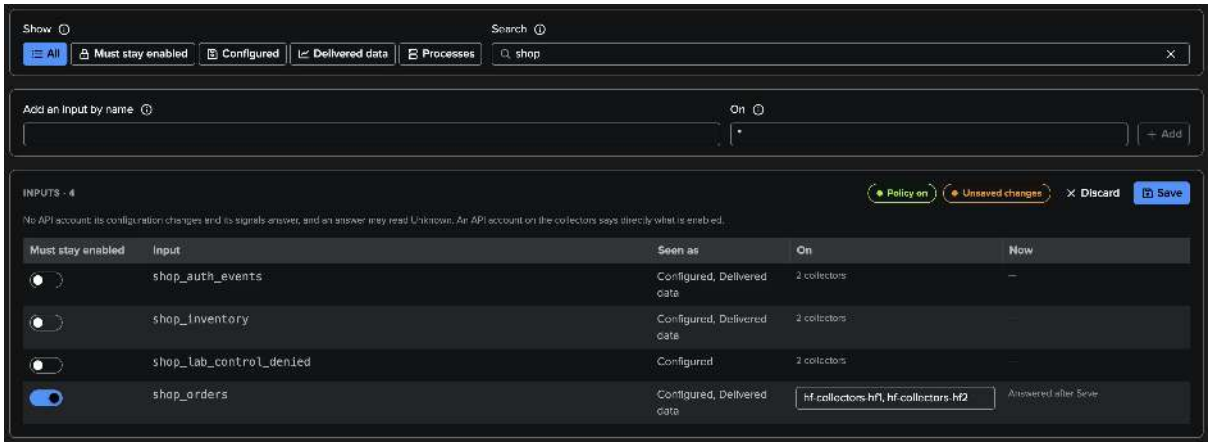
Settings → API lists the collector API accounts with the same check as the wizard, each with **Check again**; a read the account cannot do says what to do.



A collector no group claims is **ungrouped**. A group with no monitored member left (its members all stopped in the Inventory, or its patterns naming no collector) is not shown in Monitoring or on the Fleet map until it has one again; a silent member is still monitored, so its group stays. A group's **expected inputs** are patterns (`*` and `?`) over a command line, a modular input scheme or a stanza (`scheme://stanza`, or a stanza of any `.conf` file such as a DB Connect input in `db_inputs.conf`): **Expected input not running** opens when one does not run on every member, or on one member when the members share their inputs.

Inputs that must stay enabled

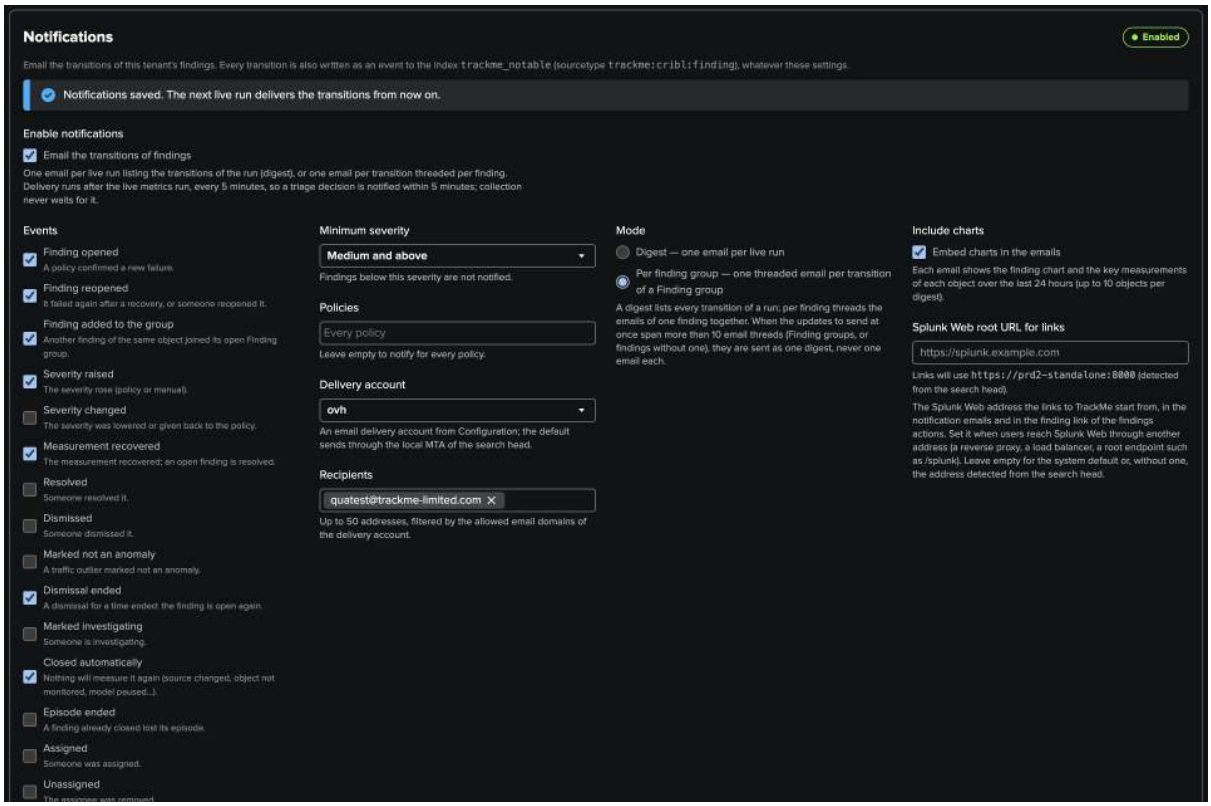
An add-on may disable its own input (DB Connect, after repeated connection failures), and an operator may disable one by mistake. **Inventory → Inputs** lists every input the tenant can see, with a **Must stay enabled** switch on each: a change is a draft until **Save**, and **Discard** undoes it. Each one is required on the collectors you name (names or patterns, `*` for every collector), whatever their group, and an input never seen can be added by name.



The answer is read from an API account first (the current state), then from the configuration changes, then from what the input shows. A disable stands until activity comes more than 10 minutes after it (a last batch arrives after a disable). When no source can say, it reads *unknown*, never absent. The finding is always at least high, and the AI Agent can never dismiss it while it fails.

Notifications

Settings → Notifications emails the transitions of the tenant's findings: the events notified, the minimum severity, the policies, the email delivery account, the recipients, and the mode — one digest per live run, or one threaded email per transition of a Finding group — with the finding's chart and key measurements embedded.



Every transition is also written as an indexed event to the tenant's notable index (`sourcetype=trackme:cribl:finding` and `trackme:cribl:finding_group`, the sourcetypes every findings tenant shares, with `object_category=splunk_collector`), whatever these settings, for your own Splunk alerts, Enterprise Security or SOAR:

```
index=trackme_notable sourcetype=trackme:cribl:finding␣
↪object_category=splunk_collector tenant_id="<tenant>"
```

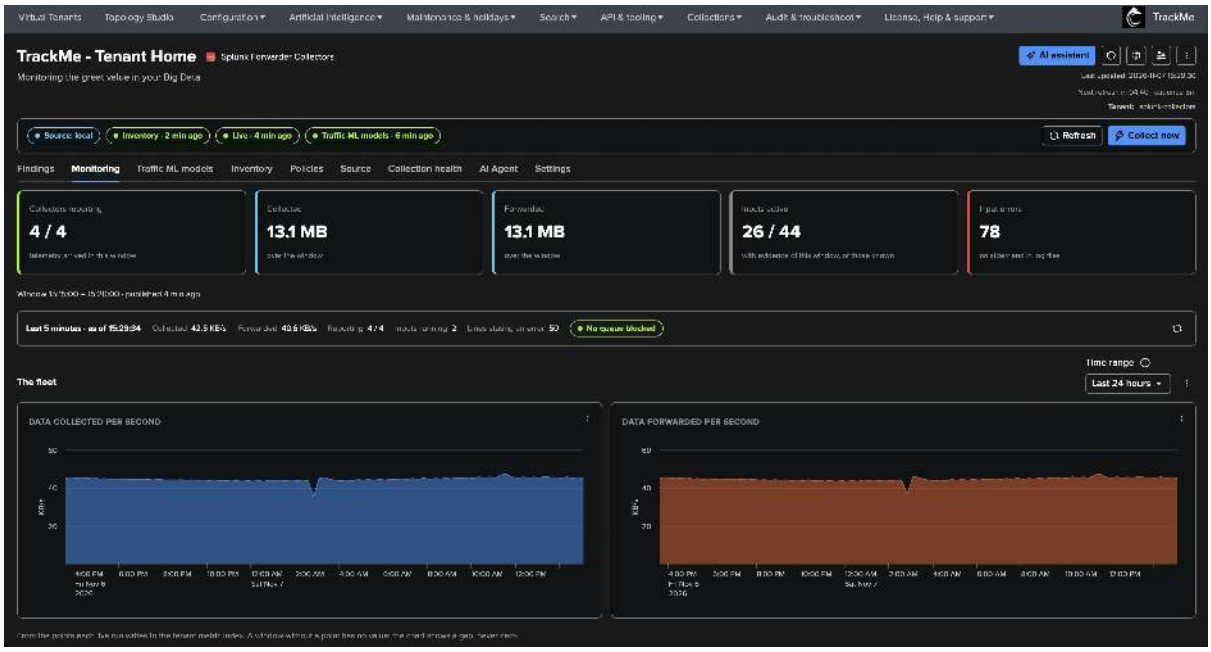
See *Findings, Finding groups and findings actions*.

The workspace

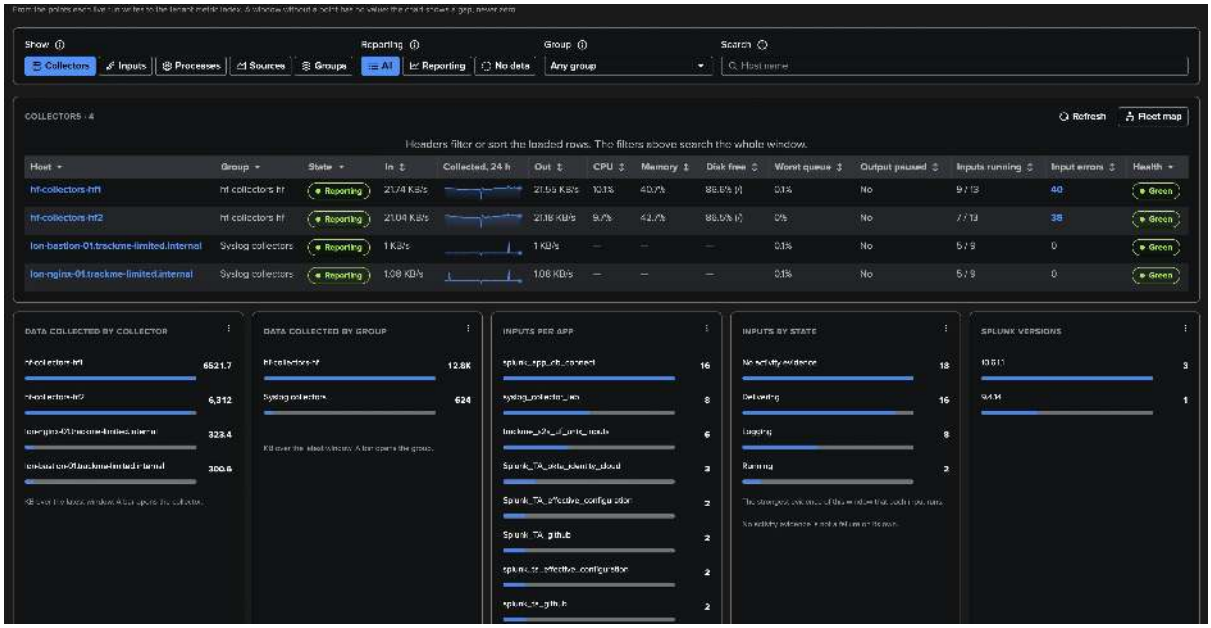
The workspace is the Tenant Home of the tenant: **Findings**, **Monitoring**, **Traffic ML models**, **Inventory**, **Policies**, **Source**, **Collection health**, **AI Agent** and **Settings**. A status strip above the tabs shows the source and the **Inventory**, **Live** and **Traffic ML models** pills (each opens Collection health), with **Refresh** and **Collect now** (runs the live job).

Monitoring

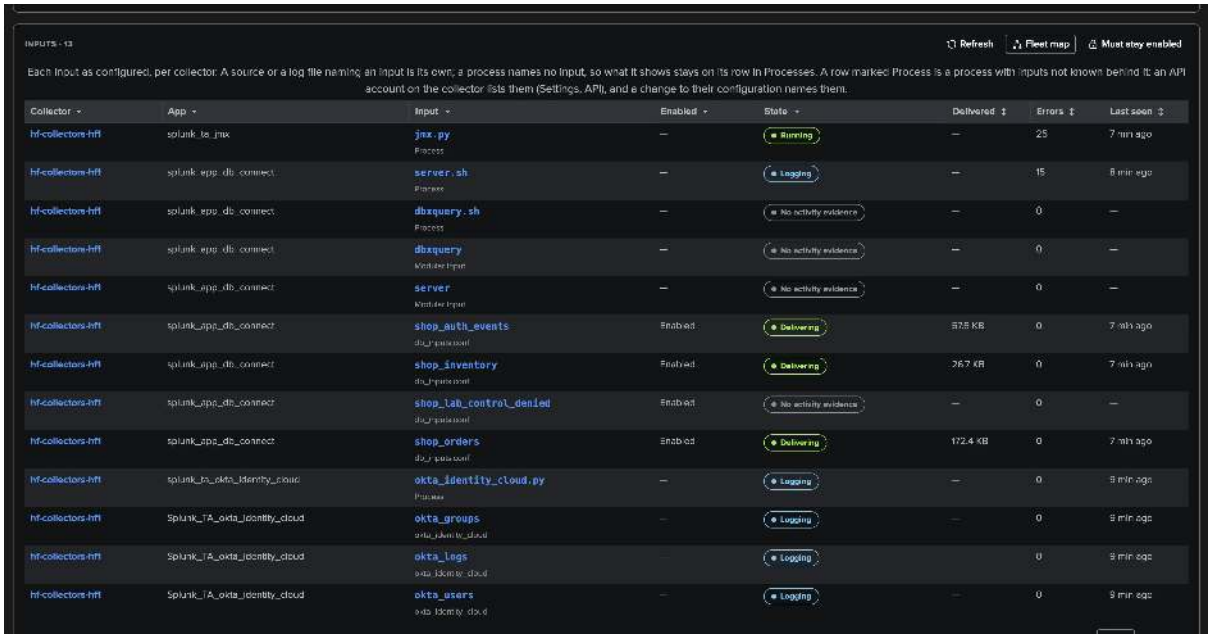
Monitoring is the main view: the tiles of the latest window (collectors reporting, collected, forwarded, inputs active, input errors), the **last 5 minutes** read on demand, and the fleet's history, what the collectors collected and forwarded per second.



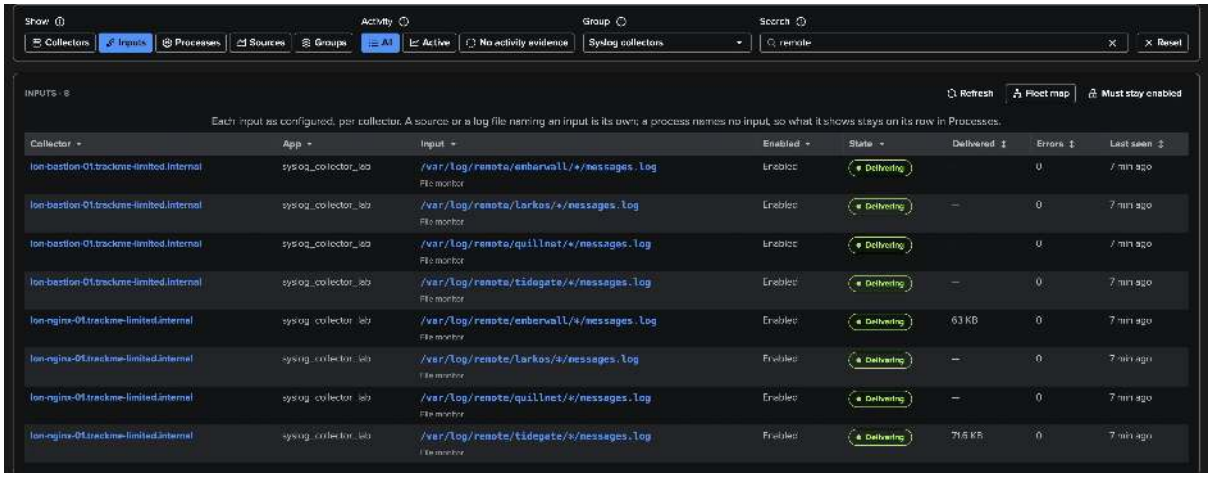
Below, **Show** switches the table between **Collectors**, **Inputs**, **Processes**, **Sources** and **Groups**, with the **Reporting** (or **Activity**) and **Group** filters and a search; breakdown cards under the table count the whole window whatever the filters.



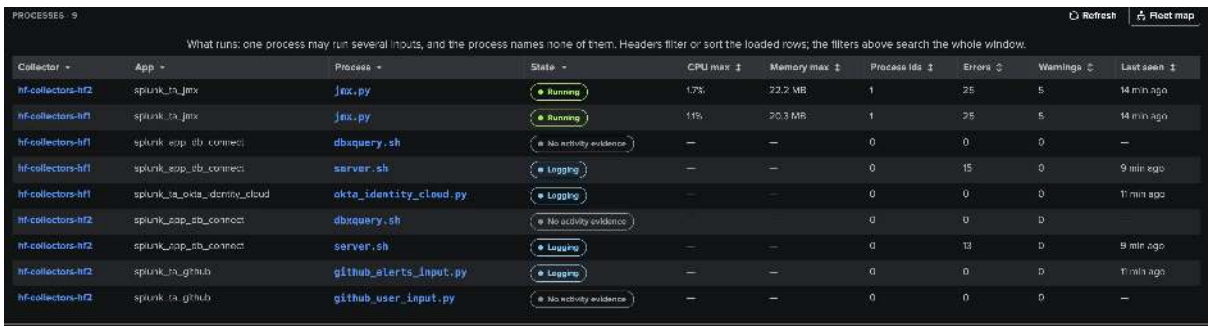
The **Inputs** view of one collector shows the three ways an input is known. The JMX add-on runs as a process (**Running**, its errors on stderr); DB Connect runs its own server process (**Logging**) and keeps its inputs in `db_inputs.conf`, named by the configuration changes, each **Delivering** its own source; the Okta add-on runs one script for three stanzas, each **Logging** in its own file. The modular input defaults show *No activity evidence*, which is not a failure.



On the syslog collectors, each monitor stanza is an input. Here the four vendor feeds of both forwarders read **Delivering**, from what reached the indexes. **Delivered** shows a figure only when the forwarder's own metrics name the file: Splunk lists the largest sources only, and a universal forwarder on 9.4 lists none.

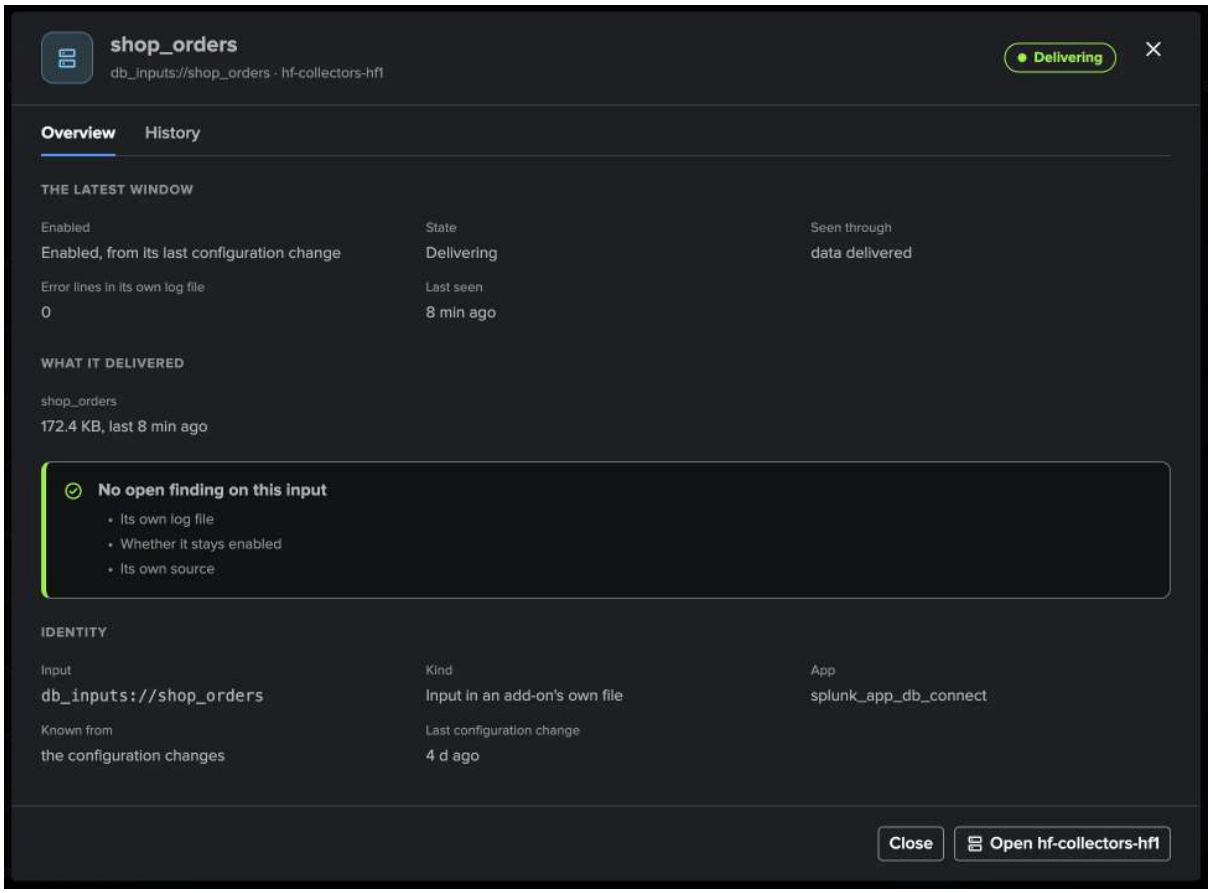


Processes shows what runs, with its CPU and memory, its process ids and the error and warning lines it wrote; **Groups** shows each group's members reporting against its minimum, what they collected and forwarded, and their inputs.

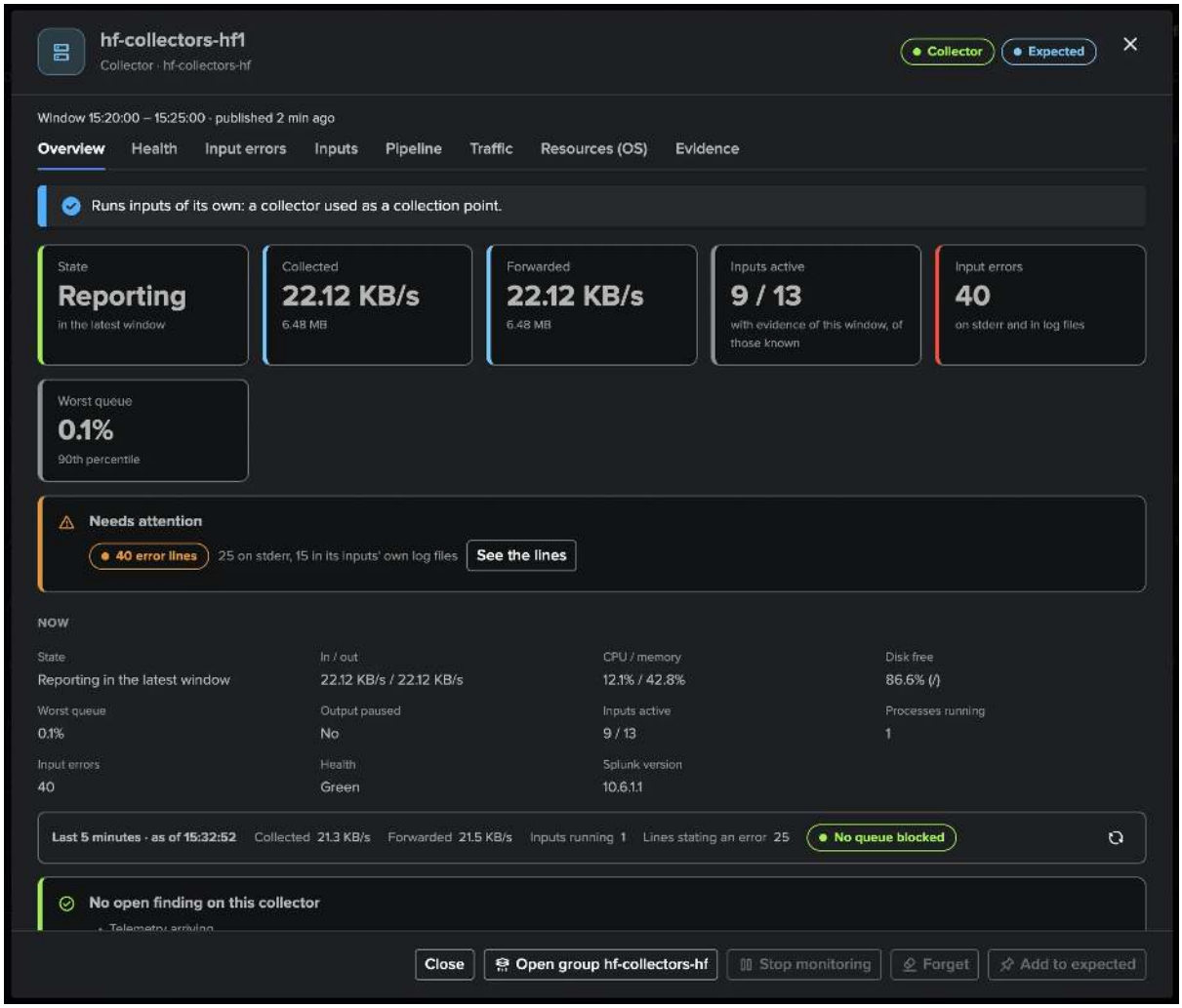


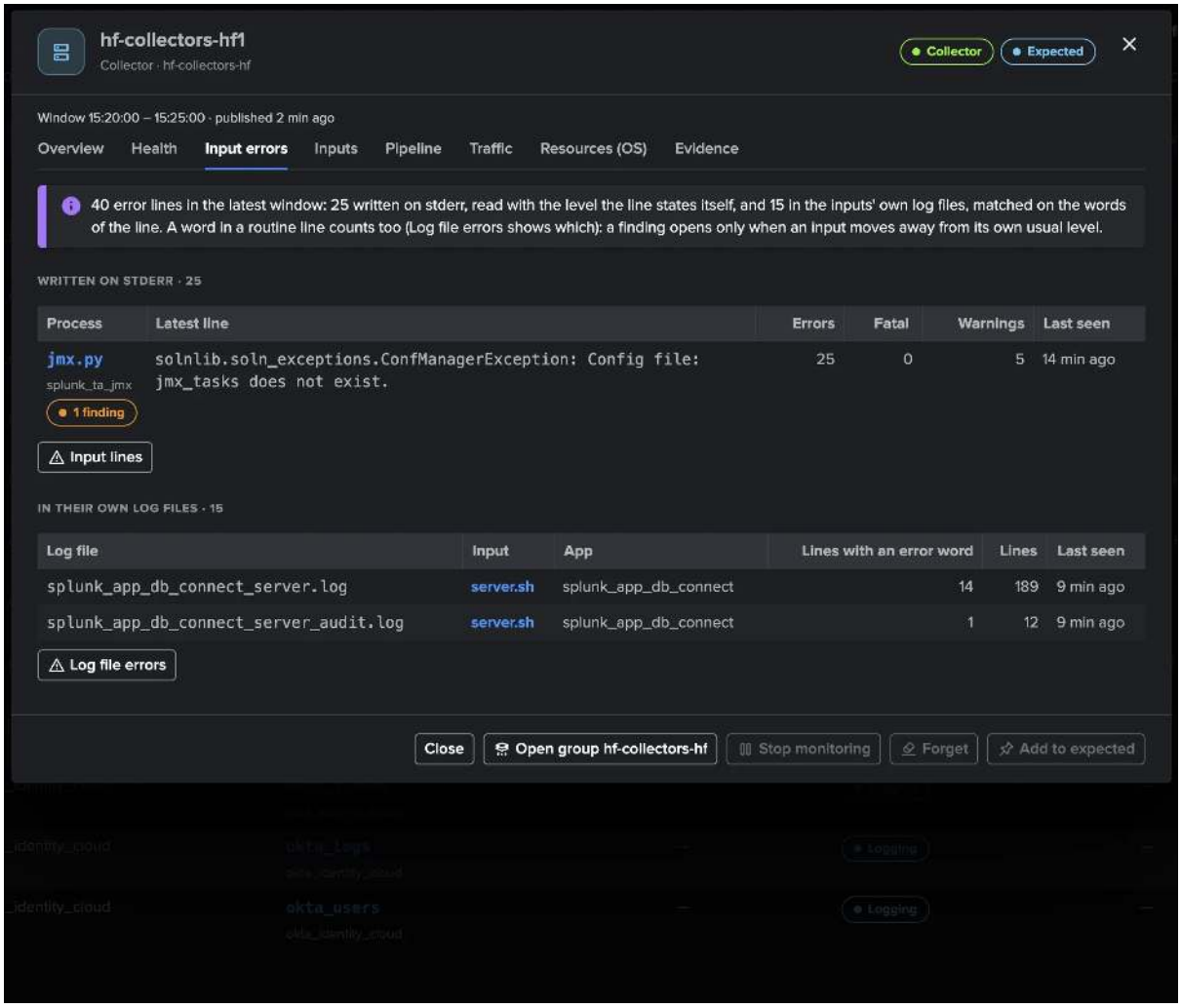


An input opens its detail: its state and how it was seen, what it delivered, the error lines of its own log file, whether it stays enabled, its identity (the input, its kind, its app and where it is known from) and its history.



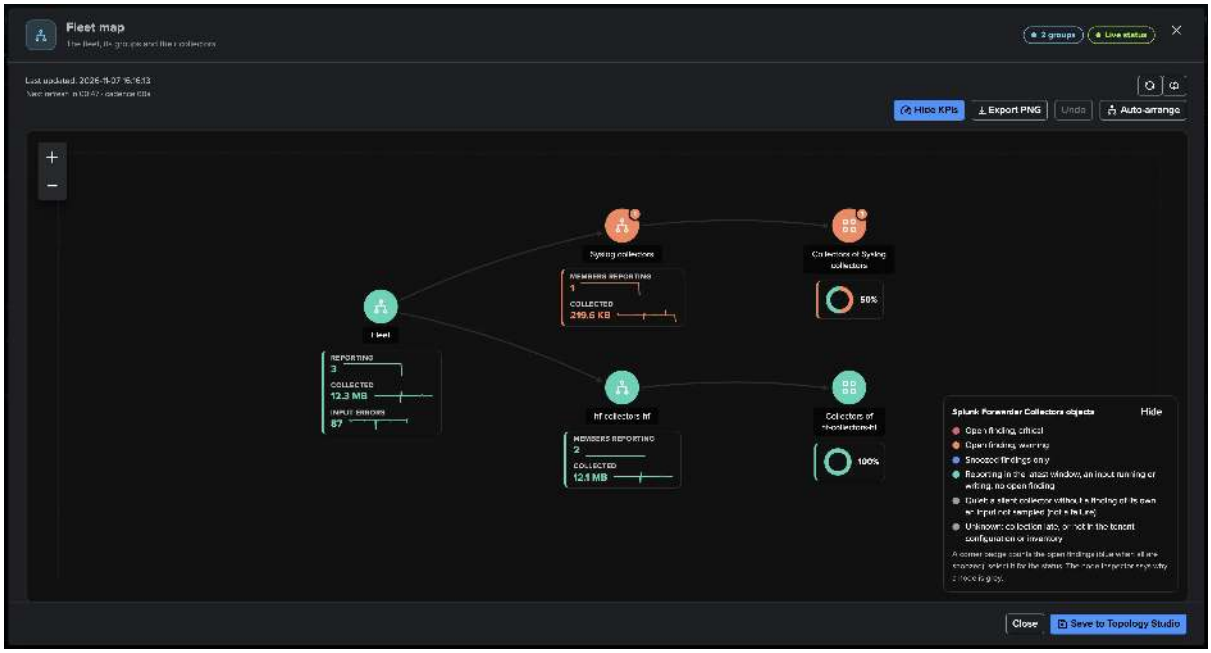
A collector opens in full and says what is wrong before anything else: its **Overview** lists each problem (a yellow or red health report, the lines stating an error, destinations failing, output pauses) with a button to the tab that explains it. **Health** shows each yellow or red indicator with its colour, measured value, threshold and Splunk's own reason; **Input errors** every error line by where it was written (on stderr, read by the level the line states itself, and in the inputs' own log files); **Inputs**, **Pipeline**, **Traffic** (data collected and forwarded apart, and what it sent per output group and destination), **Resources (OS)** and **Evidence** complete it.


hf-collectors-hf1
Collector · hf-collectors-hf
Collector
Expected
Window 15:20:00 – 15:25:00 · published 2 min ago
Overview
Health
Input errors
Inputs
Pipeline
Traffic
Resources (OS)
Evidence
Runs inputs of its own: a collector used as a collection point.
State
Reporting
in the latest window
Collected
22.12 KB/s
6.48 MB
Forwarded
22.12 KB/s
6.48 MB
Inputs active
9 / 13
with evidence of this window, of those known
Input errors
40
on stderr and in log files
Worst queue
0.1%
90th percentile
Needs attention
40 error lines
25 on stderr, 15 in its inputs' own log files
See the lines
NOW
State
Reporting in the latest window
In / out
22.12 KB/s / 22.12 KB/s
CPU / memory
12.1% / 42.8%
Disk free
86.6% (/)
Worst queue
0.1%
Output paused
No
Inputs active
9 / 13
Processes running
1
Input errors
40
Health
Green
Splunk version
10.6.1.1
Last 5 minutes · as of 15:32:52
Collected 21.3 KB/s
Forwarded 21.5 KB/s
Inputs running 1
Lines stating an error 25
No queue blocked
No open finding on this collector
Close
Open group hf-collectors-hf
Stop monitoring
Forget
Add to expected



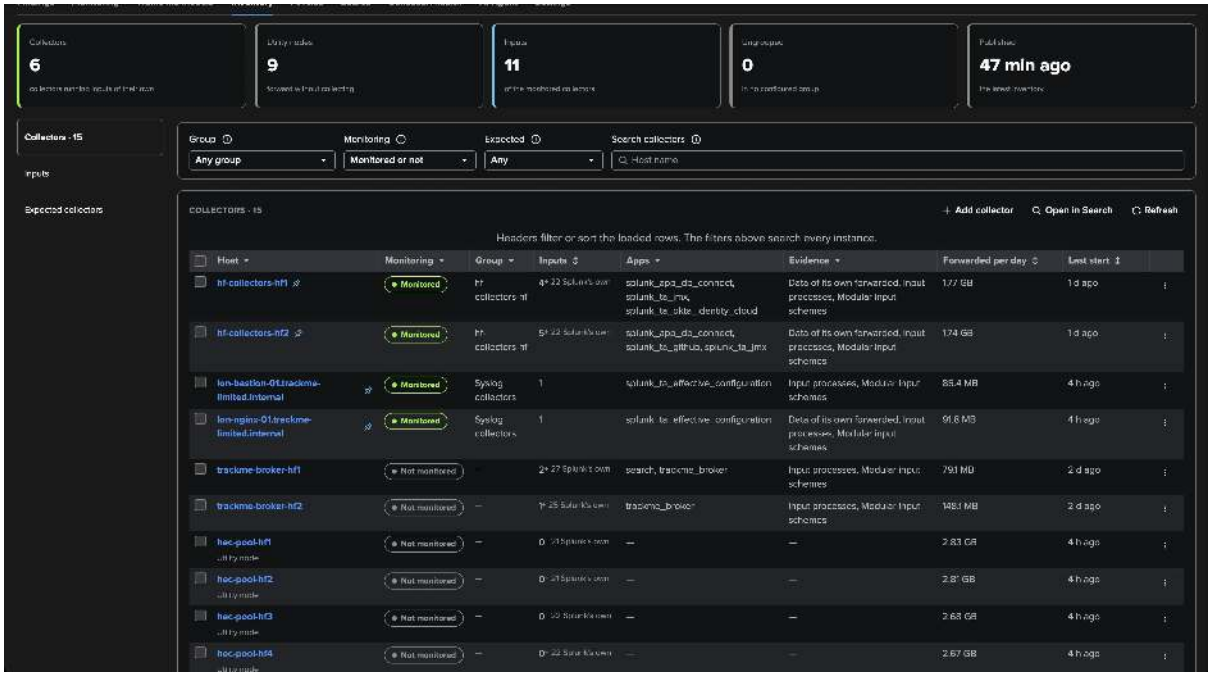
Input errors counts both places an input writes errors. On stderr, each line is read with the level it states itself (splunkd relays all of stderr as ERROR). In an input's own log file, words are matched in the line, so a routine line can count: the **Input errors** policy opens only when an input moves away from its own usual level.

Fleet map (a button of the Monitoring tab) draws the fleet, its groups and the collectors of each group on the Topology Studio canvas, with their status and their KPIs over time, and saves it as a view on request. A group appears while it has a monitored member, and a collector whose monitoring is stopped leaves the map. See *Topology Studio*.



Inventory

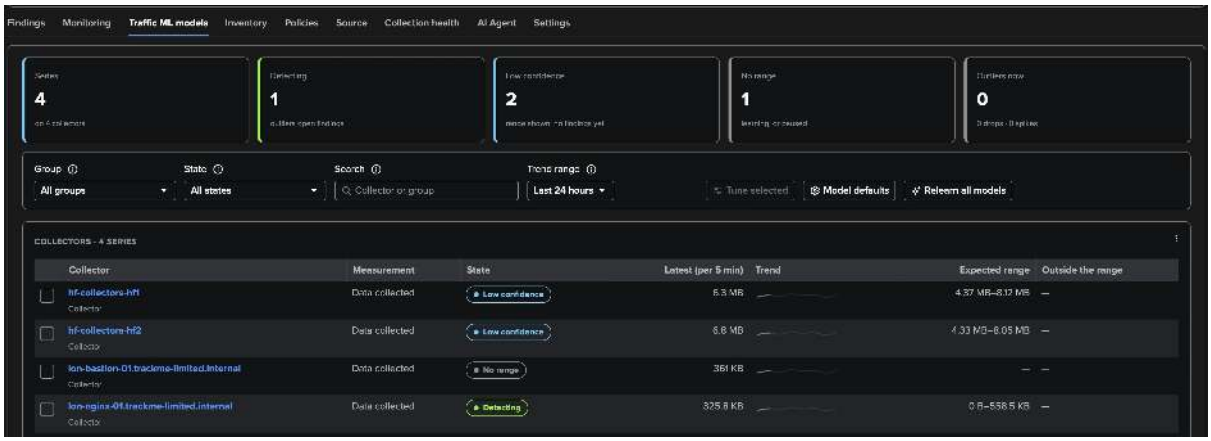
Inventory lists every instance the inventory published, the collectors and the utility nodes, with a left-hand navigation: **Collectors**, **Inputs** (must stay enabled, above) and **Expected collectors**. The **Monitoring** column says whether TrackMe follows an instance (*Monitored* or *Not monitored*), never whether the forwarder runs: its tooltip says who stopped monitoring it and when, or that it is a utility node, and selecting it opens **Stop monitoring** or **Resume monitoring**. A pin marks an expected collector.



Stop monitoring (the tenant's power and admin roles, audited) takes an instance out of the live reads, the policies and the Monitoring tab at once, and resolves its open findings; it stays listed in the Inventory. An expected collector cannot be stopped until it leaves the selection. **Resume monitoring** follows it again from the next live run. **Forget** retires a collector that left (silent), and **Add to expected** adds the selected collectors to the **Expected collector missing** policy.

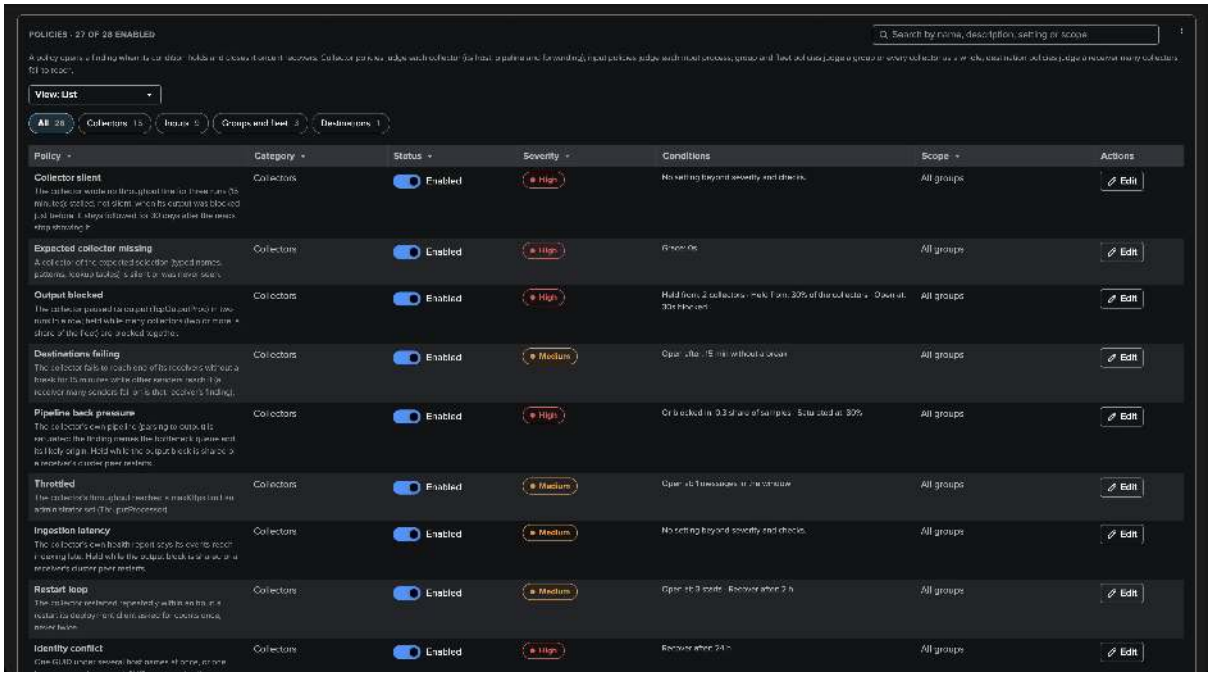
Traffic ML models

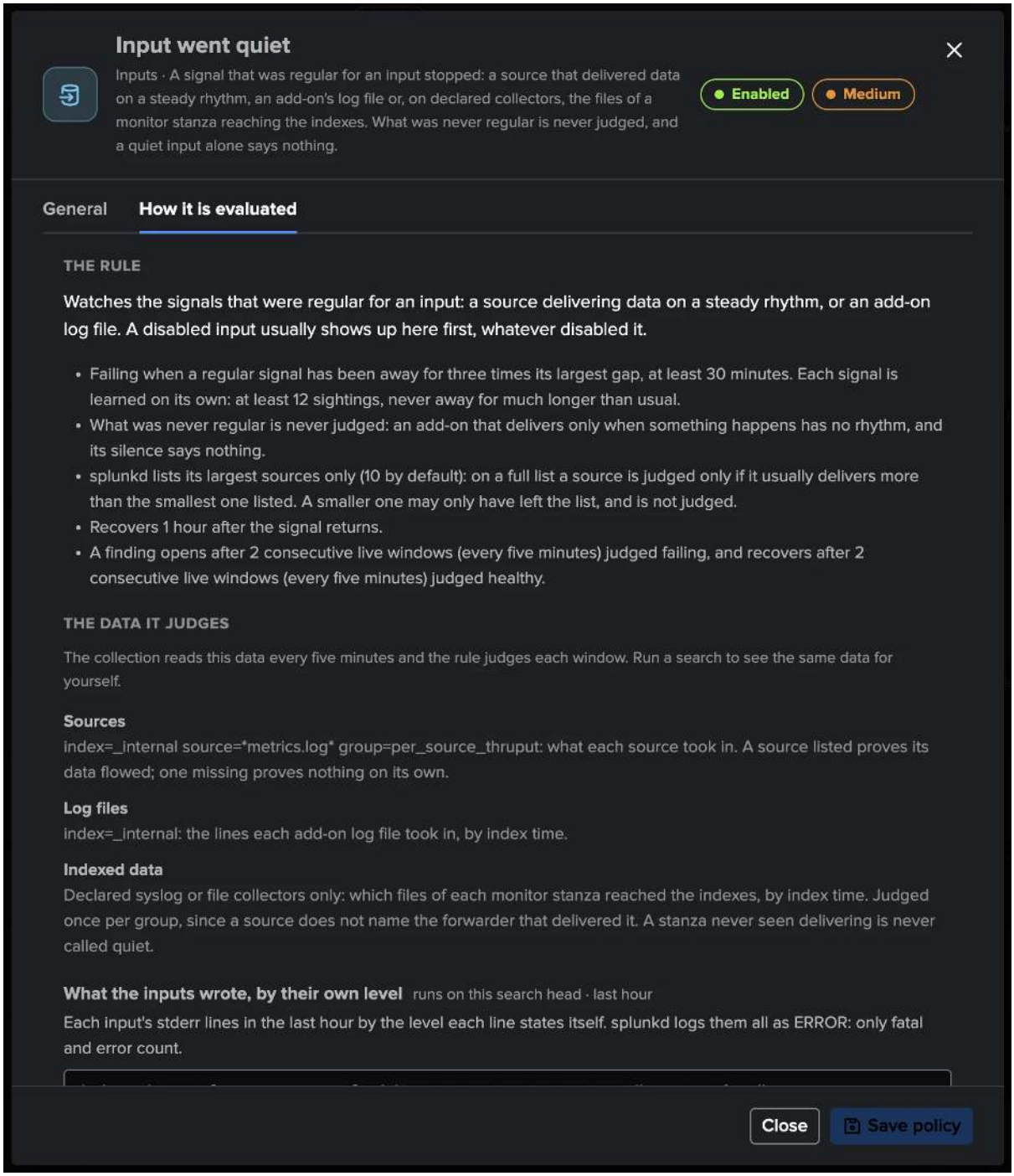
Each collector's collected volume has a **traffic model**: learned per hour of the week, it draws the expected range on the collector's chart and opens **Collected volume outlier** when the volume leaves it. The tab lists every series with its state (*Detecting*, *Low confidence*, *No range*), its latest value, its trend and range, with **Tune selected**, **Model defaults** and **Relearn all models**. A collector whose monitoring is stopped, or that was forgotten, is not listed, as in Monitoring.



Policies

Twenty-eight built-in policies cover the collectors, their inputs, the groups and the fleet, and the destinations; every policy is evaluated on each live window. Each has its severity, the windows it needs to open and to recover, a scope (**Groups**), a bank holiday hold, and a **How it is evaluated** tab: the rule as configured, the data it judges with ready-to-run searches on the tenant's own source, how to investigate and how to resolve.





The most useful policies to know first:

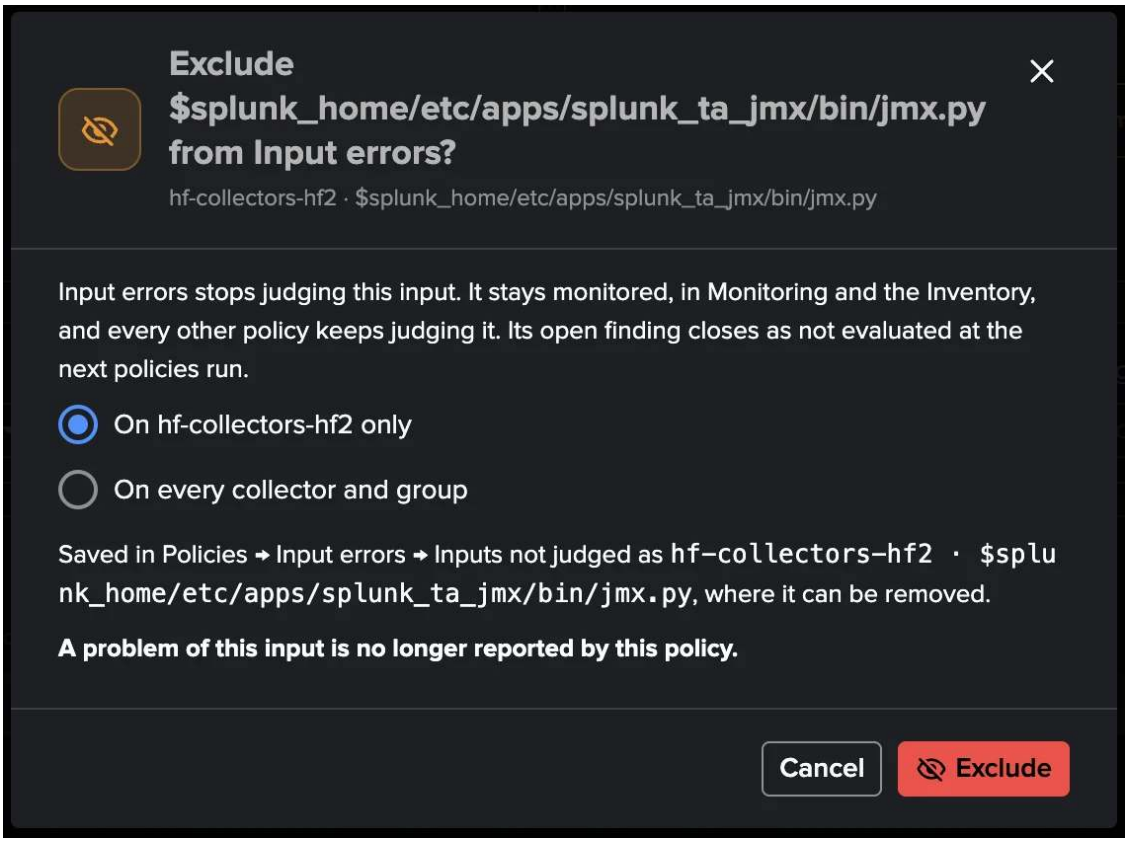
- **Collector silent** (high) — no throughput line from the collector in the live window; stalled, not silent, when its output was blocked just before.
- **Expected input not running** (high) — an expected input of a group does not run where it must.
- **Input process stopped** (high) and **Input crash loop** (medium) — a long-running input process stopped, or is restarted again and again.
- **Input errors** (medium) — an input writes errors on stderr, or in its own log file beyond its own usual level. An add-on that always writes such lines does not open it, and learning stops while it fails, so a lasting failure never becomes its usual.
- **Input went quiet** (medium) — a signal with a rhythm of its own stopped: a source delivering on a steady rhythm, an add-on's log file, or a syslog feed reaching the indexes. A rhythm must hold for a whole day, with

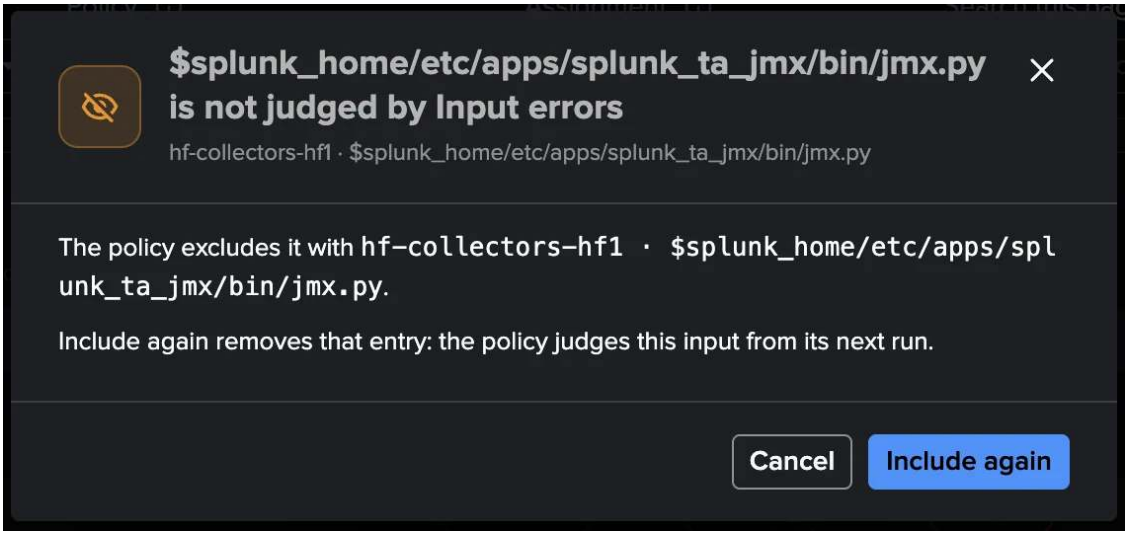
long gaps rare (at most one a day), and the finding waits three times the longest gap the signal ever had; what delivers only when something happens is away for long again and again, has no rhythm and is never judged.

- **Group below minimum** (high, critical when none is healthy) — fewer members healthy than the minimum: reporting, with their output not blocked. It judges only a group that sets a minimum.
- **Output blocked** (high), **Pipeline back pressure** (high), **Disk space** (high) — the collector's output paused, a queue of its own path saturated, its fullest partition filling.
- **Mass silence** (critical) — many collectors silent together: one finding, never one per collector.

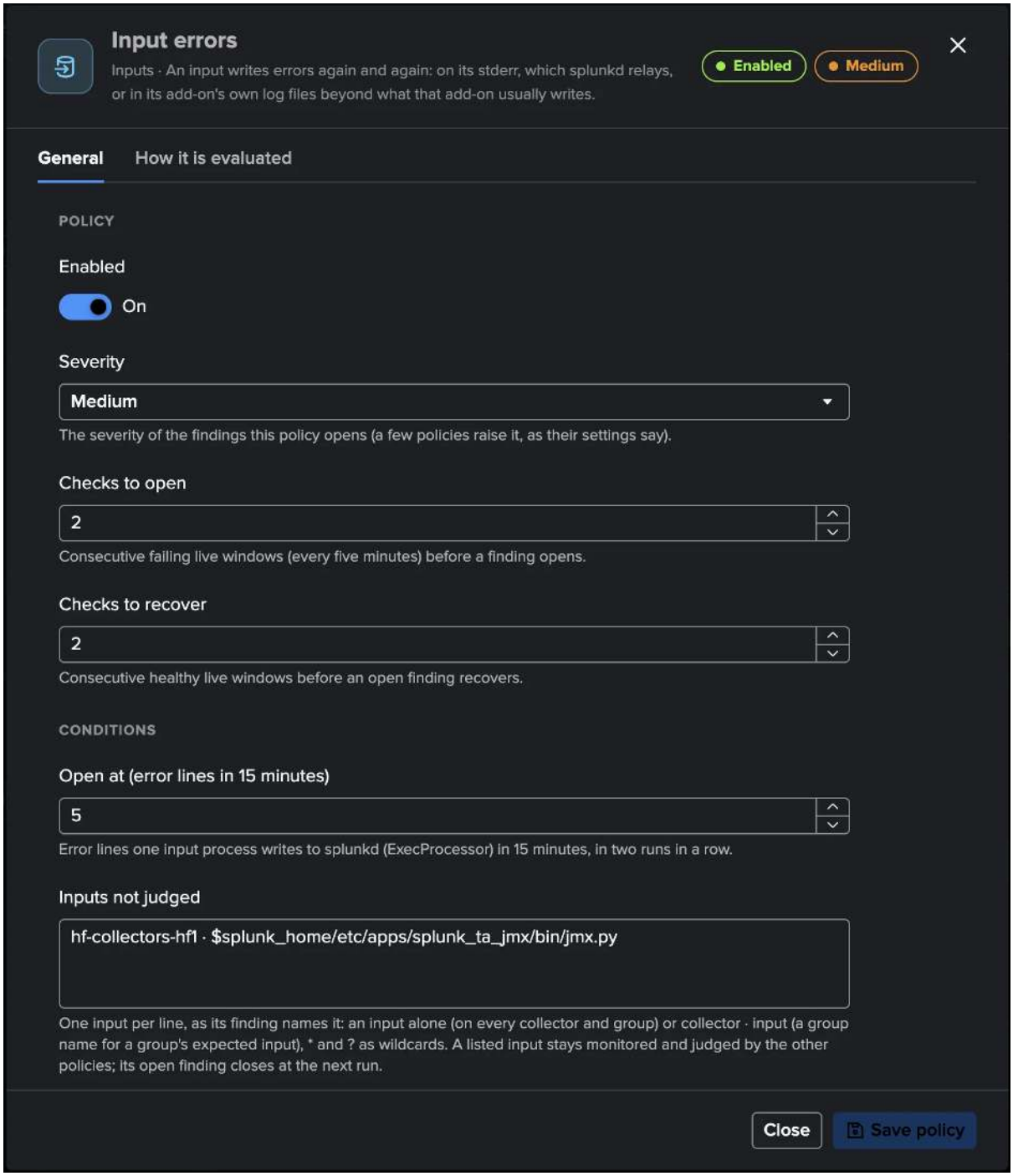
Every policy, its defaults and its settings are in *Built-in policies*.

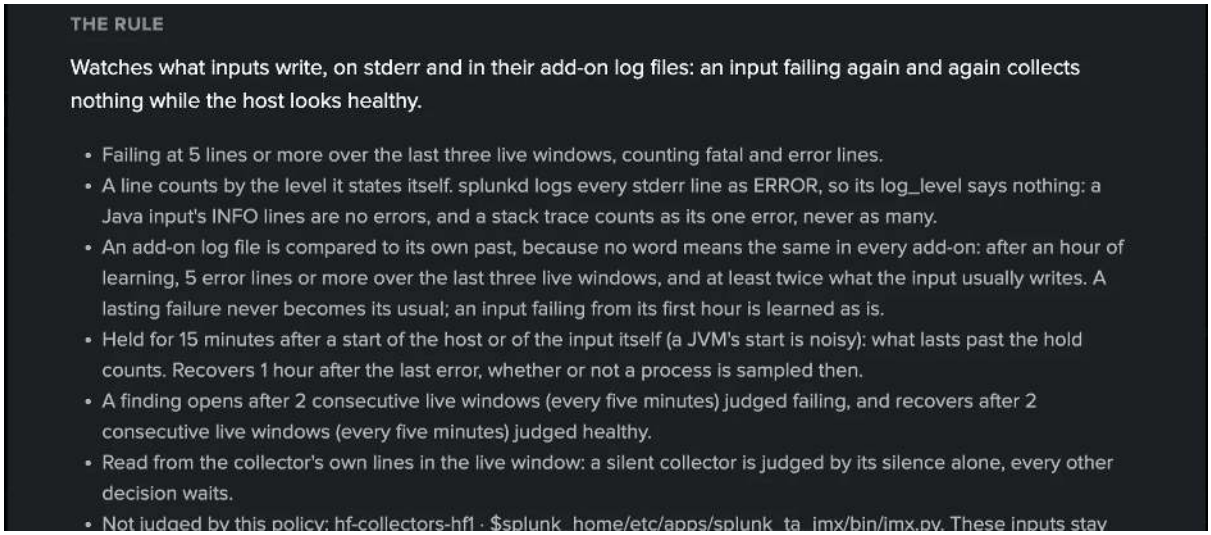
Stop a policy judging one input. When one input keeps raising false positives in one input policy (Expected input not running, Input errors, Input went quiet, Input process stopped, Input crash loop, Input resources), take it out of that policy only: in its finding's menu, *Exclude this input from this policy…* (TrackMe administrators of the tenant), on its collector only or on every collector and group. The input stays monitored, in Monitoring, the Inventory and the Fleet map, and every other policy keeps judging it. Its open finding closes as not evaluated at the next policies run, within five minutes. The entry is saved in the policy's settings, **Inputs not judged**, where it can be edited or removed, and **How it is evaluated** lists it; the same menu entry on an excluded input says which entry excludes it, with **Include again**. A problem of that input is then no longer reported by that policy: prefer it to disabling a whole policy, and to the tenant's *Excluded inputs*, which take an input out of every policy and of Monitoring.





In **Policies**, each of the six input policies lists its **Inputs not judged**, one per line, where they can be added, edited or removed, and its **How it is evaluated** ends with them:

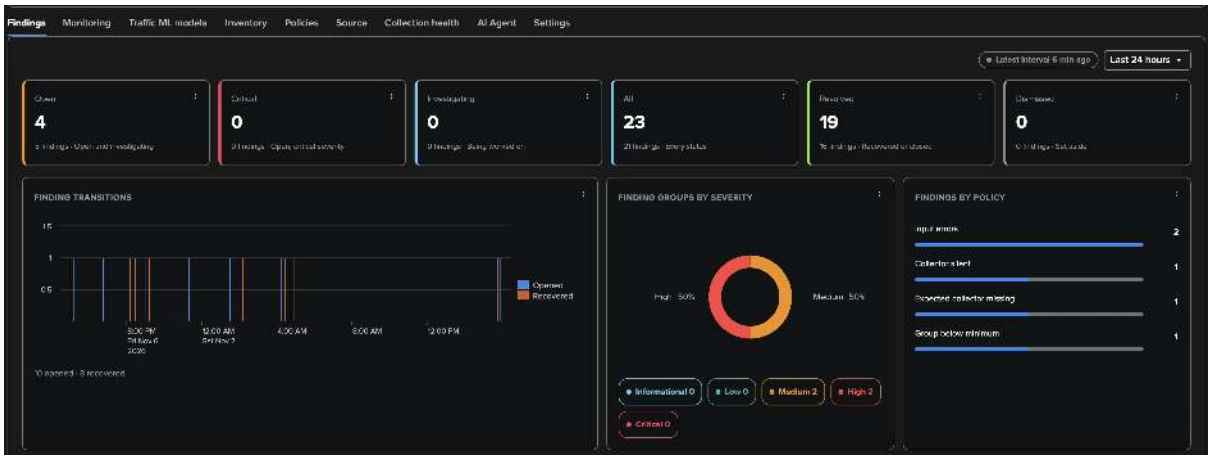

Input errors
Inputs · An input writes errors again and again: on its stderr, which splunkd relays, or in its add-on's own log files beyond what that add-on usually writes.
Enabled
Medium
General
How it is evaluated
POLICY
Enabled
On
Severity
Medium
The severity of the findings this policy opens (a few policies raise it, as their settings say).
Checks to open
2
Consecutive failing live windows (every five minutes) before a finding opens.
Checks to recover
2
Consecutive healthy live windows before an open finding recovers.
CONDITIONS
Open at (error lines in 15 minutes)
5
Error lines one input process writes to splunkd (ExecProcessor) in 15 minutes, in two runs in a row.
Inputs not judged
hf-collectors-hf1 · $splunk_home/etc/apps/splunk_ta_jmx/bin/jmx.py
One input per line, as its finding names it: an input alone (on every collector and group) or collector · input (a group name for a group's expected input), * and ? as wildcards. A listed input stays monitored and judged by the other policies; its open finding closes at the next run.
Close
Save policy



Custom policies. Above the catalogue, TrackMe administrators of the tenant write their own SPL: each result row names a collector, one of its inputs, a group or the fleet, and opens a finding like a built-in policy. See *Custom policies*.

Findings in practice

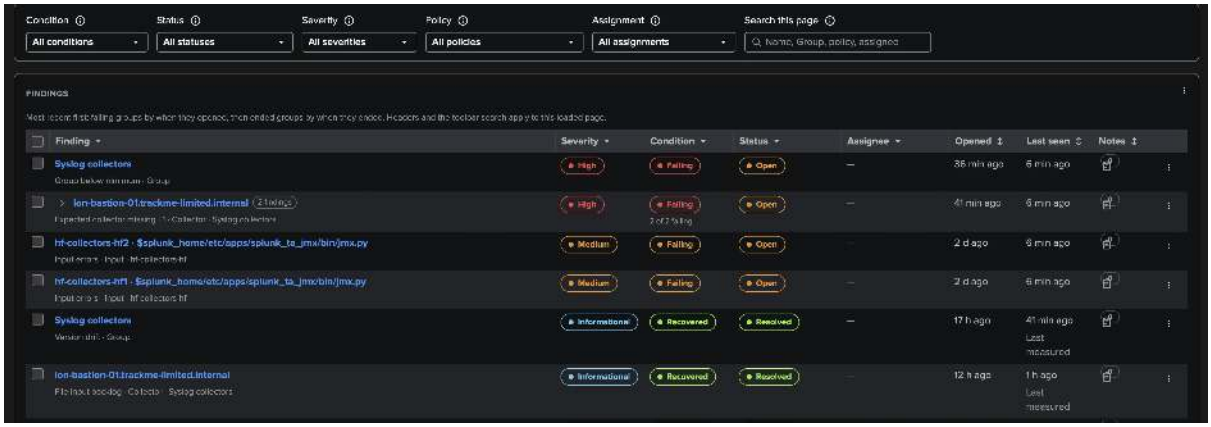
The **Findings** tab lists one row per Finding group, with the usual triage: status, assignee, severity, snooze, notes and the AI Agent. Above the list, the tiles count the open, critical, investigating, resolved and dismissed findings of the period, with the transitions over time, the Finding groups by severity and the findings by policy.



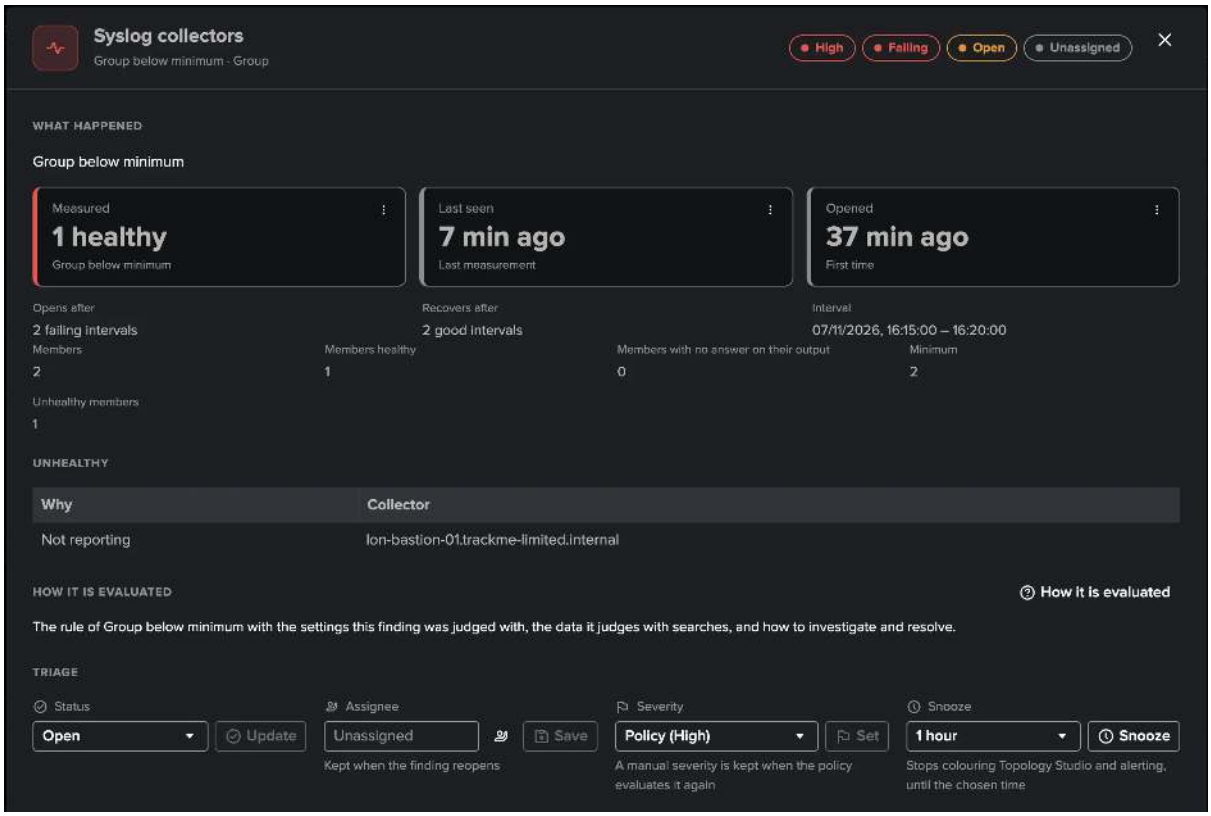
Here, one of the two syslog collectors of a group that needs both was stopped. Three findings follow, in two Finding groups:

- **Group below minimum** on the group *Syslog collectors*: one member healthy, the minimum is 2.
- **Collector silent** and **Expected collector missing** on the stopped collector: both are about the same collector, so they form one Finding group (*2 findings*, *2 of 2 failing*), triaged once.

The two **Input errors** rows are the JMX add-on's input on both add-on collectors: it writes its errors on stderr, read on each collector, so each collector has its own finding.



A finding opens on what it judged: for a group, its members, how many are healthy, the minimum and each unhealthy member with the reason (here *Not reporting*), then **How it is evaluated** and the triage. It recovers by itself after the policy's good intervals, once the collector reports again.



A syslog feed that stops while the collectors keep running opens **Input went quiet** once its rhythm is learned (a day of steady deliveries): one finding for the group and the monitor stanza, since a source does not say which member delivered it.

The AI Agent

The **Splunk Forwarder Collectors AI Agent** reviews findings on the shared findings AI Agent framework (see *AI Agents for findings*). It reads what TrackMe holds about the finding (the collector, its inputs and processes, its group, the latest window, collection health, the Finding group and past decisions) and may read further through its tools. It is taught that a process is not an input, that recovered is not a false positive, and that an input that must stay enabled and is disabled is a real problem: TrackMe corrects a review that says otherwise before it is shown, and says so in its summary. It writes one `[AI Agent]` note per review and, in act mode, applies only what the tenant allows. It never changes anything on a Splunk instance.

Scheduled triage is off by default (**Settings → AI Agent**, or *Manage AI Agents*): hourly, up to 10 findings per run.

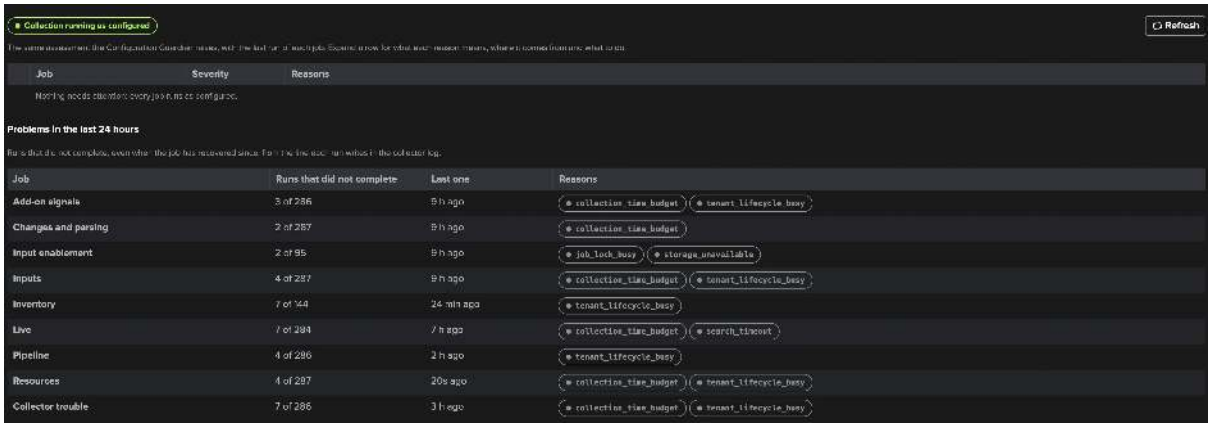
It needs the AI Assistant enabled and an LLM provider; a scheduled run works with a least-privilege tenant owner, not with `nobody`.

Collection health

The **Collection health** tab shows whether the tenant collects as configured: the **Status** view lists every problem with its meaning, source, causes and steps; **Trackers** shows each job's last run with the searches it sent, and **Run now**; **Data & storage**, **Troubleshoot & help** and **Custom policies** complete it. The same assessment raises the Configuration Guardian check `cribl_collection_health`.



Problems in the last 24 hours lists, per job, the runs that did not complete even when the job has recovered since, with each reason as a chip: hover it for its meaning, click it for what to do.



Troubleshooting

An input reads No activity evidence. Is it broken? Not on its own: a daily input between two runs reads this way. An input is judged by its findings: process stopped, crash loop, errors, went quiet, or not running when expected.

My syslog forwarder does not appear. A universal forwarder is monitored only when declared: **Add collectors** (Settings → Groups, or the Inventory) lists the ones the receivers saw sending in the last 15 minutes; one that sent nothing in that window can still be typed by name or pattern. A declared host that never reports shows nowhere: add it to the expected collectors once seen, so that its absence is reported.

A syslog feed reads No activity evidence on every collector. The tenant owner needs search access to the index the files go to.

An input of an add-on appears only as a process. The add-on keeps its inputs in a file whose name does not hold `inputs`: add an API account on the collector, or name the input as an expected input.

Input went quiet never opens on a new tenant. It needs a learned rhythm, a day of data.

Two groups share the same monitor stanza path. A source names no forwarder, so the indexed data of that path cannot be told apart between the groups: the stanza shows no indexed evidence and is not judged on it. Put those collectors in one group, or give the paths a part of their own.

What the Splunk Forwarder Collectors tenant does not do

- It does not judge the receivers, intermediates or pools: that is *Splunk S2S*; nor the fleet of forwarding agents: that is *Splunk UF Fleet*.
- It does not monitor a Deployment Server.

- It does not detect syslog or file collectors: you declare them.
- It does not judge one device of a syslog feed: what the indexes say of a feed is of the stanza. One device going quiet is a job for Data Host Monitoring.
- A universal forwarder sends no resource usage by default: no CPU, memory or disks for it, and no resource policy.
- Without an API account and with no configuration change within the retention of `_configtracker`, an input appears only while it delivers.
- Splunk lists the 10 largest sources per block by default: an input missing from Sources proves nothing unless it usually delivers more than the smallest source listed.
- It never changes anything on a collector.

Next: the reference

Every policy with its defaults and settings, the custom policy subjects, the jobs, the settings, the input states, the Collection health codes, the commands and the permissions are in *Splunk Forwarder Collectors reference*.

8.3.26 Splunk Forwarder Collectors reference

The reference of the Splunk Forwarder Collectors tenant: every built-in policy with its defaults and settings, custom policies, the collection jobs, the settings, the input states, Collection health, the AI Agent, the indexed finding events, the permissions and the commands. The guide is *Splunk Forwarder Collectors*.

Built-in policies

Twenty-eight policies in four categories. Every policy is evaluated on each live window (five minutes). Common settings of every policy:

- **Enabled** and **Severity** (Informational, Low, Medium, High, Critical; a few policies raise it, as their row says).
- **Checks to open** and **Checks to recover**: consecutive failing, then healthy, live windows.
- **Scope** (**Groups**): empty means every group of the tenant and the collectors no group claims (*Ungrouped collectors*, which can also be chosen).
- **Inputs not judged** (the six input policies: Expected input not running, Input errors, Input went quiet, Input process stopped, Input crash loop, Input resources): up to 200 inputs this policy no longer judges, one per line, as their finding names them: `collector · input` on one collector (`group · input` for a group's expected input), or the input alone on every collector and group; `*` and `?` are wildcards, every other character is literal. A listed input stays monitored and every other policy keeps judging it; its open finding closes as not evaluated at the next run. Set from a finding (*Exclude this input from this policy…*) or here. See *Stop a policy judging one input*.
- **During bank holidays**: keep opening findings, or hold (Collected volume drop, Collected volume outlier and Mass silence hold by default; see *Bank holidays*). Expected collector missing keeps opening: those collectors were selected as always reporting.

General rules:

- A **partial window** (a read that failed, or one Splunk finalized early) may open a finding, never close one. The policies that judge an absence or a count (Collector silent, Expected collector missing, Group below minimum, Mass silence, Collected volume drop, Expected input not running, Input process stopped, Input went quiet) hold a failing decision on a partial window.
- A silent collector is judged by its silence alone: every other decision on it waits.
- The policies judge the latest live publication only: when it is more than 20 minutes old, they judge nothing until the live run publishes again.
- A finding not evaluated for 24 hours closes as not observed.
- An *input* is what an operator configured; a *process* is what runs. One script may run every stanza of its modular input, so a process names no input.

Collectors

Policy	What it detects	Default	Severity
Collector silent	No throughput line from the collector in the live window. Stalled, not silent (held), when its output was blocked in the last 15 minutes: its own telemetry queues behind its data. Held while a Mass silence finding is open. A collector the reads no longer show stays listed as silent, with its finding, for 30 days.	On	High
Expected collector missing	A collector of the expected selection (typed names or patterns, lookup tables) is silent in the live window, or an exact name no instance of the inventory carries (never seen). Decides nothing until a selection is set.	On	High
Output blocked	The output paused (`TcpOutputProc` "paused the data flow") for at least the blocked seconds in the live window. Held while 30 % or more of the collectors, and at least 2, paused together (the receivers' problem), and while its output queue is the bottleneck of its path without a pause long enough yet.	On	High
Destinations failing	The collector fails to reach one of its destinations without a break (failures at most 10 minutes apart) for the open delay while other senders reach it (a destination many senders fail on is that destination's Destination unreachable finding).	On	Medium
Pipeline back pressure	A queue of the collector's own path (parsing to output) fills to the saturation at the 90th percentile, or is reported blocked in the blocked share of its samples. Names the bottleneck queue (the furthest downstream full queue) and its origin: the output waiting on the receivers, a stage on this host, or a surge from upstream (taking in more than 1.5 times its usual rate at this hour). Held while the output block is shared or a receiver's cluster peer restarts.	On	High
Throttled	At least the minimum of `ThruputProcessor` "has reached maxKBps" lines in the last 15 minutes: a throughput limit an administrator set.	On	Medium
Ingestion latency	The collector's health report shows the ingestion latency indicator red (yellow is shown and opens nothing). Held while the output block is shared or a receiver's cluster peer restarts.	On	Medium
Restart loop	At least the minimum of starts within an hour, on the larger of the collector's own starts and the restarts its Deployment Server asked for: a deployment wave restarts it once, whatever its lines repeat. Recovers after the recovery period without a start.	On	Medium
Identity conflict	One GUID reports from several host names, or one host name from several GUIDs, with periods that overlap in time (a cloned collector image). GUIDs that follow each other on a host are a rebuild and open nothing.	On	High
Version unsupported	The collector's version is below the minimum, past the grace on its current version. The lowest listed version is the minimum; a listed version of a release (major.minor) raises it for that release. Decides nothing until a minimum is set.	Off	Medium

Inputs

File input backlog, File input errors and Parsing errors are listed here and judged on the collector.

Policy	What it detects	Default	Severity
Expected input not running	An input that must stay enabled (Inventory → Inputs) on a collector it names, whatever its group, or a group's expected input on a member: disabled, or absent for the absent delay according to a source able to say so (an API listing, or the registrations since a start that was read, for a pattern naming a command). In a group whose members share their inputs: an expected input (and, with *Learn the shared inputs*, one a member ran in the last day) runs on no member; an input moving between members opens nothing. No source able to say reads *unknown* and holds, never absent. Held while the collector is silent and for 15 minutes after a start.	On	High; an input that must stay enabled is at least High
Input errors	At least the minimum of fatal and error lines over the last three live windows on the input's stderr, counted by the level each line states itself (splunkd relays every stderr line as ERROR; a stack trace counts as one error). Or, in its add-on's own log files, after an hour of learning: at least the minimum and twice what the input usually writes (learning stops while it fails). Held for 15 minutes after a start of the collector or of the input; recovers after the recovery period without an error.	On	Medium
Input went quiet	A signal that was regular for the input stopped: a source delivering on a steady rhythm, an add-on log file or, on syslog or file collectors, the files of a monitor stanza reaching the indexes (judged once per group). Away for three times the longest gap it ever had and at least the quiet minimum. A rhythm needs at least 12 sightings over at least a day, with long gaps (more than three times its usual one) rare, at most one per day: a late delivery, a nightly batch or a weekly maintenance keeps it watched, and the same pause never opens a finding. What was never regular is never judged, and a quiet input alone says nothing. Recovers after the recovery period.	On	Medium
Input process stopped	A long-running input process (sampled in at least the presence share of the live windows over at least 6 hours, with a process older than 15 minutes in nearly all of them) not sampled for the absent delay. A periodic input is never long-running. Held for 10 minutes after a start of the collector and while it is silent; an input excluded, or past the 50 listed per collector, is never judged.	On	High
Input crash loop	A long-running input process restarted by splunkd at least the minimum of times within the hour (its extra process ids).	On	Medium
Input resources	An input process at or above the CPU share of one core (90th percentile) or the memory size, or whose memory grew by the growth share over a reference at least 6 hours old (processes of 100 MB or more). A short process reporting no CPU reads unknown, never 0.	On	Medium
File input backlog	At least the minimum of files queued by the file monitor, rising over three live windows in	On	Medium

Groups and fleet

Policy	What it detects	Default	Severity
Group below minimum	Fewer healthy members than the minimum set on the group. Healthy means reporting in the live window with its output not blocked; the health report does not count. A member blocked just before it went quiet counts as blocked. Held while members whose output was not read could make up the minimum, or while the members are blocked along with the rest of the fleet. Judges only a group that sets a minimum.	On	High → Critical when no member is healthy
Mass silence	At least the open share of the collectors, and at least the minimum, silent together in the live window. Recovers below the recover share; holds the members' Collector silent findings. A collector the reads stopped showing earlier counts for neither side.	On	Critical
Version drift	Members of a group below the group's version baseline (the most common version among its reporting members, the higher on a tie) after the rollout grace. One finding per group listing the versions, never one per collector.	On	Low

Destinations

Policy	What it detects	Default	Severity
Destination unreachable	At least the shared minimum of monitored collectors, and the shared share of the destination's senders, fail to reach it without a break for the open delay (longer while its cluster peer restarts). One finding on the destination listing the senders; the collectors' own Destinations failing findings are held. Only monitored collectors count as failing senders: a receiver only universal forwarders fail on is no finding here.	On	High

Default settings

Each policy's checks to open and to recover (one check per live window, every five minutes) and its own settings with their defaults.

Policy (id)	Open / recover	Settings and notes (defaults)
Collector silent (fwc_host_silent)	3 / 2	No setting.
Expected collector missing (fwc_expected_host_miss	1 / 1	Host names or patterns; up to 20 lookup tables; grace 0 s
Output blocked (fwc_host_output_blocke	2 / 2	Blocked 30 s; held from 30 % of the collectors and 2 collectors
Destinations failing (fwc_host_destinations_	1 / 2	Open after 900 s
Pipeline back pressure (fwc_host_back_pressure	2 / 2	Saturation 80 %; blocked share 0.3
Throttled (fwc_host_throttled)	3 / 2	Lines 1
Ingestion latency (fwc_host_ingestion_lat	2 / 2	No setting.
Restart loop (fwc_host_restart_loop)	1 / 2	Starts 3; recover after 2 h
Identity conflict (fwc_host_identity_conf	3 / 2	Recover after 24 h
Version unsupported (fwc_host_version_unsup	2 / 2	Minimum versions (up to 20, one per line); grace 24 h
Host resources (fwc_host_resources)	3 / 3	CPU 85 %; memory 90 %
Disk space (fwc_host_disk_space)	2 / 2	Free 10 %; critical at 5 % (below the free share)
Configuration changed (fwc_host_config_change	1 / 1	Files deploymentclient.conf, db_inputs.conf, indexes.conf, inputs.conf, limits.conf, outputs.conf, props.conf, server.conf, transforms.conf (1 to 30 files); group changes within 60 minutes; settle after 24 h
Collected volume drop (fwc_host_volume_drop)	2 / 2	Drop 50 %
Collected volume outlier (fwc_host_volume_outlie	2 / 2	Measure KB collected; direction drops and spikes; at least 100 KB per five minutes; open from normal confidence (7 days learned; low confidence: 1 day)
Expected input not running (fwc_input_not_enabled)	2 / 2	Absent after 1,800 s; learn the shared inputs on; inputs that must stay enabled (up to 500, each on up to 50 collector names or patterns), set in Inventory → Inputs; inputs not judged
Input errors (fwc_input_errors)	2 / 2	Error lines 5; count warnings off; recover after 1 h; inputs not judged
Input went quiet (fwc_input_quiet)	2 / 2	Quiet for at least 1,800 s; recover after 1 h; inputs not judged
Input process stopped (fwc_input_process_stop	1 / 2	Long-running at 90 % of samples; absent after 900 s; inputs not judged
Input crash loop (fwc_input_crash_loop)	1 / 2	Restarts 3; recover after 2 h; inputs not judged
Input resources (fwc_input_resources)	3 / 3	CPU 50 % of one core; memory 1,024 MB; growth 50 % in 6 hours; inputs not judged
File input backlog (fwc_input_file_backlog	3 / 2	Files 100
File input errors (fwc_input_file_errors)	1 / 2	Files 3; recover after 24 h
Parsing errors (fwc_parsing_errors)	1 / 2	Lines 50; recover after 24 h
Group below minimum (fwc_group_below_minimu	2 / 2	No setting: the minimum is the group's (Settings → Groups).
Mass silence (fwc_fleet_mass_silence	1 / 2	Open 20 %; at least 2; recover 10 %
Version drift (fwc_version_drift)	1 / 1	Grace 24 h
Destination unreachable (fwc_destination_unreac	1 / 2	Open after 900 s (1,800 s while the peer restarts); shared from 30 % and 2 collectors

Custom policies

TrackMe administrators of the tenant write their own policies in the **Policies** tab, above the catalogue: an SPL search run on this search head as the tenant owner, on its own schedule. Each result row names one subject, which opens a finding like a built-in policy and joins the Finding group of its object.

Subject type	Result field	Resolves to
Collector	`host` (a host name or a GUID)	The monitored collector of the directory, with its group. A name the directory does not list is not monitored; a GUID on several host names is ambiguous.
Input	`host` and `input`	The collector as above; the input matched against the inputs it registered, as a command line or a pattern. A name only an add-on logs (a DB Connect input name) stands for itself; a pattern naming several inputs of the collector is ambiguous.
Group	`group` (a group id or name, or `ungrouped`)	A group of the tenant's settings. Another value is not monitored; a name two groups carry is ambiguous.
Fleet	None	The fleet.

A row whose subject is not monitored is skipped and the run's other findings recover; an unresolved row (ambiguous, or any row before the first directory) withholds every recovery of the run. A remote source is searched through `| splunkremotesearch` in the SPL. The examples offered are *DB Connect inputs failing* and *Collectors whose inputs log errors*. Drafts, published revisions and activation are managed independently, Splunk validates the SPL before every preview and run, and each policy's runs are in Collection health. See *Findings, Finding groups and findings actions*.

Collection jobs

Job	Schedule	What it does
Inventory (`inventory`)	Every 10 minutes	The Splunk instances of the host scope (roles and GUIDs; a universal forwarder only when a group declares it) and what they forwarded, indexed and ran over a day; with what the registrations job published, the directory: the collectors, the utility nodes, their inputs and groups. Rebuilt about hourly, sooner when the registrations moved, always on a manual run or after a saved setting.
Registrations (`registrations`)	Every 10 minutes	What splunkd registered (input commands, modular input schemes) and the starts, forward from its index-time cursor; a new collector back to its last start (up to 30 days), a slice at a time, carried over. Most runs send no search. Until every collector is read, the enablement decisions hold.
Live (`live`)	Every 5 minutes	The index-time window since the previous run (with catch-up): what each monitored collector took in and forwarded. Assembles the latest facts of the jobs below into the snapshot, writes the series, then evaluates every policy in the same search. Never builds the directory: a run before the first inventory waits for it (`fwc_directory_pending`).
Pipeline (`pipeline`)	Every 5 minutes	The queues, the output trouble, and what each collector sent per output group and destination.
Resources (`resources`)	Every 5 minutes	CPU and memory; the disks over the last 30 minutes.
Inputs (`inputs`)	Every 5 minutes	The running input processes and the lines they wrote, by the level each line states itself.
Add-on signals (`addons`)	Every 5 minutes	The lines and error lines of each add-on log file, what each source delivered; on syslog or file collectors, what reached the indexes from their monitor stanzas.
Changes (`changes`)	Every 5 minutes	The configuration changes, the parsing quality (the last 15 minutes) and the file backlog.
Collector trouble (`trouble`)	Every 5 minutes	The health report flags and indicators, the collectors' own trouble lines, the files of the flagged collectors only; the day-long read once an hour, read last.
Destinations (`destinations`)	Every 5 minutes, just before the live run	The destinations the senders fail to reach, their writers, and the cluster peers restarting.
Enablement (`enablement`)	Every 15 minutes	The configuration changes of the files defining inputs, replayed in time order (a new collector from 90 days back, a week at a time), then the optional API accounts.
Traffic ML models (`mltrain`)	Twice an hour	Learns each monitored collector's collected volume per hour of the week over the Model learning history, then the new buckets every 6 hours.
Policies (`policies`)	On demand	Evaluates the policies on the latest live publication.

Twelve reports are scheduled per tenant (the Policies report runs on demand): the most of any product, so a Splunk Cloud stack's search concurrency applies first here. Schedules are offset per tenant. **Collect now** (the status strip) runs the live job; **Run now** in Collection health runs a job; **Scheduled collection** (Settings → Collection) is on by default. The reports are `trackme_splunk_collector_<job>_<generation>_tenant_<tenant>`, and the AI Agent report `trackme_fwc_agent_tracker_tenant_<tenant>` (scheduled only once the scheduled triage is enabled).

Each job has its own report, time budget (its interval less 30 seconds), lock, receipt and Collection health line, and a search may use what is left of its job's budget: there is no fixed per-search timeout and no row limit. The fact jobs run in the minutes between two live runs; a job that misses a slot makes the policies hold, never count twice, and facts older than three runs of their job read as not read. Every read about the collectors names them, so a universal forwarder fleet never adds to its cost. With **Scheduled collection** off, a **Collect now** starts the other jobs beside the live run, and

it takes a second **Collect now** before the policies judge anything. The reports run `| trackmesplunkcollect` (log `trackme_splunk_collect.log`); several jobs of a tenant can overlap under the tenant owner, so a least-privilege owner may need a higher search quota.

Settings reference

Settings is one browser with a single **Save settings**. A tenant role that is not a TrackMe administrator of the tenant sees it read-only. Changing a field marked *starts a new source* restarts collection, and the findings, run history and traffic models of the previous source start again. Every policy threshold is a policy setting; the expected collectors are set in their policy and the inputs that must stay enabled in Inventory → Inputs.

Section	Setting	Default	Notes
Source	Source (search tier)	`local`	Fixed once the tenant exists: create another tenant for another deployment.
	Internal, Metrics, Introspection index	`_internal`, `_metrics`, `_introspection`	Fixed once the tenant exists; an empty Metrics index means `_internal` only.
	Telemetry form	Automatic	Automatic, `_metrics` mirror, or `_internal`; starts a new source.
	Host patterns	`*`	The Splunk instances that may be collectors, by host name (`*` and `?`, up to 100); starts a new source.
	Excluded hosts	None	Never `*` alone; starts a new source. Unchecking a collector's **Include** box in the discovery adds its exact name here (it stays listed in the discovery).
	Utility nodes	Left out	Licence managers, monitoring consoles and other full instances that forward without collecting: listed apart, monitored only when included; starts a new source.
	Included utility nodes	None	Utility nodes followed one by one while the others stay left out: host names or patterns, never `*` alone. Checking a utility node's **Include** box in the discovery adds its name here; starts a new source. A configuration saved before this field existed reads it as empty, so the upgrade never starts a new source.
	Repair workspace	—	Recreates the tenant's collections and reports; findings and settings are kept.
Groups	Groups	None	Up to 200. Each: a name (1 to 80 characters), host patterns (up to 50), expected inputs (up to 100 patterns over a command line, a modular input scheme or a `scheme://stanza`), an optional **Minimum healthy** (1 to 10,000), **Members share their inputs** (off) and **Syslog or file collectors** (off). A collector no group claims is ungrouped.
	Syslog or file collectors	Off	Declares the group's members as forwarders collecting other machines' data from files (rsyslog or syslog-ng writing to disk): TrackMe never detects them. Its host patterns need a literal part. A universal forwarder is monitored only in such a group. **Add collectors** (the wizard, Settings → Groups, the Inventory) lists the universal forwarders seen sending and declares the ones picked, or host names and patterns, into a group.
	How collectors join groups	By host patterns, and by hand	*By host patterns, and by hand*: a collector joins the first group whose host patterns match it, unless it was assigned by hand. *By hand only*: only the assigned collectors belong to a group.
	Assigned collectors	None	Up to 500 host names, each to one group, which it belongs to first whatever the patterns say: the discovery's **Group** column, a group's dialog, or the forwarders picked in **Add collectors**.
Inputs	Excluded inputs	Splunk's own inputs	Hidden from the views and never evidence of collection: up to 100 patterns over a command line or a modular input scheme; empty shows every input. **Re-**

Groups, assignments, the input exclusions and the API accounts apply from the next collection and never start a new source.

Input states

Monitoring → Inputs lists each input as configured, per collector; Processes lists what runs. The state is the strongest evidence of the window:

State	Means
Disabled	Its configuration says so (an API account when it lists the input, else its last configuration change).
Delivering	Its own source took data in, or, on a syslog or file collector, its monitor stanza's files reached the indexes.
Logging	Its own log file took lines.
Running, Writing	Its process was sampled, or wrote to stderr: on an input, only when it is the only input of that process.
Its process runs	A process ran: a process names no input, so this says nothing more of this one.
No activity evidence	Nothing in this window. **Never a failure on its own**: a daily input between two runs reads this way.

Collection health codes

The Collection health tab and the Configuration Guardian check `cribl_collection_health` (warning) report, while Scheduled collection is on:

Code	Severity	Meaning
`live_never_ran`, `live_not_running`, `live_failing`	Warning, Failure, Failure	No live run 20 minutes after configuration; none for 20 minutes; runs not completing for 30 minutes.
`inventory_never_ran`, `inventory_not_running`, `inventory_failing`	Warning, Failure, Failure	No inventory run; none for 3 hours; not completing for 90 minutes.
`job_never_ran`, `job_not_running`, `job_failing`	Warning, Failure, Failure	A fact job missed three of its runs (15 minutes for a five-minute job, 30 for Registrations, 45 for Enablement): never ran since the first assembled snapshot, or none since; its runs not completing for 30 minutes. Until its next run, the policies reading its facts hold.
`policies_failing`	Failure	Policy evaluation not completing for 30 minutes.
`models_failing`	Failure	Traffic model training not completing for 90 minutes: Collected volume outlier reads the last expected ranges, then its collectors read as learning.
`no_forwarders`	Warning	An hour after configuration, the source shows no Splunk instance in the host scope writing its metrics.
`no_collectors`	Warning	The last inventory found no collector running an input of its own: the instances of the scope are utility nodes, or run only Splunk's own inputs.
`api_read_failing`	Warning	An API account could not read the inputs of its collector: the telemetry speaks for it meanwhile.
`notification_delivery_failing`	Warning	Notification emails failing.
`custom_*`	Warning	Custom policies: schedule drift, failing runs, missed schedules, recovery required.

A run also states its reason on its receipt (Collection health → Trackers), such as `fwc_directory_pending` (waiting for the first directory), `fwc_live_stale` (the live window is more than 20 minutes old: the policies judge nothing) or `live_read_failed` (one read of the live run failed: its facts are missing from this window).

AI Agent

The Splunk Forwarder Collectors AI Agent settings (**Settings → AI Agent**, or *Manage AI Agents*):

Setting	Default	Notes
Scheduled triage	Off	Hourly when on (`ai_fwc_agent_enabled`).
Mode	Inspect	Inspect writes assessments and notes; act also applies the decisions allowed below.
AI provider	First available	
Findings per run, time budget per run	10, 3,000 s	
Review interval	1 day	A finding reviewed by a scheduled run is skipped that long unless it moved.
Allow investigating, allow temporary dismissal, allow resolve	Yes, yes (up to 7 days), no	A scheduled run never overrides an operator decision. It never enables, disables or restarts an input, never forgets a collector and never adds one to the expected collectors.
Custom instructions	None	Added to the agent's instructions; **Evaluate…** tries them on a finding.

An input that must stay enabled and is still disabled or absent is a real problem: TrackMe corrects a review that says otherwise before it is shown, and says so in its summary. The agent cannot dismiss that finding while it fails. Runs are audited with the sourcetypes `trackme:ai_agent:fwc_agent:inspect` and `trackme:ai_agent:fwc_agent:act`. See *AI Agents for findings*.

Indexed finding events

Every finding transition is written to the tenant's notable index as `sourcetype= trackme:cribl:finding` (and Finding groups as `trackme:cribl:finding_group`), with `object_category=splunk_collector`, whatever the notification settings. The tenant's series are in its metric index under `trackme.fwc.host`, `trackme.fwc.input`, `trackme.fwc.group` and `trackme.fwc.fleet`. The lookup `trackme_splunk_collector_collectors_tenant_<tenant>` holds one record per collector and utility node of the current directory, for `| inputlookup` or `| lookup` in your own searches.

Permissions

- Any role of the tenant reads the workspace.
- The tenant's power and admin roles triage findings, stop and resume the monitoring of a collector, and **Forget** collectors.
- A TrackMe administrator of the tenant (`trackmeadminoperations` and an admin role of the tenant) edits Settings, policies, groups, custom policies, the expected collectors (**Add to expected** is a policy save) and the inputs that must stay enabled (Inventory → Inputs, a policy save too).
- The scheduled runs search as the **tenant owner**: every remote account (the source, the API accounts) and lookup table the tenant reads must be usable by the owner, and on syslog or file collectors the owner needs search access to the indexes their files go to (without it, a stanza stays *No activity evidence*).

Commands

Command	What it does
`| trackmefwcdiscover`	The wizard's discovery (read-only, nothing saved): two searches of five minutes each, the instances of the host scope over the hour of their instance line, then what they forwarded, indexed and ran over the last 15 minutes. A preview, never the tenant's list. With `mode=api api_accounts="<accounts>"`, only the check of the collector API accounts: who each one is, which collector answers, the inputs it lists, and the read-only role that fixes a read it cannot do. With `mode=universal` (optional `name_filter`), the search behind **Add collectors**, planned but not run: one row per universal forwarder the receivers saw sending in the last 15 minutes (`tcpin_connections`, `fwdType=uf`), the busiest first, with its version, OS, receivers and volume, and no row limit; the dialog runs it as its own job and reads it 50 rows a page. `declared_hosts` lists the draft's declared hosts and patterns as collectors.
`| trackmefwctrend`	TrackMe's series of collectors, inputs, a group or the fleet (`kind=host|input|group|fleet`, up to 25 ids, `preset=1h|6h|24h`, up to four measurements), from the tenant metric index.
`| trackmefwcrecent`	The Last 5 minutes figures of the tenant, a group or a collector (`scope=tenant|group|host`): data collected and forwarded, blocked queues and inputs. Never a state, writes nothing.
`| trackmefwcevidence`	The raw lines behind a figure of a collector or a group's members (`kind=back_pressure|health|queue_history|input_messages|log_errors|parsing|config_changes|restarts` `range=3600|14400|86400`, `900` too for `queue_history`): a timeline, a breakdown and at most 50 lines, the input lines masked.

8.3.27 Hybrid trackers

A **hybrid tracker** is a scheduled tracker, created and lifecycle-managed by TrackMe, that runs alongside a component's default tracker. It can discover and manage entities and/or run a custom SPL check that contributes an extra signal — built from your own SPL, run locally or against a remote Splunk. Hybrid trackers are available for every component (DSM, DHM, MHM, FLX, FQM, WLK).

Note

This is one of the two advanced tracker types. To mirror entities from *another* TrackMe instance rather than build a signal from SPL, see *Replica trackers*.

When to use one

- The default tracker **cannot reach the data** — run the check against a **remote Splunk** via a configured remote account (see *Integrations & Remote Deployments*).
- You have **high-cardinality** or **custom break-by** entity sets that do not fit a single dynamic threshold.
- You want to **combine** the standard component logic with an extra per-entity signal (for example a CMDB lookup, or a remote row count).

Contribution modes

What a hybrid does with its result is the **contribution mode**:

Mode	Effect
score	Add a fixed score when the result breaches a threshold (composes cleanly with TrackMe scoring). The most common mode.
state	Force the entity's state directly, overriding the default evaluator.
anomaly_reason	Append to the entity's anomaly reason without changing state or score.
enrichment	Decorate the entity record with extra fields for the UI and downstream consumers.

Each hybrid runs on its own schedule; the UI shows its cron, last run, and last result. Manage them through the wizards or REST — **do not edit the generated saved searches directly.**

Note

- Keep a hybrid's query runtime well **under** its schedule interval, or runs will overlap themselves.
- Hybrid-tracker availability depends on edition: **Foundation** does not support them, **Enterprise** caps the number per tenant, and **Unlimited** is uncapped.
- A remote account's own RBAC must permit the caller — that is separate from the edition limits.

See also

- *Hybrid trackers — in depth* — the full creation wizard, search-mode tradeoffs, break-by/time mechanics, and the REST delete endpoint.
- *Replica trackers* — the other advanced tracker type: mirror entities from a remote TrackMe instance.
- *Integrations & Remote Deployments* — the remote accounts hybrid trackers rely on.
- *Monitoring Components* — hybrid trackers are available per component.
- *Impact scoring & thresholds* — how the *score* contribution mode composes.

8.3.28 Hybrid trackers — in depth

Tip

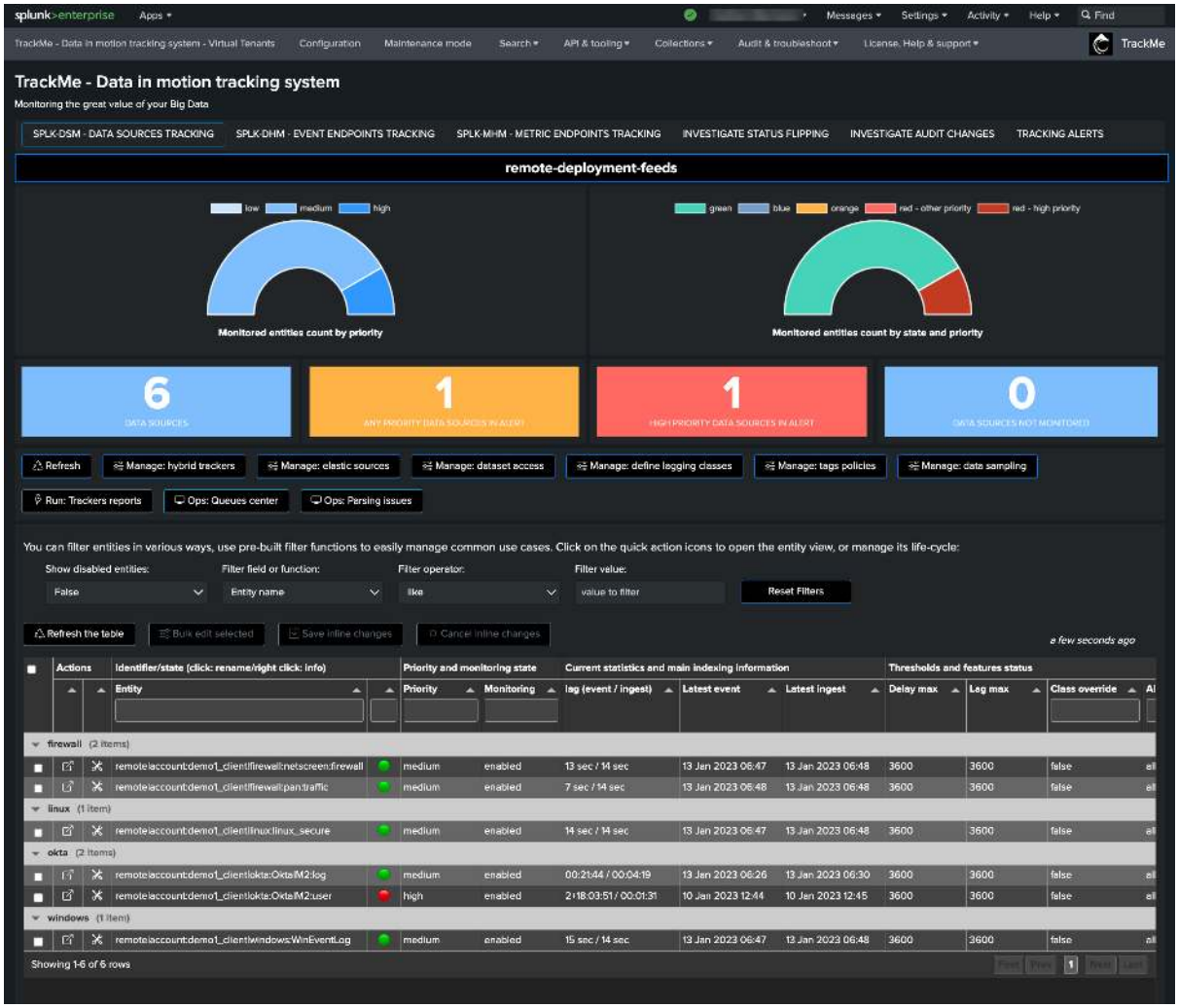
This is the **in-depth reference** for hybrid trackers. For the readable overview, start with *Hybrid trackers*. This page gives the full creation wizard, the search-mode tradeoffs (`tstats` / `raw` / `lookups`), the break-by and time-quantifier mechanics, deletion semantics, and the REST delete endpoint.

Introduction to Hybrid Trackers

Hybrid Trackers are created and managed through TrackMe. These are scheduled backend jobs that orchestrate entity discovery and management for TrackMe splk-feeds components:

- Hybrid Trackers are scheduled reports that involve various TrackMe backend tools depending on the TrackMe component
- A single Hybrid Tracker can discover and manage a few or many entities according to the needs
- Hybrid Trackers integrate into a main application workflow, which involves concepts such as registering their execution statuses and run time performance
- Hybrid Trackers can be created at any time through guided user interfaces or TrackMe REST endpoints
- In the context of Splunk feeds tracking, Hybrid Trackers can also be created during the initial creation of the Virtual Tenant

- When creating trackers, the related knowledge objects will be owned by the owner defined at the Virtual Tenant level
- TrackMe keeps records of the knowledge objects related to the Hybrid Trackers; therefore, you need to manage their lifecycle through TrackMe



How a hybrid tracker executes: the composite chain

A hybrid tracker is a **composite chain**, not a single search. Understanding it explains both what you see in the Splunk scheduler and where the knobs live:

- The scheduled tracker calls the `trackmetrackerexecutor` command, which runs a **wrapper** report.
- The wrapper calls an **abstract** report via `| savedsearch`.
- The abstract report holds the root `| tstats` search inline (its constraint comes from a per-tracker root-constraint macro), and finishes with the shared post-processing macro — `trackme_dsm_tracker_abstract(<tenant>, tstats)` for DSM (the DHM equivalent for hosts).

To validate any tracker's mode, open its `..._abstract...` / `..._wrapper...` report definition, or the Job Inspector's `normalizedSearch` — you will see the `| tstats` root and the index-time bounds.

The default window is narrow — wide windows are custom

By default, TrackMe creates DSM / DHM hybrid trackers with `earliest = -4h`, `latest = +4h`, and matching `_index_earliest = -4h` / `_index_latest = +4h`. The index-time bounds mean each run reads only roughly the last four hours of *indexed* data, regardless of the configured event-time window.

Trackers with wider ranges (say `-48h`) are therefore **deliberately created customs**, not the default. Creating them is legitimate and supported — for example to catch feeds with a permanent high-latency pattern the default 4-hour window would miss — they just need to be designed with intent, balancing the monitoring value against the extra scan cost.

Also remember TrackMe already maintains state for entities that fall outside the main window, automatically, through the **delayed-entities inspector** — it re-checks late / high-latency feeds over progressively wider windows (24h, 7d, then longer), so they do not require a permanently wide main tracker (see *DSM — Data Source Monitoring*). On long-lived deployments it is worth reviewing whether legacy wide-window custom trackers are still needed at all.

The span is the main cost dial — and it does not affect delay

The generated `tstats` search buckets data by a time **span**. A larger span means fewer buckets — a faster, cheaper search; a smaller span gives finer resolution. What matters is *what* the span affects:

- span changes the accuracy of the **latency** metric only (latency = `_indextime - _time`, measured per bucket);
- span has **no effect on delay detection** (delay = `now() - _time`), which depends only on the latest event time.

Early TrackMe versions generated a very granular span (`1s`); the current defaults are **DSM 30s**, and **DHM 5m** via the UI wizard (`1m` via the REST API). Reviewing the `span=` in your trackers is worthwhile: increasing it improves performance at only a small cost to latency resolution — you still catch the meaningful latency peaks.

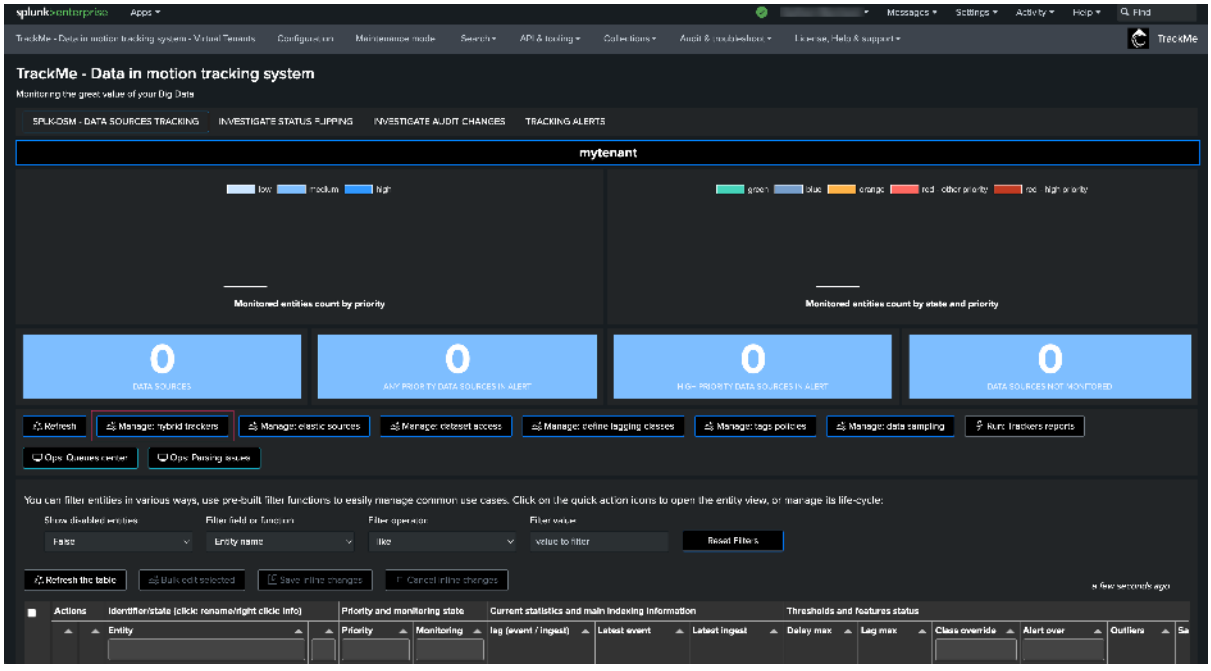
The UI also supports `span = none` when creating a tracker. This produces a tracker that does not bucket by `_time` at all (one snapshot per run): a substantial performance and cost win, with no effect on delay, but only a coarse, high-level latency figure. That trade is often ideal for **host / endpoint (DHM)** tracking, where delay is usually the priority — precise latency generally matters more for **data-source (DSM)** feeds than for endpoints.

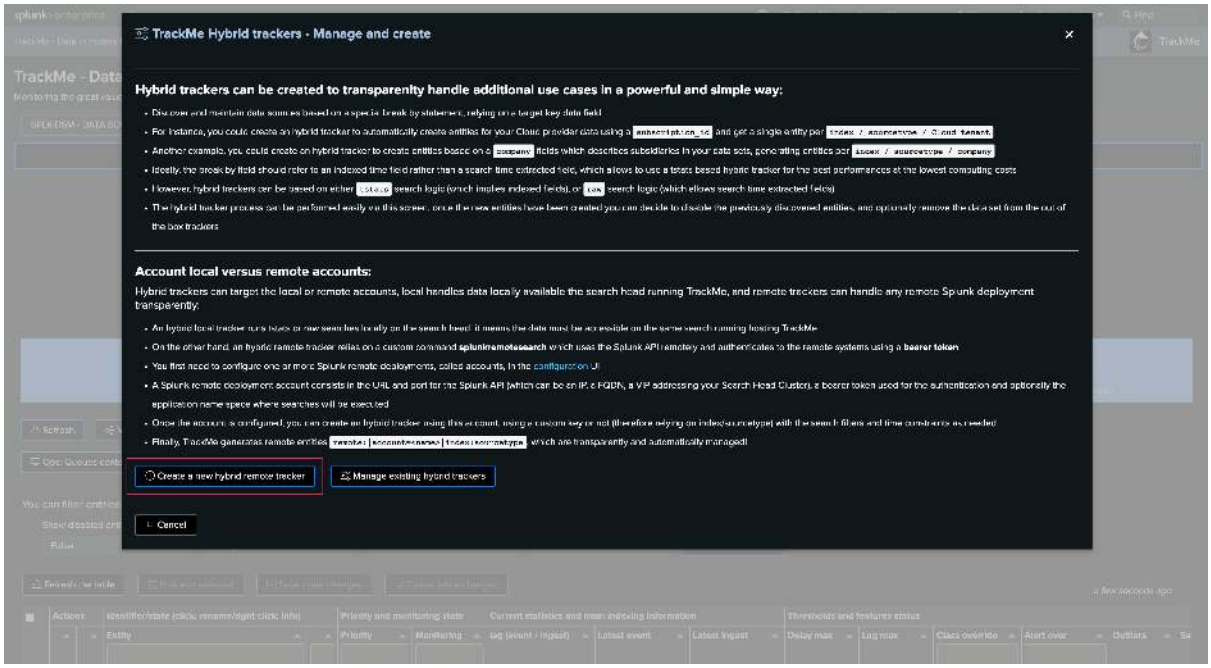
See *Delay vs latency* for the two KPIs themselves, and *Large-scale environments* for the broader scaling guidance.

Creating an Hybrid Tracker for splk-feeds

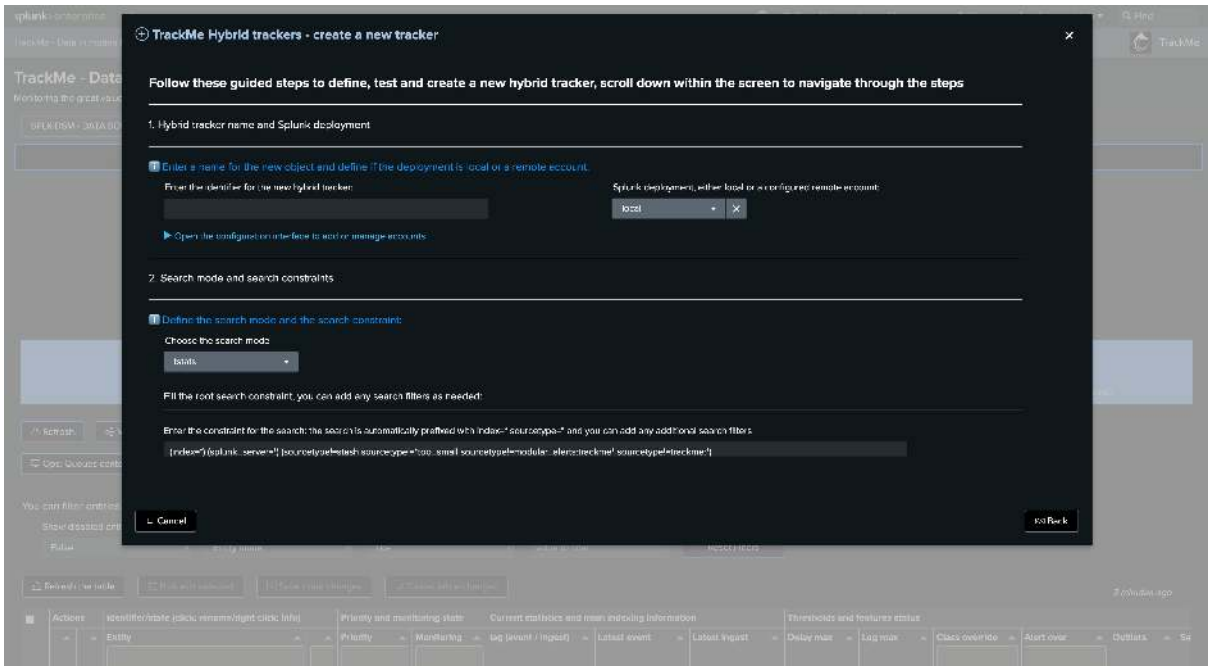
These instructions are related to the splk-dsm component. Options for splk-dhm/splk-mhm may differ, but the underlying logic is similar.

To create a new Hybrid Tracker, access the tenant and click on "Manage: Hybrid Trackers":





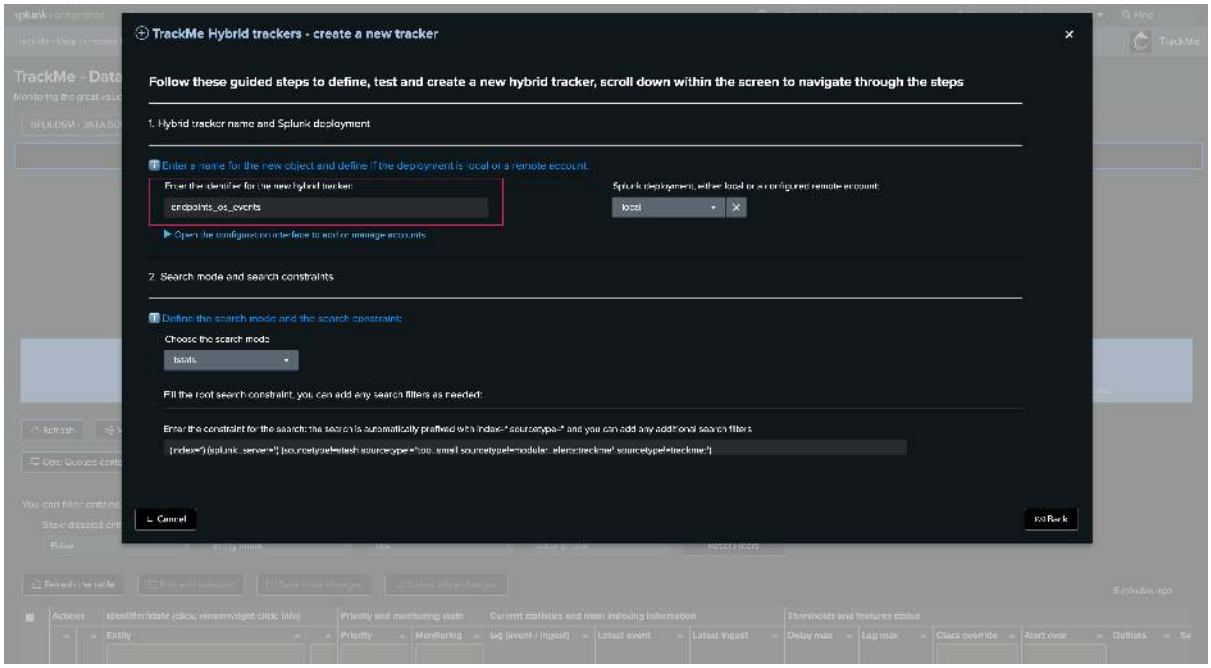
splk-dsm Hybrid Tracker creation wizard:



Once in the creation wizard, follow the guided steps:

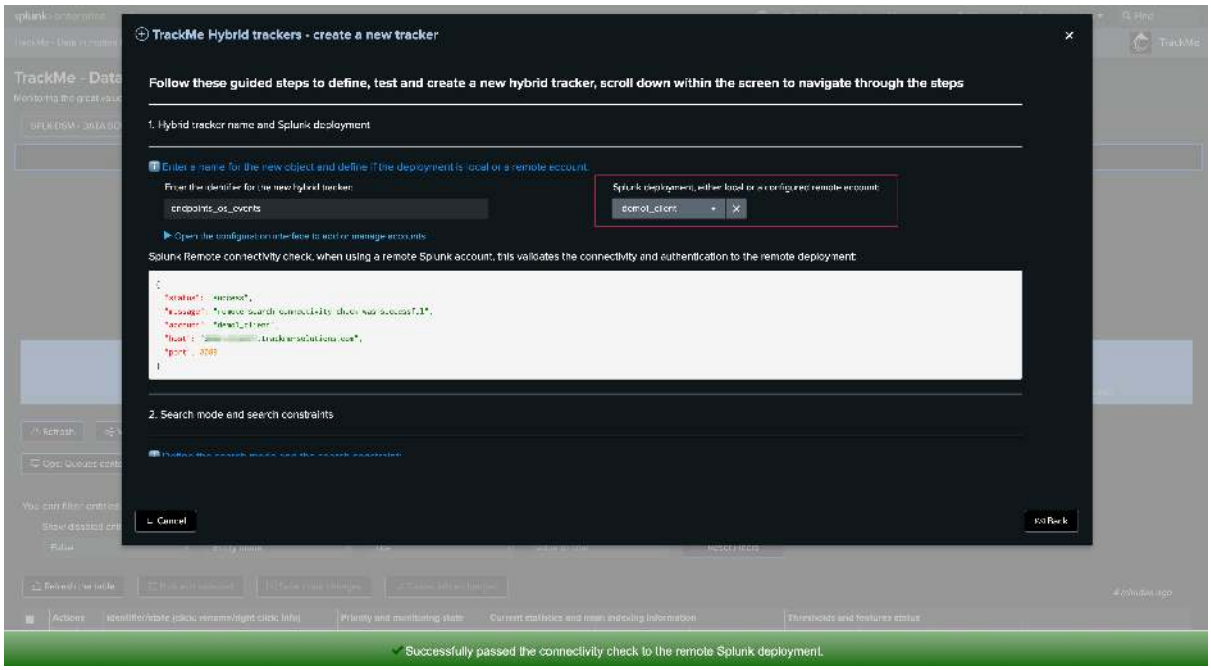
Hybrid Tracker identifier:

- Provide a name for the Hybrid Tracker. This will be included in the name of the Splunk Knowledge Objects related to this tracker
- In the example below, we will name our tracker “endpoints_os_data” as it deals with events originating from Operating Systems



Target Splunk deployment:

- Specify whether the data is searchable locally on the Splunk deployment or if the trackers deal with a remote Splunk deployment
- If a remote Splunk deployment is selected, TrackMe performs a connectivity check to that environment first



Search mode and search root constraint:

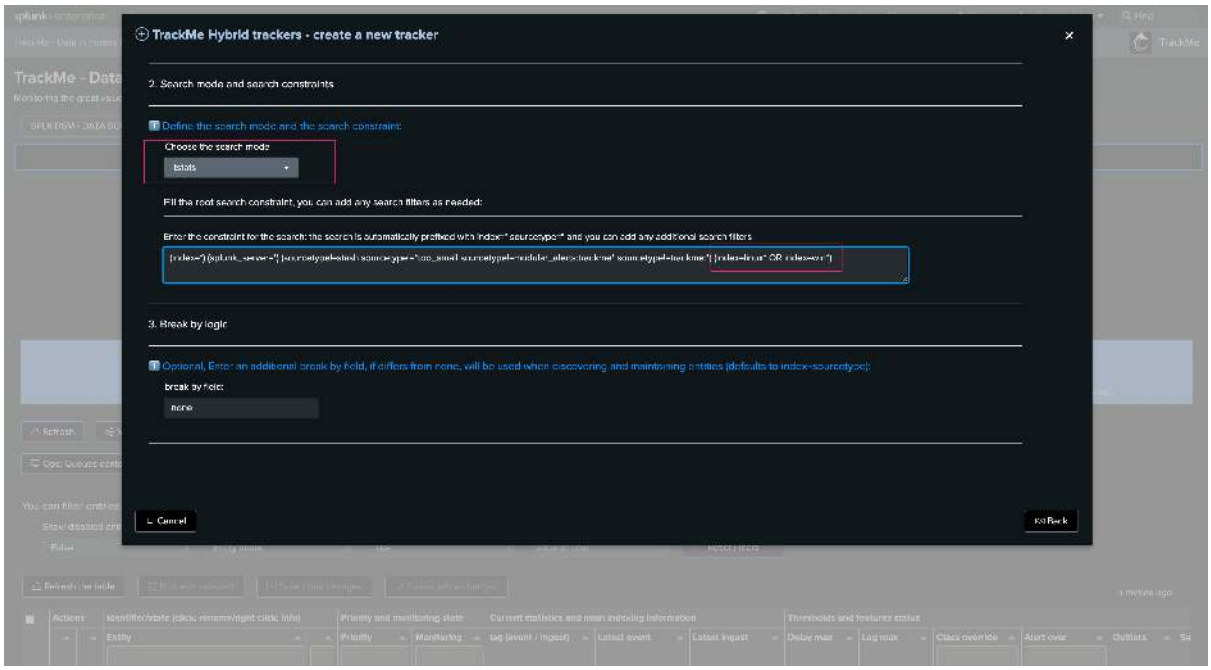
tstats versus raw versus lookups

- Then, define the search mode. You can choose between `tstats`, `raw` and (for splk-dsm only) `lookups`
- `tstats` is generally recommended as it provides much faster and more efficient searches relying on Splunk tsidx files
- However, tstats requires all fields to be indexed fields, while a raw search can deal with search-time extracted fields

- Therefore, raw search provides much more flexibility, but the cost is also much higher
- Depending on your context, raw searches may be fully valid, but if a tstats search can be used equally, use tstats
- `lookups` is a dedicated search mode for monitoring CSV files and KVstore collections as DSM entities — see the dedicated guide: *Lookup monitoring*. The wizard hides the SPL constraint editor when this mode is selected and replaces Steps 3/4 with lookups-specific inputs (app namespace glob, lookup name regex, lookup type, KVstore time fields, delay threshold).

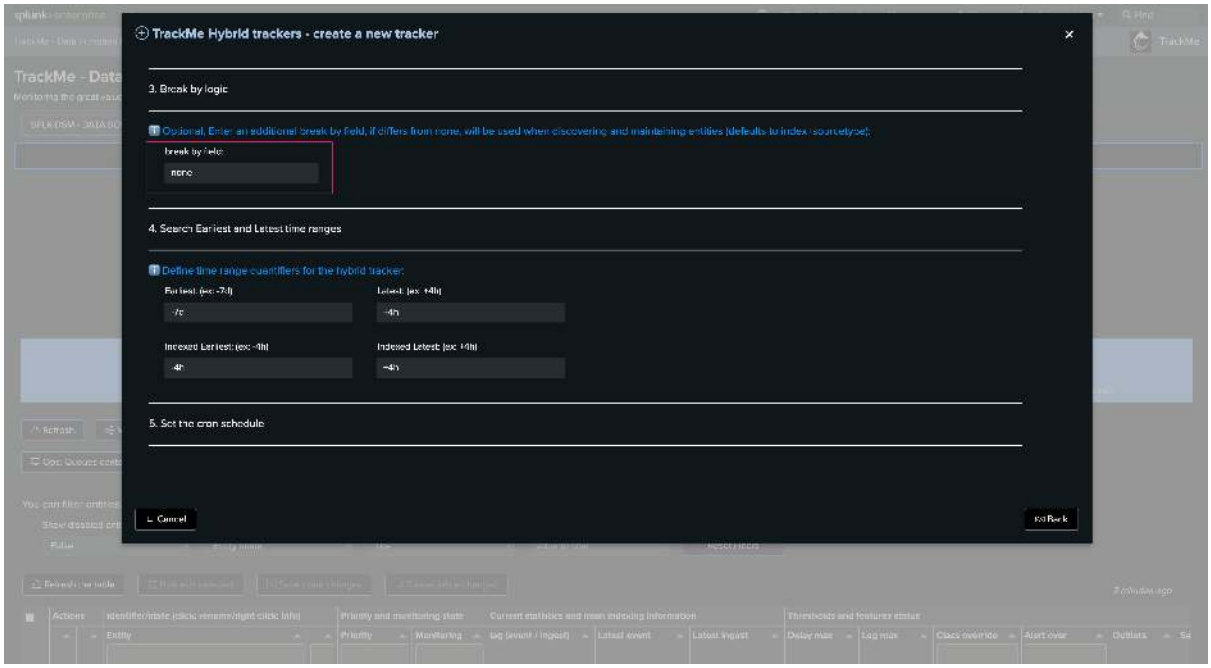
root search constraint:

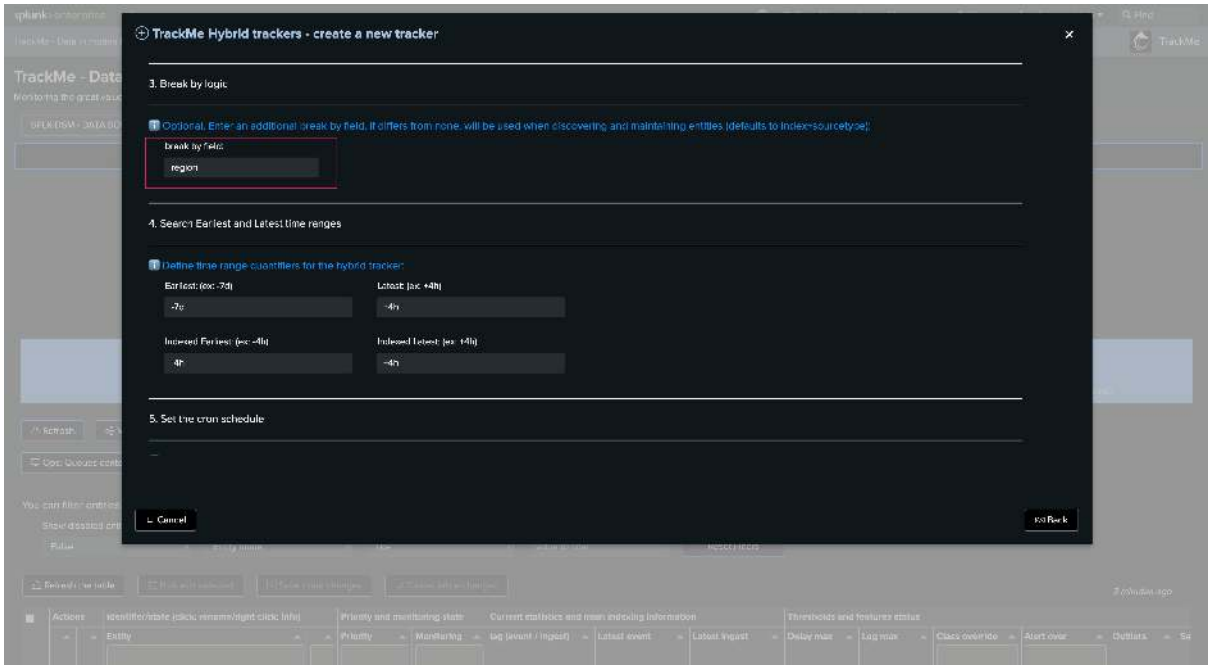
- Define the Splunk root search constraint. The constraint comes pre-filled with normally expected constraints which require valid data, exclude TrackMe related items, etc.
- Add your own search filters according to your needs. In our example, we add an index filter "(index=linux* OR index=win*)"



break by logic:

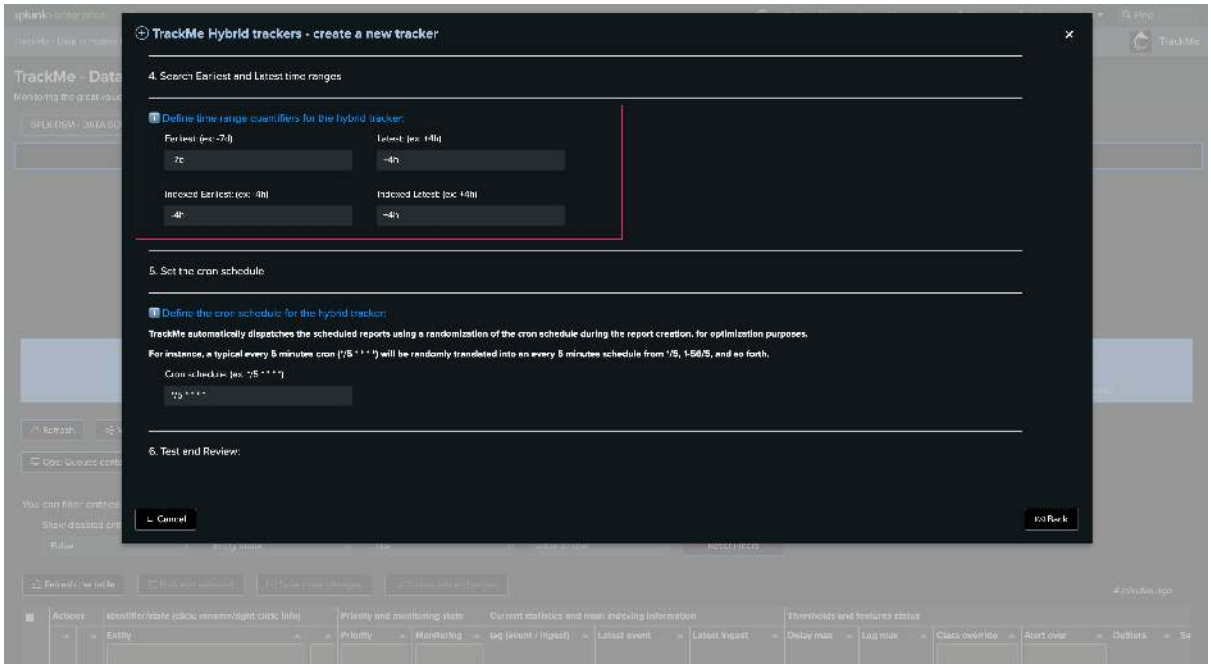
- You can optionally add an additional break by logic field
- This defaults to "none," which means entities are going to match the combo `index + ":" + sourcetype`
- For instance, if we have an indexed field `region`, we can leverage it here to distinguish entities per region. Our entity creation logic becomes `index + ":" + sourcetype + ":" + region`





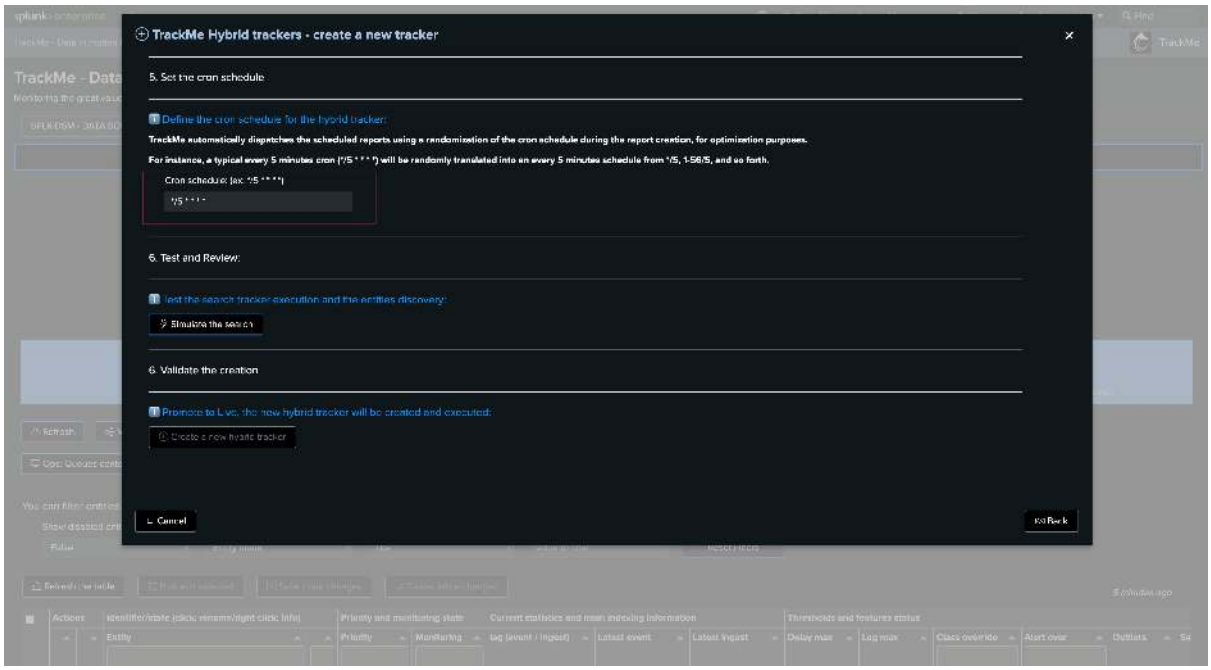
Time quantifiers:

- Review and update if necessary the indexed time earliest and latest, as well as time range earliest and latest
- These time quantifiers drive the period of data that the tracker is going to cover
- Generally, you will want to have a large event time range period to cover data with high latency, while the period for indexed time range can be more restricted for performance optimization purposes
- What will work best and be the most efficient depends a lot on your context and environment. Start with these values, review and adapt if necessary



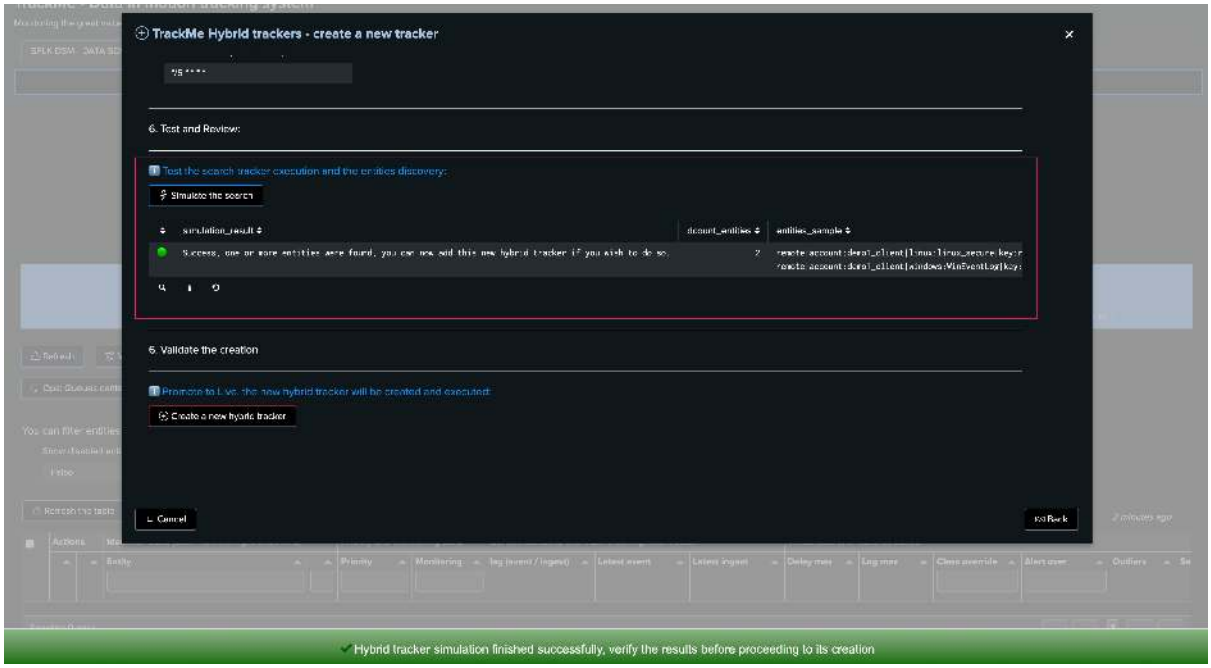
Cron schedule:

- Define the cron schedule for the Hybrid Tracker
- It defaults to every 5 minutes. Note that TrackMe will automatically dispatch cron schedules for optimization purposes

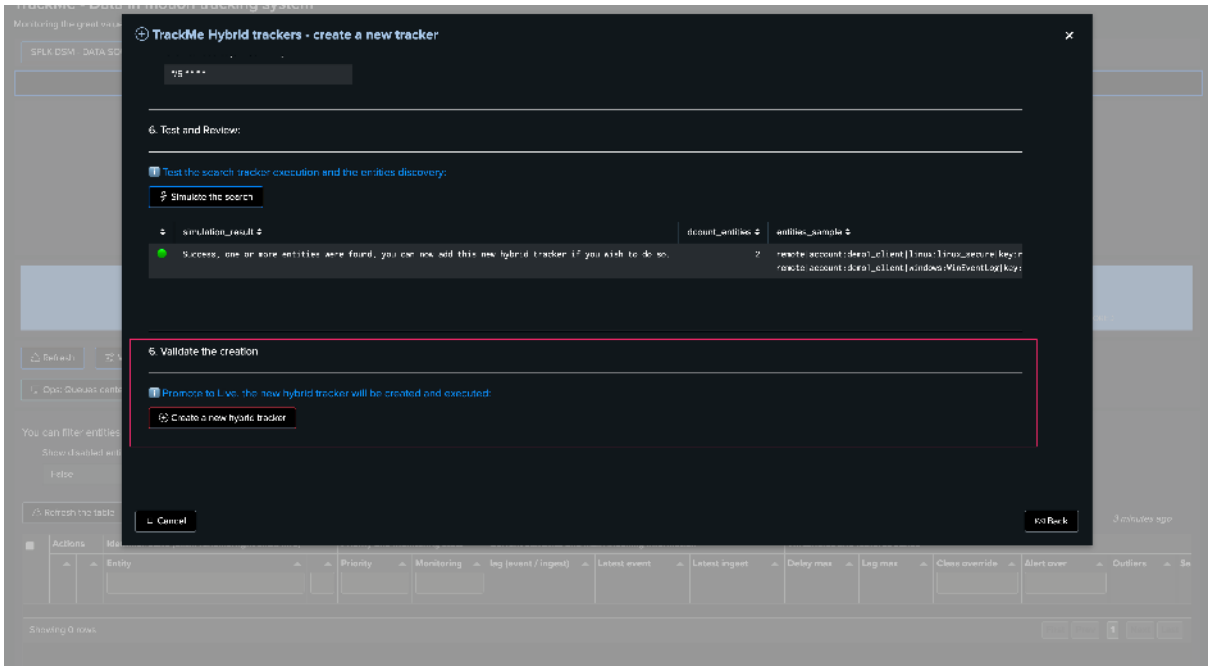


Test and review:

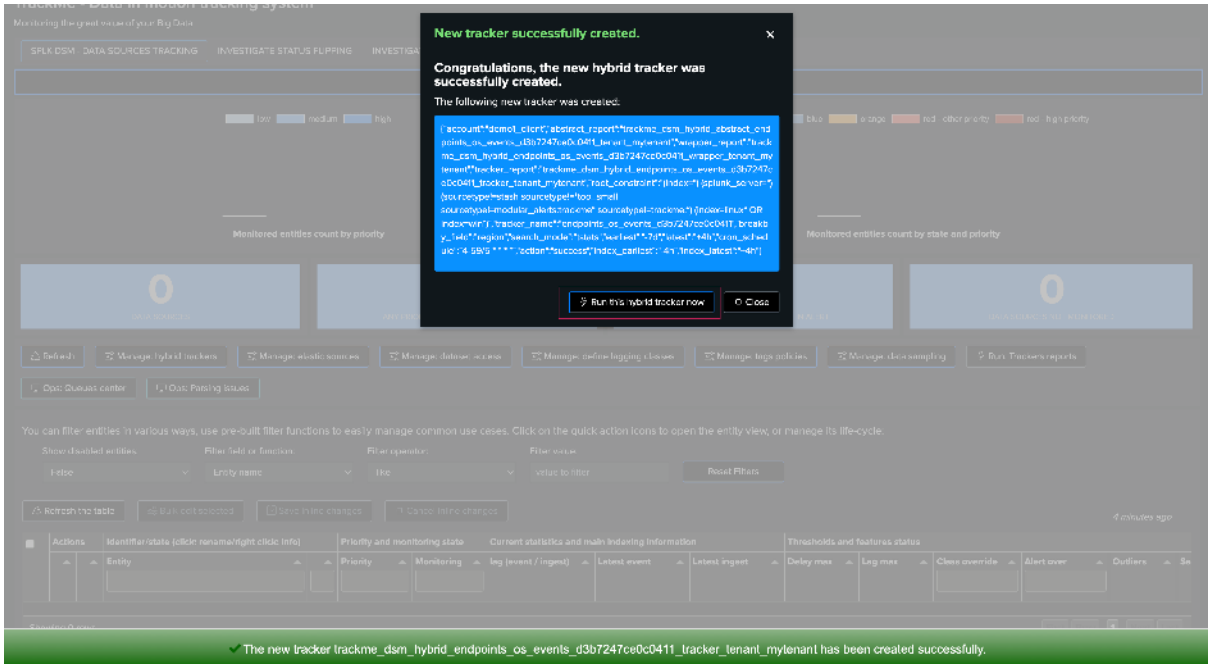
- Click on the button to execute the Hybrid Tracker in preview



Finally, validate the Hybrid Tracker creation:



Once created, you can choose to run the Tracker immediately to discover and create entities in the Virtual Tenant:



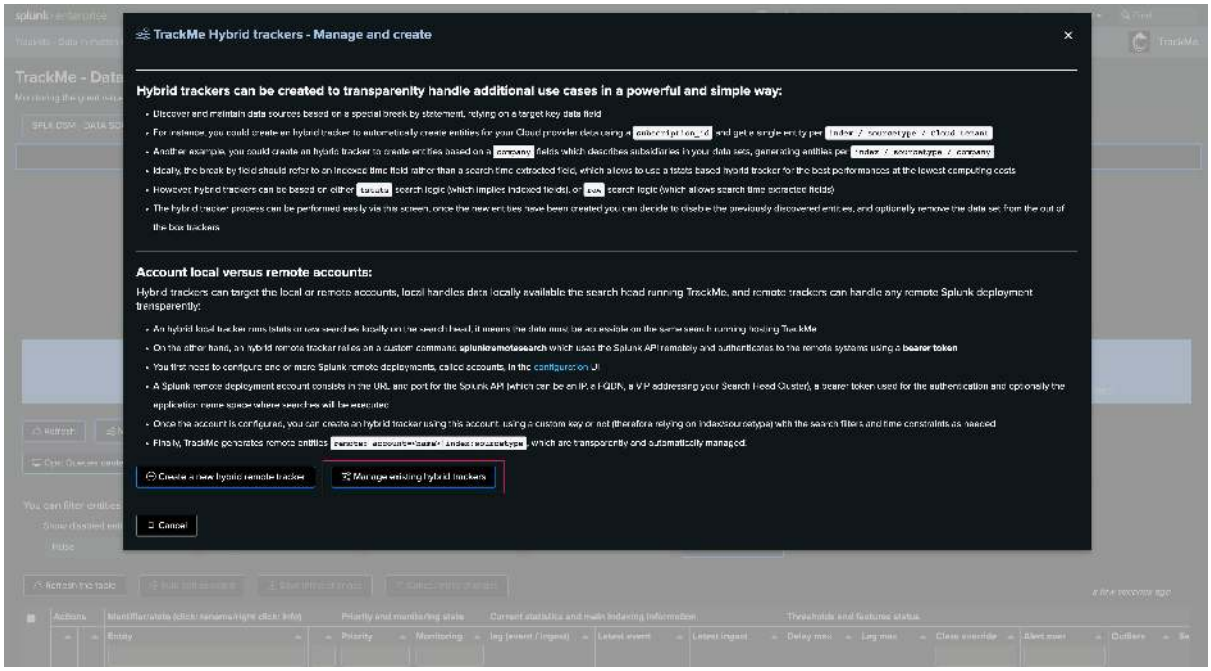
Managing Hybrid Trackers for splk-feeds

Deleting an Hybrid Tracker through the UI

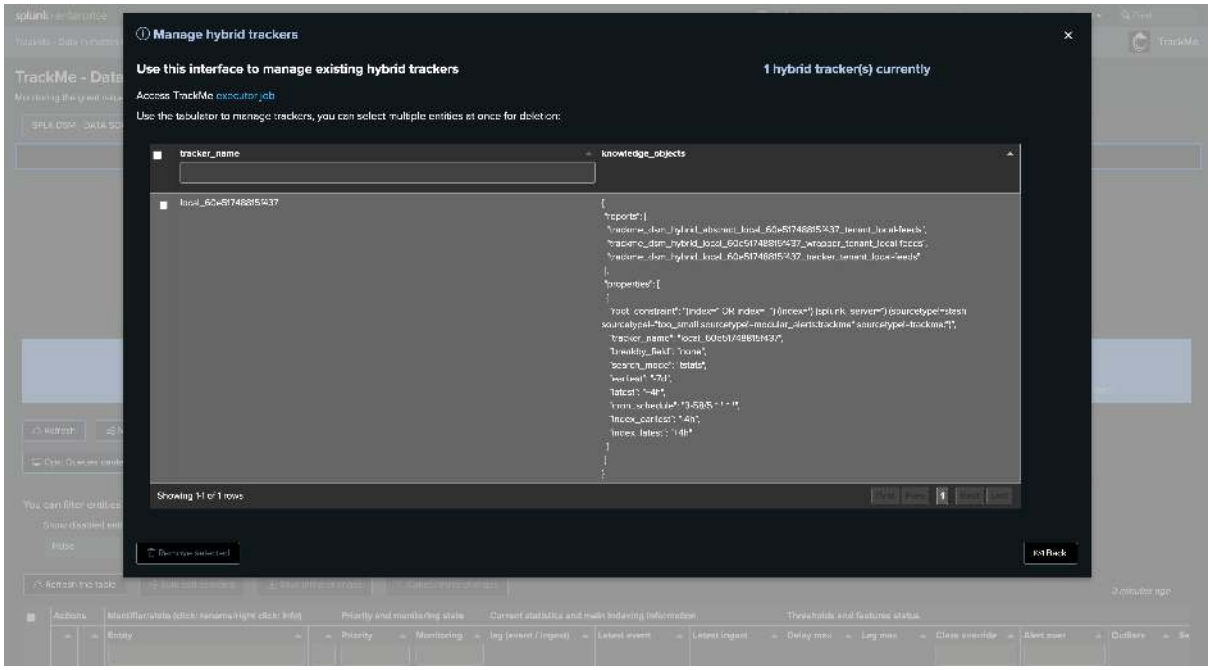
If you want to delete an existing Hybrid Tracker, this operation must be done via TrackMe.

The reason is that the application keeps track of all knowledge objects that were created for a given tenant to honor various features such as managing the lifecycle of the tenant (enabling/disabling, etc.) or the lifecycle of the tracker itself.

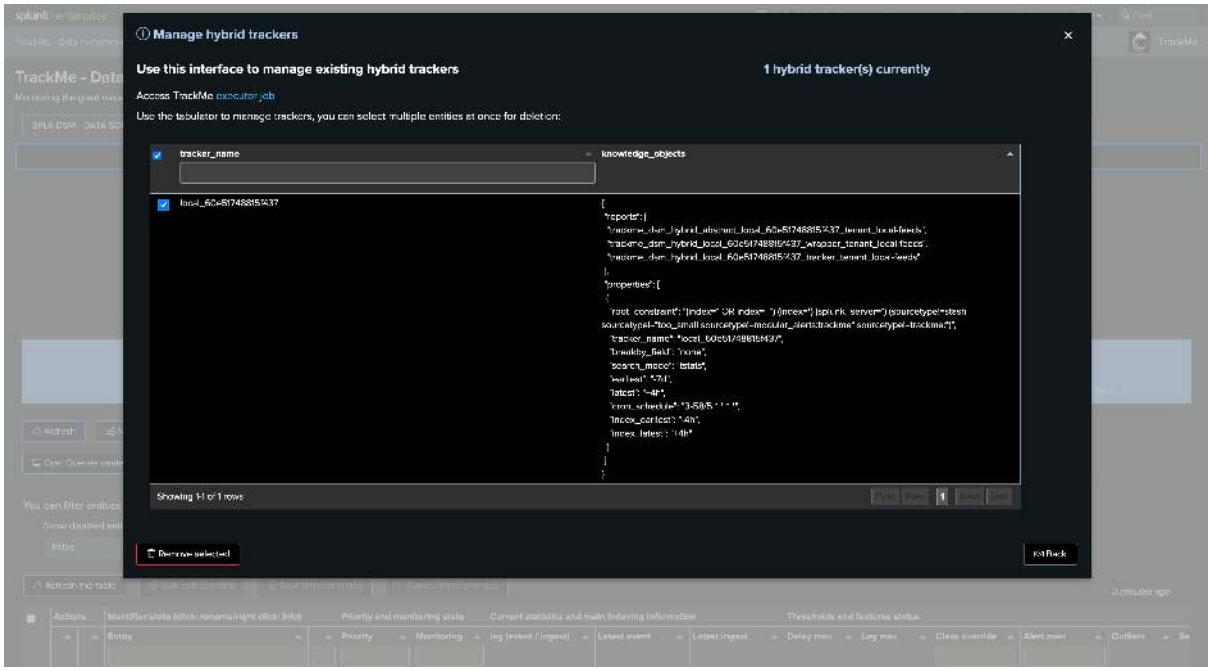
To manage Hybrid Trackers, click on:



The user interface shows available trackers and their related objects:



Select one or more trackers to be deleted:



The related knowledge objects will be deleted, and the Virtual Tenant record will be cleaned up automatically.

For splk-feeds, the entities that were created through these Hybrid Trackers will **not** be deleted. (However, unless another Tracker is created, these will not be maintained anymore)

Deleting an Hybrid Tracker through REST

You can delete a Tracker through the following REST endpoint, example in SPL:

```
| trackme mode=post url="/
↪services/trackme/v2/splk_hybrid_trackers/admin/hybrid_tracker_delete" body="{
↪'tenant_id': 'mytenant', 'component': 'dsm', 'hybrid_trackers_list':
↪'test:001,test:002'}"
```

8.3.29 Replica trackers

A **replica tracker** mirrors an entity set from a **remote TrackMe instance** into the local tenant — the pattern for a single central search head reporting across many environments that each run their own TrackMe.

Note

This is one of the two advanced tracker types. To build a tracker from your own SPL (local or remote) rather than mirror another TrackMe, see *Hybrid trackers*.

What is replicated

Per-entity **runtime** data only — state, score, priority, SLA, tags, timestamps — **not** configuration. Thresholds and policies stay on the source, and replication is **one-way**.

How it behaves

Replicated entities carry the same state, score, and anomaly reason as on the source; the local decision maker does **not** re-evaluate them, and the UI marks them as replicated. A global executor refreshes them every few minutes, so they are as fresh as the last run.

Setup

A wizard drives it — pick the source remote account, the source tenant and component, and a target prefix, preview the entity count, and create.

Note

- Replicated entities count against the **target** tenant's license entity count.
- Avoid cascading replication (A → B → C) — prefer a flat spokes-to-hub model — and keep source and target TrackMe versions aligned.

See also

- *Hybrid trackers* — the other advanced tracker type: build a signal from your own SPL.
- *Integrations & Remote Deployments* — the remote accounts replica trackers rely on.
- *Monitoring Components* — replicated entities land in the same components as native ones.

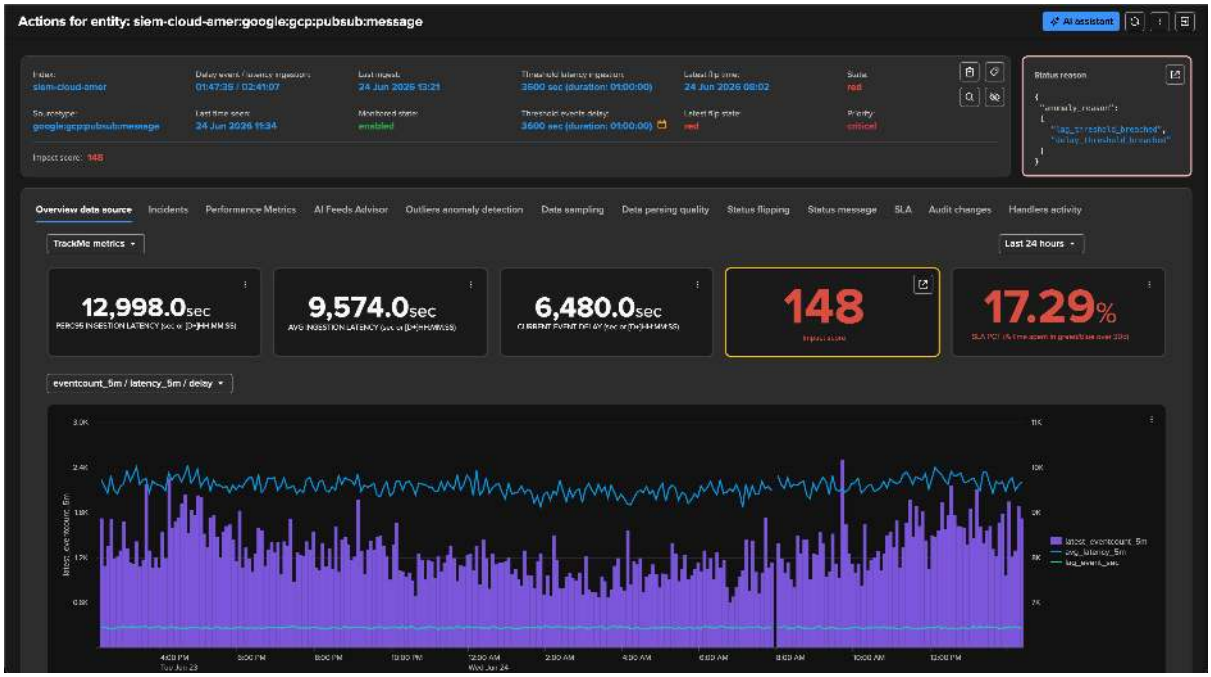
8.4 Entity State & Scoring

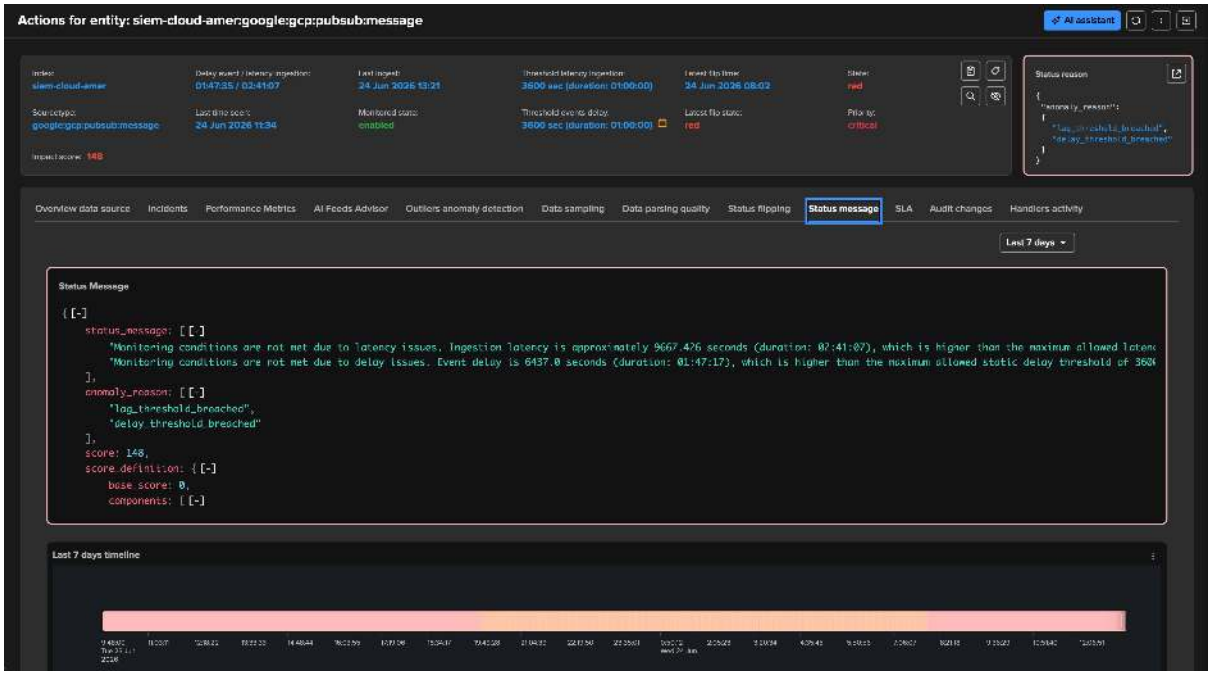
This part explains how TrackMe decides whether an entity is healthy. It is the engine room of the product: the four-colour state machine, the impact score that drives it, the delay and latency KPIs and dynamic thresholds that feed that score, and how to read an entity that keeps flipping between states.

- **The entity state machine** — green, orange, red, blue, the status message, and the enabled/disabled monitored state.
- **Impact scoring & thresholds** — how the score is built, tuned, and what delay, latency, and dynamic thresholds contribute.
- **Status flapping** — detecting and taming entities that flip between states.

8.4.1 The entity state machine

Every entity carries a single **state** — a colour that tells you, at a glance, whether it is healthy. The state is recomputed from scratch on every monitoring cycle, so it always reflects the latest data. This page explains the four states, what moves an entity between them, and the two fields that explain *why* an entity is where it is.





The four states

State	Score	Meaning
green	`0`	Healthy. No active anomaly, no override.
orange	`0 < score < 100`	Warning — a non-critical anomaly is active. *Also* used as a muted state when an entity is outside its expected monitoring window.
red	`score ≥ 100`	Critical — the entity has crossed the critical threshold.
blue	(any)	**Suppressed.** Not an alarm. An anomaly may be present, but the entity is deliberately muted. Blue always wins over red and orange.

The numeric thresholds (`0` / `< 100` / `≥ 100`) come from the impact score, covered in *Impact scoring & thresholds*.

What drives a transition

Each cycle is **idempotent** — the same inputs always produce the same state — and runs in three tiers:

1. **Raw state from anomalies and thresholds.** No anomalies and all thresholds met → green. Otherwise the impact score decides orange or red.
2. **Logical-group protection.** If an entity would be red but belongs to a logical group whose green percentage is still above the group's minimum, it is demoted to **blue** — the healthy group carries its struggling member (see *Classification & Protection*).
3. **Monitoring window and disruption grace.** Outside its expected monitoring window the entity is muted to **orange**; inside a disruption grace period it is held **blue** so a brand-new anomaly does not alert before it has persisted (see *Classification & Protection*).

Why an entity is blue

Blue is never a failure — it means TrackMe is intentionally holding back. Four mechanisms cause it:

- **Logical-group protection** — the group is mostly healthy.
- **Disruption grace** — the anomaly is too new; TrackMe waits to see if it persists.
- **Maintenance** — the entity, tenant, or instance is in a maintenance window (see *Operations & Administration*).
- **Acknowledgment** — someone already accepted the problem (see *Alerting*).

Note

Orange has two meanings. It is both a genuine warning (a non-critical score) and a muted "out of expected window" state. The status message text always tells you which.

Monitored vs unmonitored

Independently of its colour, every entity has a **monitored state**:

- **enabled** — actively monitored and eligible to alert.
- **disabled** — retained but not monitored, and it never alerts.

The main views show enabled entities by default; disabled ones are hidden but still counted — use the table header filter on `monitored_state` to reveal them. Toggle it per entity, in bulk, or via the API — e.g. `| trackme url="/services/trackme/v2/splk_dsm/write/ds_monitoring" mode="post" body="{'tenant_id': 'mytenant', 'action': 'disable', 'object_list': '<entity1>, <entity2>'}"` (`monitored_state` is the persistent KV Store field this sets). It is the right tool when a hybrid tracker discovers entities you do not care about, or when you want to park a noisy entity without deleting it.

Note

Keep three terms distinct: the **state** (the colour), the **monitored state** (enabled / disabled), and the **score** (the number behind the colour).

Status message and anomaly reason

Two fields explain an entity's state:

- **Status message** — a human-readable, colour-matched sentence explaining *why* the entity has its state. It is the fastest way for an analyst to understand what is wrong, and it appears in the entity's *Status message* tab, inside notable events, and in alert results.
- **Anomaly reason** — a short machine code for why the entity is not green, such as `delay_threshold_breached`, `variable_delay_threshold_breached`, `latency_threshold_breach`, `outliers_breach`, `flx_dynamic_threshold_breach:<metric>`, `wlk_skipped_pct_threshold_breach`, or `inactivity_threshold_breach`. Stateful alerting watches this code and re-opens an alert when the reason changes.

Both are arrays — an entity can be unhealthy for more than one reason at once. Which reasons are possible depends on the component (delay / latency for the feeds family; dynamic thresholds for FLX / FQM / WLK; orphan, version drift, and app-disabled for WLK).

Status message examples

A few real `status_message` / `anomaly_reason` payloads, as TrackMe stores and surfaces them.

Green — every condition met (here a feed with healthy delay and latency, carried in a logical group):

```
{
  "status_message": [
    "monitoring conditions for event delay are met. Event delay is 49.565 seconds␣
↪(duration: 00:00:49), which is lower than the maximum allowed delay of 3600␣
↪seconds (duration: 01:00:00), latest event available (_time): 15 Aug 2025 10:55",
    "monitoring conditions for ingest latency are met. Ingestion latency is␣
↪approximately 21.828 seconds (duration: 00:00:21), which is lower than the␣
↪maximum allowed latency of 3600 seconds (duration: 01:00:00), latest event␣
↪indexed (_indextime): 15 Aug 2025 10:55",
    "Logical Group cribl ... is in normal state. The current green percentage of␣
↪the group is 33.33% which is higher or equal to the minimal green percentage of␣
↪5.0%, object_group_members_count=3"
  ],
  "anomaly_reason": ["none"]
}
```

Red — latency breach (`lag_threshold_breached`):

```
{
  "status_message": [
    "Monitoring conditions are not met due to latency issues. Ingestion latency is␣
↪approximately 353.0 seconds (duration: 00:05:53), which is higher than the␣
↪maximum allowed latency of 300 seconds (duration: 00:05:00) ... this indicates␣
↪that the source is receiving delayed events only"
  ],
  "anomaly_reason": ["lag_threshold_breached"]
}
```

Red — delay breach, static policy (`delay_threshold_breached`):

```
{
  "status_message": [
    "Monitoring conditions are not met due to delay issues. Event delay is␣
↪17880.563 seconds (duration: 04:58:00), which is higher than the maximum allowed␣
↪delay of 3600 seconds (duration: 01:00:00) ... the source is receiving events␣
↪with timestamps older than the threshold defined for this entity."
  ],
  "anomaly_reason": ["delay_threshold_breached"]
}
```

Red — delay breach, variable policy (`variable_delay_threshold_breached`) — the message names the **active slot** and its threshold:

```
{
  "status_message": [
    "Monitoring conditions are not met due to variable delay threshold breach␣
↪(active slot: 'business_hours'). Event delay is 7200.0 seconds (duration:␣
↪02:00:00), which is higher than the maximum allowed delay of 3600 seconds␣
↪(duration: 01:00:00) for the active slot ..."
  ],
  "anomaly_reason": ["variable_delay_threshold_breached"]
}
```

See also

- *Impact scoring & thresholds* — how the score produces the state.
- *Status flapping* — when an entity flips between states.
- *Classification & Protection* — logical groups, disruption grace, priority.

8.4.2 Impact scoring & thresholds

Behind every state colour is a number: the **impact score**. It is the most powerful — and the most misunderstood — dial in TrackMe. This page builds the model up step by step: how the score is composed, the crucial difference between problems that score *immediately* and problems that *accumulate* over time, and how you can clear or push an entity's score by hand.

Tip

This page is the conceptual guide. For the exhaustive reference — every weight, every field, the SPL behind it — see the *Impact Scoring white paper*.

The score in one line

Each cycle, TrackMe sums everything contributing to an entity into a single `total_score`:

```
total_score = base_score            # rolling 24h history (the running total of␣
↪events)
            + impact_score_delay     # while the delay threshold is breached
            + impact_score_latency   # while the latency threshold is breached
            + impact_score_outliers  # the accumulated ML-outlier sub-score
            + threshold_scores       # breached dynamic thresholds (FLX/FQM/WLK)
            + impact_score_<…>       # inactivity, future data, host-count drift, …
```

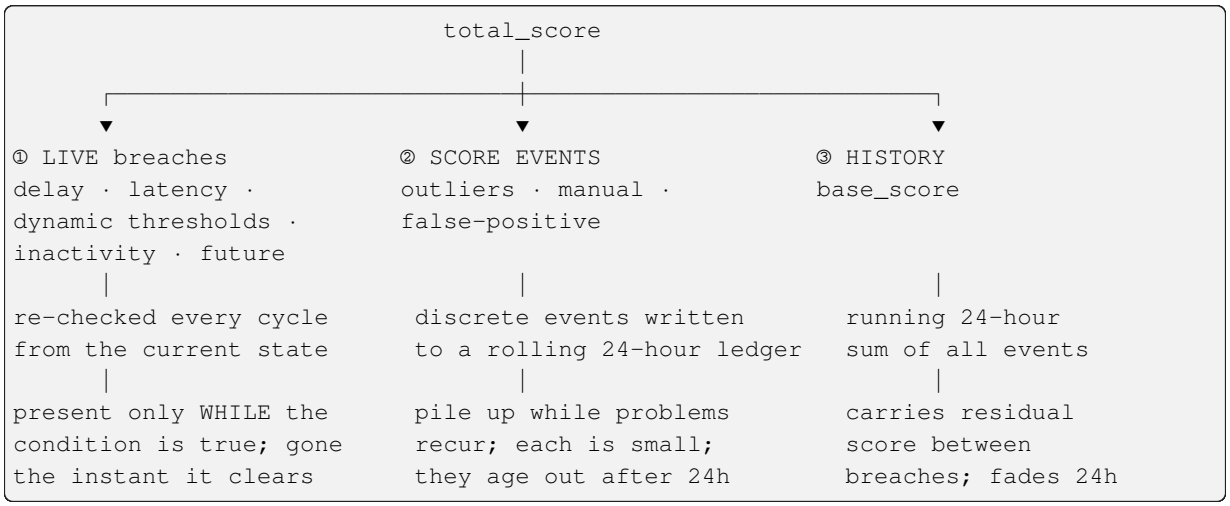
…and maps that number to a colour:

- `total_score == 0` → **green**
- `0 < total_score < 100` → **orange**
- `total_score ≥ 100` → **red**

By convention **100 is the critical line**. A weight of 100 means a single breach is enough to turn an entity red; smaller weights mean several problems must add up first. (A separate guard can still override the colour to **blue** when an entity is suppressed — see *The entity state machine*.)

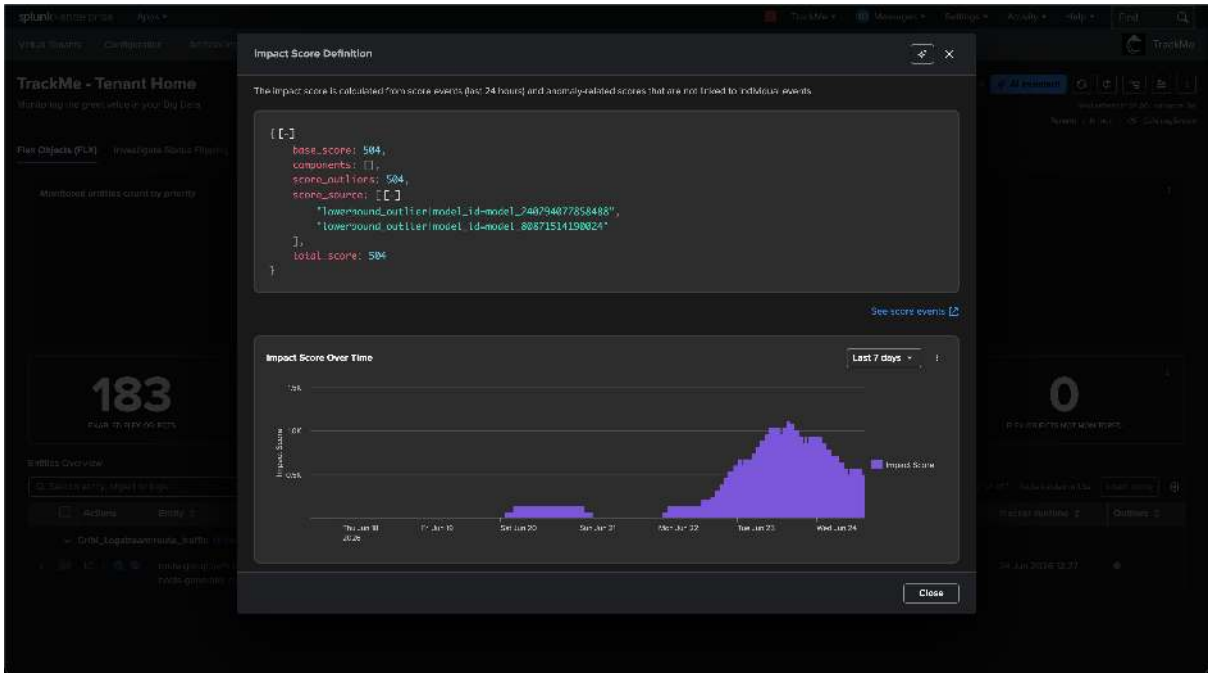
The three kinds of contribution

This is the part worth slowing down on, because almost every "why is the score doing that?" question comes back to it. Everything that adds to the score is **one of three kinds**, and they behave very differently.



1. **Live breaches** answer *"is it broken right now?"*. A delay, latency, dynamic-threshold, inactivity, or future-data breach is re-evaluated every cycle. While the condition holds, its weight is added; the moment it clears, the contribution is simply not added on the next cycle — it disappears instantly, with no lag.
2. **Score events** answer *"how many anomalies lately?"*. Some signals — above all **ML outliers**, plus your own manual and false-positive adjustments — are written as individual **events** into a rolling **24-hour** ledger. Each event adds a little, they accumulate while problems recur, and they fall off the back of the window after 24 hours.
3. **History** (`base_score`) is the running sum of all those events over the last 24 hours. It is why a long-troubled entity carries weight even between individual breaches, and why it takes a little while to settle fully back to green.

The entity's **Impact Score Definition** view shows this directly — the `base_score`, the `score_outliers` sub-score, the contributing event sources, and the score over time:



Why outliers don't behave like a delay breach

The classic surprise. A single **delay breach** (weight 100) turns an entity red at once, and green again the moment data flows. A single **ML outlier** is deliberately worth only about **36** — *below* the red line — so one anomaly leaves the entity **orange, not red**. Repeated outliers within the 24-hour window add up:

```
1 outlier   →  ~36    orange   (a nudge, not an alarm)
2 outliers  →  ~72    orange
3 outliers  → ~108    red      (now it's a sustained pattern)
… time passes, events older than 24h drop off → the score eases back down
```

This is TrackMe's core **false-positive defence**: a one-off blip cannot page you; only a recurring pattern escalates. (An undertrained model contributes nothing at all — see *Machine Learning*.)

Important

It is normal for a fixed problem to stay orange for a while. The *live* breach clears instantly, but the *accumulated* outlier events and the `base_score` history linger until they age out of the 24-hour window. The entity is remembering, not malfunctioning.

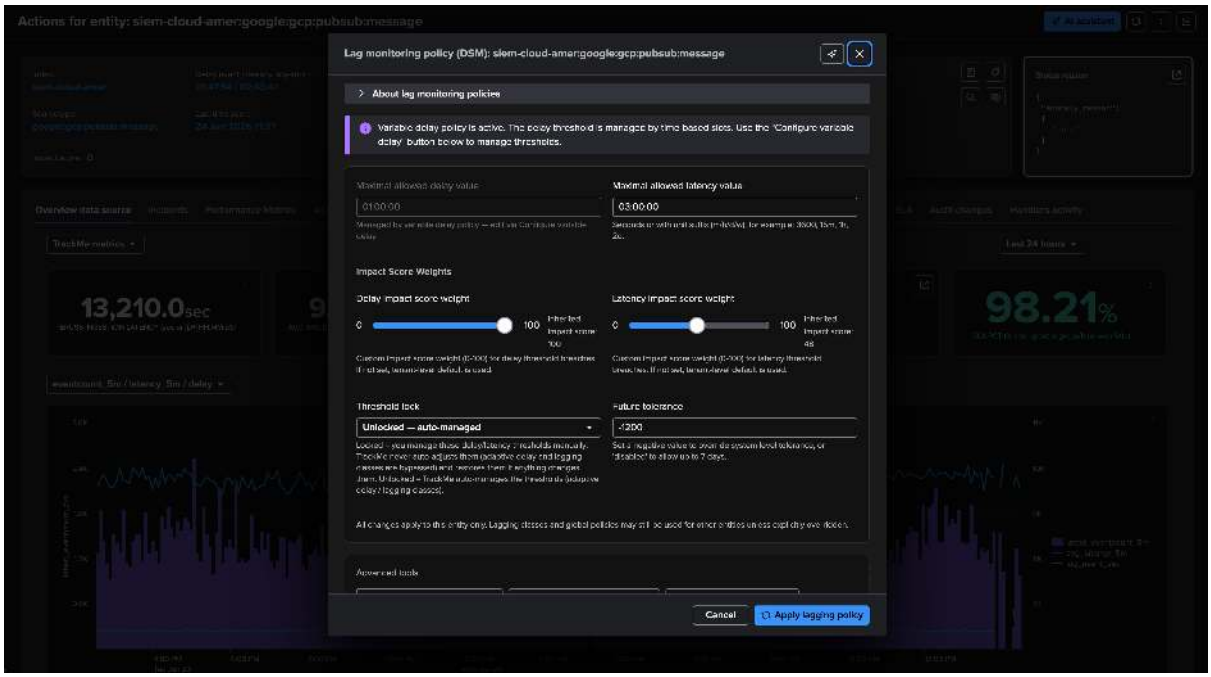
Delay and latency

For the feeds family (DSM, DHM, MHM), two live KPIs do most of the work:

KPI	Definition	A breach usually means
Delay	`now() - last_event_time` — how late the newest data is.	A broken or stalled data flow (process, config, or network failure).
Latency	`_indextime - _time` — the lag between an event happening and being indexed.	Network pressure, time-zone misconfiguration, or bad timestamp parsing.

The effective **delay threshold** for an entity is resolved in order — the first that applies wins: a **per-entity override** (optionally **locked** so nothing rewrites it); a **variable-delay** rule that changes by time slot; a **lagging class** (a named, reusable threshold profile); then the **tenant default**. **Adaptive delay** can compute these from history. The full delay configuration lives with the *DSM component*.

The per-entity lag-monitoring policy is where you see the delay and latency weights, the threshold lock, and the future-tolerance setting together:

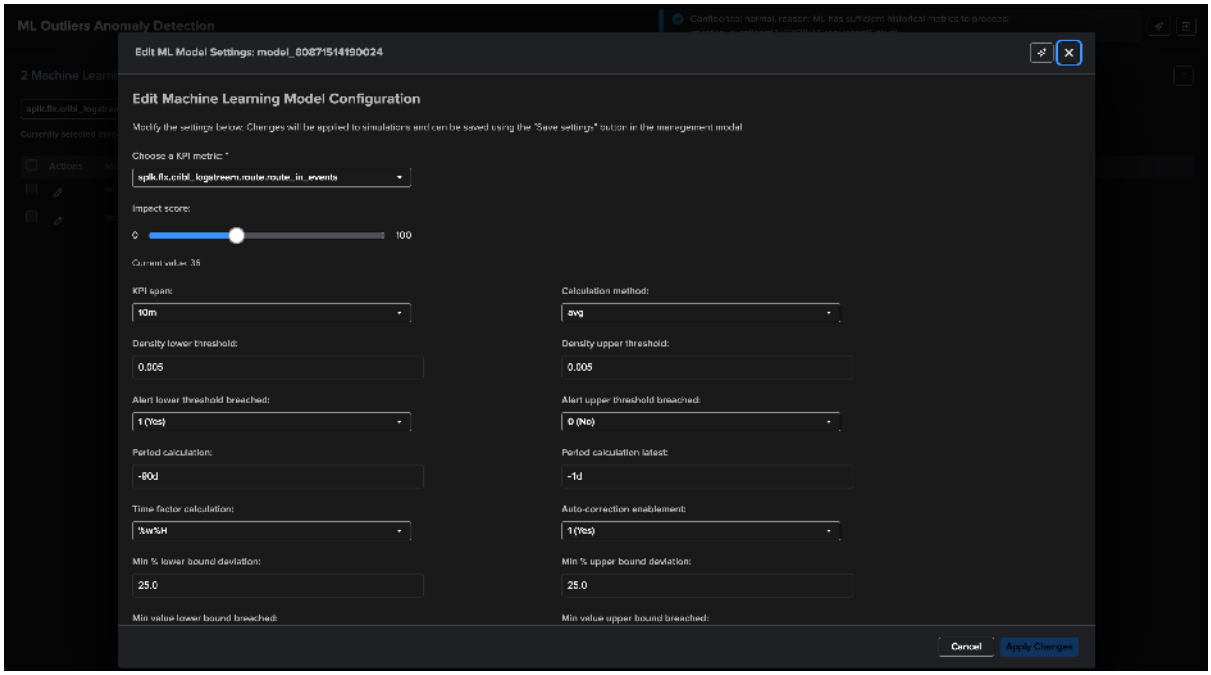


Dynamic thresholds (FLX, FQM, WLK)

FLX, FQM, and WLK do not use delay/latency; they evaluate **dynamic thresholds** on the metrics they emit. A rule is a metric name, an operator (`lt`/`le`/`gt`/`ge`/`eq`/ `ne`), a target value, and its own **score**. A breach adds that score on top of the component's threshold-breach weight — these are **live** contributions, present only while the rule is breached. FLX thresholds can also be time-sliced; WLK supports a tenant-wide default threshold.

Outliers as score events

Each ML model carries its own **impact score** for a detected outlier (the default is about 36). When the model flags a value, it writes one score event — that is what accumulates into the `score_outliers` sub-score described above. The model's impact score, KPI, seasonality and thresholds are all set in its configuration:

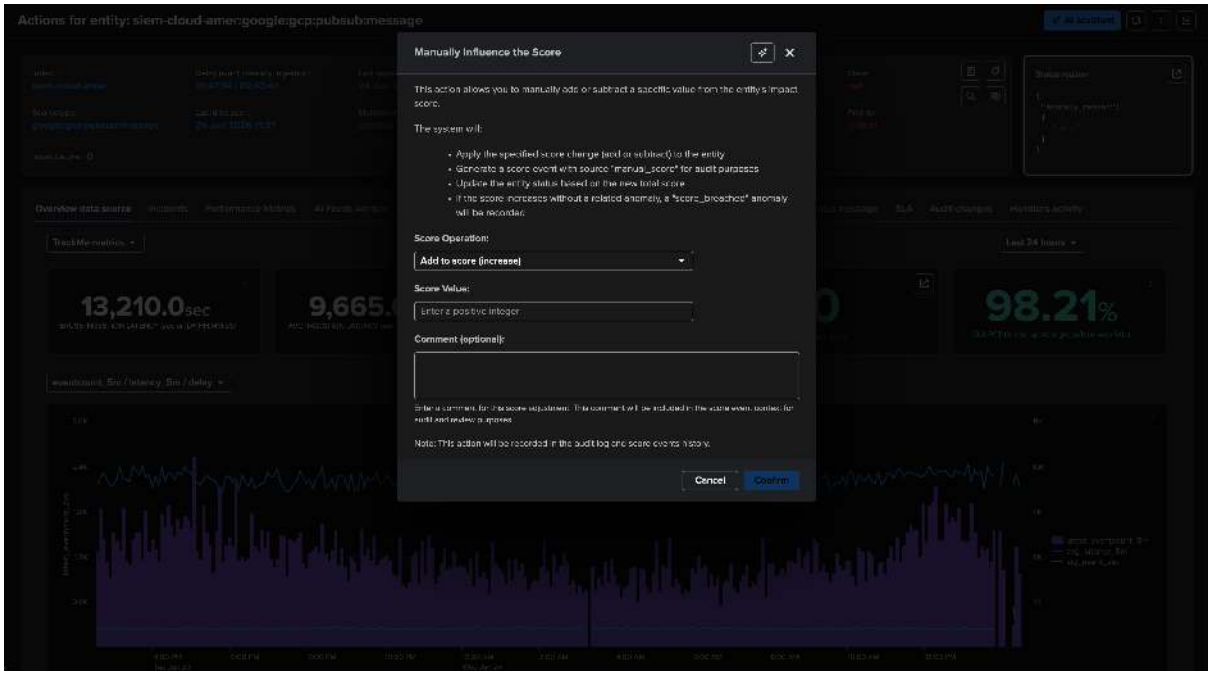


Outlier detection, training, confidence, and seasonality are covered in *Machine Learning*.

Clearing or pushing the score by hand

Because the score is partly a **ledger of events**, you can write your own events to it — to **clear a false alarm immediately**, or to **force attention**. There are three actions, and the right mental model is *"I am adding a negative (or positive) score event"*, never *"I am changing a weight"*.

Action	What it does
Set as false positive (entity)	Writes a negative event equal to the entity's current score, so the **total drops to zero** on the next cycle and the entity goes green. Clicking again just *replaces* the previous one — you cannot over-correct. It does **not** touch the outlier sub-score.
Set as false positive (Outliers tab)	The same idea, but clears **only** the accumulated outlier score, leaving the rest of the total intact.
Manually influence the score (add / subtract)	Nudge the score up or down by a value you choose. Manual adjustments **accumulate**. Pushing the score *up* without a real anomaly records a `score_breached` anomaly so the change is visible and audited.



Important

These adjustments take effect on the **next cycle** (an immediate cache means you do not wait for indexing) and they **fade after 24 hours**. They *suppress for the window* — they do not permanently change the entity. To stop outlier alerts for good, **fix or retrain the model**, do not keep marking false positives.

Warning

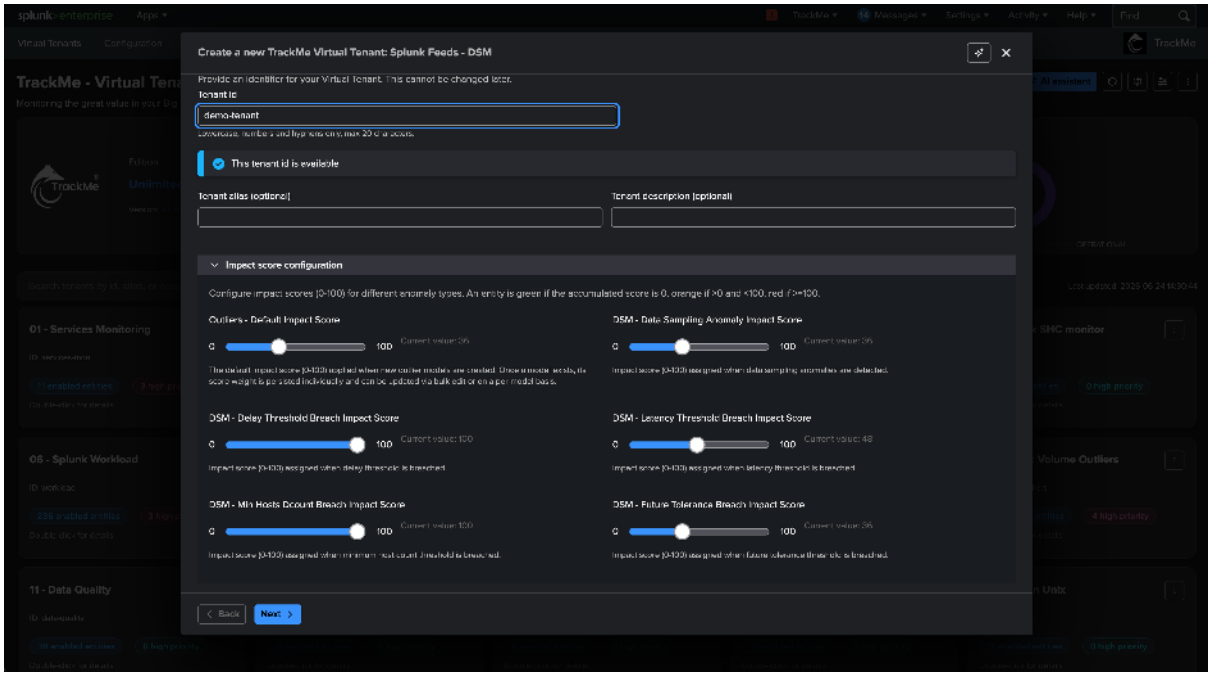
Never try to cancel a condition with a **negative weight**. To reduce score, lower the relevant weight, or use the false-positive / manual-decrease actions above.

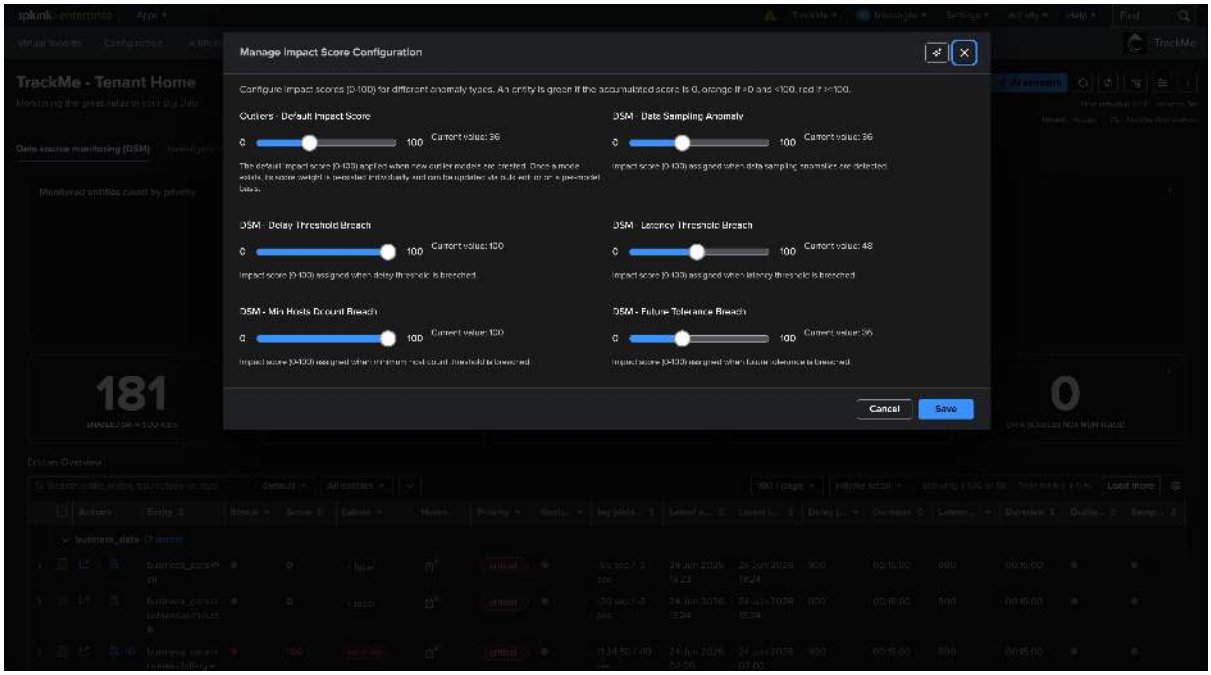
Tuning the weights

Every contribution above has a **weight** you can change. Weights cascade — **later wins**:

1. a built-in **fallback**;
2. the **system default** (affects *new* tenants only);
3. **per tenant** (affects every entity in that tenant);
4. **per entity** (overrides the tenant value for one entity).

Setting a weight to **0 disables** that contribution for the chosen scope. The weights are set at tenant creation and from the per-tenant **Impact Score Configuration**:





Note

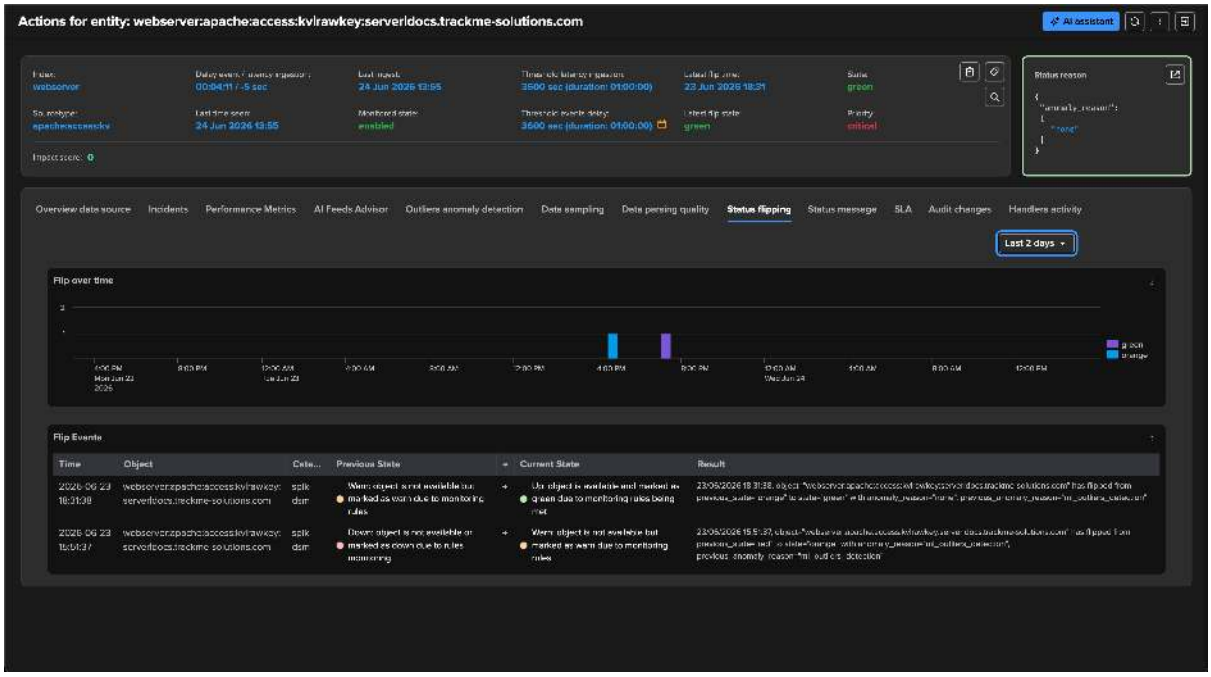
The defaults are deliberately not all 100. Typically a delay breach and a host-count breach weigh **100** (red on their own), while latency weighs around **48** and outliers, data-sampling, and future-tolerance weigh around **36** — so a latency breach *alone*, or a *single* outlier, lands at **orange**, not red. That is the false-positive tuning at work, not a bug. The exact values live in your tenant settings; tune from there.

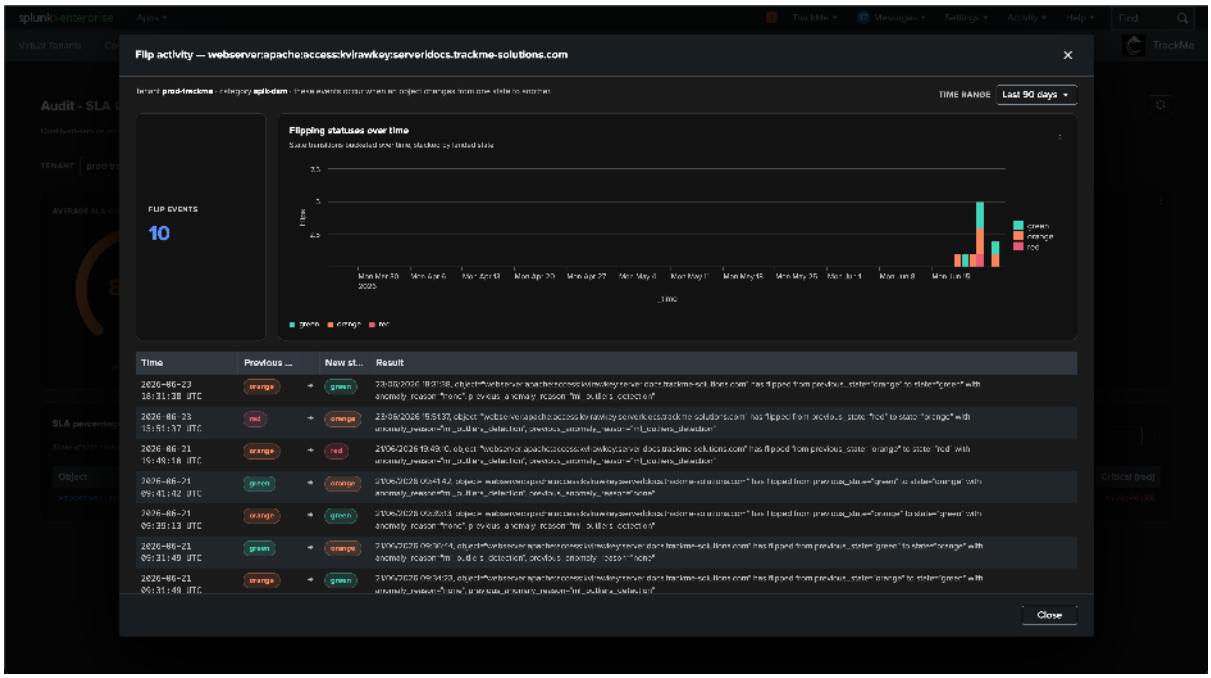
See also

- *Impact Scoring white paper* — the full reference.
- *The entity state machine* — how the score becomes a colour.
- *Machine Learning* — outlier detection, training, and false-positive handling.
- *DSM — Data Source Monitoring* — delay and latency configuration in depth.

8.4.3 Status flapping

An entity that rapidly flips between healthy and unhealthy — *flapping* — is telling you something different from one that is steadily broken. TrackMe records every state transition so you can spot, and tame, unstable entities.





What flapping is

Every time an entity changes state, TrackMe writes a **flipping event** recording the previous state, the new state, the anomaly reason, and the time. The first event for any entity is its discovery — a flip from the synthetic `discovered` status to its first computed state. For example:

```
object=… flipped from previous_state=discovered to state=green  anomaly_reason=none
object=… flipped from previous_state=green      to state=red    ␣
↪anomaly_reason=delay_threshold_breached
object=… flipped from previous_state=green      to state=red    ␣
↪anomaly_reason=variable_delay_threshold_breached
object=… flipped from previous_state=red        to state=green  anomaly_reason=none
```

Flapping is tracked for the `splk-dsm`, `splk-dhm`, `splk-mhm`, `splk-cim` and `splk-flx` components.

Why it matters

The **frequency** of flips is itself a signal. A steadily-red entity is a known problem; an entity bouncing red ↔ green every few minutes usually means a **threshold set too tight for a bursty feed**, not a real outage. The repeated `anomaly_reason` on each flip tells you *which* condition is responsible — a red → green → red pattern with the same reason is the classic "threshold too tight" signature.

Where to review it

- Open the entity's **Status Flipping** tab — an over-time chart of flip frequency plus a table ordered by the most recent transitions.
- Or search directly in Splunk, for example:

```
`trackme_idx(mytenant)` sourcetype=trackme:state tenant_id="mytenant"␣
↪object="myobject"
```

Flipping data also feeds TrackMe's **SmartStatus** assessment of an entity.

Taming flapping

Flapping is a *symptom*, not a separate alarm — the fix is to absorb the transient blips:

- **Widen the disruption grace window** so a brand-new anomaly is held **blue** until it has persisted, instead of immediately flipping to red (see *Classification & Protection*).
- **Tune the threshold** that keeps tripping — for a bursty feed, a variable-delay rule or a more appropriate lagging class is usually the answer (see *Impact scoring & thresholds*).
- Remember that **logical-group protection** already dampens flapping: a member of a healthy group shows as a stable blue rather than flipping.

See also

- *The entity state machine* — the states and the blue suppression that reduces flapping.
- *Impact scoring & thresholds* — thresholds and grace settings to tune.
- *Classification & Protection* — the disruption queue, your main lever against flapping.

8.5 Classification & Protection

The decision maker tells you *what state* an entity is in. The features in this part tell you *how to treat* that state: how important the entity is, what service target it carries, how it is grouped, and which entities are protected from noisy alerting.

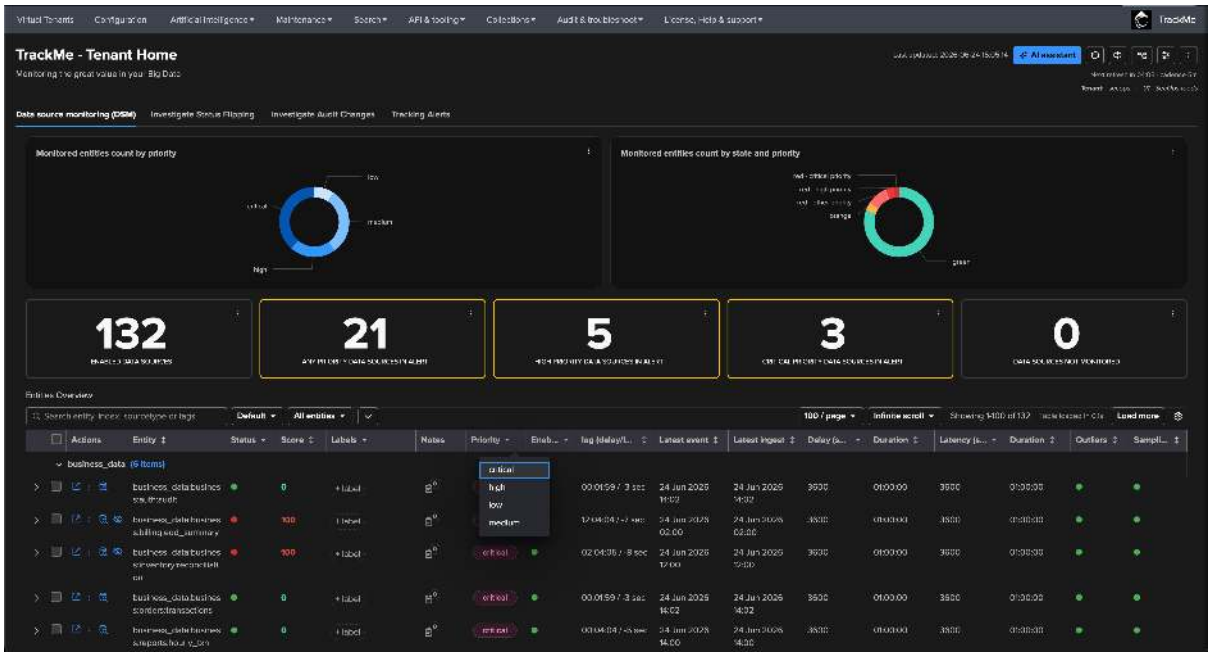
- **Priority** — rank entities by importance.
- **Tags** — free-form metadata for filtering and routing.
- **Labels** — a curated, colour-coded catalogue (and rule-driven auto-labels).
- **SLA** — service-level classes, targets, and attainment.
- **Logical groups** — collective health and group protection.
- **Disruption queue** — grace periods that absorb transient anomalies.
- **Notes** — free-form context attached to an entity.

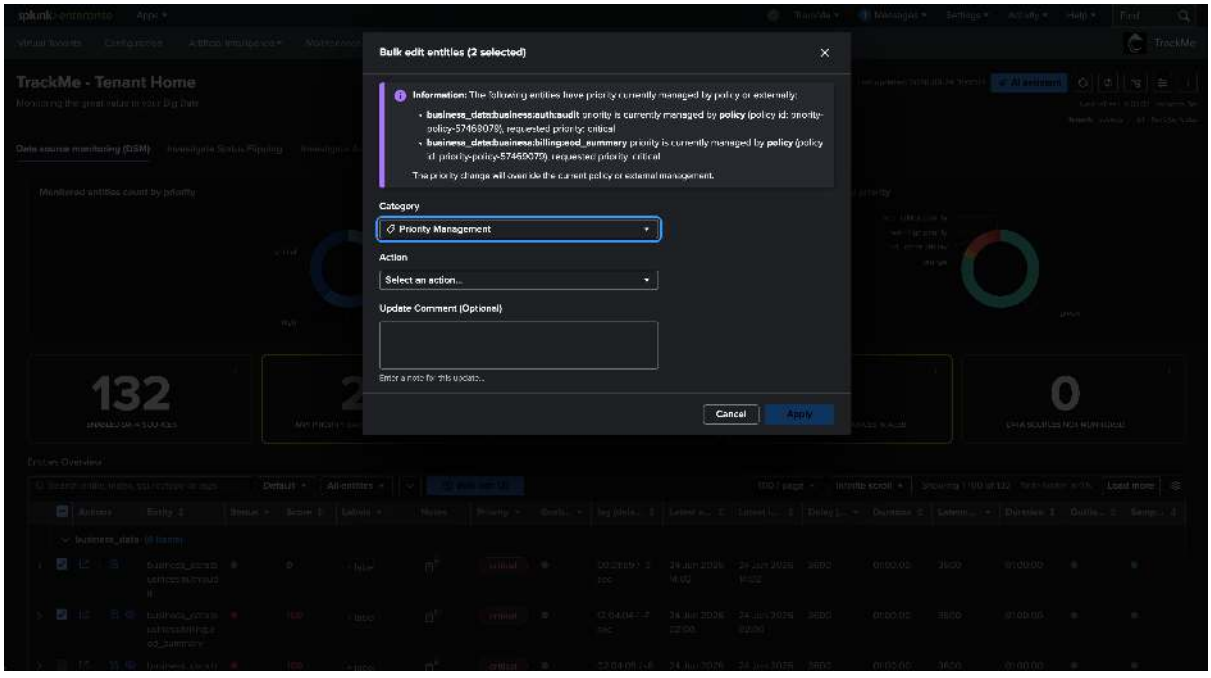
Note

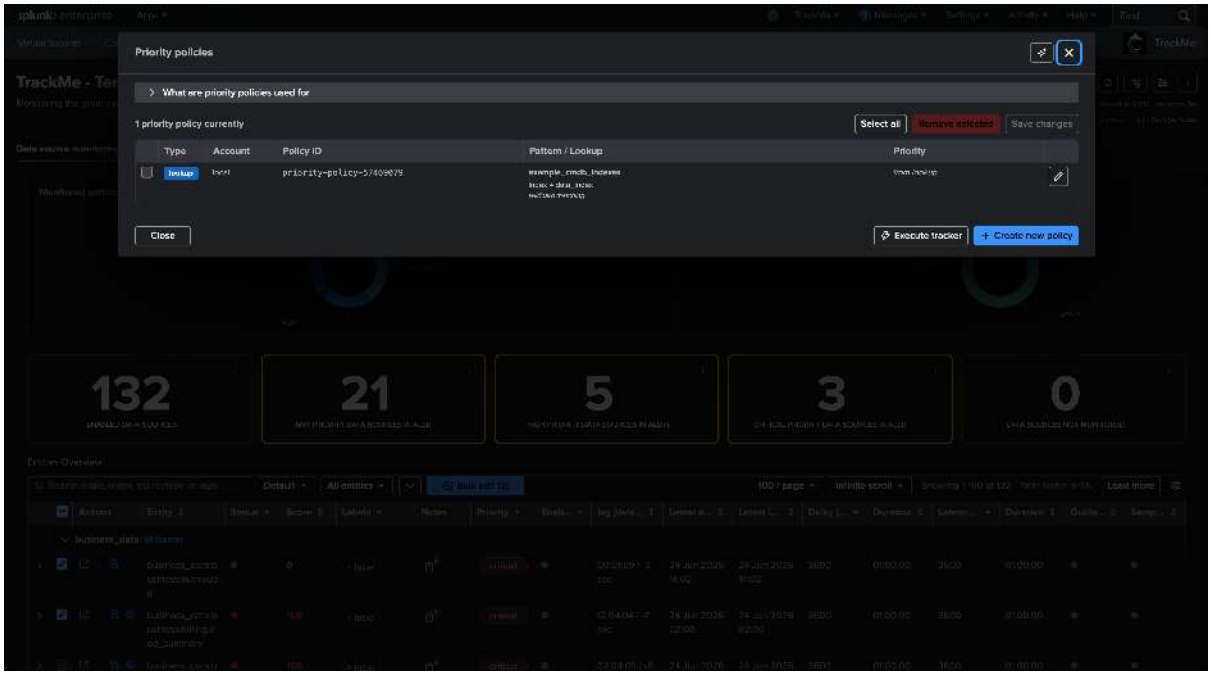
Priority, tags, and SLA share one mechanism — policies. A policy assigns metadata automatically using a rule (regex, a Splunk lookup, or an SPL search), per component. Policies *suggest*; a manual override always wins; and removing a policy auto-cleans the assignments it made.

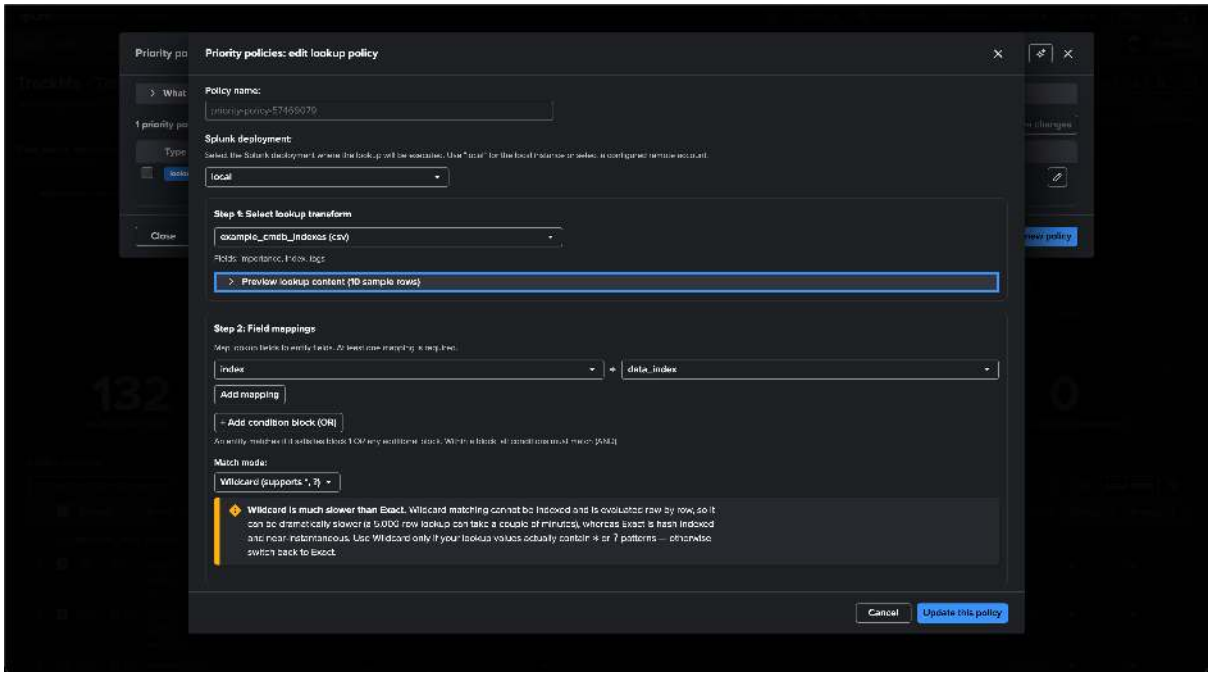
8.5.1 Priority

Priority ranks entities by business importance. It is one of the most widely *leveraged* classifications in TrackMe — alerting, virtual groups, automation, and the UI all use it to decide which entities to act on or surface first. You can set it by hand or assign it automatically with priority policies.









The priority levels

Level	Meaning
critical	The highest business importance.
high	
medium	
low	The lowest business importance.
pending	A “not yet triaged” / discovery bucket — **not** the highest priority despite sorting last.

Newly discovered entities take the tenant’s **default priority** (`medium` unless you change it).

Where priority is used

Priority is **classification, not scoring** — it never changes an entity's score or state (see *Impact scoring & thresholds*). Instead, it is the lever many features use to decide *which* entities to notify on, surface, or act on:

Alerting & notification routing
: Stateful alerting can **route by priority**. The alert action's `priority_levels_emails` and `priority_levels_commands` settings choose which entities actually send emails and run commands — so you can, for example, email only **critical** and **high** entities while everything is still tracked and recorded. An entity off the list still alerts internally; only its email/command notifications are suppressed. (The ingest channel is not priority-gated.) The entity's current priority is read at alert time and shown in the email and on the incident record. See *Stateful alerting*.

Notable events
: Every TrackMe *notable event* carries the entity's priority as a field, so your Enterprise Security or correlation logic can prioritise and route on it.

Virtual Groups
: A *virtual group* can be **scoped to chosen priority levels** (for example only high and critical), and its card shows a priority breakdown across the tenants it aggregates.

Finding and focusing in the UI
: Priority is a **filter and search axis** on every entity table. Tenant cards show a **red-by-priority** breakdown, and a **High-priority overview** highlights entities that are red at high or critical priority — the fastest "what should I look at first?" view.

Automation scoping
: Several features can be **limited to chosen priority levels**: *auto-label rules*, *ML outlier detection* (train and score only where it matters), and **AI Advisor automated actions** (so autonomous tuning only touches the entities you trust it with). AI investigations and chat also include the entity's priority in the context they send to the model.

Setting priority

- **Manually**, per entity — a manual priority is flagged as such and is **never** overwritten by a policy. Clear the manual flag to return the entity to policy control.
- **By policy** — a priority policy assigns a level automatically, using one of three resolution methods (see below).

When several priority policies match one entity, the **highest level wins**.

Priority policies

Priority is one of the three policy-driven classifications (with tags and SLA). A policy matches entities using one of three methods, which you can mix in a single policy set:

Method	How it matches
regex	A regular expression against a field such as `object_id`, `index`, `sourcetype`, or `host`. Matching uses `re.search` — an unanchored pattern matches anywhere.
lookup	A Splunk lookup (CSV or KV Store); the priority value can be pulled from a lookup column.
SPL search	The results of a saved search.

Policies are **per component** (DSM / DHM / MHM / FLX / FQM / WLK). Removing a policy auto-cleans the assignments it made on the next cycle. The add/edit modal can **preview matches** so you can test a regex or search before saving.

Every priority change is **audited** — review an entity's history in the **Audit** tab of its main screen. You can also set priority in bulk from SPL, e.g. `| trackme url="/services/trackme/v2/splk_dsm/write/ds_update_priority" mode="post" body="{'tenant_id': 'mytenant', 'priority': 'high', 'object_list': '<entity1>,<entity2>'}"`. In high-scale environments, roll out priority progressively and pair it with tiered alerting to reduce alert fatigue.

Note

`pending` is a discovery state, not an escalation — do not assume it ranks above `critical`. And because conflicts resolve by *level*, not by policy order, adjust the levels (or narrow the policies) if the "wrong" policy wins. Prefer lookups over SPL for static data — SPL policies add a little search load each cycle.

See also

- *Priority — in depth* — the regex/lookup policy procedures, the numeric hierarchy, the policy tracker, and external management SPL.
- *Tags* and *SLA policies* — the sibling policy types.
- *Virtual Groups* — virtual groups can filter on priority.

8.5.2 Priority — in depth

Tip

This is the **in-depth reference** for priority management. For the readable overview, start with *Priority*. This page covers the regex and lookup-based policy procedures, the numeric priority hierarchy, the policy tracker, and external priority management with SPL.

About priority levels in TrackMe

Hint

Priority levels are an important concept in TrackMe which can easily be leveraged to categorize the importance of your different Splunk feeds, and for instance redirect notifications depending on an entity's priority level.

- **TrackMe supports the following priority levels:**
  - **critical**
  - **high**
  - **medium**
  - **low**
  - **pending**
- The priority level of an entity is defined by default when **discovered**, and can be updated **per entity**, defined by **policy**, or managed **externally**.
- Entities are usually filtered when forwarding alerts to a third party, such as email or a ticketing system, based on their priority level.
- Priority is a valuable concept that can be leveraged to reduce noise and alert fatigue. A frequent and recommended practice is to focus on high and critical priorities, while other priorities can be considered over time.

About the pending priority level

- The **pending** priority is a special priority level added which can be used to qualify newly discovered entities before allowing them to be parts of the alerting framework.
- Especially, when creating alerts, TrackMe calls the following macro:

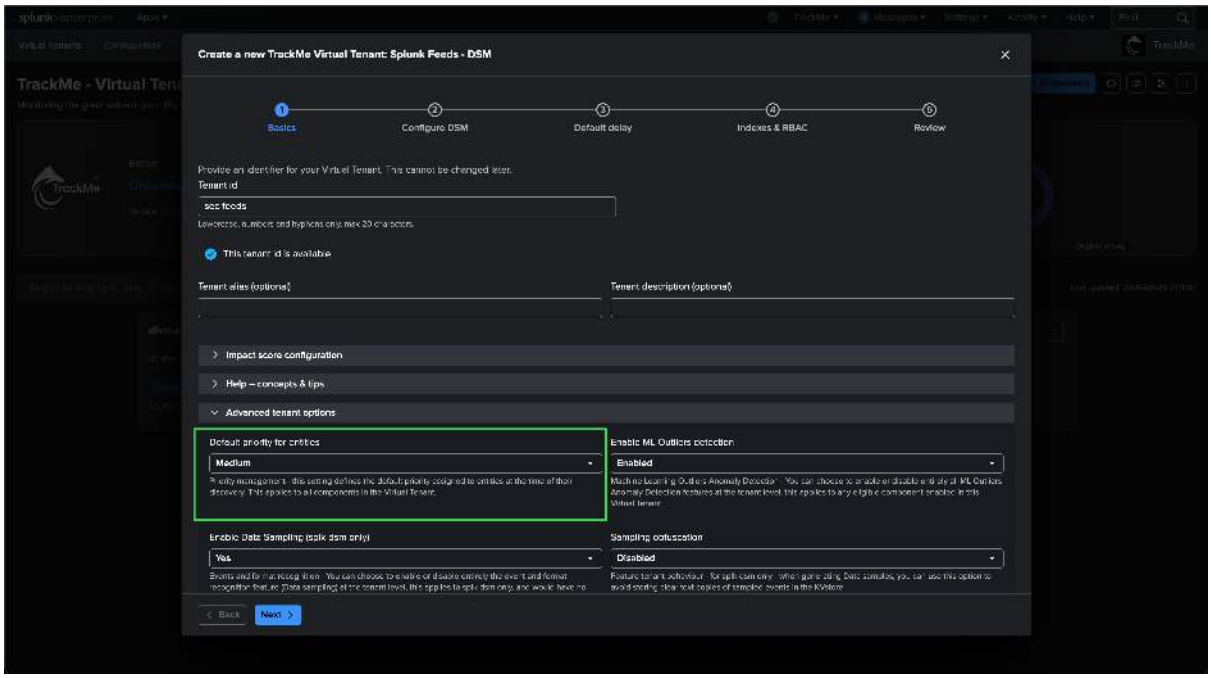
`trackme_apply_maintenance_mode_v2` which contains the following exclusion:

```
``` exclude entities with priority pending ```
| where NOT (priority="pending")
```

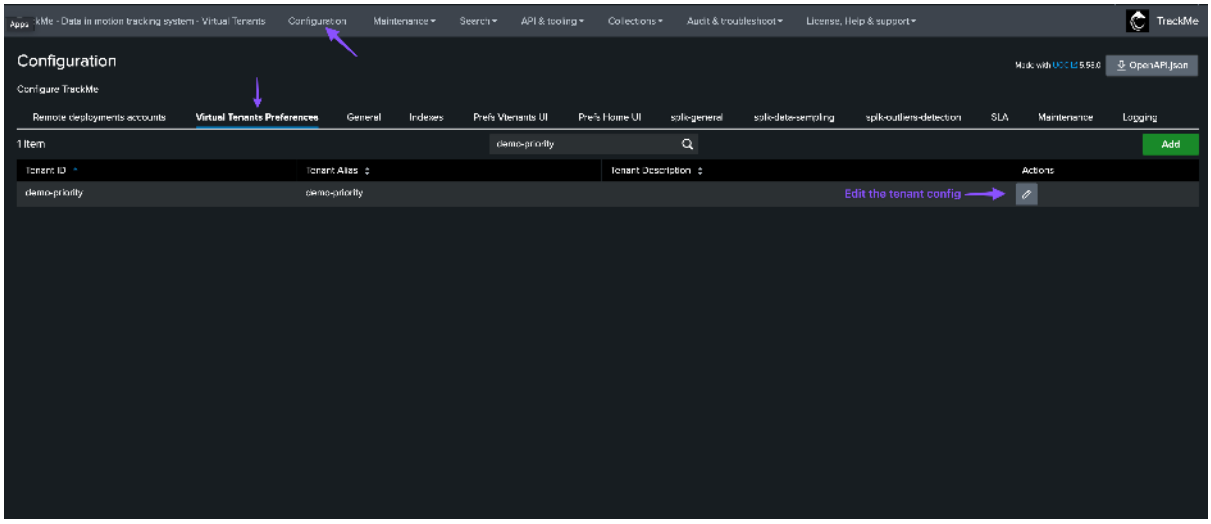
Priority at discovery time

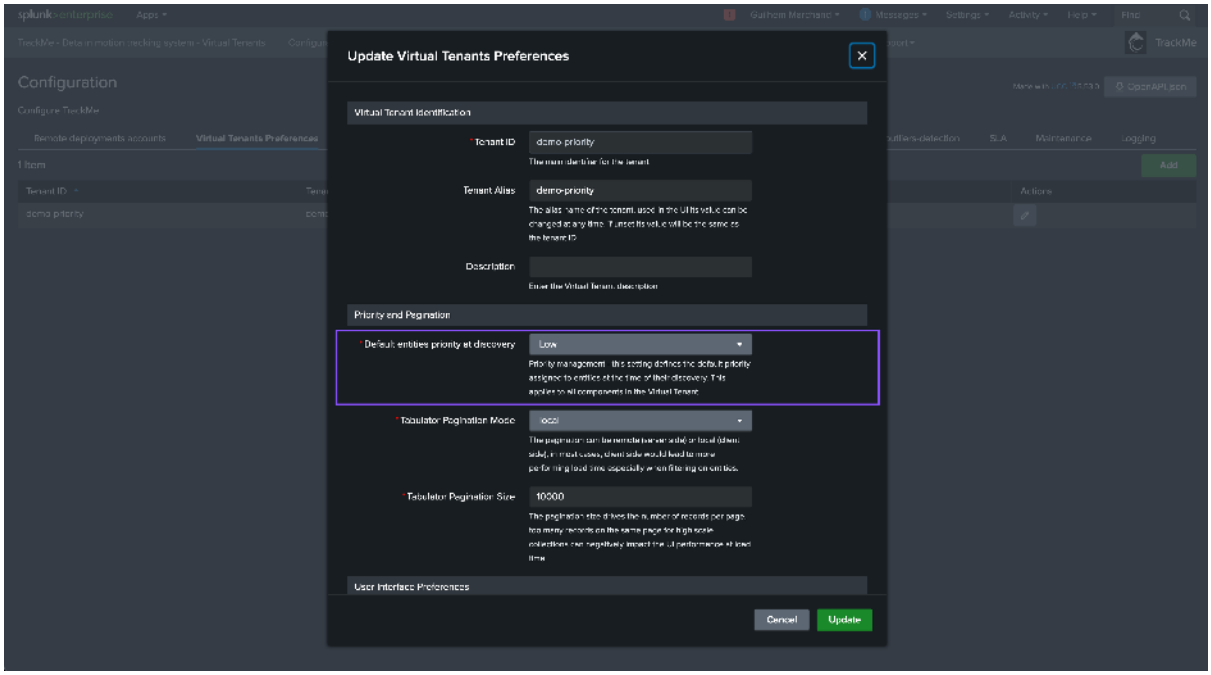
TrackMe applies the priority level at discovery time, which is configurable per Virtual Tenant:

When a Virtual Tenant is created, you can define the default priority that will be applied to entities as they are discovered:



Once the Virtual Tenant is created, you can update the default priority level in the Virtual Tenant's settings:

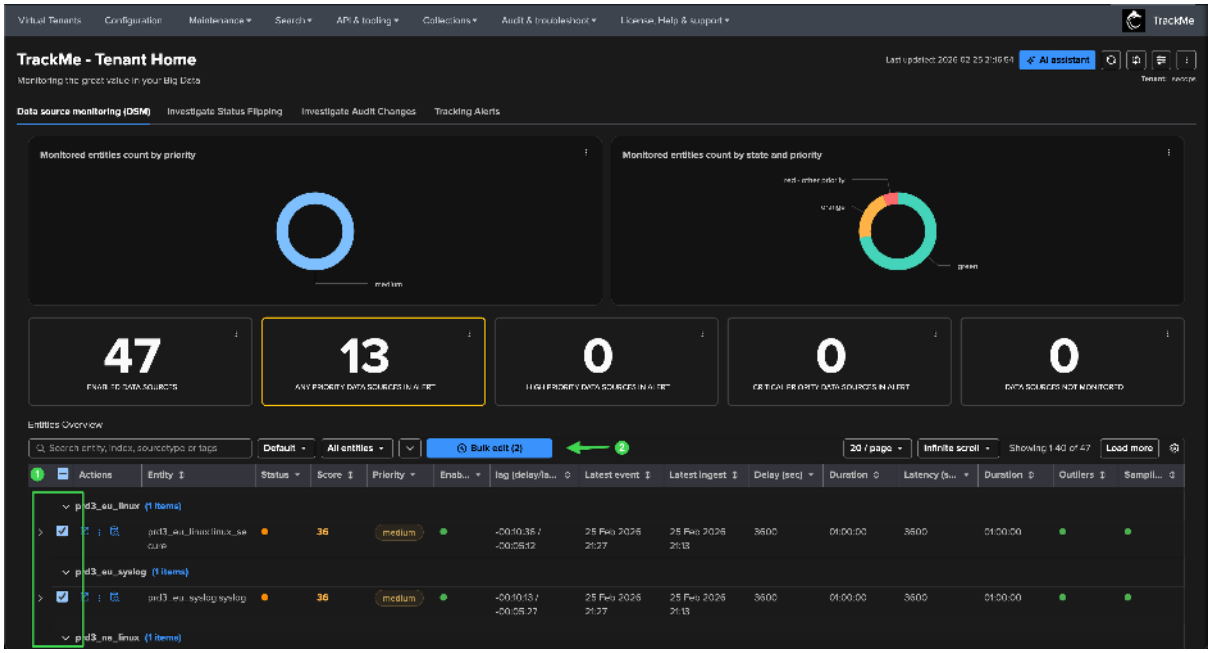

```

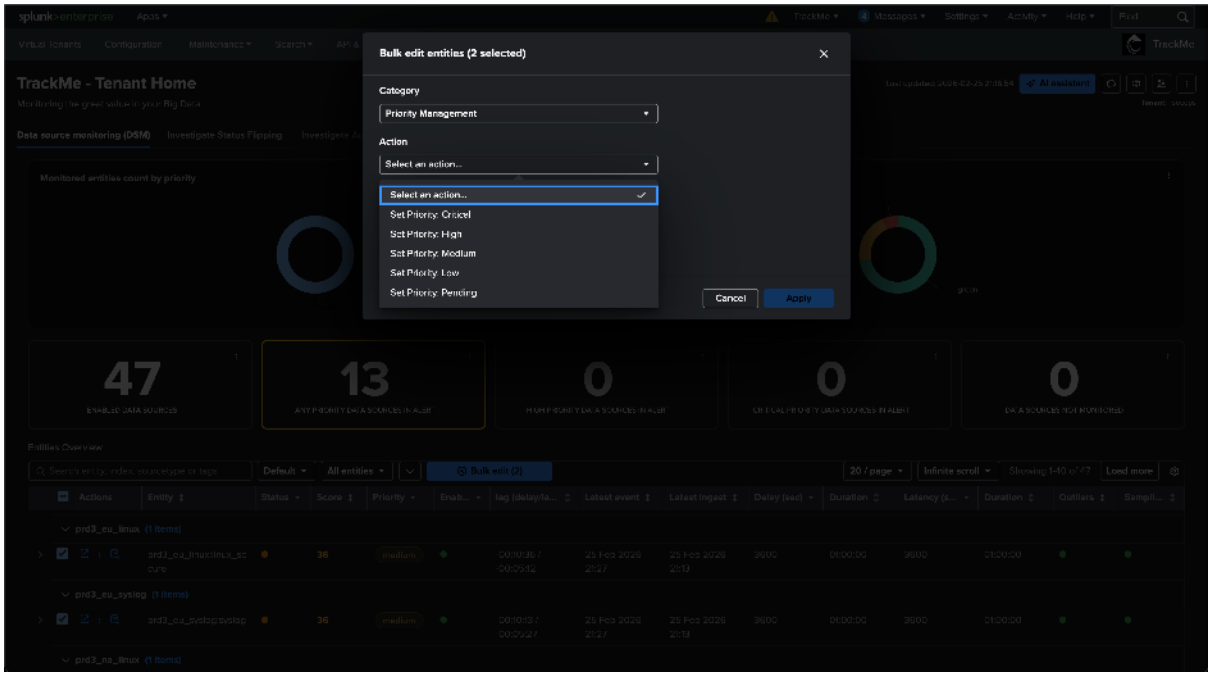


Managing priority at the entity level

You can update the priority level of an entity in different ways:

- Click on the 3-dot menu on the entity row and select *Priority Management*
- Open the entity view, click on the 3-dot menu and select *Priority Management*

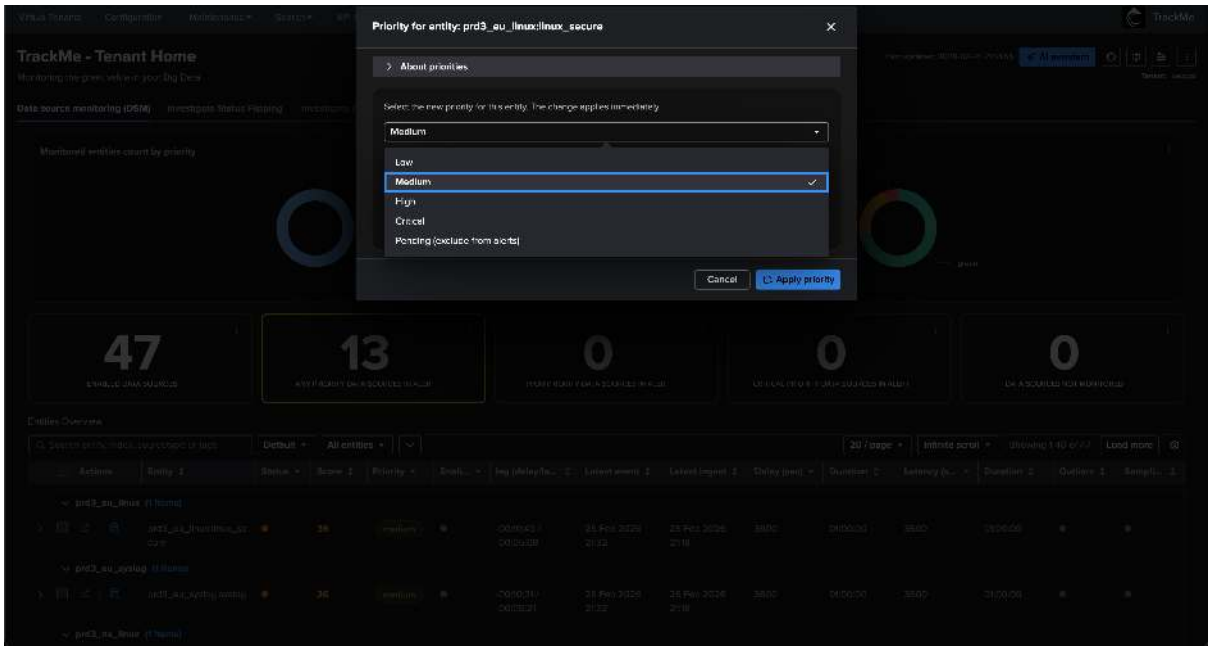


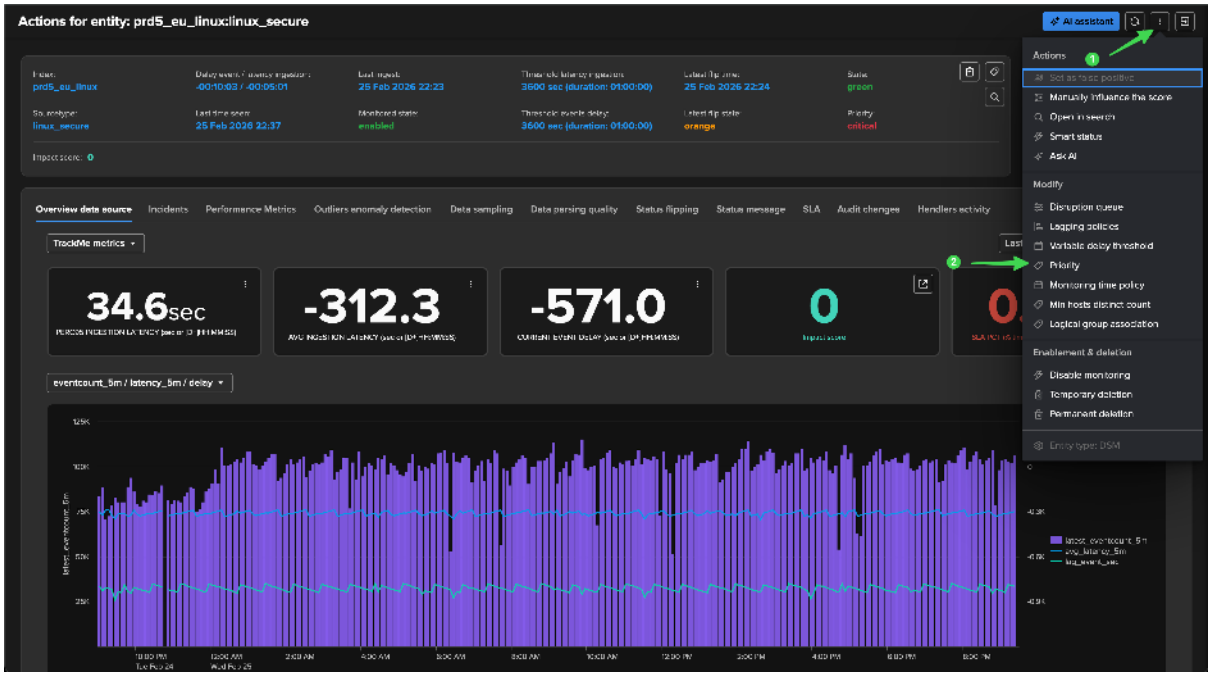


Updating entities in bulk

To update entities in bulk:

- Check the boxes of entities to be updated, or search for a pattern in the search bar and check the top header checkbox to select all matching entities
- The bulk edit buttons appears
- Select *Priority Management* and the wanted priority level





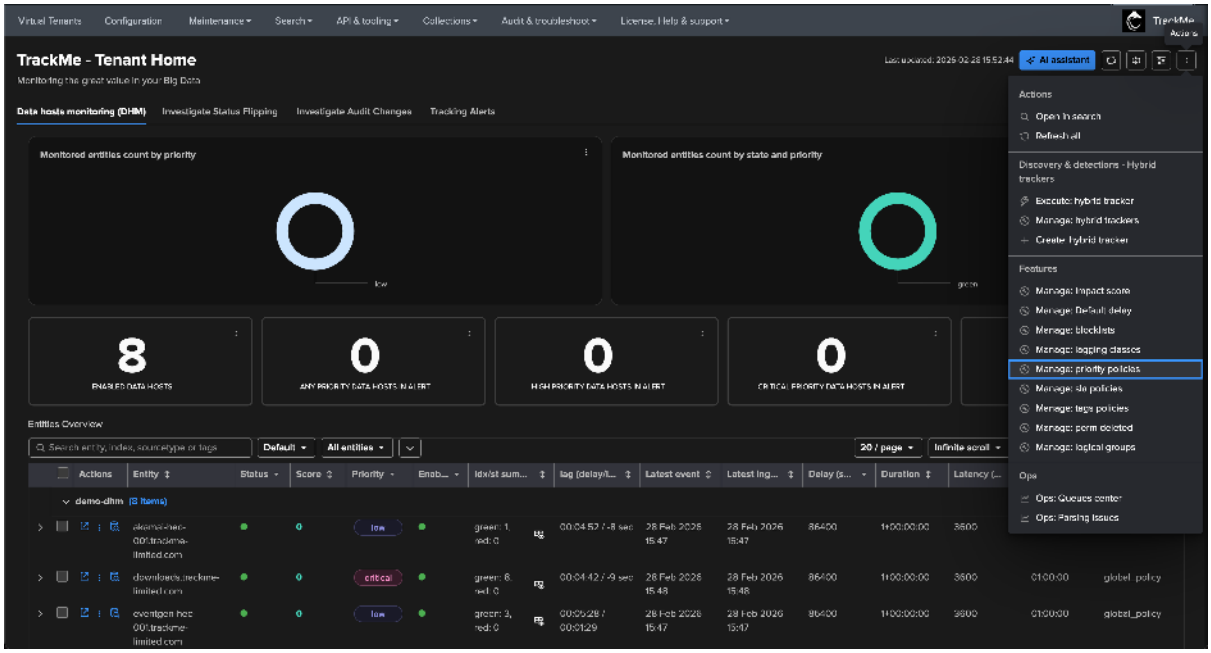
Managing priority via policy

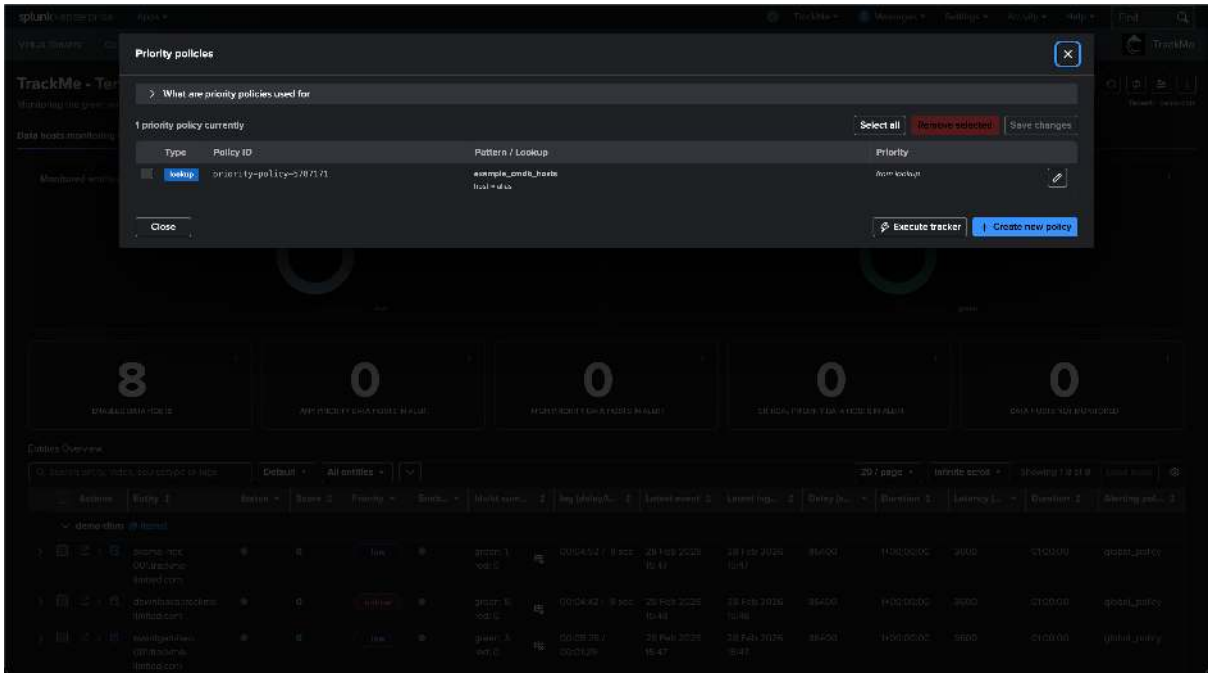
TrackMe supports the management of priority levels via policy, which can be defined per Virtual Tenant:

- Priority policies allow you to automate priority assignment across entities using **regex-based**, **lookup-based**, or **search-based (SPL)** rules.
- **Regex-based policies** match regular expressions against a configurable entity field (default: the entity name).
- **Lookup-based policies** leverage existing Splunk lookups (CSV or KVstore) to assign priorities based on field mappings, making it easy to integrate with a CMDB or asset inventory.
- **Search-based (SPL) policies** match against the live results of a Splunk search, mapped exactly like a lookup.
- Matching entities are automatically updated with the priority level defined in the policy.
- If multiple policies match a given entity, the **highest priority level takes precedence** across all policy types.
- An entity managed by policies can still be updated manually, and the policy will not override the manual update.
- TrackMe will show an informational message in the entity screen, displaying the policy managing the entity, the requested priority level, and the effective priority level.

Accessing the policy management screen:

- From any Tenant Home page, click the **priority icon** in the left-hand panel, then select **Manage priority policies**





Lookup-based policies

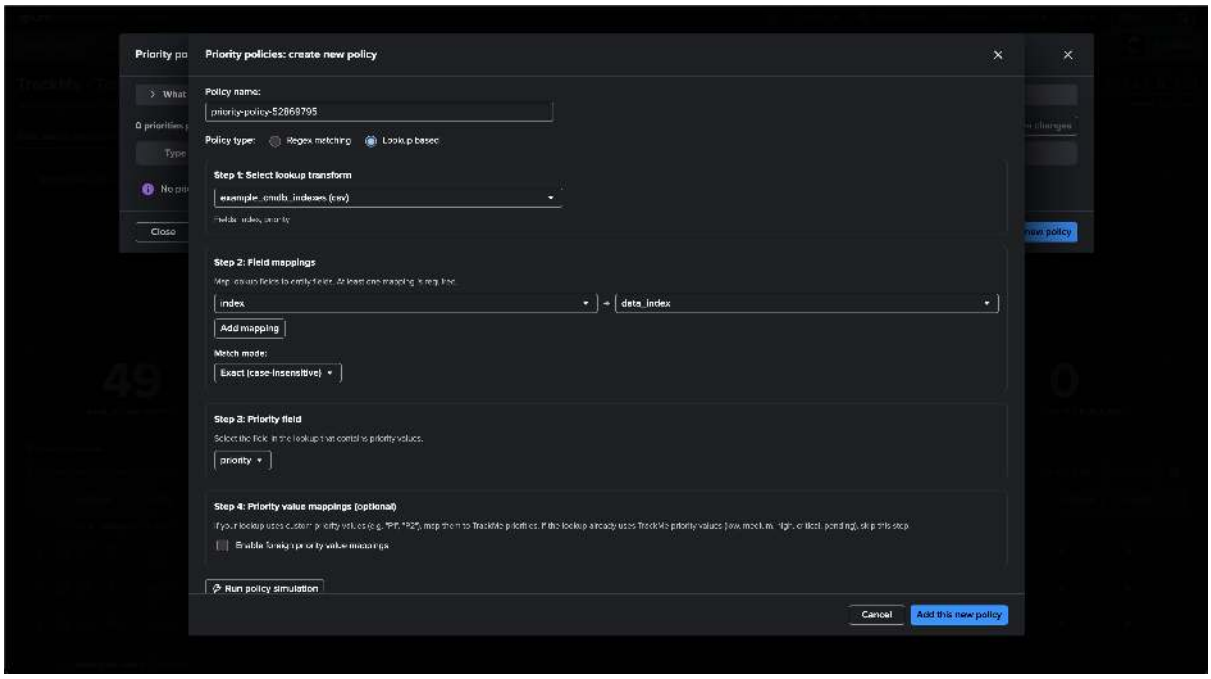
Lookup-based policies leverage existing Splunk lookup transforms (CSV files or KVstore collections) to assign priorities to entities based on field mappings. This is the recommended approach when integrating with an external CMDB, asset inventory, or any structured data source.

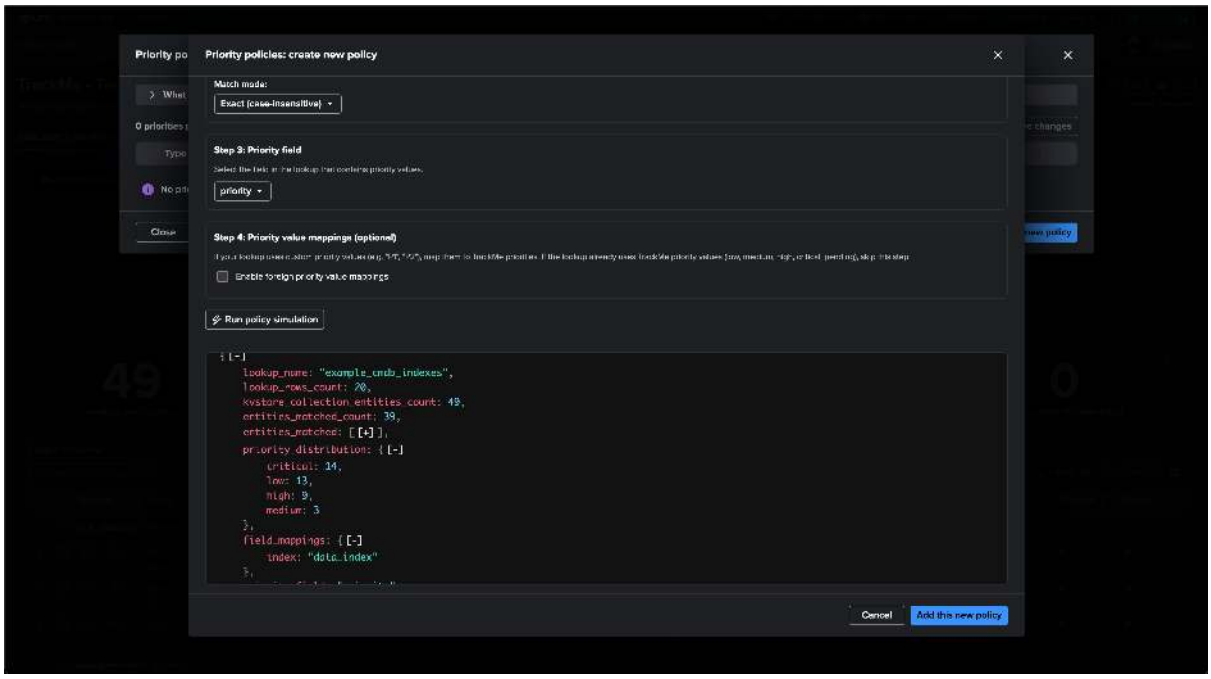
Creating a lookup-based policy:

- Click **Create new policy** and select the **Lookup** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)
- Select a **Splunk lookup transform** from the dropdown — TrackMe lists all available lookup transforms in your Splunk environment
- Configure **field mappings** to map lookup fields to entity fields — for example, map the lookup field `index` to the entity field `data_index`, and the lookup field `host` to the entity field `object`

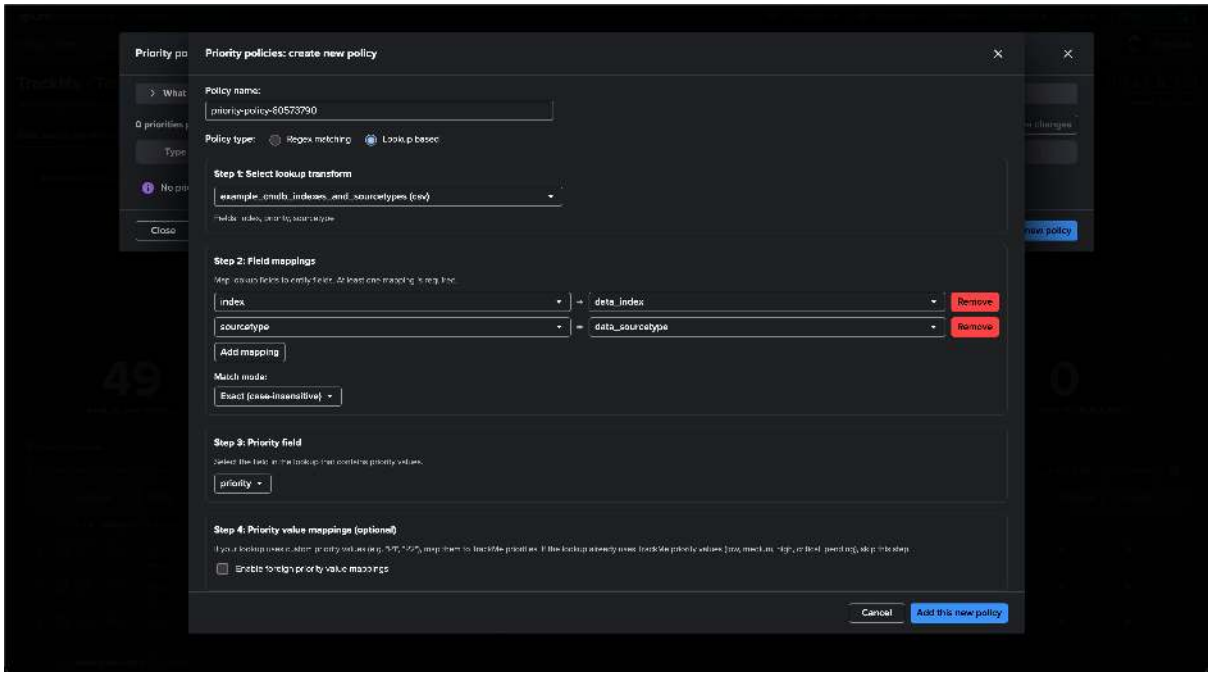
- Select the **priority field —** the field in the lookup that contains the priority value
- Optionally configure **priority value mappings** to translate foreign priority values to TrackMe's format (e.g., `P1` → `critical`, `P2` → `high`)
- Select the **match mode**: **Exact** (case-insensitive string matching) or **Wildcard** (supports `*` and `?` patterns)
- Click **Simulate** to preview which entities would be matched and what priorities would be assigned

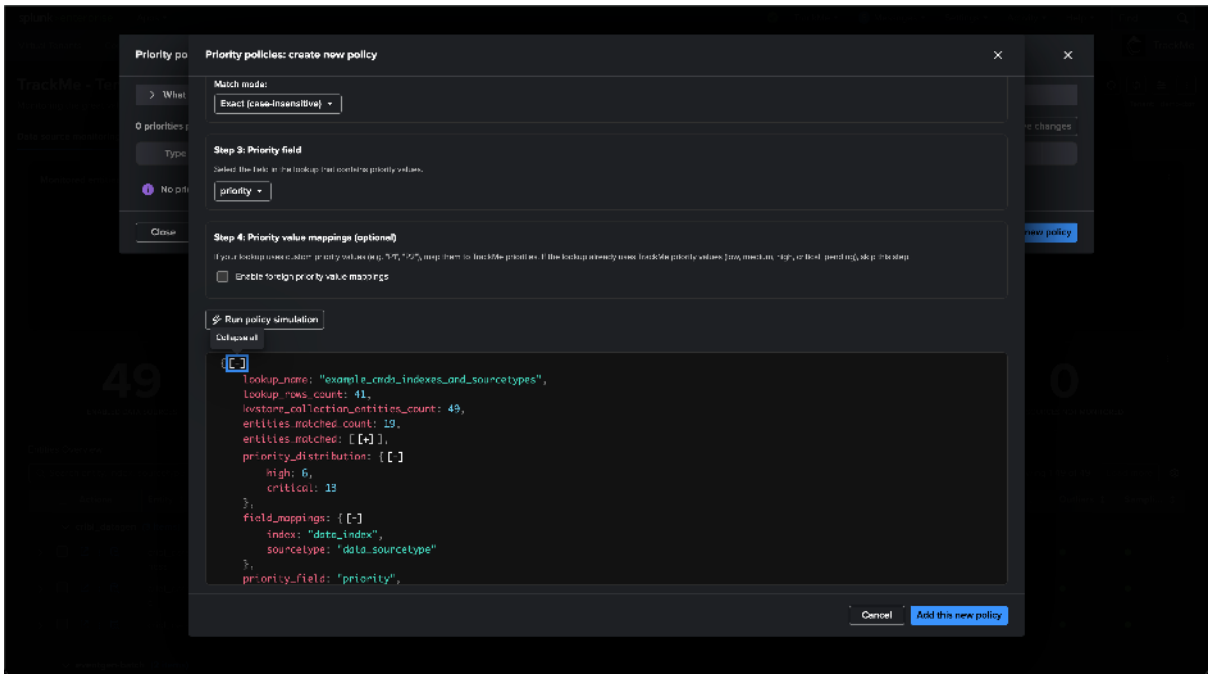
Data source (dsm) example, with a lookup file containing indexes and their priority:



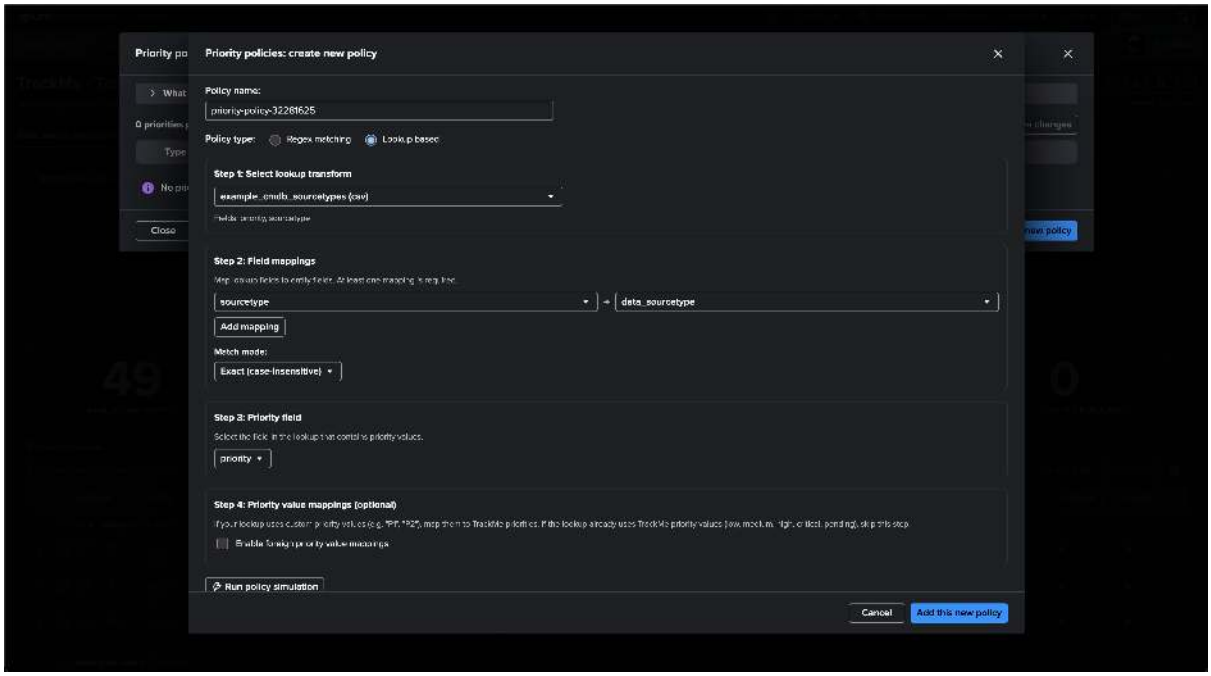


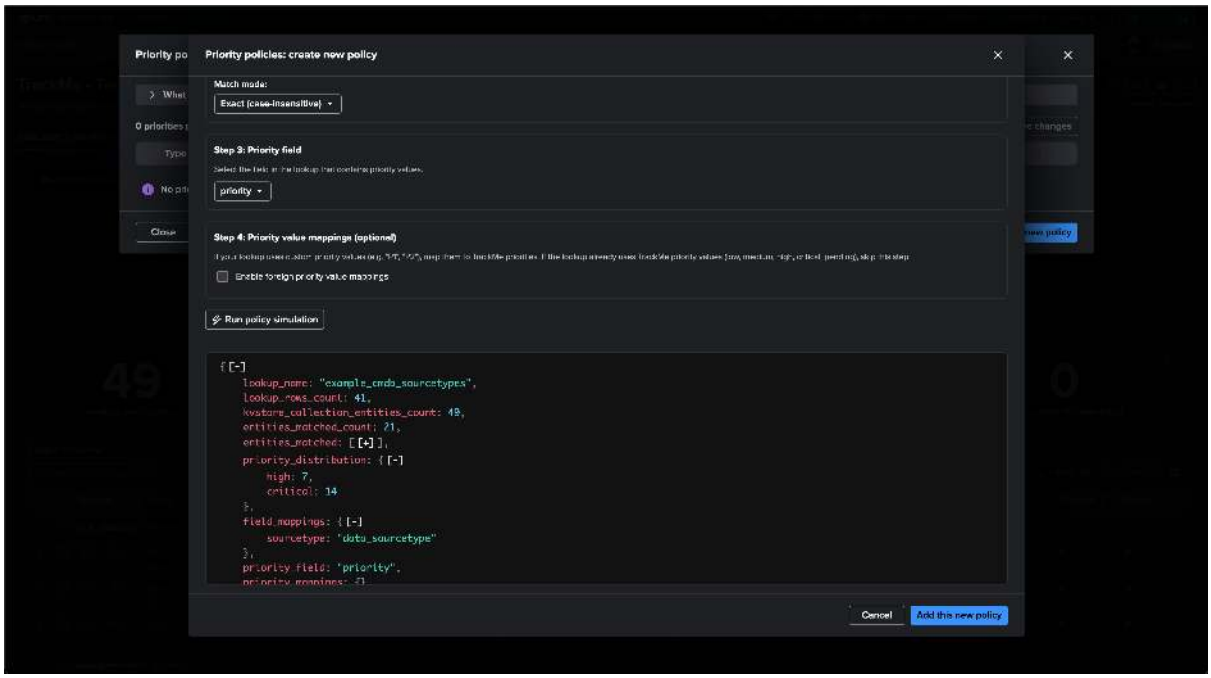
Data source (dsm) example, with a lookup file containing indexes, sourcetypes and their priority:



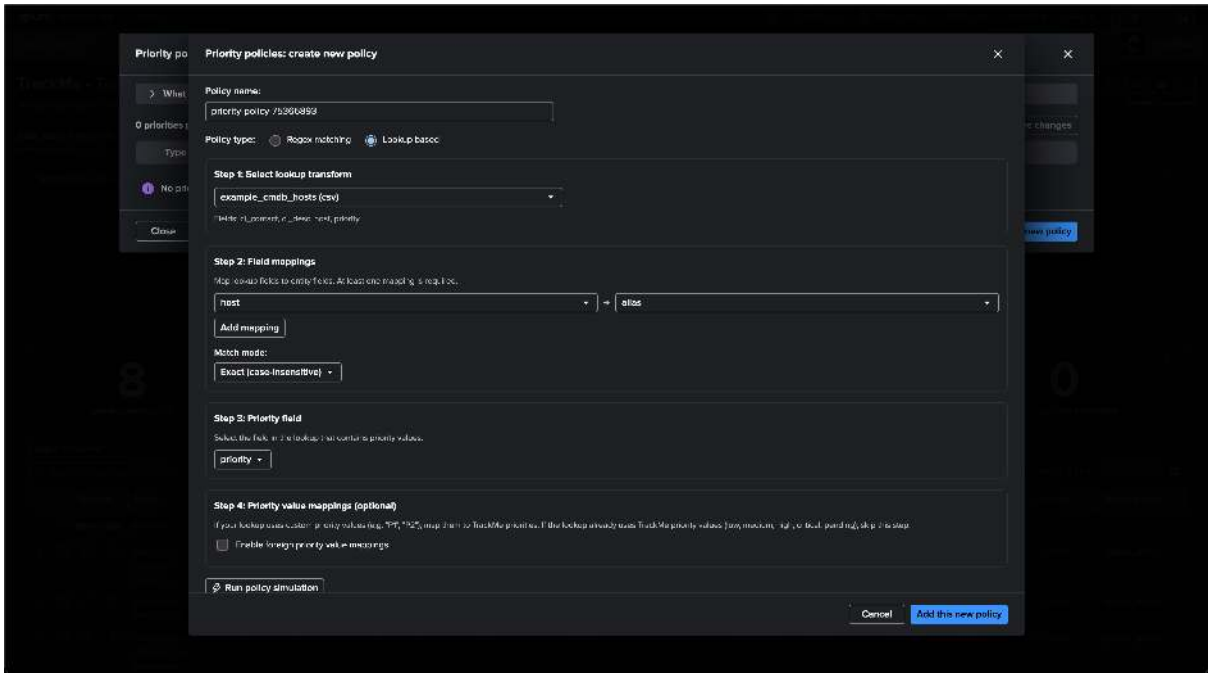


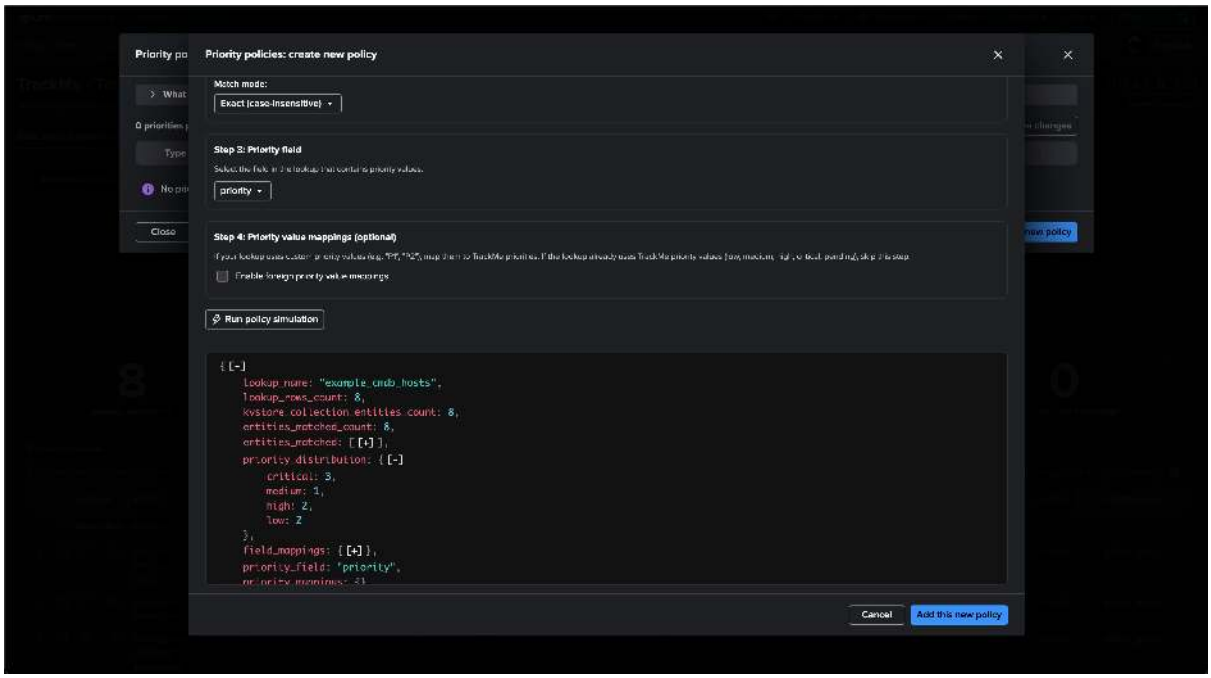
Data source (dsm) example, with a lookup file containing sourcetypes and their priority:





Hosts tracking (dhm) example, with a lookup file containing host names and their priority:





Lookup policy configuration details

- **Field mappings**: Define how lookup fields map to entity fields. All mapped fields must match for an entity to be selected. For example, if you map `index` → `data_index` and `host` → `object`, both conditions must be satisfied.
- **Priority field**: The column in the lookup containing the priority value (e.g., a `severity` or `priority` column).
- **Priority value mappings** (optional): If your lookup uses custom values like `P1`, `P2`, `P3`, you can map them to TrackMe priority levels (`critical`, `high`, `medium`, `low`, `pending`). If no mappings are configured, the lookup values must already match TrackMe's priority names.
- **Match modes**: `Exact` performs case-insensitive string comparison. `Wildcard` supports `*` (matches any characters) and `?` (matches a single character) for flexible pattern matching.

Advanced matching (lookup & search policies)

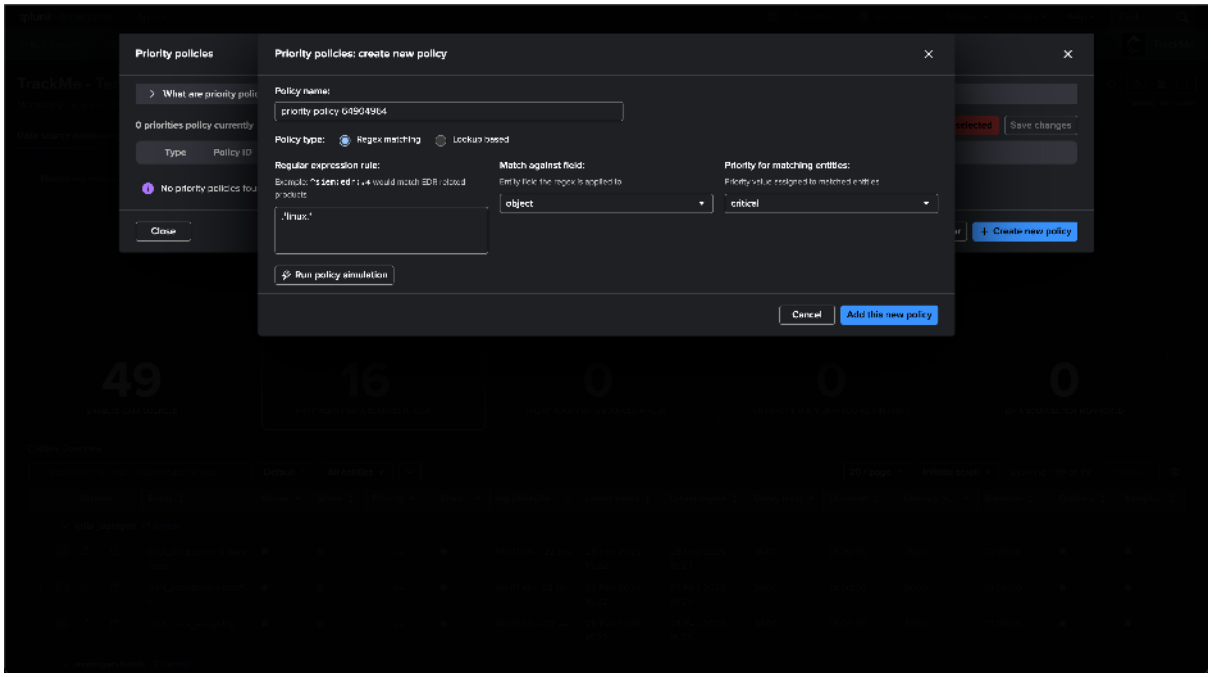
- **Multiple match blocks (OR/AND).** A single set of field mappings is AND-ed (all must match). You can instead define several match blocks that are **OR-combined** — an entity matches if it satisfies *any* block — useful when the same identifier lives in different lookup columns.
- **DHM `asset` multivalue.** For DHM, match against the `asset` multivalue field to recognise a host by *any* of its known identifiers against a hostname-keyed CMDB.
- **Wildcard foreign values.** In wildcard match mode, the lookup / search column values are treated as `*` / `?` patterns matched against the entity values.

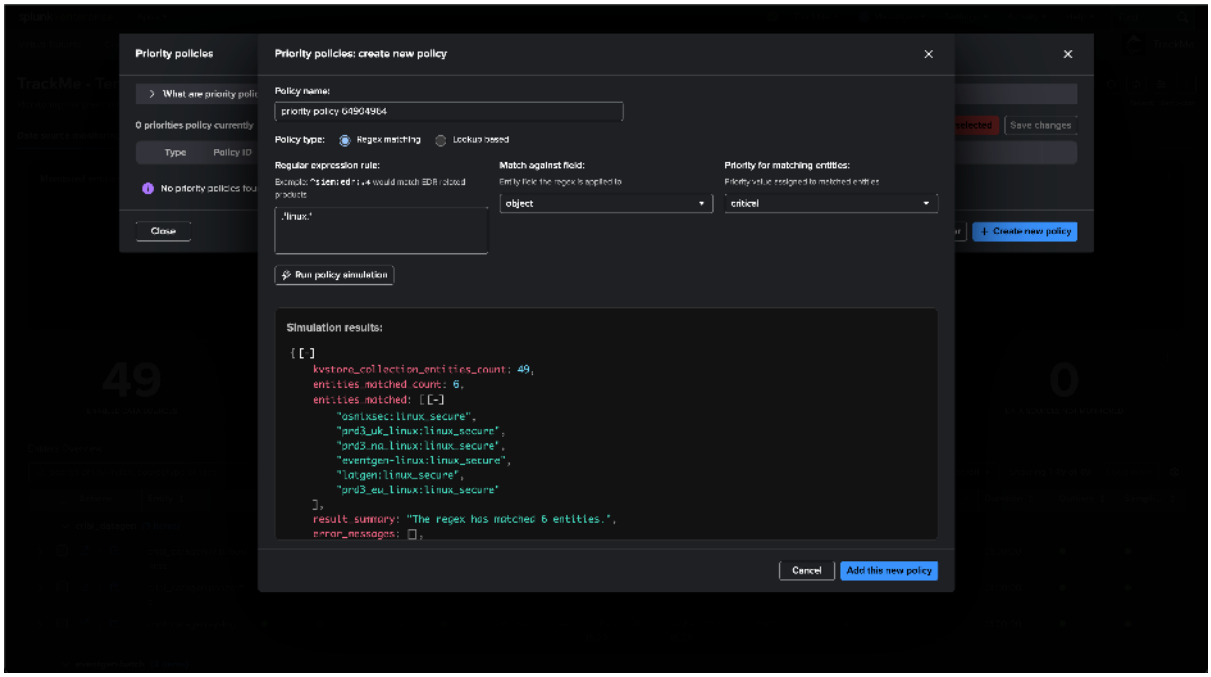
Regex-based policies

Regex-based policies match a regular expression pattern against an entity field. When the pattern matches, the configured priority value is assigned to the entity.

Creating a regex-based policy:

- Click **Create new policy** and select the **Regex** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)
- Enter the **regex pattern** to match against entities
- Select the **match against field** — by default, the regex matches against the `object` field (entity name), but you can choose any available field such as `alias`, `data_index`, `data_sourcetype`, or enter a custom field name
- Select the **priority level** to assign to matched entities
- Click **Simulate** to preview which entities would be matched before saving





About the Match against field option

- By default, regex policies match against the `object` field, which is the entity name.
- You can target any entity field available for the component, such as `alias`, `data_index`, `data_sourcetype`, or even a custom field name.
- This is particularly useful when you want to assign priorities based on entity attributes rather than their names. For example, matching all entities from a specific index using `data_index`.

Search-based (SPL) policies

A third method matches entities against the **results of an SPL search** rather than a static lookup — useful when the mapping is computed (a join, an enrichment, a derived field) rather than stored in a CSV / KV Store. Create one with **Create new policy → Search** mode and provide:

- the **search query** (raw SPL) and, optionally, its **earliest / latest** time range (default `-5m` / `now`);
- the same **field mappings** as a lookup policy — the search's result columns are matched against entity fields exactly as a lookup table would be;
- the **priority field** in the results that carries the value to assign, with the same optional value mappings and match modes.

In short, a search policy is a lookup policy whose "table" is produced live by SPL on each run, so it supports the same field-mapping, value-translation, wildcard, OR/AND-block, and simulate options.

Priority hierarchy and conflict resolution

When multiple policies (regex, lookup, or search) match the same entity, TrackMe applies the **highest priority level** across all matching policies:

Table 14: Priority hierarchy (highest to lowest)

Priority Level	Numerical Value
pending	5 (special)
critical	4
high	3
medium	2
low	1

For example, if a regex policy assigns `medium` and a lookup policy assigns `critical` to the same entity, the entity will receive the `critical` priority.

Once a priority is assigned by a policy, TrackMe sets the `priority_updated` flag on the entity, which prevents the default priority logic from overriding the policy-assigned value during regular evaluation cycles.

Running the policy tracker

After creating one or more policies, you can execute the policy tracker to apply all configured policies to your entities:

- Click the **Run policy tracker** button in the policy management screen
- TrackMe will evaluate all regex, lookup and search policies against all entities in the selected component
- A summary of the results is displayed, including the number of entities updated, matched, and any errors encountered

Entities managed by policies

When an entity is managed by a policy, TrackMe displays an informational message in the priority management screen, showing the policy identifier and the requested priority level:

You can still override the priority level manually at the entity level — the policy will not overwrite manual changes.

Accessing entities managed by policies using trackmegetcoll:

Entities managed by policies can be identified using the `priority_reason` field:

Notes:

- Update the `tenant_id` and `component` to match your environment
- The following SPL can be accessed via the "Search table" button in TrackMe's UI; it leverages the real-time decision maker and TrackMe REST API

```
| trackmegetcoll tenant_id=feeds-secops component=dsm

| where isnotnull(priority_reason) AND match(priority_reason, "^priority_policy_id:
↪")
| table object, priority, priority_reason
```

Managing priority externally

TrackMe supports the management of priority levels externally:

- External management allows you to update the priority level of entities using Splunk and any logic of your own.

- Priority policies take precedence over external management.
- If an entity is managed externally, the priority level can still be updated manually, and external management will not override the manual update.
- TrackMe will show an informational message in the entity screen, displaying the external management managing the entity, the requested priority level, and the effective priority level.

Hint

Consider using lookup-based priority policies instead

With the introduction of **lookup-based priority policies** (see *Managing priority via policy*), the same use cases previously handled through external priority management can now be achieved directly within TrackMe's built-in policy framework. Lookup-based policies offer several advantages:

- **Easier configuration** — no custom SPL searches or scheduled searches to maintain
- **Full built-in control** — policies are managed, simulated, and applied through the TrackMe UI
- **Continuous enforcement** — the built-in priority tracker automatically applies policies, ensuring newly discovered entities are handled without delay
- **Audit and visibility** — policy-managed entities are clearly identified in the UI with their associated policy

External priority management remains available for advanced use cases requiring custom logic beyond what policies can express.

Example of SPL:

- Update the tenant_id
- Update the search to match your needs, lookup files and logic

In this example, the lookup field contains the index names which we use as the lookup key to define the priority_external field.

```
| inputlookup trackme_dsm_tenant_feeds-secops | eval keyid=_key

``` In this example, we leverage a Splunk lookup file referencing indexes and used␣
↪to define the field priority_external ```
| lookup feeds_priorities.csv index as data_index OUTPUT priority as␣
↪priority_external
| where isnotnull(priority_external)

``` The field priority_reason will be used by TrackMe to display an informational␣
↪message ```
| eval priority_reason="lookup: feeds_priorities.csv"

``` Finally, the KVstore records will be updated; schedule this search so that any␣
↪newly discovered entity will retrieve the expected externally managed priority␣
↪```
| outputlookup append=t key_field=keyid trackme_dsm_tenant_feeds-secops
```

In this example, the lookup field contains both index and sourcetypenames which we use as the lookup key to define the priority_external field.

```
| inputlookup trackme_dsm_tenant_feeds-secops | eval keyid=_key

``` In this example, we leverage a Splunk lookup file referencing indexes and used␣
↪to define the field priority_external ```
```

(continues on next page)

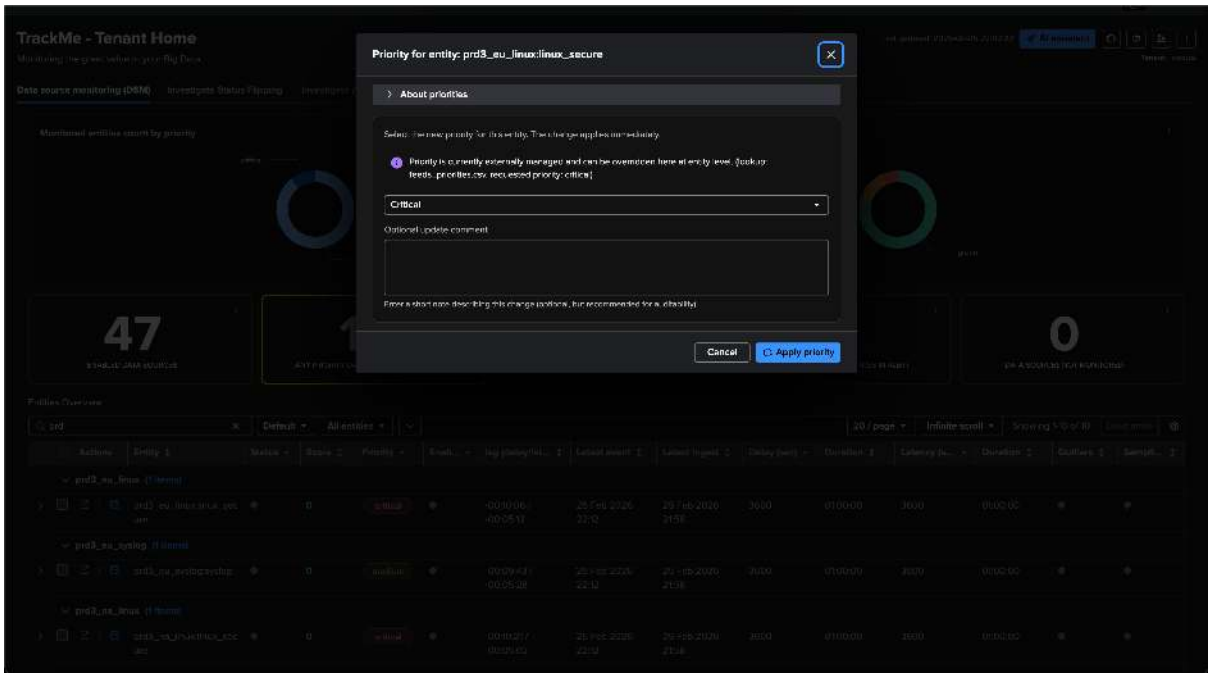
(continued from previous page)

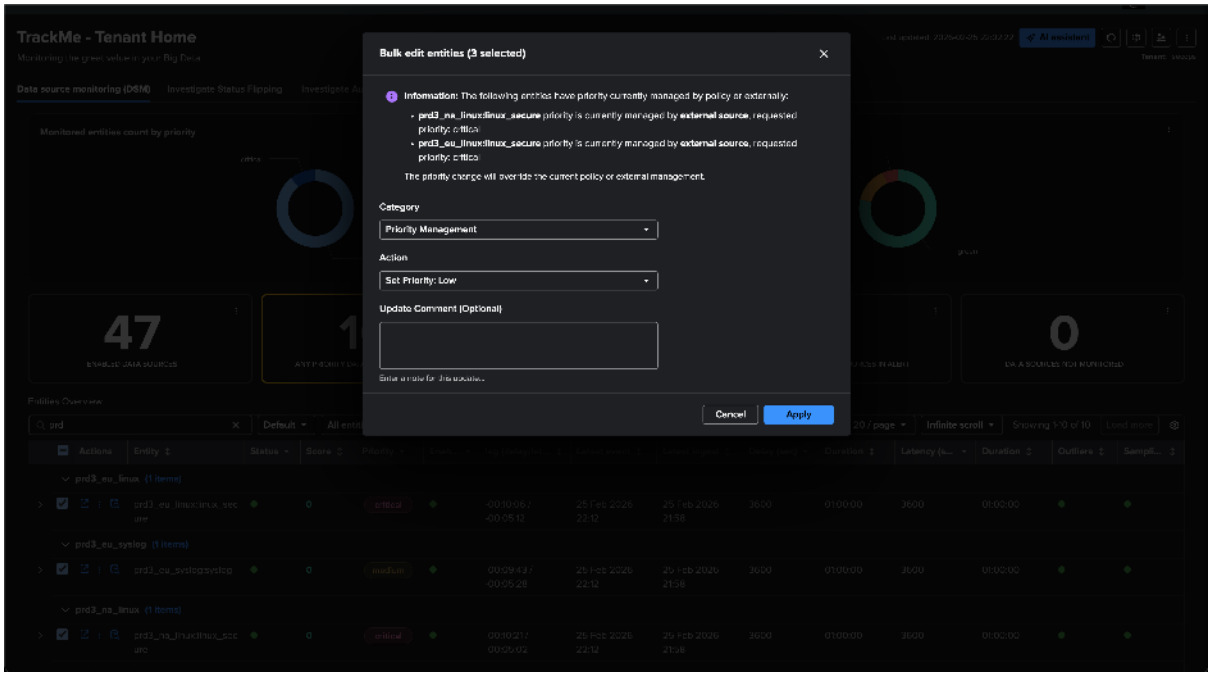
```
| lookup feeds_priorities.csv index as data_index, sourcetype as data_sourcetype␣
↪OUTPUT priority as priority_external
| where isnotnull(priority_external)

``` The field priority_reason will be used by TrackMe to display an informational␣
↪message ```
| eval priority_reason="lookup: feeds_priorities.csv"

``` Finally, the KVstore records will be updated; schedule this search so that any␣
↪newly discovered entity will retrieve the expected externally managed priority␣
↪```
| outputlookup append=t key_field=keyid trackme_dsm_tenant_feeds-secops
```

TrackMe automatically detects that the priority is externally managed, and will show an informational message when accessing the priority management screen, either on a per entity basis or via bulk edit:





Entities managed by policies can be identified using the *priority_policy_id* field:

Notes:

- Update the tenant_id
- The following SPL can be accessed via the "Search table" button in TrackMe's UI; it leverages the real-time decision maker and TrackMe REST API

```
| trackmegetcoll tenant_id=feeds-secops component=dsm

| where isnotnull(priority_external)
| table object, priority*
```

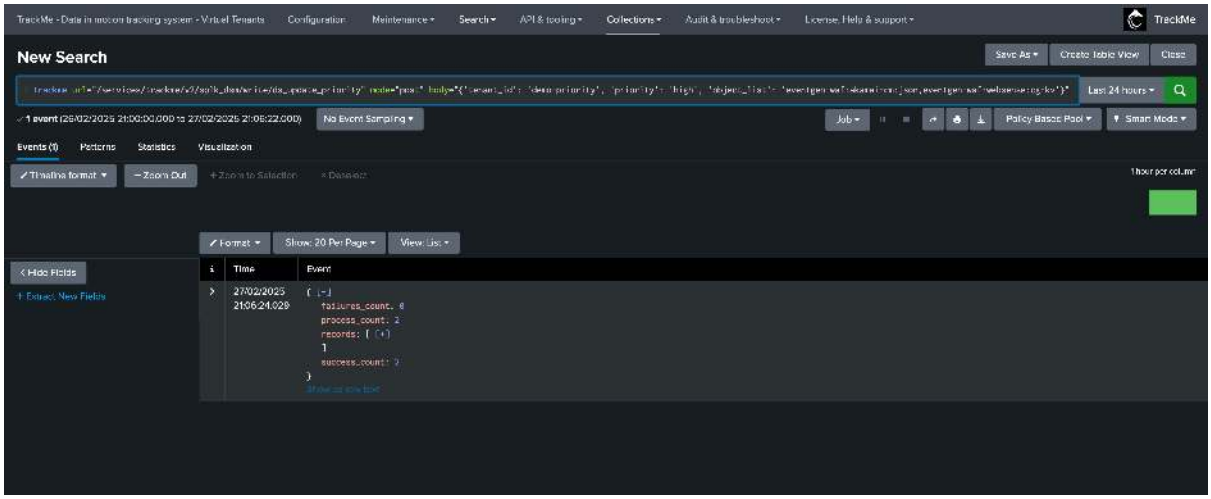
Additional priority related resources

Updating entities in SPL and REST API

You can also update the priority level of entities using SPL and the REST API, or by any other means that can interact with the REST API:

Example of SPL:

```
| trackme url="/services/trackme/v2/splk_dsm/write/ds_update_priority" mode="post"␣
↪body="{'tenant_id': 'demo-priority', 'priority': 'high', 'object_list':
↪'eventgen-waf:akamai:cm:json,eventgen-waf:websense:cg:kv'}"
```



8.5.3 Tags

Tags are free-form string metadata you attach to entities. They are the workhorse of **triage, filtering, grouping, and automation** in TrackMe — a lightweight way to say "this entity belongs to *that* team / environment / domain / regulation" and then act on it everywhere. You can set them by hand or assign them automatically with tag policies.

What tags are

- **Free-form, flat strings** — no hierarchy and no enforced namespace. Adopt a naming convention (`team:auth`, `env:prod`, `owner:alice`, `domain:network`) to give them structure.
- An entity can carry **many** tags. Its effective tags are the **union** of policy-assigned tags and manually-set tags.

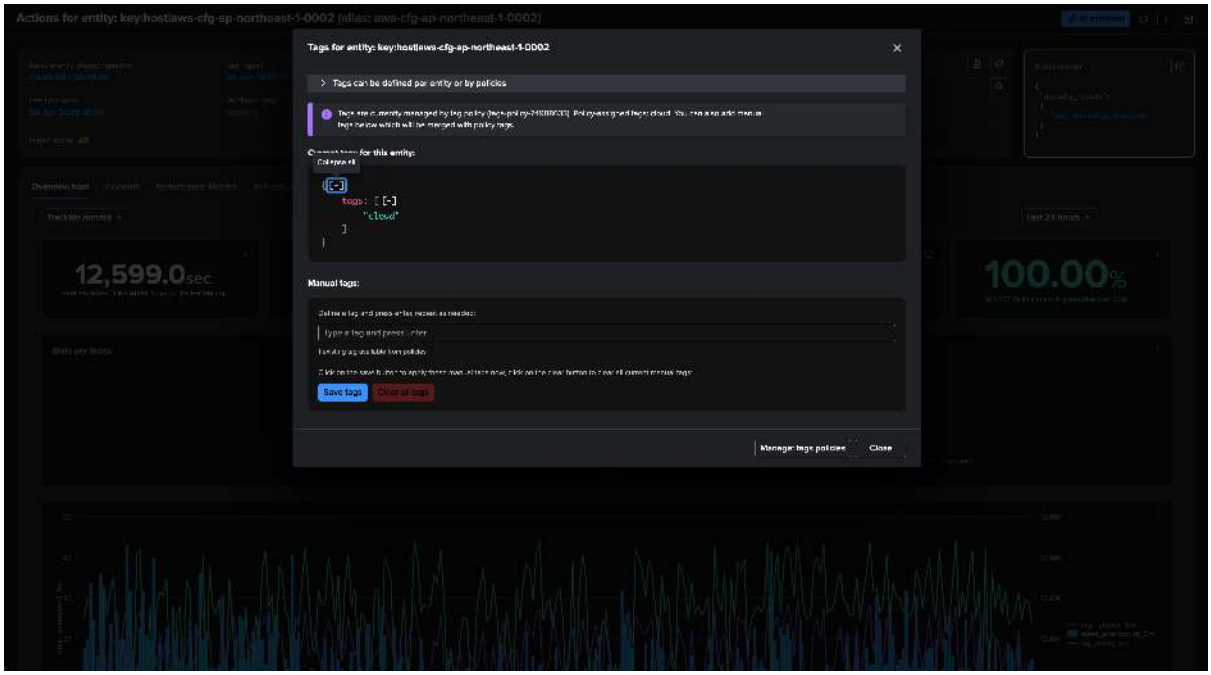
Common uses: team and ownership attribution (`team:payments`, `owner:alice`), environment segmentation (`env:prod` / `env:staging`), domain grouping (`domain:network`), and regulatory scope (`pci`, `gdpr`).

> **Note**
>
> Tags are resolved in **lowercase** and matched **case-insensitively**, so `team:Auth` and `team:auth` are the same tag — but still pick one convention, or tag sprawl will erode their value.

Setting tags

- **Manually**, per entity — manual tags are flagged and are **not** removed by policies.
- **By policy** — tag policies use the **same three methods** as priority (regex, lookup, SPL) and are **per component** (see *Tag policies*).

When both apply, manual and policy tags are **merged**. And when several tag policies match, their tags **merge** too (a union) — unlike priority, a second matching policy *adds* its tags rather than overriding. Removing a policy cleans only the tags *that* policy contributed; other policies' tags and your manual tags remain.



Important

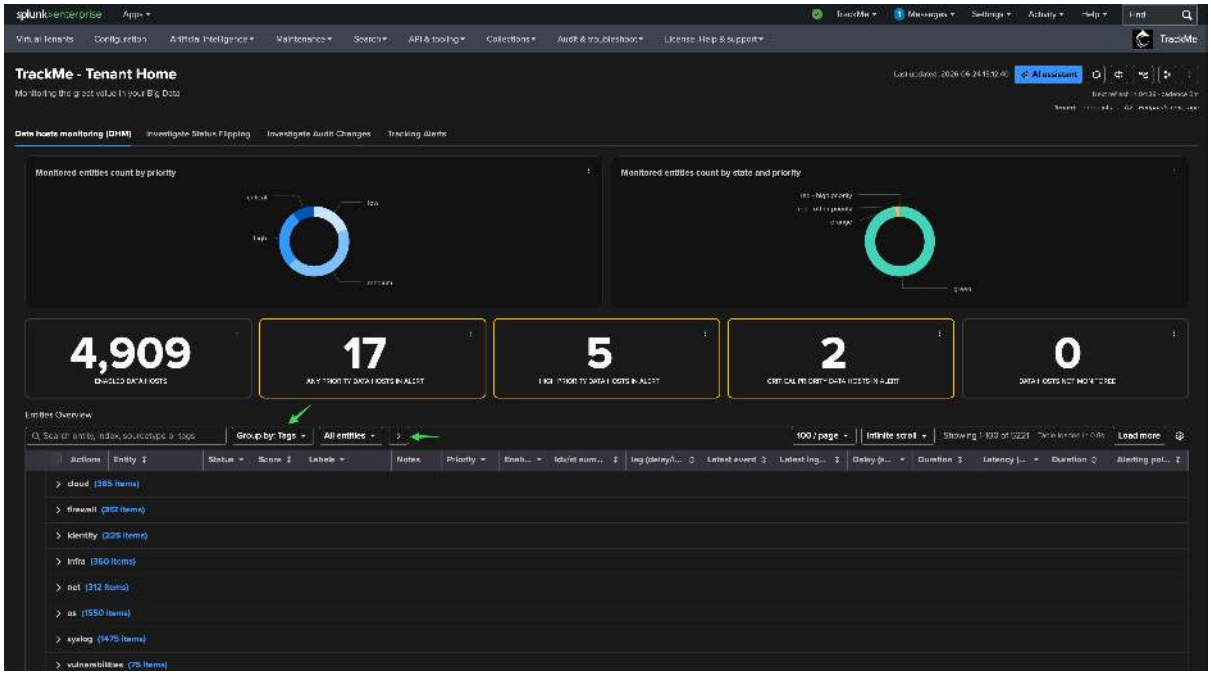
Tags are not labels. Tags are free-form, per-component, filterable strings. *Labels* are curated, colour-coded chips drawn from a per-tenant catalogue. Rule of thumb: a free-form filterable string is a *tag*; a curated coloured chip is a *label*.

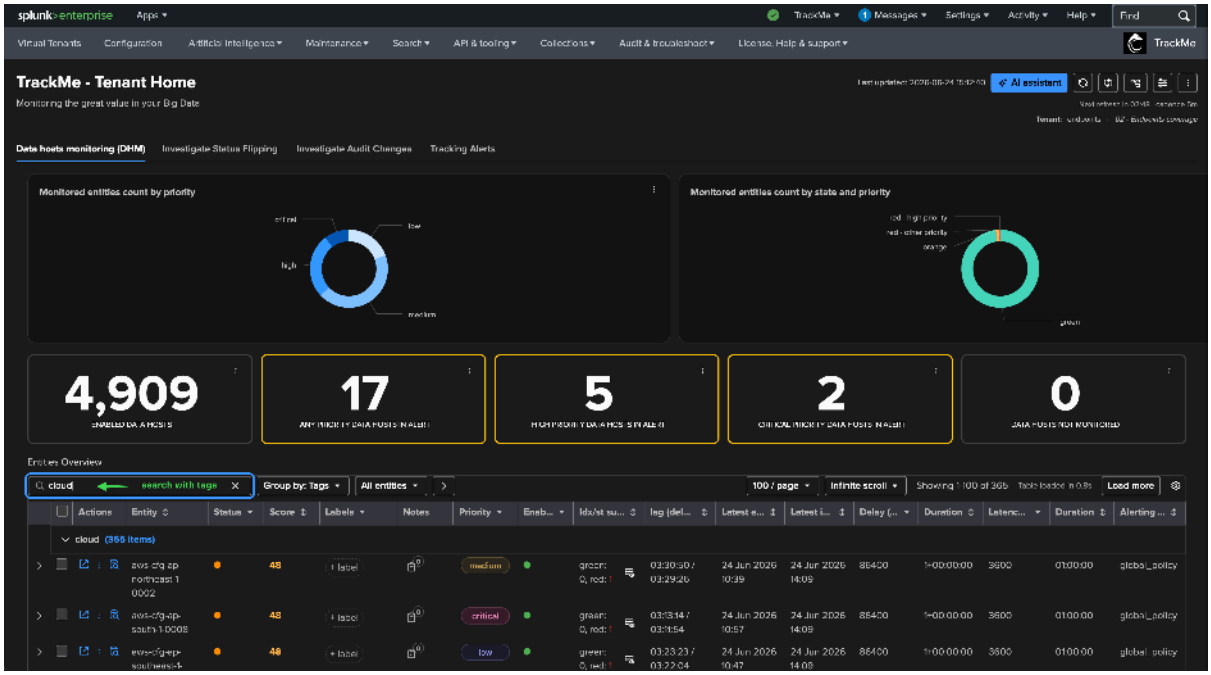
Where tags are used

Like priority, tags are **classification, not scoring** — they never change an entity's score or state. Their value is everything you can do *with* them:

Triage & filtering in the UI

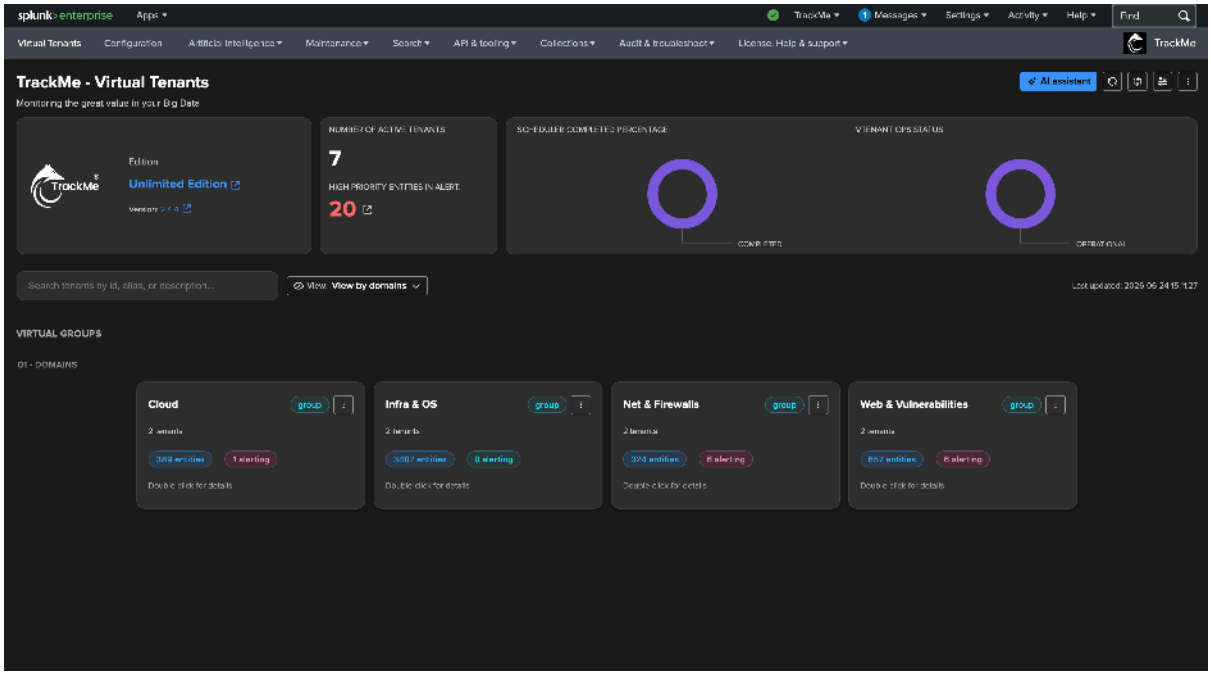
`tags` is a first-class **filter and search axis** on every entity table — type `tags=cloud` or `tags IN(env:prod, env:staging)` to narrow the view — and you can **group the entity list by tag**, turning a flat list into team- or domain-shaped sections. This is the fastest way to triage "show me everything for the network team".

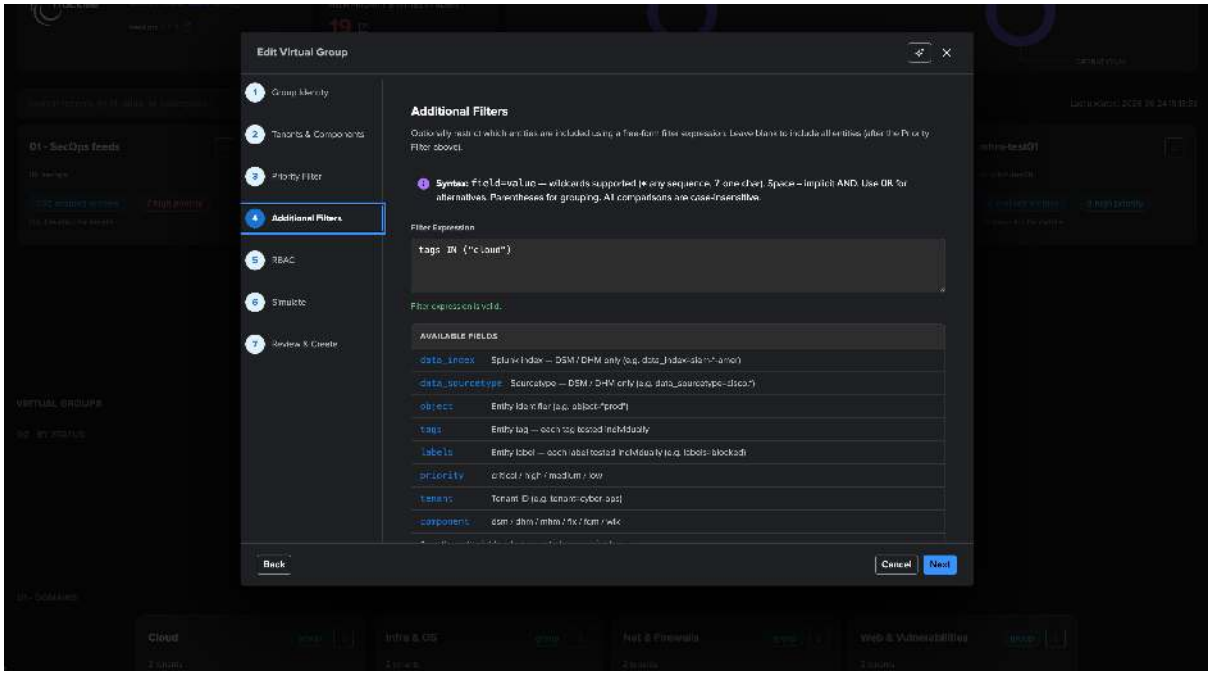




Grouping & Virtual Groups

A *virtual group* can scope its members with a filter expression that uses tags (for example `tags IN("cloud")`), so you can build a cross-tenant NOC view per domain or team — "all of the network estate", "everything tagged `pci`" — without any new monitoring.





Automation

Tags are a field in TrackMe's shared **filter language** (operators `=` and `IN (...)` with `*` / `?` wildcards, combined with `AND` / `OR` / `NOT` and parentheses, case-insensitive). That one language lets tags scope *auto-label rules*, *ML outlier detection*, and **AI Advisor automated actions** — so, for example, autonomous tuning only runs on `tags=team:payments`.

Bulk operations

Filter the entity table by tag, select the matches, and **bulk add, remove, or replace manual tags** across the whole selection — the practical way to retag at scale.

Alerting & notable events

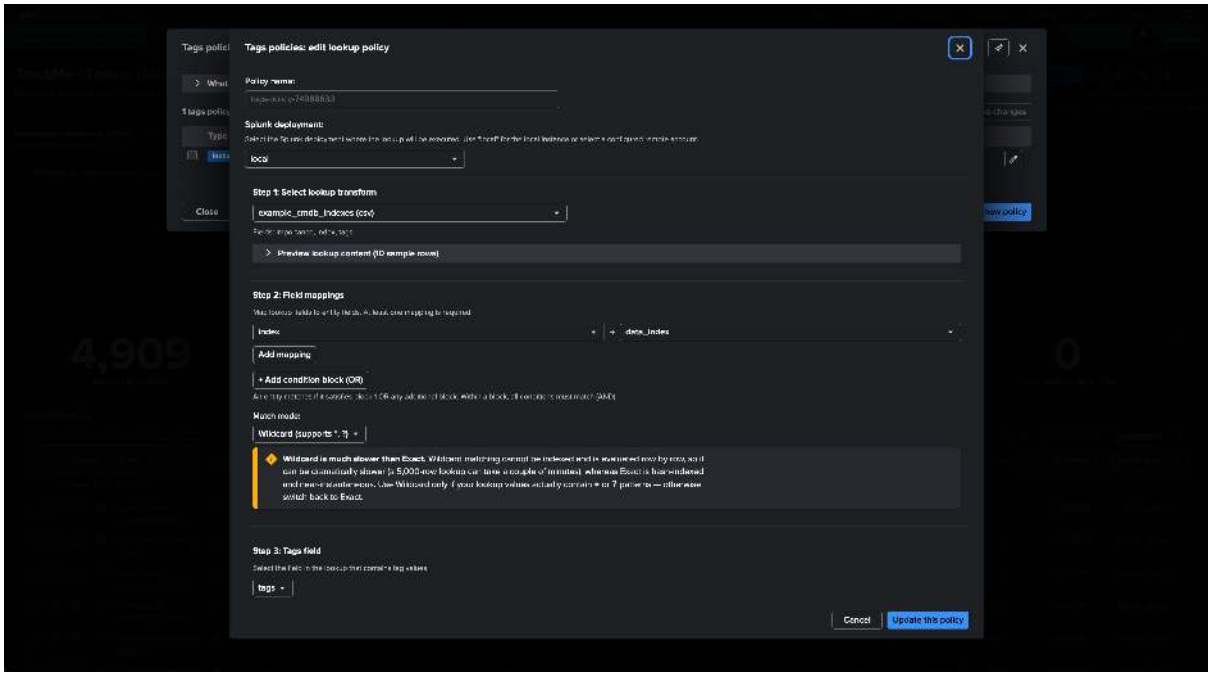
An entity's tags are **carried onto** its *stateful alerts* and *notable events* as context, so your Enterprise Security or correlation searches can route, group, and prioritise on them.

Note

TrackMe has **no built-in "tag → email recipient" routing** — recipients are set on the alert action. Tags enable routing *downstream*: they ride along on the alert and notable so your own ES correlation searches or automation can act on them. Tags also live on the entity record (they are **not** a metric dimension), so "report by tag" means filtering entities, not an `mstats … by tags` split.

Tag policies

A tag policy assigns tags automatically, matching entities by **regex**, a **Splunk lookup**, or an **SPL search**, per component. Policies *suggest*; manual tags always survive; and removing a policy auto-cleans only the tags it added.



See also

- *Tags — in depth* — the regex/lookup tag-policy procedures, the REST apply endpoint, KV Store internals, and troubleshooting.
- *Labels & auto-labels* — the curated chip feature, and how it differs from tags.
- *Priority* and *SLA policies* — the sibling policy types.
- *Virtual Groups* — virtual groups can filter on tags.

8.5.4 Tags — in depth

Tip

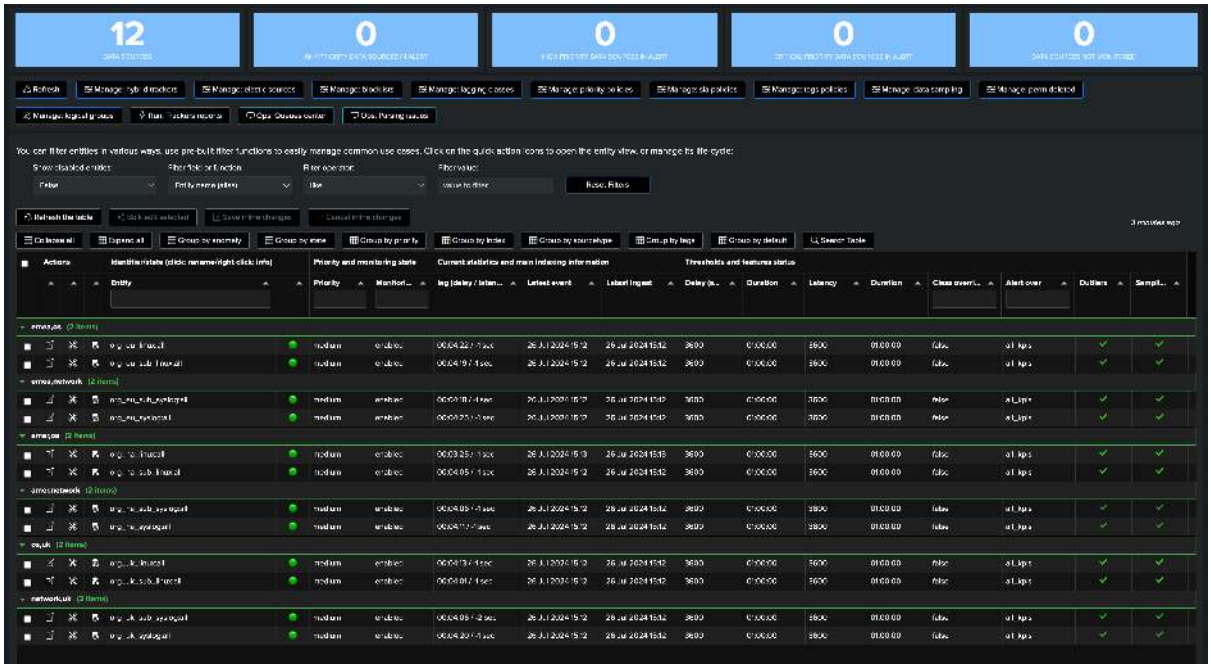
This is the **in-depth reference** for tags. For the readable overview, start with *Tags*. This page covers the regex and lookup-based tag-policy procedures, a worked example, the REST apply endpoint, the tracker job, KV Store internals, and troubleshooting.

Introduction to tags enrichments

Hint

About Tags in TrackMe

- Tags are used to provide additional context to the entities
- Tags can be used for filtering purposes in the user interface
- Tags can be used for alerting purposes
- Tags can be used for additional context in the user interface



Implementing tags enrichments in TrackMe (all components)

Extension of Tags features for all components in TrackMe 2.0.98

- Since TrackMe version 2.0.98, tags enrichments are available for all components in TrackMe with the exact same level of features, and user experience.

Tags enrichments in TrackMe can be defined by:

- **Tags policies:** Automated rules that assign tags to matching entities. Tags policies support three modes:
  - **Regex-based policies** match regular expressions against entity names to assign tags.
  - **Lookup-based policies** leverage existing Splunk lookups (CSV or KVstore) to assign tags based on field mappings, making it easy to integrate with a CMDB or asset inventory.
  - **Search-based (SPL) policies** match against the live results of a Splunk search, mapped like a lookup.
- **Manual Tags:** In addition, you can manually add tags to the entities from the user interface.

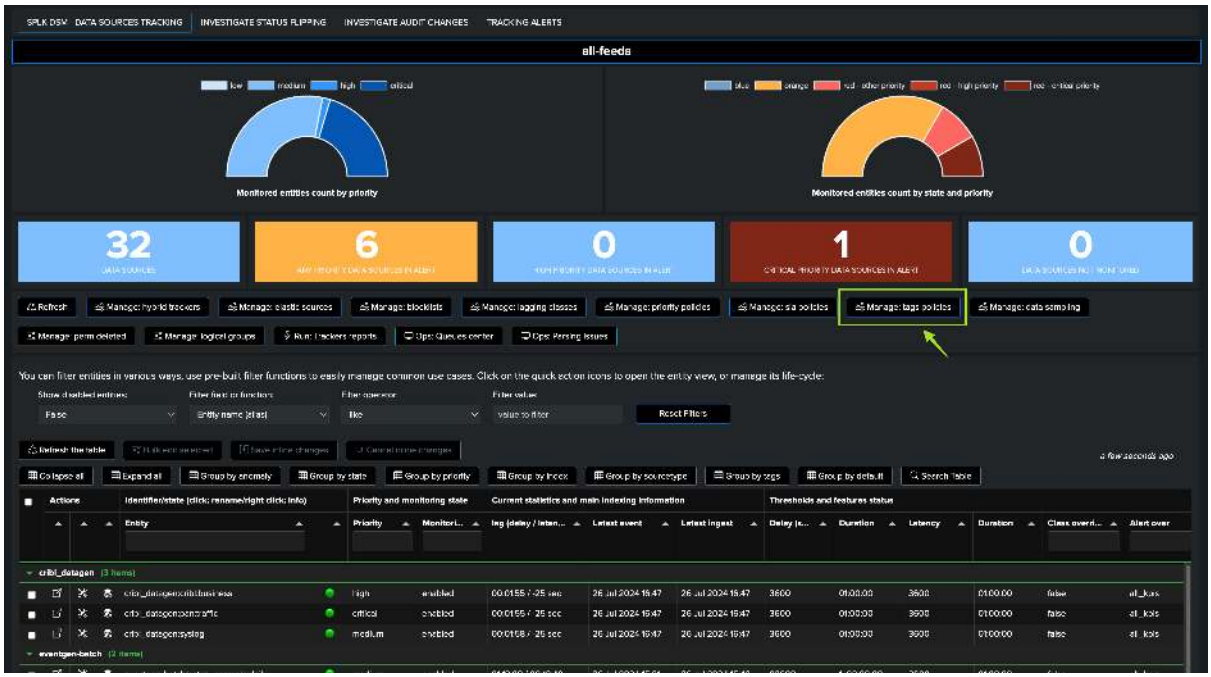
Accessing, defining and managing tags policies

TrackMe supports the management of tags via policy, which can be defined per Virtual Tenant:

- Tags policies allow you to automate tag assignment across entities using **regex-based**, **lookup-based**, or **search-based (SPL)** rules.
- **Regex-based policies** match regular expressions against the entity name (`object` field) and assign a defined set of tags.
- **Lookup-based policies** leverage existing Splunk lookups (CSV or KVstore) to assign tags based on field mappings, making it easy to integrate with a CMDB or asset inventory.
- **Search-based (SPL) policies** match against the live results of a Splunk search, mapped exactly like a lookup (same field mappings, time range, and match modes) — see *Priority — in depth* for the shared mechanics.
- **Advanced matching:** lookup and search policies also support **OR-combined match blocks**, matching the DHM `asset` multivalue field, and **wildcard** foreign-value matching (as for priority policies).
- **Multiple matching policies:** Multiple policies can match a given entity. Tags from all matching policies (regex, lookup, and search) are **merged together** (union-based).
- **Lower case:** Tags are automatically stored in lower case format.
- **Tags submission:** These policies are monitored by a scheduled job named `trackme_<component_suffix>_tags_tracker_tenant_<tenant_id>`.
- **Removing tags policies:** When tags policies are removed, the tags associated with entities from the removed policies are removed from the entities automatically.
- **Tags in the KVstore records:** Tags are stored in the entities KVstore records, and can be used for alerting filtering purposes.
- **Tags in the user interface:** Tags are displayed in the user interface for the entities.

Accessing the policy management screen:

- From any Tenant Home page, click the **Manage: tags policies** button



Lookup-based policies

Lookup-based policies leverage existing Splunk lookup transforms (CSV files or KVstore collections) to assign tags to entities based on field mappings. This is the recommended approach when integrating with an external CMDB, asset inventory, or any structured data source.

Creating a lookup-based policy:

- Click **Create new policy** and select the **Lookup** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)
- Select a **Splunk lookup transform** from the dropdown — TrackMe lists all available lookup transforms in your Splunk environment
- Configure **field mappings** to map lookup fields to entity fields — for example, map the lookup field `index` to the entity field `data_index`, and the lookup field `host` to the entity field `object`
- Select the **tags field** — the field in the lookup that contains the tag values
- Optionally configure the **tags separator** — the character used to split multiple tags from a single lookup field value (default: `,`)
- Select the **match mode**: **Exact** (case-insensitive string matching) or **Wildcard** (supports `*` and `?` patterns)
- Click **Simulate** to preview which entities would be matched and what tags would be assigned

Lookup policy configuration details

- **Field mappings**: Define how lookup fields map to entity fields. All mapped fields must match for an entity to be selected. For example, if you map `index` → `data_index` and `host` → `object`, both conditions must be satisfied.
- **Tags field**: The column in the lookup containing the tag value(s) (e.g., a `tags` or `category` column).
- **Tags separator**: The character used to split multiple tags within a single lookup cell. For example, if a lookup cell contains `network,critical`, the default separator `,` will produce two tags: `network` and `critical`.
- **Match modes**: `Exact` performs case-insensitive string comparison. `Wildcard` supports `*` (matches any characters) and `?` (matches a single character) for flexible pattern matching.

Regex-based policies

Regex-based policies match a regular expression pattern against the entity name (`object` field). When the pattern matches, the configured tags are assigned to the entity.

Creating a regex-based policy:

- Click **Create new policy** and select the **Regex** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)
- Enter the **regex pattern** to match against entities
- Enter the **tags** to assign to matched entities (comma-separated)
- Click **Simulate** to preview which entities would be matched before saving

Tags policies usage example (regex)

In the following example, we have a number of entities which have incoming some patterns we can leverage for tags policies purposes:

- geo-locations: the entities naming convention includes the source location of these feeds

- technology information: we can also leverage some of these information to associate these entities with technology related context tags

Example of tags policies definition:

In this example, we are going to rely on the index naming convention to define tags, so we can automatically classify TrackMe entities.

In short, we are going to apply the simple following logic:

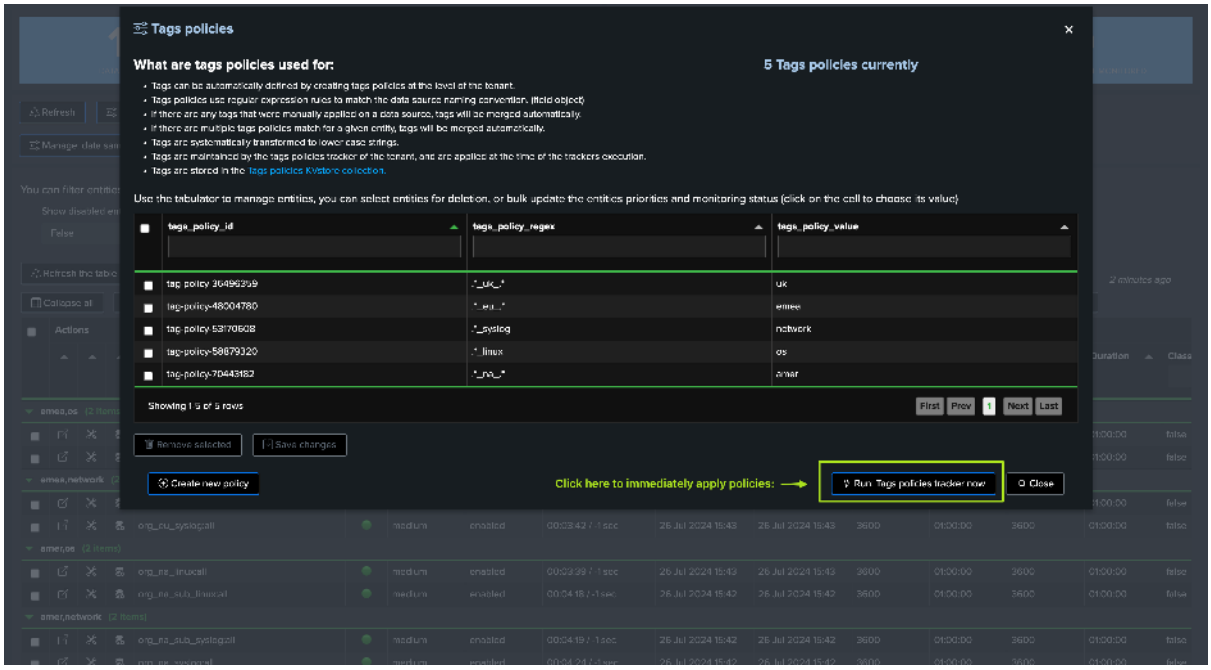
geographical location:

regular expression	tags
`.*_na_.*`	`north_america`
`.*_eu_.*`	`europe`
`.*_uk_.*`	`united_kingdom`

technology information:

regular expression	tags
`.*_syslog:.*`	`network`
`.*_linux:.*`	`os`

Once configured in TrackMe's UI:



Tag merging and conflict resolution

When multiple policies (regex, lookup, or a mix of both) match the same entity, TrackMe applies **union-based tag merging**: all tags from all matching policies are combined into the entity's tag list.

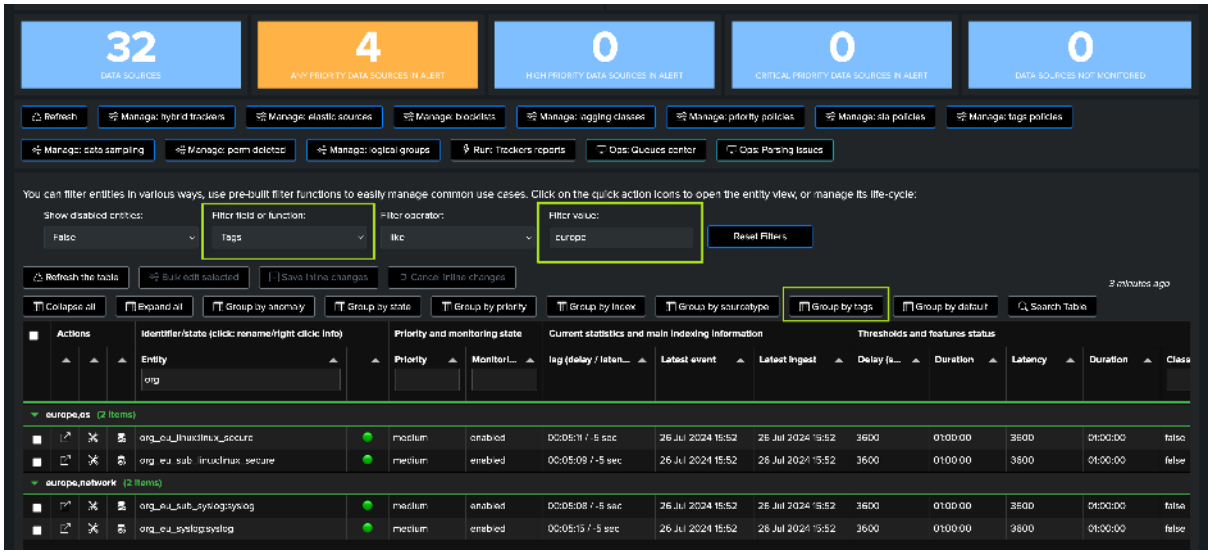
- Tags are always stored in lower case and deduplicated.
- There is no conflict resolution needed — tags from all matching policies are merged together.
- For example, if a regex policy assigns `network` and a lookup policy assigns `critical,production` to the same entity, the entity will receive all three tags: `critical`, `network`, `production`.

Running the policy tracker

After creating one or more policies, you can execute the policy tracker to apply all configured policies to your entities:

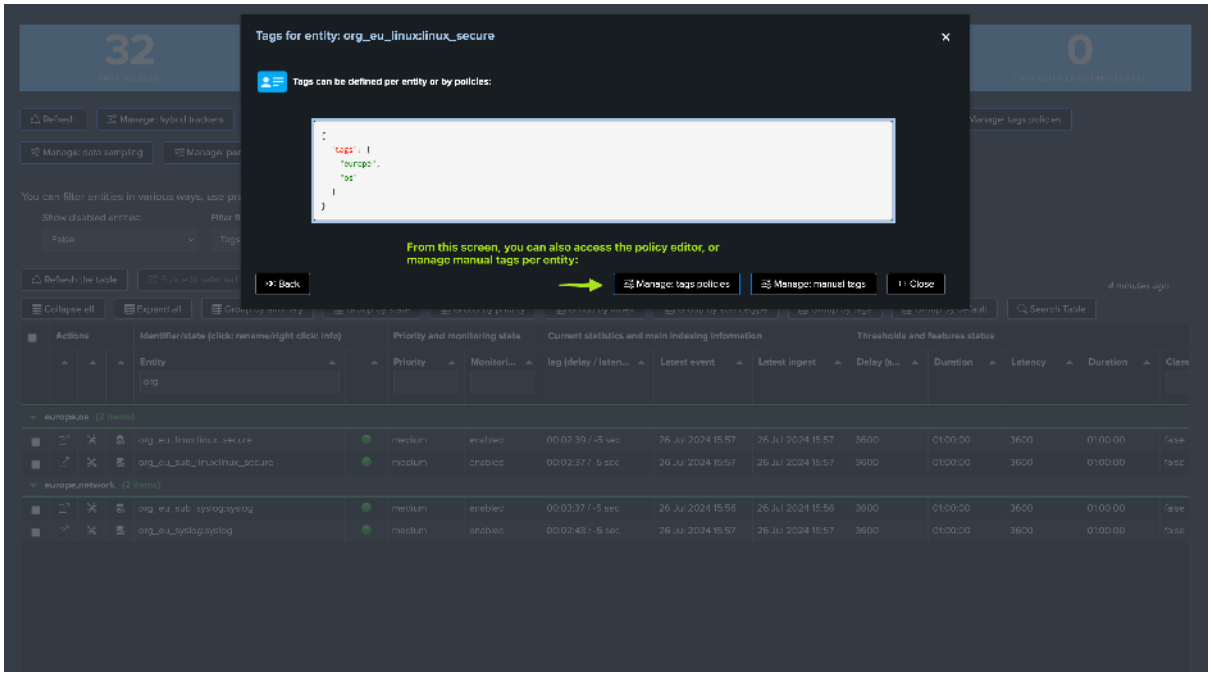
- Click the **Run policy tracker** button in the policy management screen
- TrackMe will evaluate all regex and lookup policies against all entities in the selected component
- A summary of the results is displayed, including the number of entities updated, matched, and any errors encountered

Tags can be used for filtering purposes in the user interface:



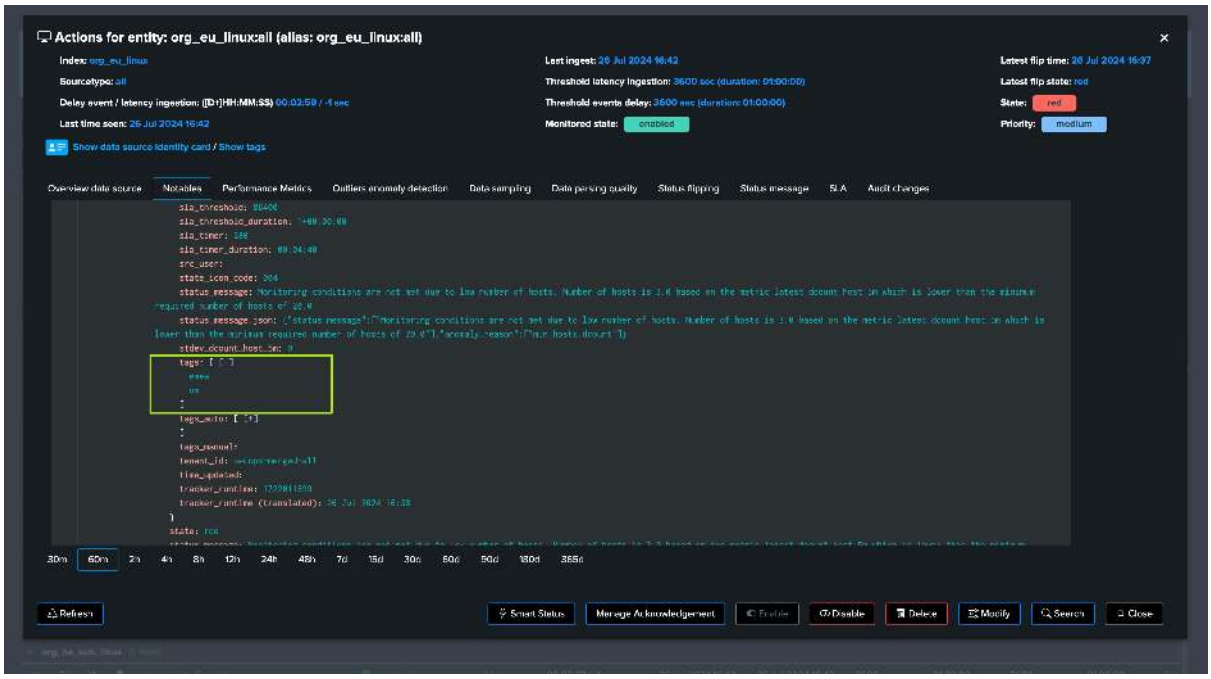
Tags can also be accessed per entity, and modified from here too:





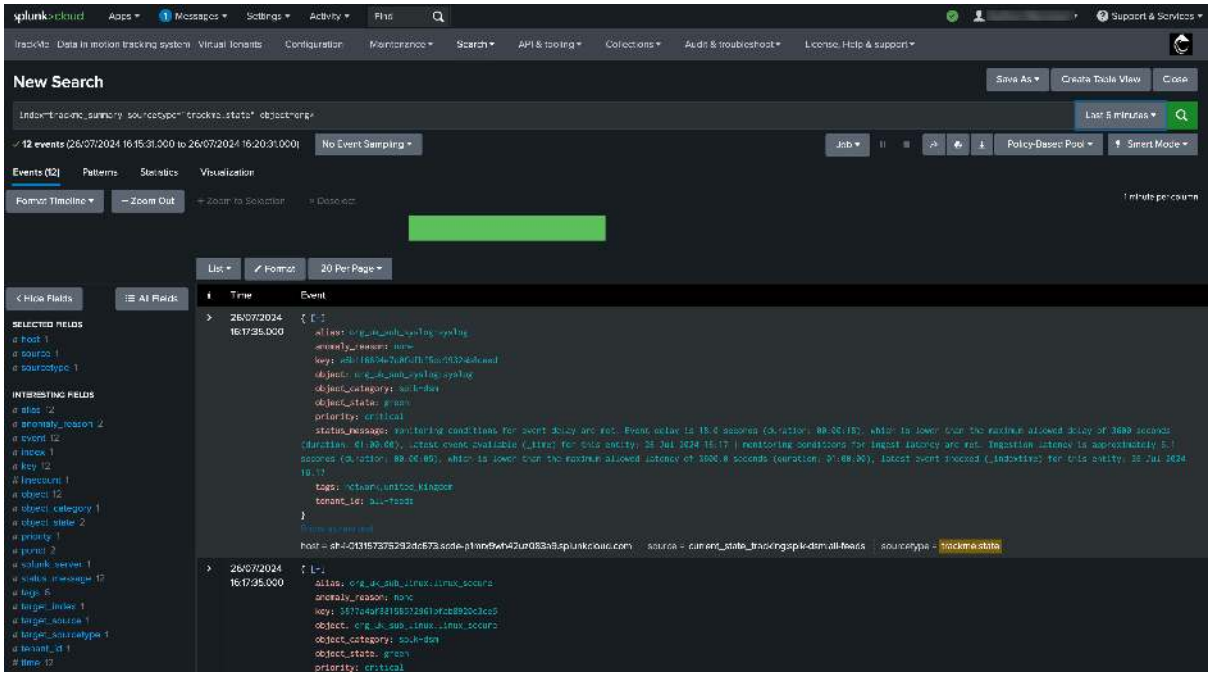
Tags are parts of TrackMe notable events, so you can use these for filtering, enrichment, decision making and so forth:

```
index=trackme_notable tenant_id=<replace with tenant_id>
```



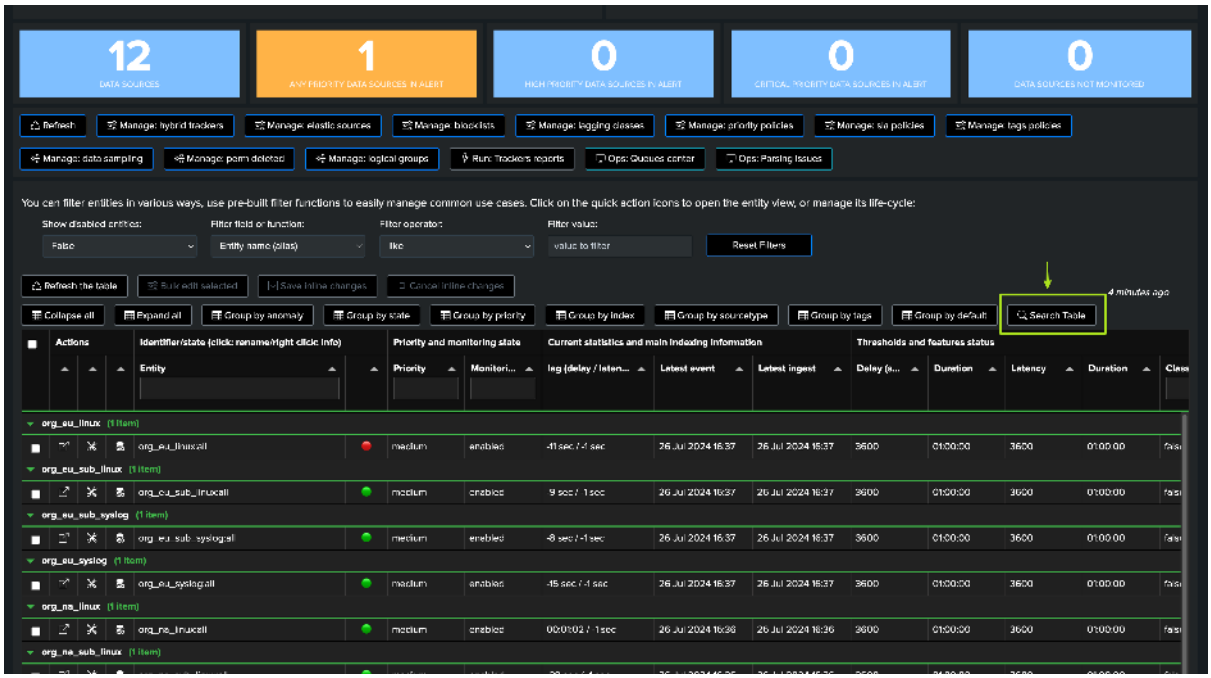
Tags are included in the trackme:state events:

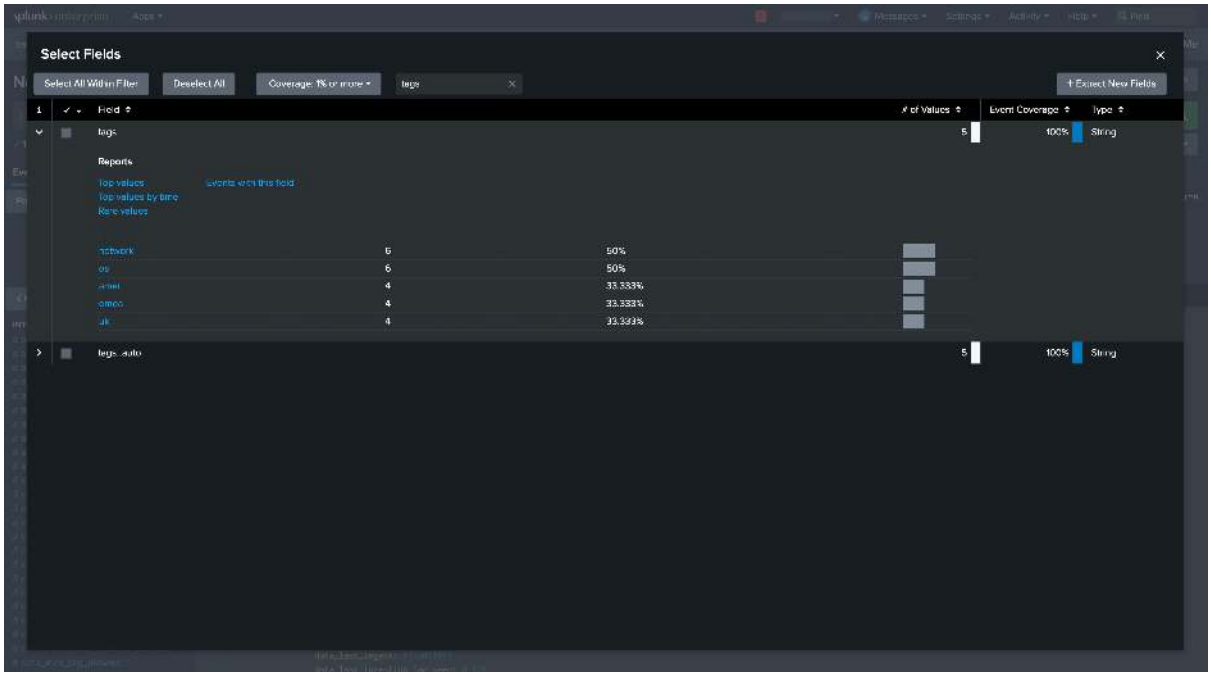
```
index=trackme_summary sourcetype="trackme:state" tenant_id=<replace with tenant_id>
```



Tags can be accessed through the TrackMe get collection command, which renders entities from Decision Maker (same as the UI view):

```
| trackmegetcoll tenant_id=<tenant_id> component=<component_suffix>
```



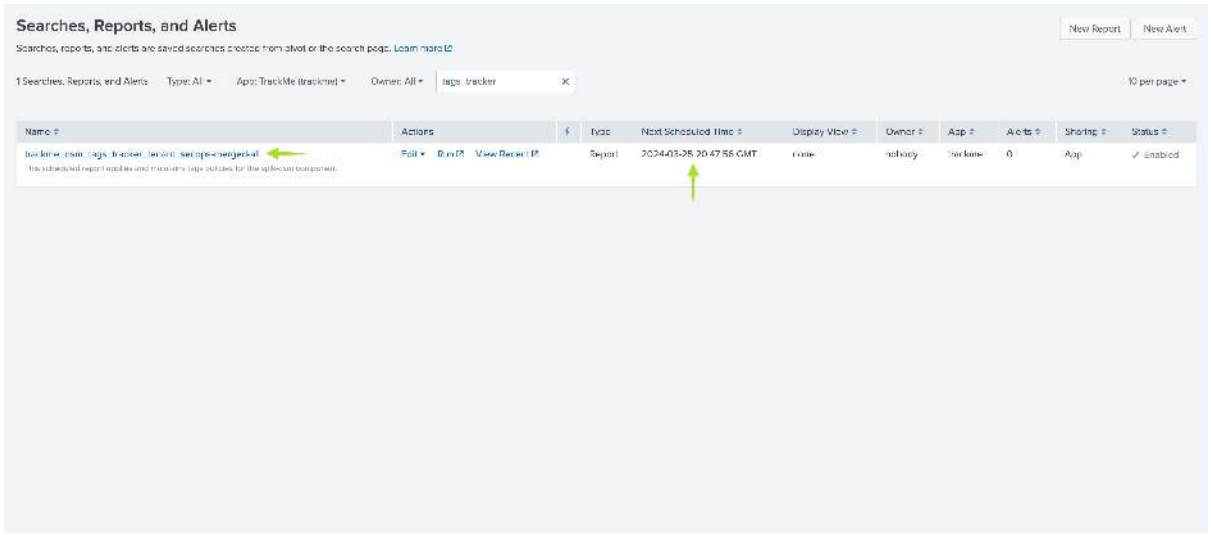


Troubleshoot tags policies

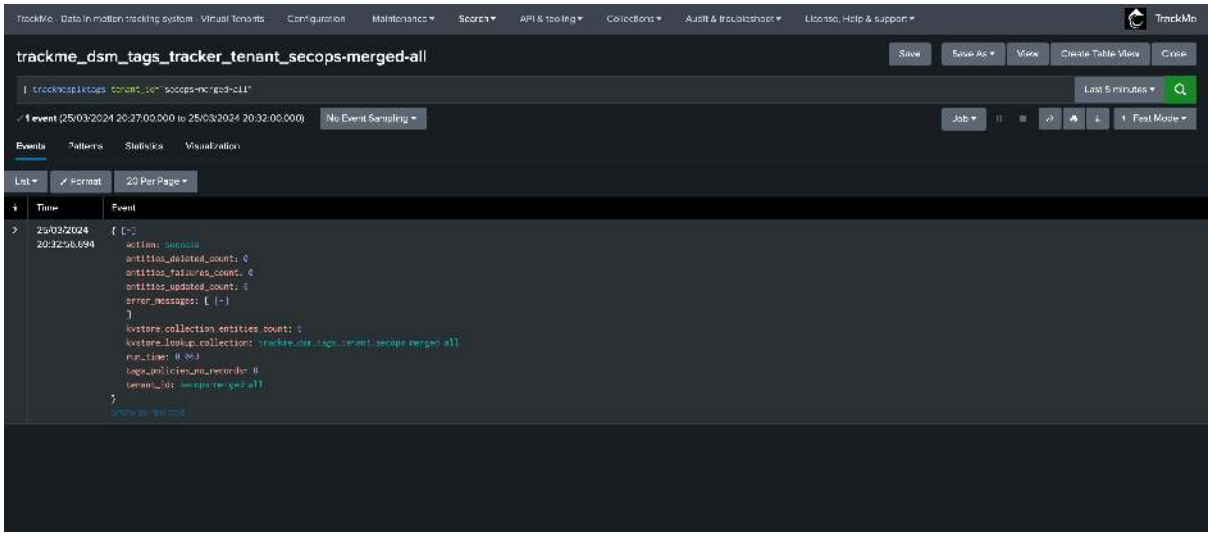
Tags policies tracker

The first component is the scheduled job which applies the tags policies to the entities, this job is called as follows:

- trackme_<component_suffix>_tags_tracker_tenant_<tenant_id>



This jobs should be scheduled, and properly executed:



You can find its logs as follows:

```
index=_internal sourcetype=trackme:custom_commands:trackmesplktags
```

When you have multiple tenants, you may want to filter on the tenant identifier:

```
index=_internal sourcetype=trackme:custom_commands:trackmesplktags tenant_id=
↪<replace with tenant_id>
```

In case of any issues with the tags policies, ensure that the job exists, is scheduled and executes properly and continue with the following steps.

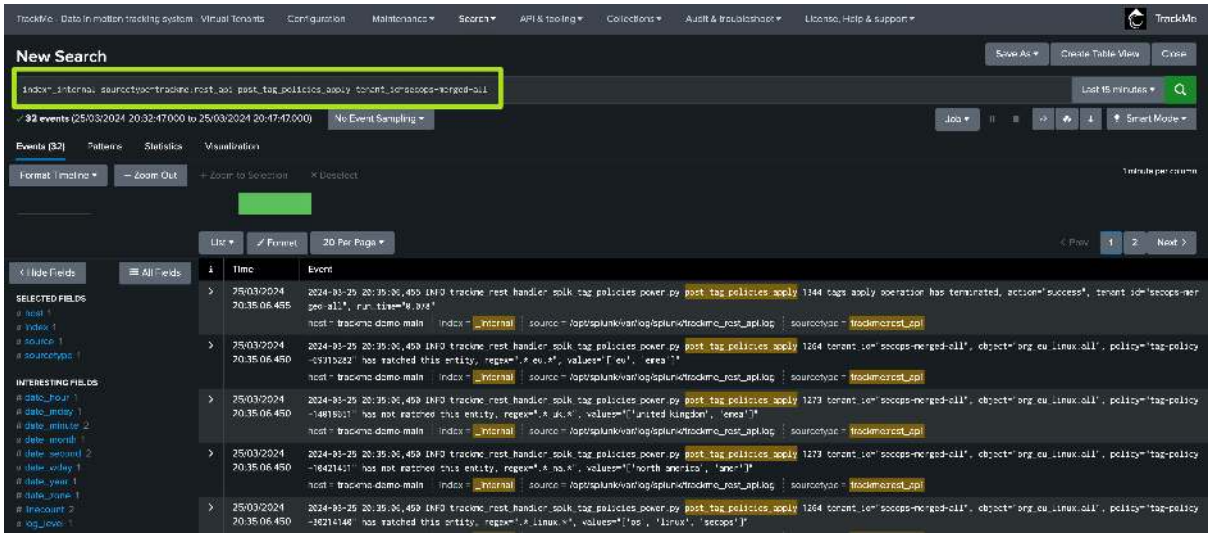
Tags policies REST API endpoint

The scheduled job calls the REST API endpoint to apply the tags policies, you can review the logs for this endpoint as follows:

```
index=_internal sourcetype=trackme:rest_api post_tag_policies_apply
```

With multiple tenants, filter on the tenant identifier:

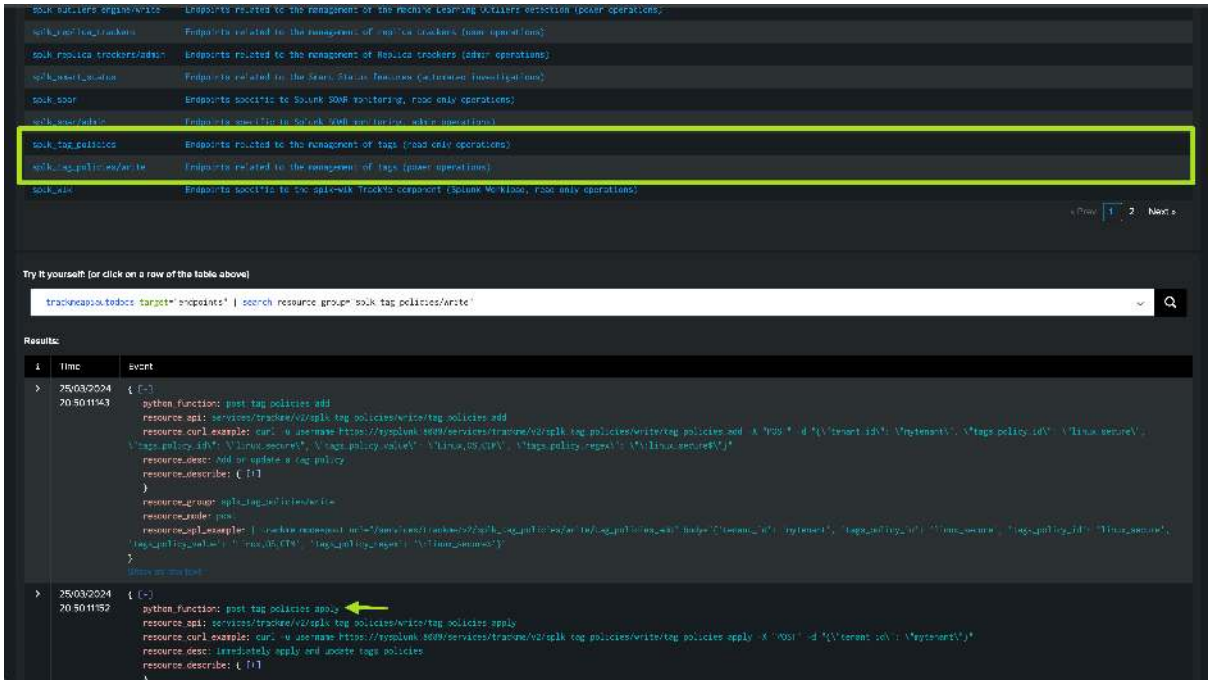
```
index=_internal sourcetype=trackme:rest_api post_tag_policies_apply tenant_id=
↪<replace with tenant_id>
```



In case of issues, carefully review the logs and verify if there are exceptions or issues with the tags policies:

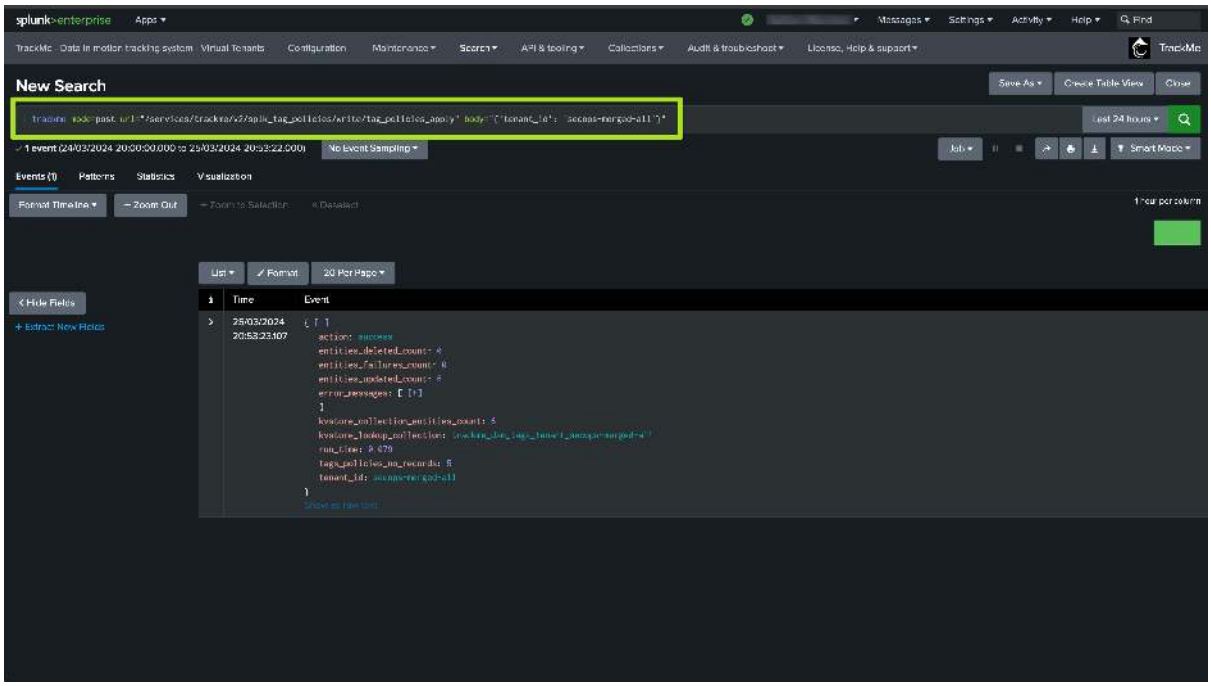
```
index=_internal sourcetype=trackme:rest_api post_tag_policies_apply log_level=error
```

The REST API endpoint can be applied manually as well, you can easily get the list of endpoints and their usage, review the REST API reference dashboard:



In fact, you can apply the tags policies manually:

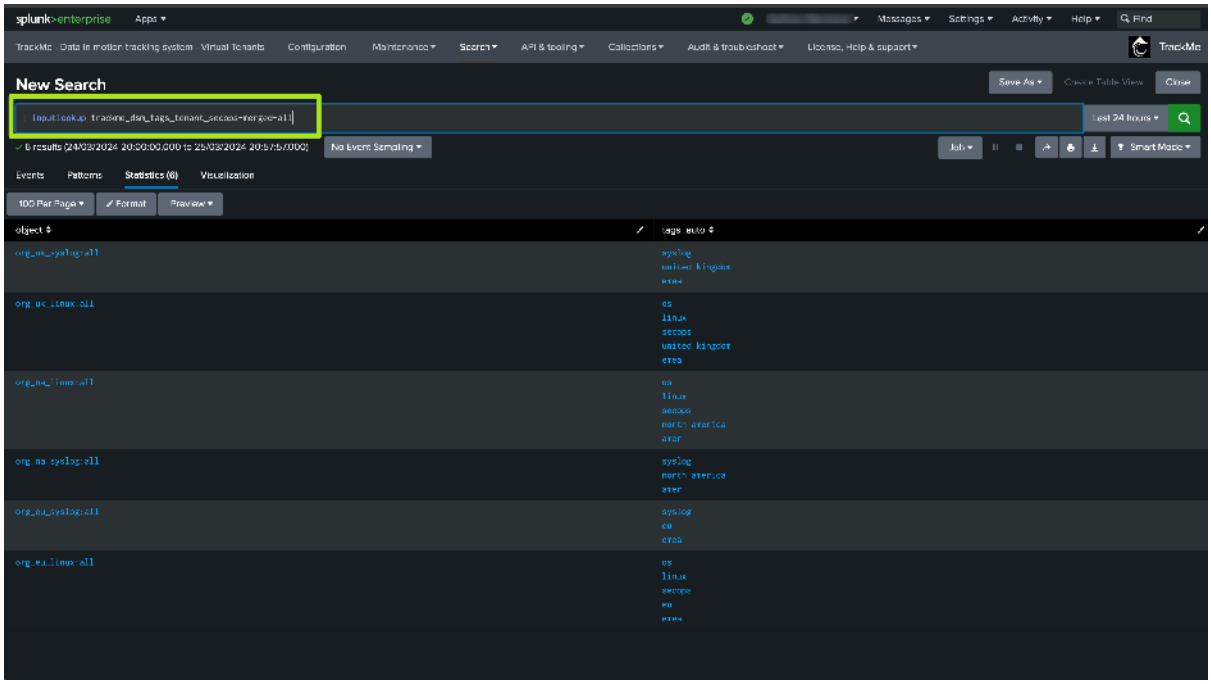
```
| trackme mode=post url="/
↪services/trackme/v2/splk_tag_policies/write/tag_policies_apply" body="{'tenant_id
↪': '<replace with the tenant id>'}"
```



Tags policies KVstore collection

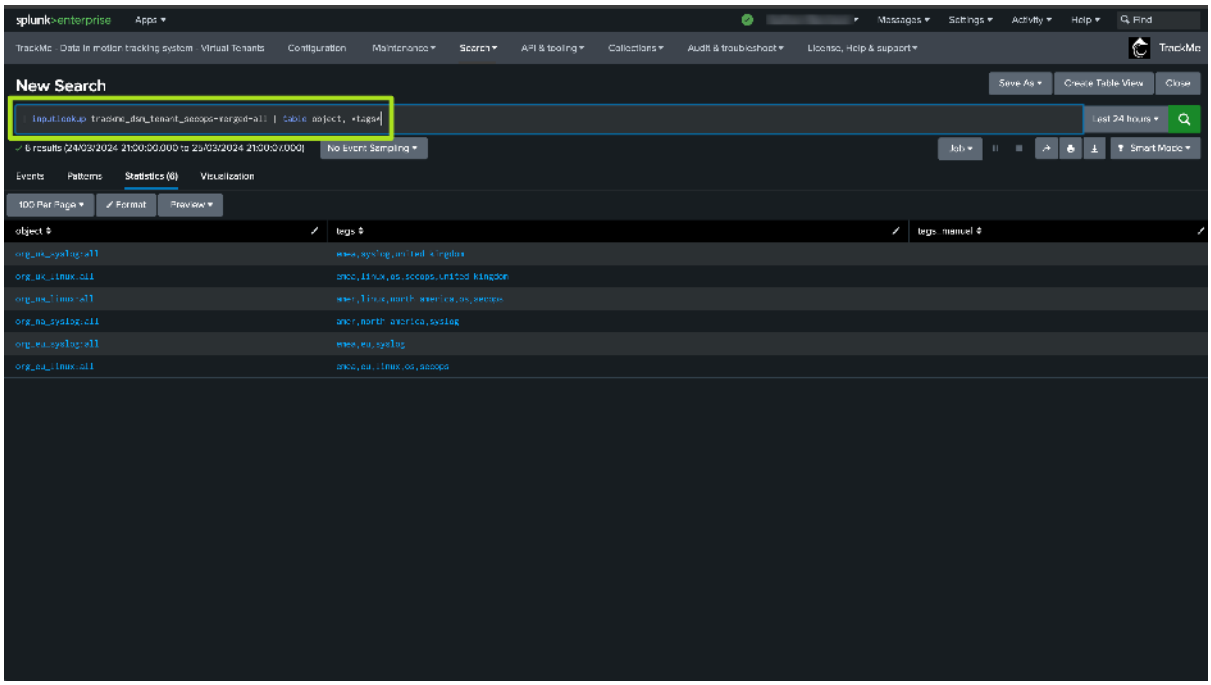
When tags policies are applied, TrackMe entities get a record with the list of associated tags, if any:

```
| inputlookup trackme_dsm_tags_tenant_<replace with tenant id>
```



When TrackMe trackers are executed, tags records are automatically crossed enriched with the entities records to form the final entities records tags fields:

```
| inputlookup trackme_dsm_tenant_<replace with tenant id> | table object, *tags*
```



8.5.5 Labels & auto-labels

Labels are curated, colour-coded chips drawn from a per-tenant catalogue — a lightweight **triage and workflow layer** over your entities. Where tags are free-form strings, labels are a small, deliberate vocabulary (`blocked`, `under-review`, `in-progress`, `resolved` …) that you and your automation use to track *what is being done* about an entity at a glance. You assign them by hand, or have TrackMe apply them automatically with auto-label rules.

Important

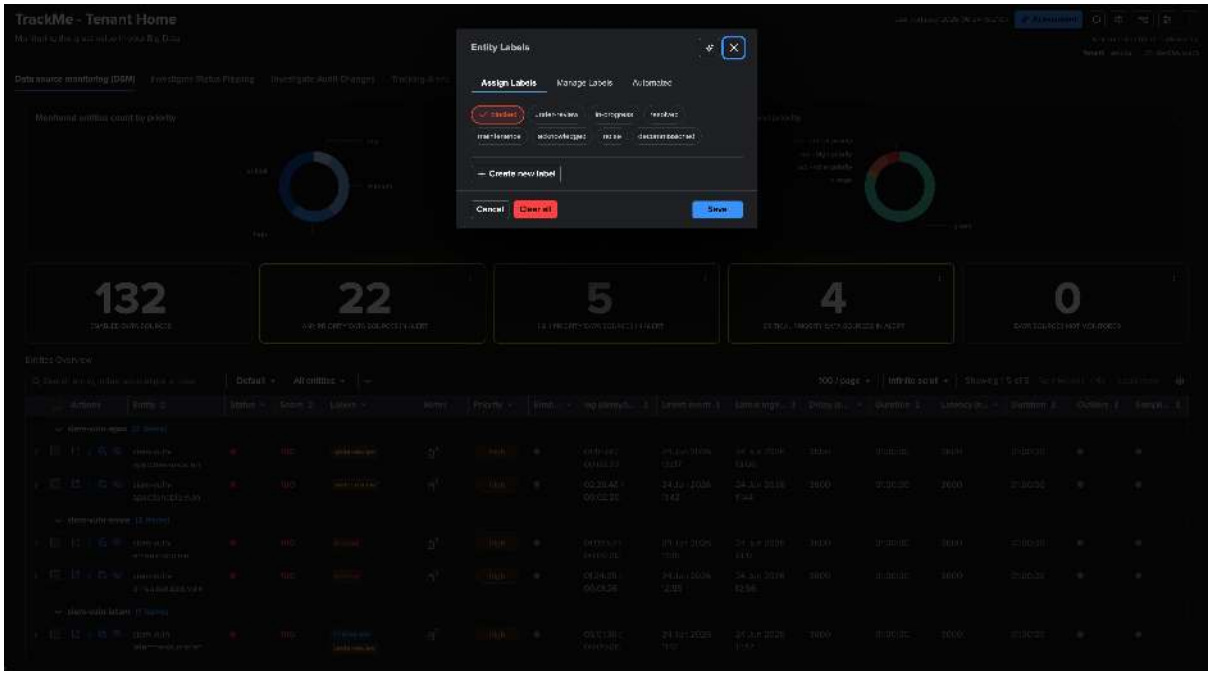
Labels are not tags. Labels come from a curated, colour-coded **catalogue**, are **cross-component**, and can be applied by rules. *Tags* are free-form strings, scoped per component. This is the single most common point of confusion in this part — if you asked for "a label" and got a tag, this is why.

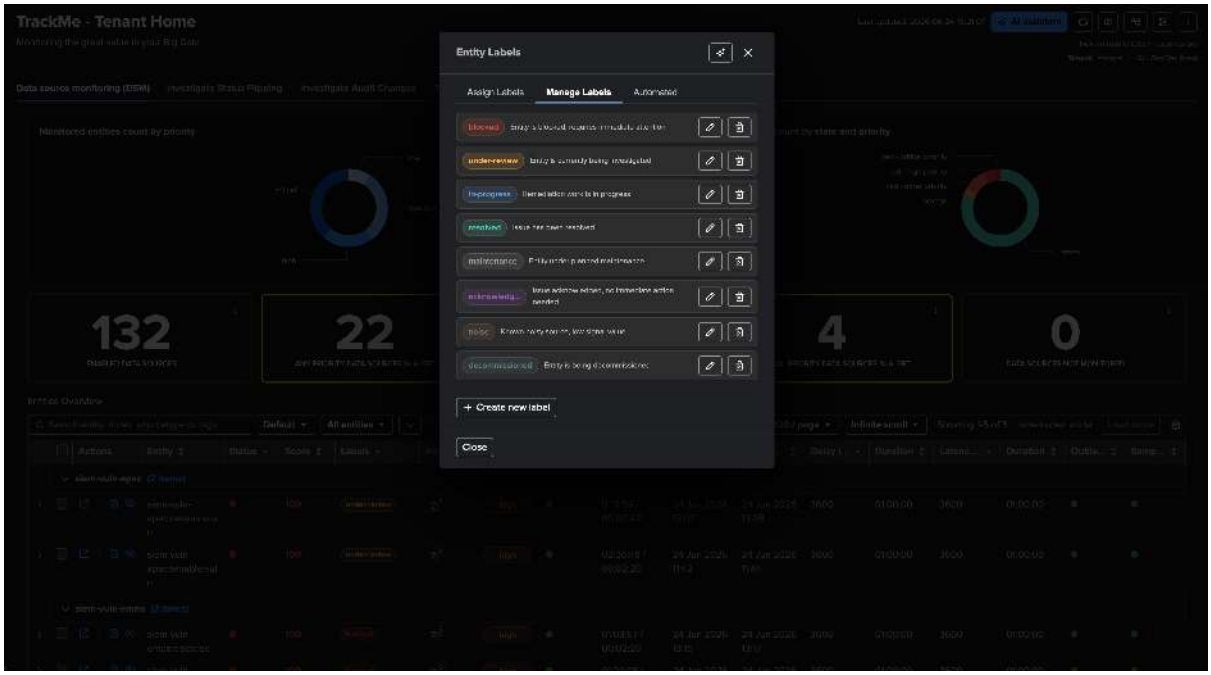
The catalogue, assignments, and rules

A tenant's labels have three pieces:

Piece	What it is
Catalogue	The curated set of label definitions for a tenant — each with a name, colour, description, and display order. You can only assign labels that exist in the catalogue.
Assignment	Attaching one or more catalogue labels to an entity. Labels are **per tenant but cross-component** — one catalogue, assignable to entities of any component. Every assignment records who changed it and when.
Rules (auto-labels)	Rules that assign labels automatically, evaluated by the scheduled decision maker (see below).

All three live in the **Manage Labels** modal — an *Assign* tab to label the entities you have selected, a *Manage* tab to curate the catalogue, and an *Automated* tab for the rules.





A tenant is seeded with **eight default labels** — `blocked`, `under-review`, `in-progress`, `resolved`, `maintenance`, `acknowledged`, `noise`, `decommissioned` — which already form a ready-made triage workflow. All are fully editable and deletable.

Warning

Deleting a catalogue label is destructive — it removes that label from *every* entity assignment, across all components, with no soft-delete. Audit before bulk deletions.

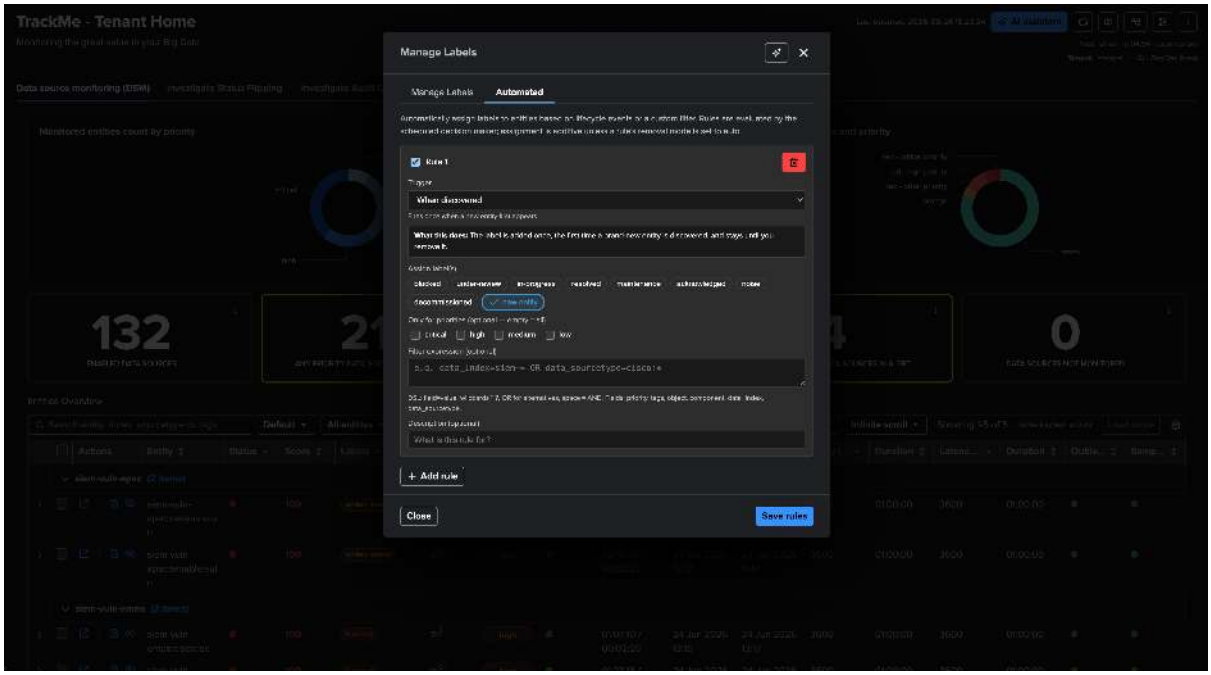
Automatic label assignment

Auto-label rules (the **Automated** tab) apply labels for you, evaluated by the scheduled decision maker. Each rule fires on one of four triggers:

Trigger	Fires when...
discovered	The entity first appears (once per entity).
enters_alert	The entity transitions green → red/orange.
recovers	The entity transitions red/orange → green.
custom_filter	The entity matches a filter expression (once per entity).

A rule can be further gated by a priority filter and/or a filter expression. Each rule has a **removal mode**:

- **manual** — additive: the label is added and then left alone. Non-edge triggers fire **once per entity**, so a label you manually remove is not resurrected next cycle.
- **auto** — reconcile: the label is present while its condition holds and removed when it clears. Auto rules only ever touch the labels *they* own — never your manual labels or another rule's labels.



Note

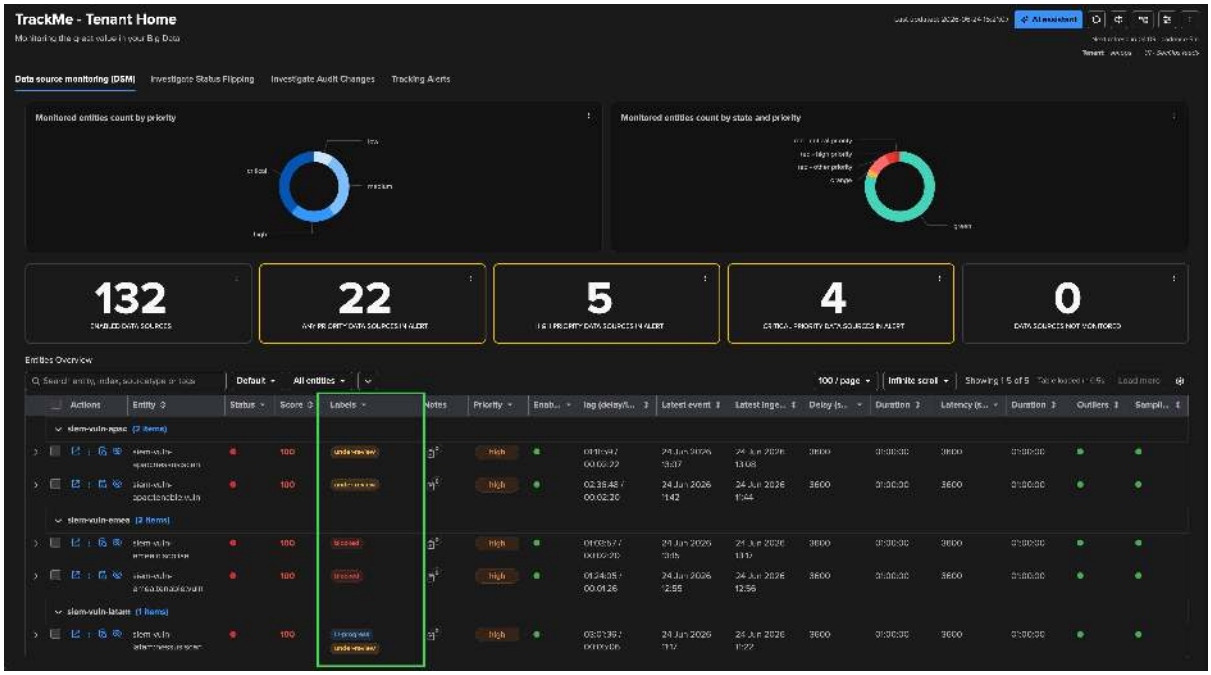
`recovers` + `auto` is a useful pattern: it latches a "recovered" badge on the recovery edge, holds it while the entity stays green, and retracts it when the entity re-enters alert. An always-green entity never gets the badge.

Auto-labels run only on the scheduled (write) path — loading the UI never changes assignments. The Automated tab is power/admin only.

Where labels are used

Like priority and tags, labels are **classification, not scoring** — they never change an entity's score or state. Their value is what you do *with* them:

Triage & filtering
: `labels` is a **column** on the entity table, a **Group by: Labels** pivot, and a first-class **filter axis** (`labels=blocked`, `labels IN(under-review, in-progress)`) — so an analyst can instantly pull up "everything blocked" or "everything someone is already working on".



A workflow vocabulary
: The default labels describe *what is being done* about an entity — `under-review` → `in-progress` → `resolved` for an investigation, `blocked` for something needing attention, `noise` for a known low-value source, `decommissioned` for one being retired. Apply them by hand, or let auto-labels apply them on the right transition.

Important

Labels are signals, not triggers. They are purely descriptive — no label has a functional side effect. In particular the `acknowledged` and `maintenance` labels do **not** perform acknowledgment or maintenance, and `noise` does **not** suppress alerting; those are separate systems (*Alerting*, *Operations & Administration*, and the blocklist). A label records a decision; it does not take an action.

Grouping & Virtual Groups
: A *virtual group* can scope its members with a filter expression that uses labels — for example a single card of everything currently `blocked` across tenants.

Automation scoping
: Labels are a field in the shared filter language, so you can scope **AI Advisor automated actions** and **ML outlier detection** by label — for instance, let autonomous tuning run only on entities that are *not* `under-review`. (Labels are used for *scoping* here; they are not an ML feature input.)

Bulk operations
: Select many entities and **bulk add or remove** labels across the whole selection — the fast way to move a batch of entities through a workflow step.

Alerting & notable events
: An entity's labels are **carried onto its** *notable events* as context, so your Enterprise Security or correlation searches can act on them. (Labels are *not* carried on stateful-alert records.)

See also

- *Tags* — the free-form chip feature, and how it differs.
- *The entity state machine* — the state transitions that drive auto-label triggers.
- *Notes* — free-form per-entity context.

8.5.6 SLA policies

An SLA policy assigns each entity an **SLA class**. A class sets the **maximum continuous time an entity may stay red** before it counts as an SLA breach — a second-level signal that a problem has gone unresolved for too long. SLA is the third policy-driven classification, alongside priority and tags.

SLA classes

An SLA class is a **duration threshold**, not an uptime percentage. The shipped classes are:

Class	Max continuous red	Notes
platinum	4 hours	The strictest — breaches soonest.
gold	1 day	
silver	2 days	The default class, the most lenient.

Important

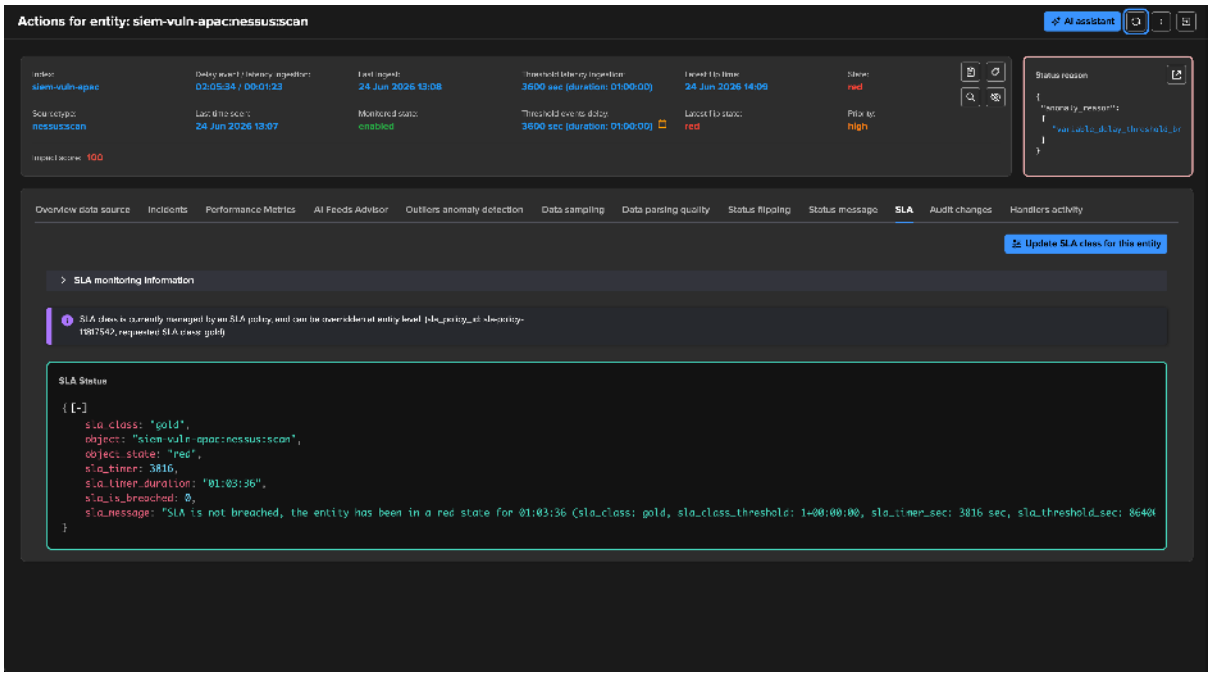
A *more premium* class is **stricter**, not more forgiving — platinum tolerates only four hours of continuous red, while silver tolerates two days. Classes are configured (added, edited, removed) on the **Configuration** page, under the SLA tab; the thresholds above are the shipped defaults.

How the SLA timer and breach work

Each entity carries an **SLA timer** — the number of seconds it has been **continuously** red. The timer starts when the entity flips to red and **resets** the moment it recovers (or is otherwise no longer red). When the timer exceeds the entity's class threshold, `sla_is_breached` becomes true:

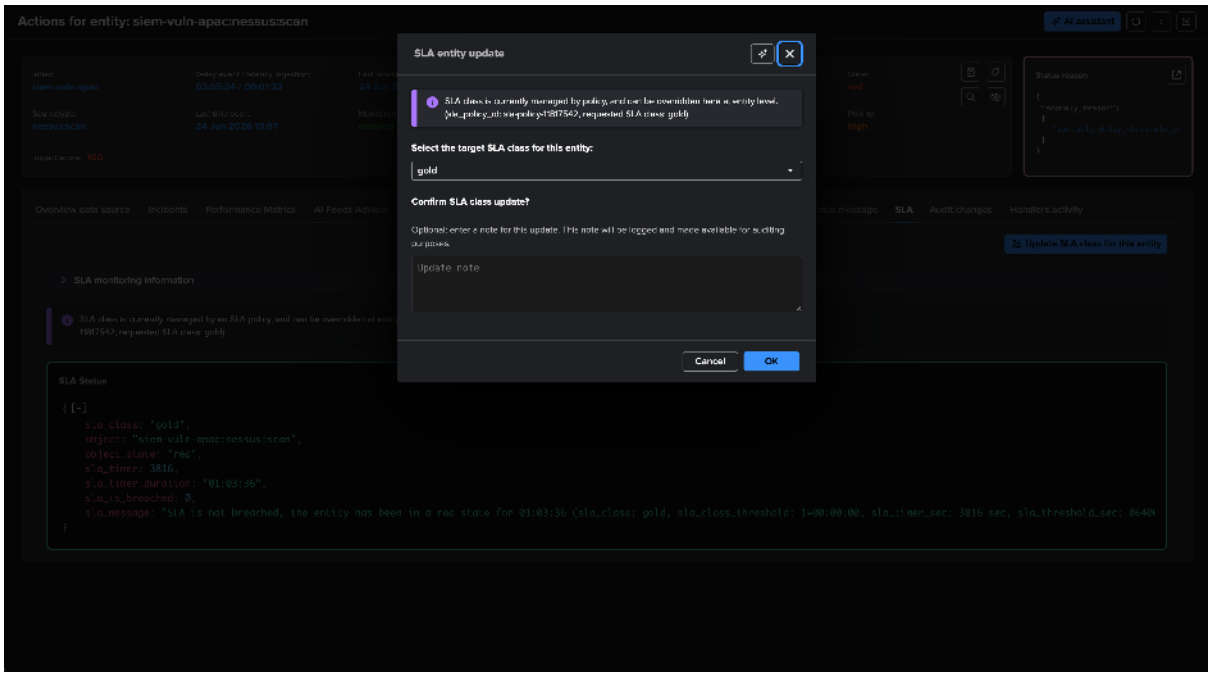
- `platinum` breaches after 4 hours continuously red;
- `gold` after 1 day; `silver` after 2 days.

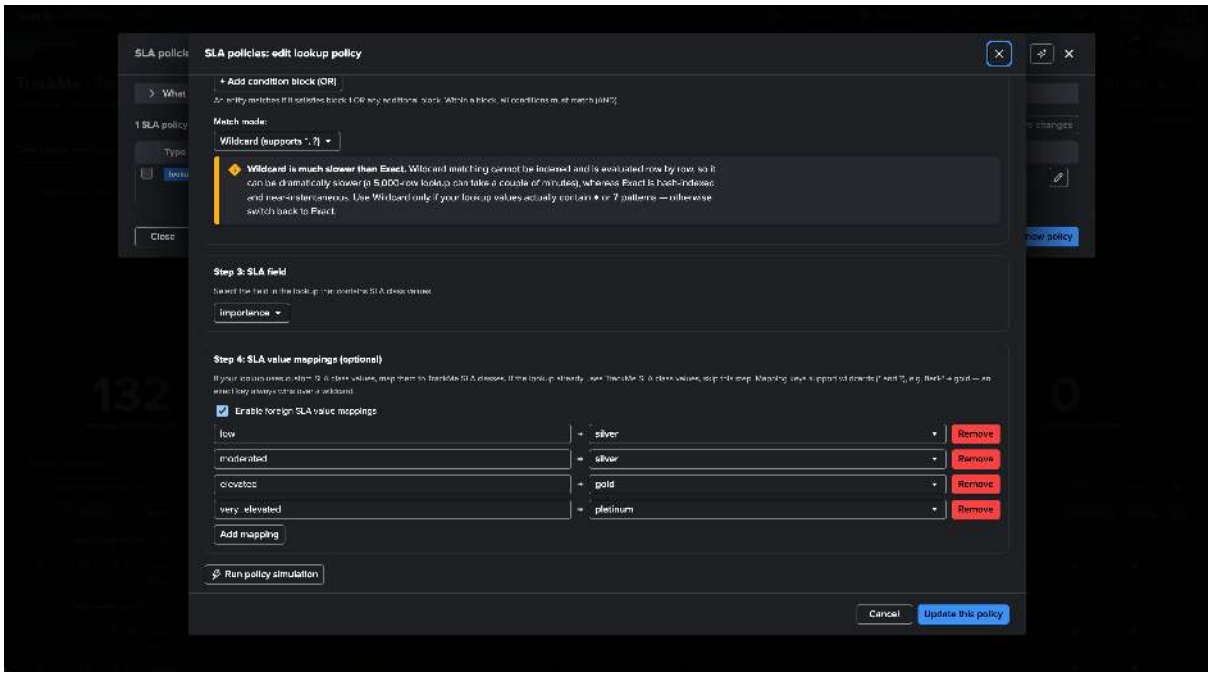
An entity that is green, blue, or suppressed is **not red**, so its timer is `0` and it is never in breach. You can see all of this on the entity's **SLA** tab:



Assigning an SLA class

- **Manually**, per entity — a manual class is flagged and not overridden by policies.
- **By policy** — SLA policies use the **same three methods** as priority and tags (regex, lookup, SPL), per component. A lookup or SPL policy can **map foreign values to TrackMe classes** — for example a CMDB column of `low` / `moderated` / `elevated` / `very_elevated` mapped to `silver` / `gold` / `platinum` (wildcards supported).



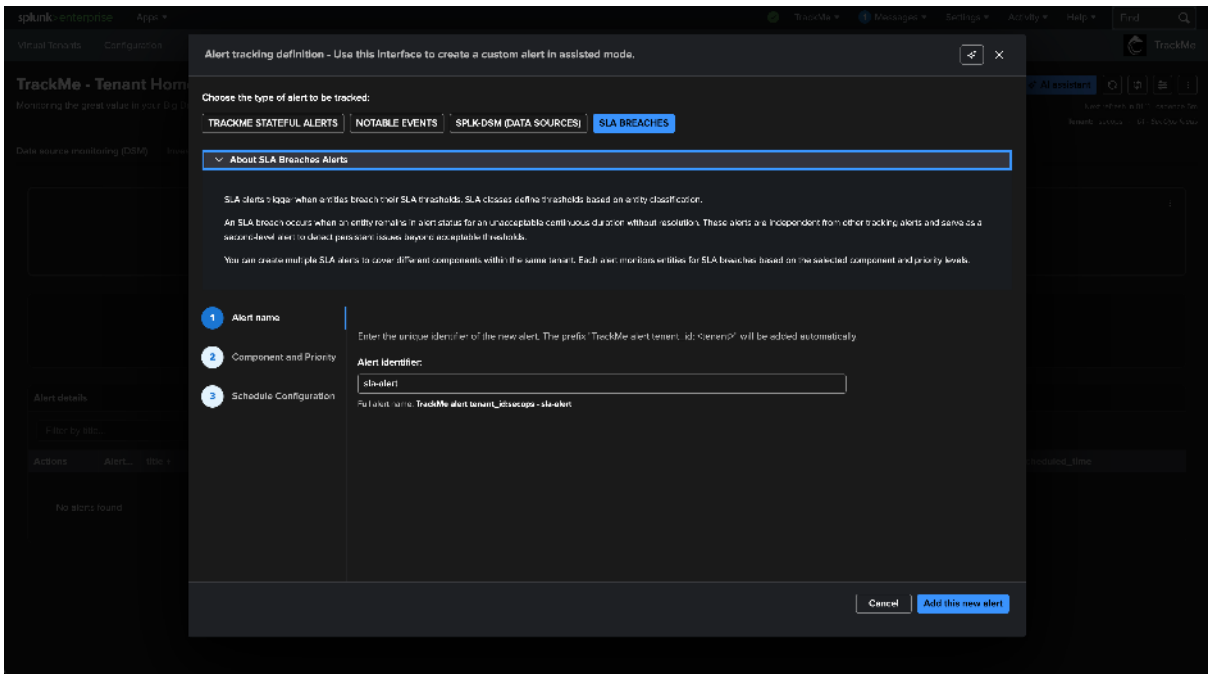


Note

Unlike priority, **SLA policies have no ranking** — when several match, the **last applied wins**. Keep SLA policies narrow and non-overlapping to avoid non-deterministic results.

SLA breach alerts

An SLA breach is a **second-level alert**. The entity's first-level *stateful alert* fires when it goes red; the SLA breach fires *separately* only once it has stayed red **beyond its class threshold** — catching the persistent problems that drift past an acceptable time to resolve. You enable it as its own **SLA breaches** alert type, and breach events are throttled per entity (weekly by default) so a long outage does not spam you.



SLA attainment reporting

Separately from the breach threshold, TrackMe reports each entity's **SLA attainment** — the **percentage of time it spent green** over a window. The window defaults to the **last 90 days** and is **configurable** (a global setting under **Configuration → SLA**), which is useful where a workload rule or RBAC restriction prevents searches from running over the full default window. A default SLA overview dashboard ships in the Tenant Home, and the figures are emitted as metrics so you can build your own attainment reports. (This percentage is the historical "how healthy has it been" view; the class threshold is the live "is it red for too long *right now*" view — two different things.)

Exclusions

Because the timer only runs while an entity is **red**, anything that takes it out of the red state stops it accruing SLA breach time:

- **Maintenance windows** and other suppression hold the entity in **blue**, so the timer is `0` (and, since recovering resets the timer, the continuous-red count restarts after maintenance).
- **Bank holidays** of the tenant's calendars hold new alerts, SLA breach alerts included, and are excluded from SLA percentages like planned maintenance (see *Bank holidays*).
- **Blue** (suppressed) entities are treated as **not in breach**.

Warning

If you remove an SLA class from configuration but entities are still assigned to it, those entities fall back to the **default** class (`silver`) — re-assign them explicitly if that is not what you want.

See also

- *Priority* and *Tags* — the sibling policy types.
- *Operations & Administration* — maintenance and bank-holiday exclusions.
- *SLA alerting use cases* — worked examples.

8.5.7 Logical groups

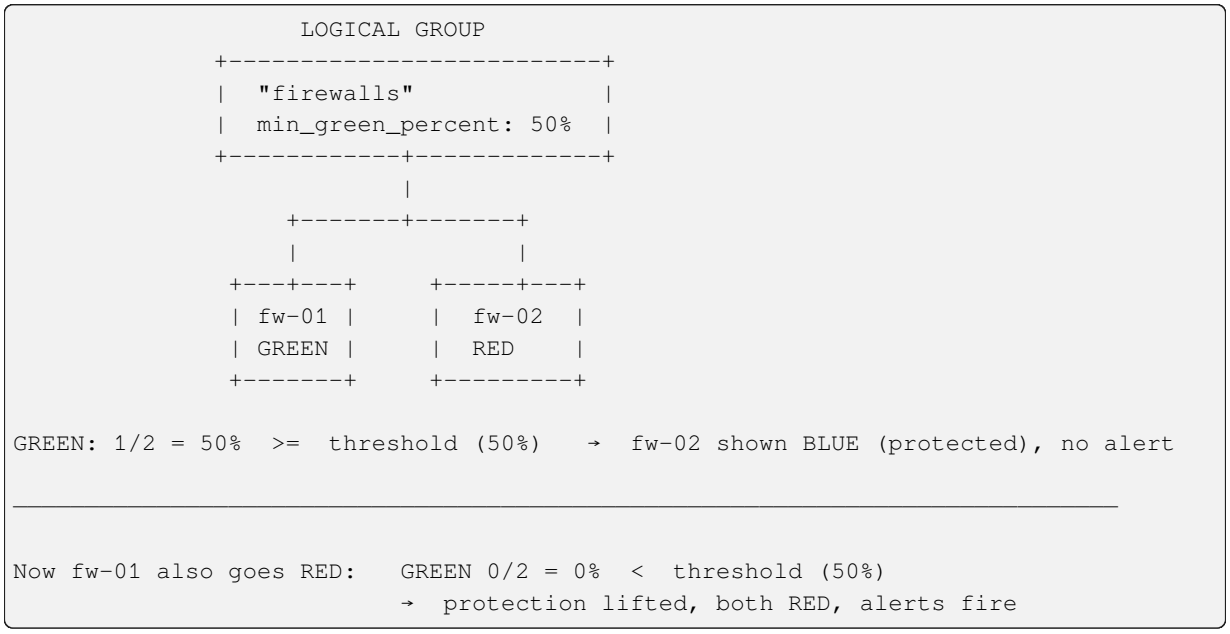
A **logical group** associates several related entities into an ensemble and judges them *collectively*. As long as enough of the group stays healthy, a member that would normally go **red** is instead shown **blue** — the anomaly is acknowledged but suppressed from alerting. It is the right tool for active/passive pairs, load-balanced pools, and anywhere a single failing member is expected rather than alarming.

Logical groups are available on every monitoring component except CIM — DSM, DHM, MHM, FLX, FQM, and WLK.

How logical groups work

A group defines a set of member entities and a **minimum green percentage** (`object_group_min_green_percent`). On every cycle TrackMe computes the group's green percentage (healthy members ÷ total members):

- if the green percentage is **at or above** the minimum, a member that would be **red** is demoted to **blue** — the healthy group carries it, and no alert fires;
- if the green percentage falls **below** the minimum, protection is lifted and red members alert normally.



This prevents alert storms for *expected* partial failures — active/passive failovers, rolling restarts, single-node maintenance in a pool.

Note

- Protection only applies to **red** members, and only when the group has **more than one member**.
- **If every member is red, there is no protection** (green % = 0) — by design.
- For **small groups** the threshold matters: a two-member group needs a threshold of 50% (one of two healthy) to protect; at a higher threshold neither is protected once one fails.

Logical-group protection is one of the **two sources of the blue state**; the other is the *disruption queue*. When a member is protected, its anomaly reason includes `in_logical_group` and its status message names the group and its health.

Group record

Each group is stored in a per-tenant collection (`kv_trackme_common_logical_group_tenant_<tenant_id>`) and carries:

Field	Meaning
`object_group_name`	The group's name.
`object_group_members`	Its member entities (multi-value).
`object_group_min_green_percent`	The minimum green percentage for the group to protect its members.
`object_group_members_green` / `object_group_members_red`	The current healthy / unhealthy member counts, maintained each cycle.
`object_group_mtime`	When the group was last modified.

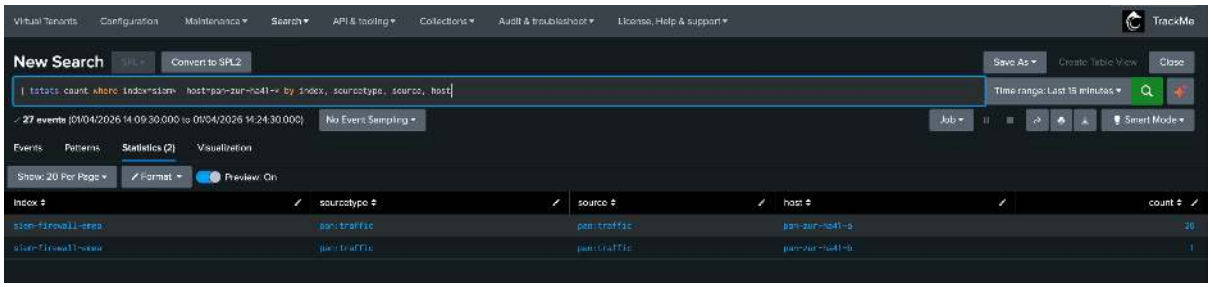
Use cases

- **Active/passive pairs** — firewalls, load balancers, or databases in HA pairs where only one member sends data at a time.
- **Rolling maintenance** — server pools where one member at a time is taken offline for patching.
- **Redundant data pipelines** — several entities carrying the same feed, tracked individually but assessed as a group.
- **Geographically distributed systems** — entities across regions where partial failure is expected and tolerable.

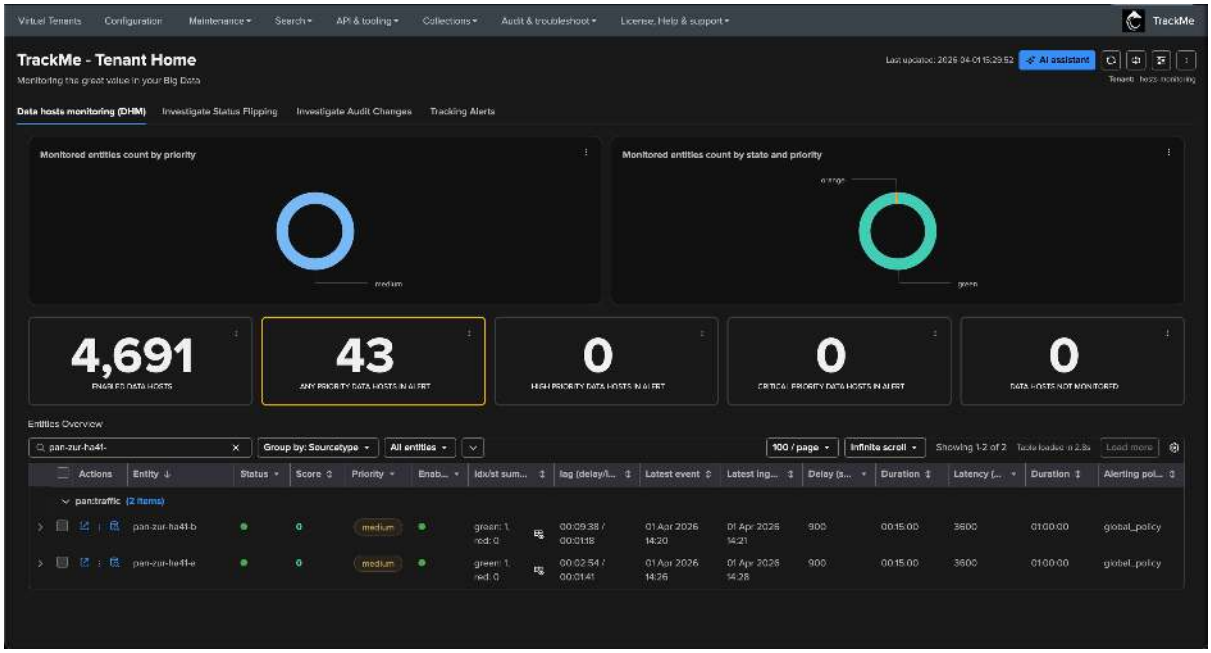
Walkthrough: pairing two firewalls

A concrete lifecycle, with a pair of firewalls (`pan-zur-ha41-a` / `pan-zur-ha41-b`) in active/passive mode.

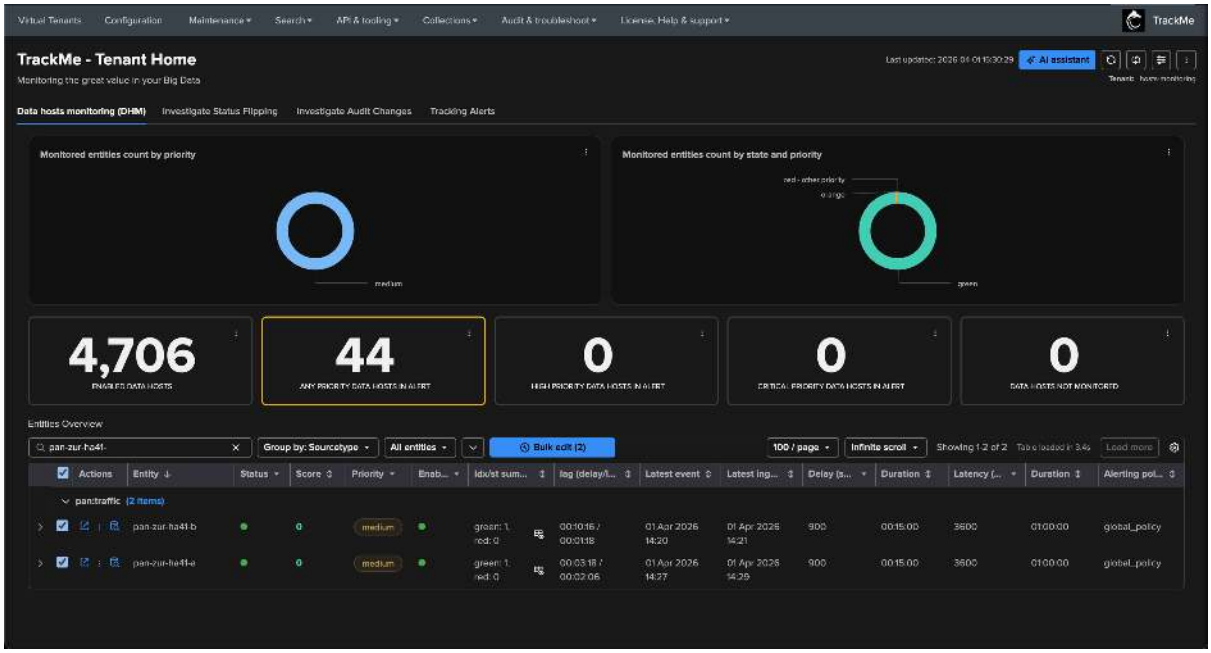
1. Identify the pair. In Splunk, find the hosts that form the pair — here they share an index, sourcetype, and naming convention.

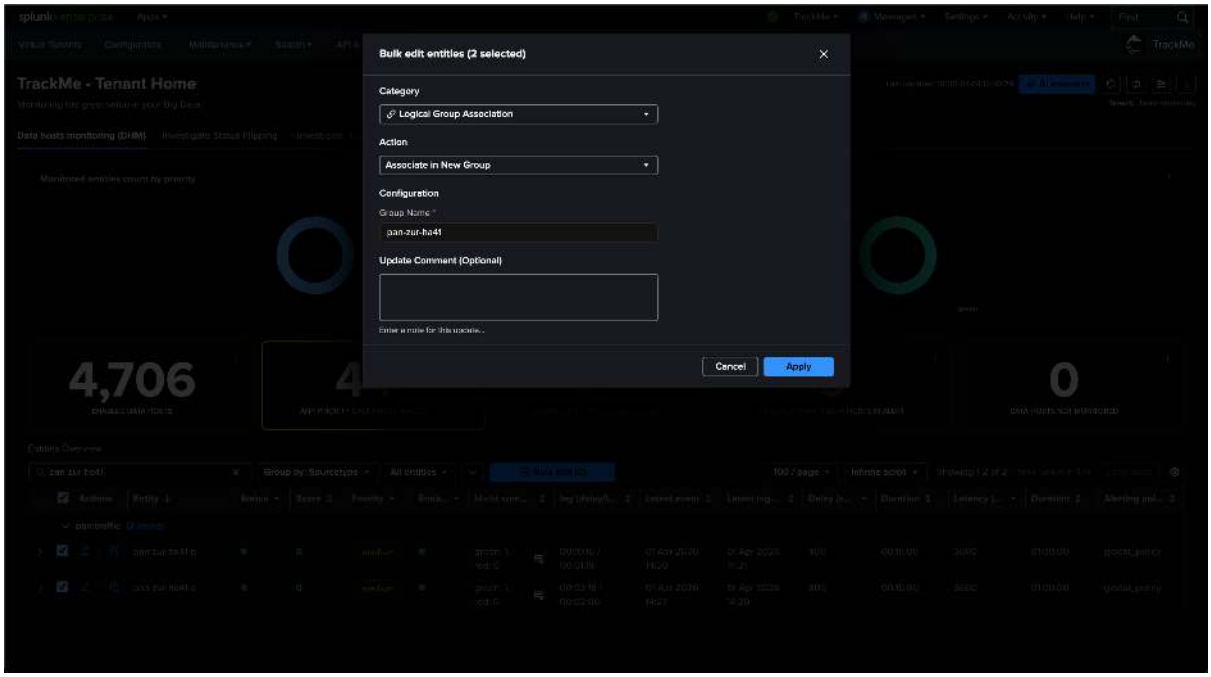


2. Review them in TrackMe. In the DHM Tenant Home, both hosts are green and healthy.

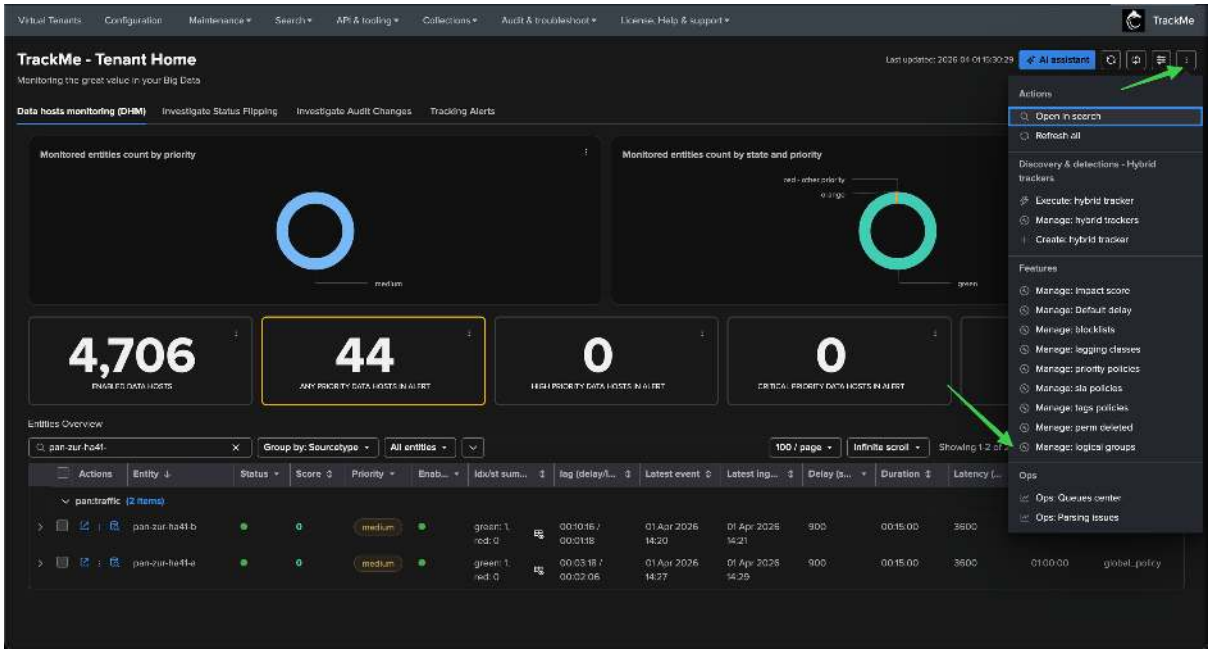


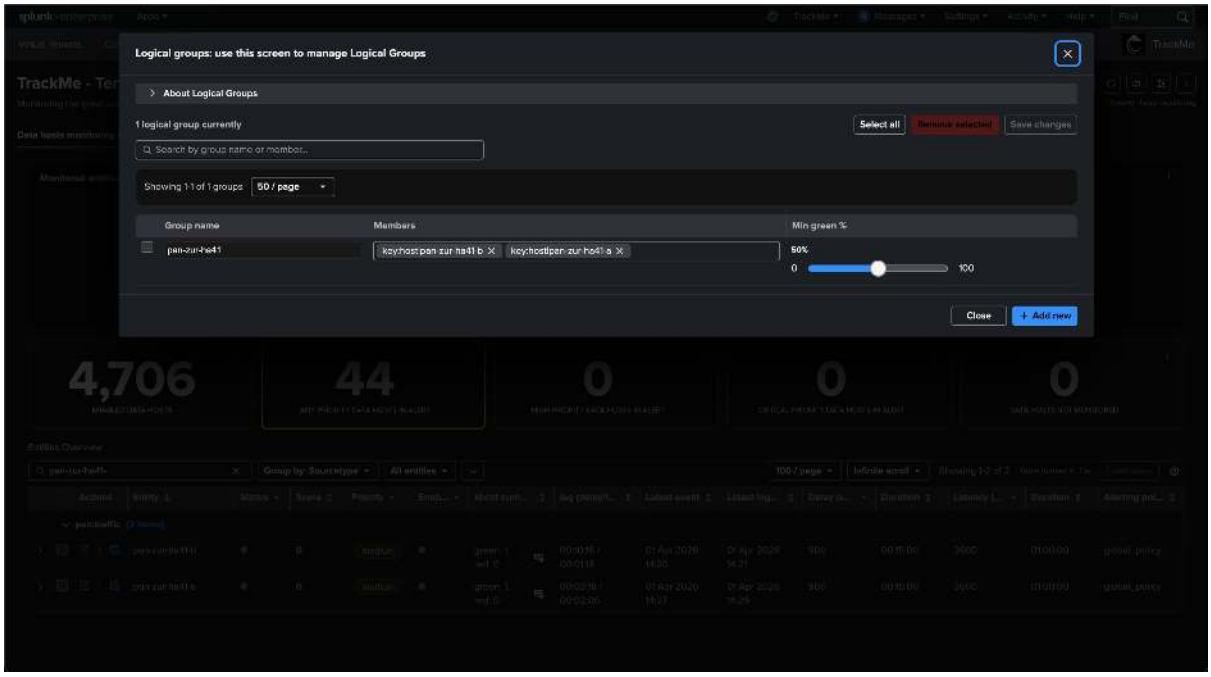
3. Associate them into a group. Select both entities, click **Bulk edit**, choose **Logical Group Association → Associate in New Group**, and name the group. TrackMe sets the `min_green_percent` automatically from the member count (50% for a pair).



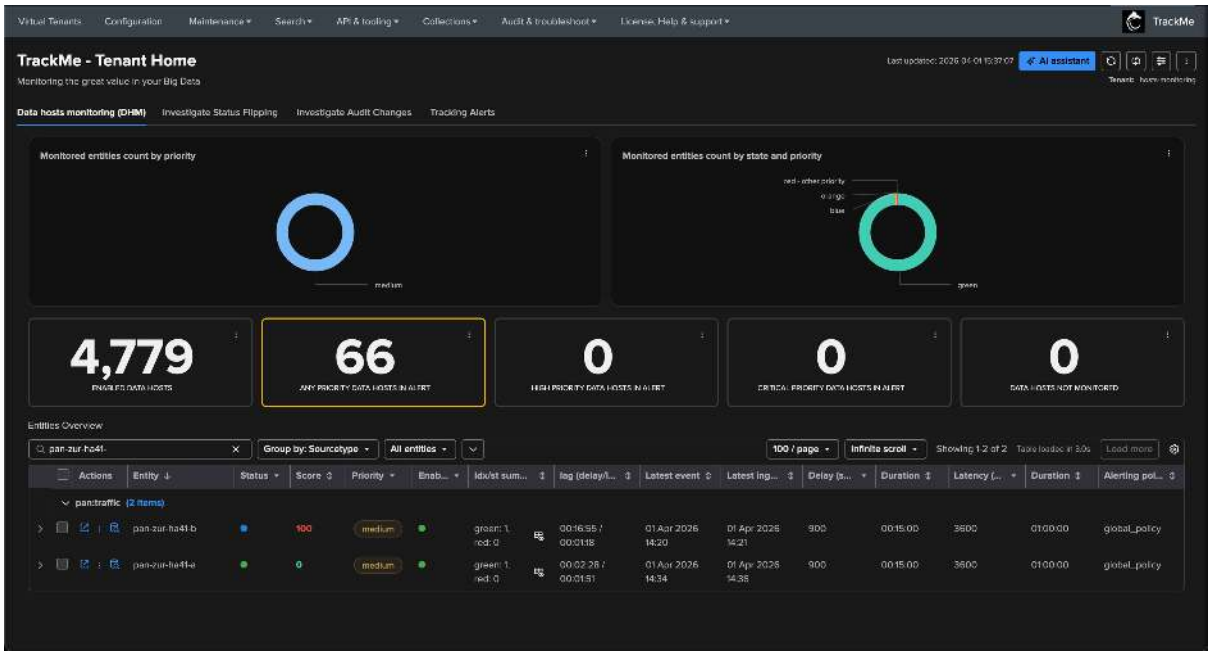


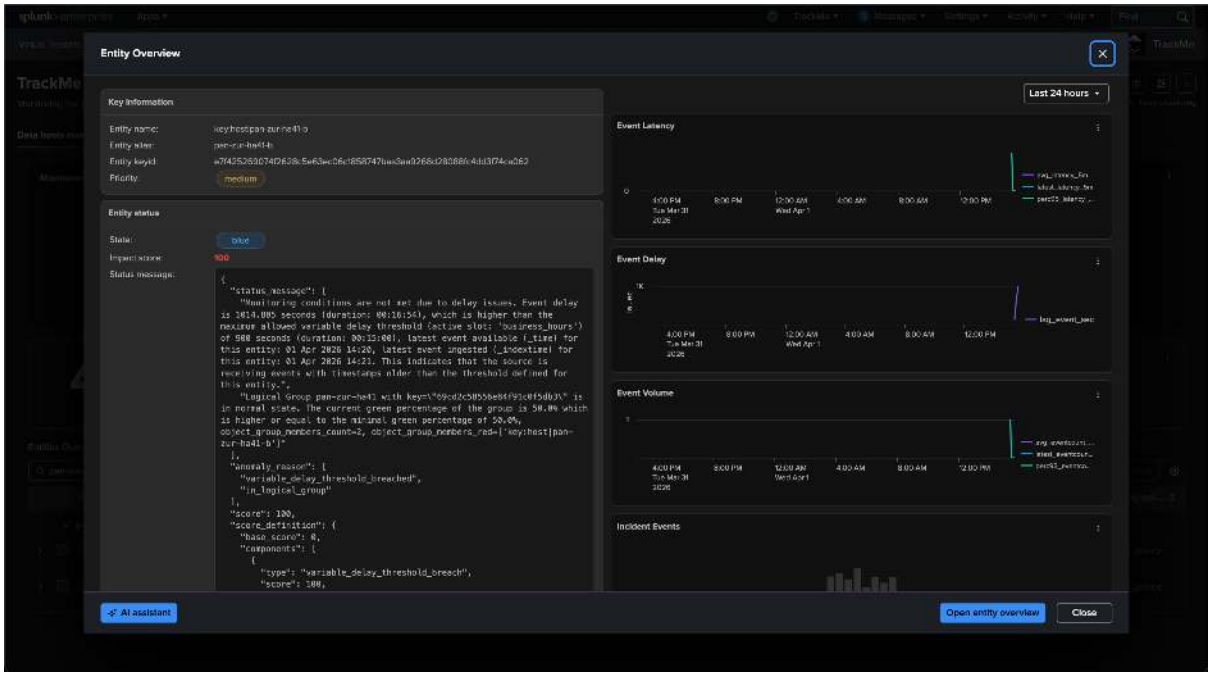
4. Verify in the Logical Groups manager. From the **Actions** menu choose **Manage: logical groups**. The manager lists the group, its members, and the minimum green percentage, which you can adjust with the slider.

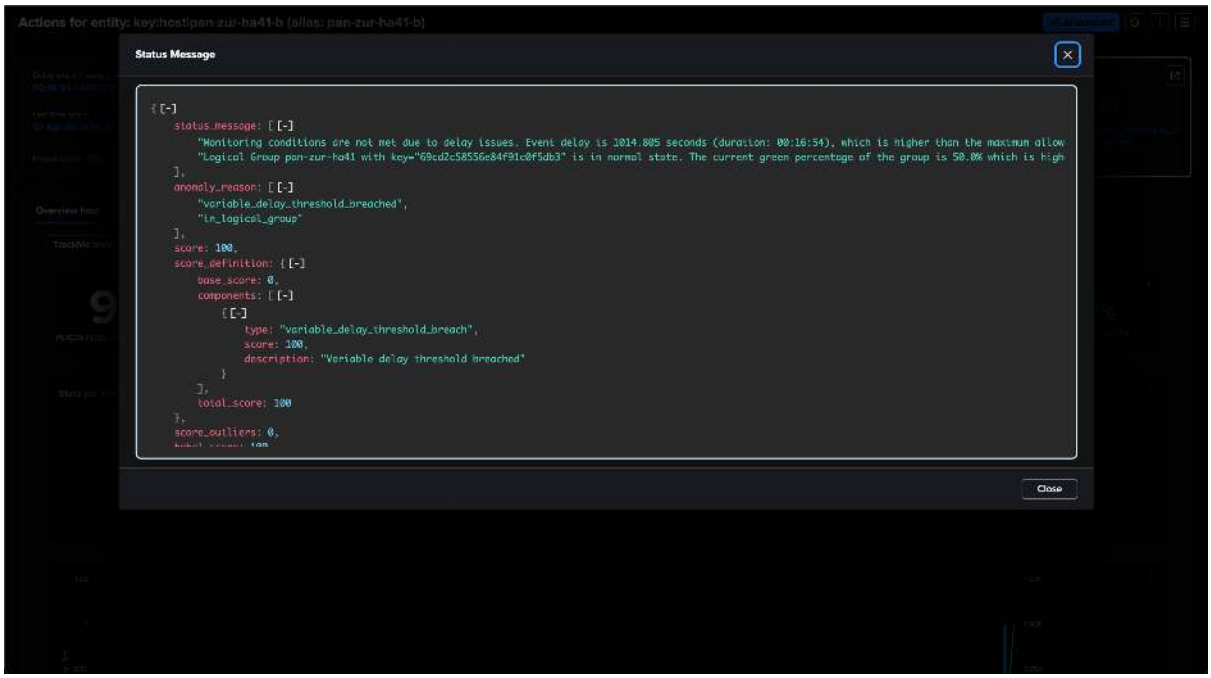




5. See the protection in action. When one firewall stops sending (an active/passive failover) it would normally go red — but because its partner is healthy and the group is above threshold, TrackMe shows it **blue**. The entity overview and status message confirm it is protected by its logical group (anomaly reason `in_logical_group`).



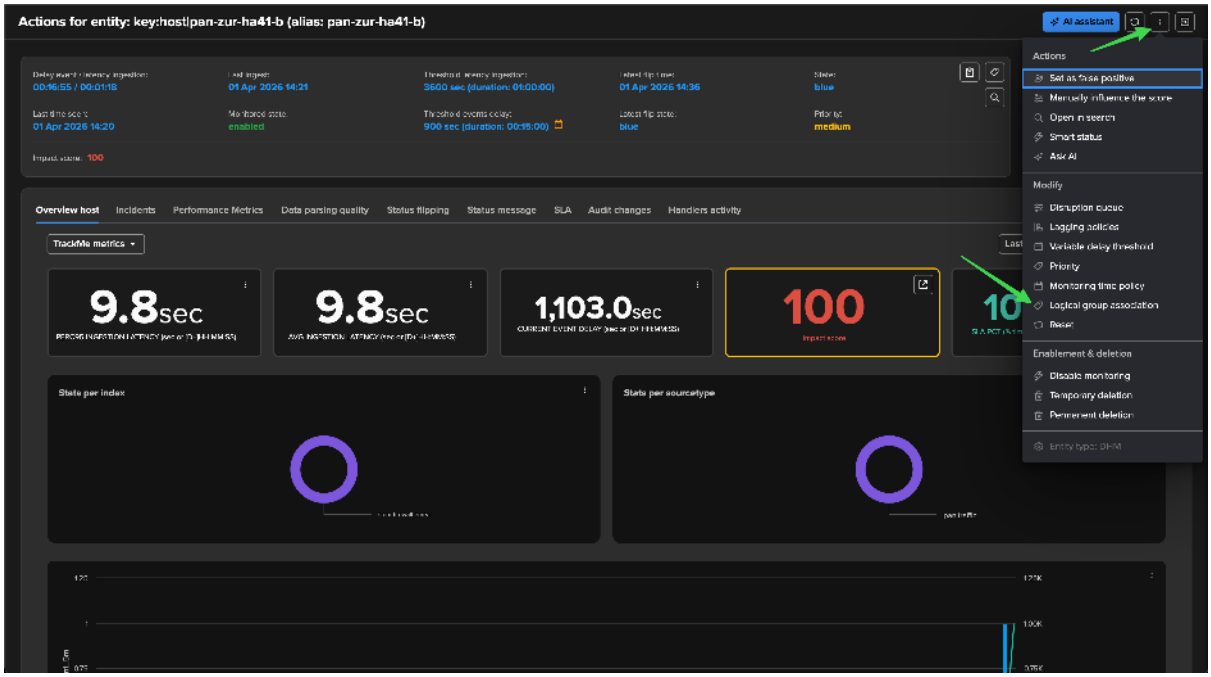




Managing groups

There are three interactive ways to manage logical groups, plus the REST API and the auto-group command below.

- **The Logical Groups manager** (Actions → *Manage: logical groups*) — the central screen to view, search, add, and remove groups and adjust each group's minimum green percentage.
- **Bulk edit** — select entities and *associate into a new group*, *associate into an existing group*, or *remove* their current association.
- **Per entity** — from an entity's actions, view, set, change, or clear its group membership.



Automating with the REST API

Every action is available through the REST API (resource groups `splk_logical_groups` for reads and `splk_logical_groups/write` for writes); all endpoints are called with `mode=post`. The common ones:

```
# List all groups in a tenant
| trackme mode=post␣
↪url="/services/trackme/v2/splk_logical_groups/logical_groups_collection"
    body="{'tenant_id': 'mytenant'}"

# Get one group
| trackme mode=post␣
↪url="/services/trackme/v2/splk_logical_groups/logical_groups_get_grp"
    body="{'tenant_id': 'mytenant', 'object_group_name': 'group-emea'}"

# Create or update a group and its members
| trackme mode=post␣
↪url="/services/trackme/v2/splk_logical_groups/write/logical_groups_add_grp"
    body="{'tenant_id': 'mytenant', 'object_group_name': 'group-amer',
           'object_group_members': 'entity1,entity2',
↪'object_group_min_green_percent': '50'}"

# Delete a group and clear its members' associations
| trackme mode=post␣
↪url="/services/trackme/v2/splk_logical_groups/write/logical_groups_del_grp"
    body="{'tenant_id': 'mytenant', 'object_group_name': 'group-amer'}"
```

Further endpoints exist for bulk and per-entity association — see the *REST API reference*.

Automating at scale with `trackmeautogroup`

In a large estate you do not want to build groups by hand. The custom command ``**trackmeautogroup**`` creates, updates, and prunes logical groups automatically from an SPL search you write — the practical way to keep groups in sync with reality.

It is a **streaming** command: it consumes the entities your upstream SPL produces, and each input row must carry two fields:

- `object_group_name` — the name of the group the entity belongs to;
- `object_group_members` — the group's members, as a multi-value field.

From those rows it:

- **creates** the group if it does not exist;
- **updates** the membership if it has changed, and sets `object_group_min_green_percent` automatically from the member count (`round(100 / members, 2)` — 50% for two, 33.33% for three, …);
- **prunes** a group down to nothing once only **one** active member remains — controlled by `purge_single_member_grp` (`True` by default; set `False` to keep single-member groups).

It requires `tenant_id`.

Identifying members at scale

The real work is deciding *which entities belong together*. Common approaches:

- **Naming convention** — extract a shared pair/zone identifier from the entity name with a regular expression.
- **CMDB enrichment** — join a Splunk lookup (your CMDB) to derive the grouping key.
- **Custom logic** — any SPL that emits `object_group_name` and `object_group_members`.

Example — by naming convention:

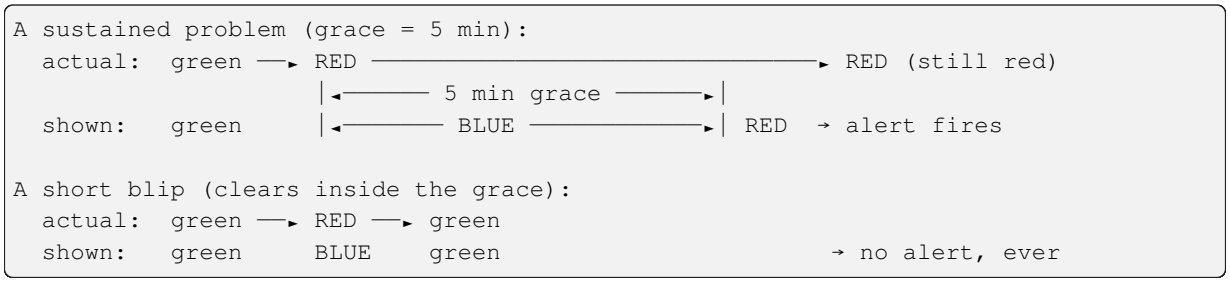
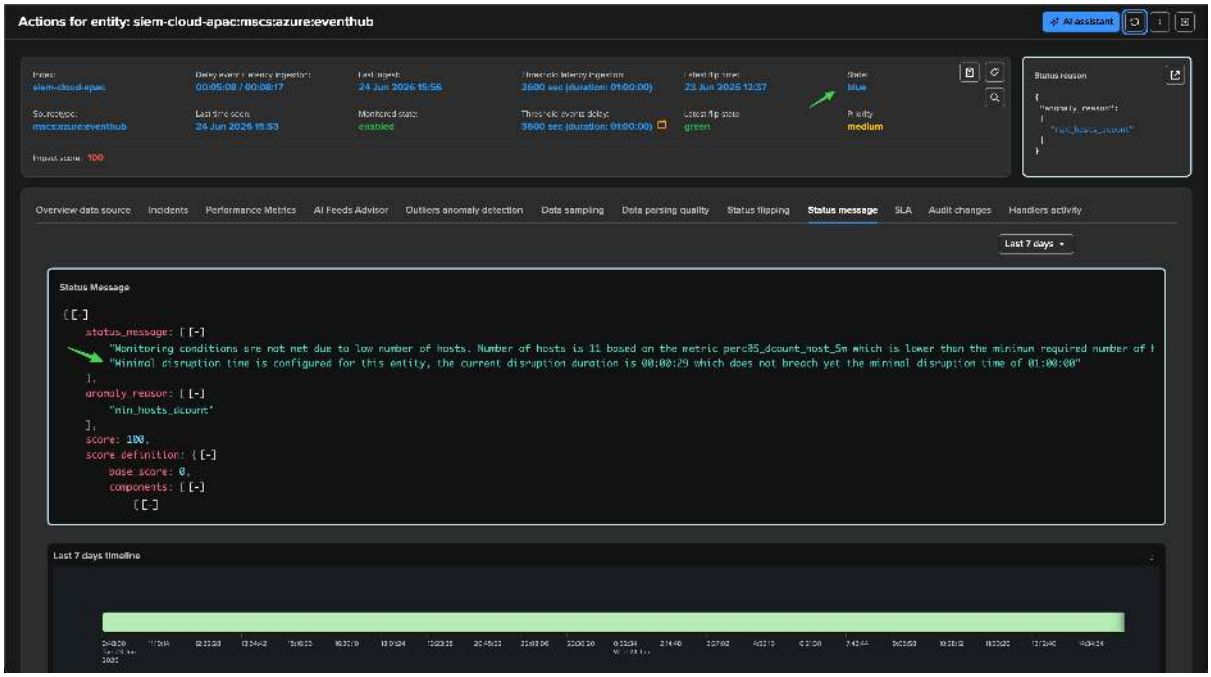
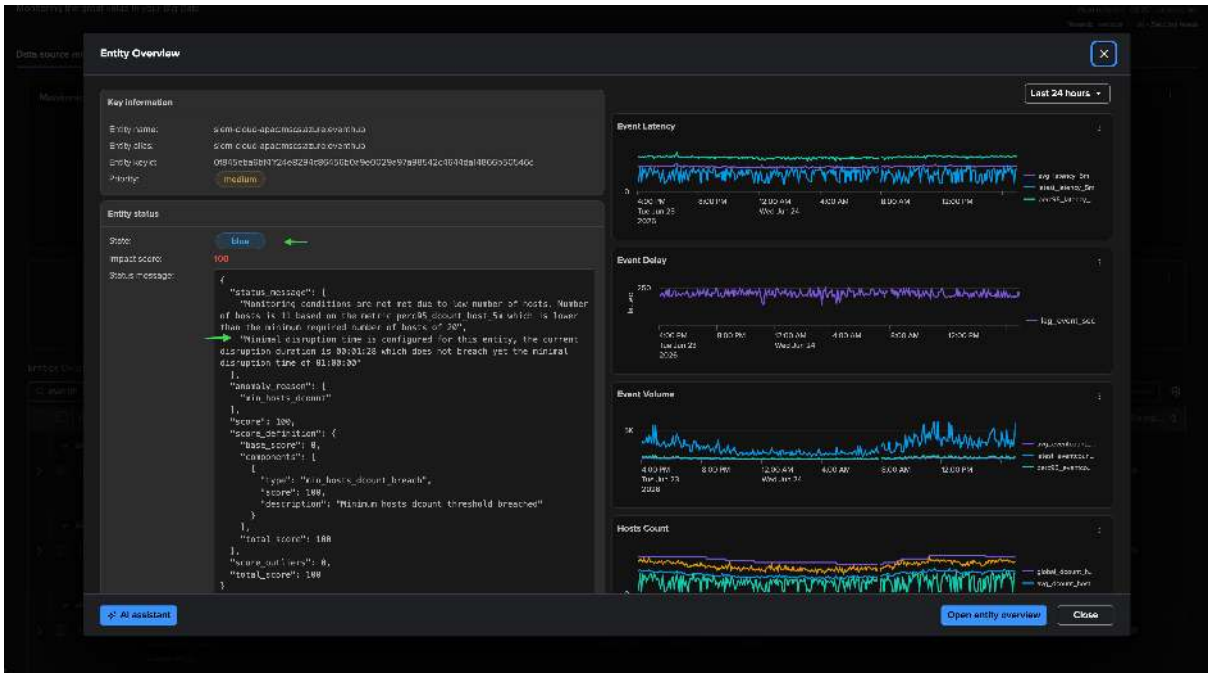
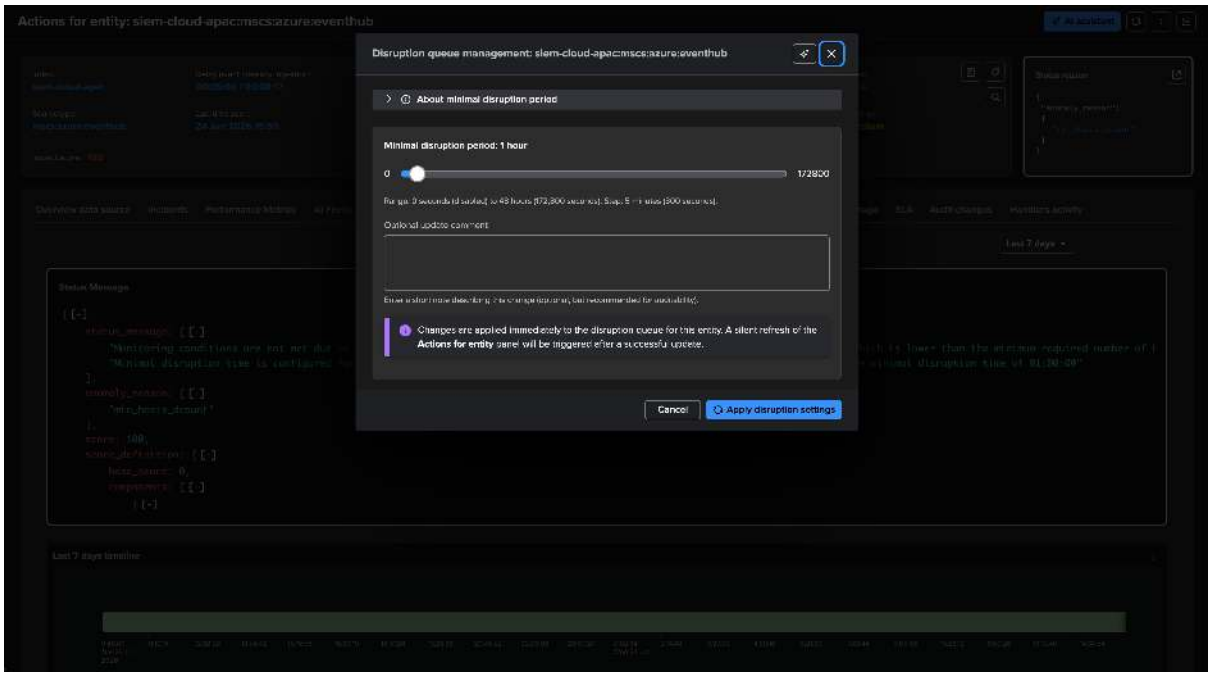
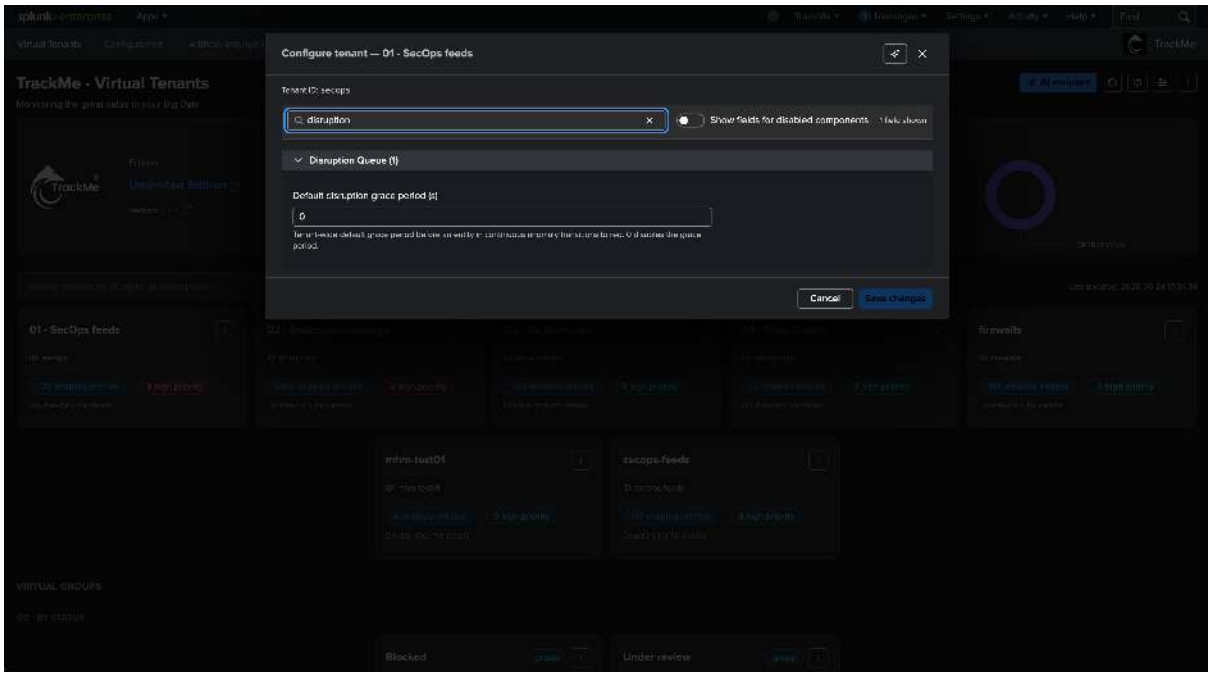
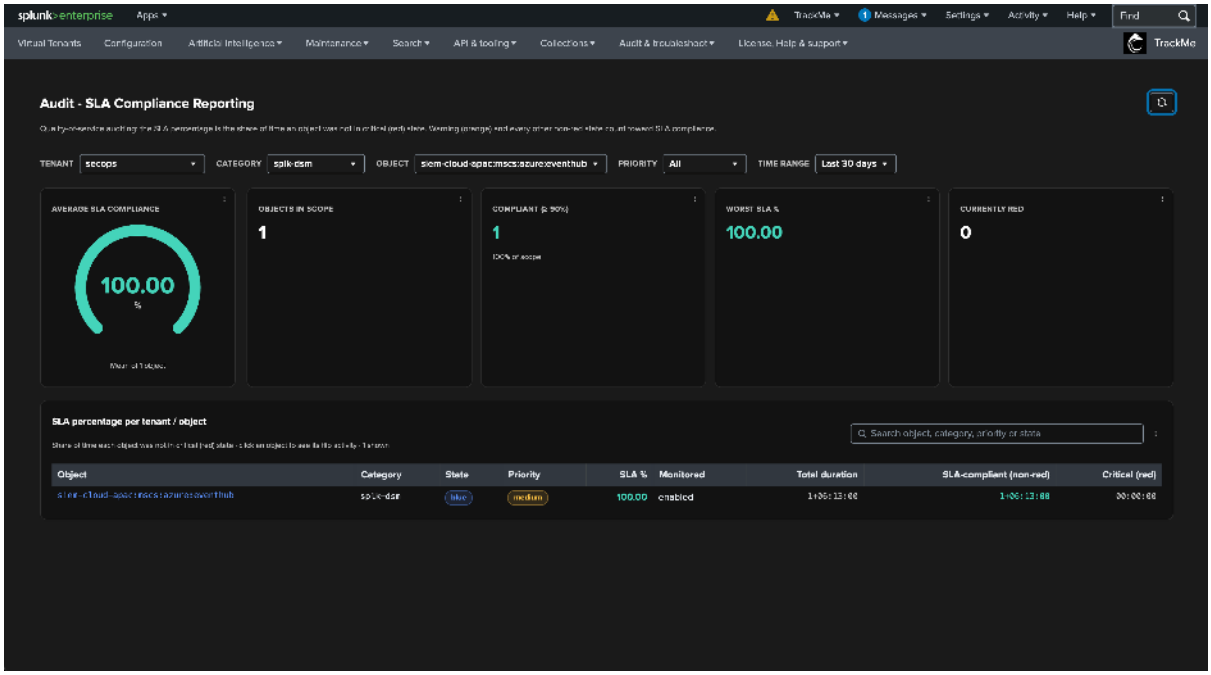
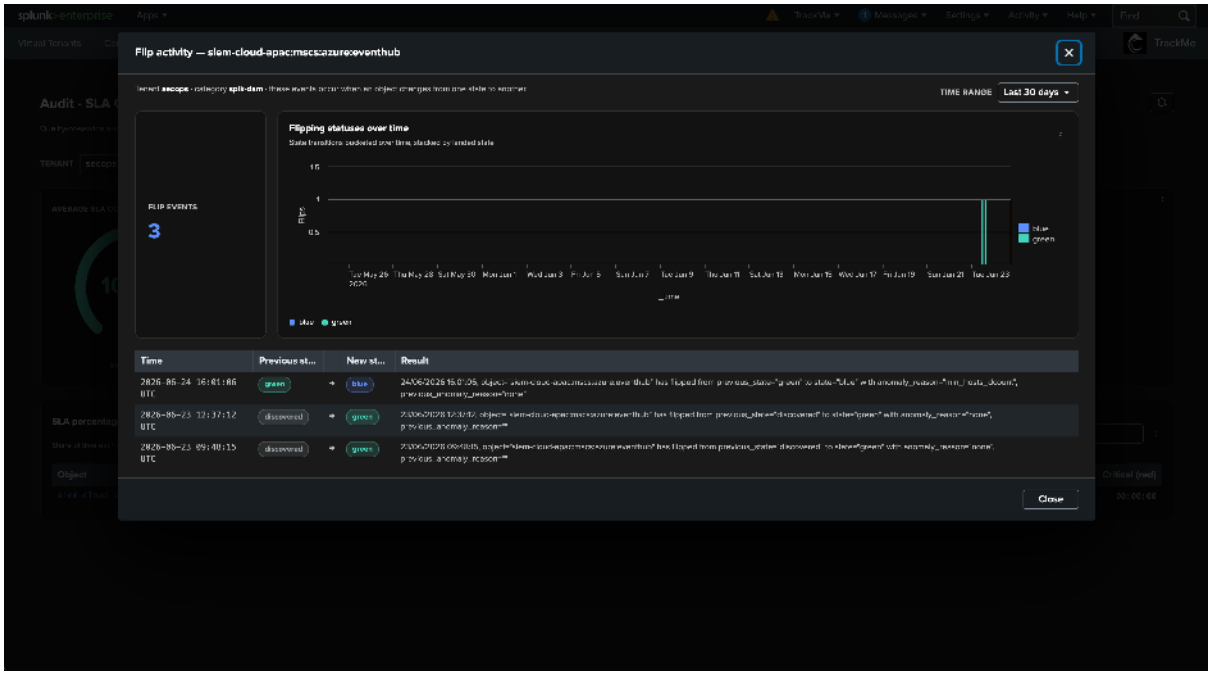
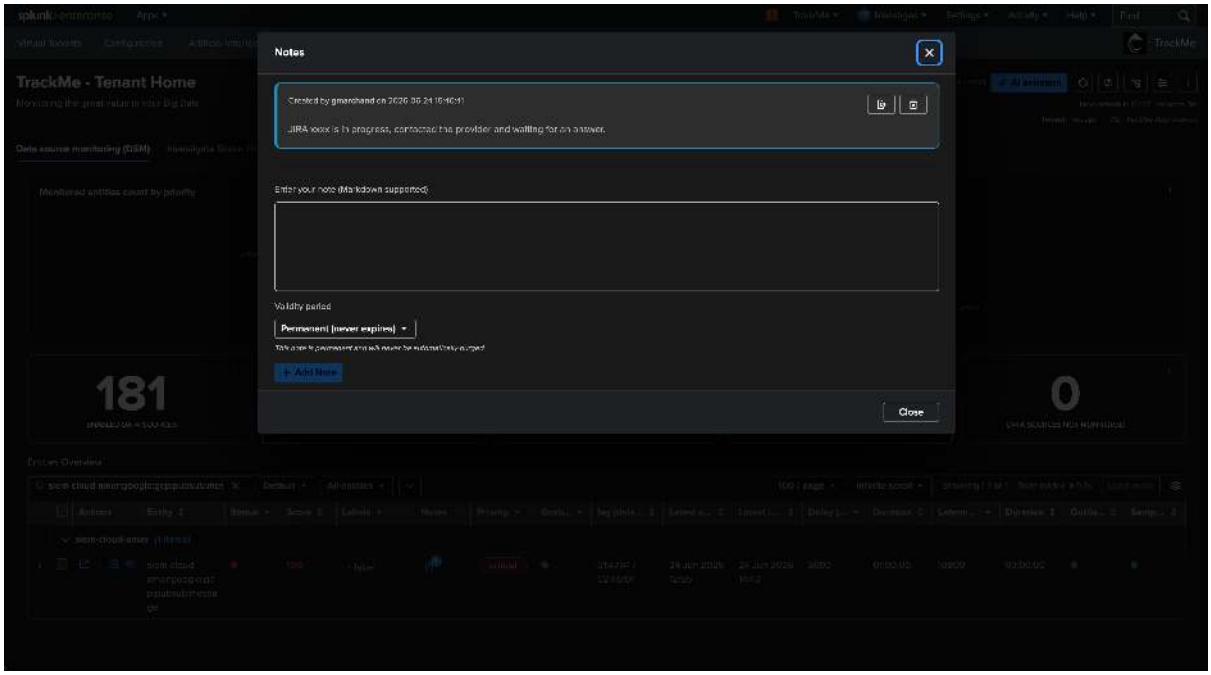
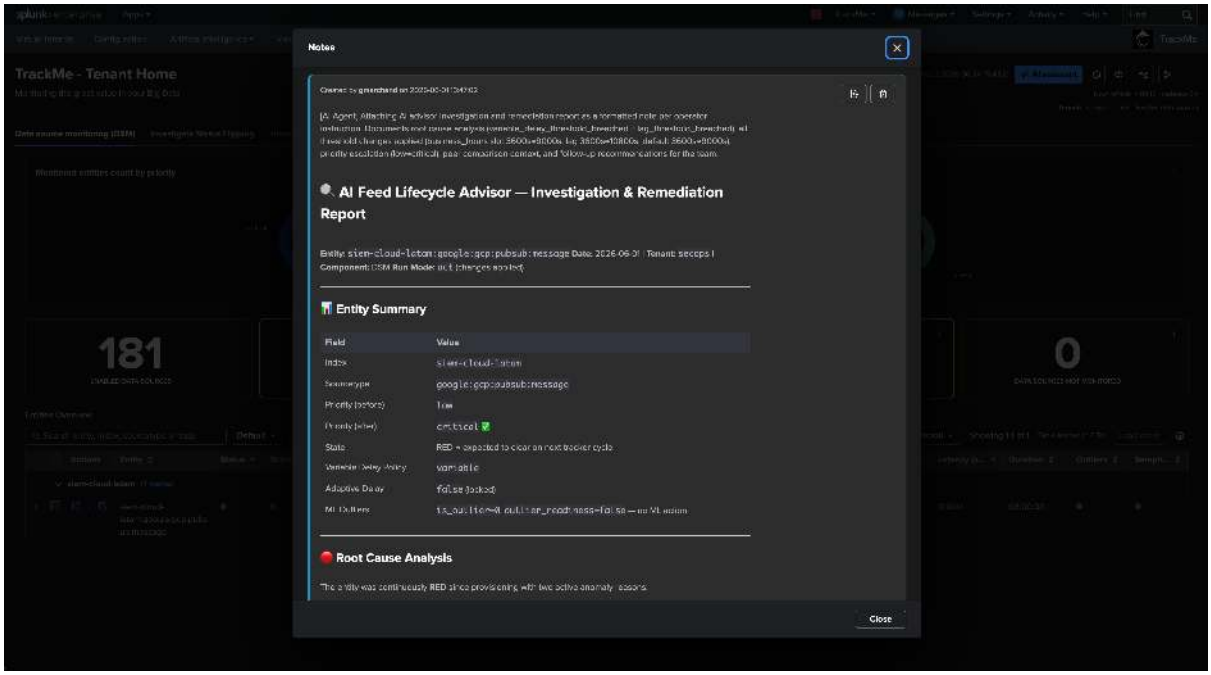
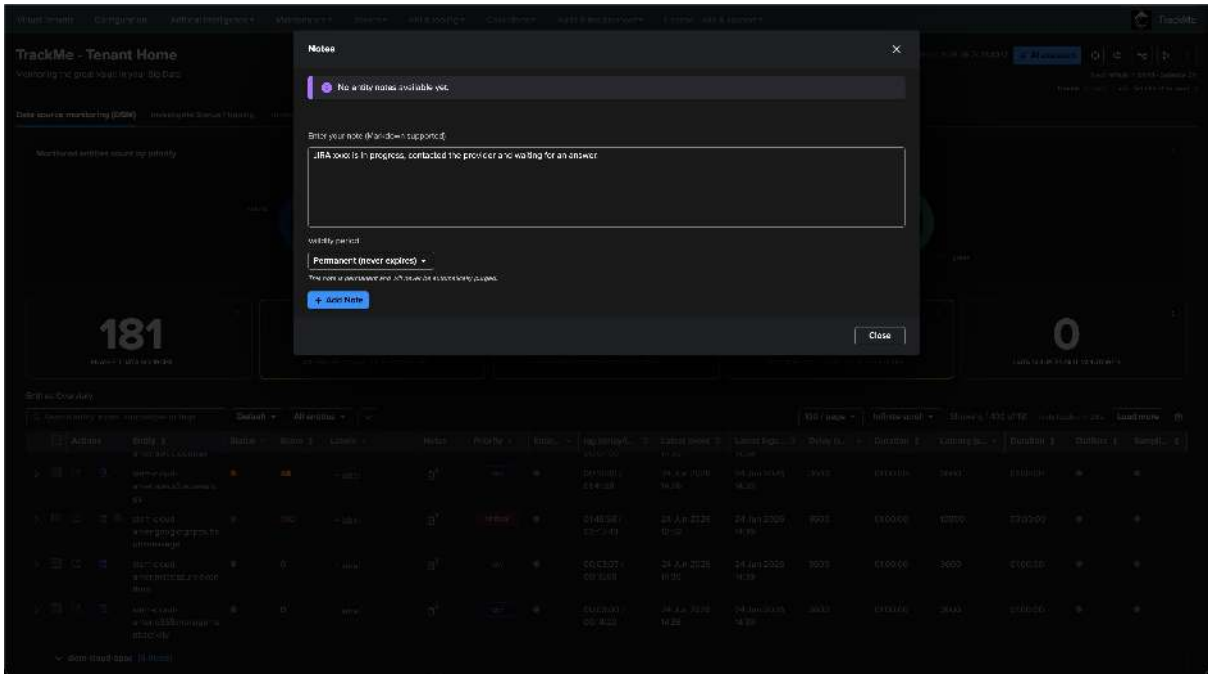
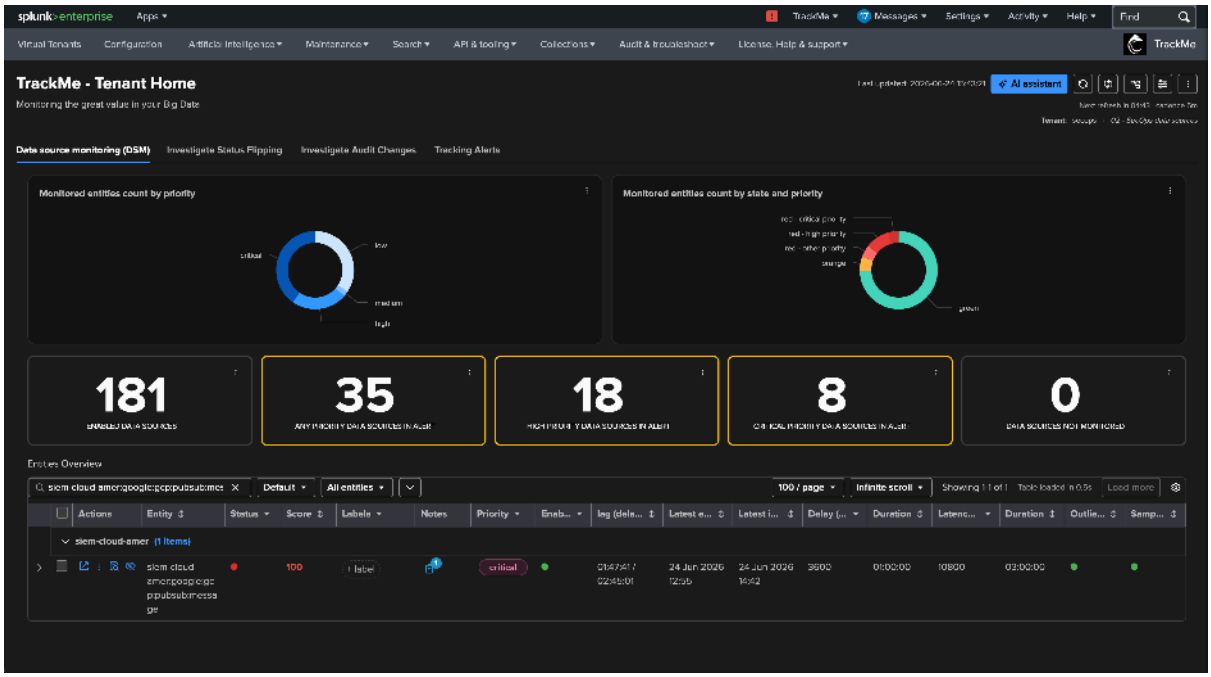
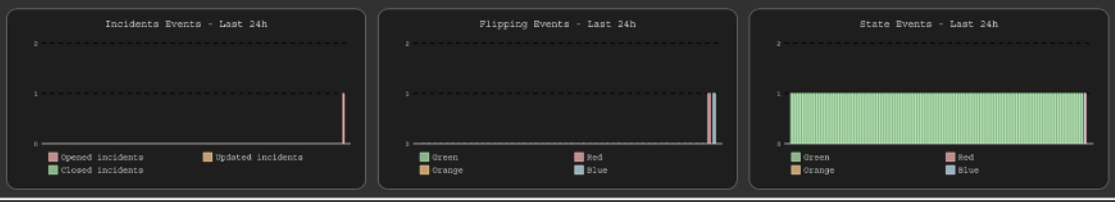
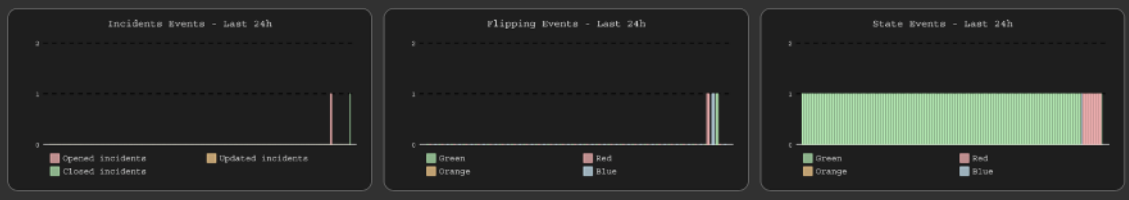
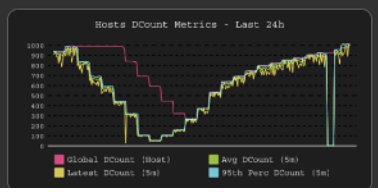
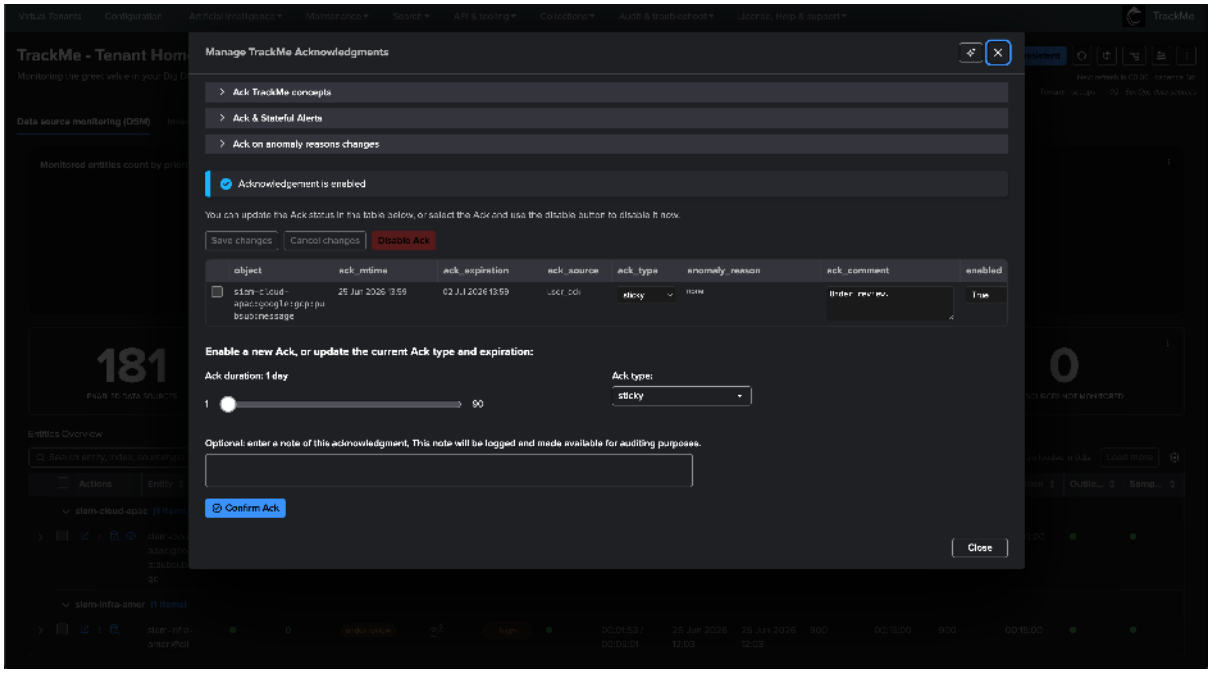
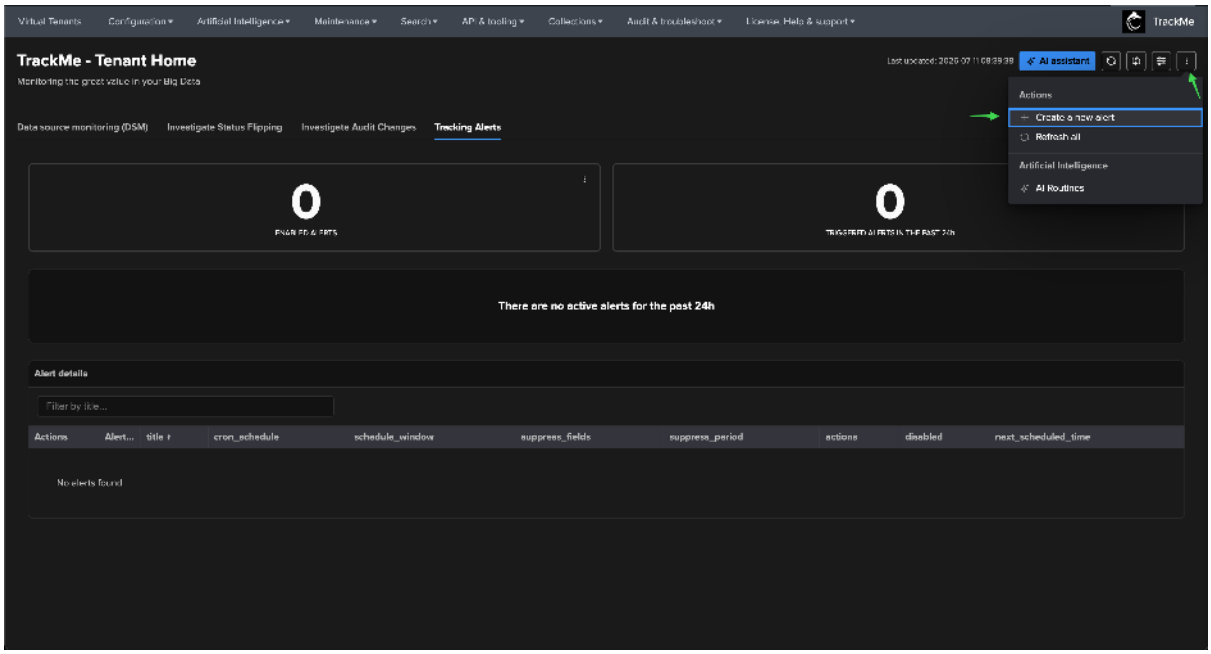
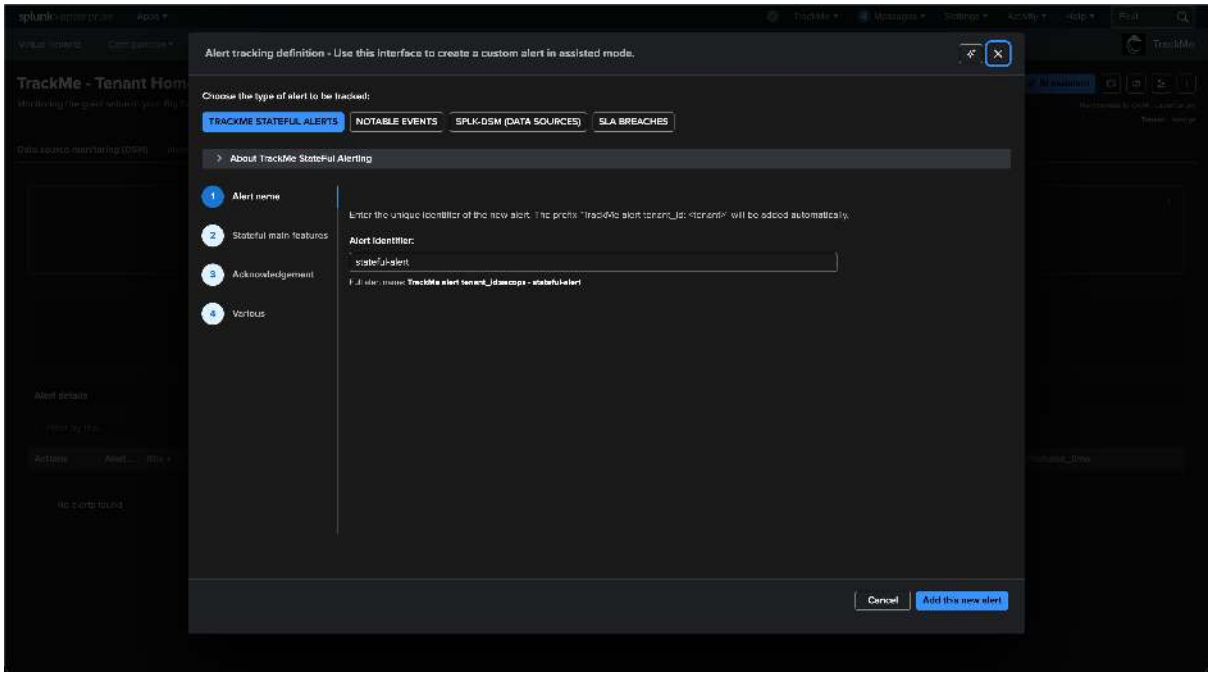
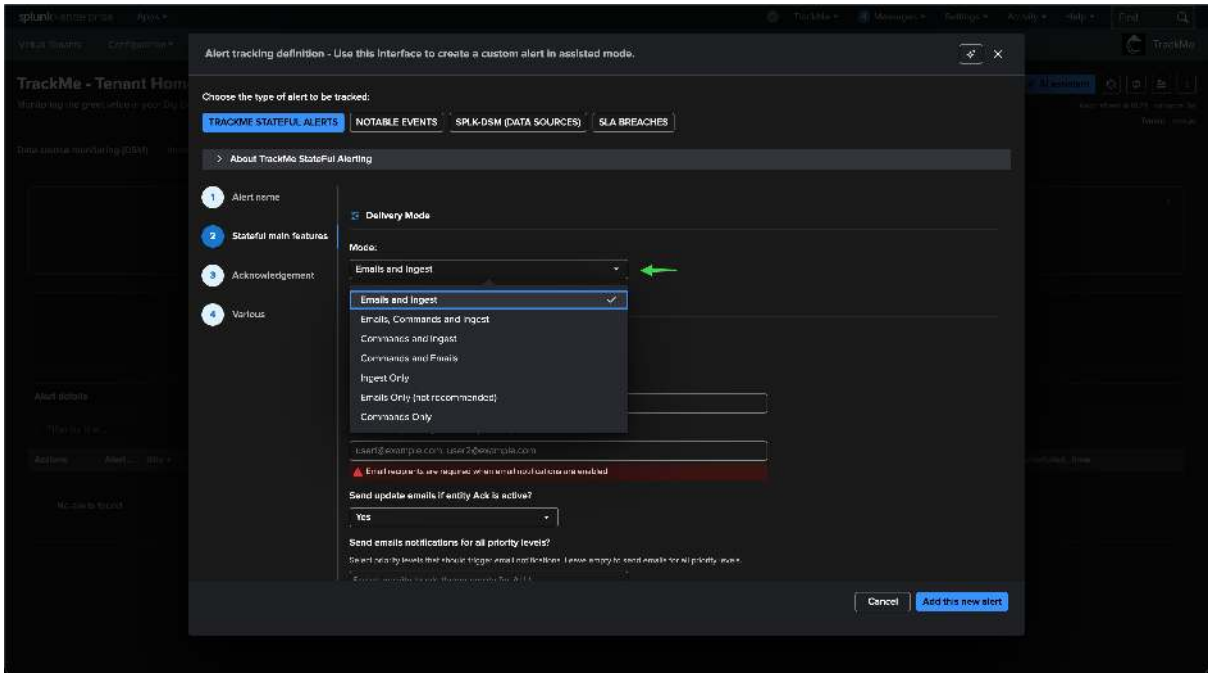
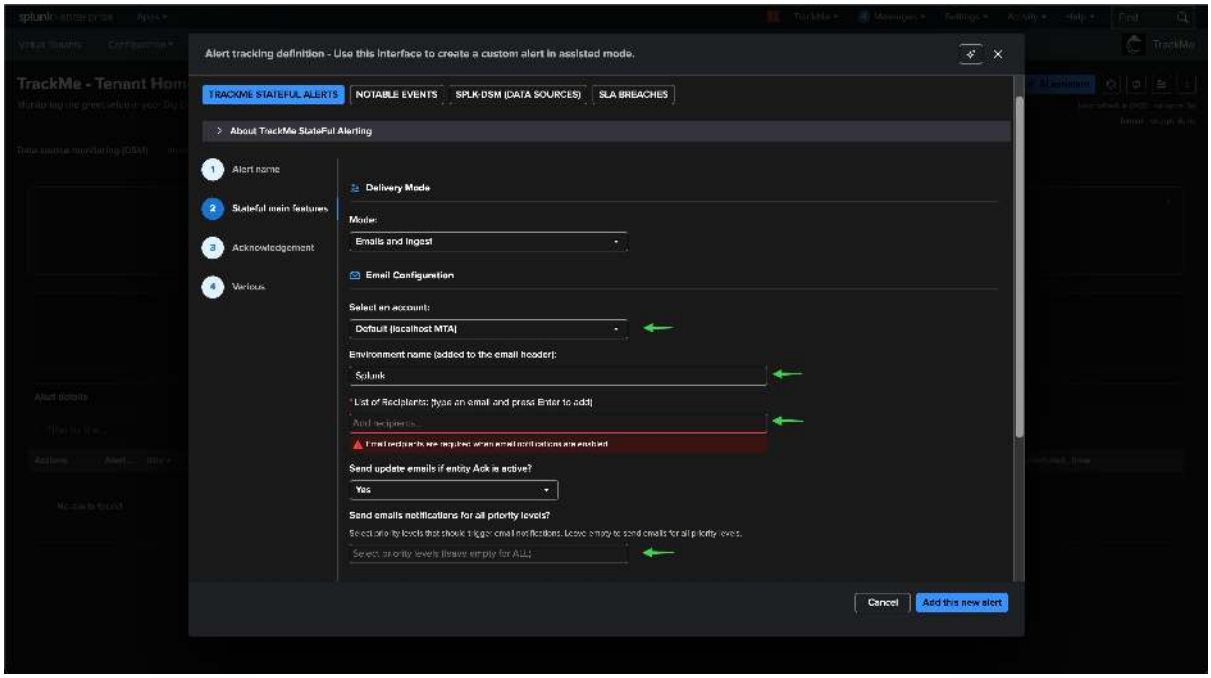
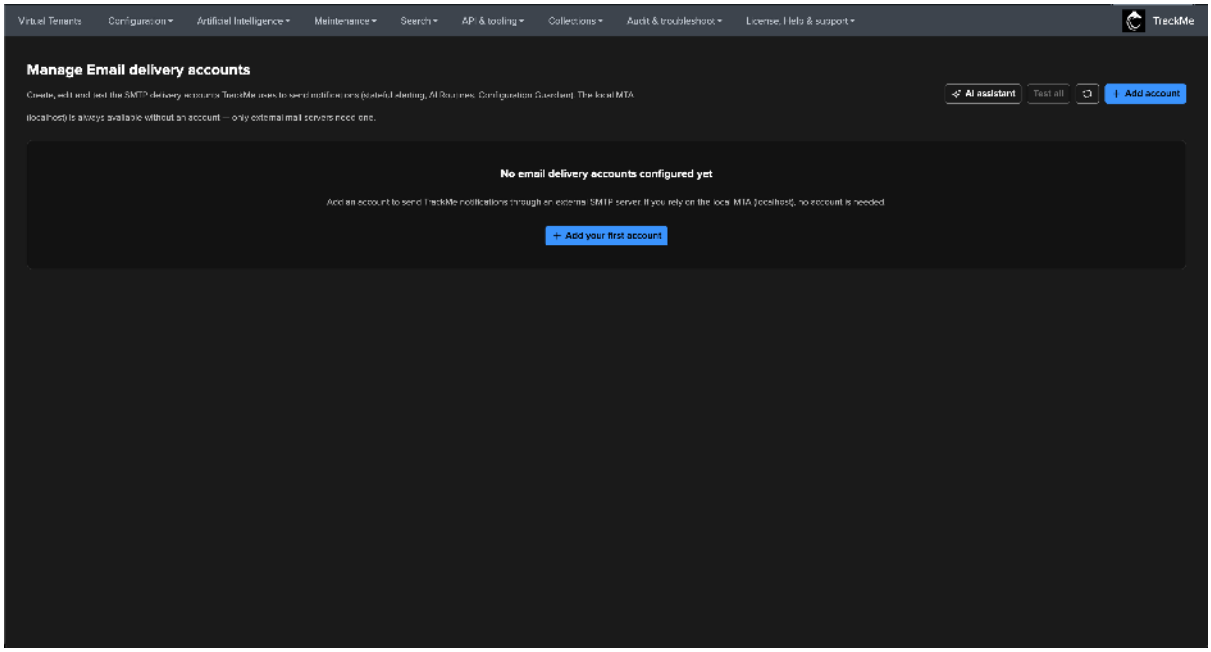
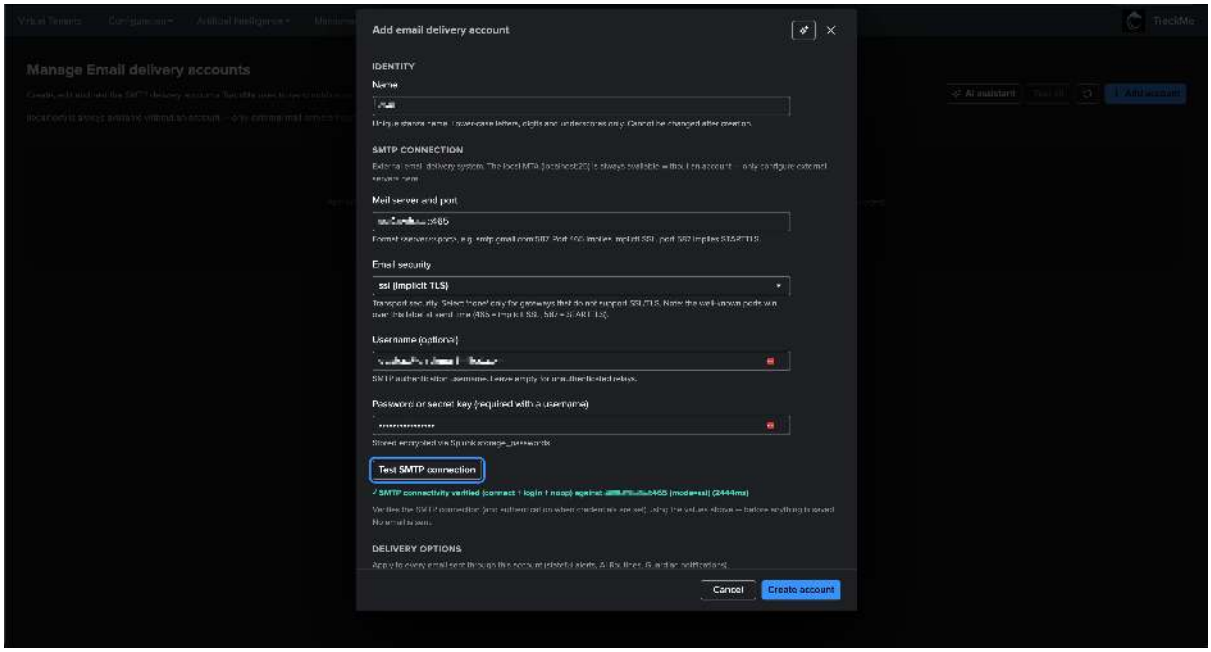
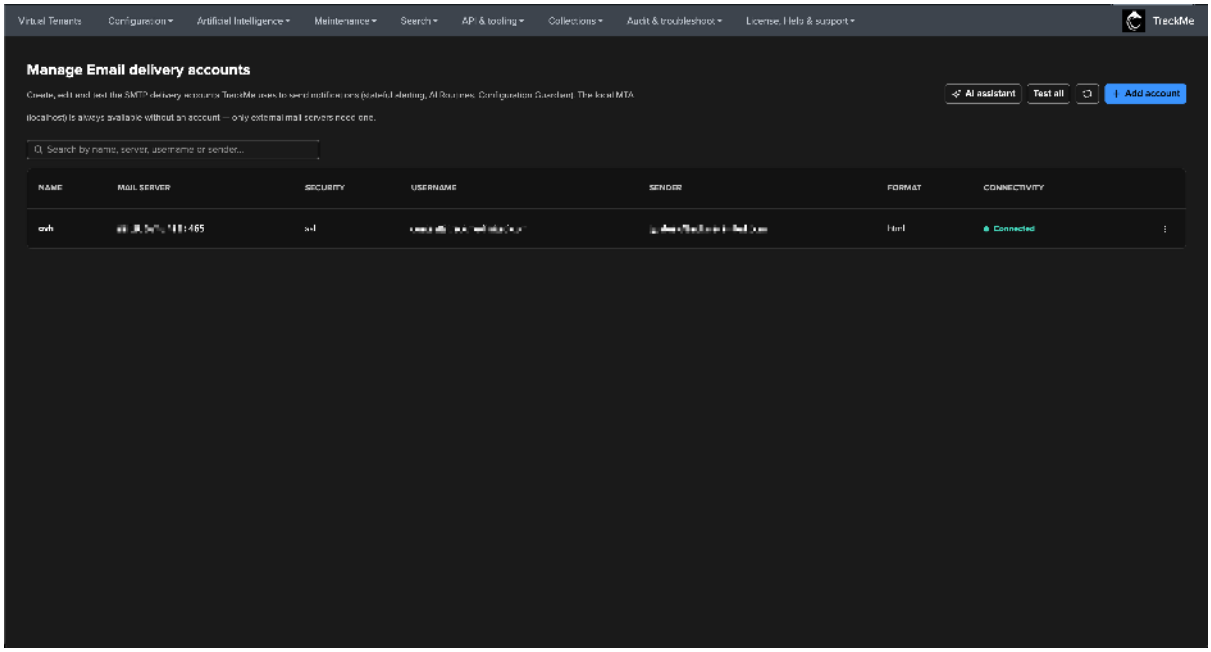
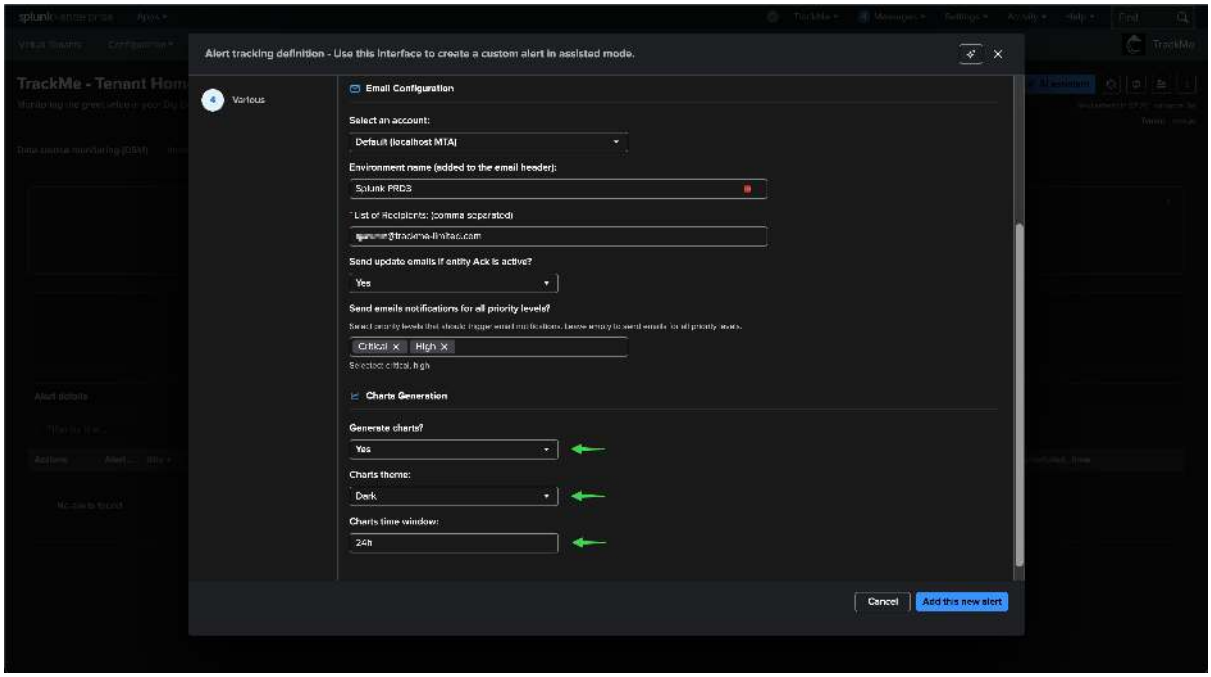
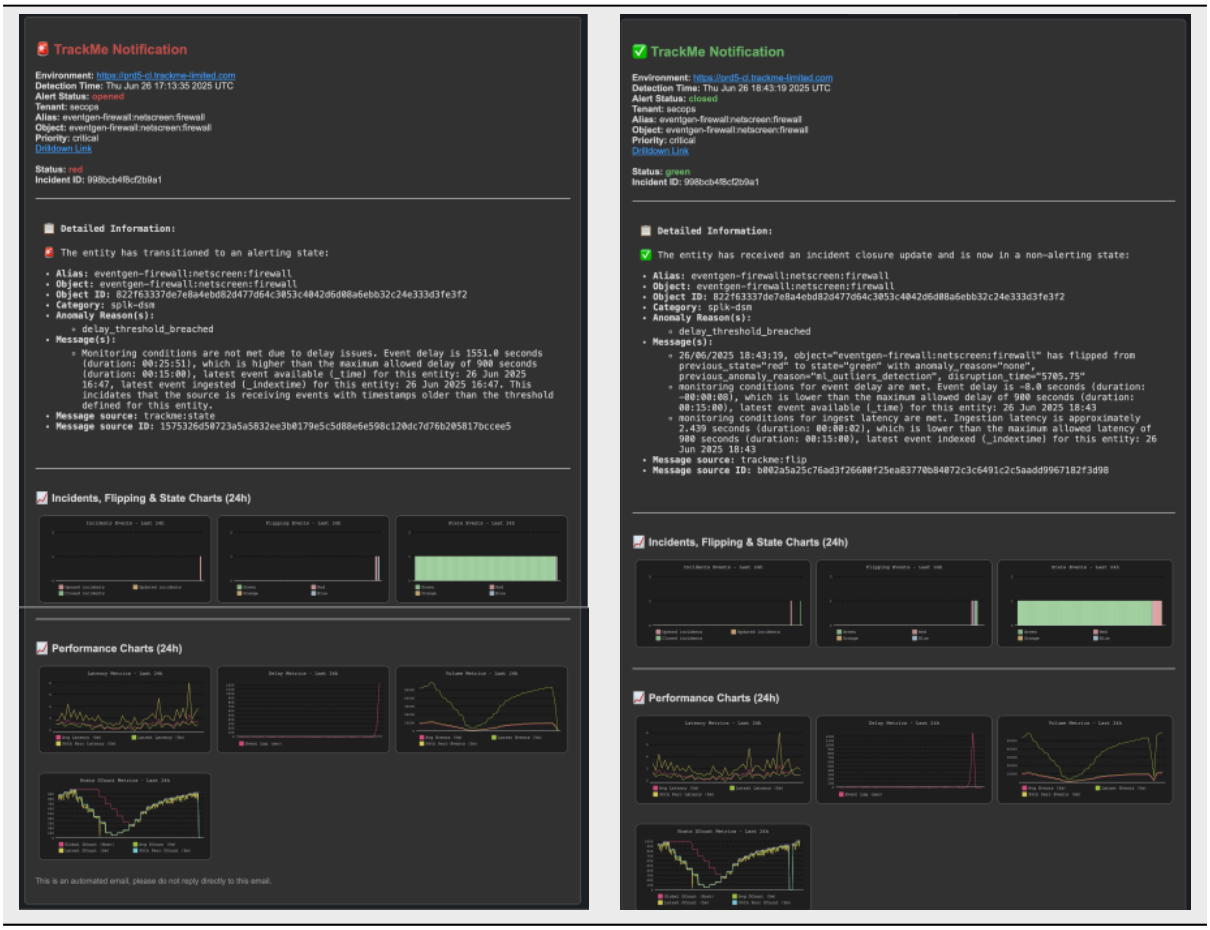
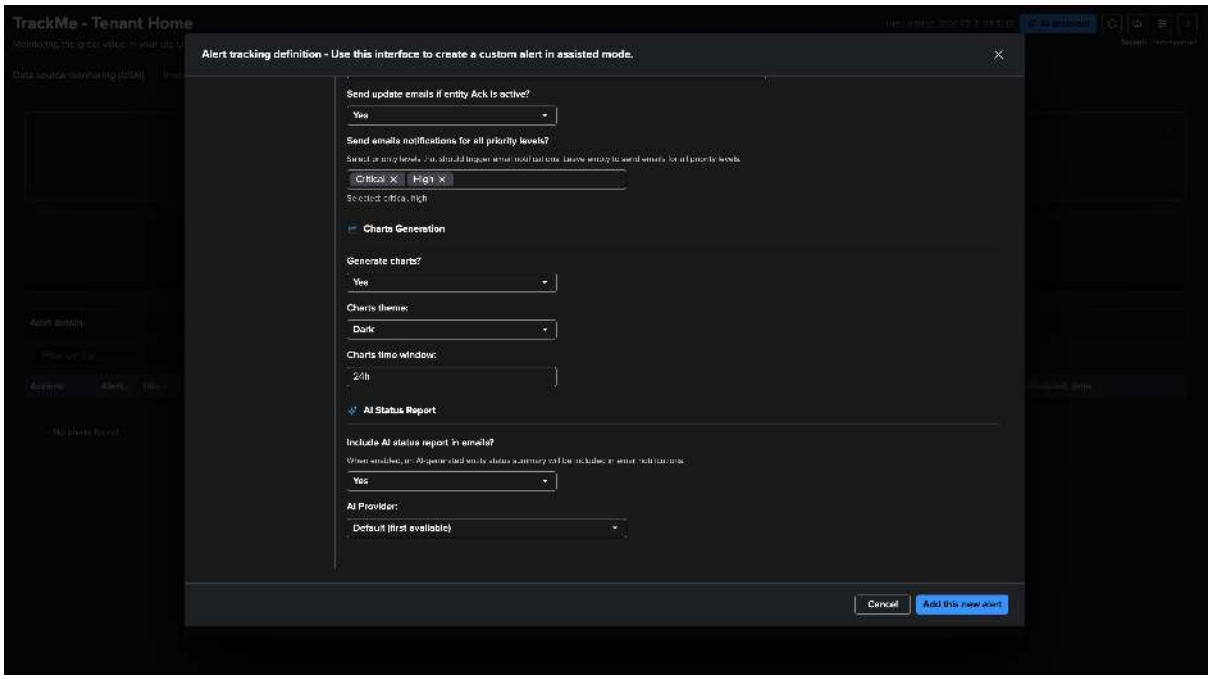
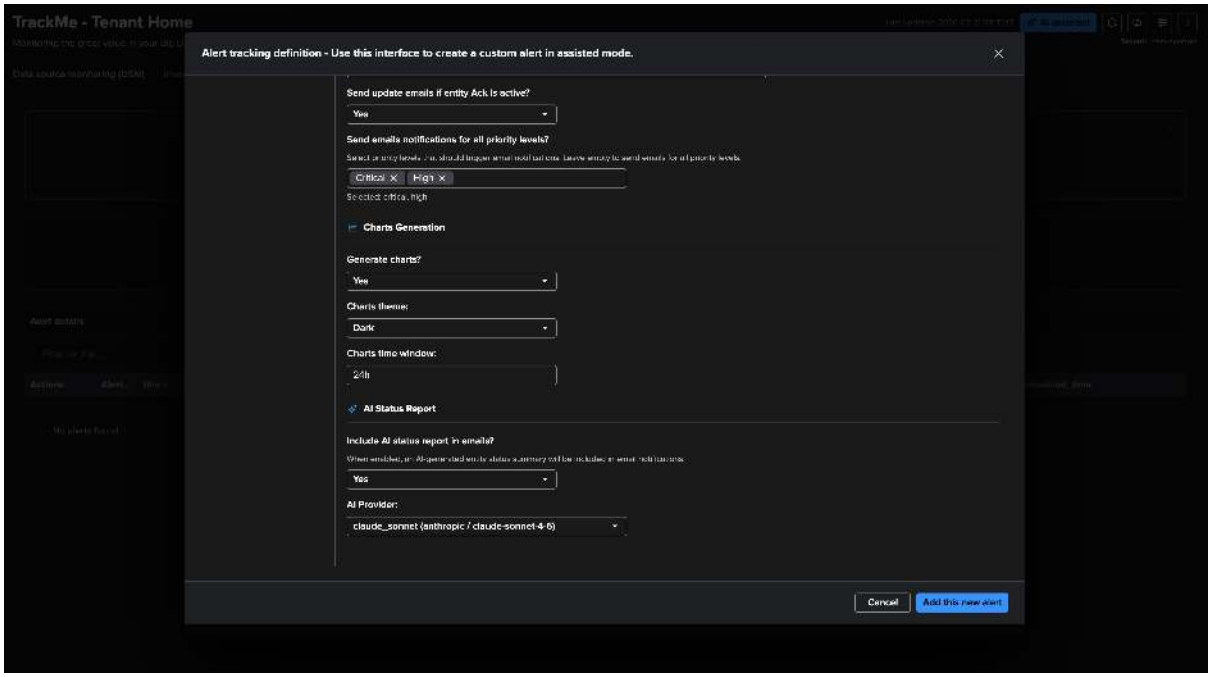
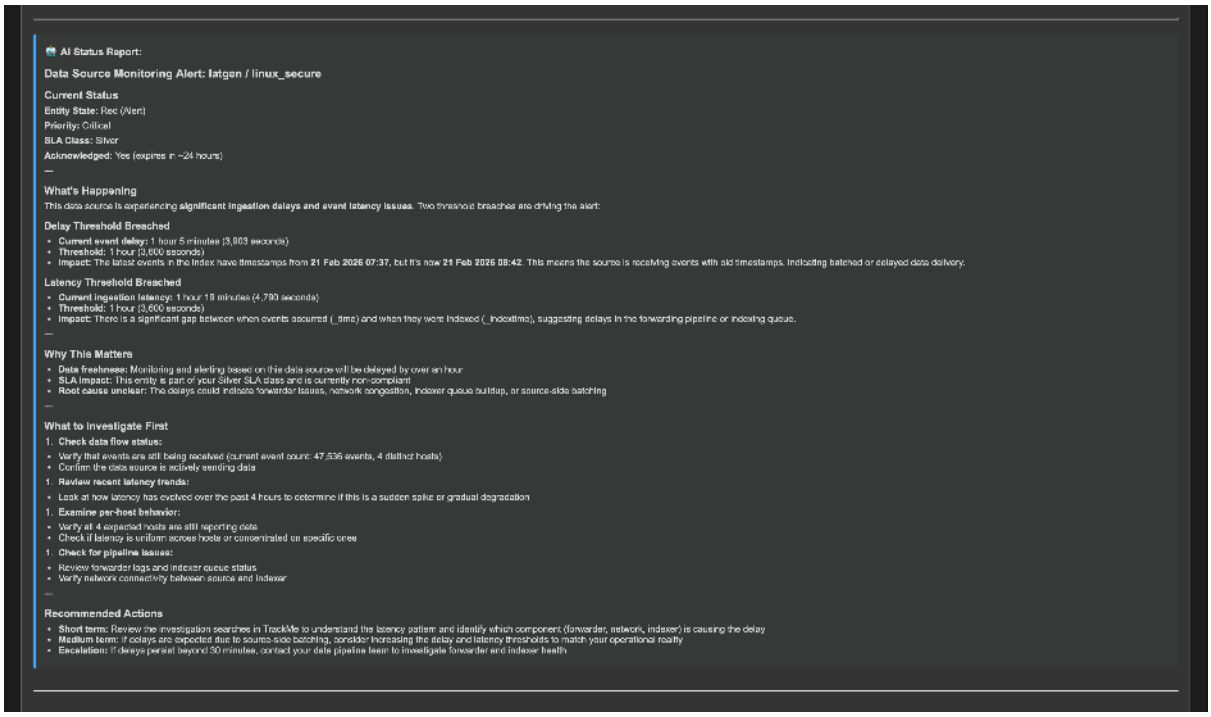
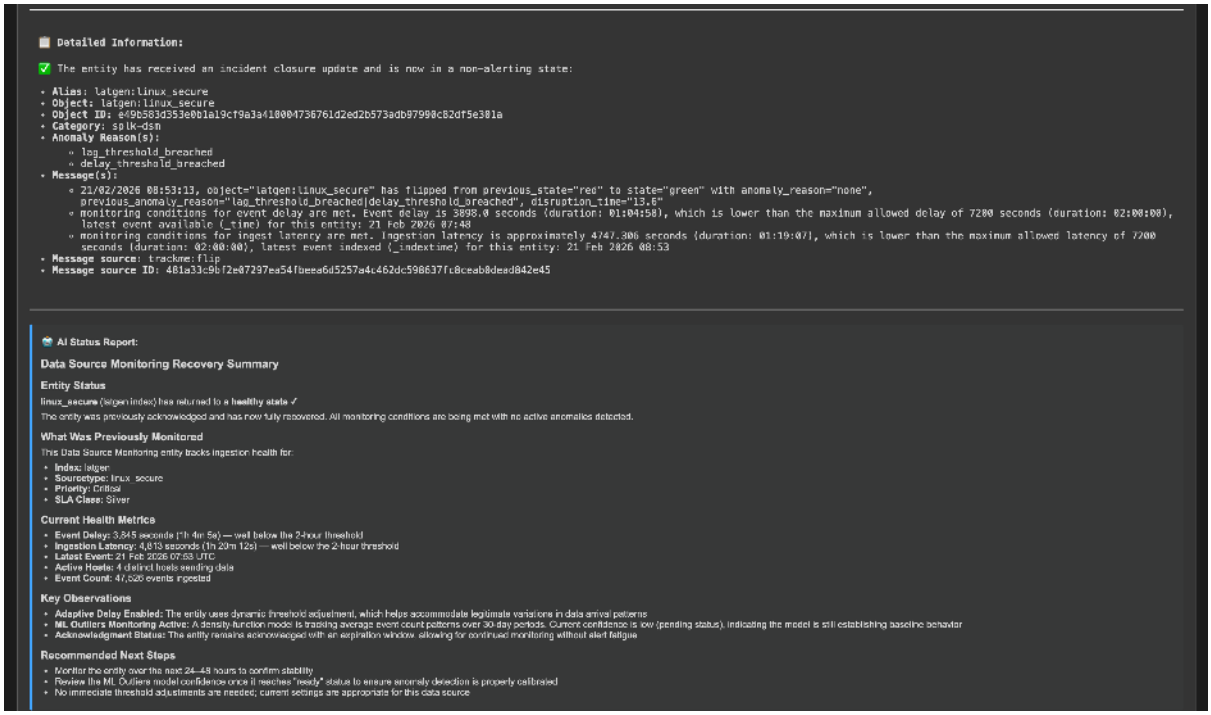
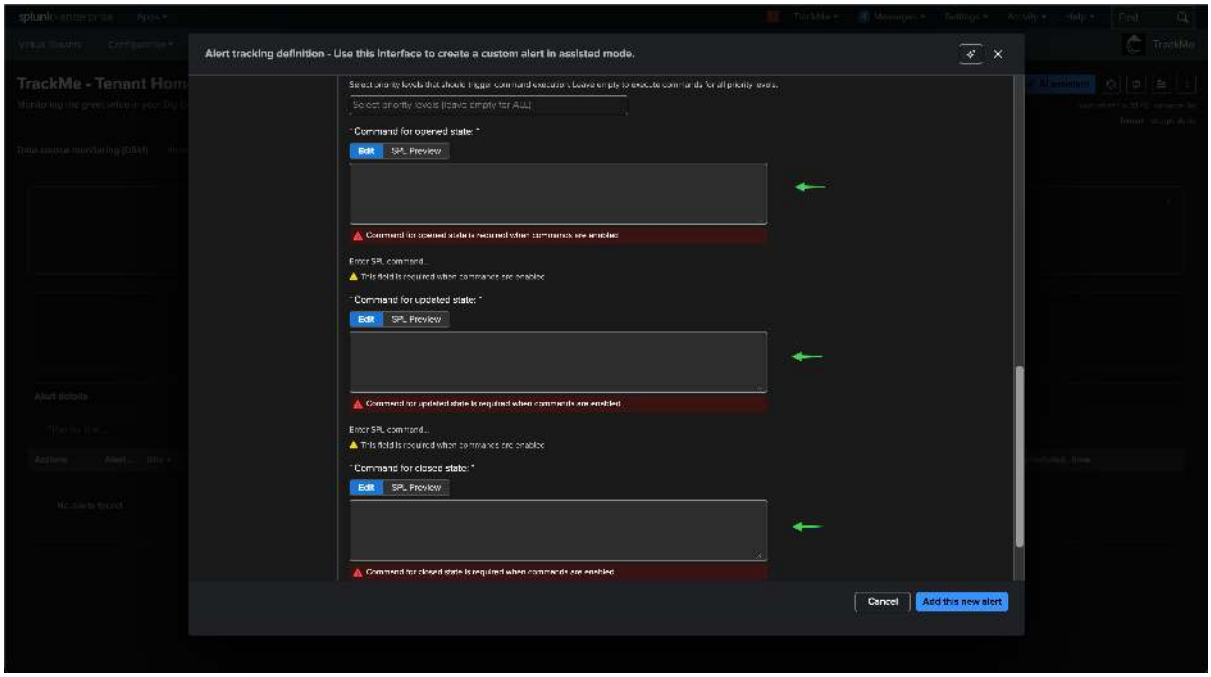
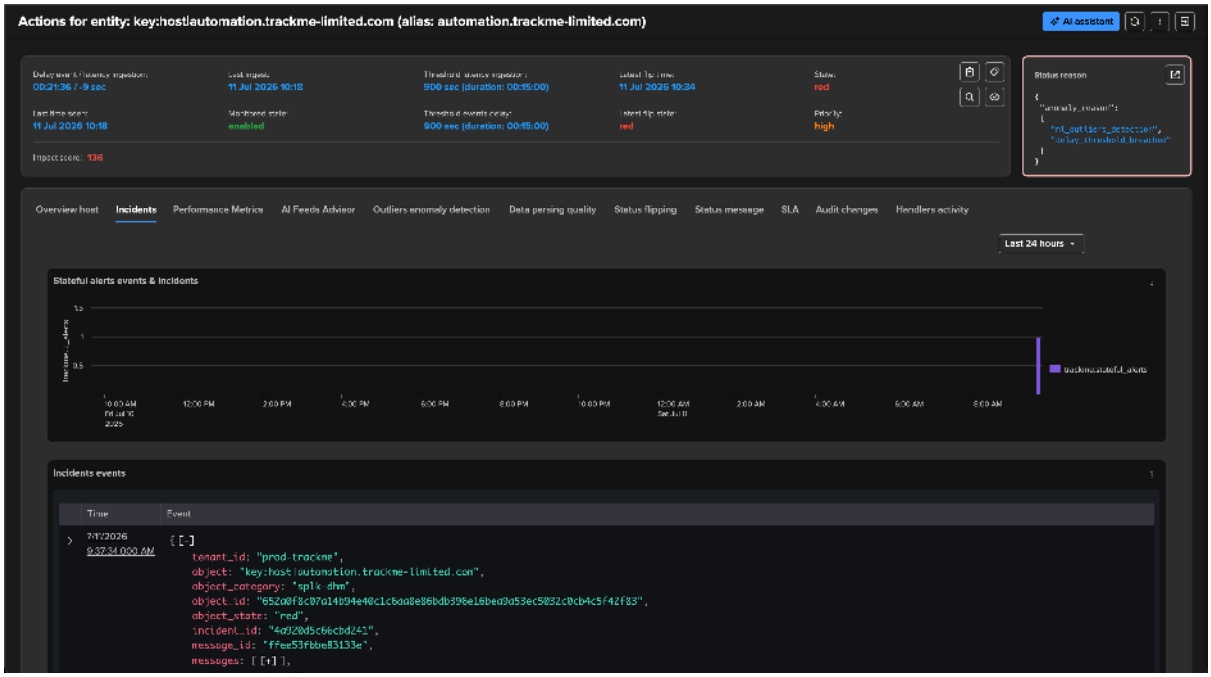
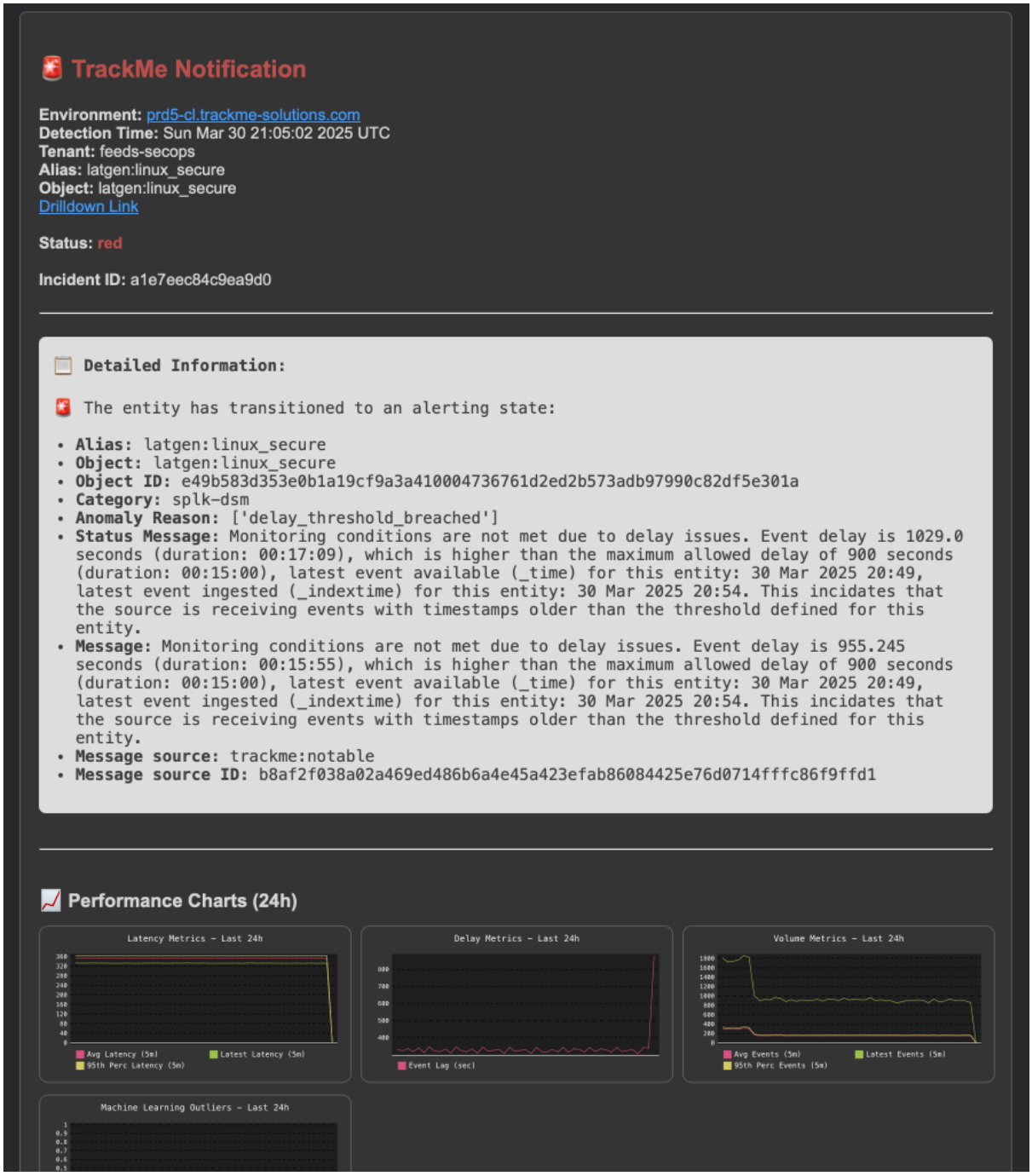
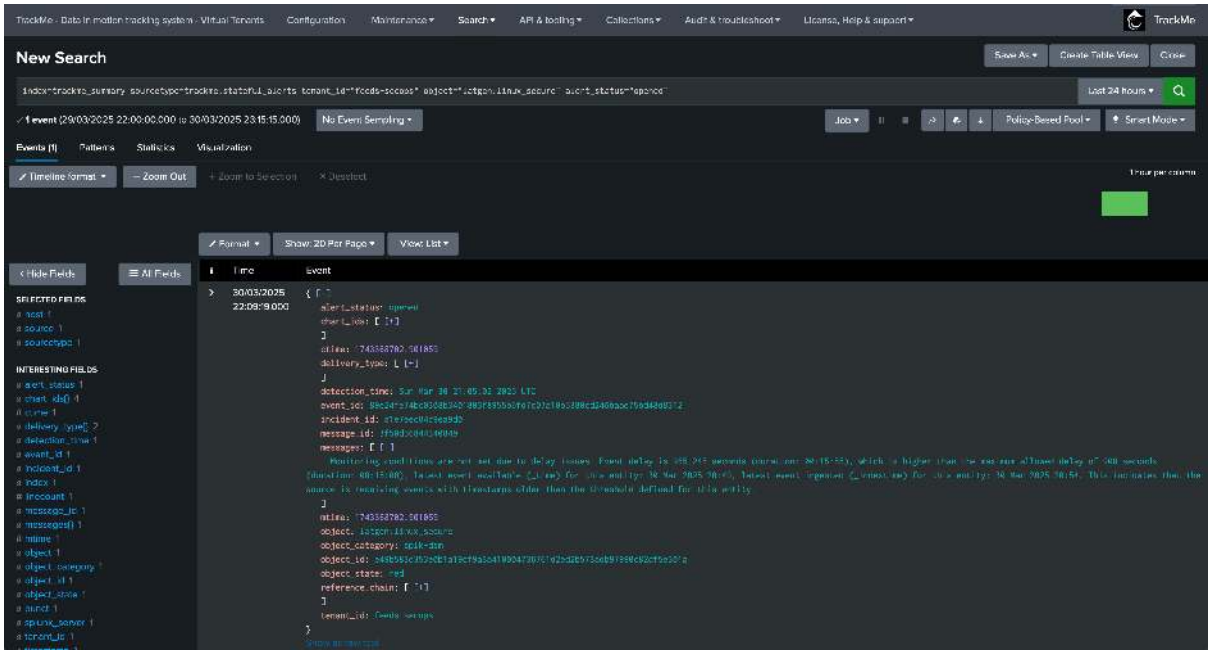
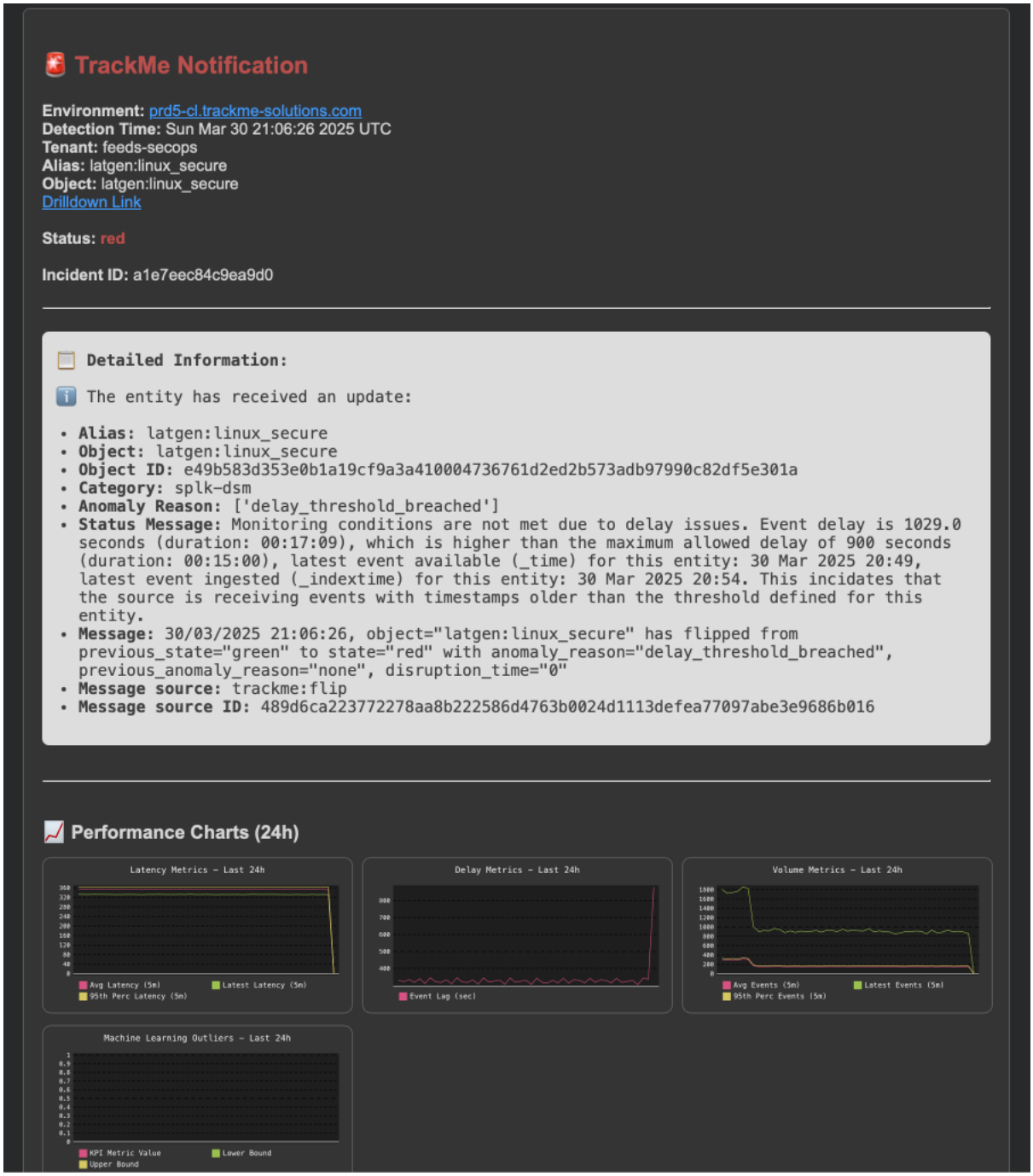
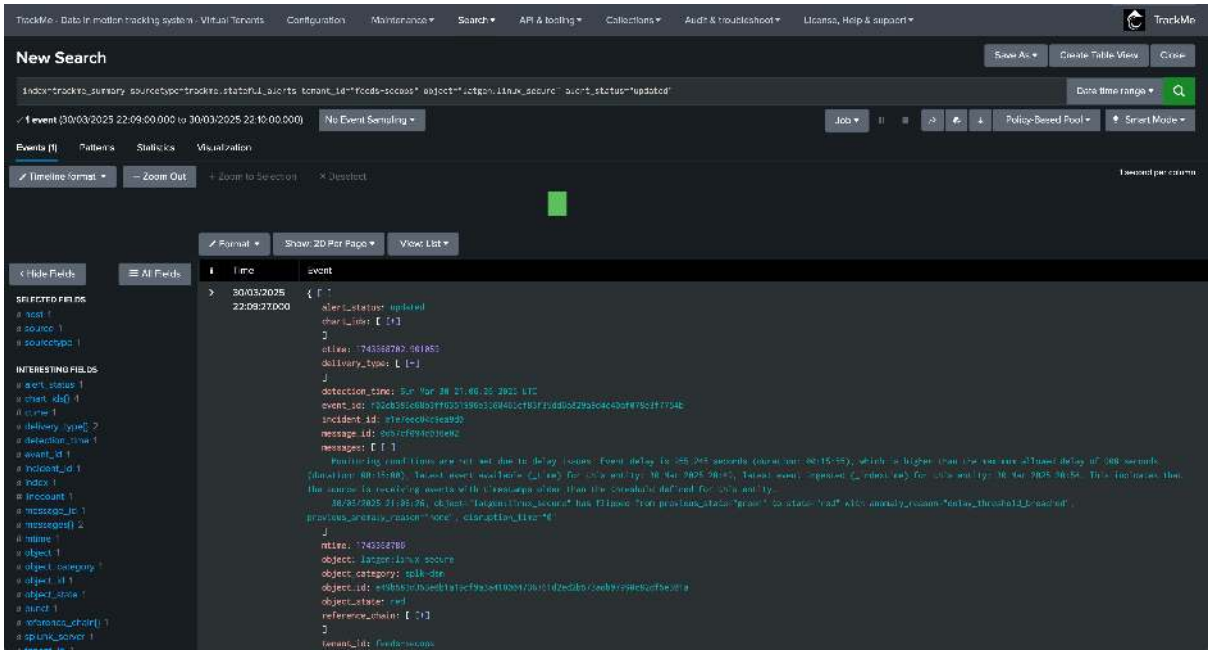
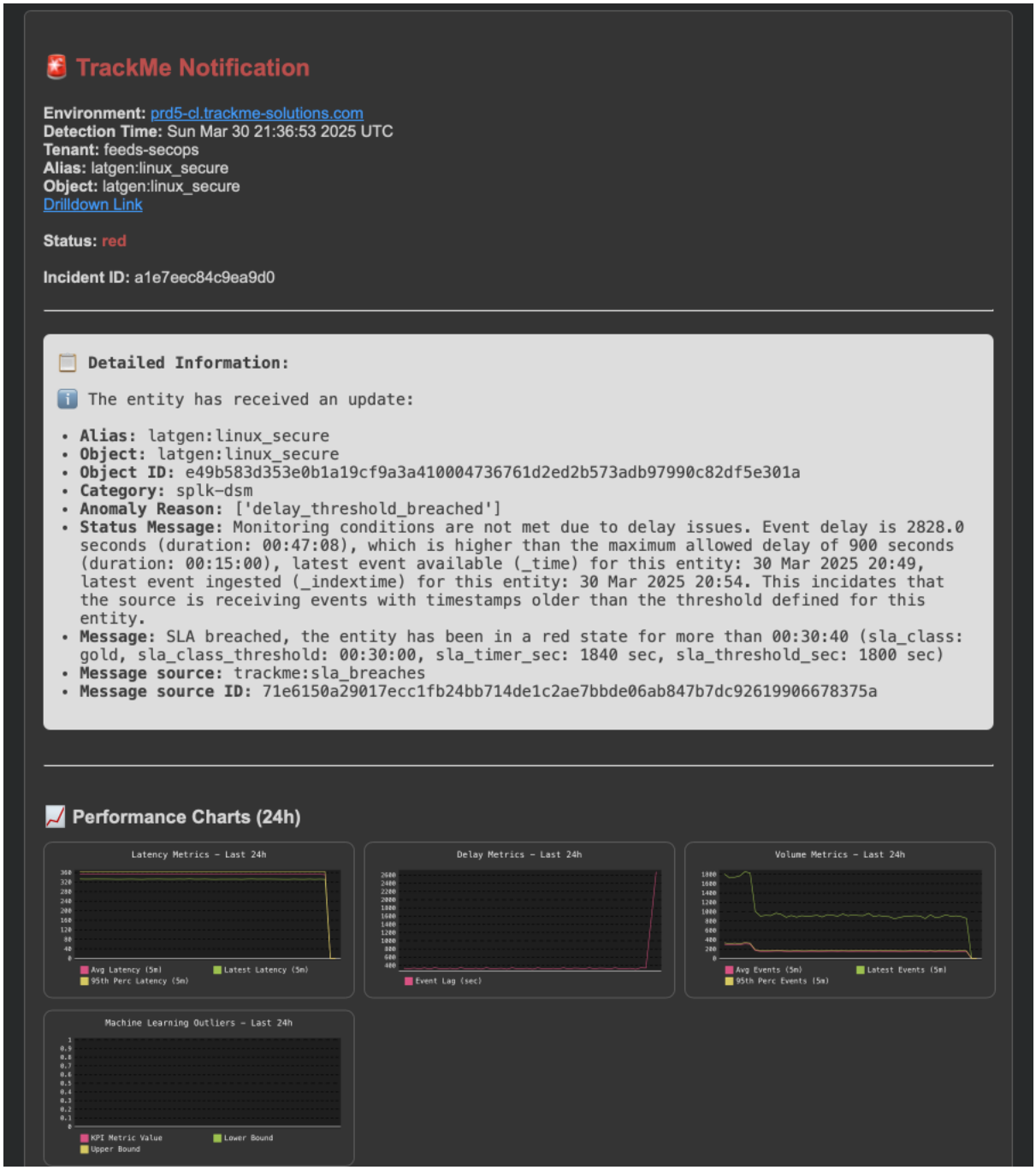
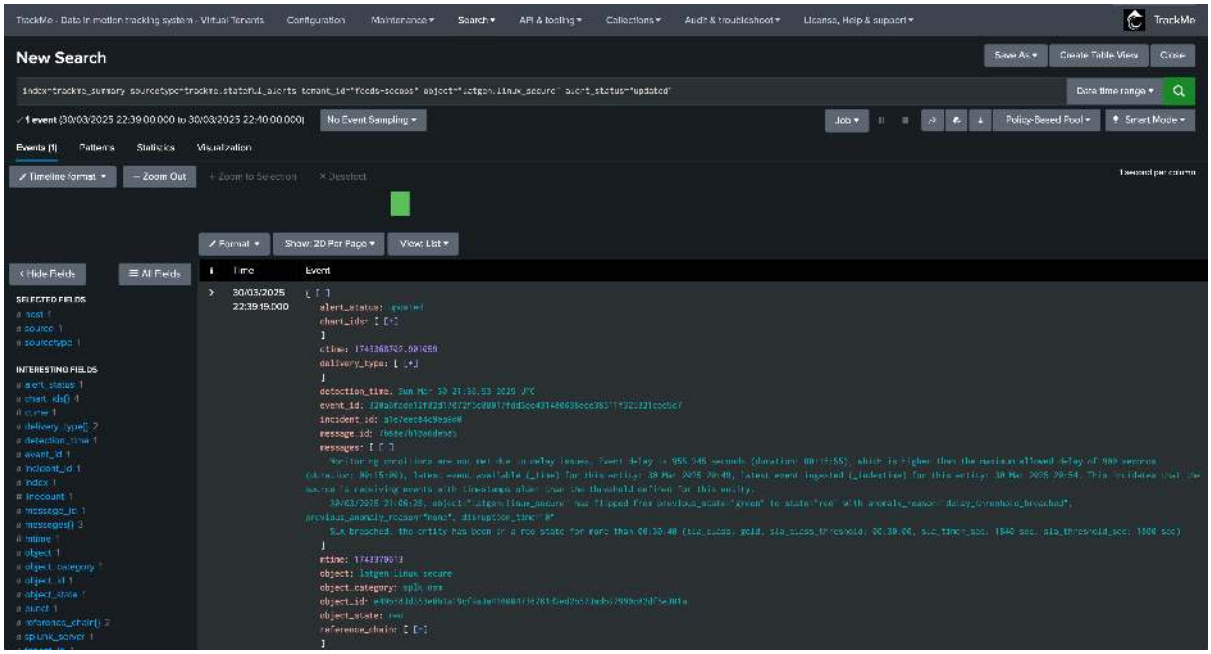
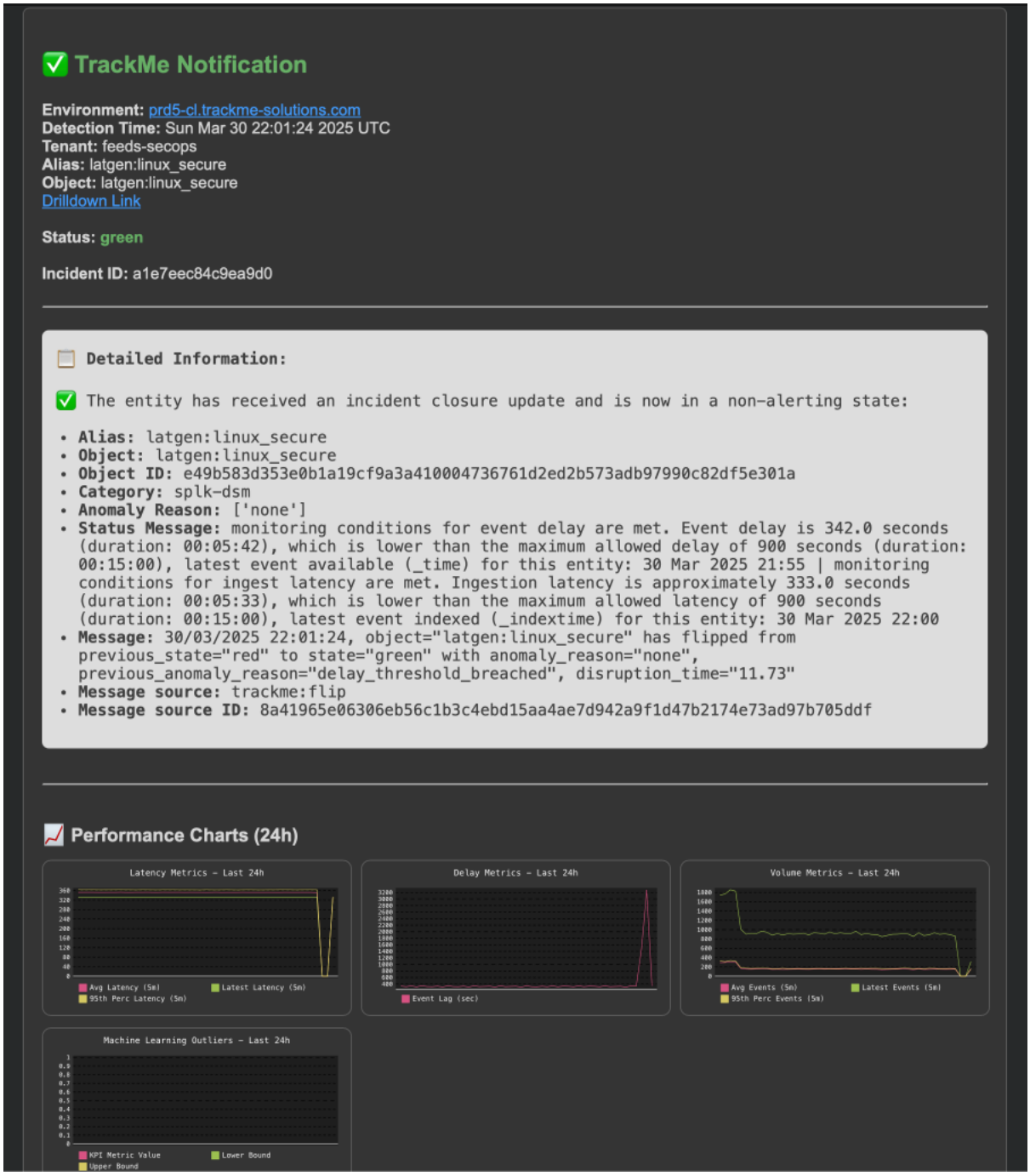
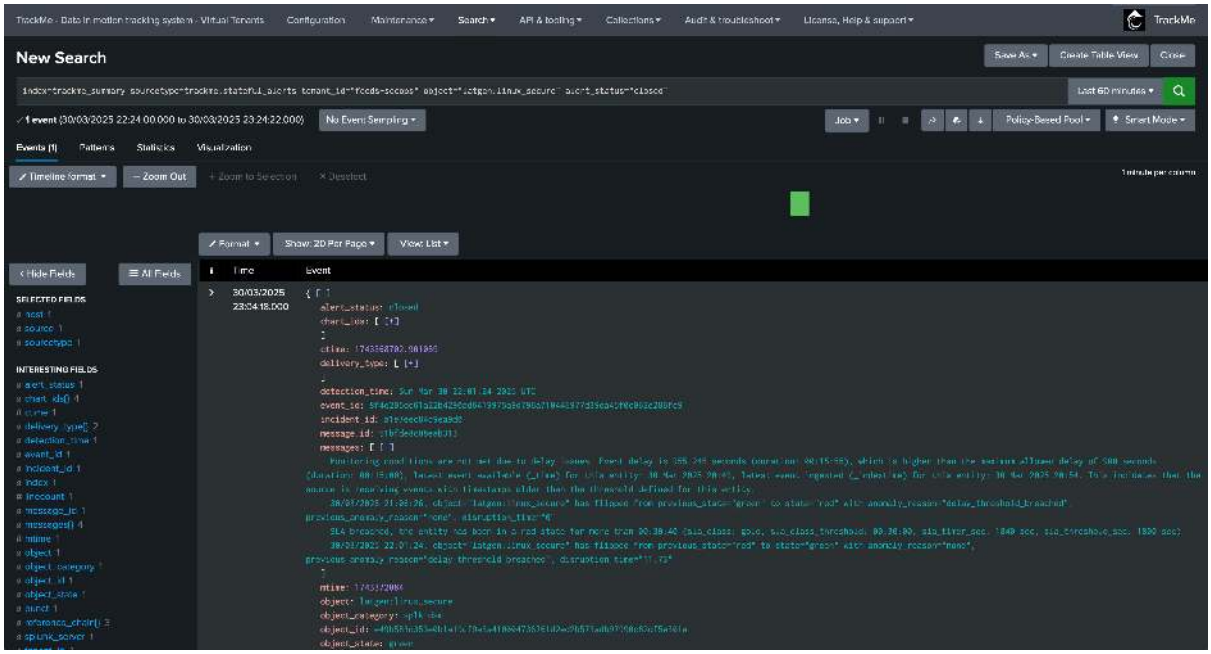
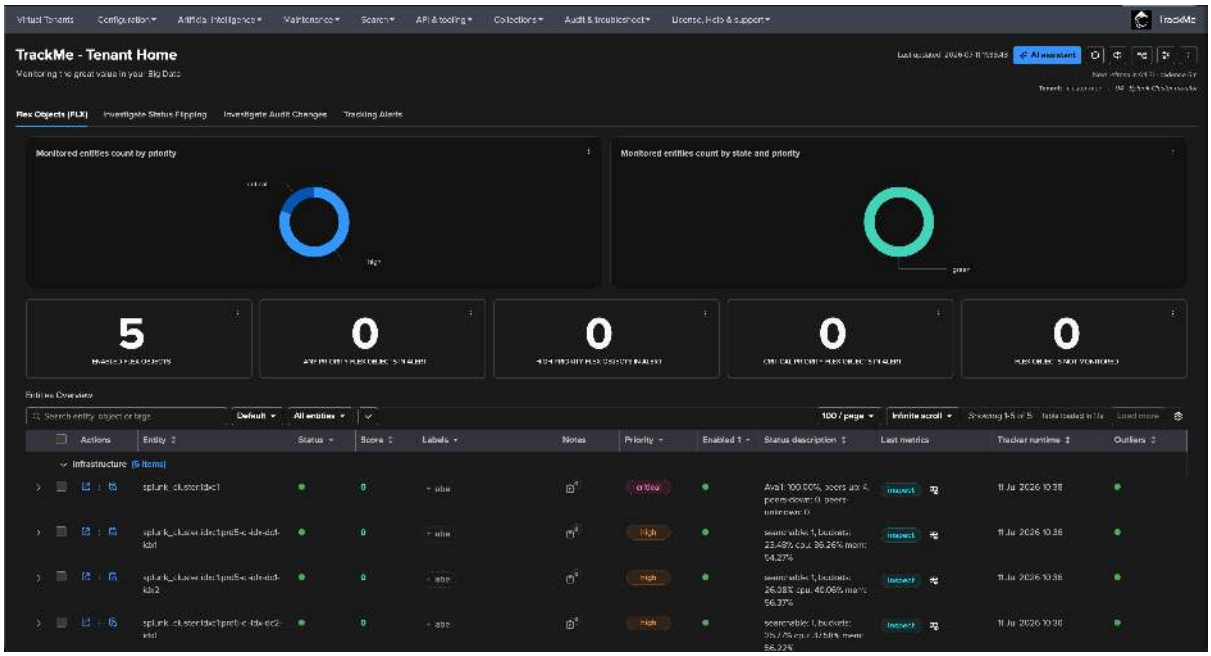
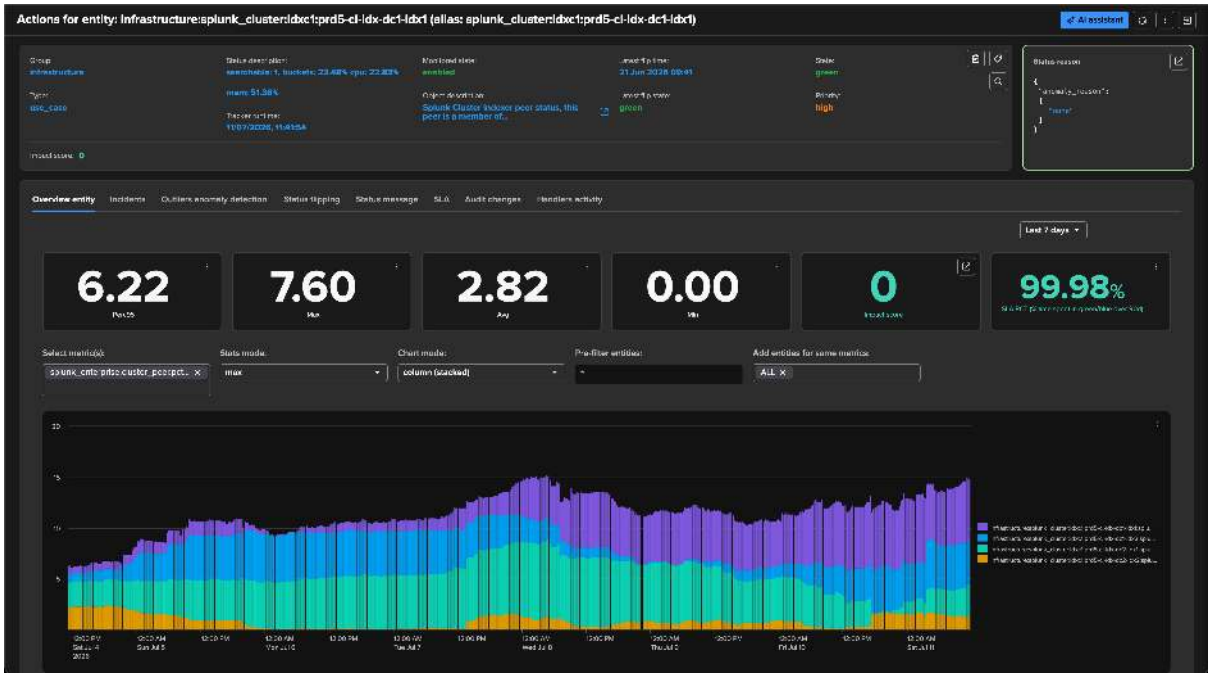
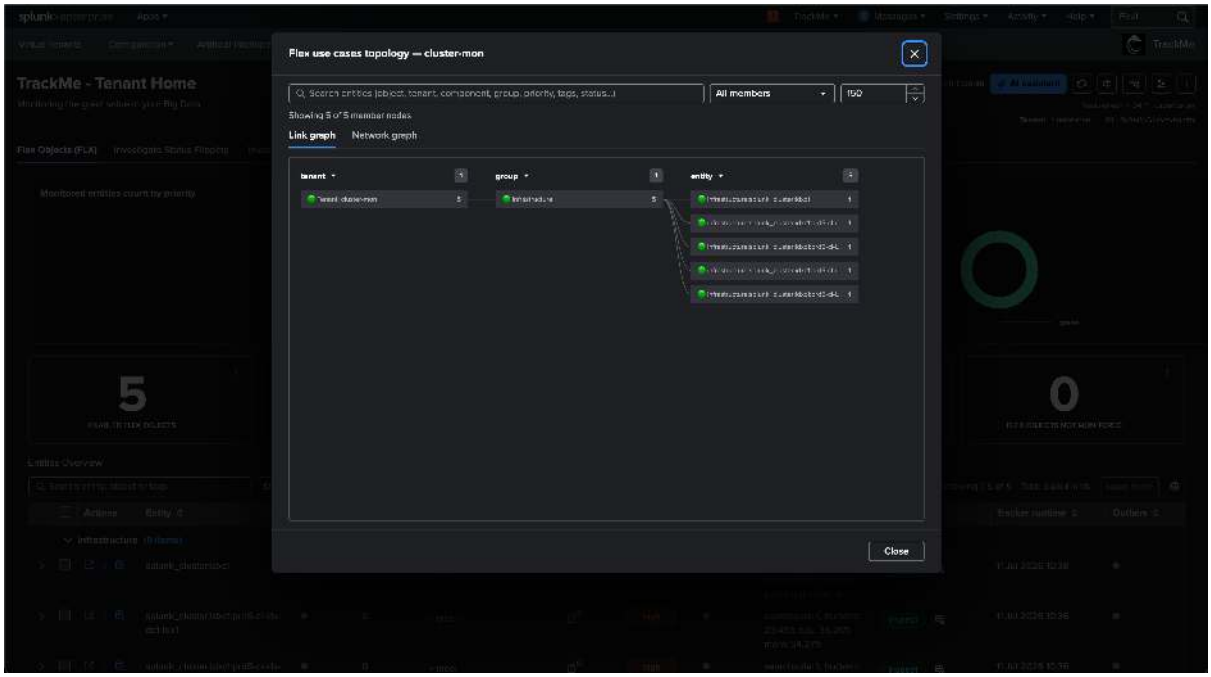
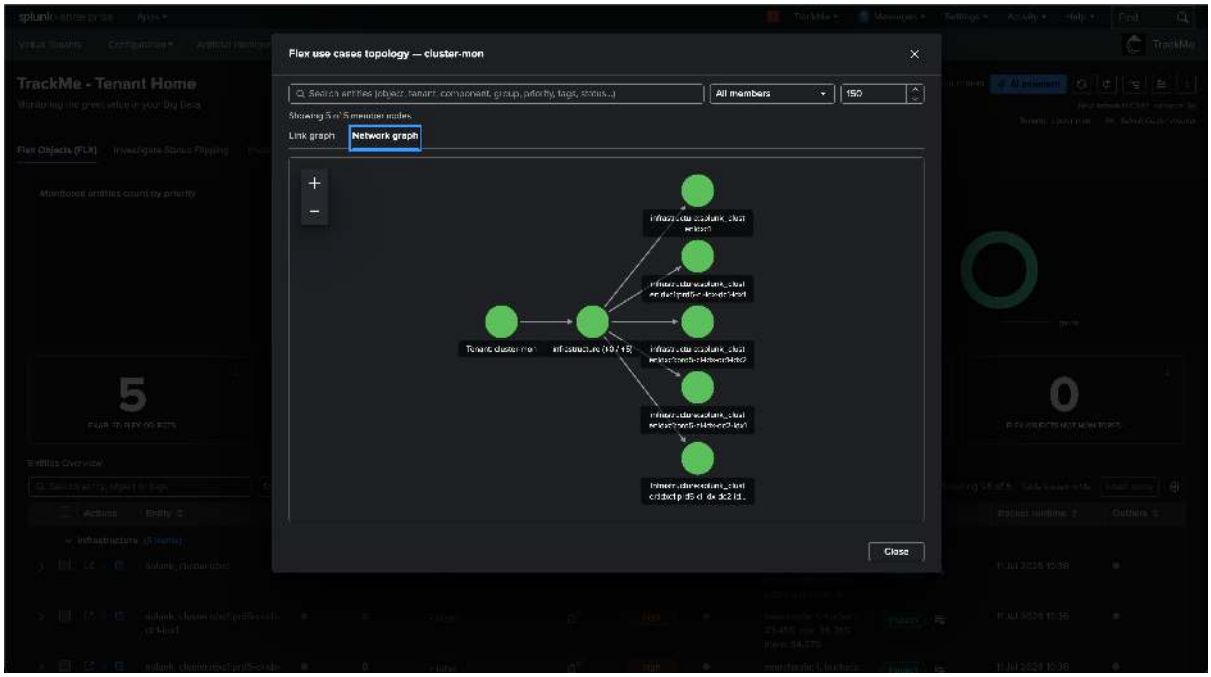
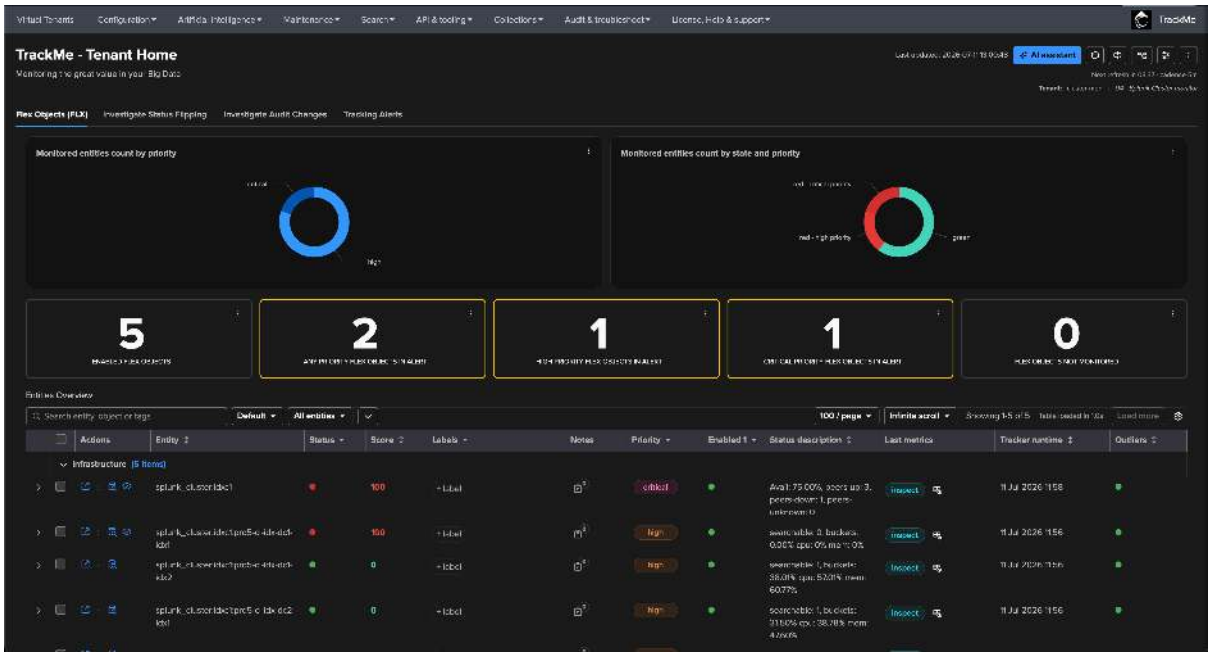
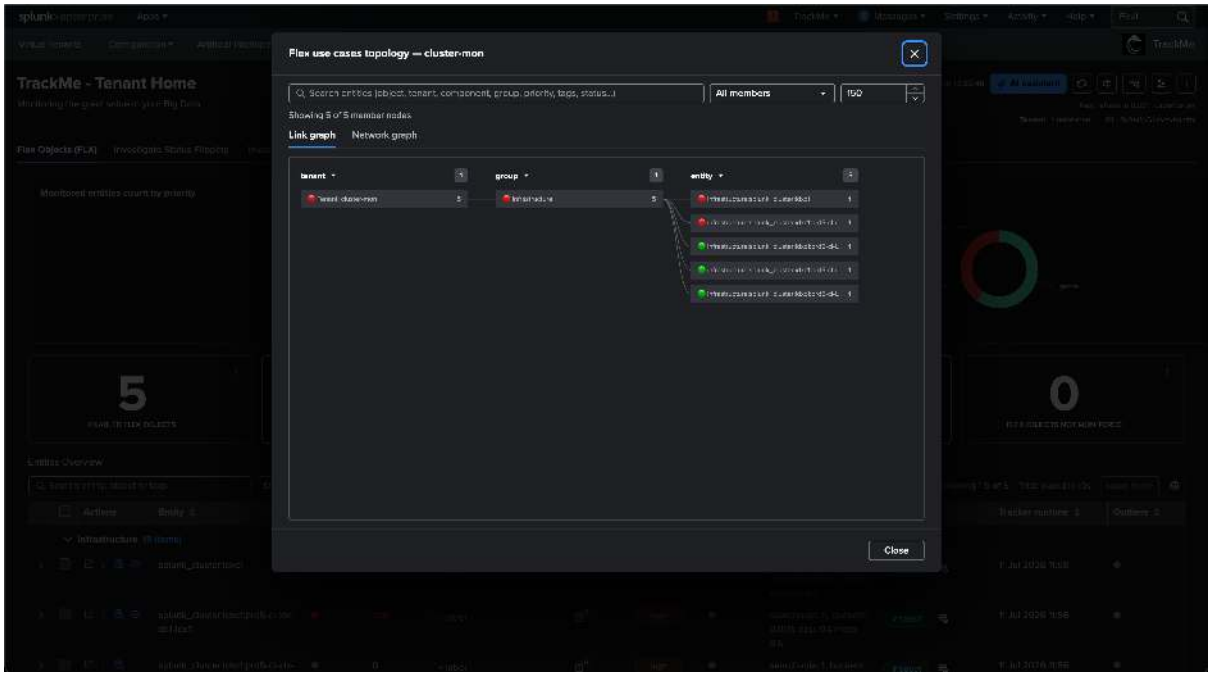
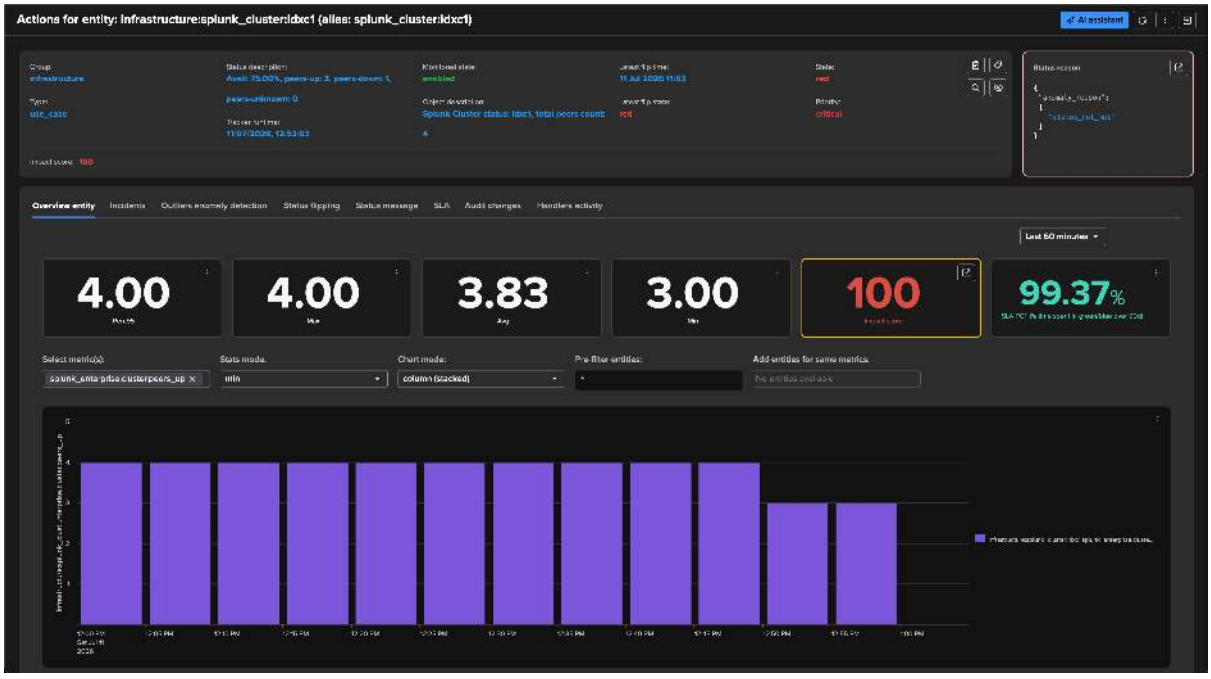
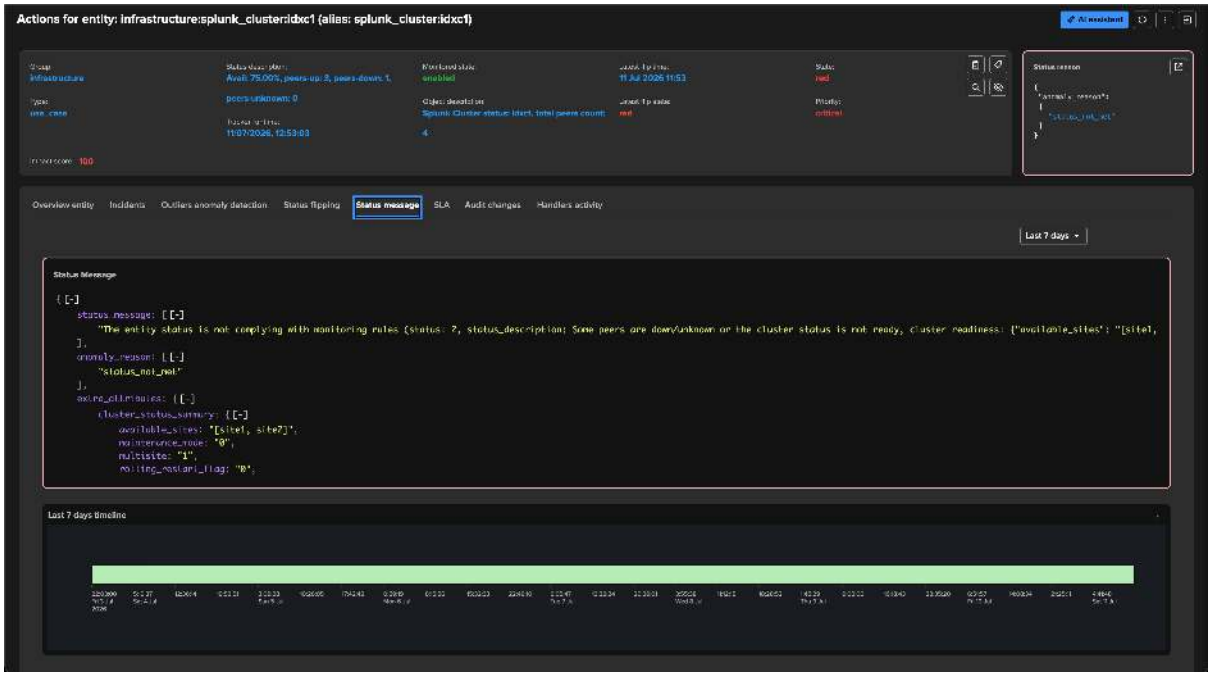
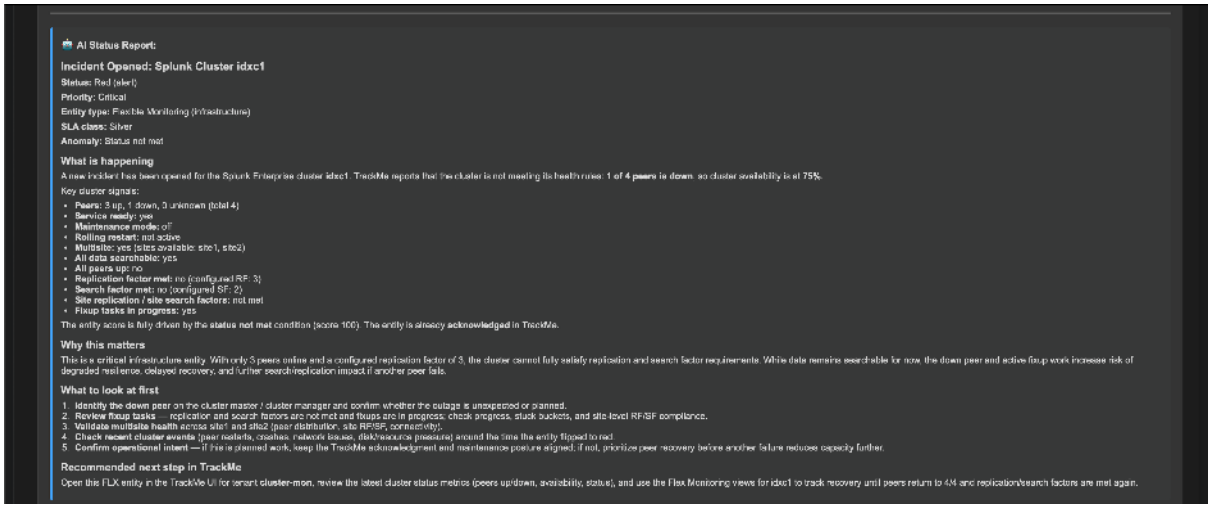
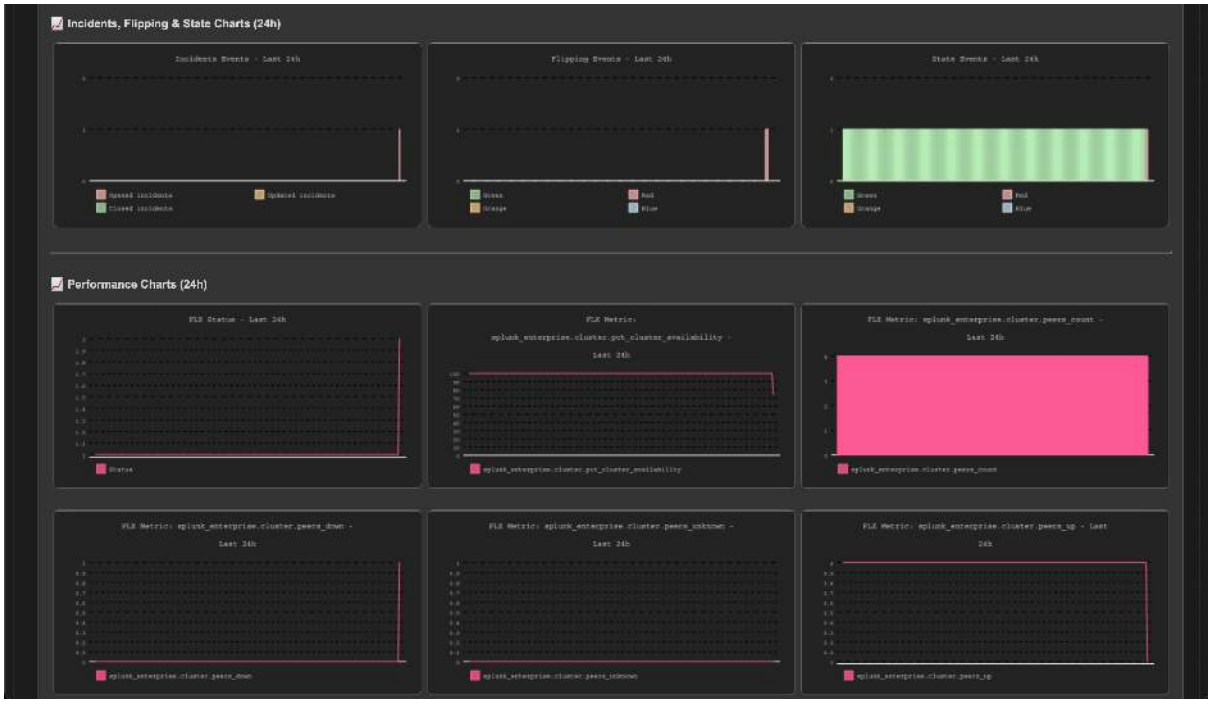
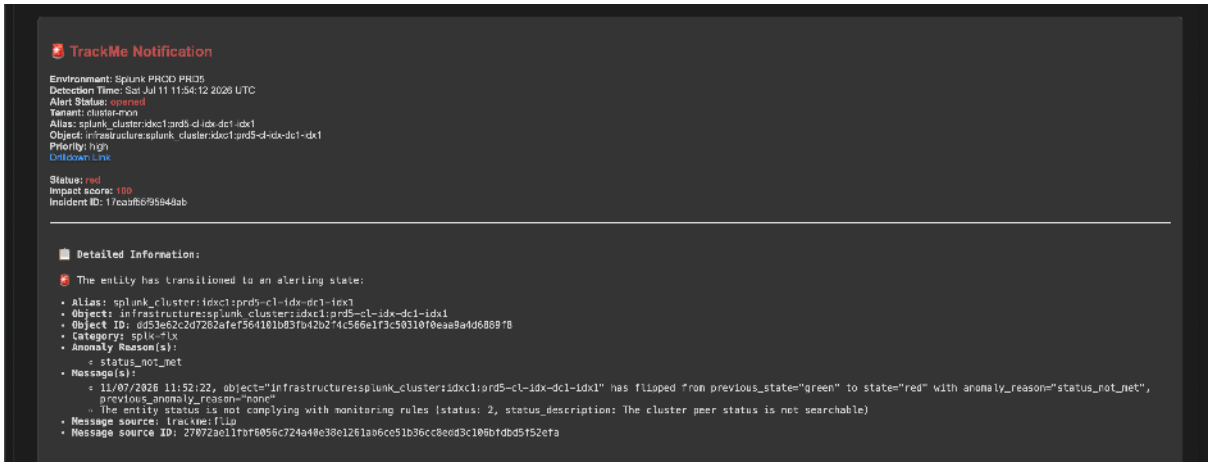
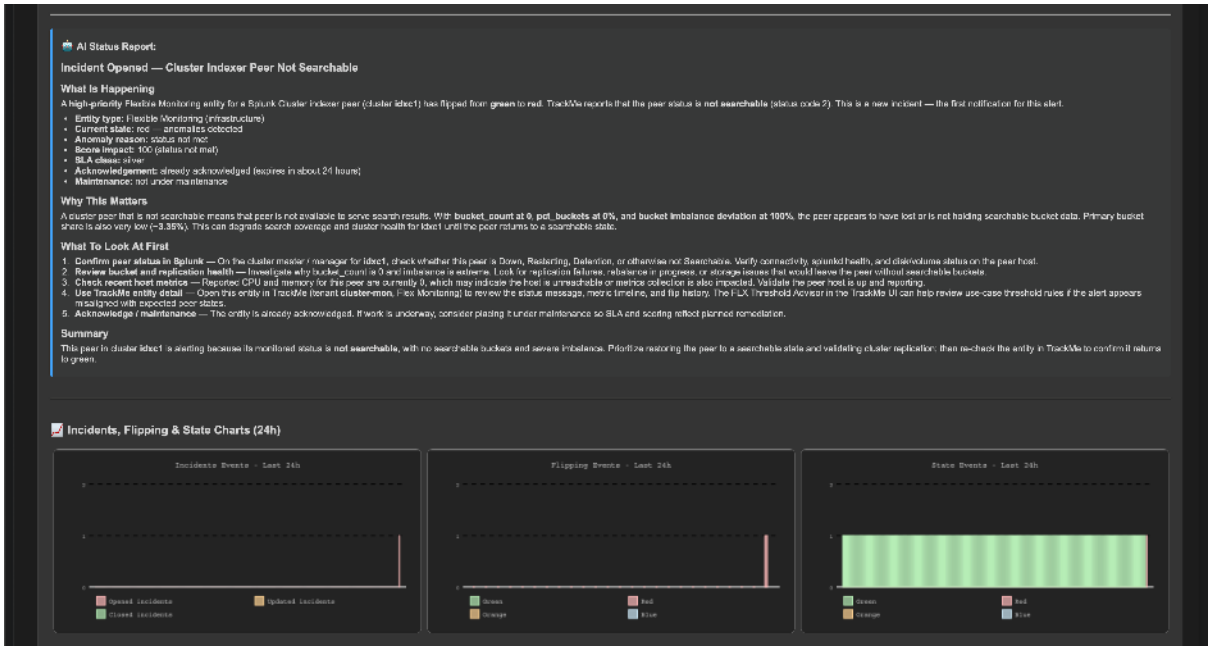
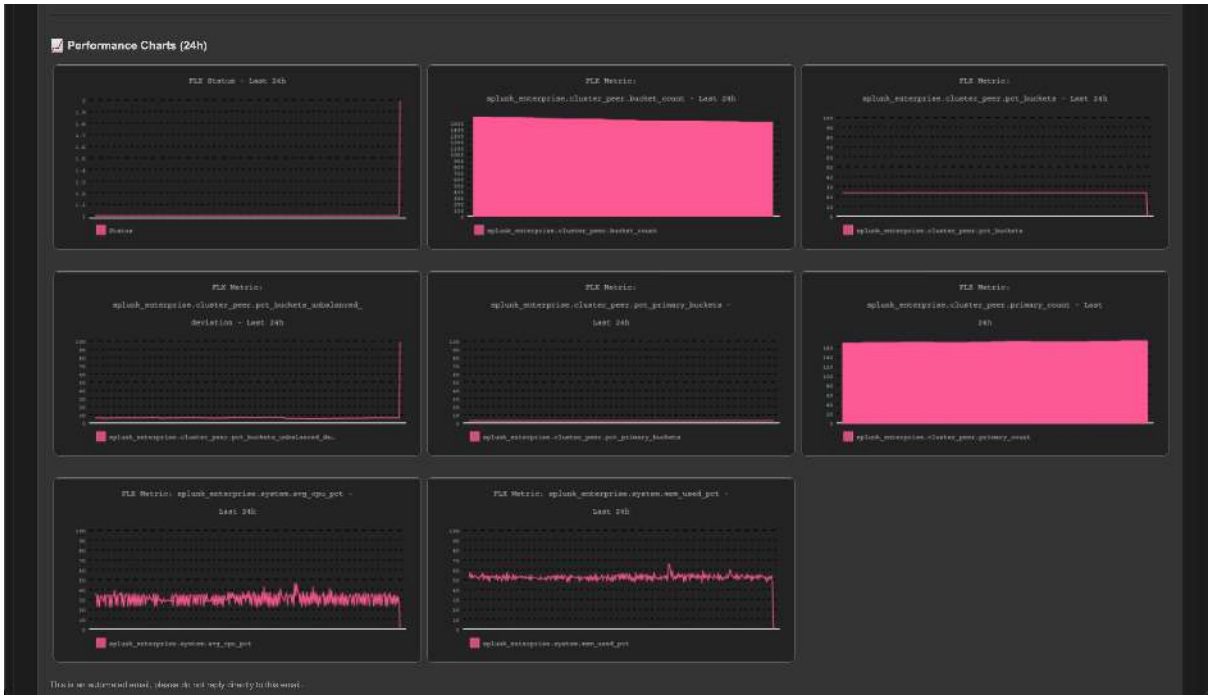
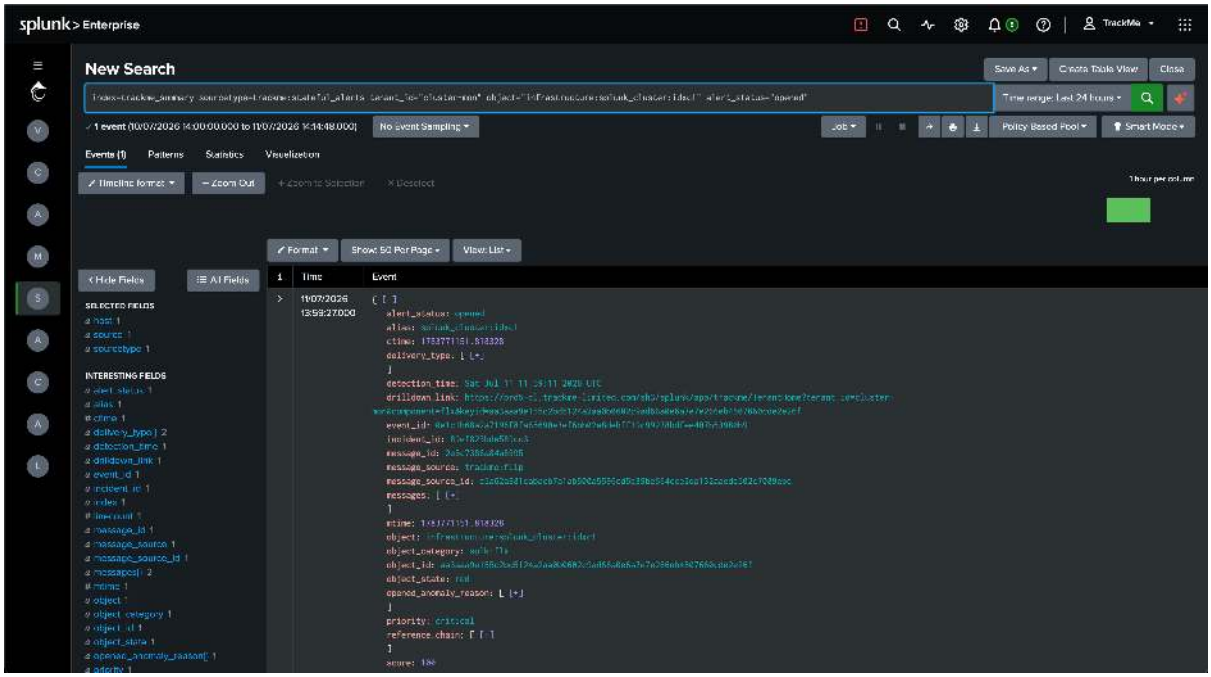
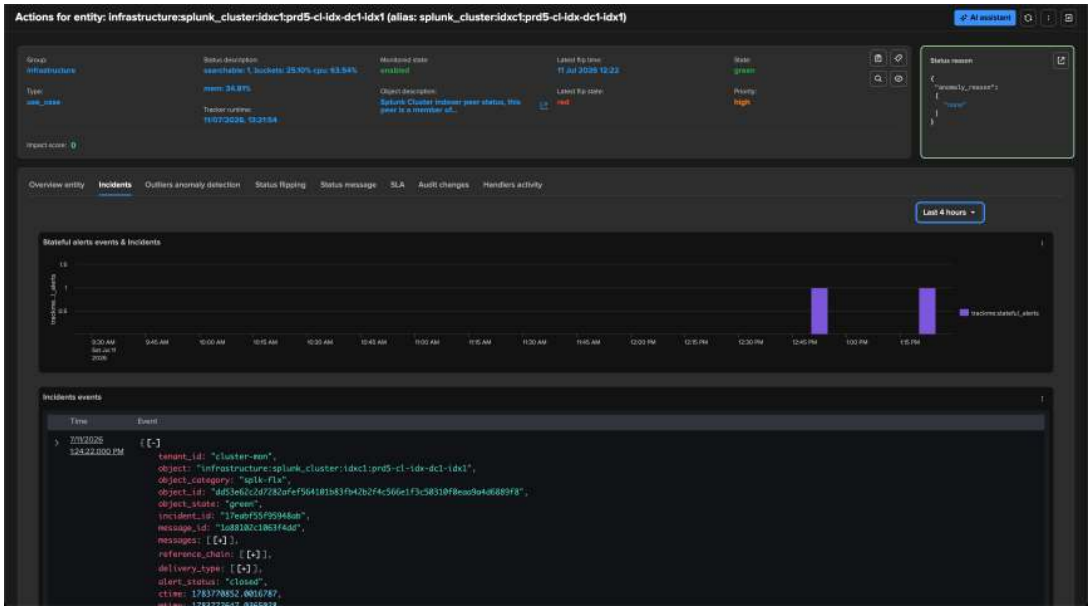
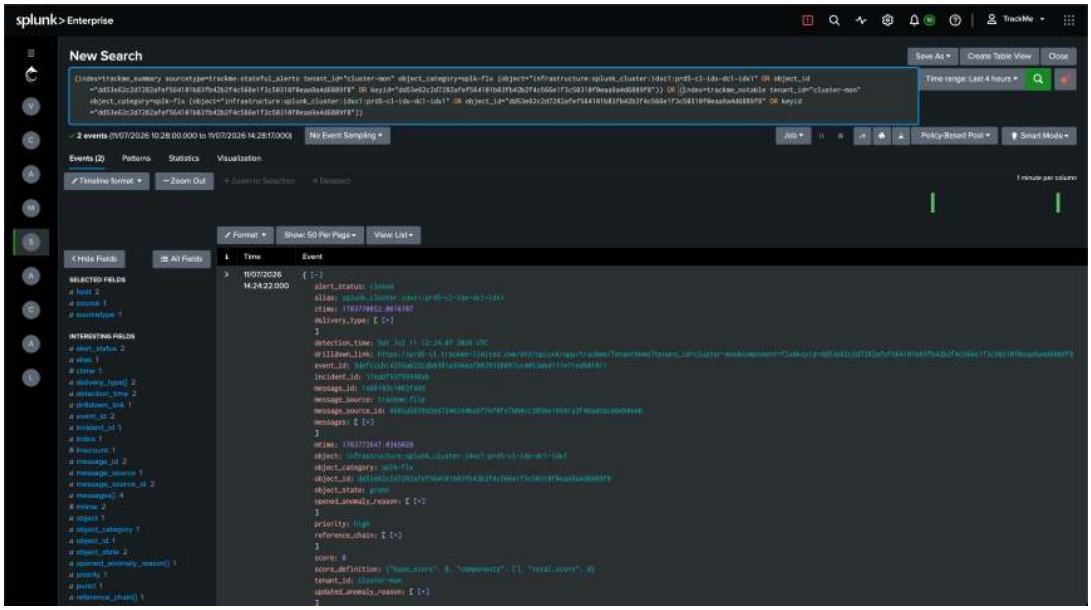
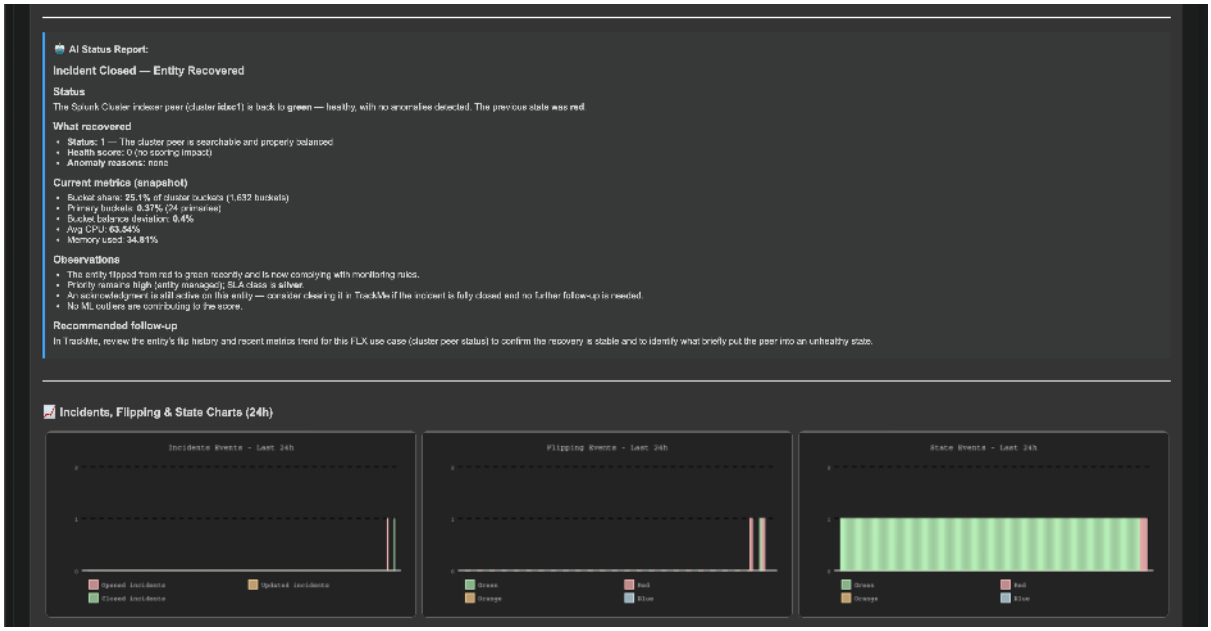
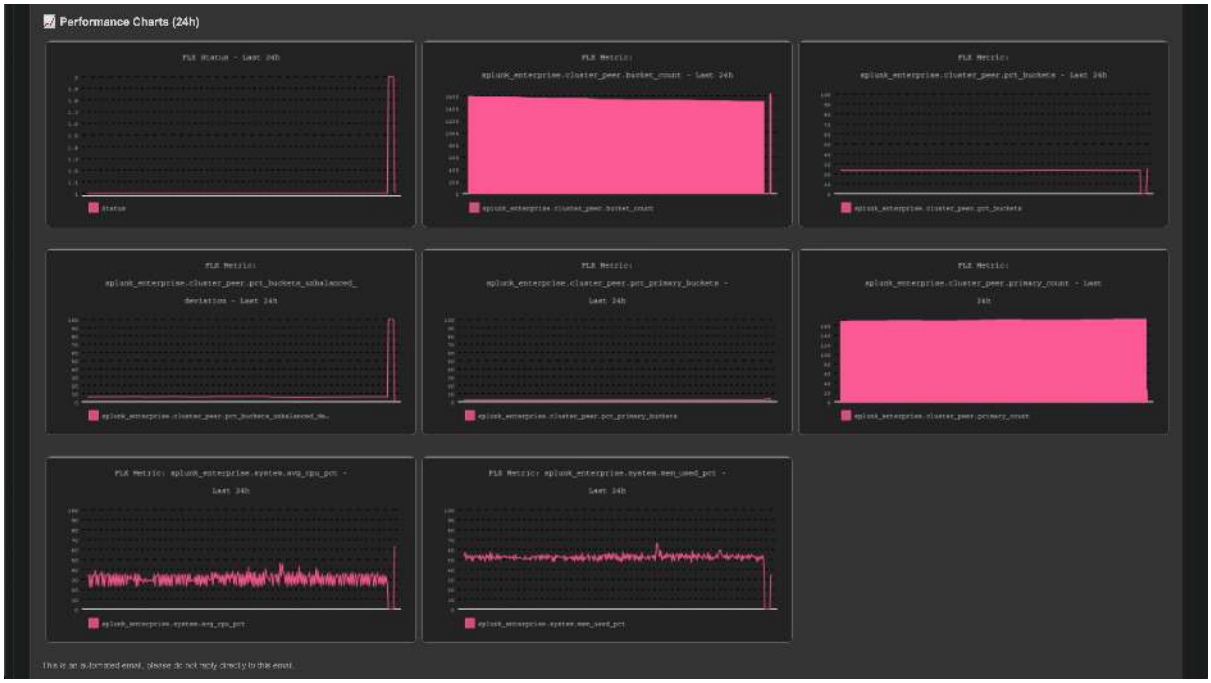
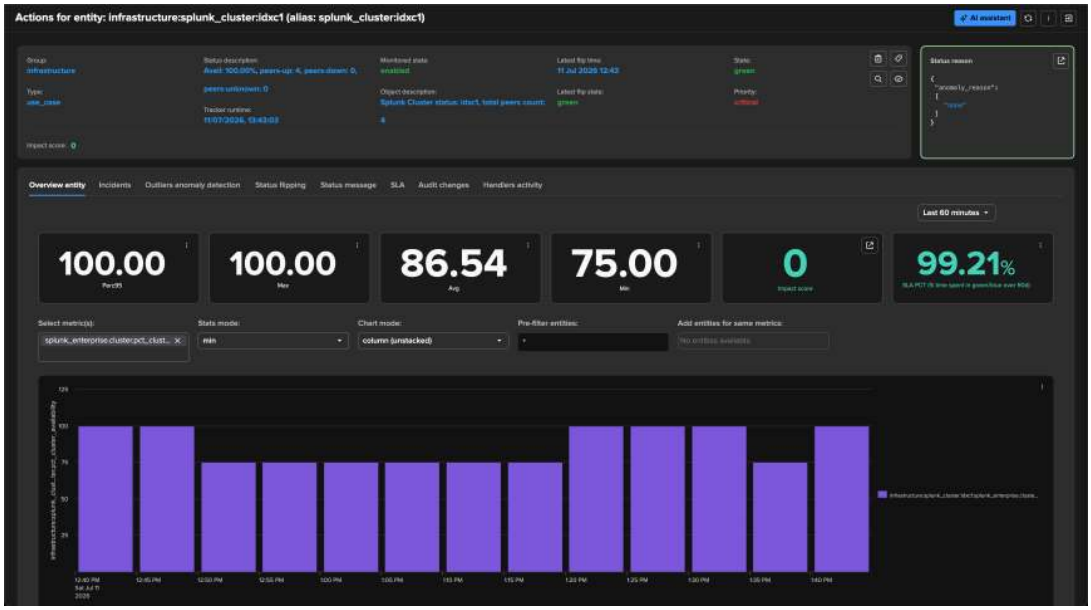
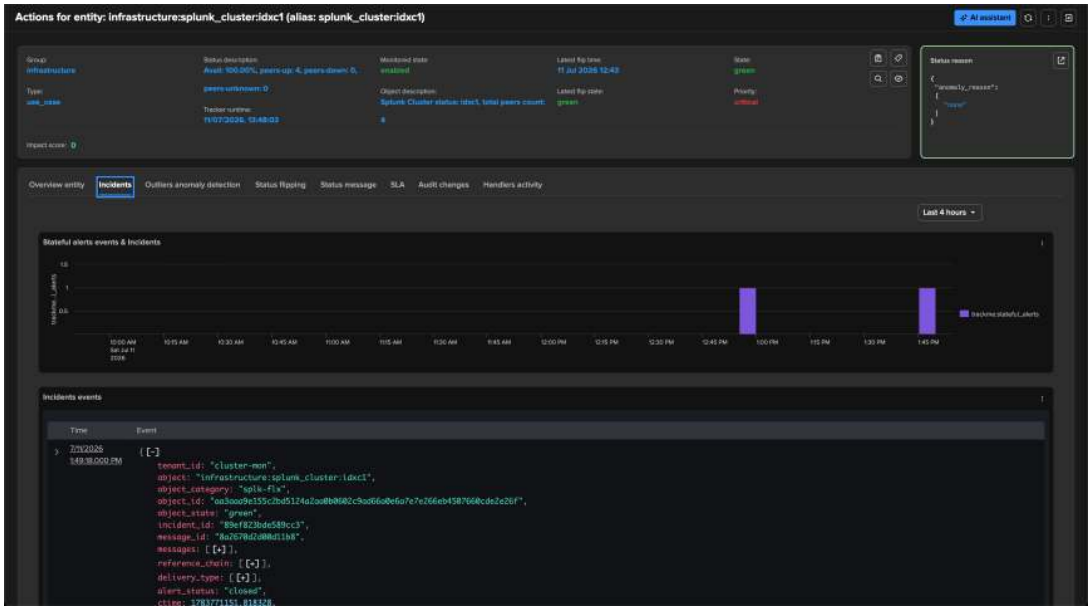
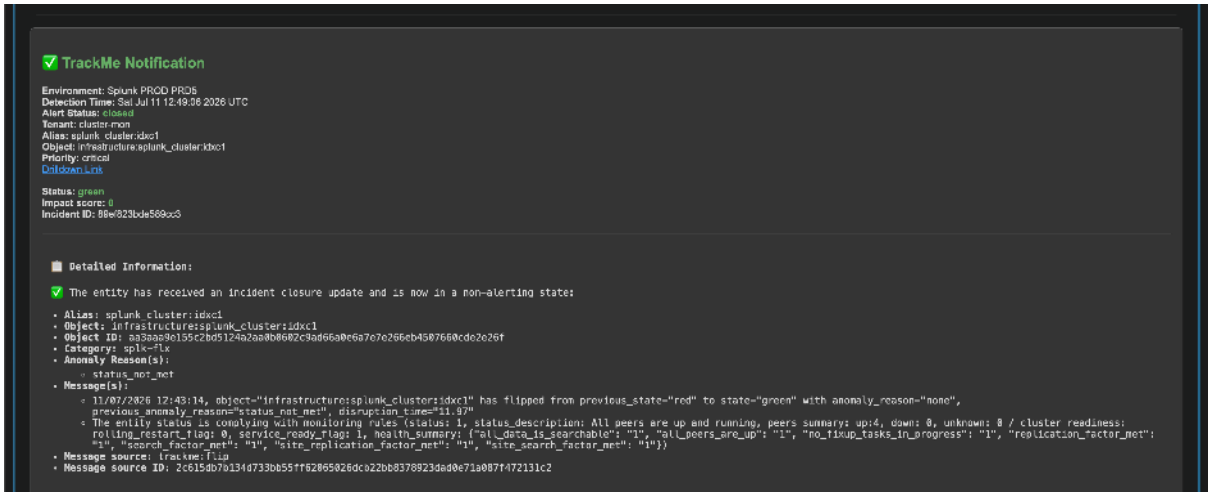
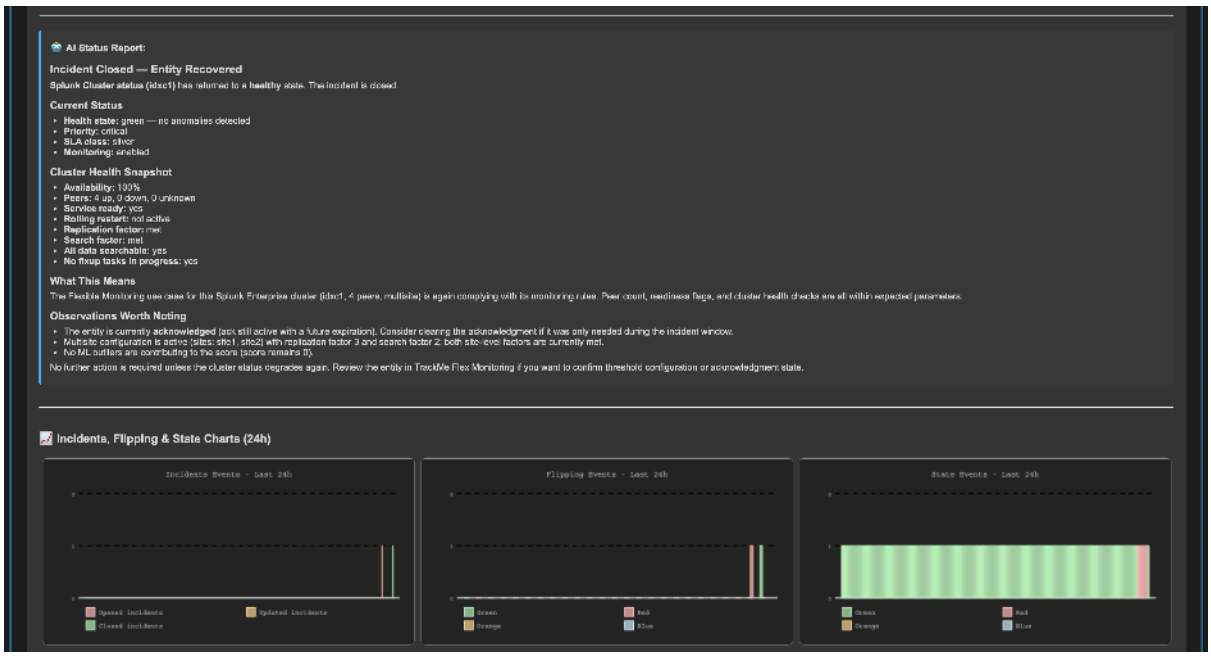
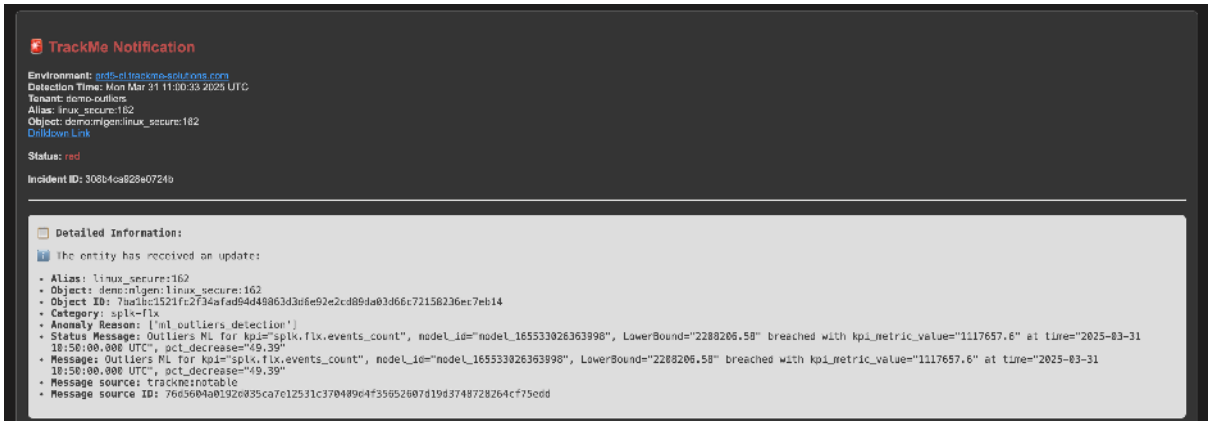
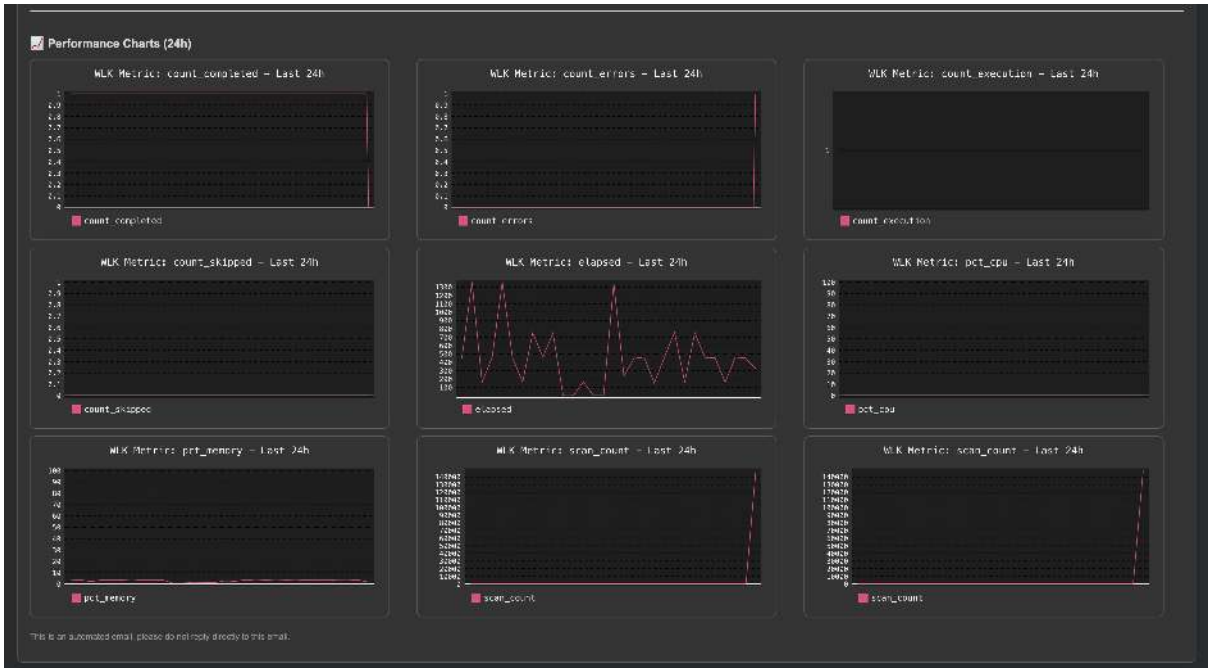
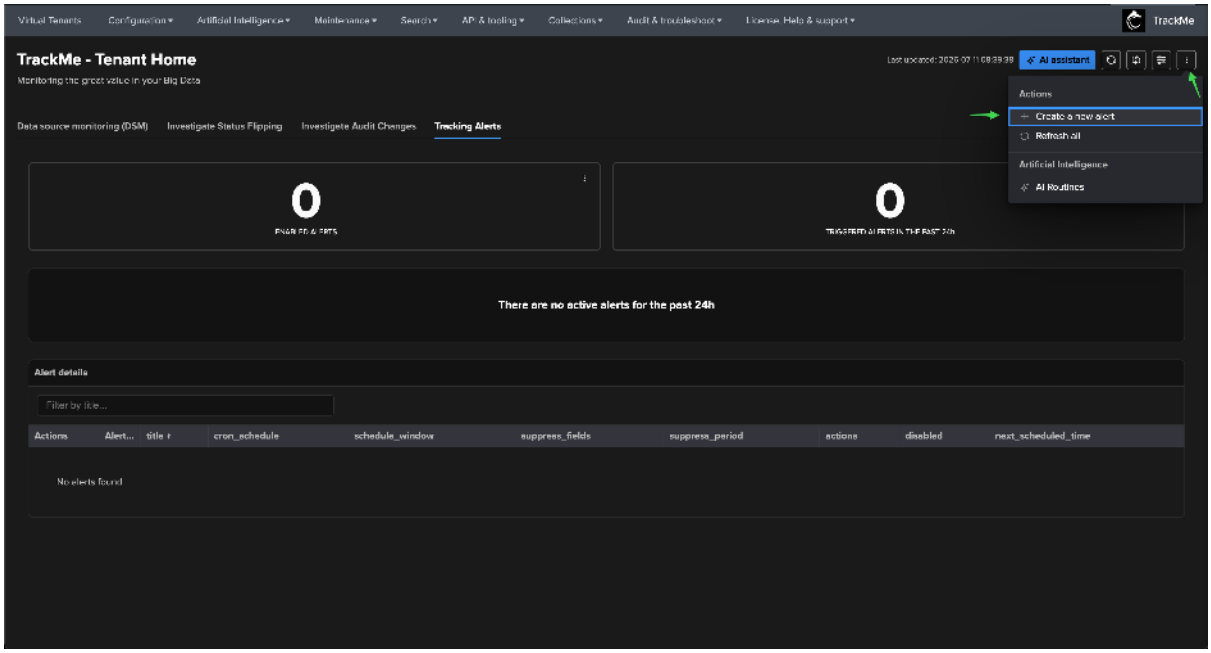
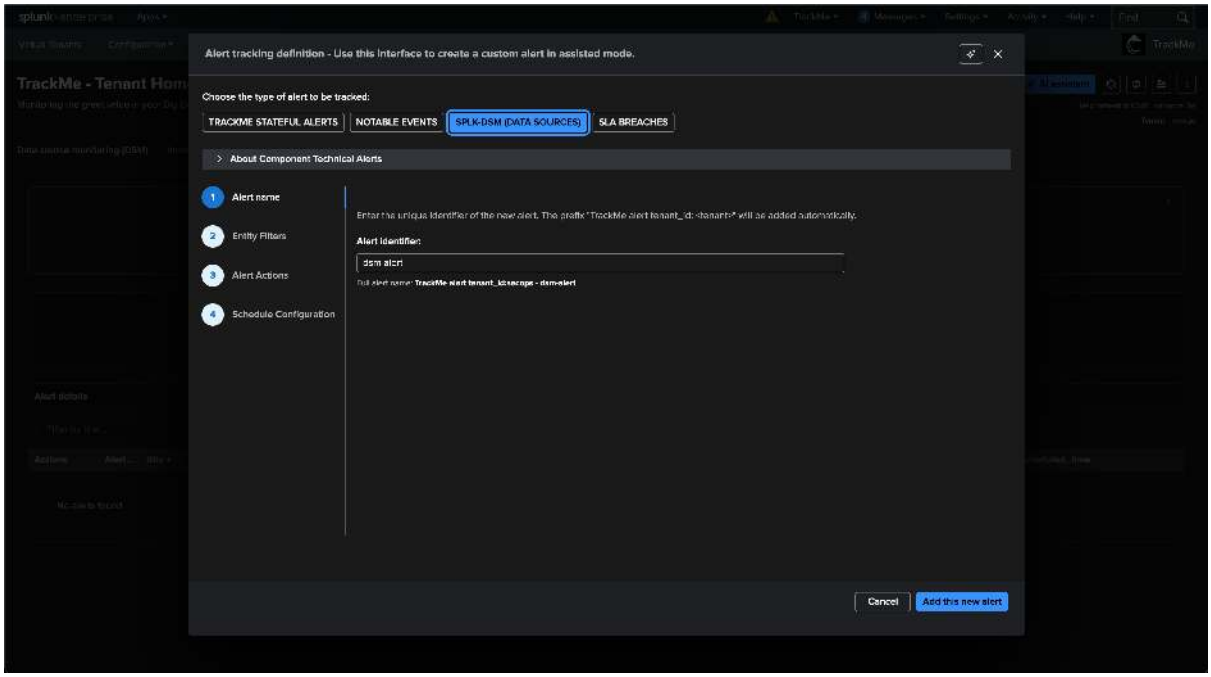
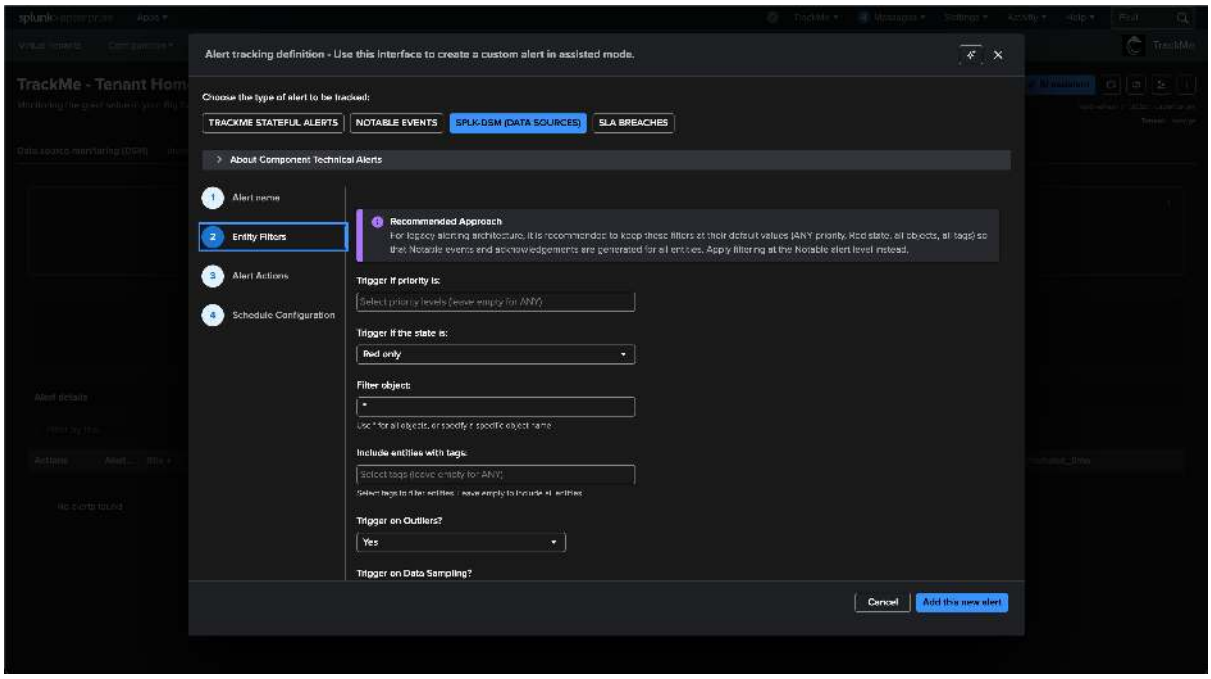
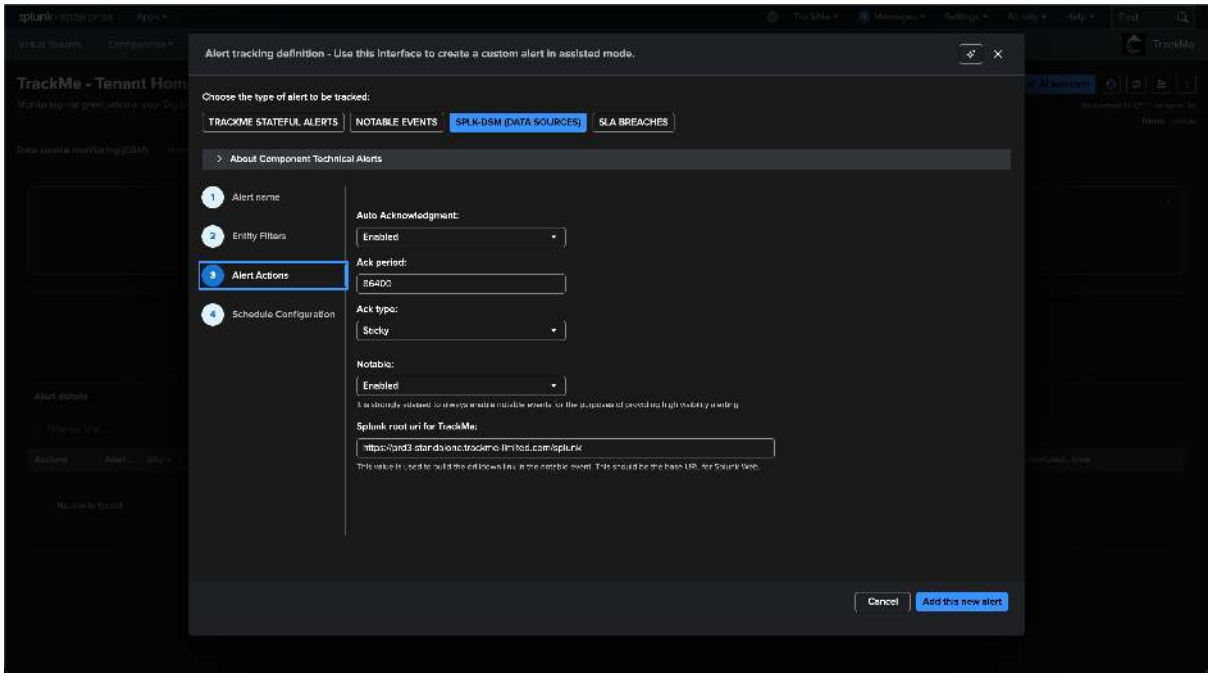
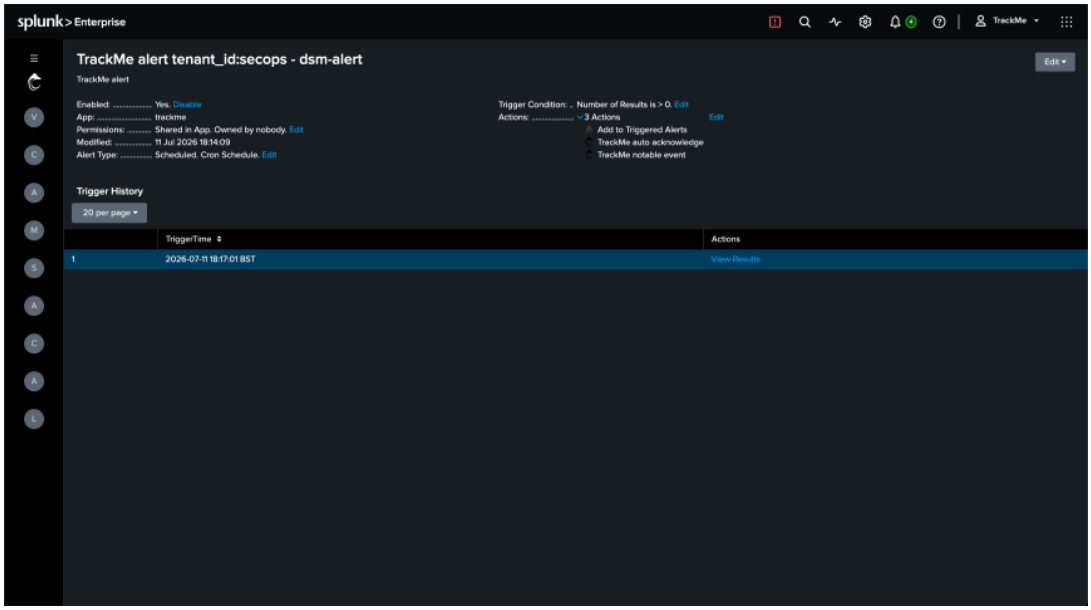
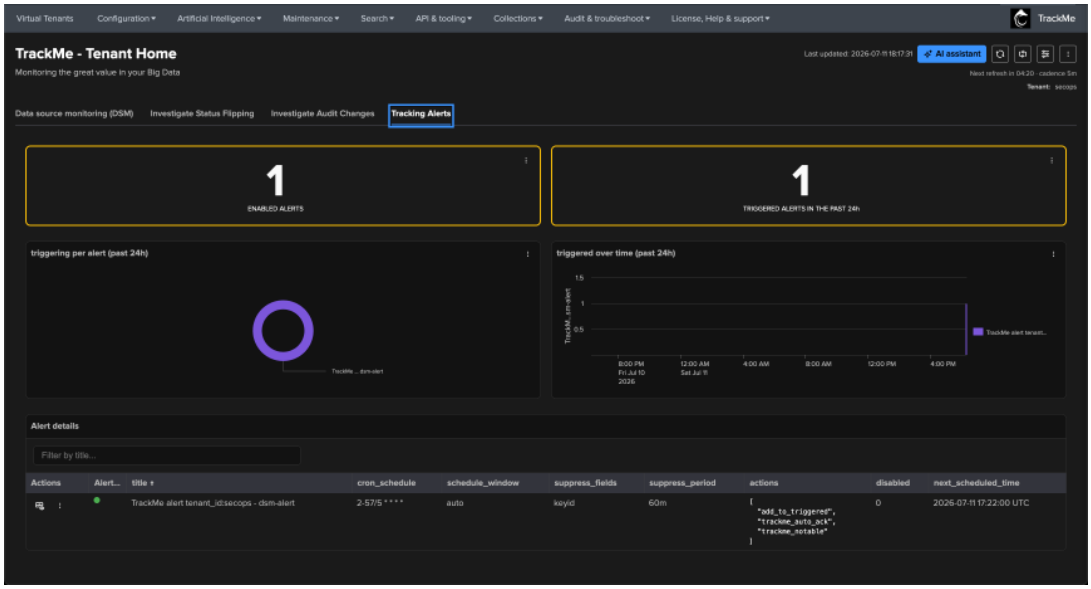
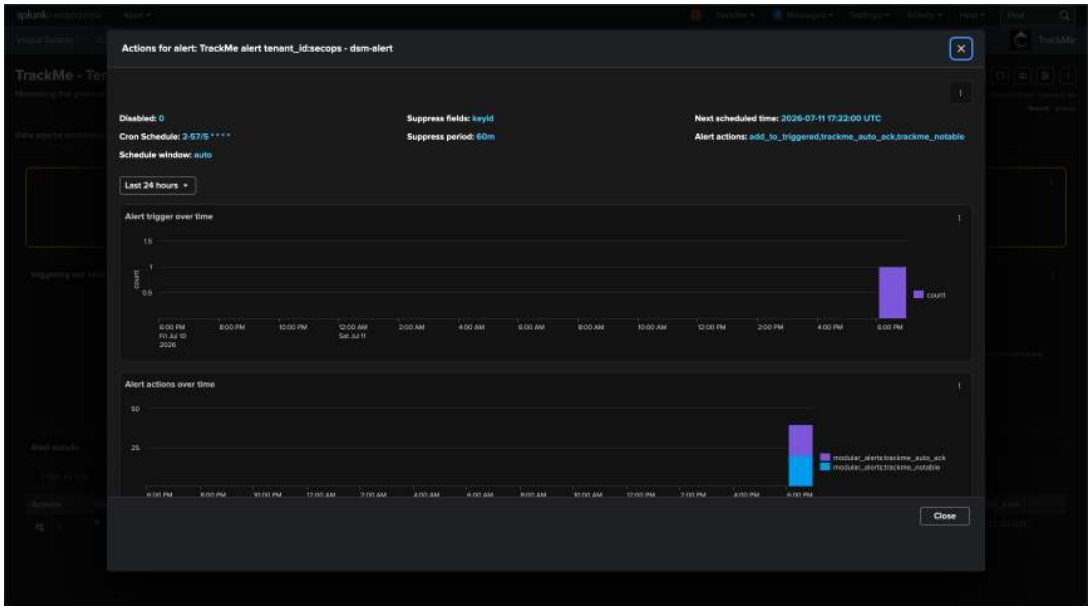
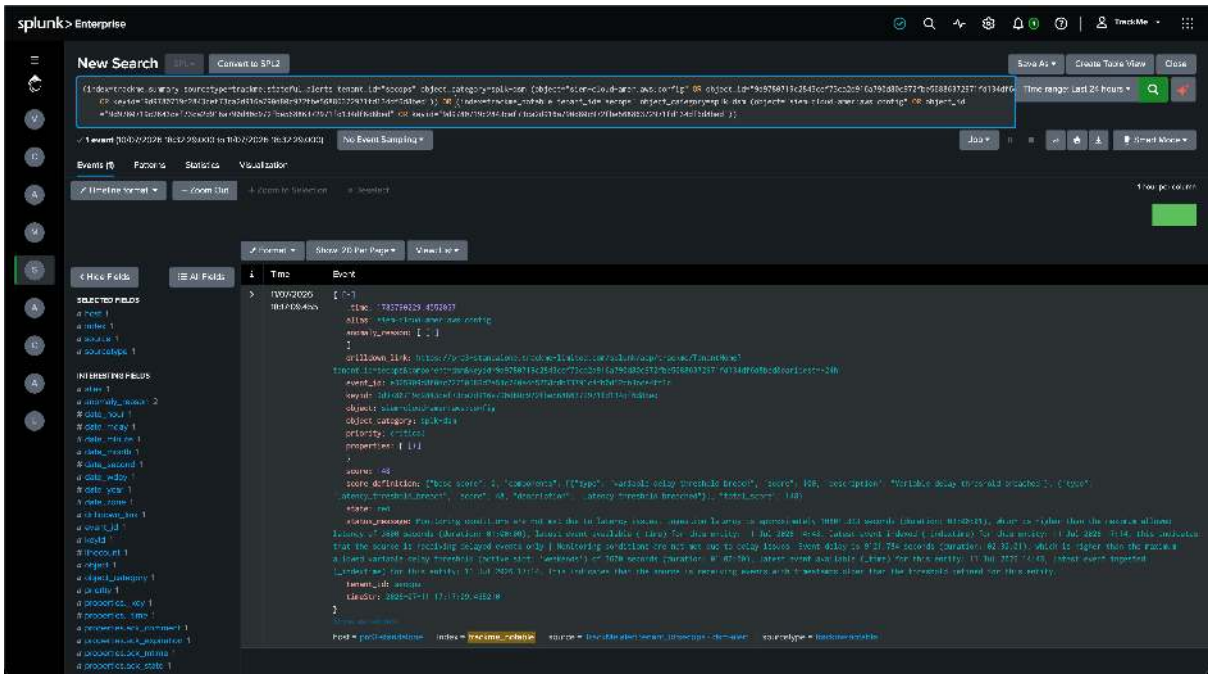
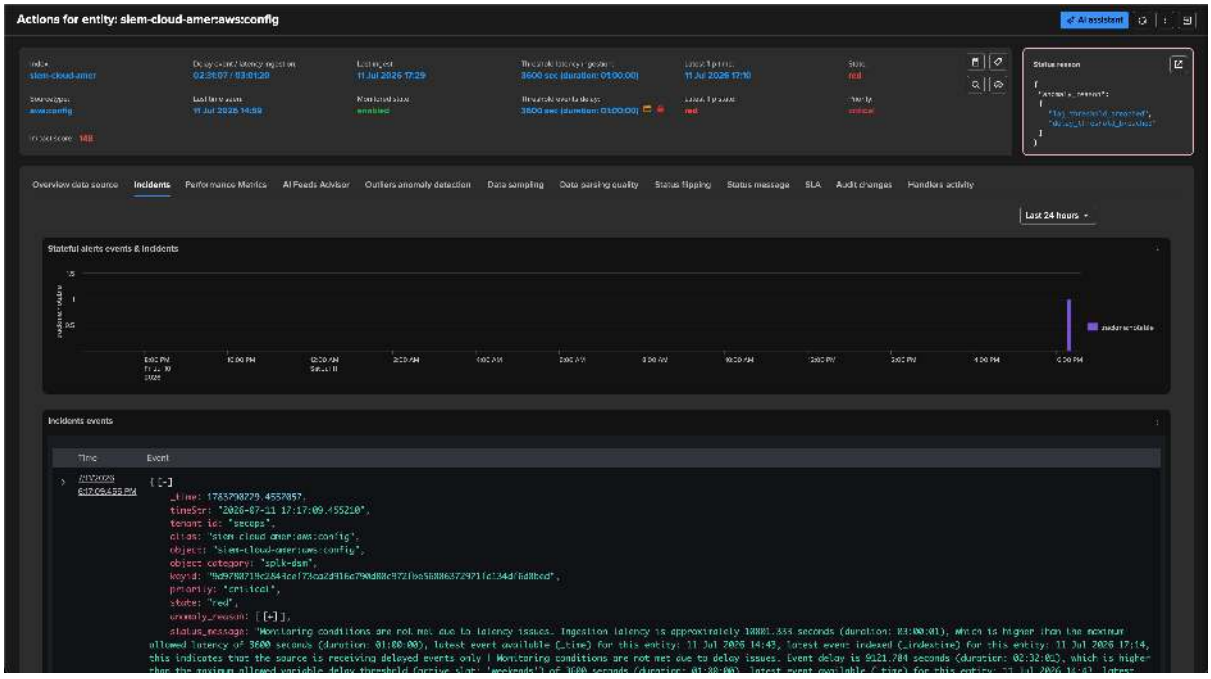
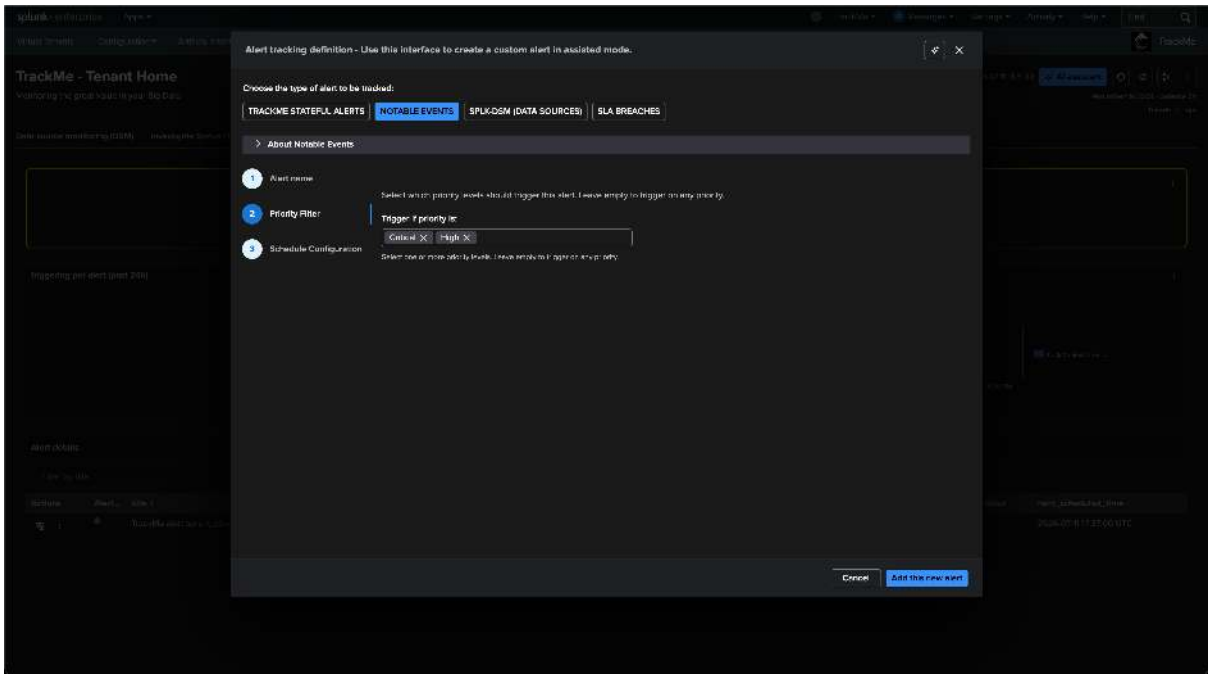
```
| inputlookup trackme_dhm_tenant_hosts-monitoring where monitored_state="enabled"
| fields object, alias
``` extract the pair identifier from the alias ```
| rex field=alias "(?<grp_prefix>.*)-(?<grp_suffix>ha\d{1,})-"
| eval object_group_name = grp_prefix . ":" . grp_suffix
| where isnotnull(object_group_name) AND object_group_name!=""
``` group members and keep only real groups (>1 member) ```
| stats values(object) as object_group_members by object_group_name
| eventstats dc(object_group_members) as dcount_members by object_group_name
| where dcount_members>1
``` create / update / prune the groups ```
| trackmeautogroup tenant_id="hosts-monitoring" purge_single_member_grp=True
```

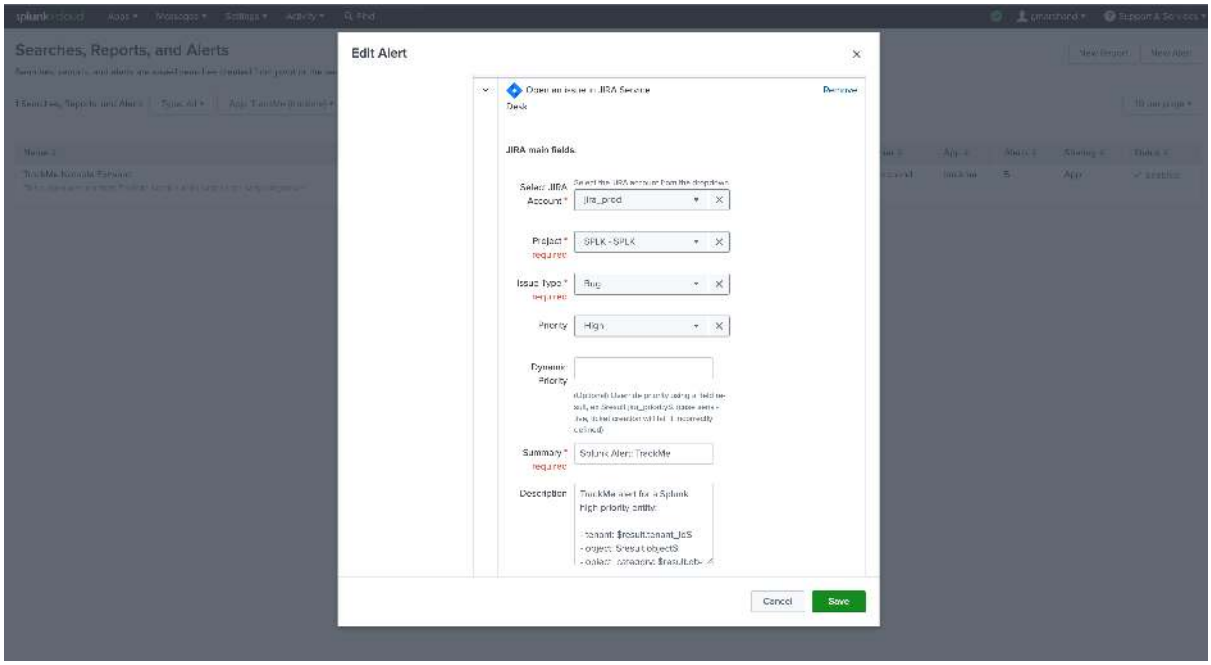
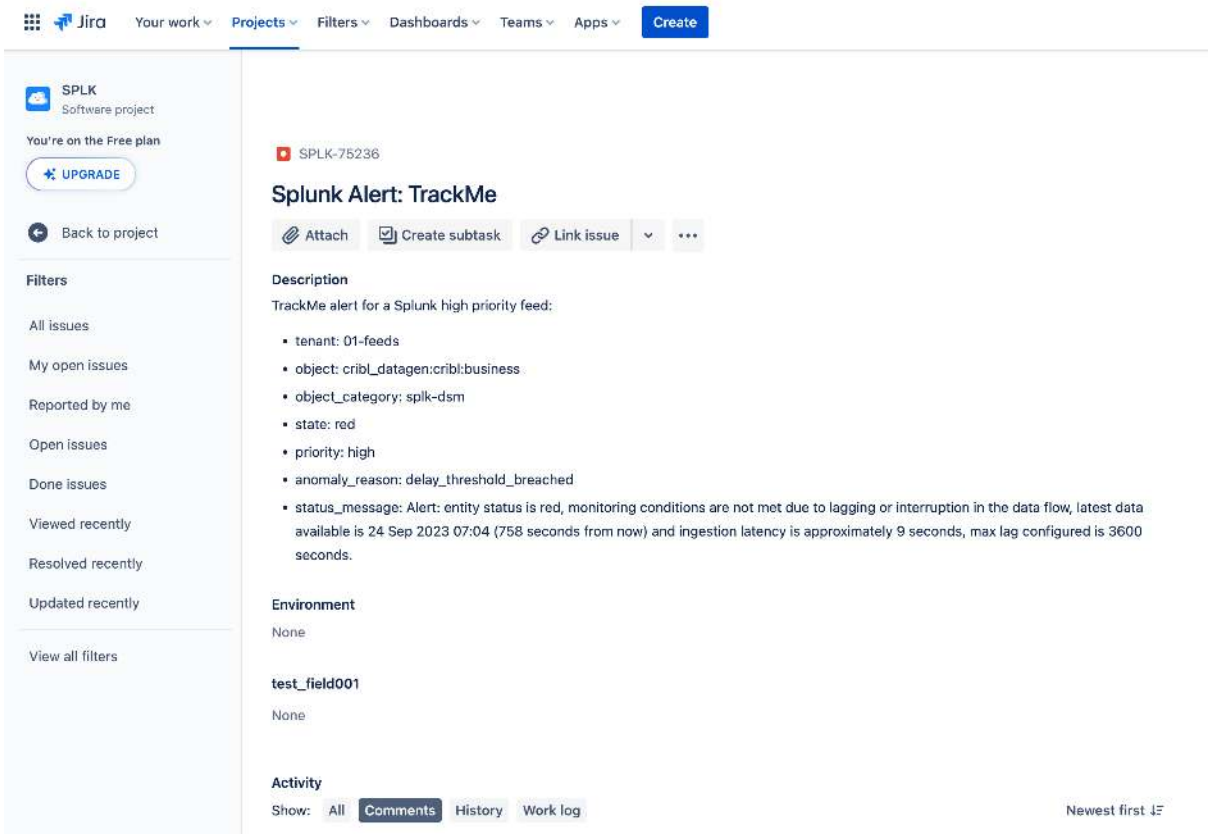
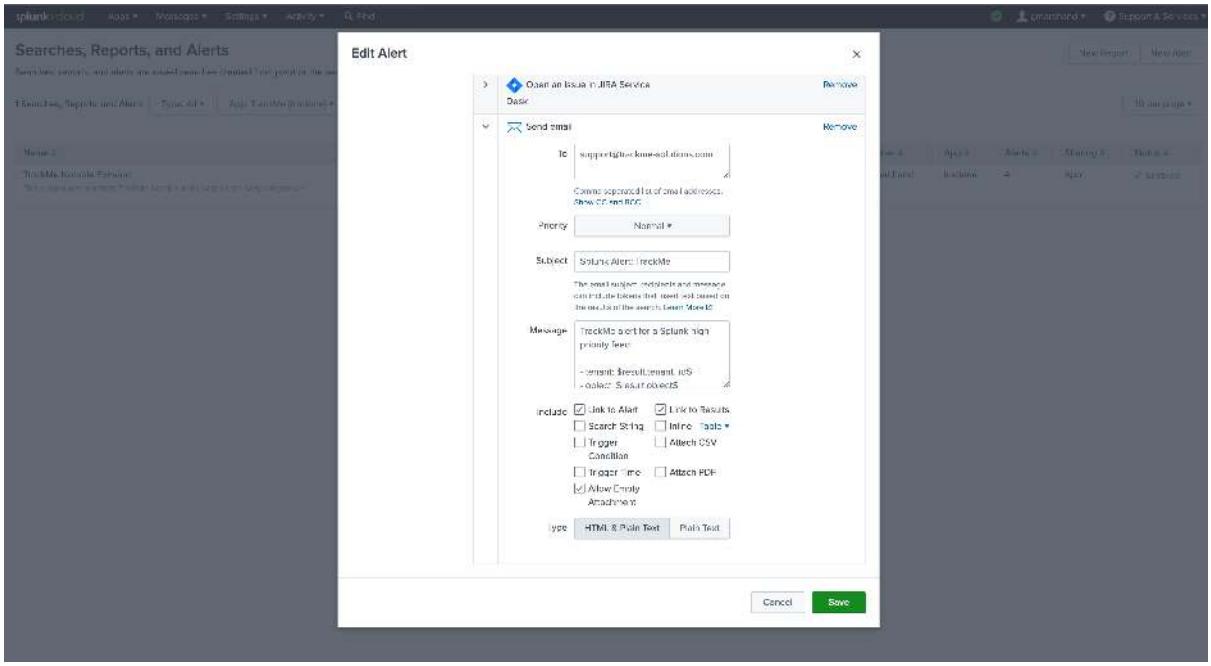
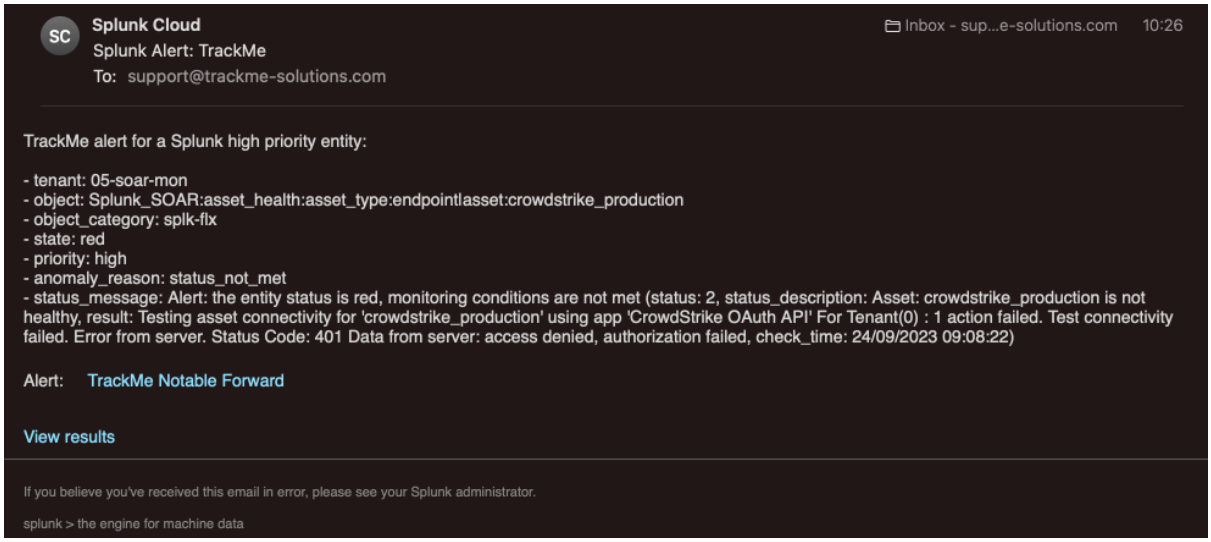
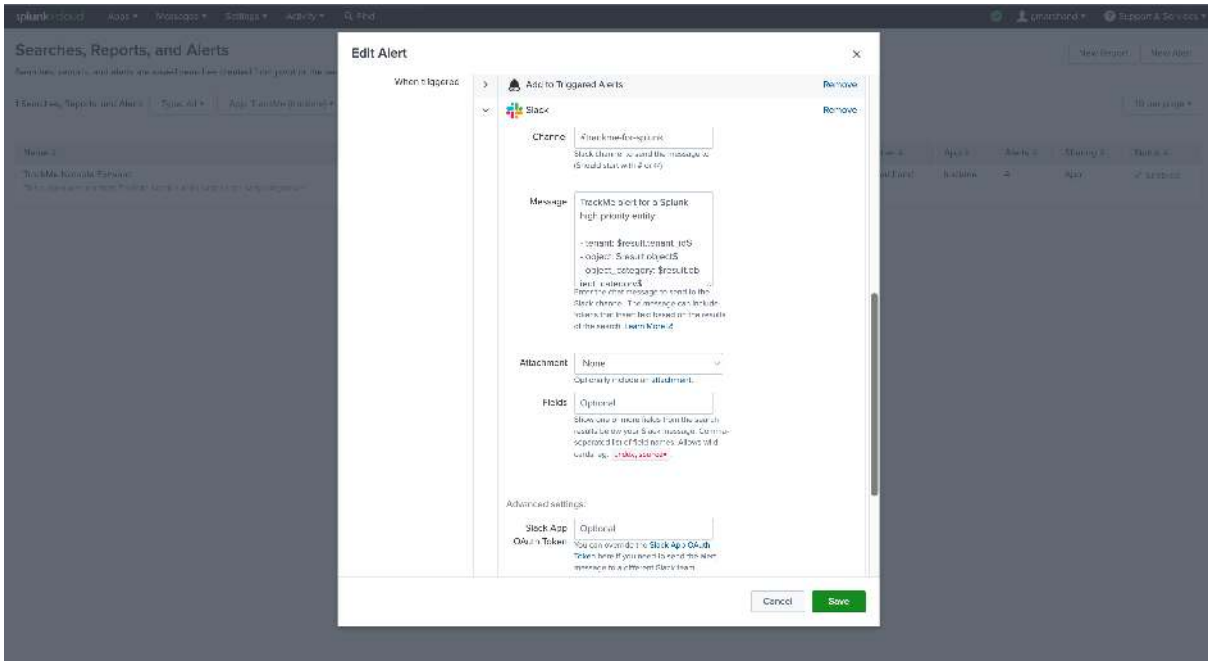
Example — by CMDB lookup:

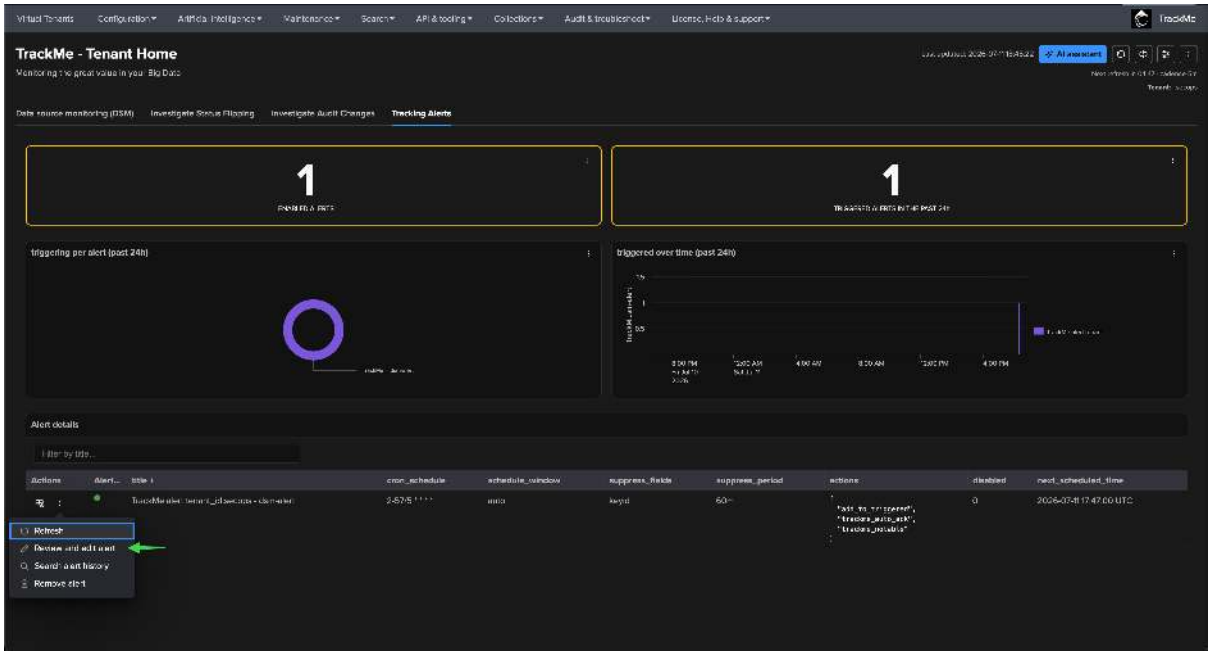
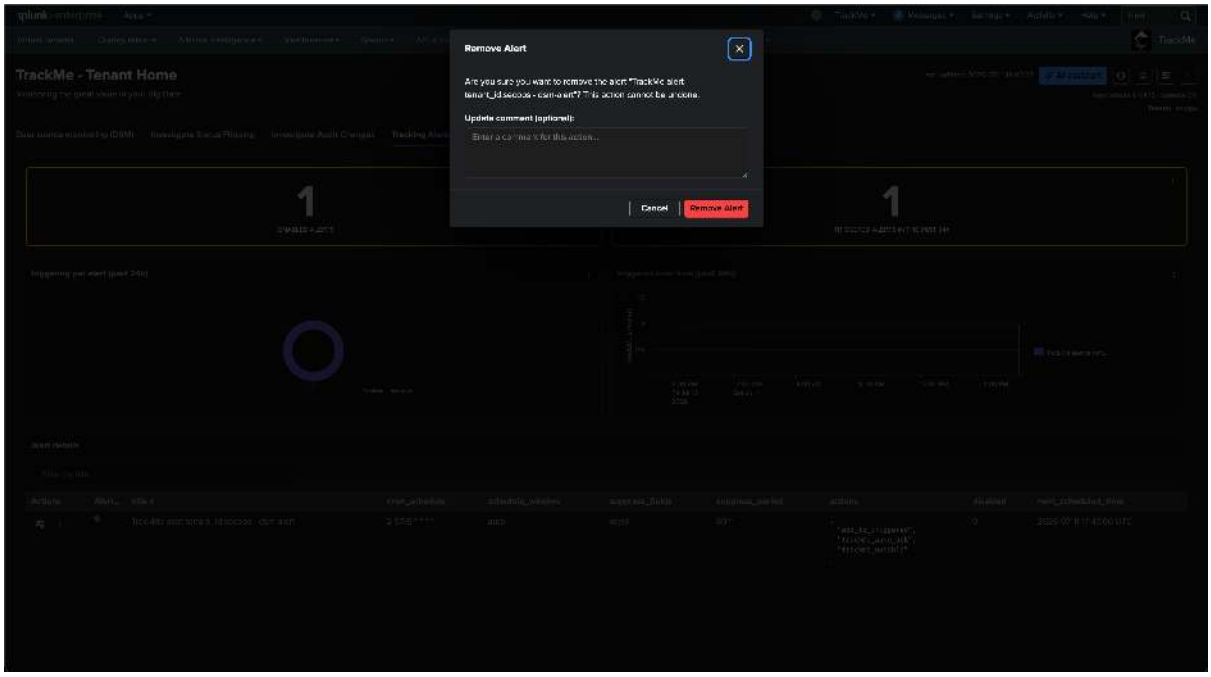
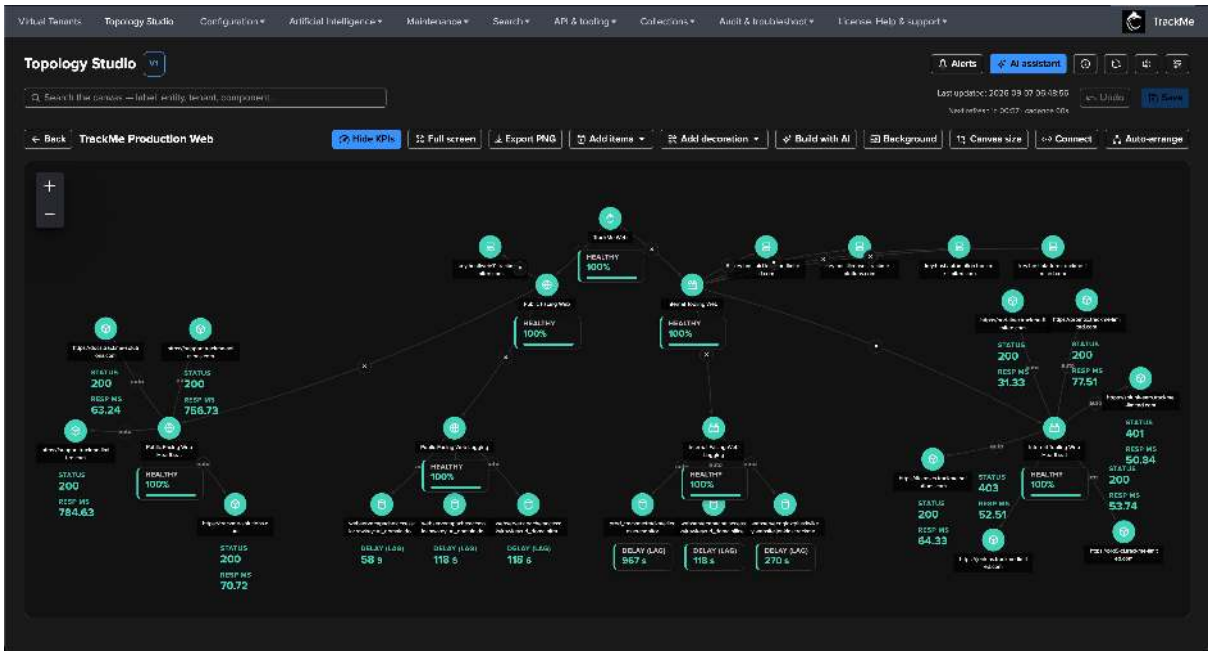
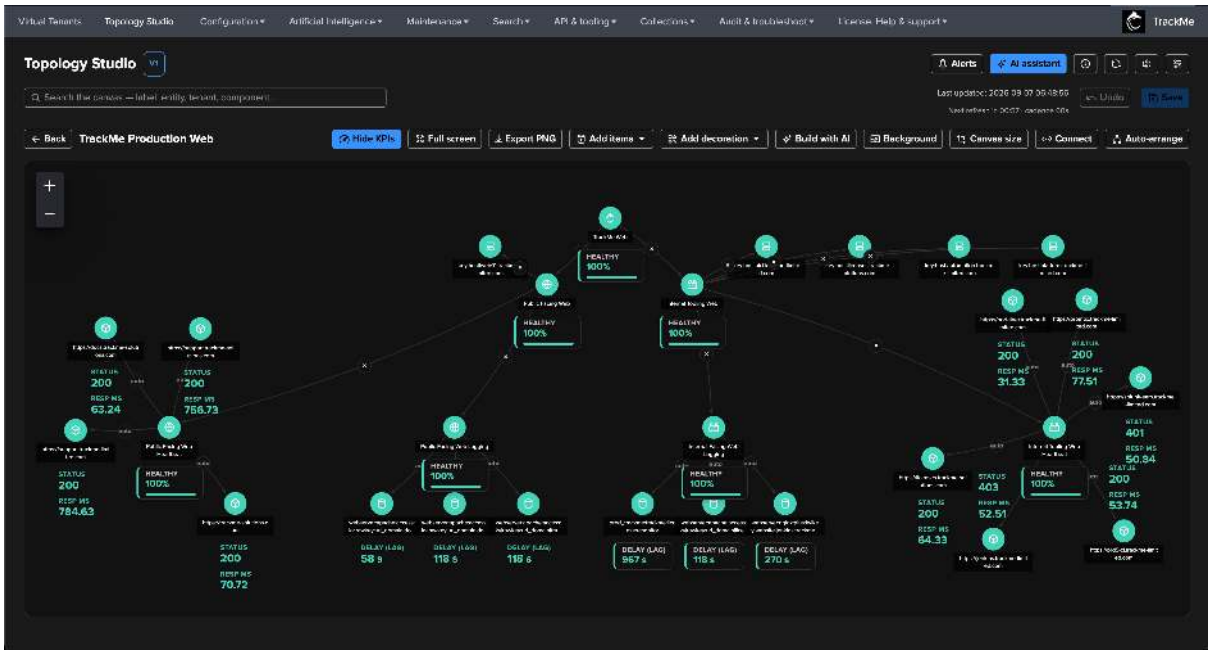
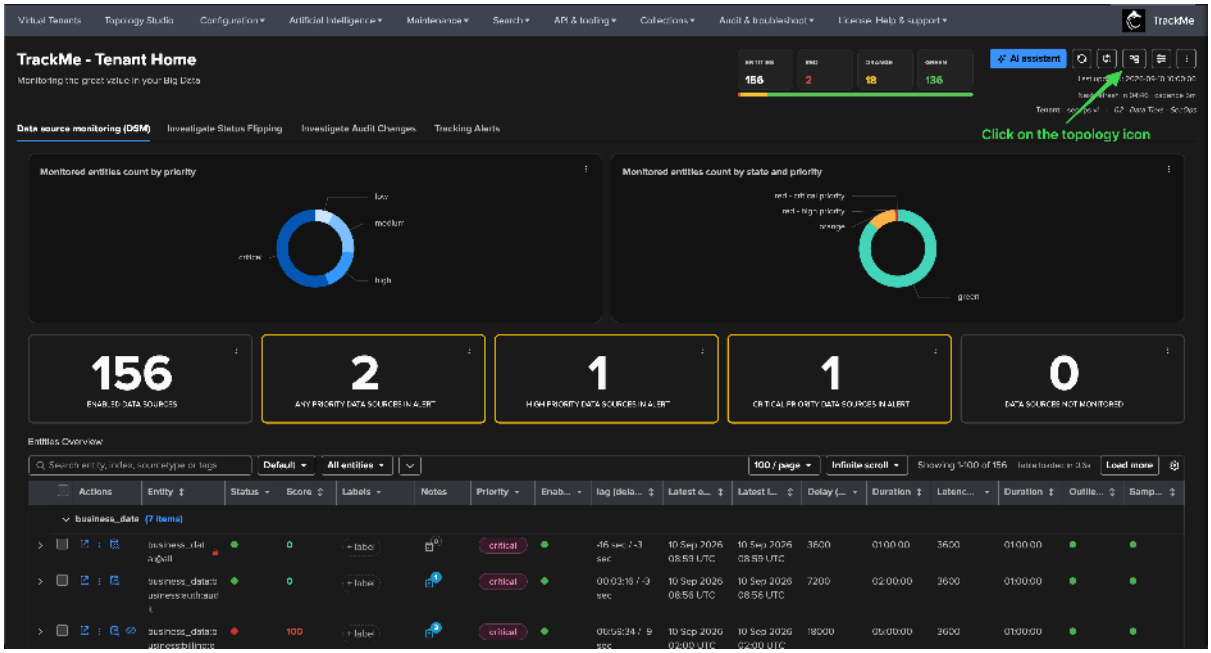
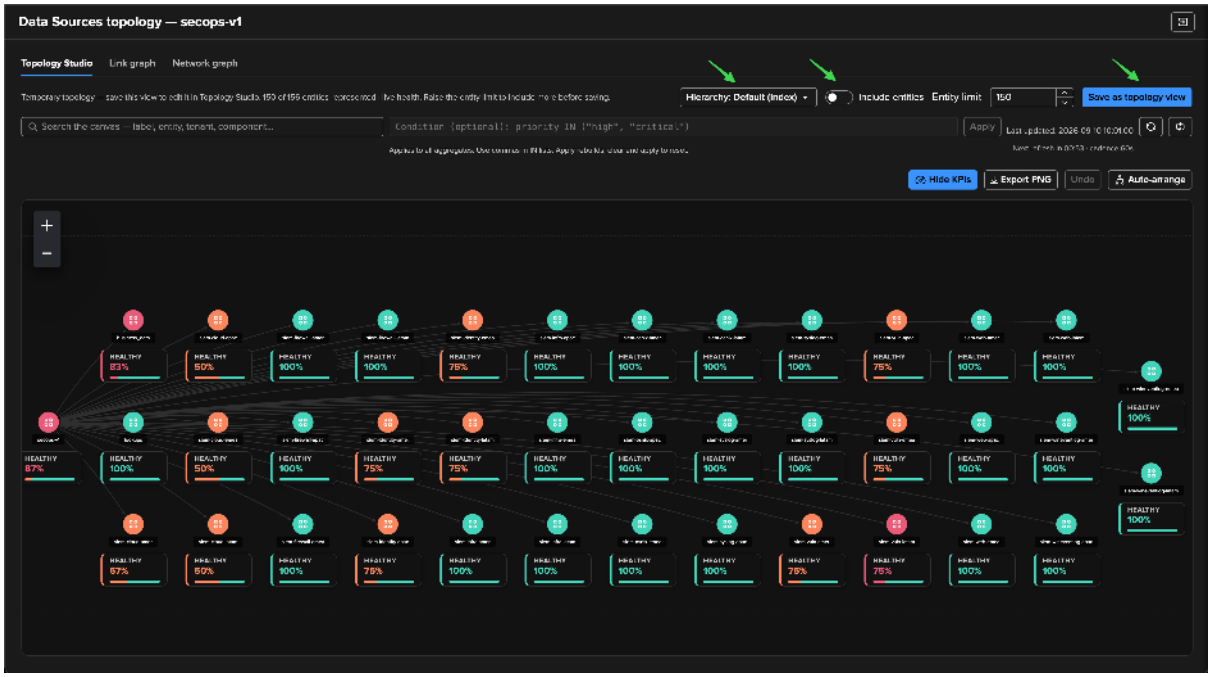
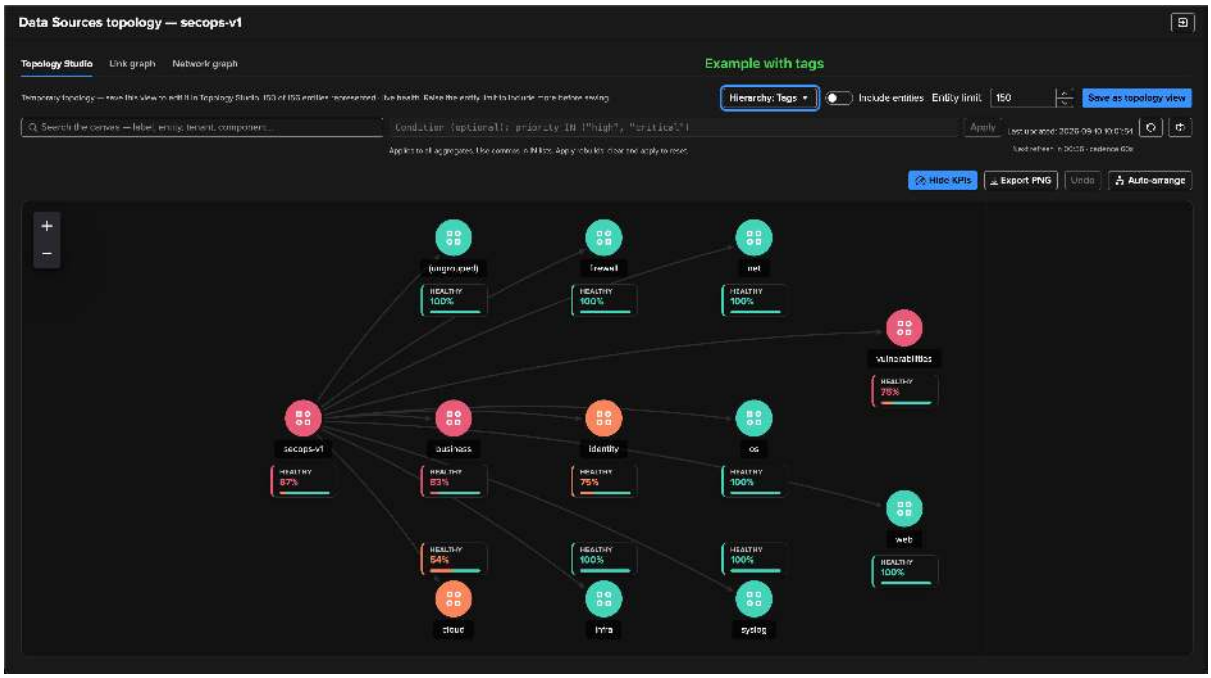
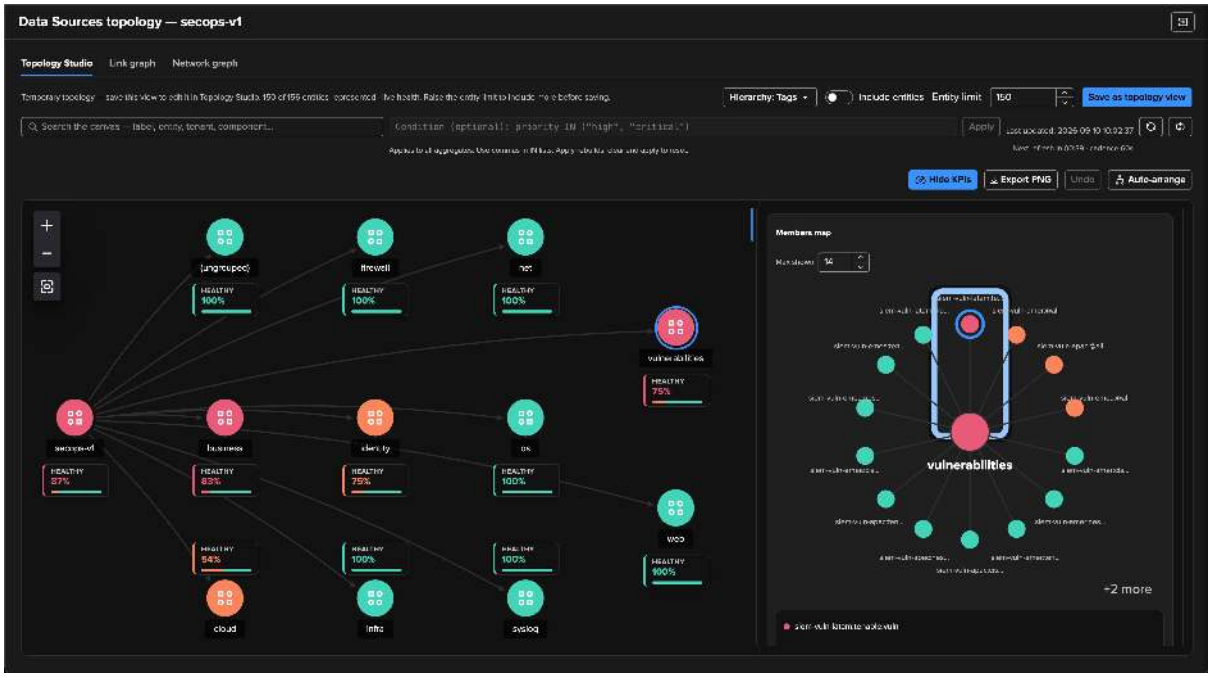
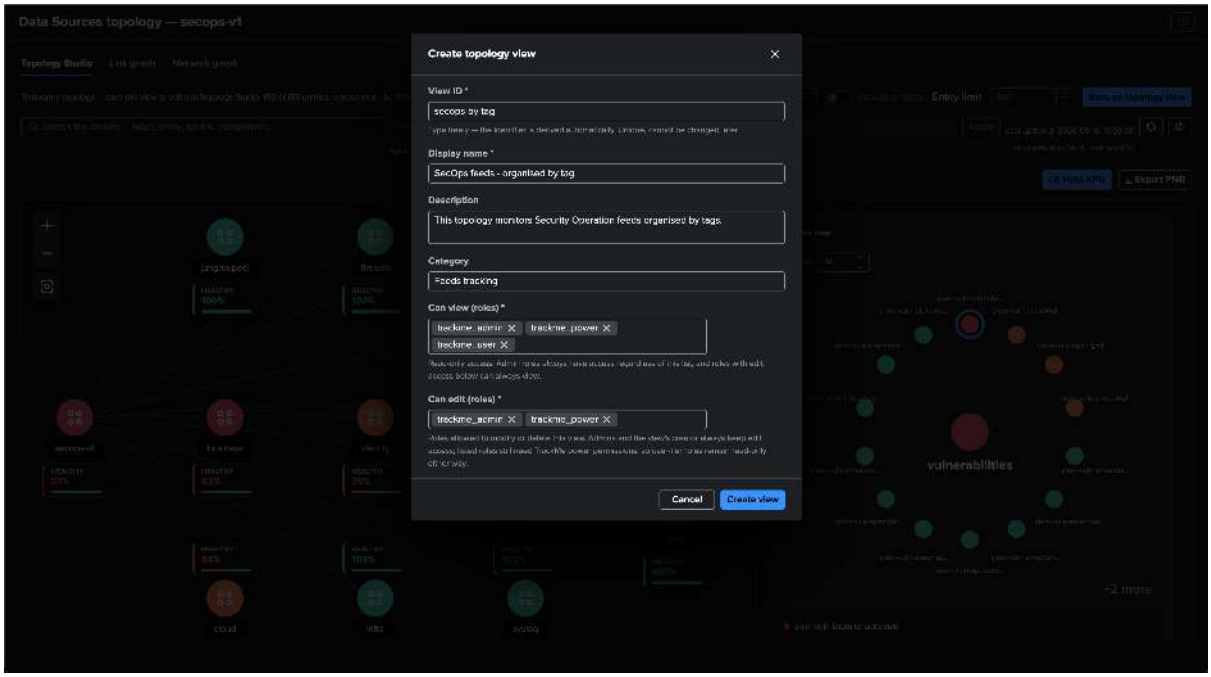
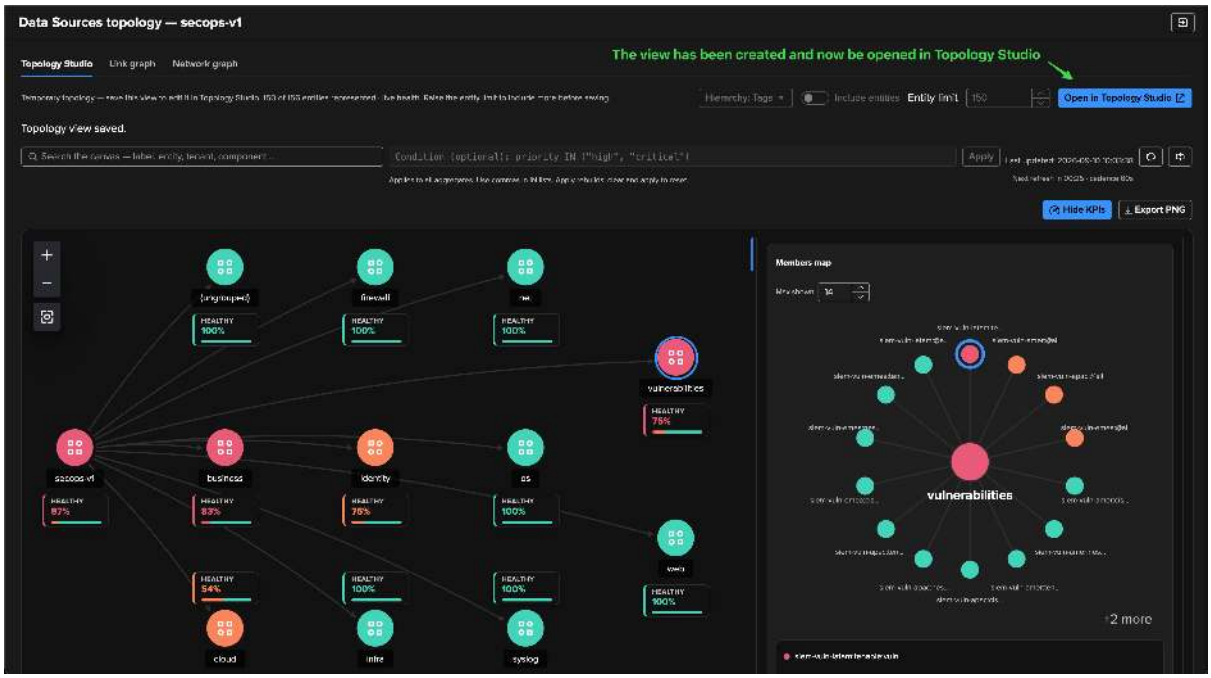
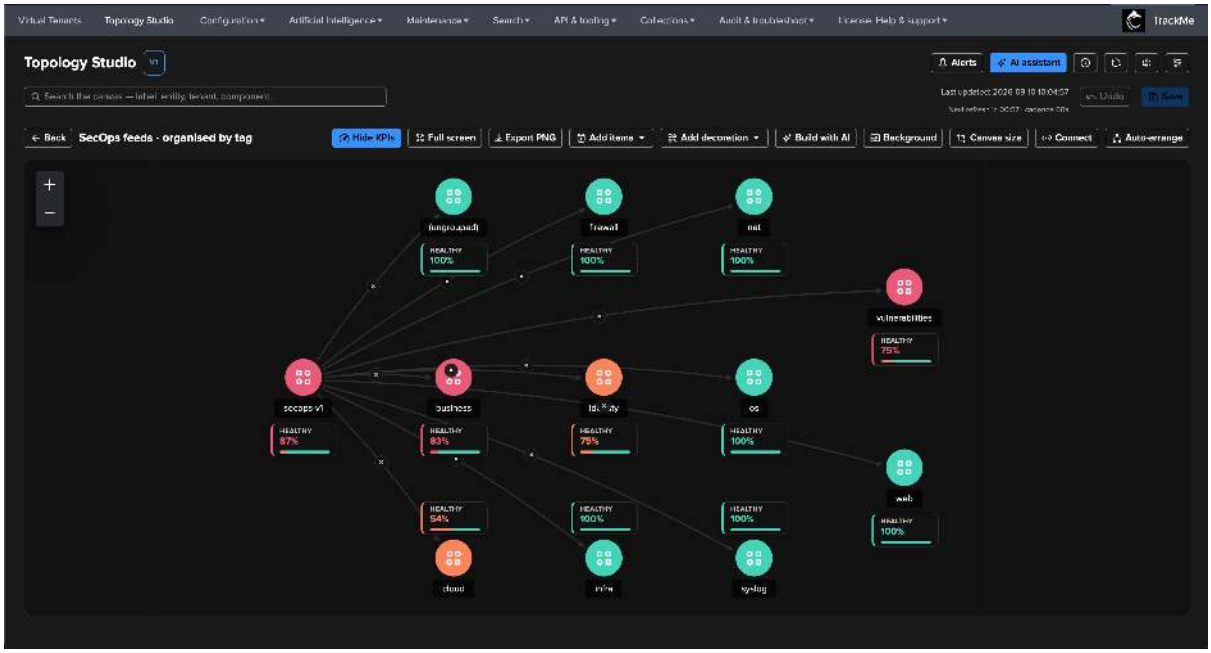
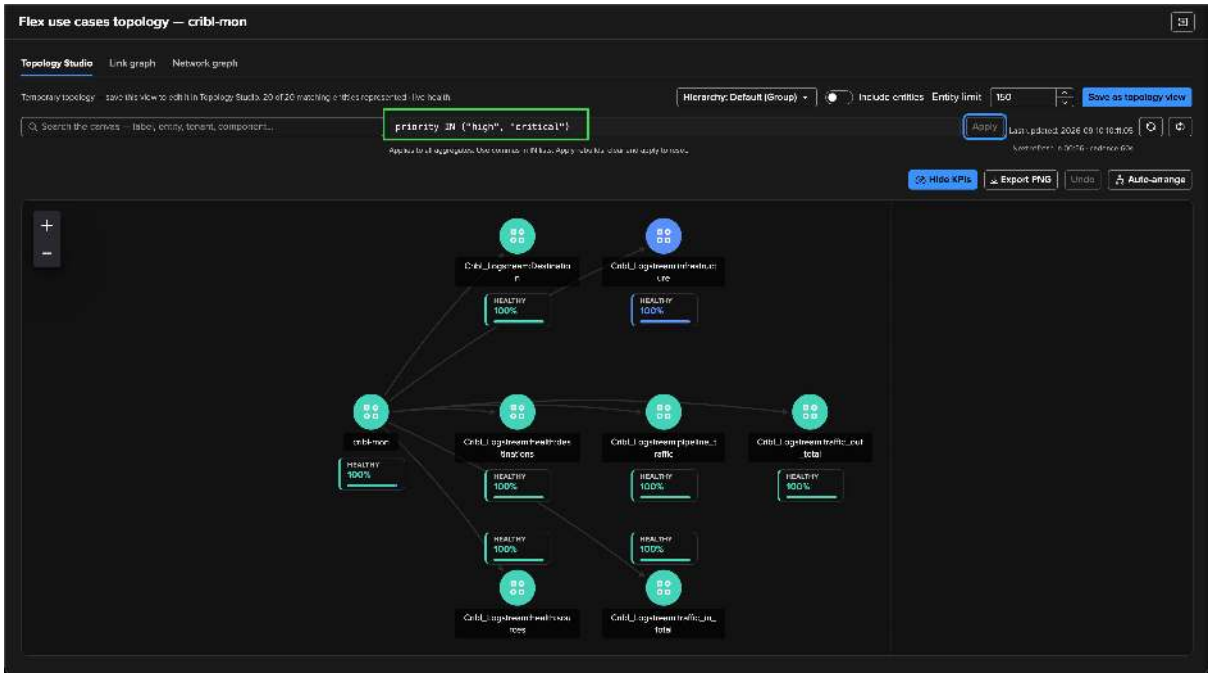
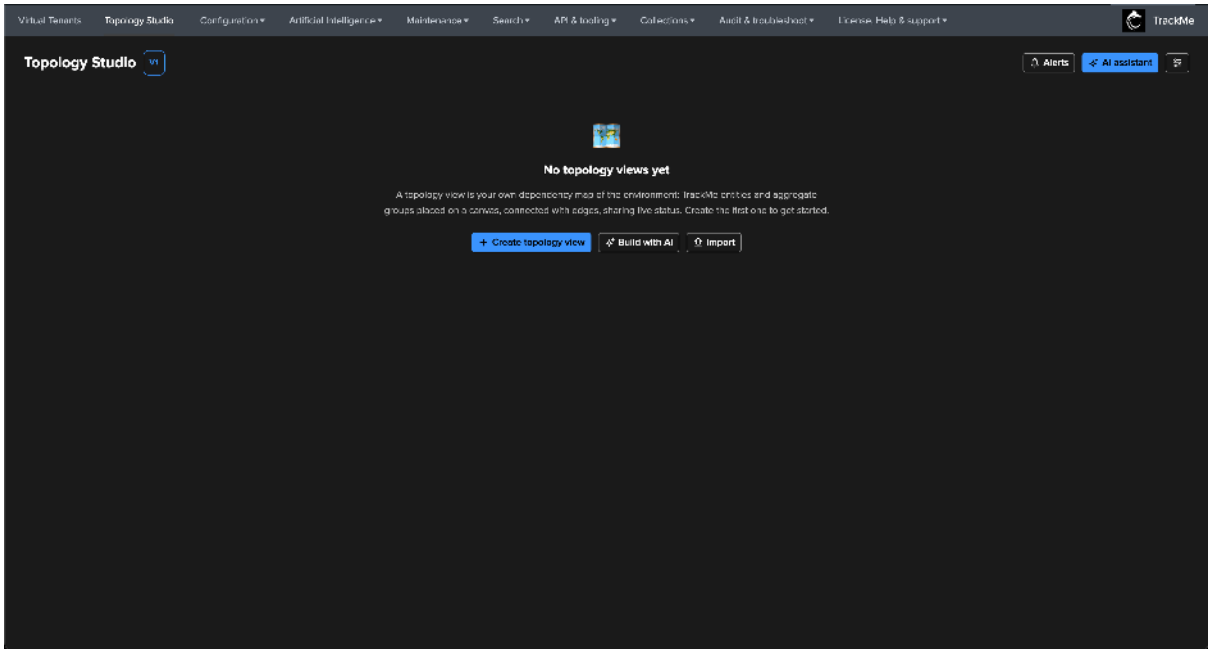
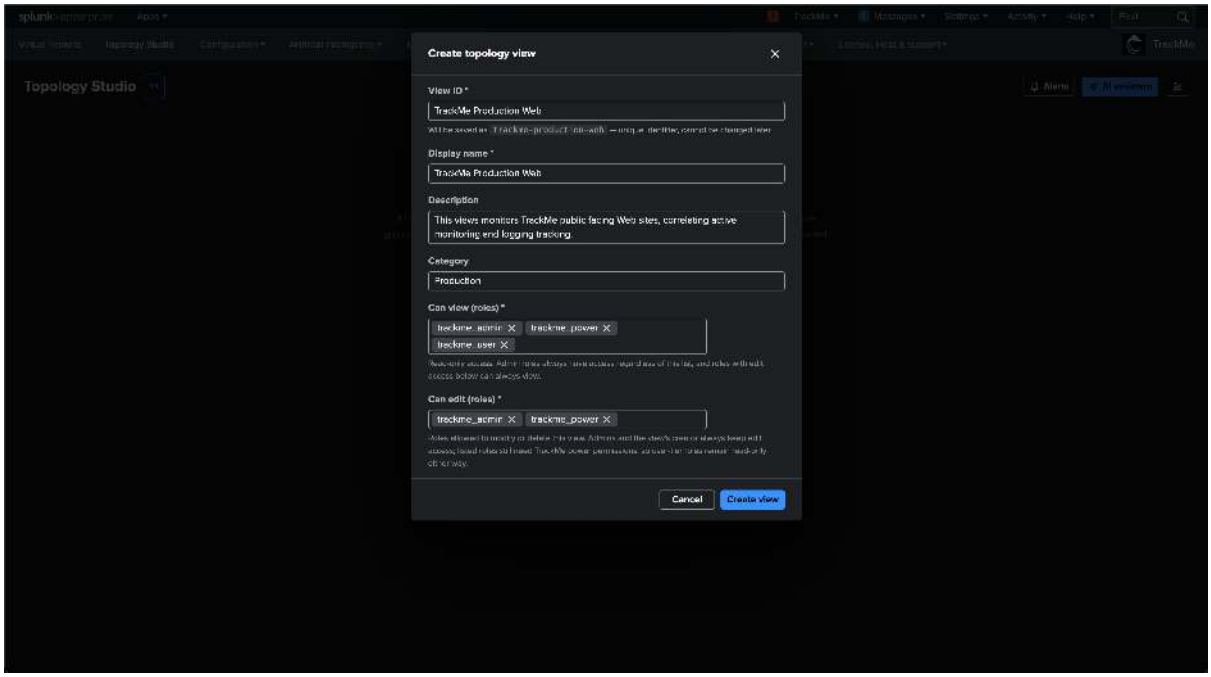
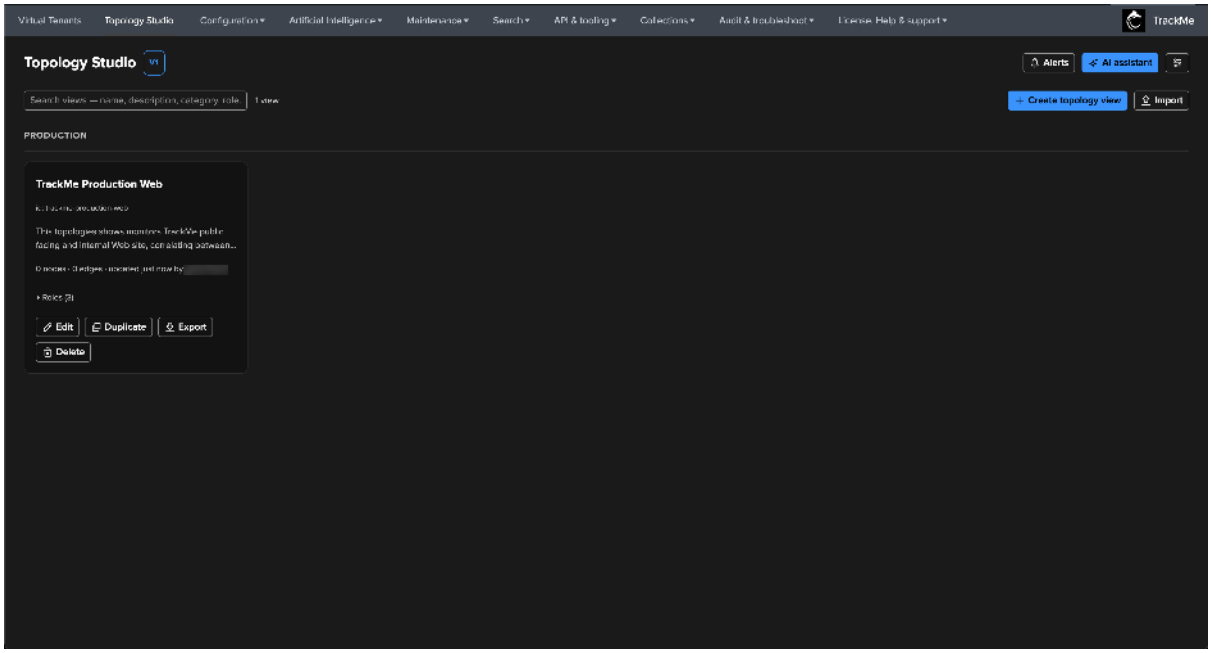
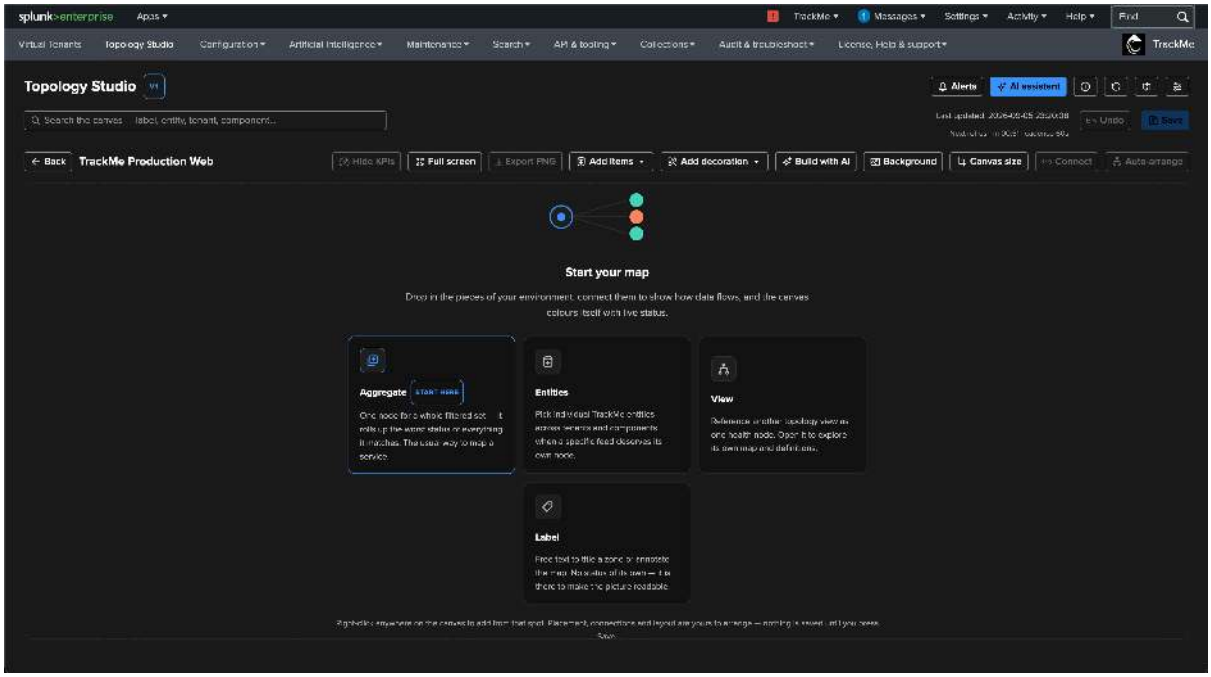
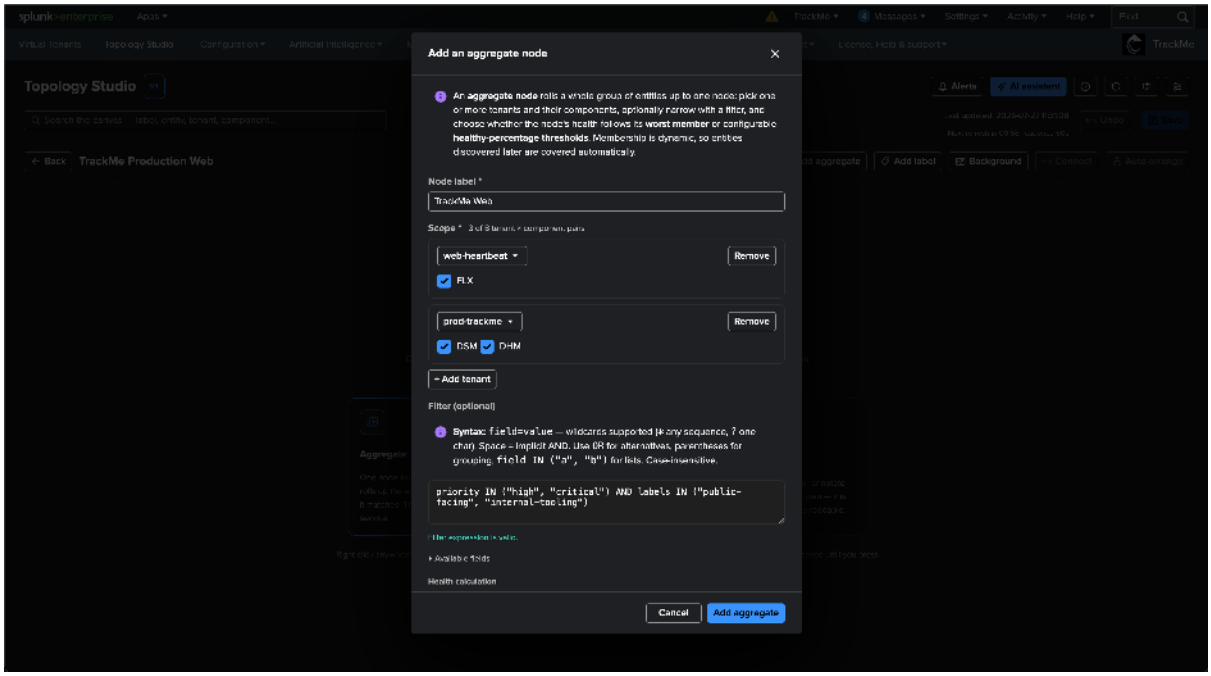
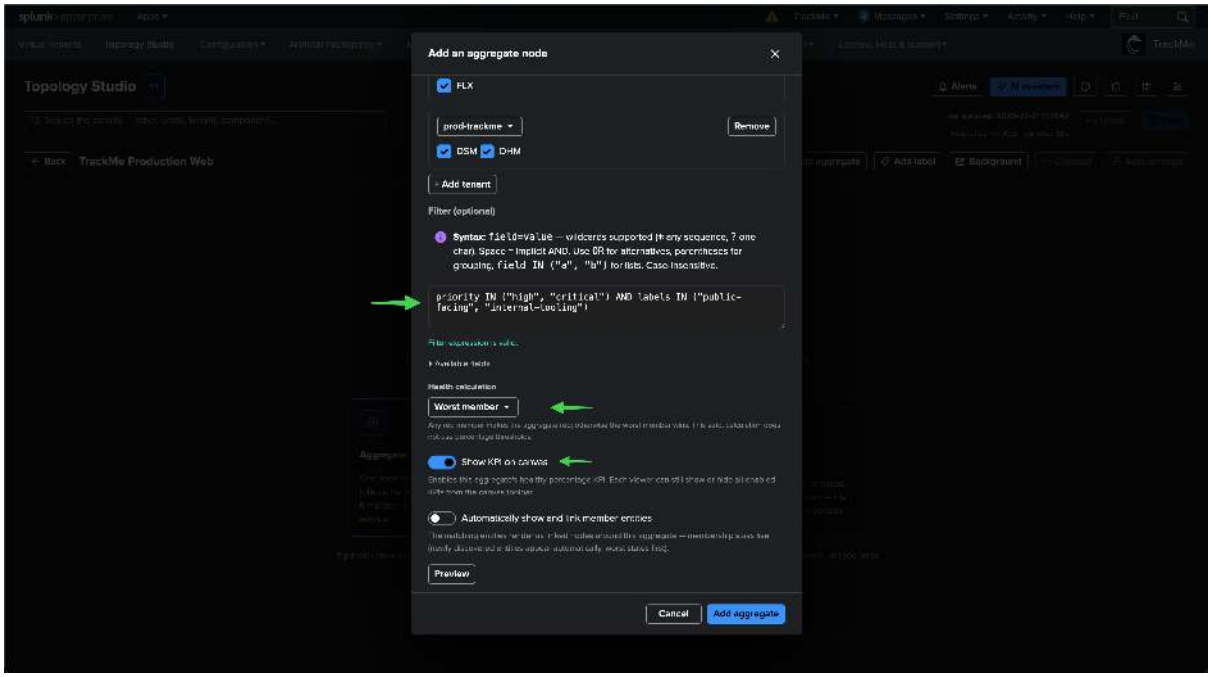
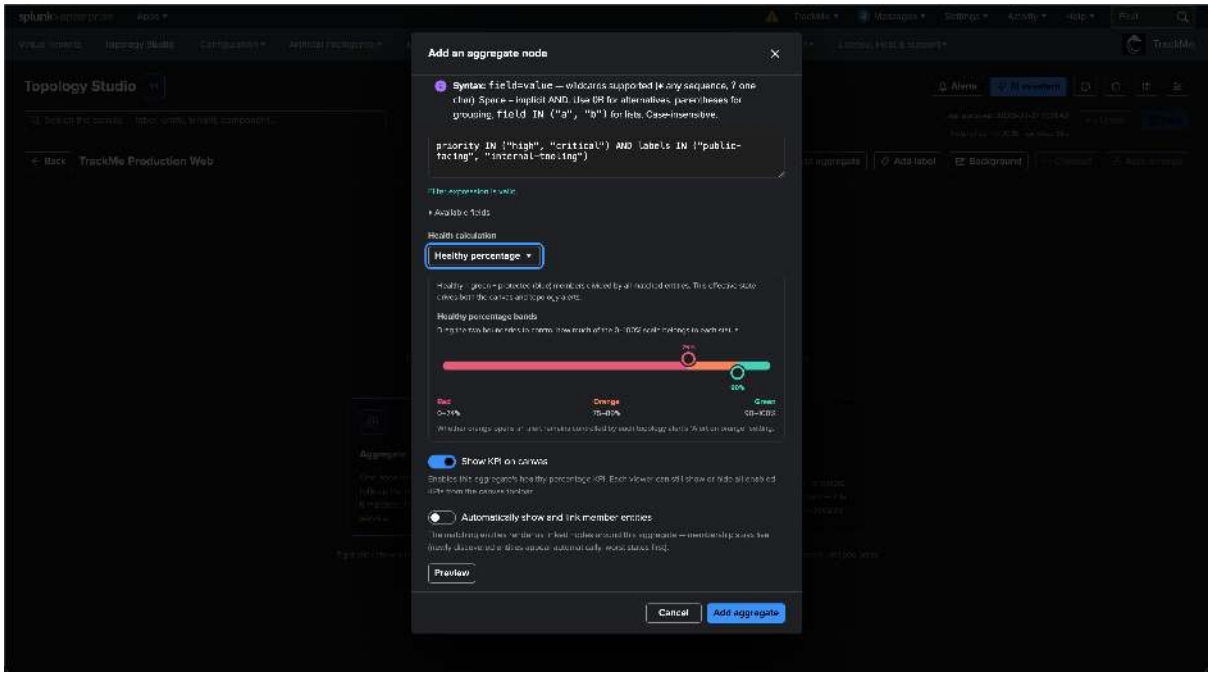
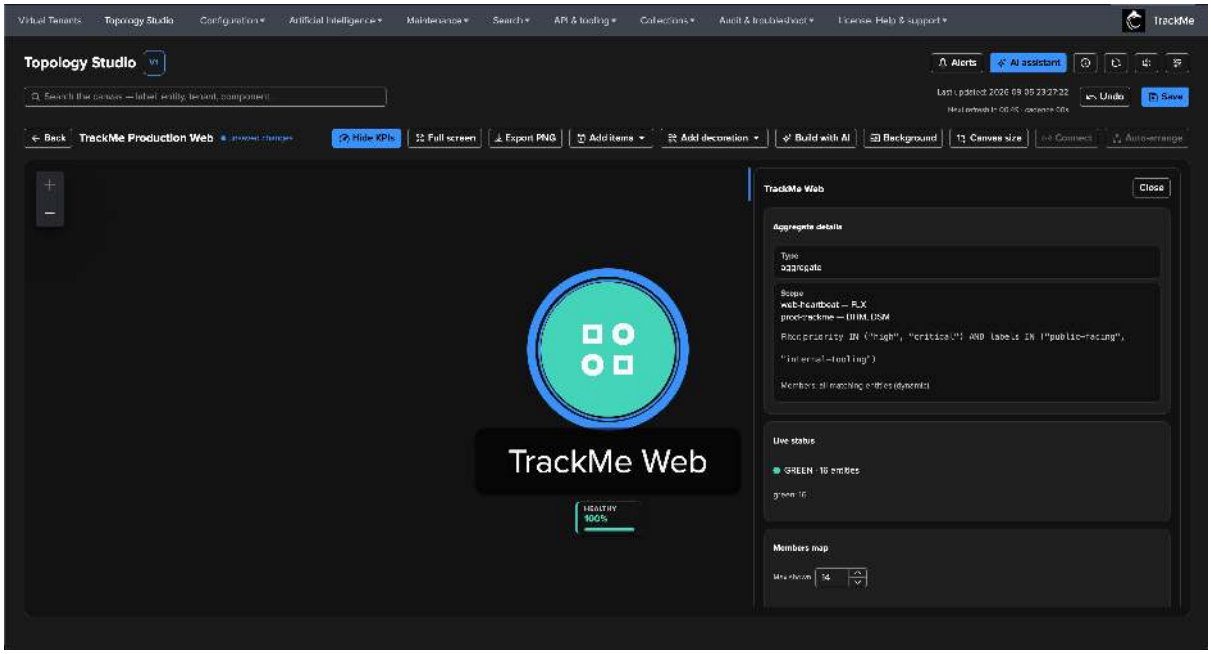
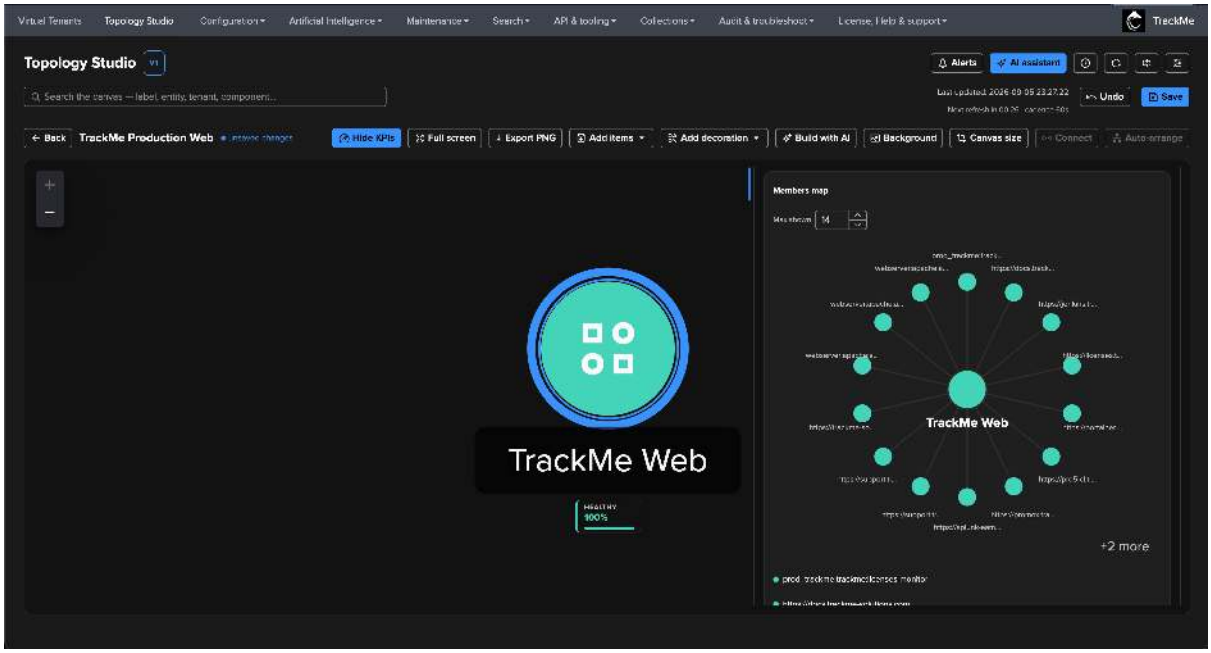
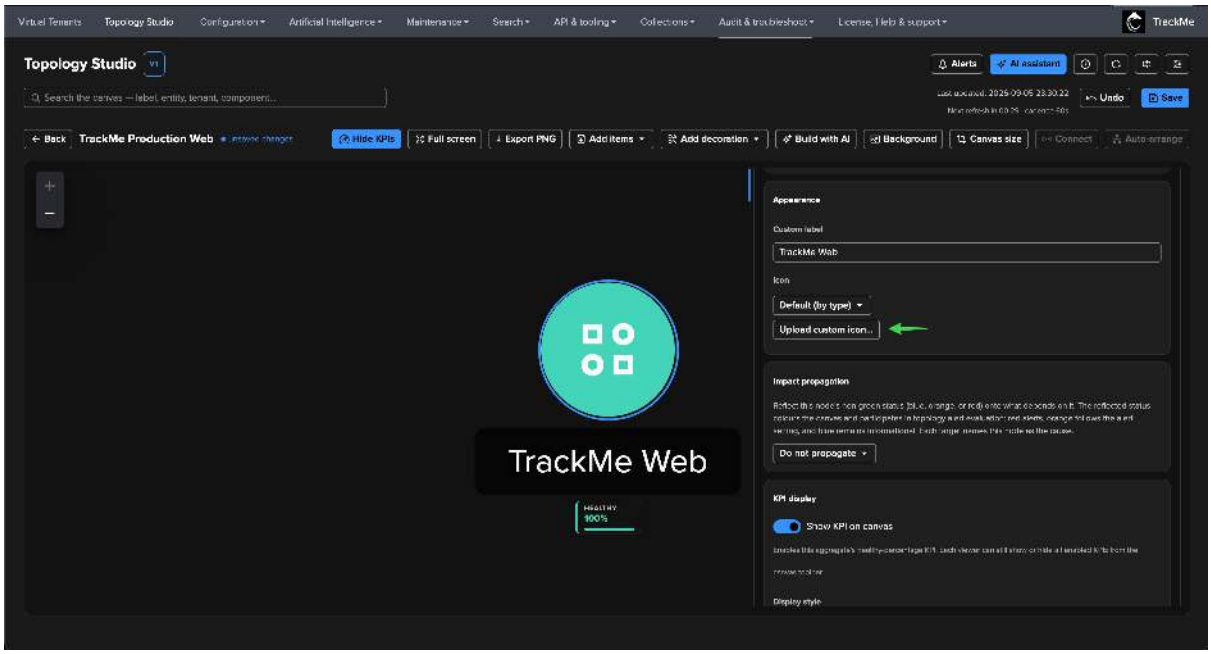
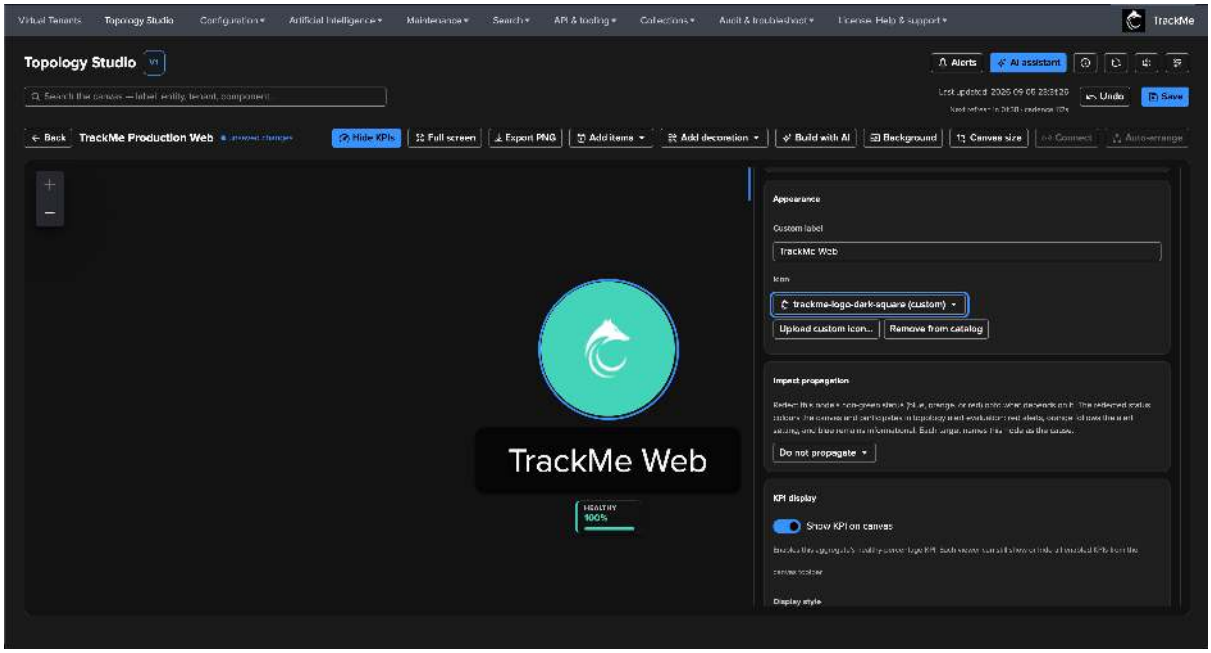
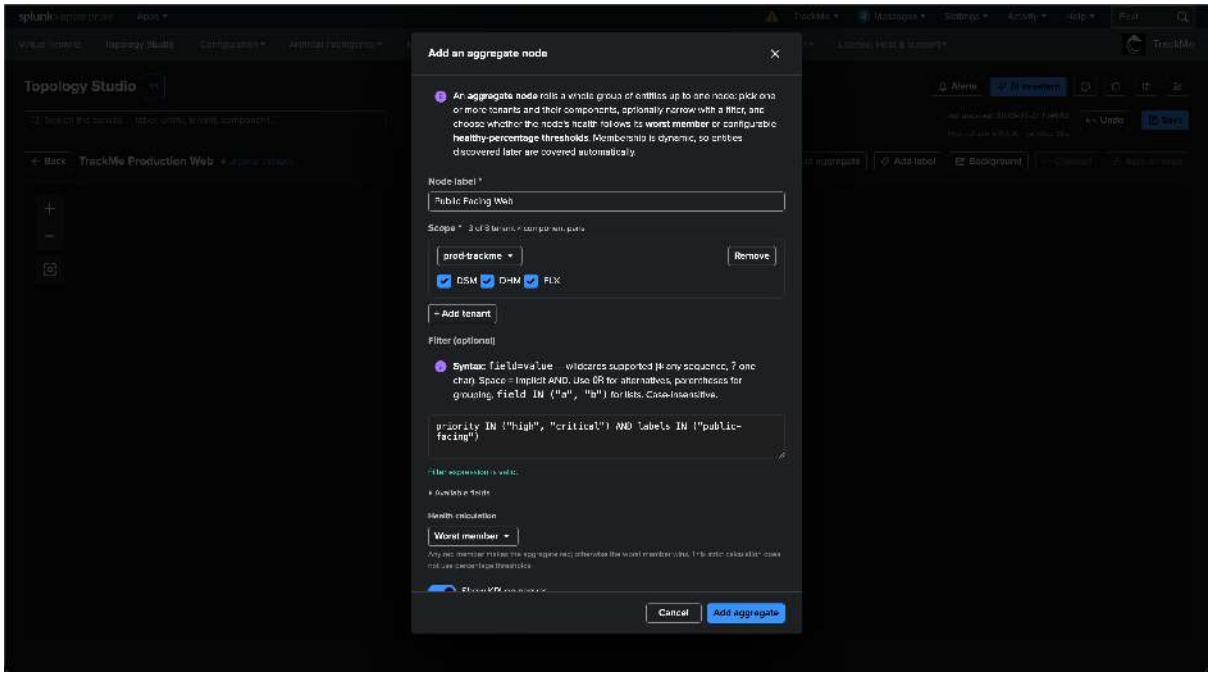
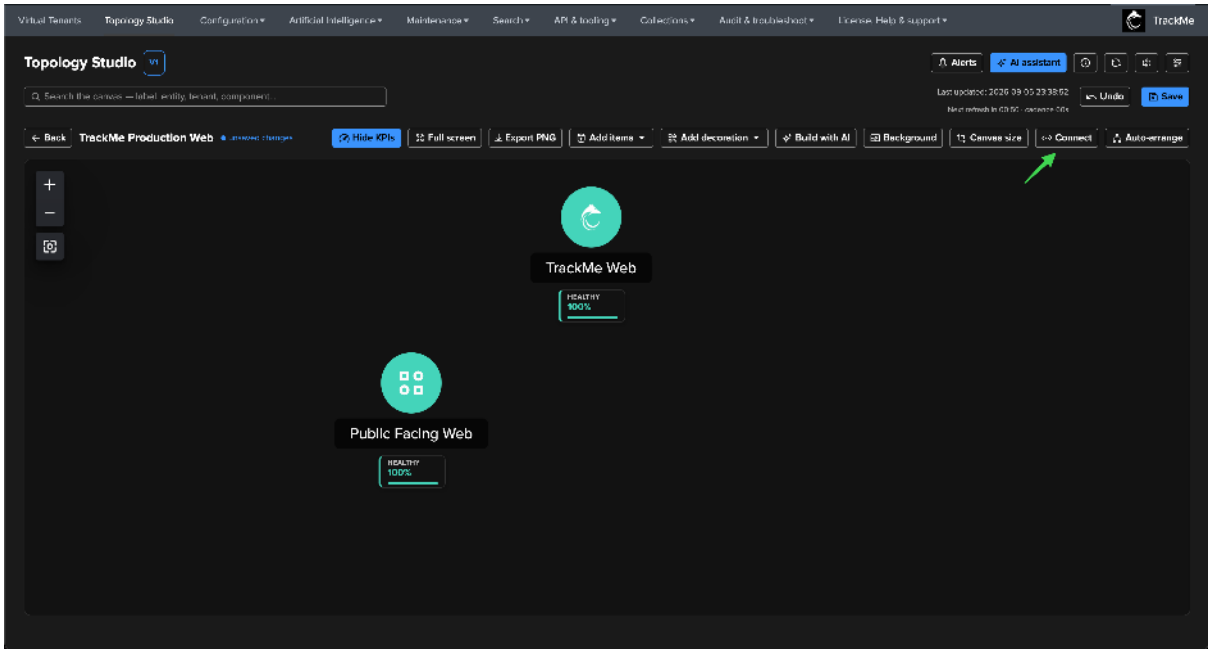
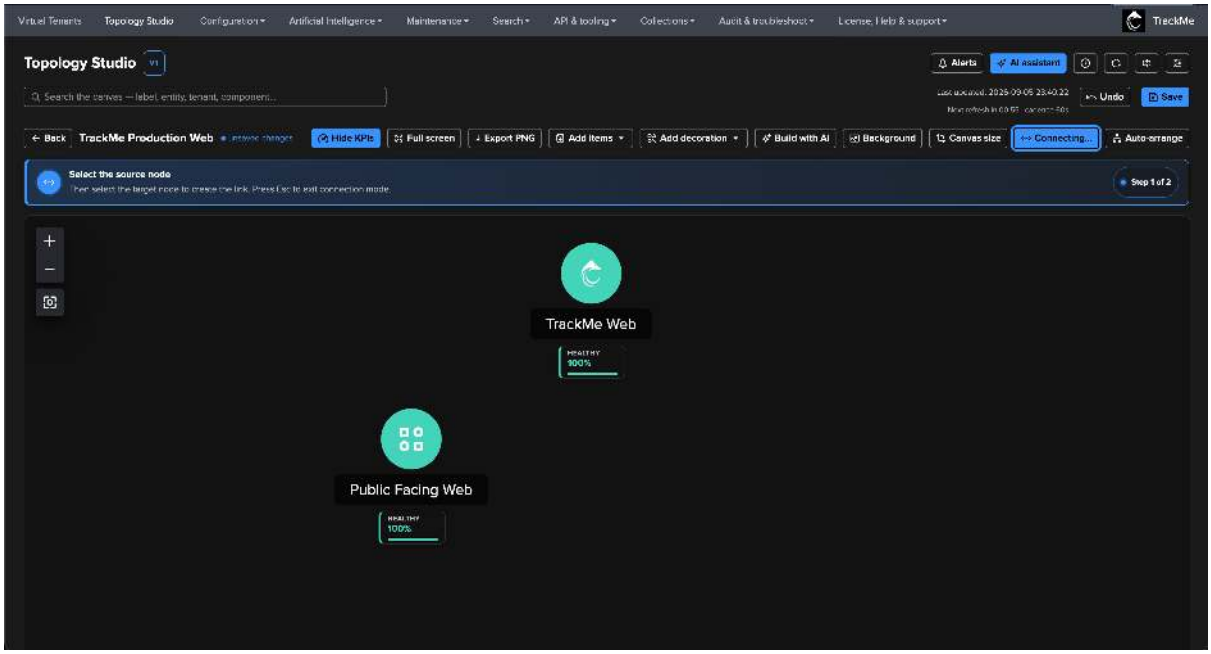
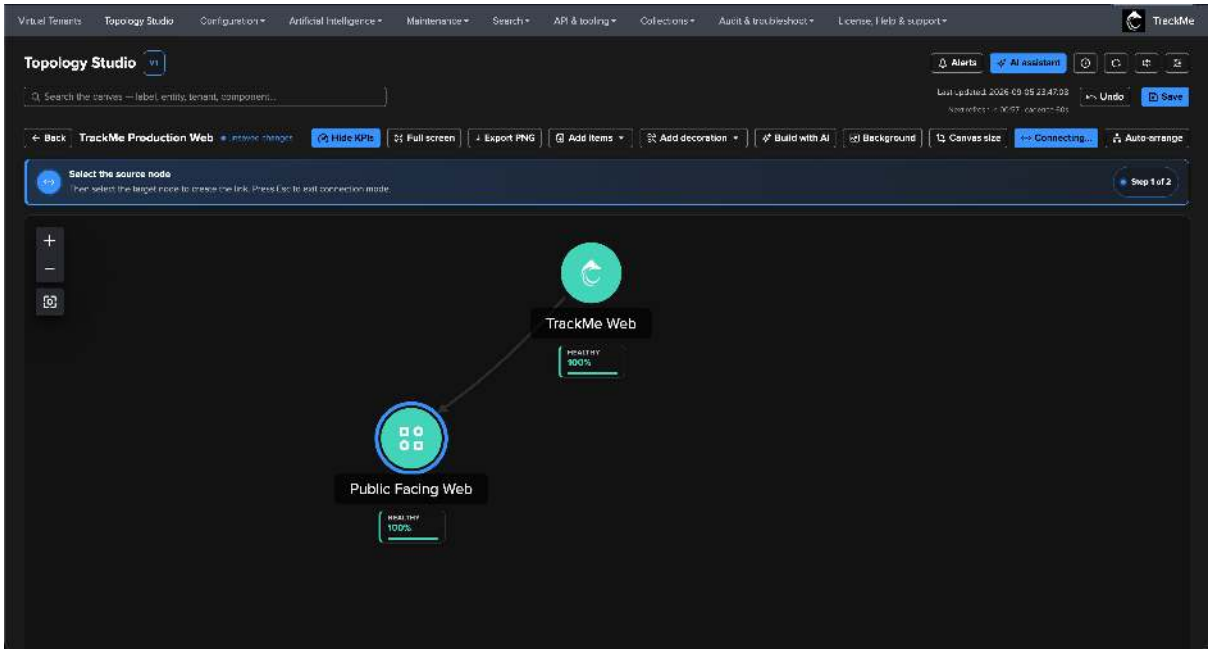
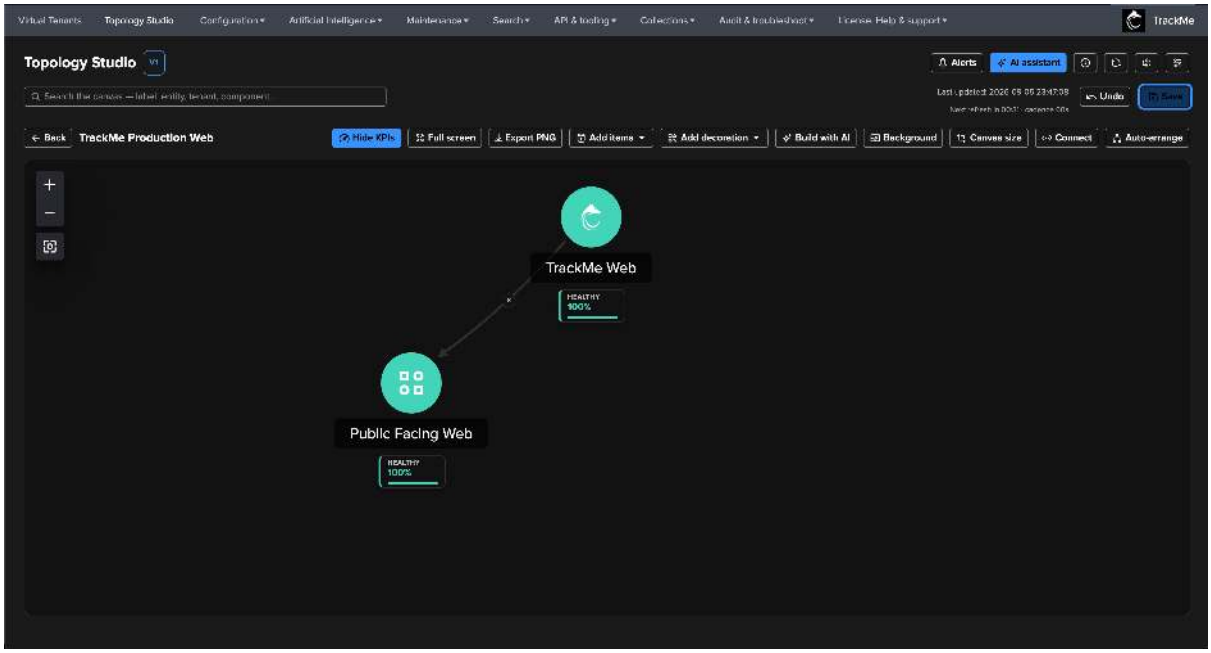
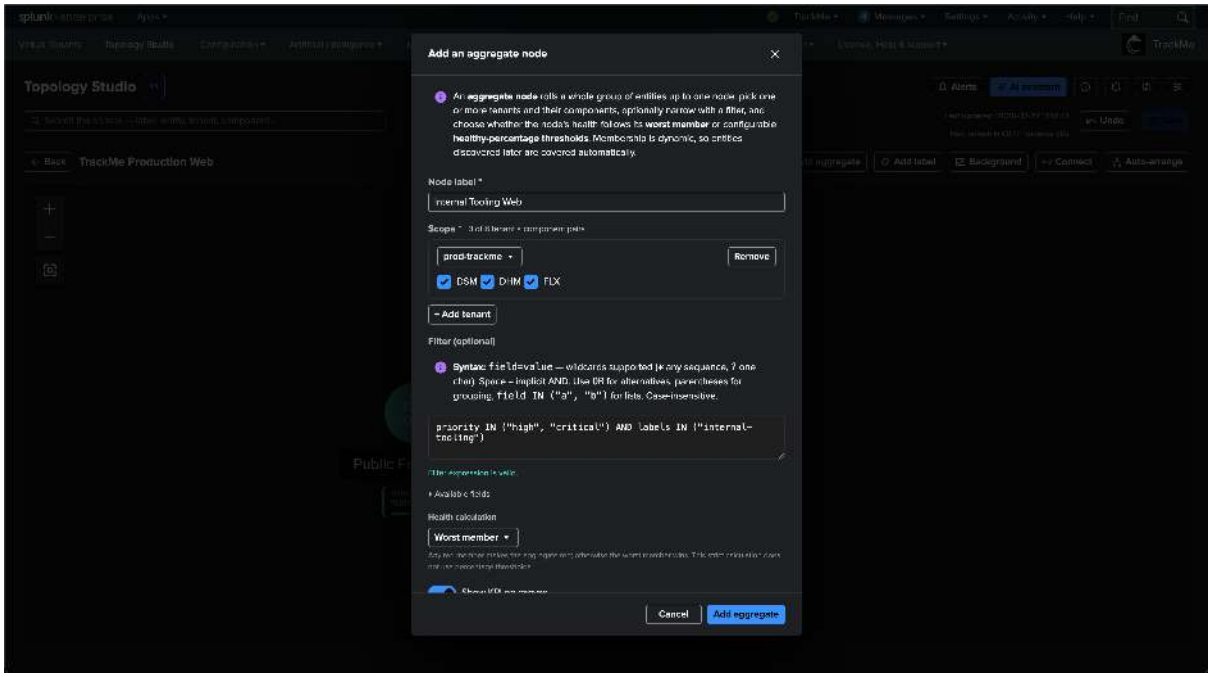
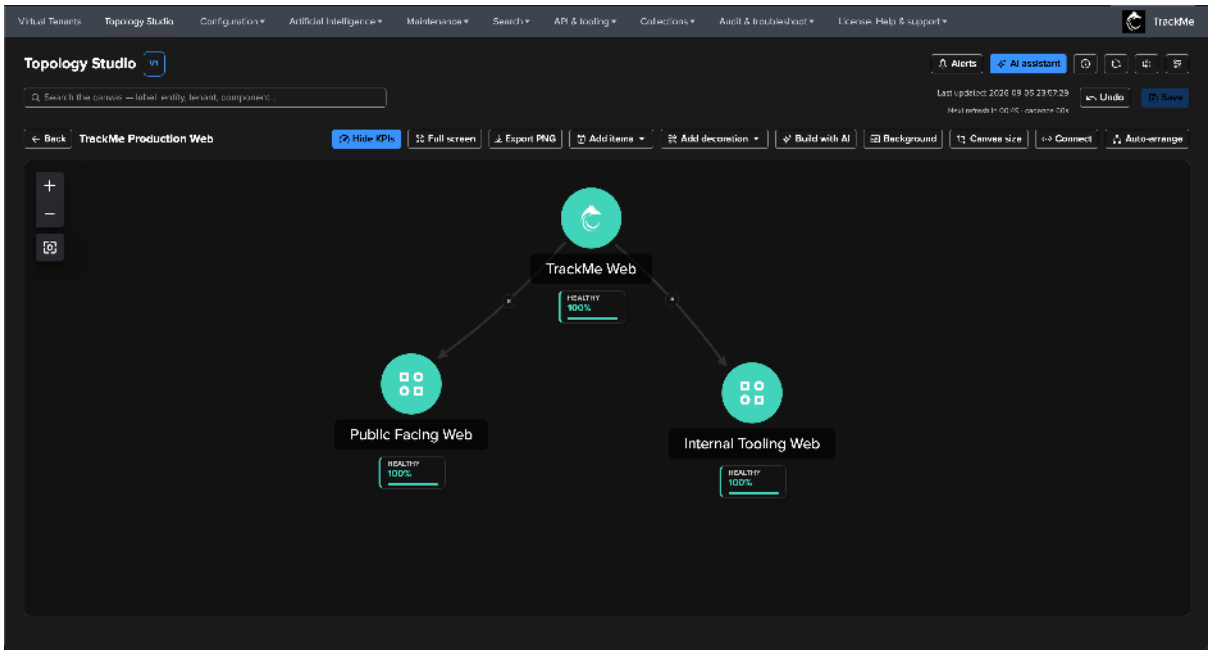
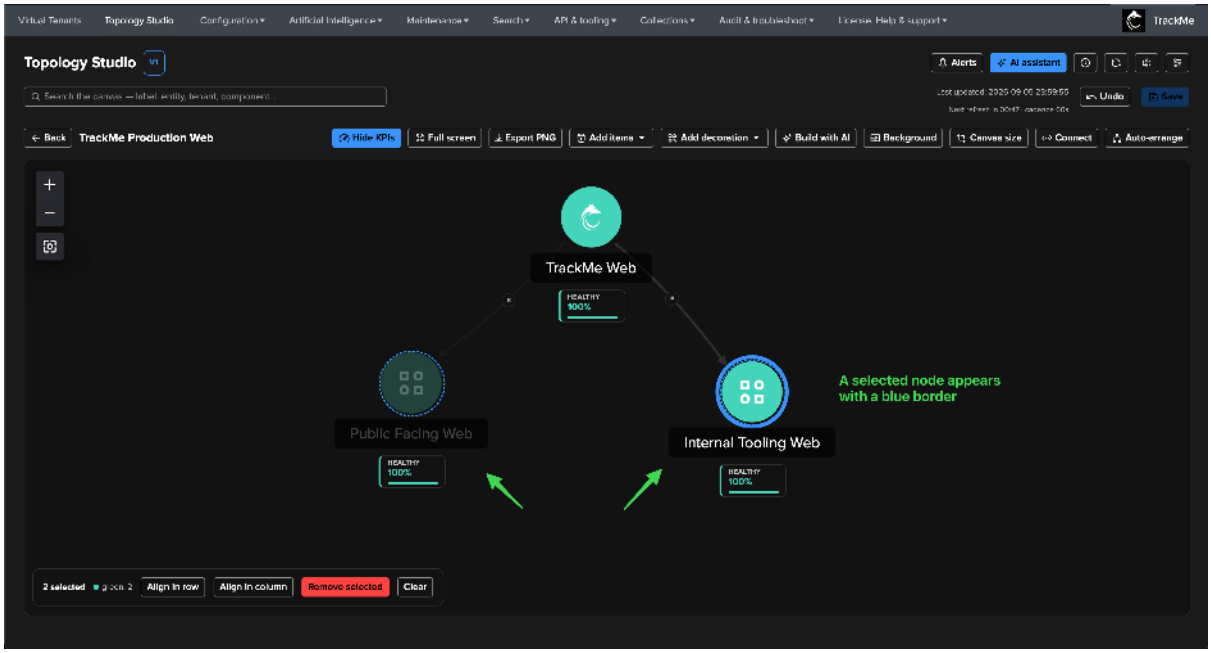
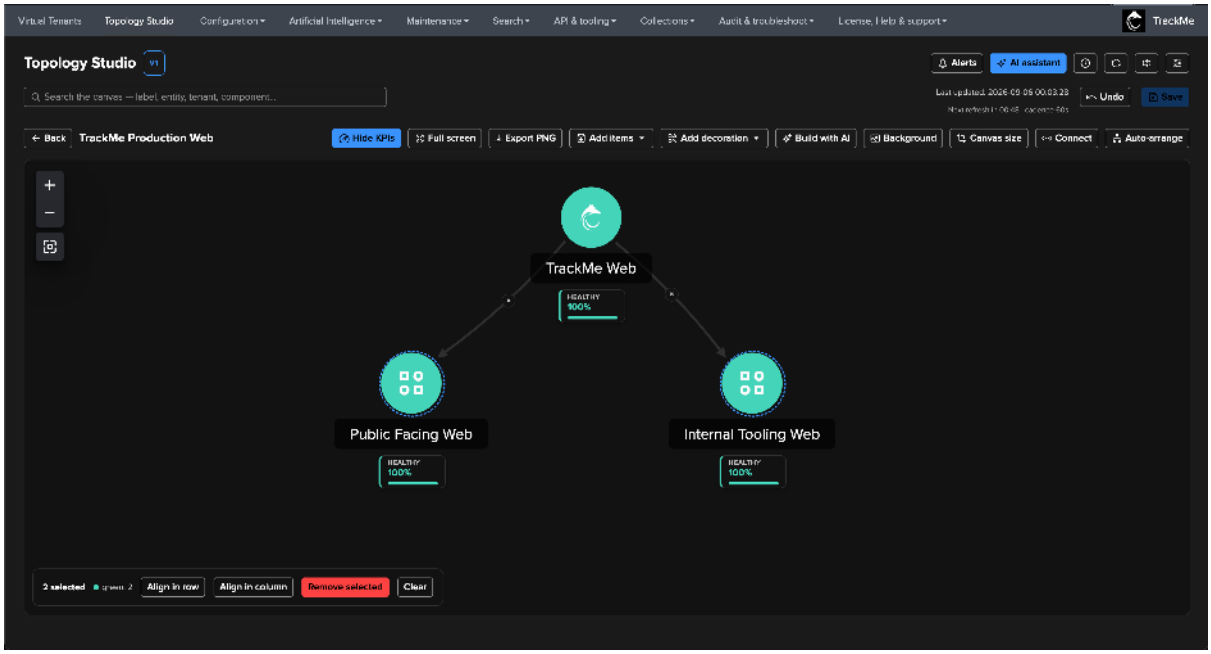
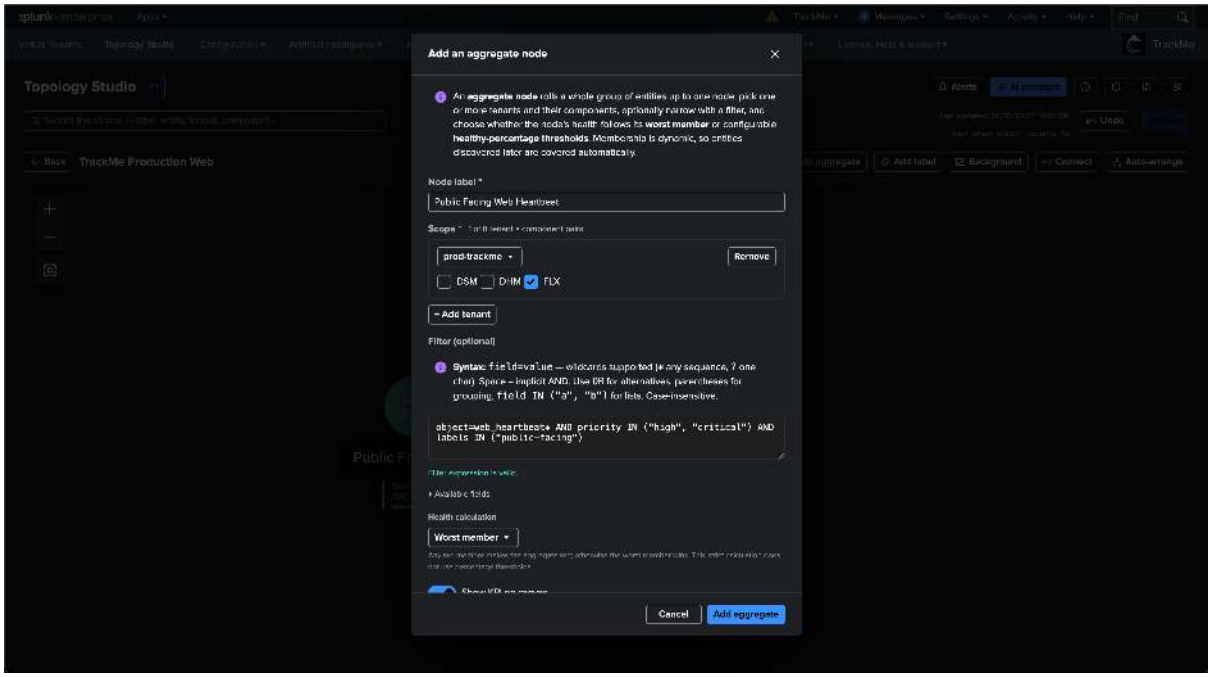
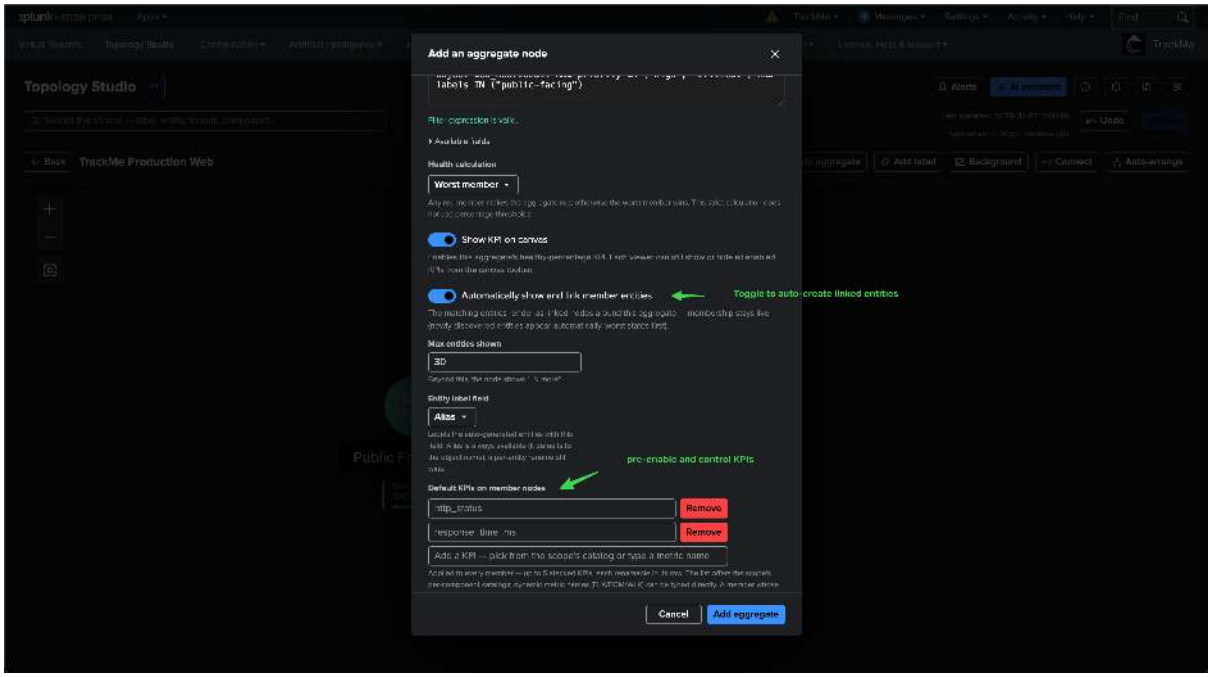
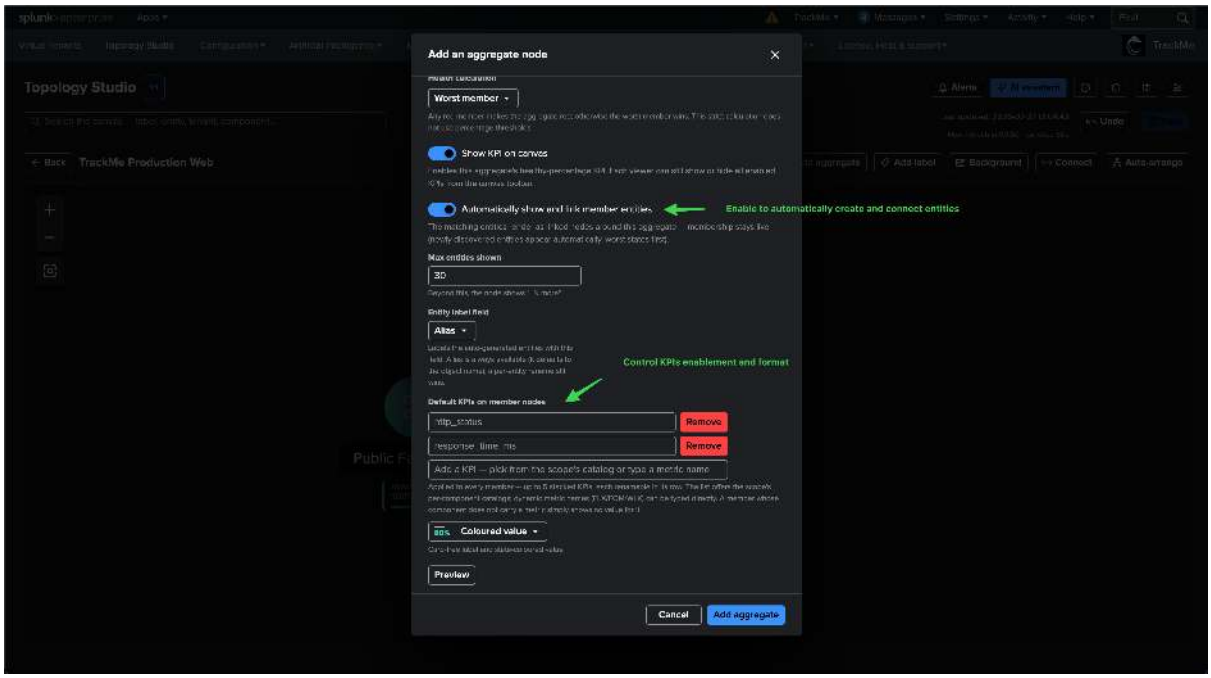
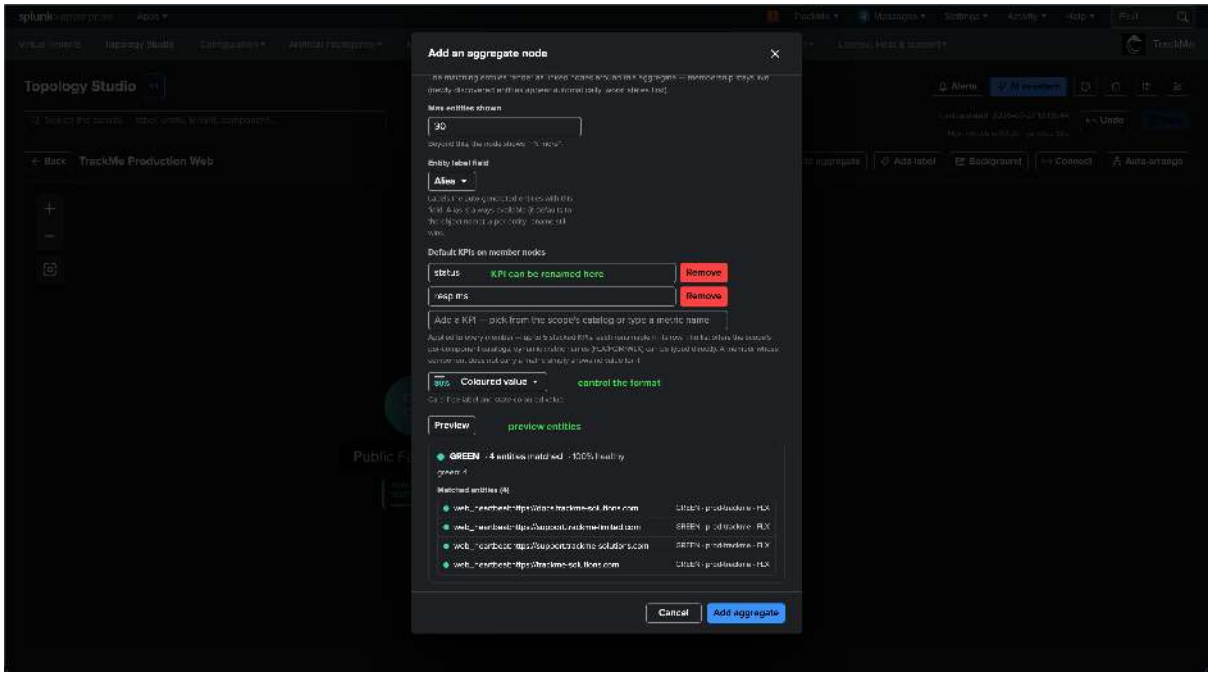
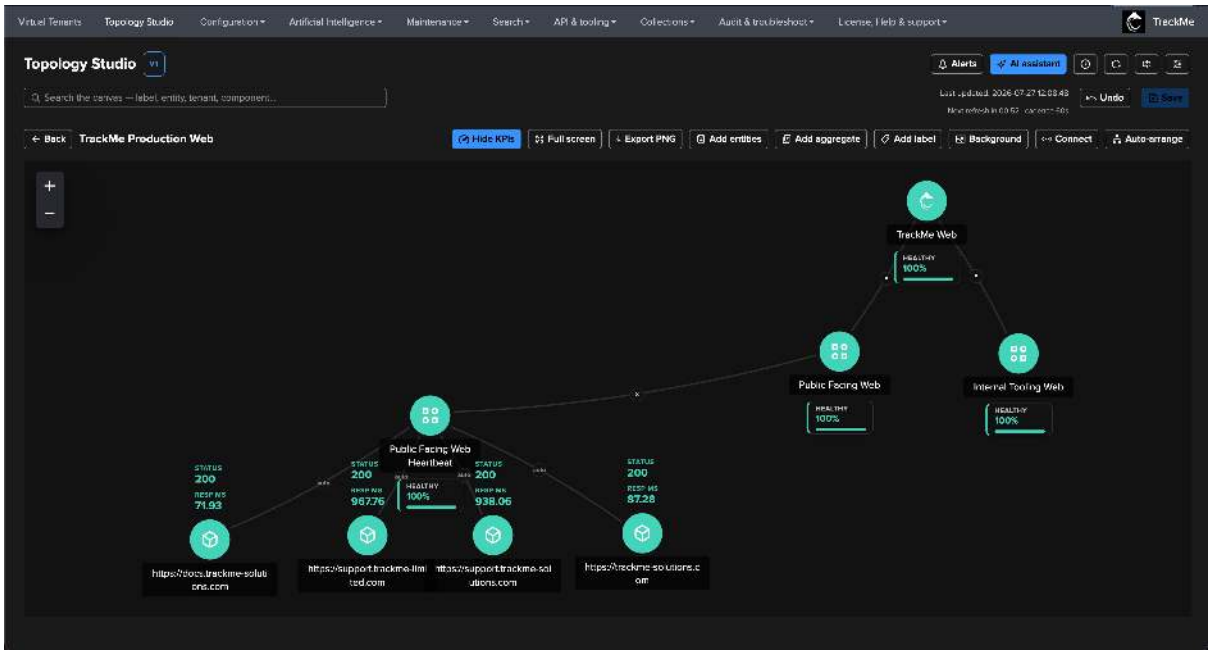
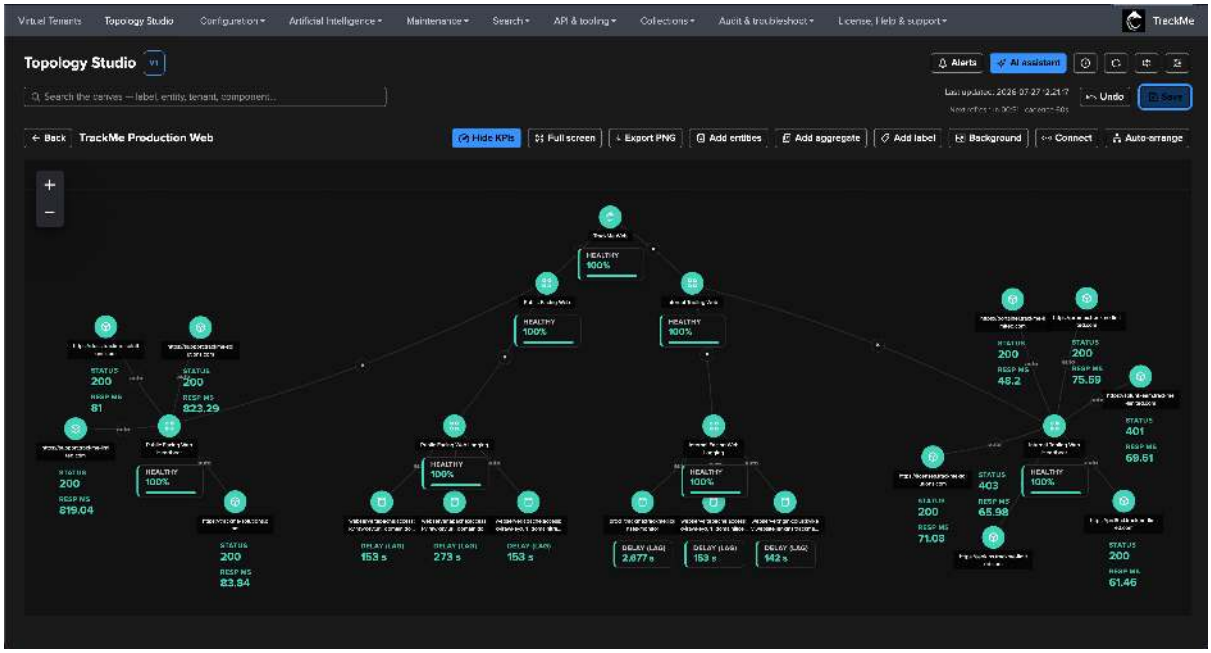
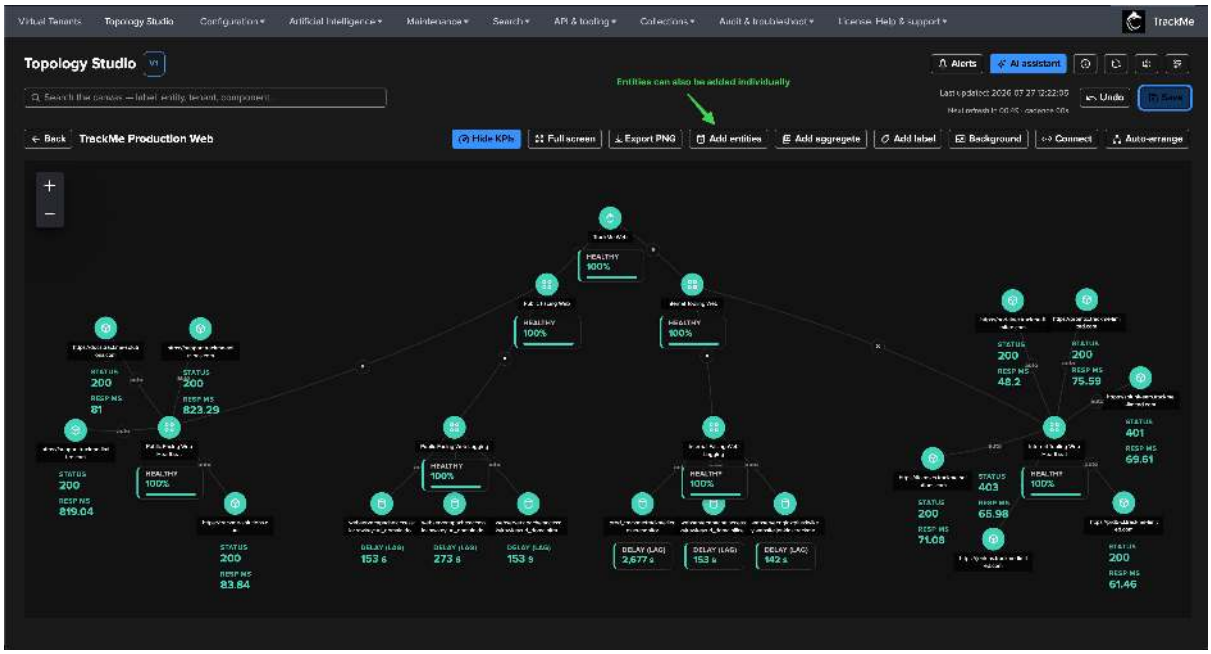
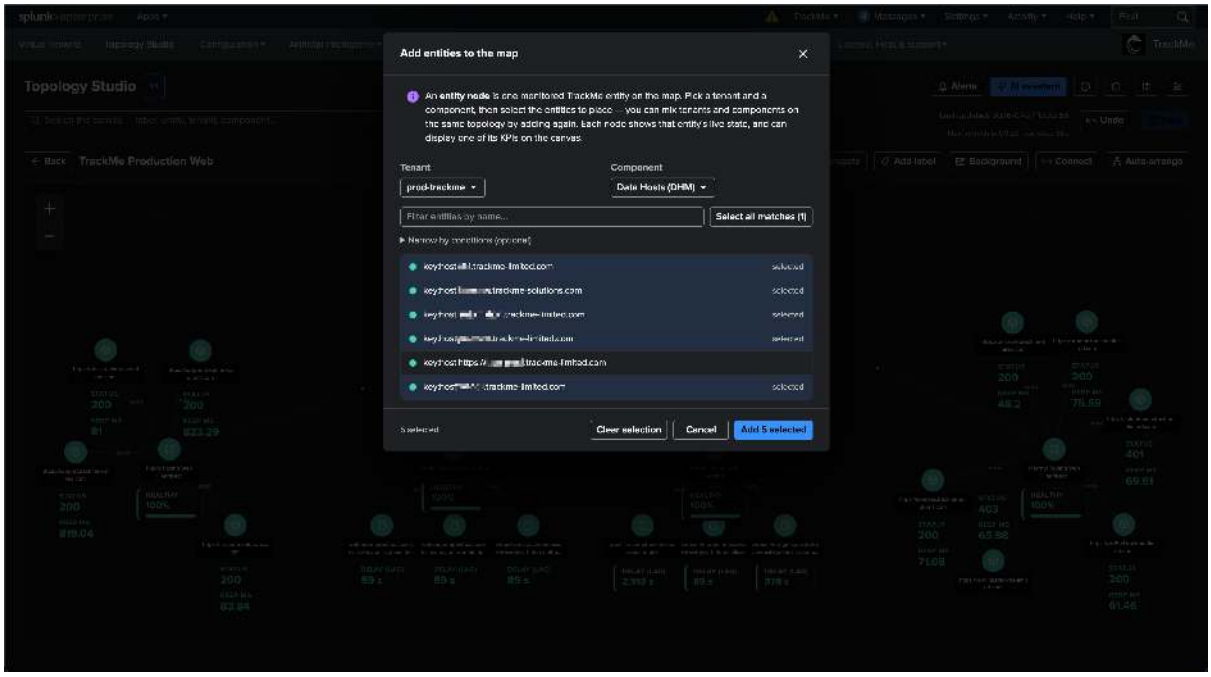
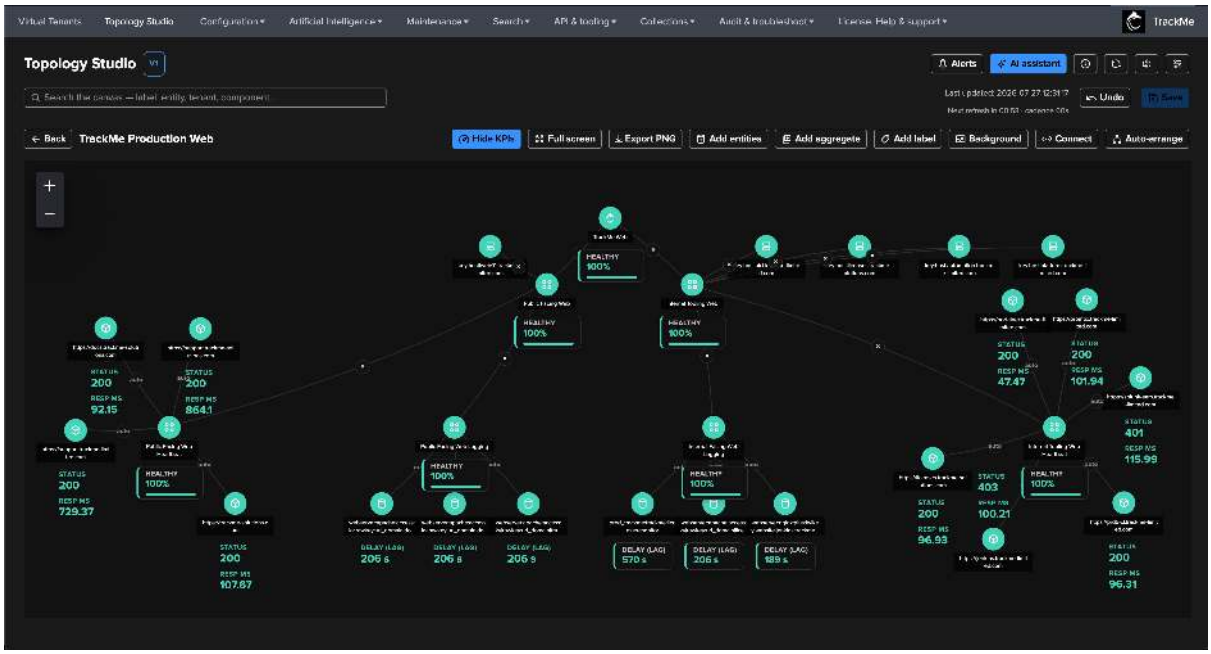
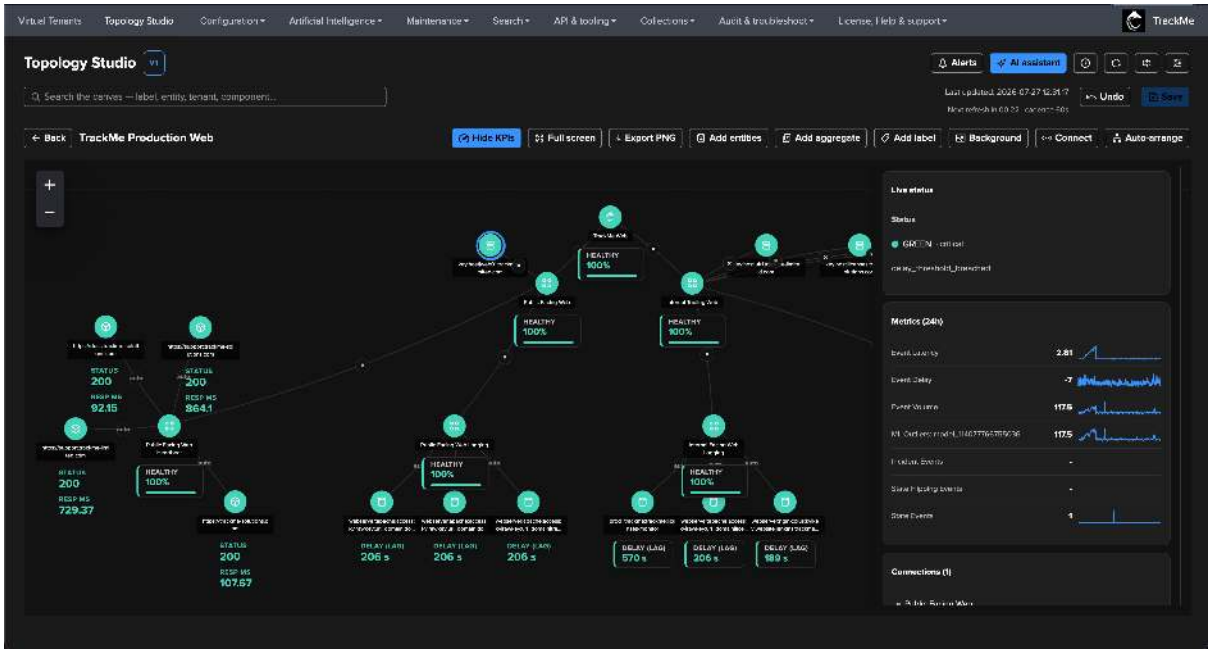
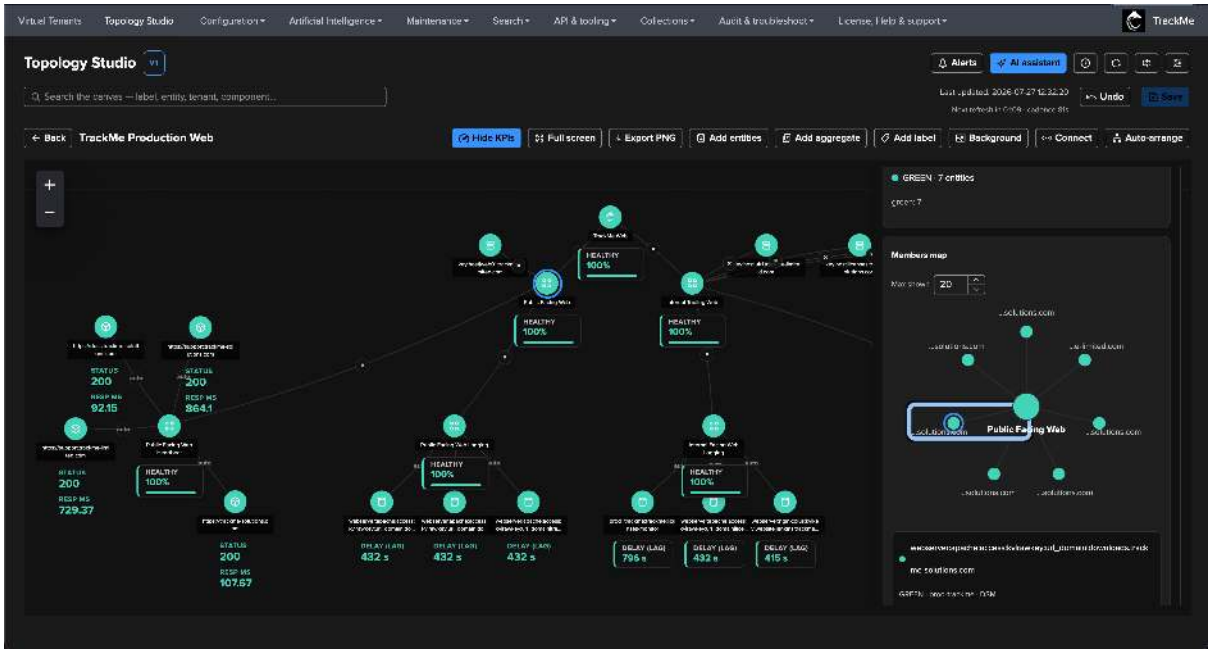
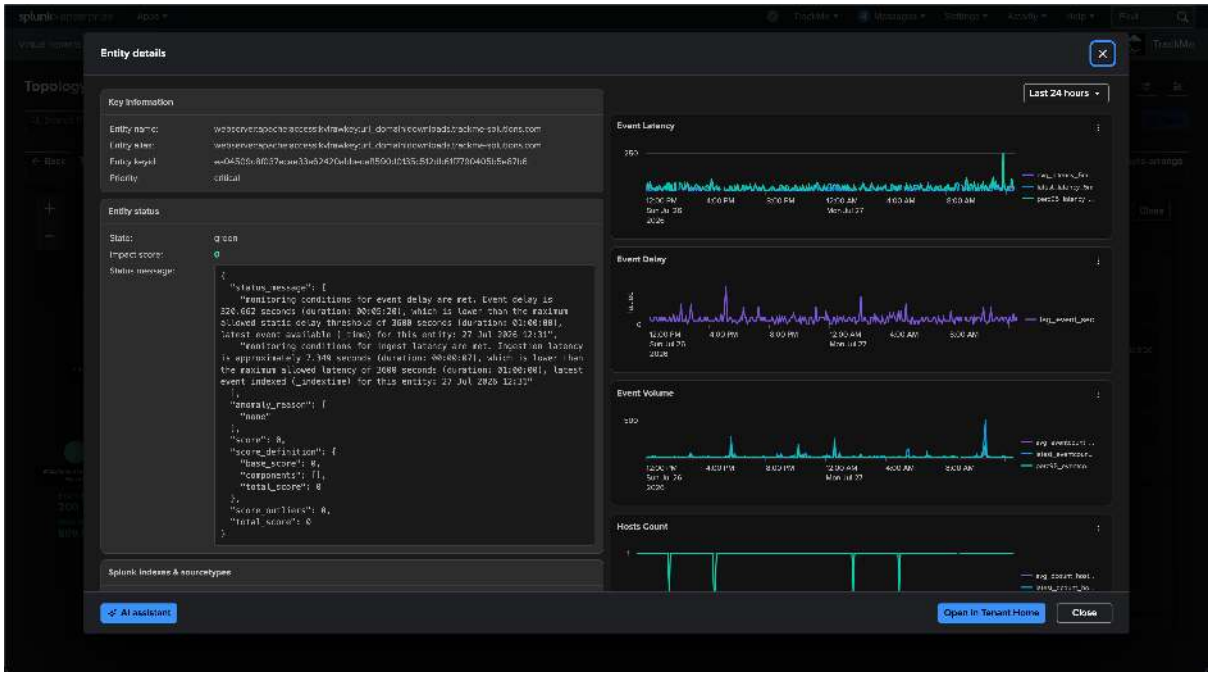
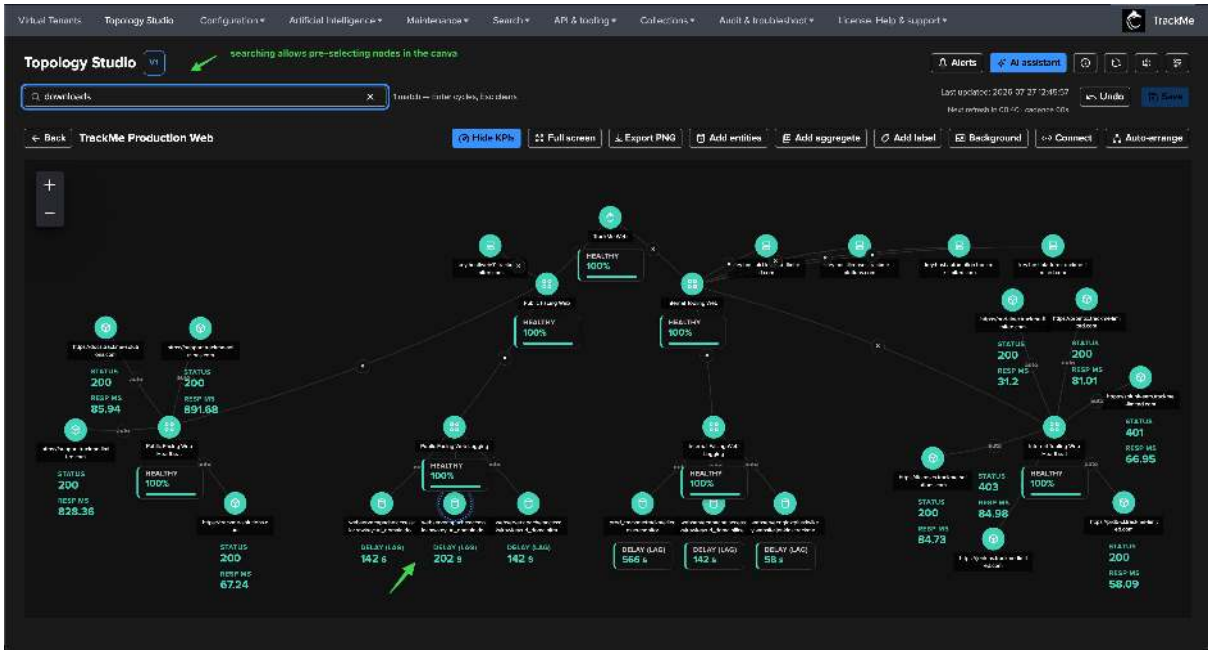
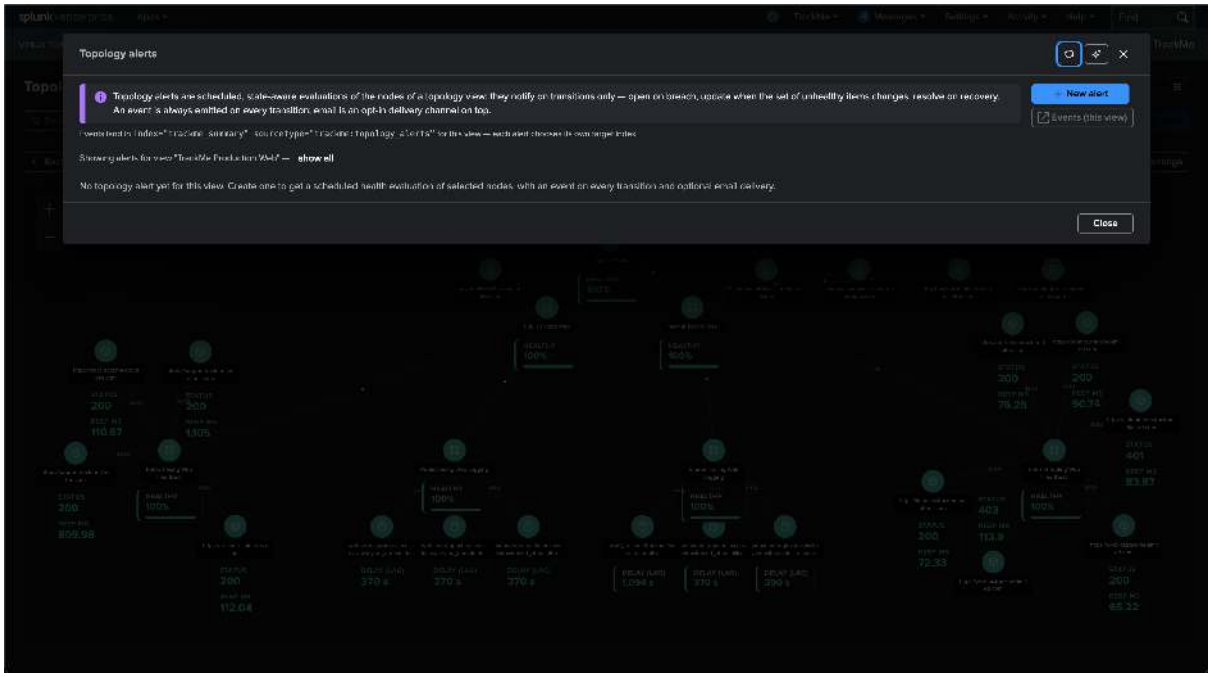
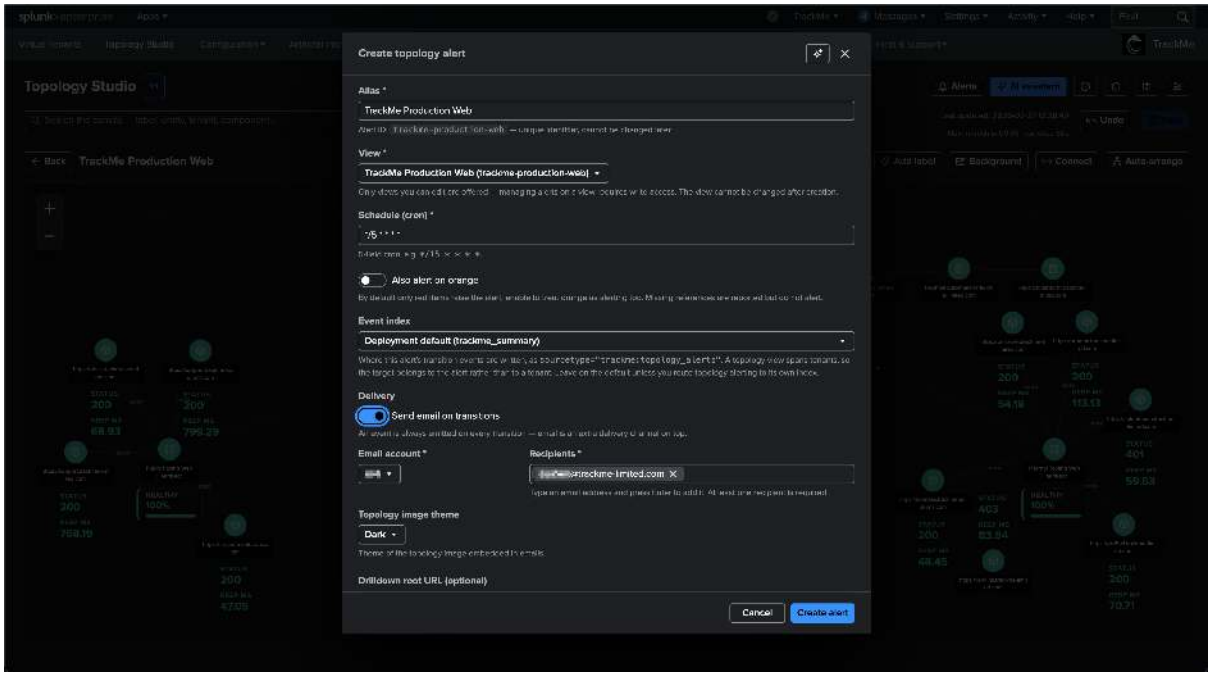
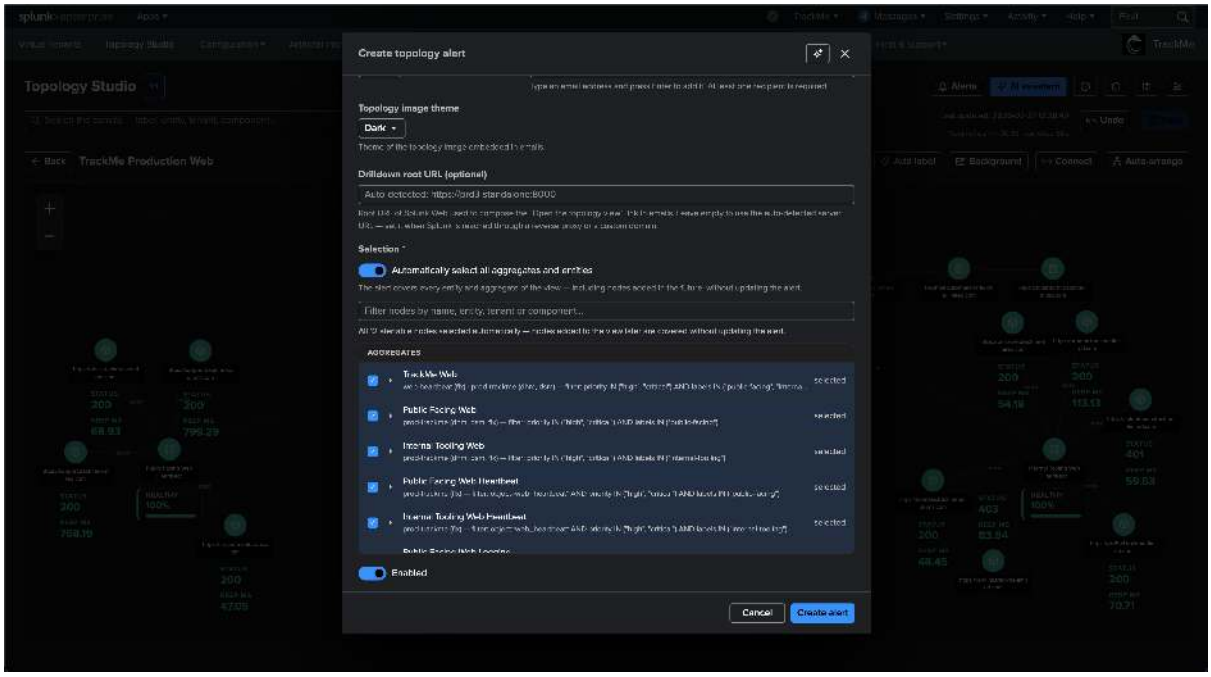
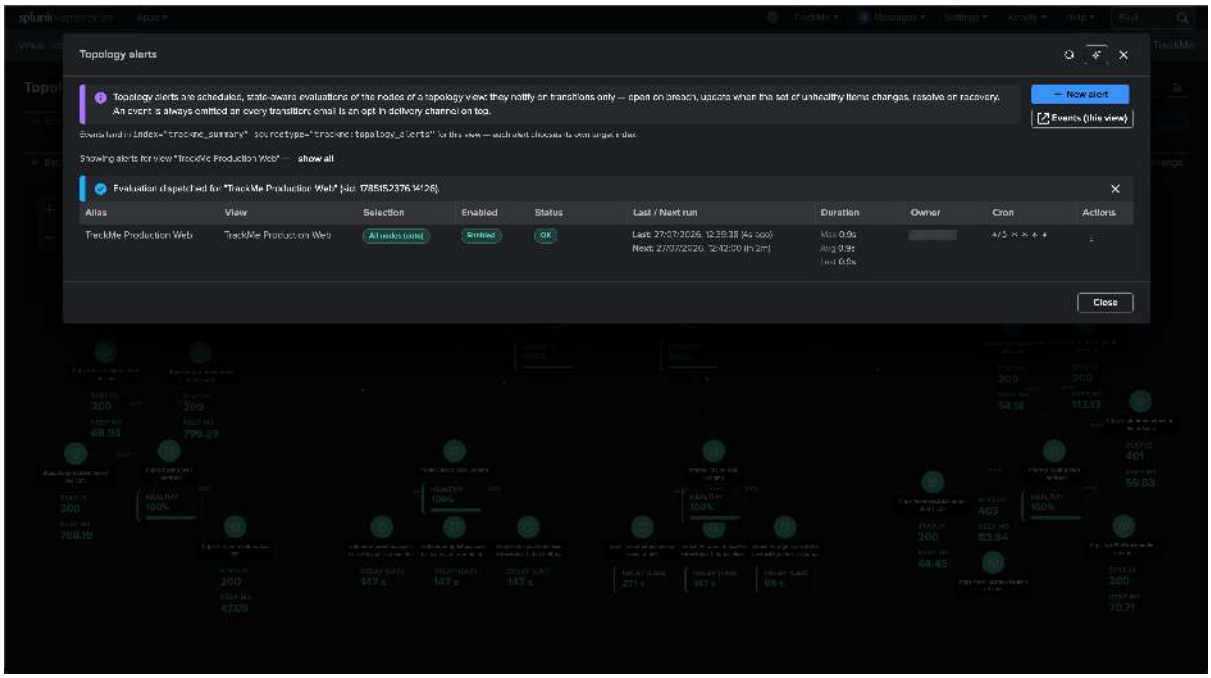
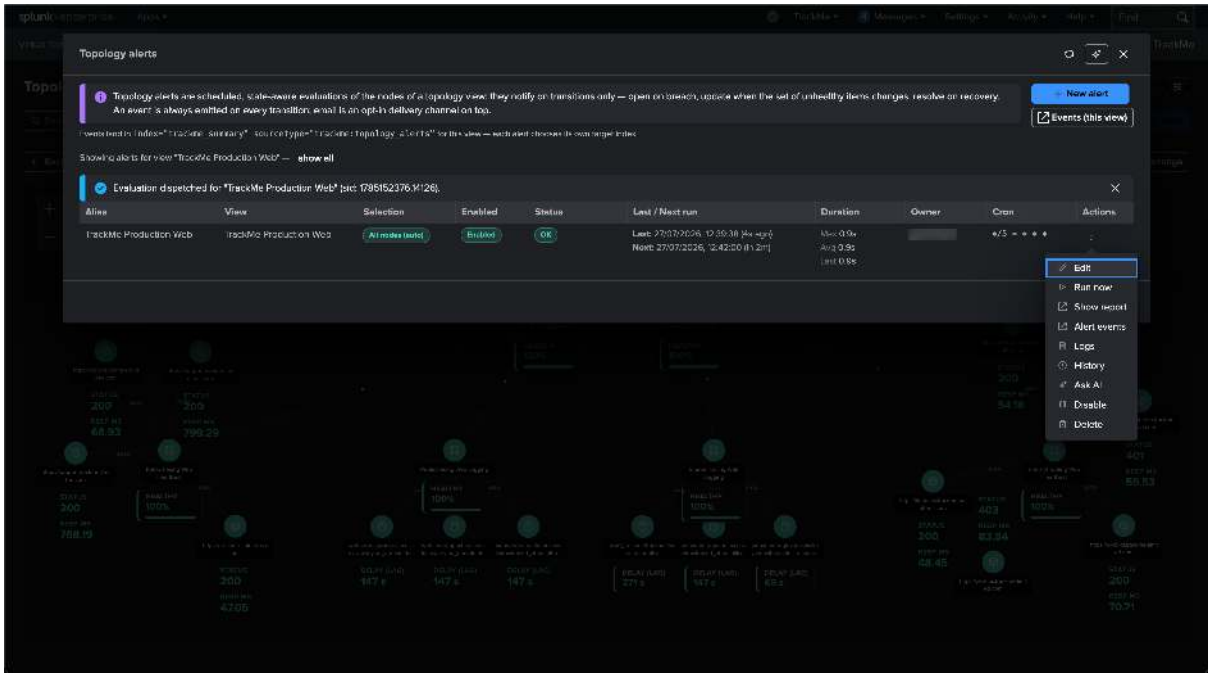
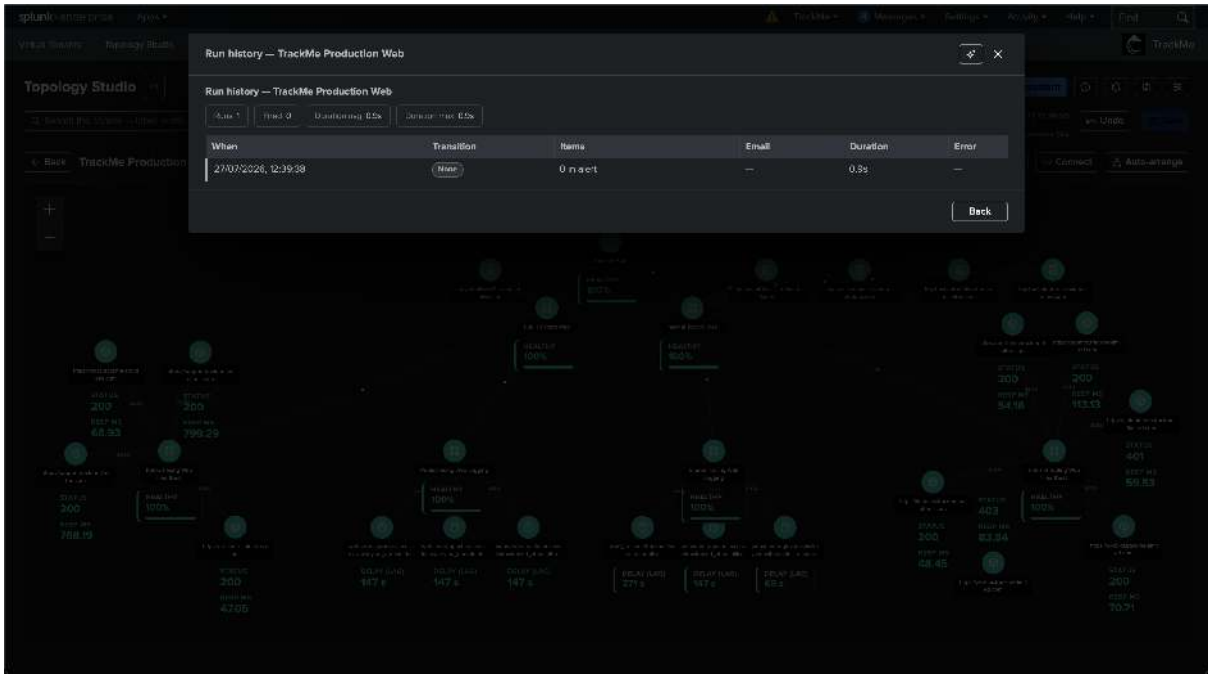
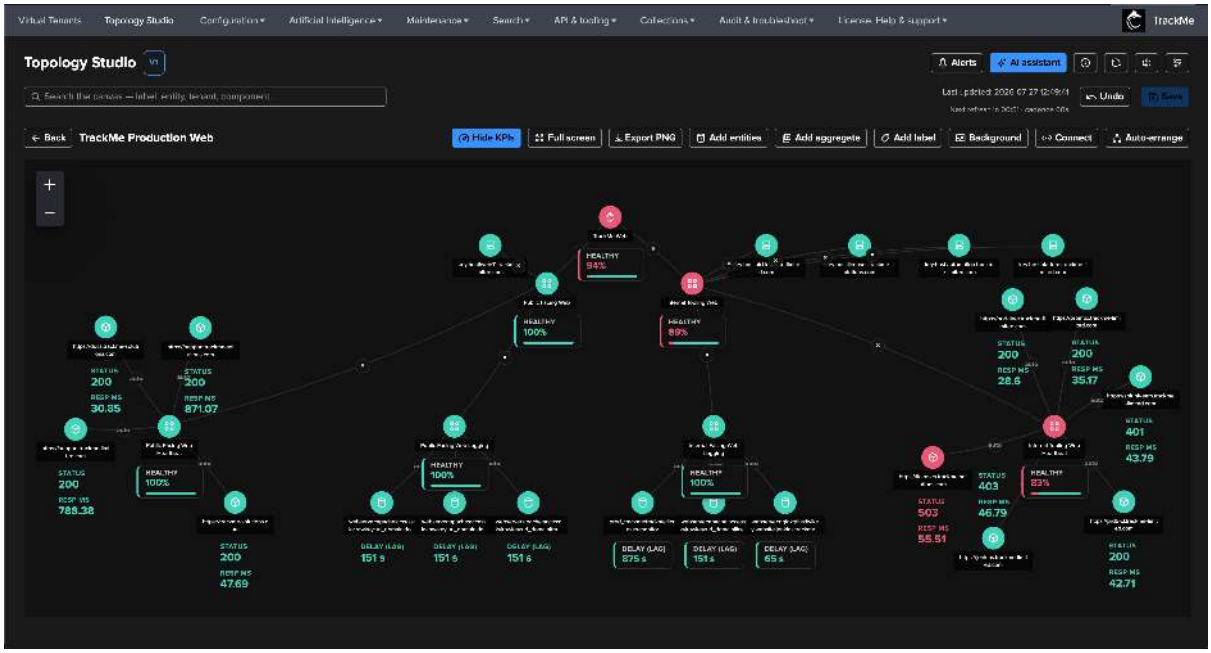
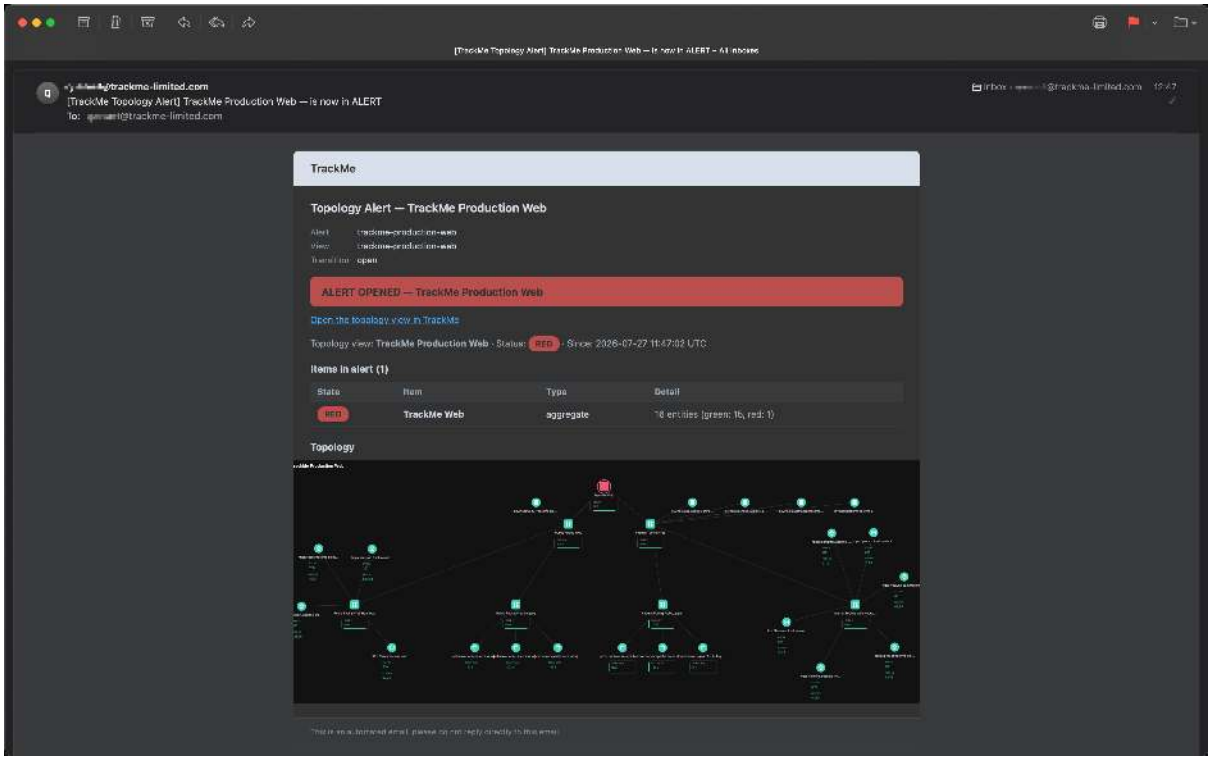
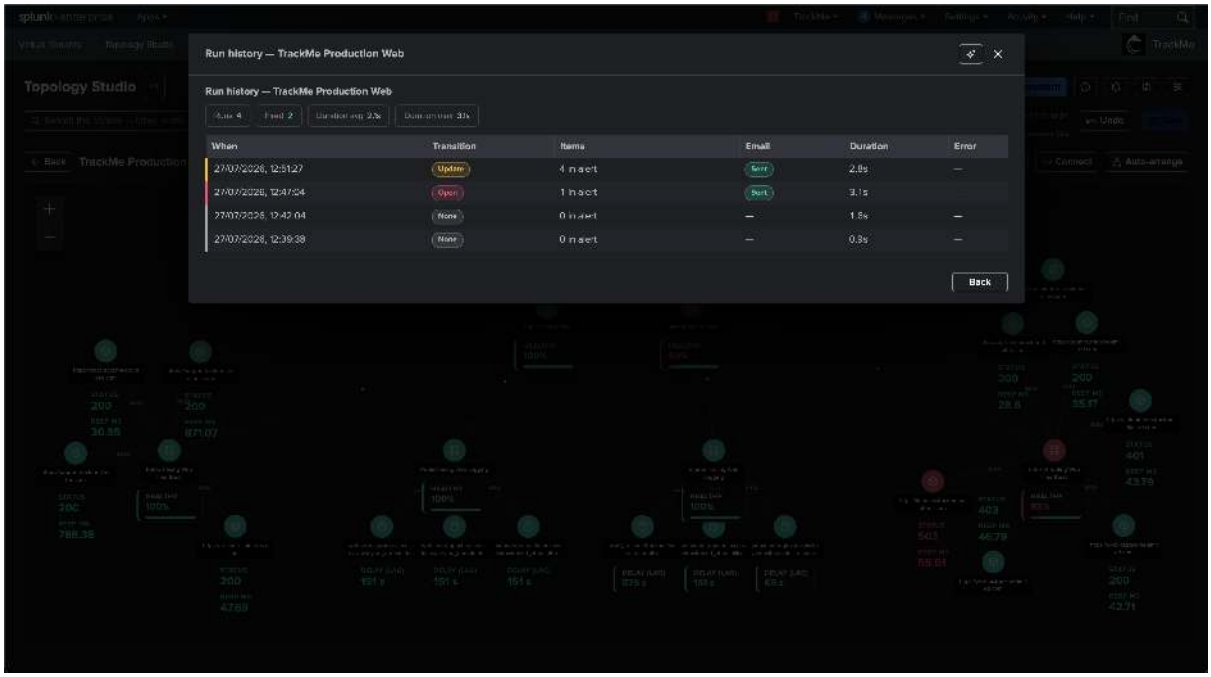
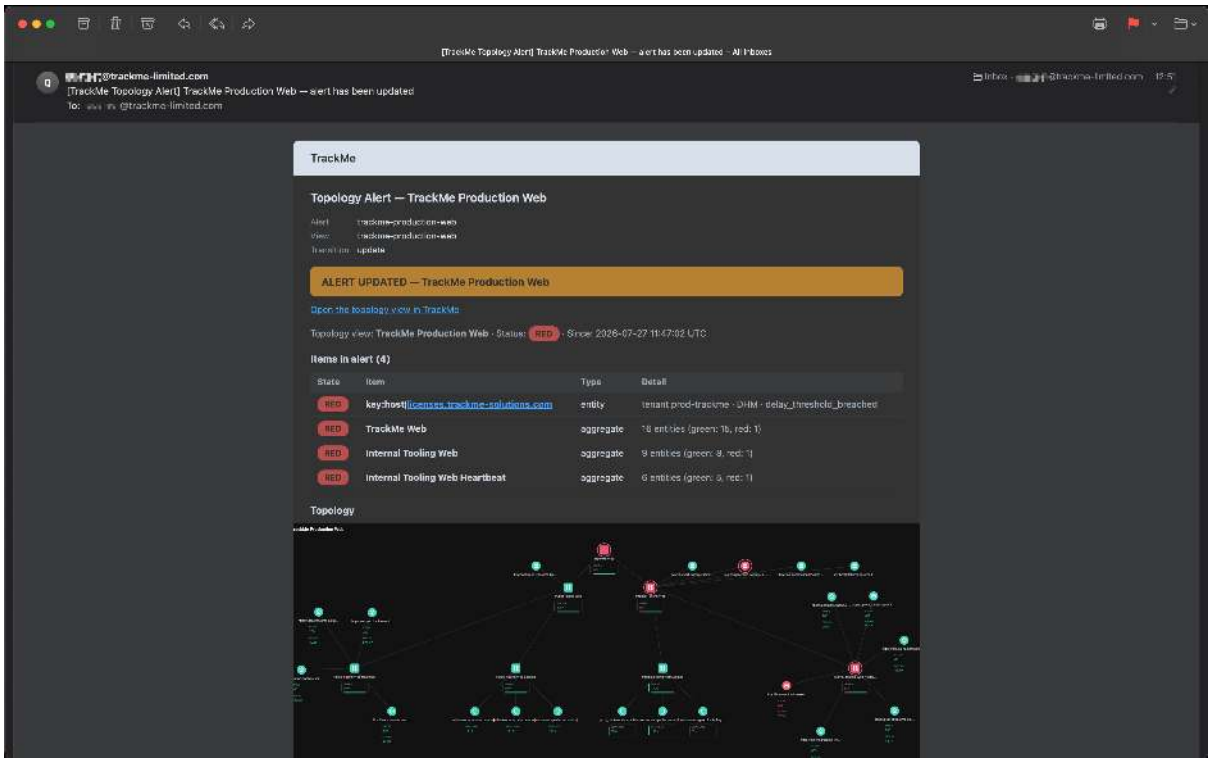
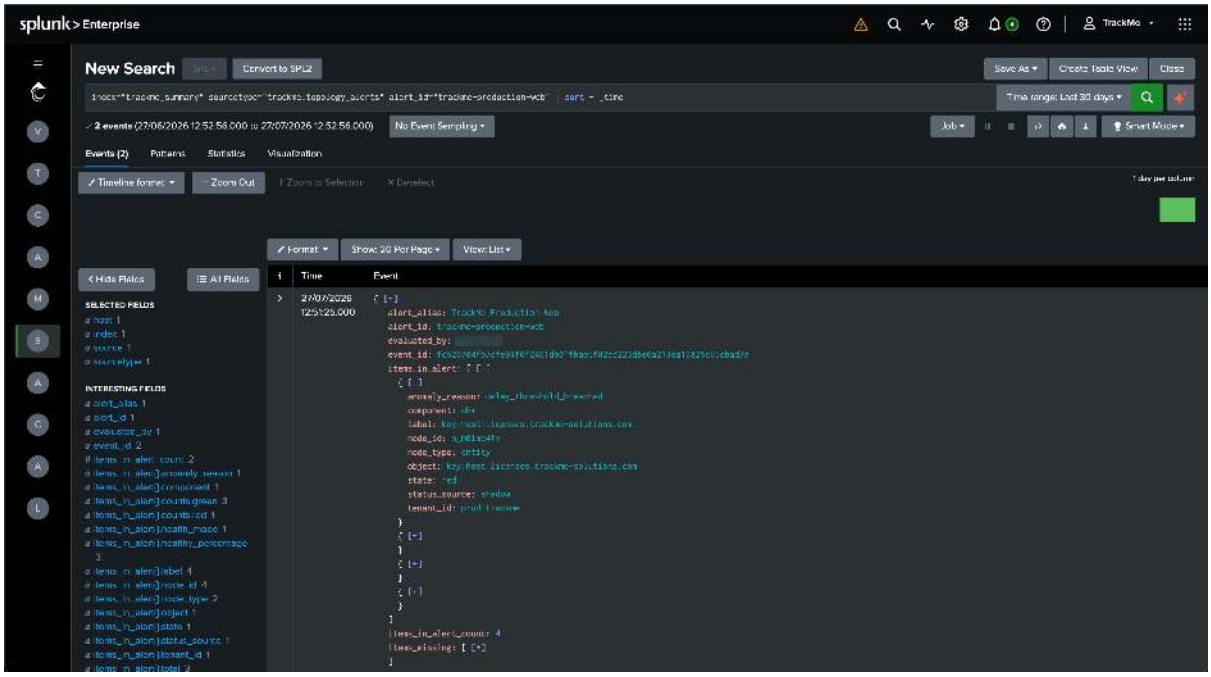
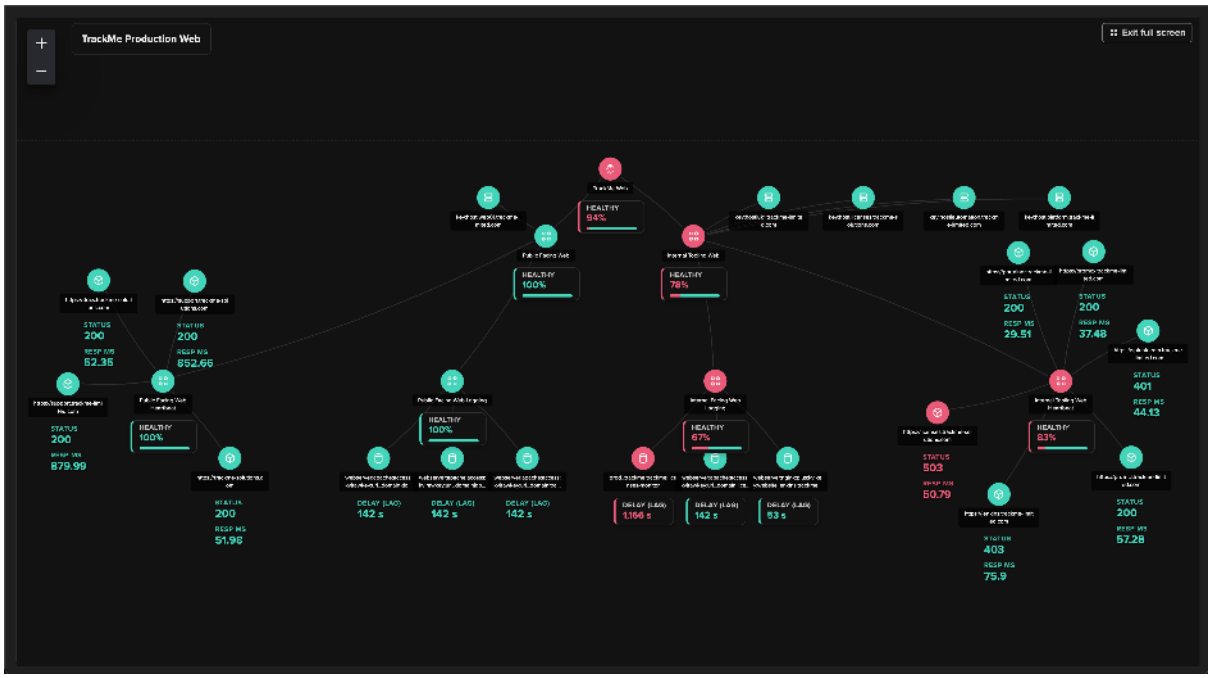
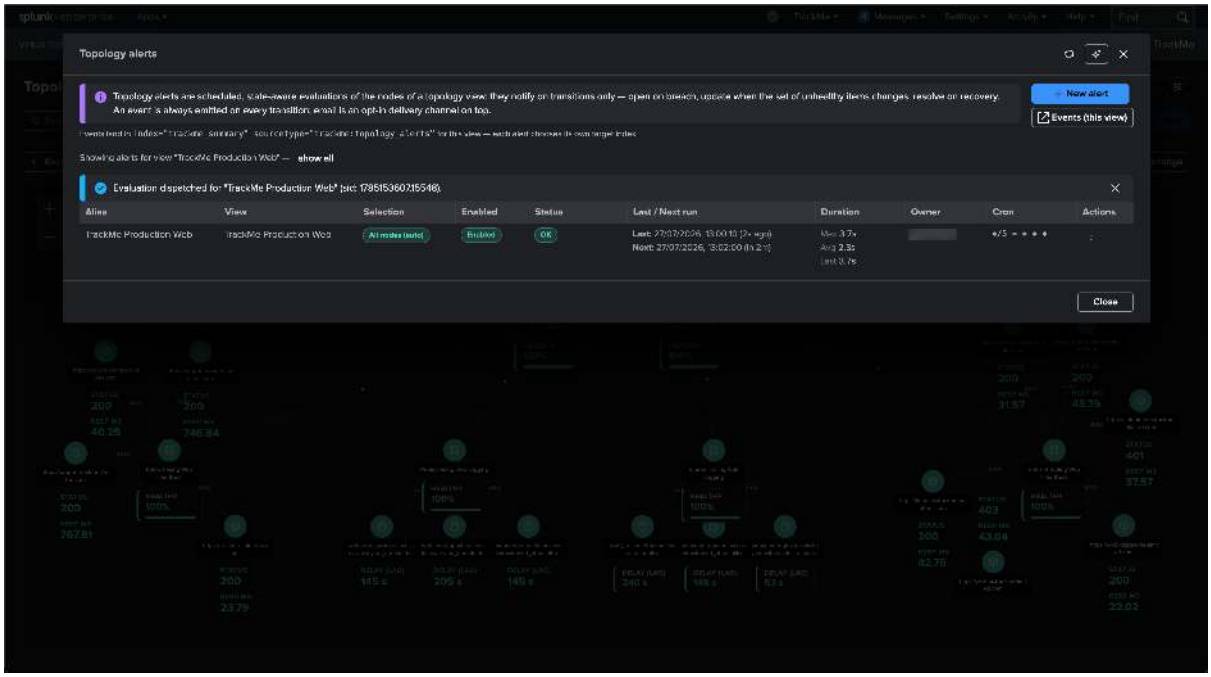
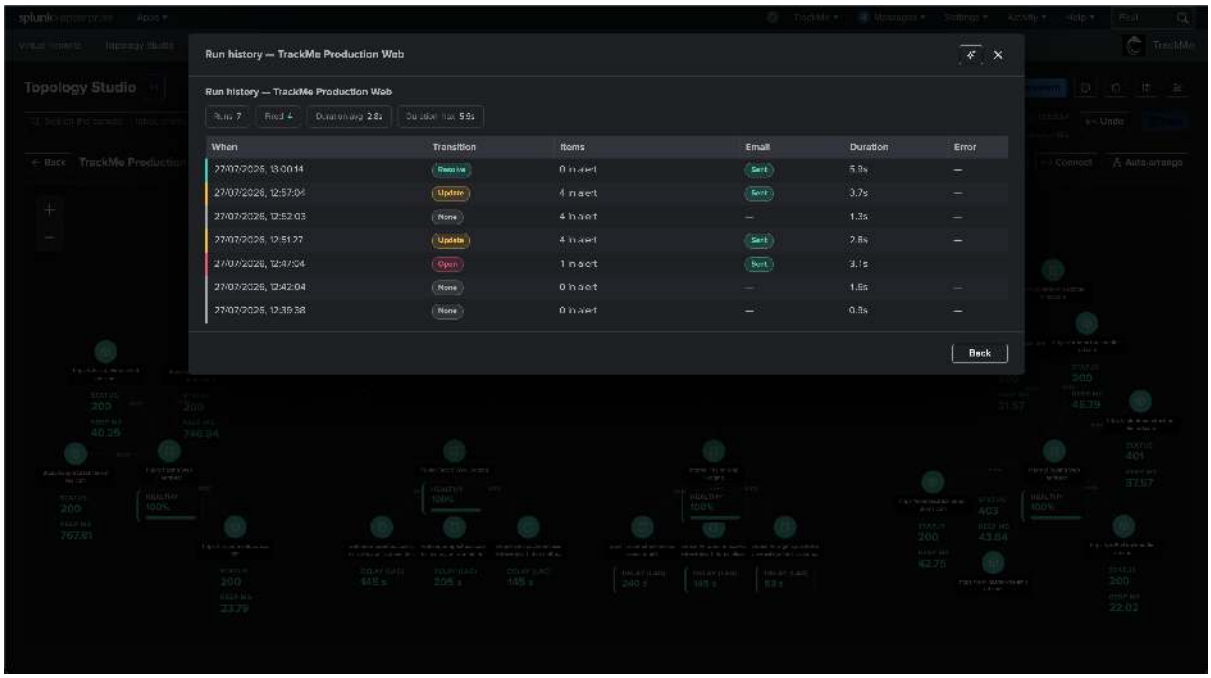
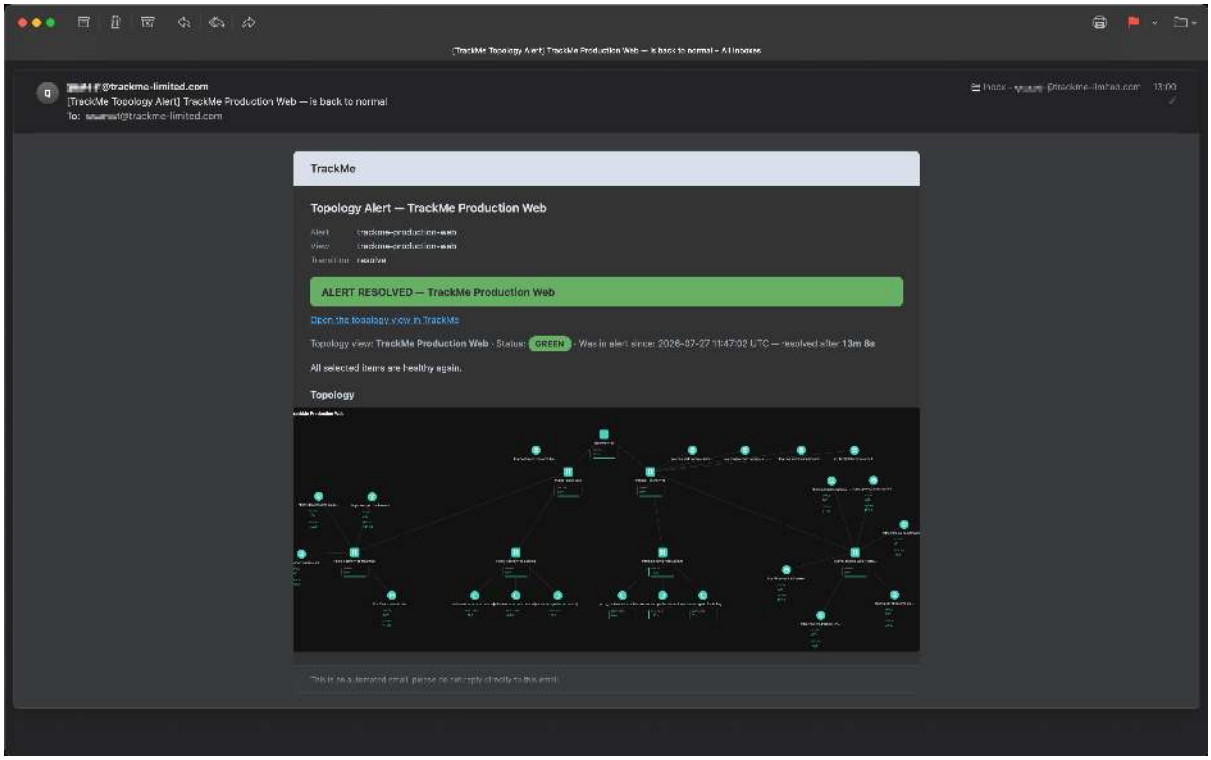
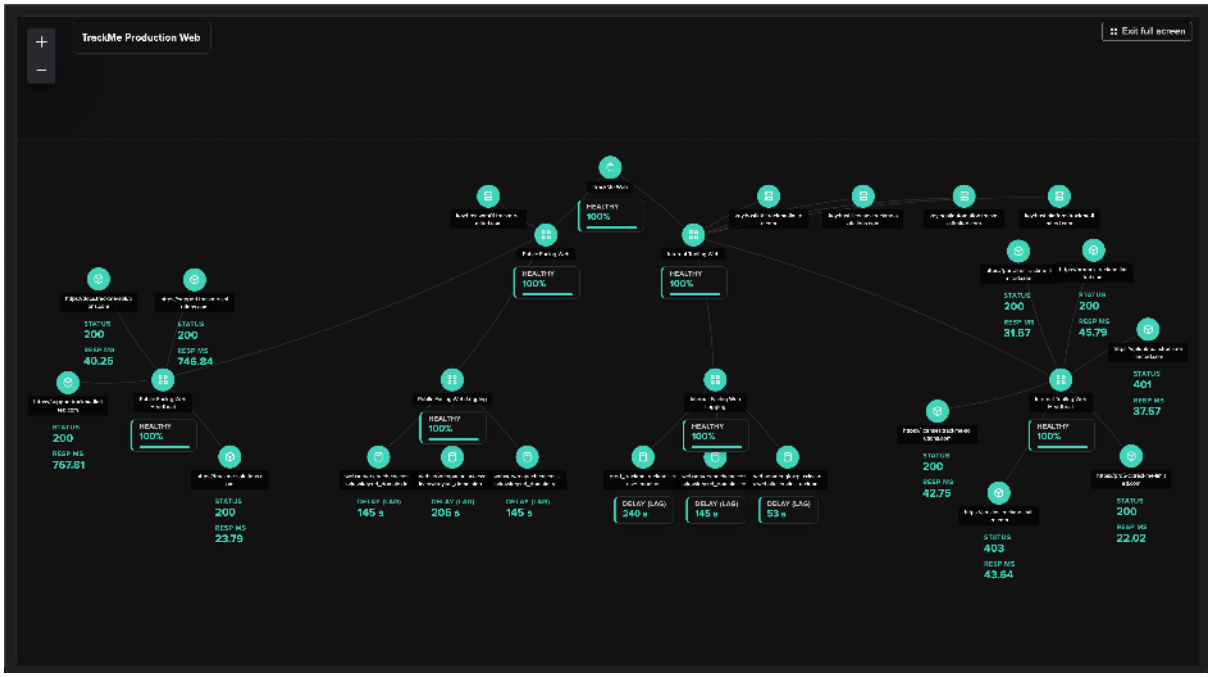
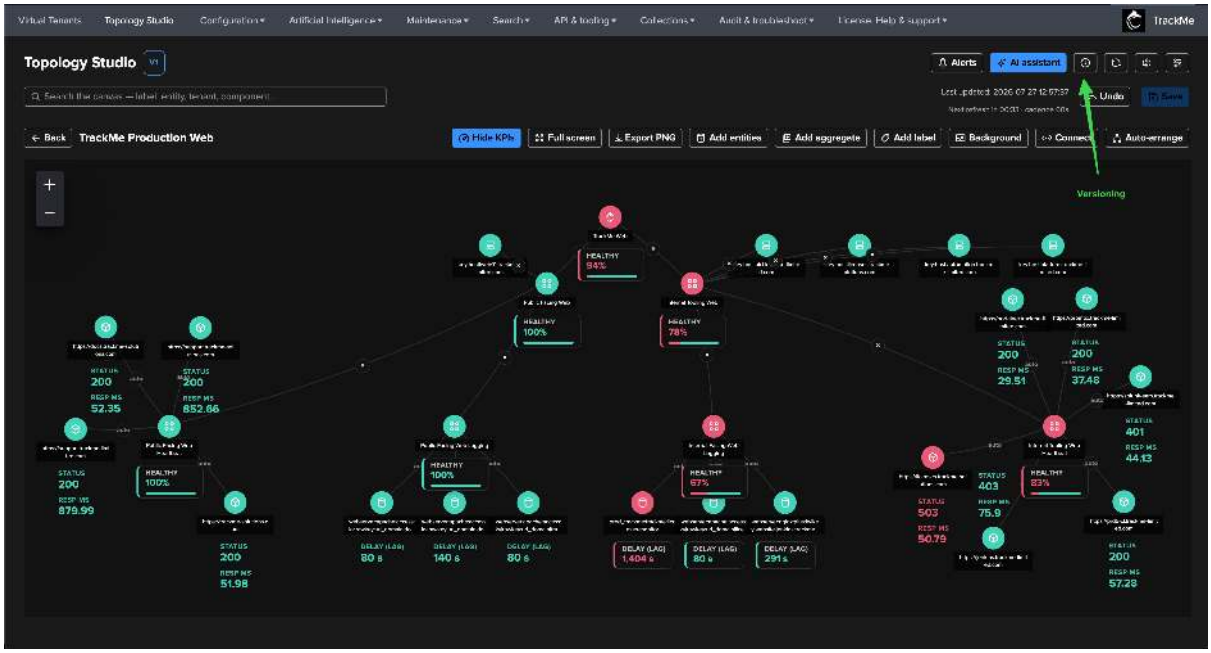
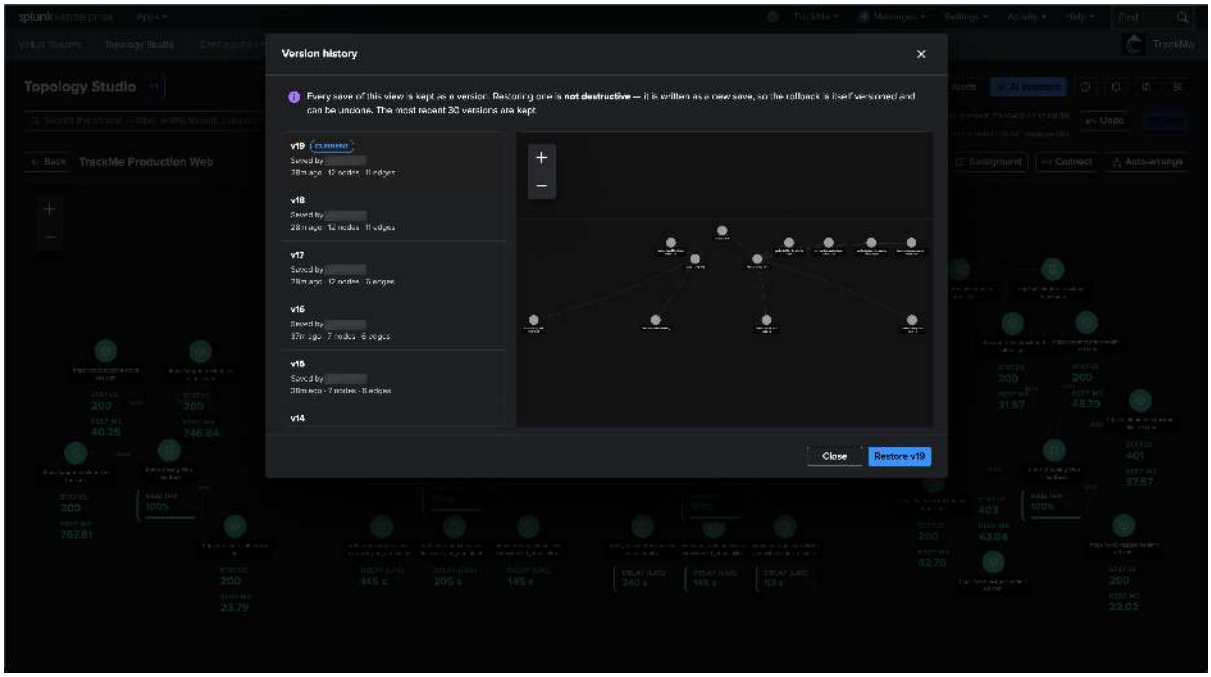
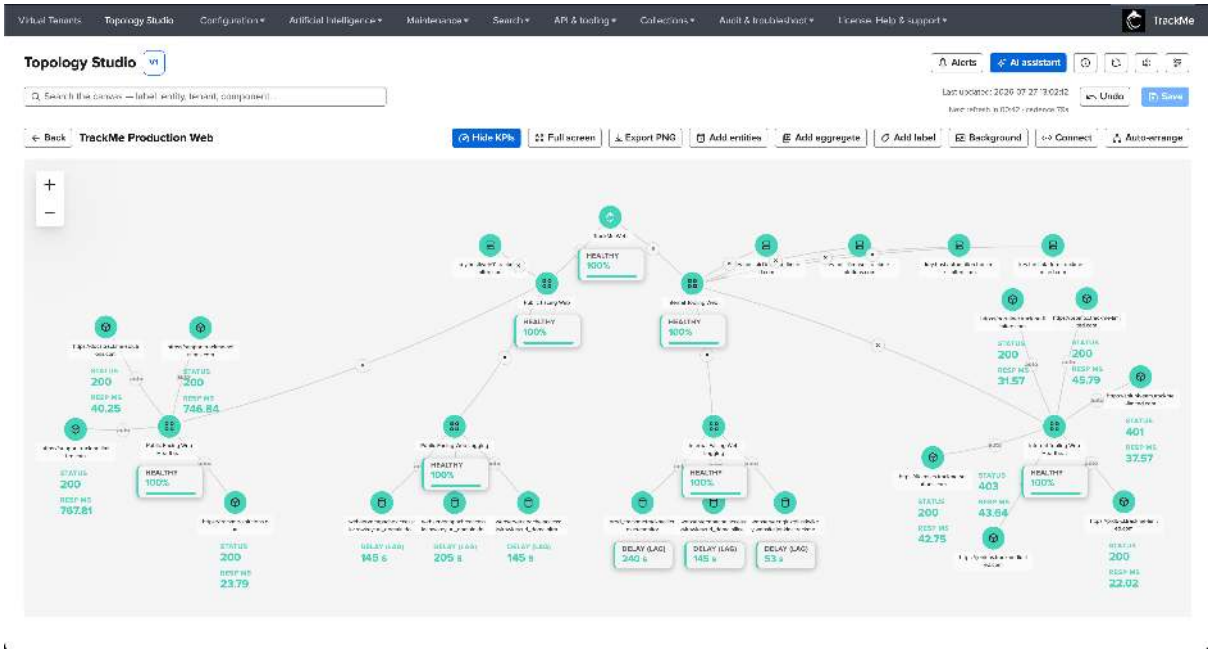
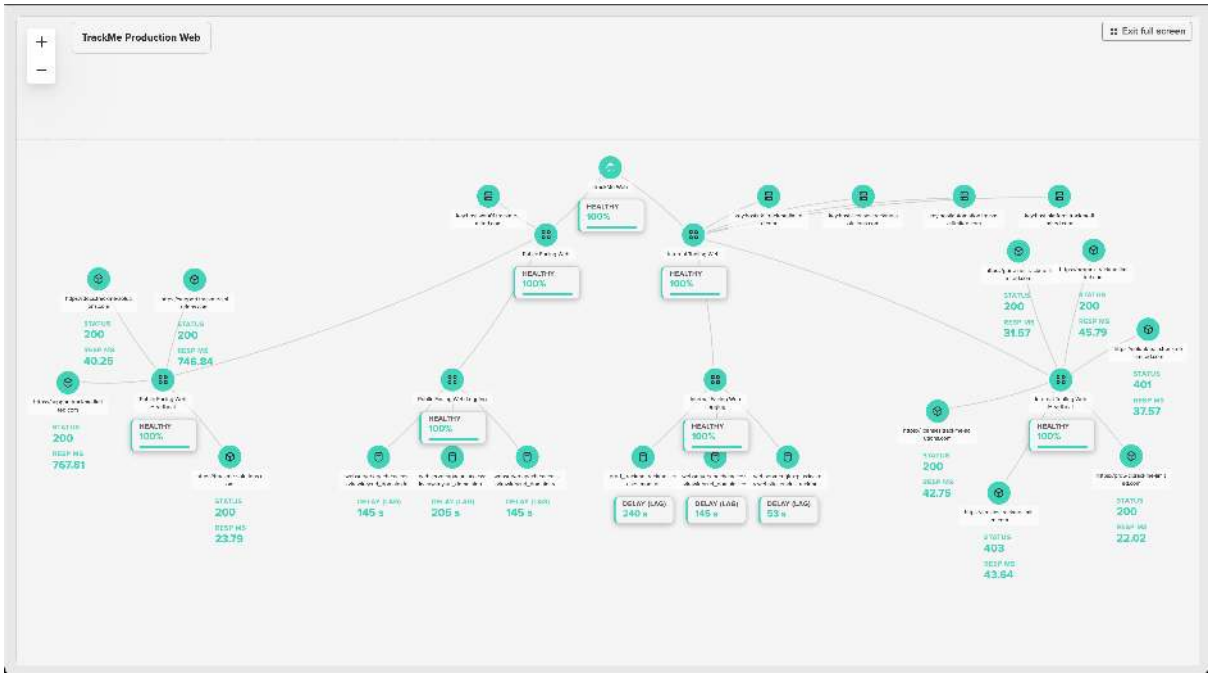
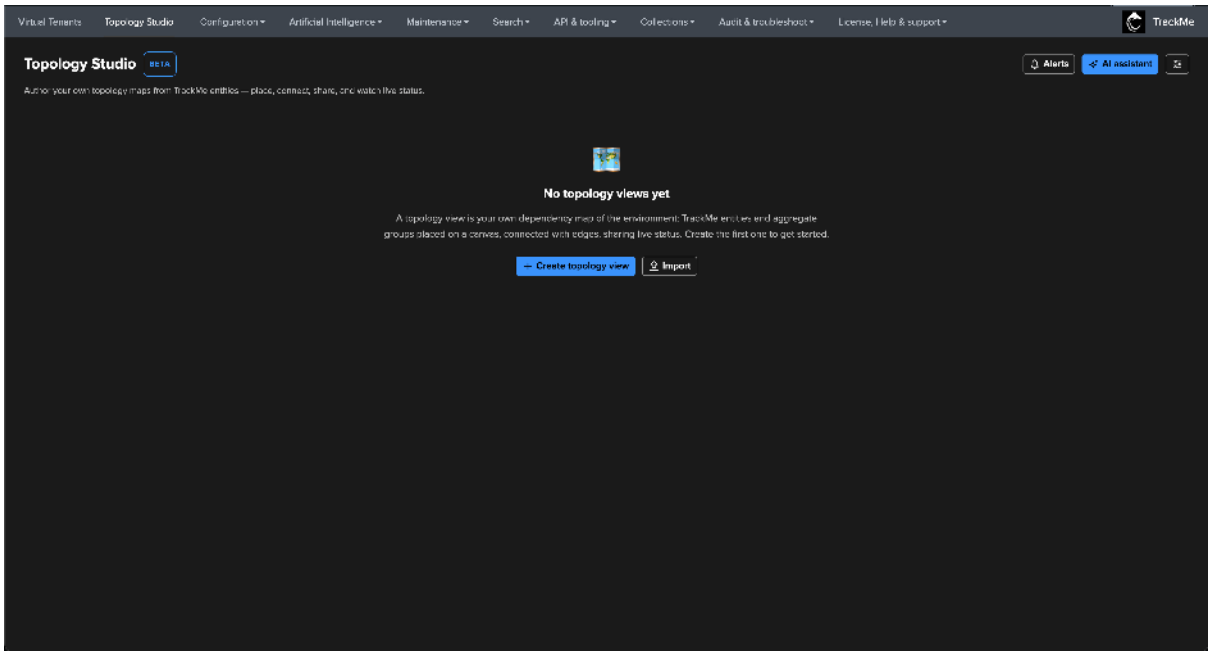
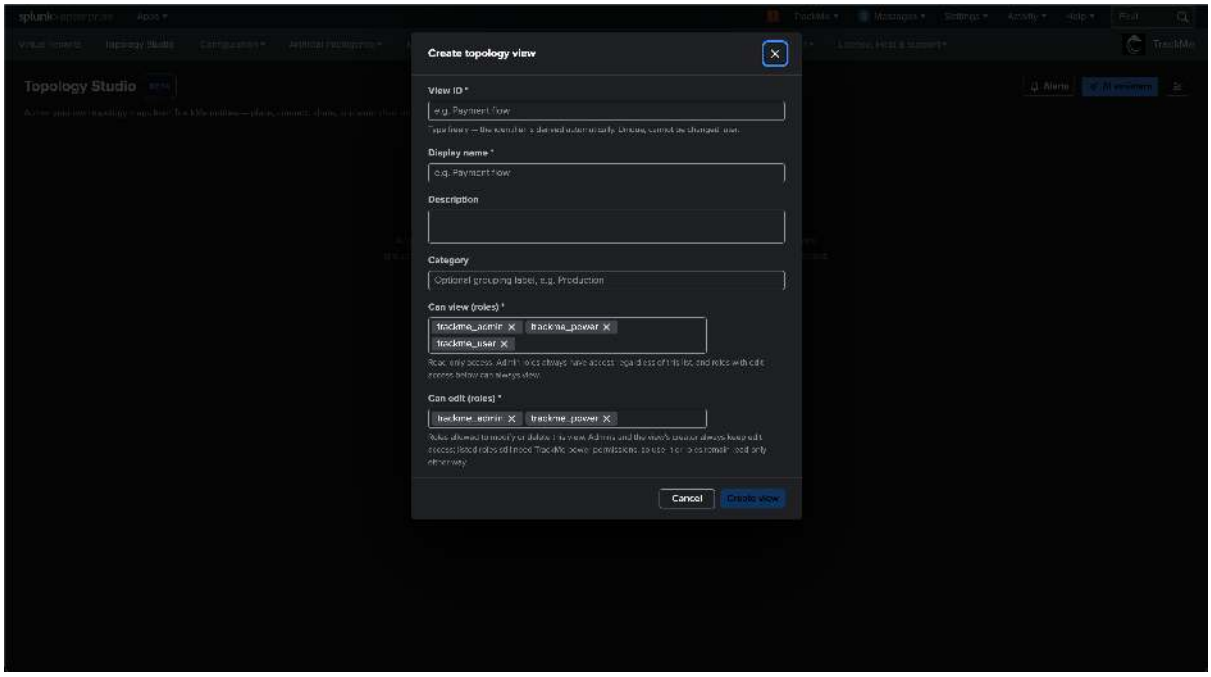
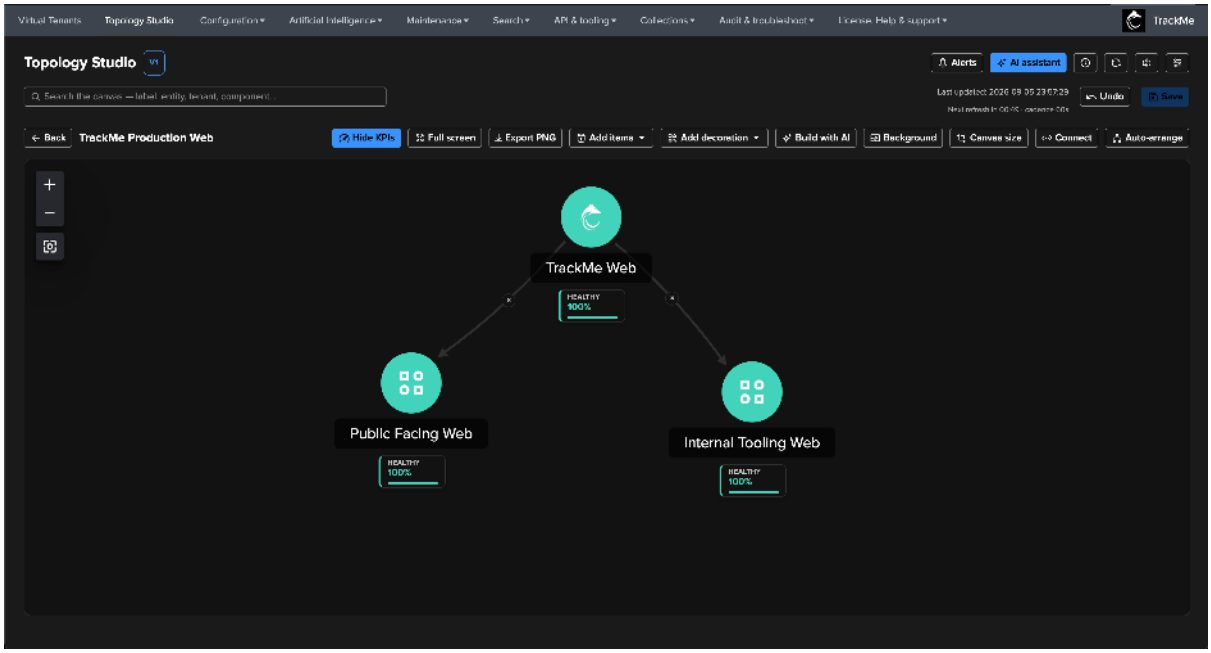
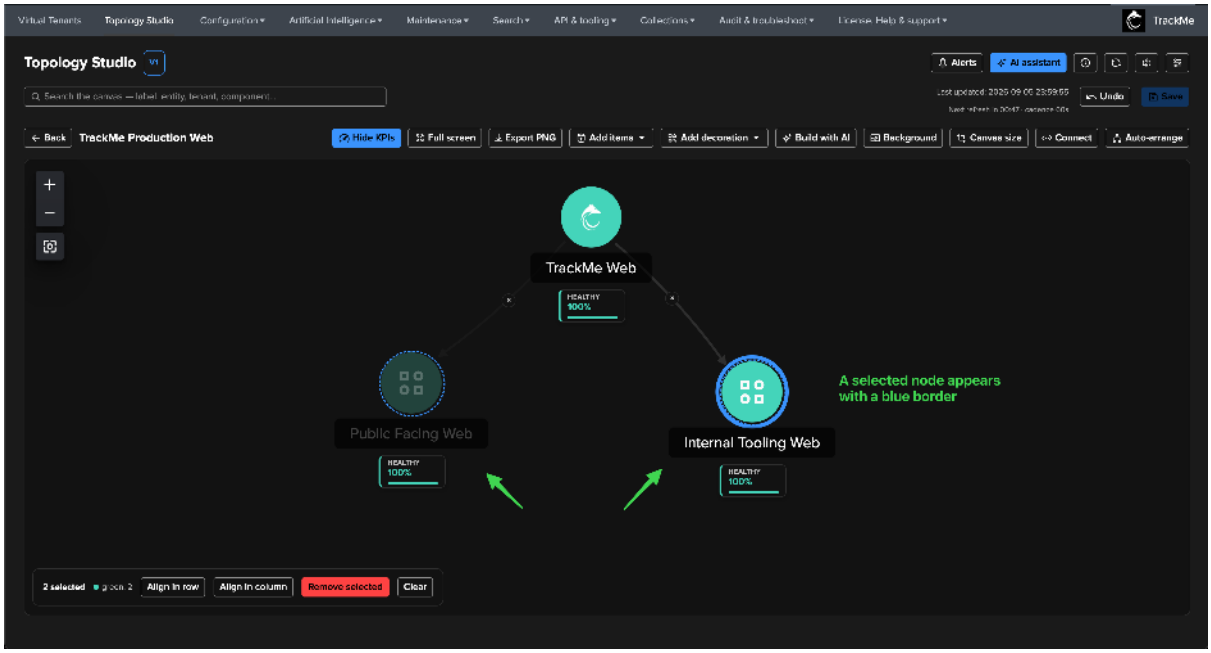
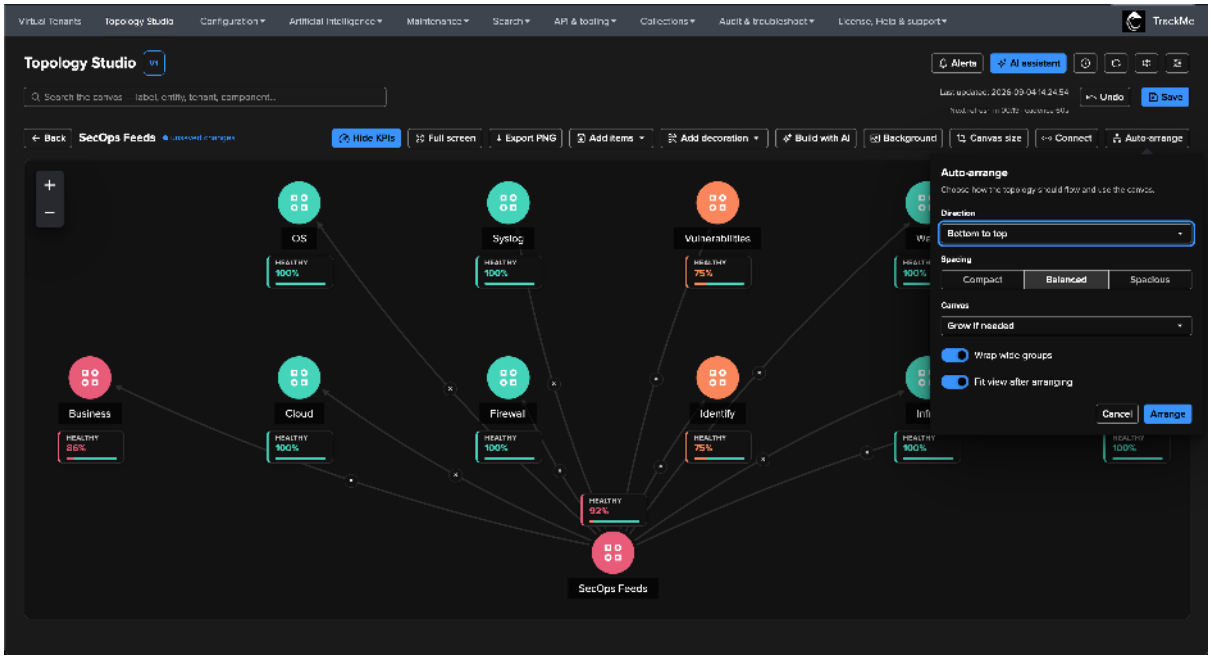
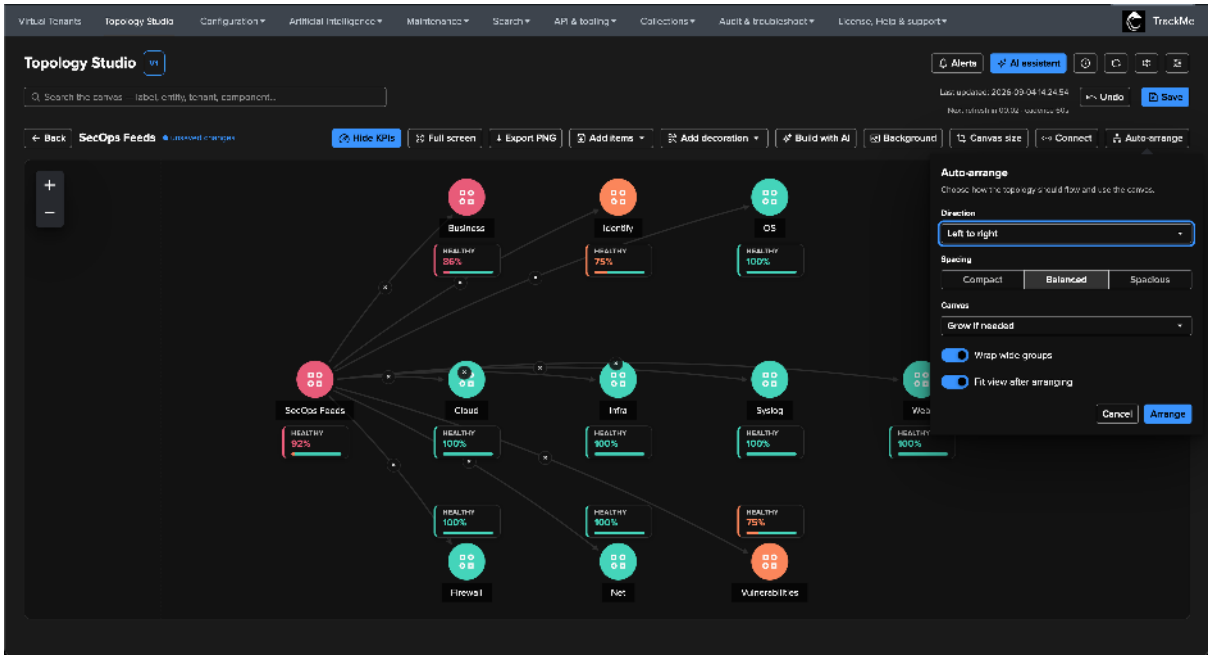
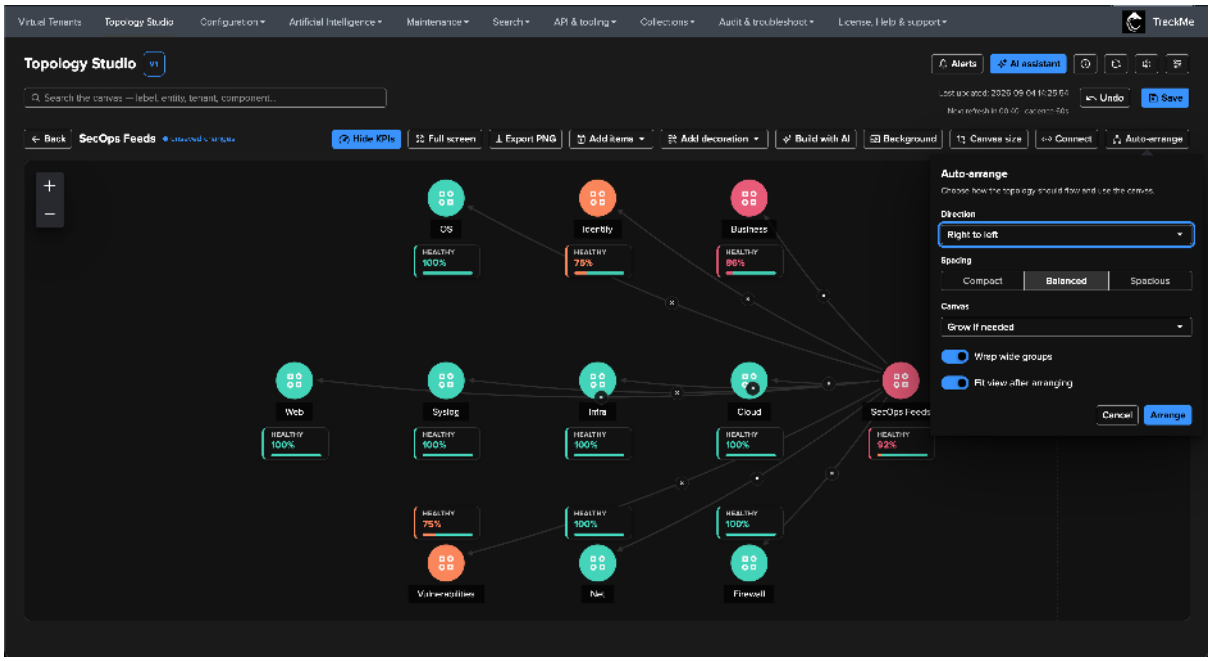
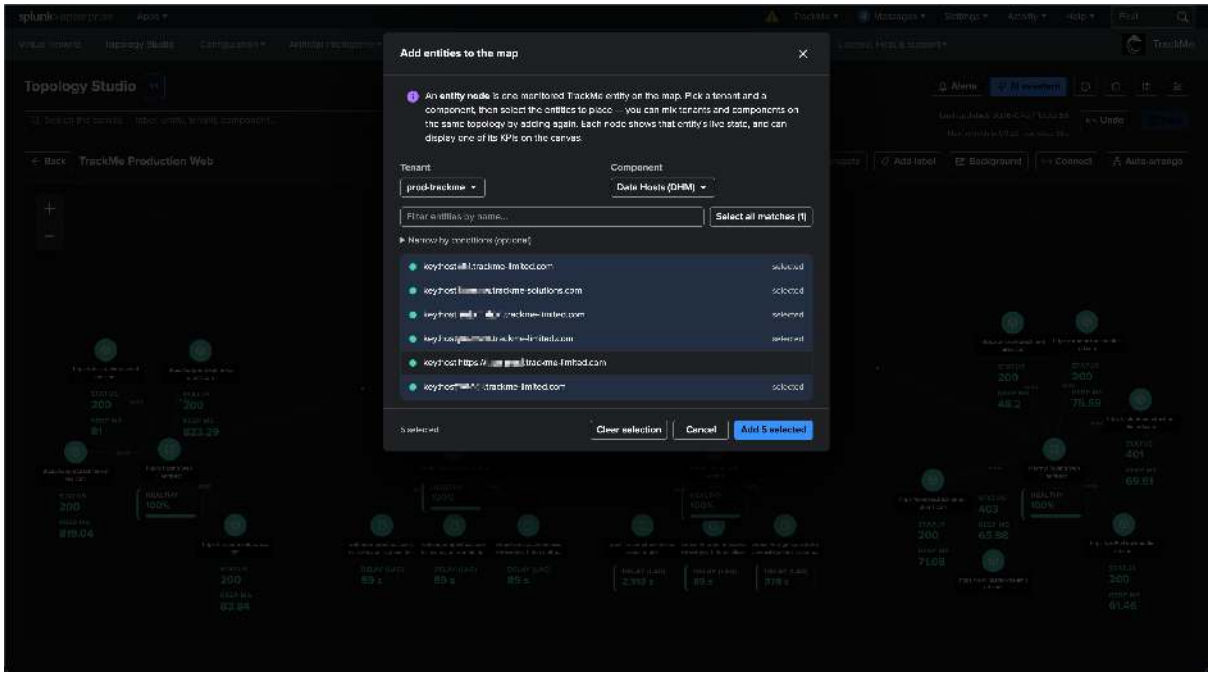
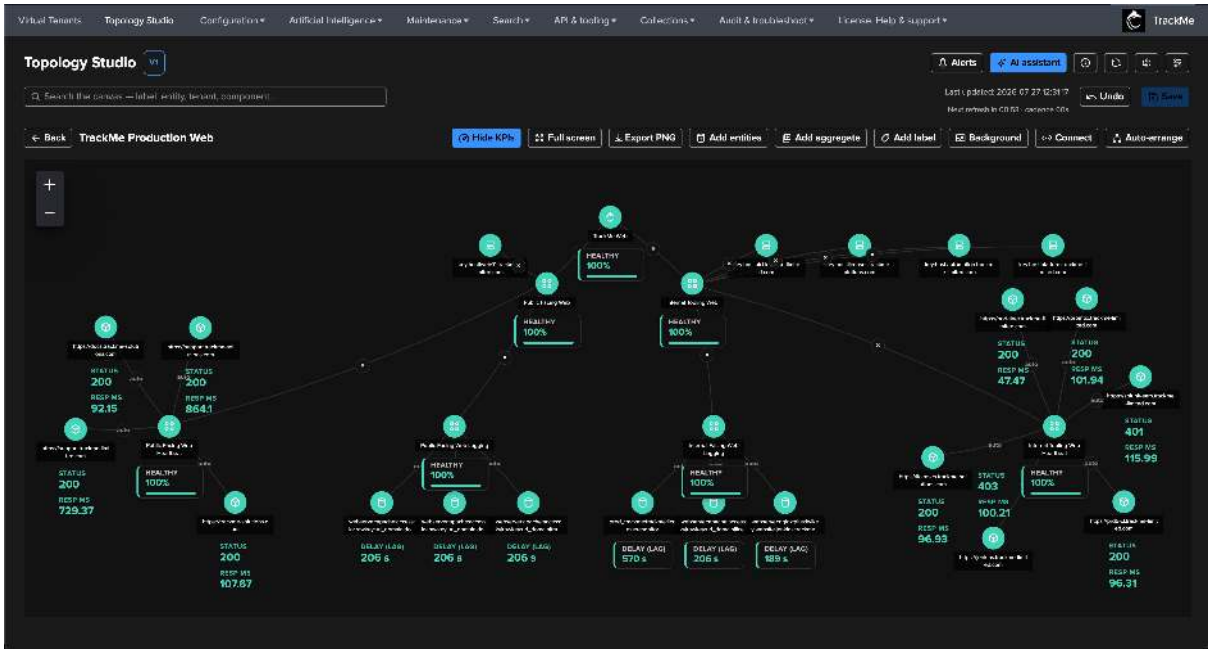
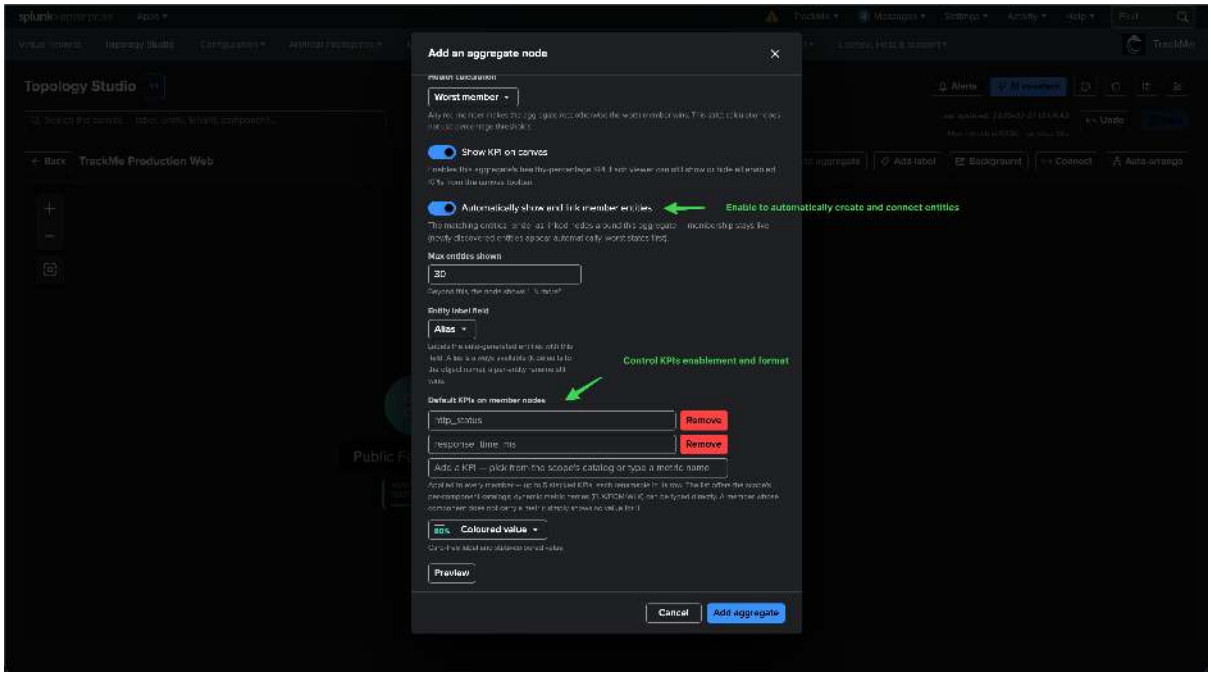
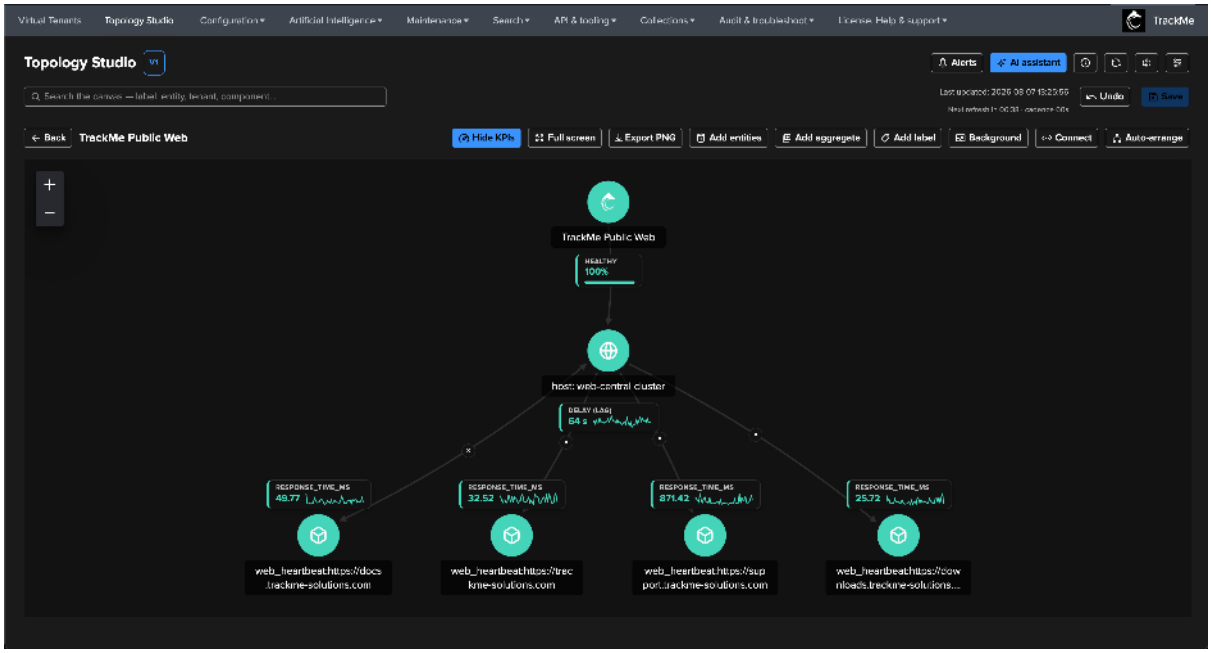
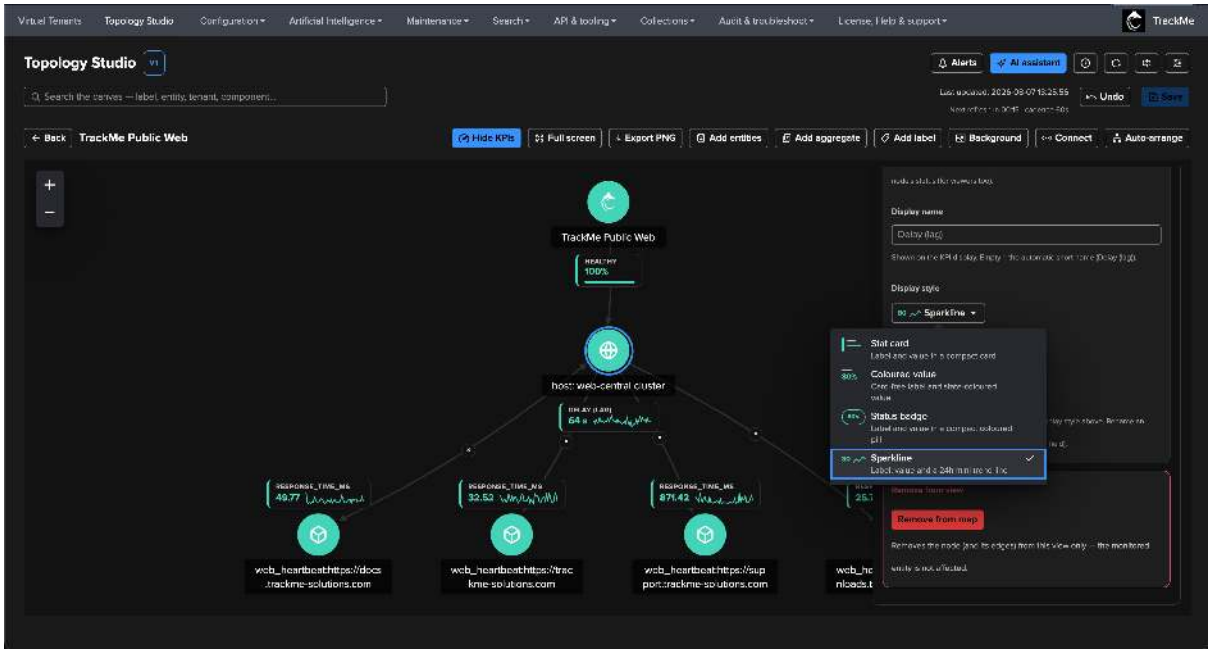
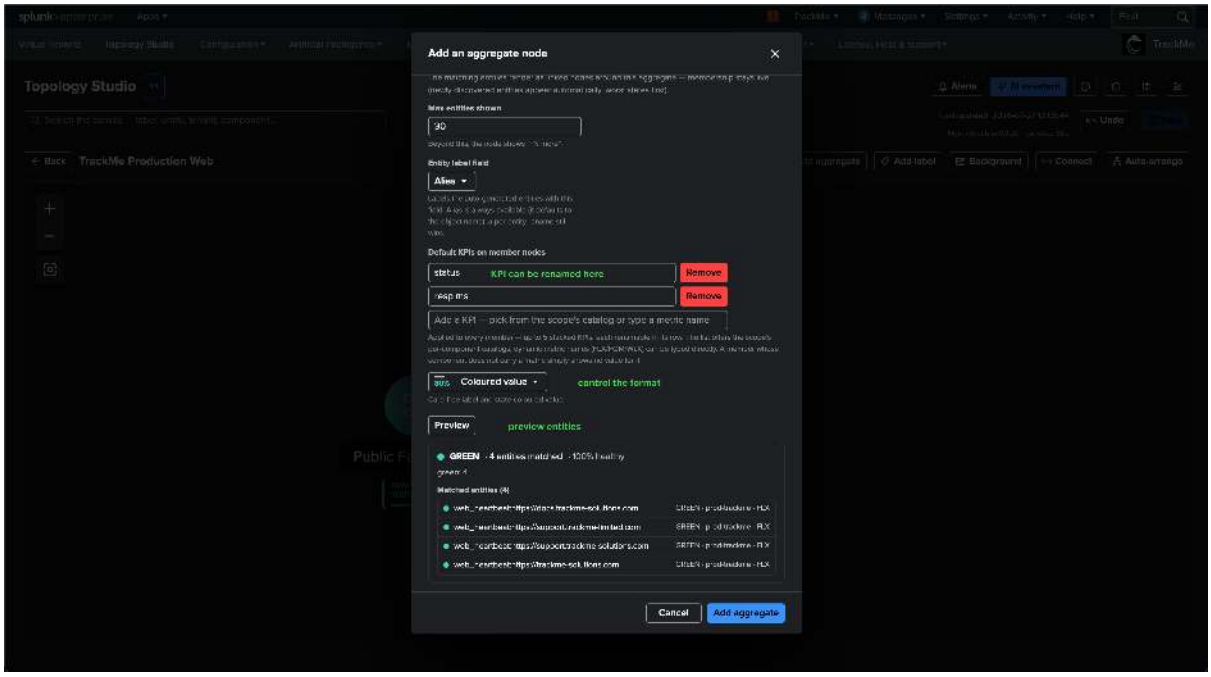
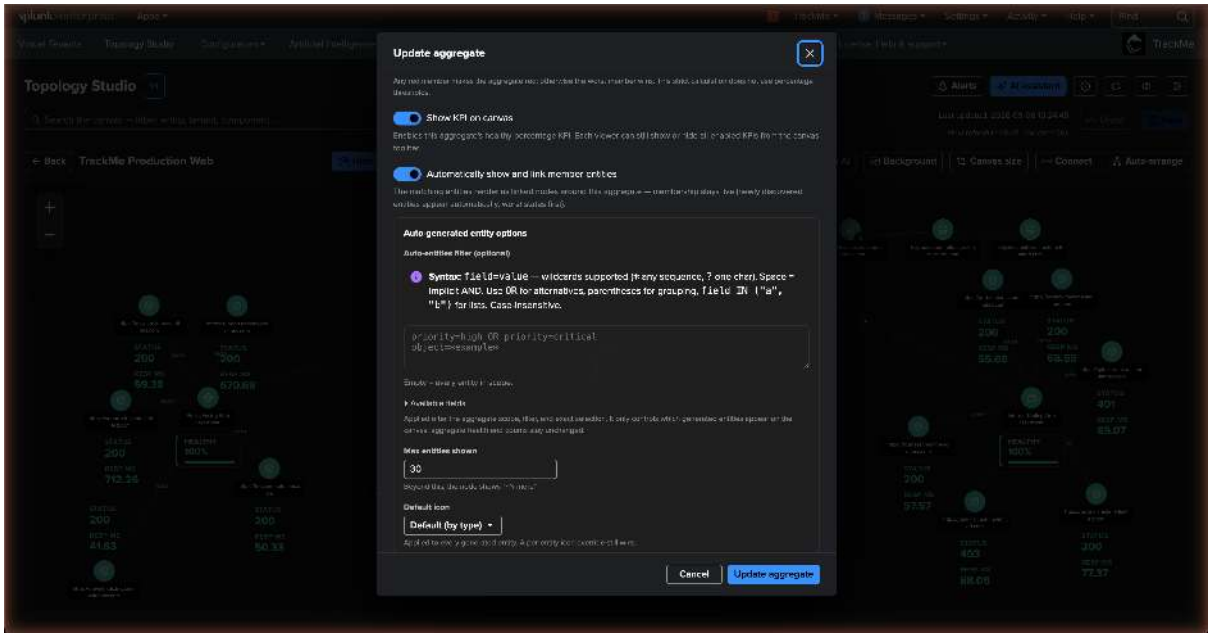
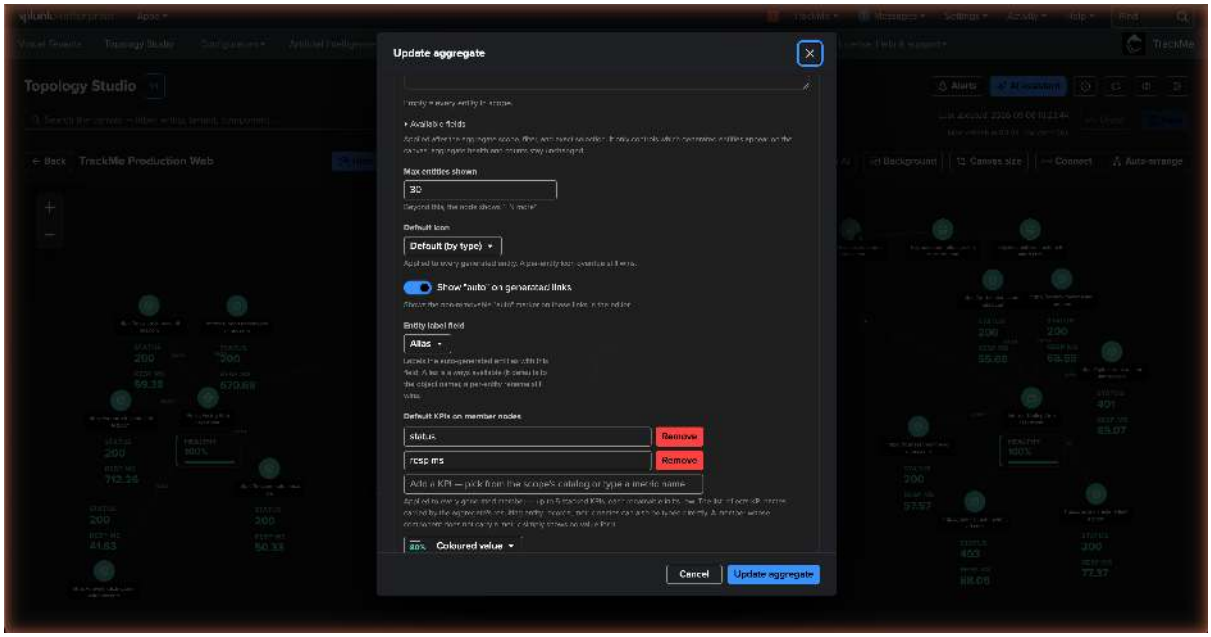
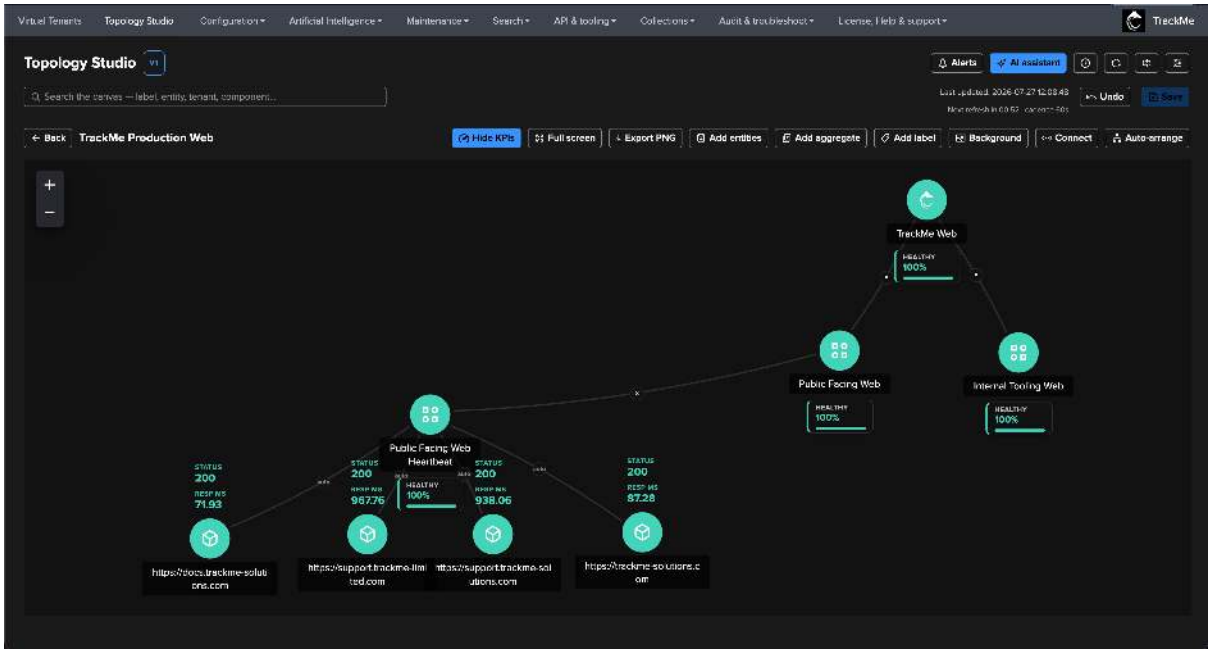
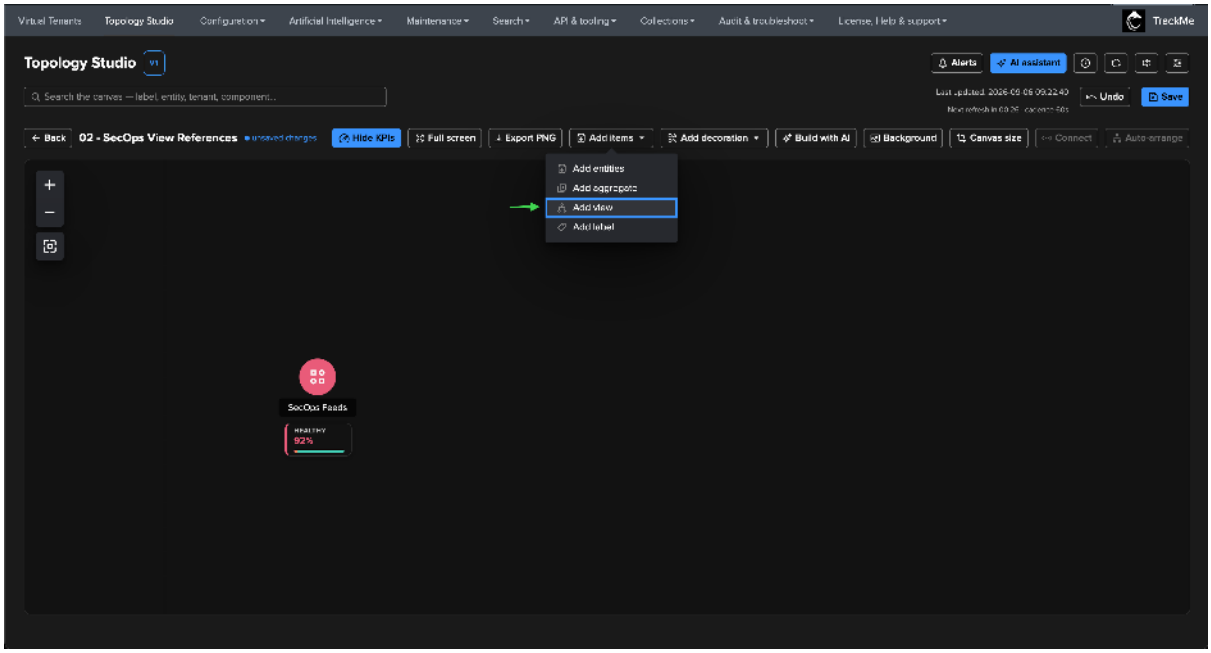
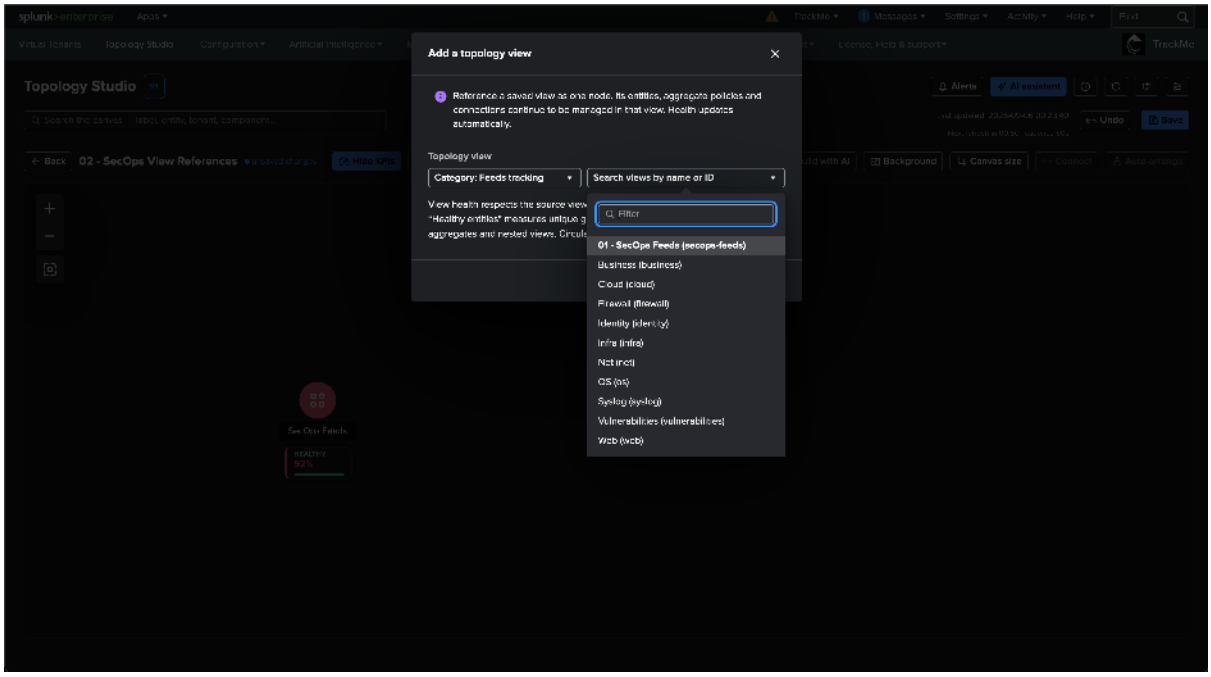
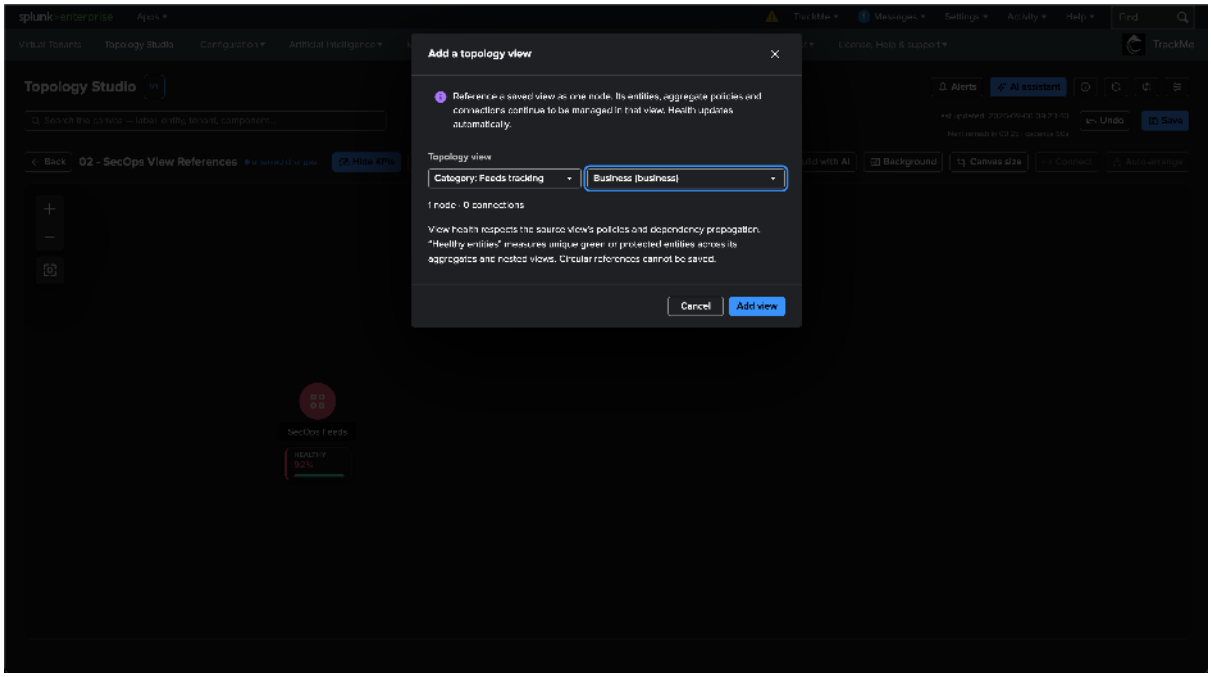
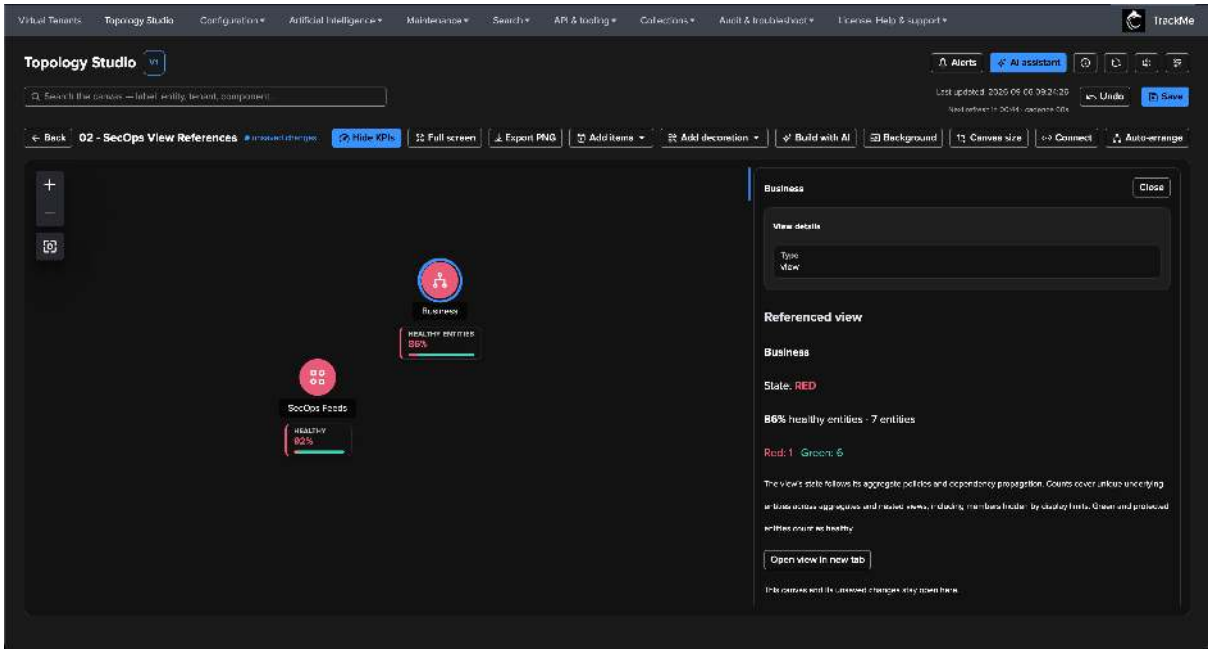
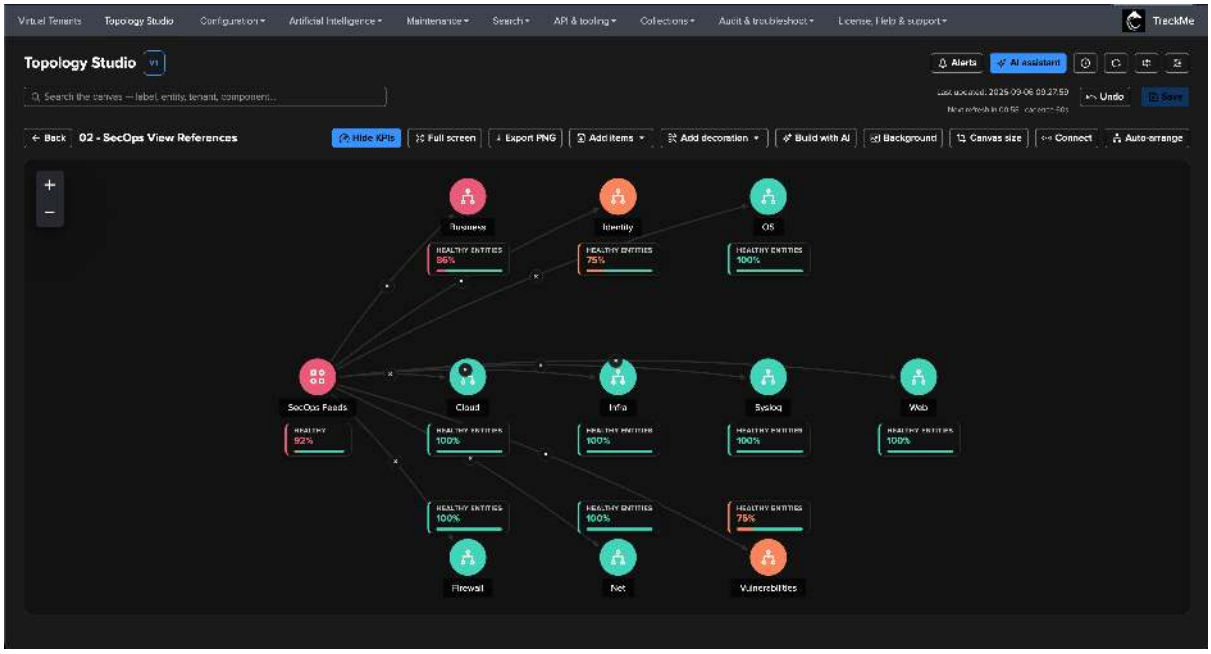
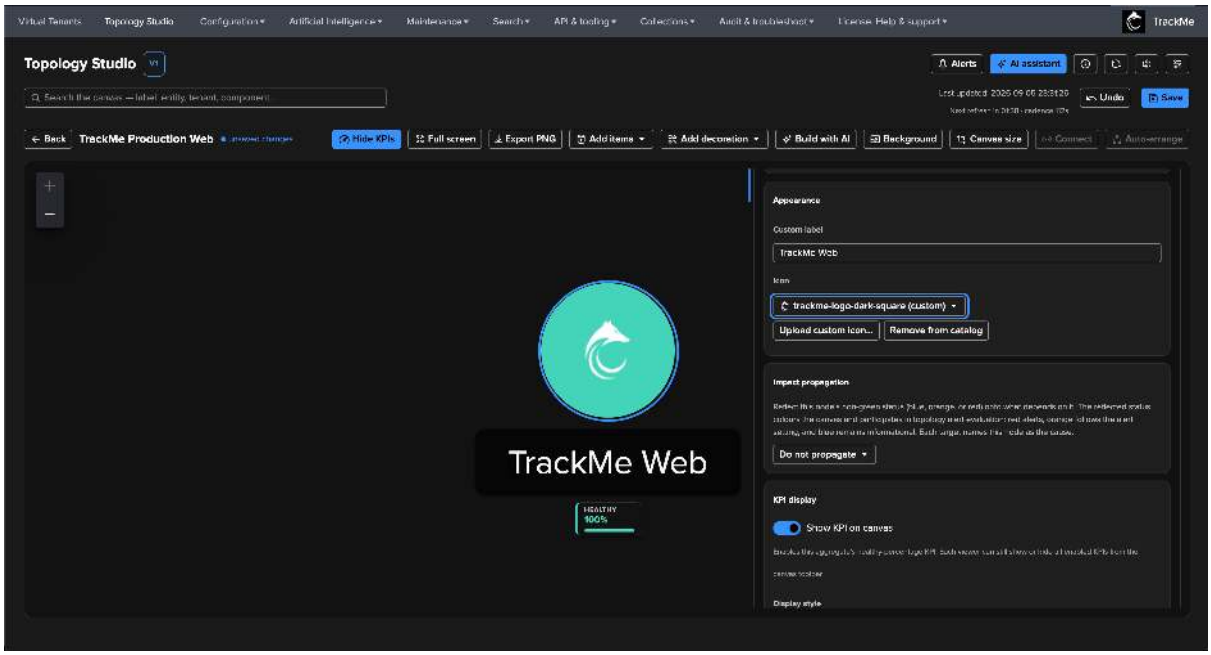
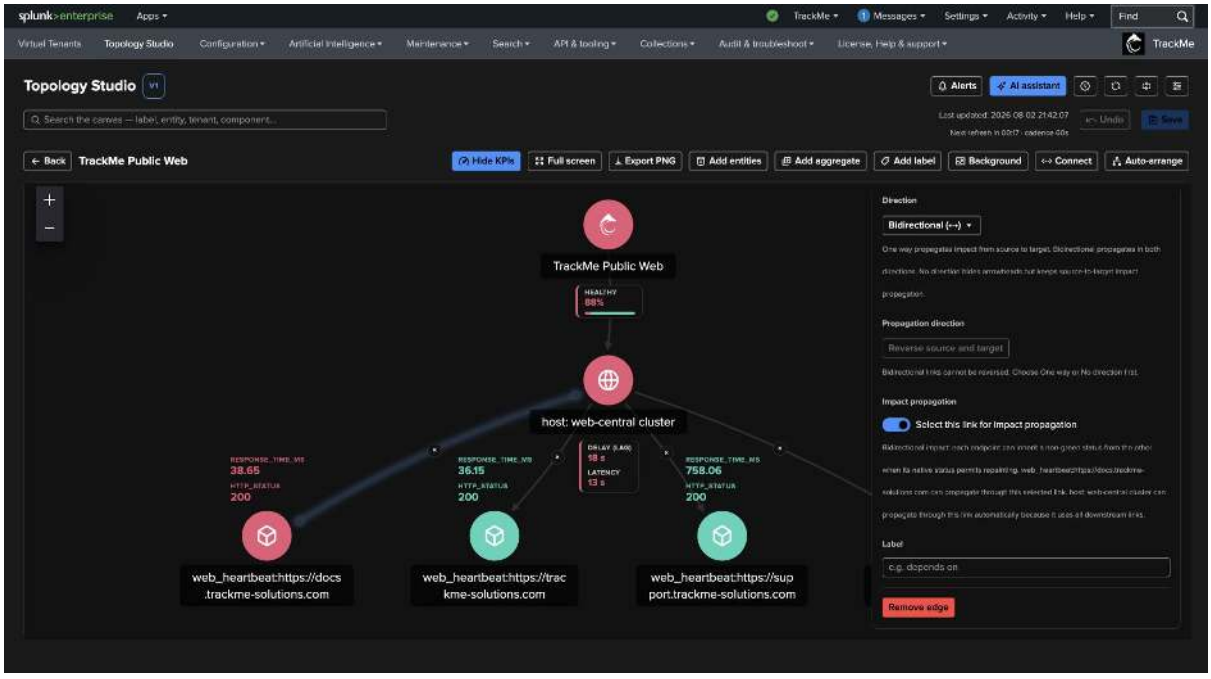
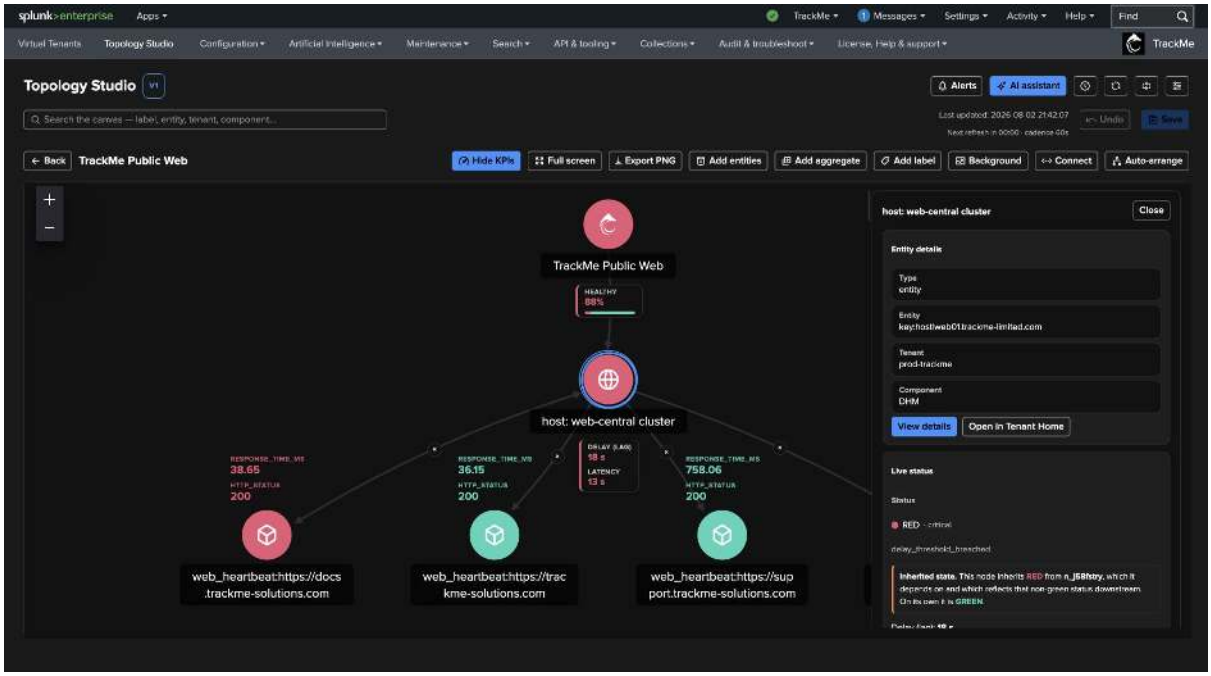
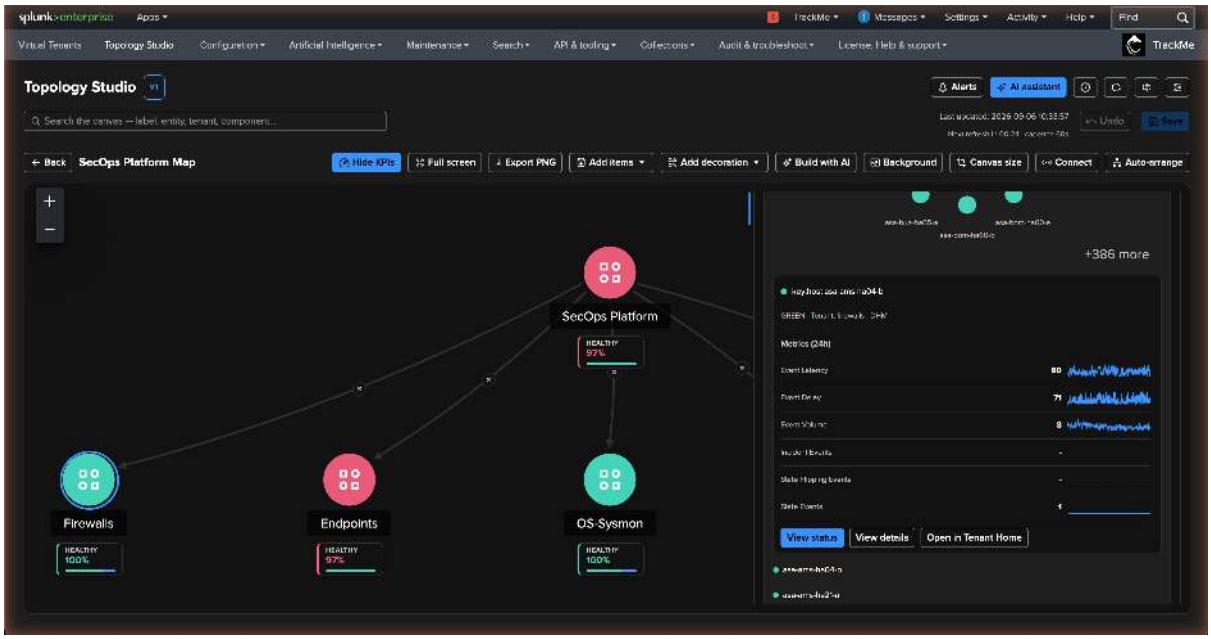
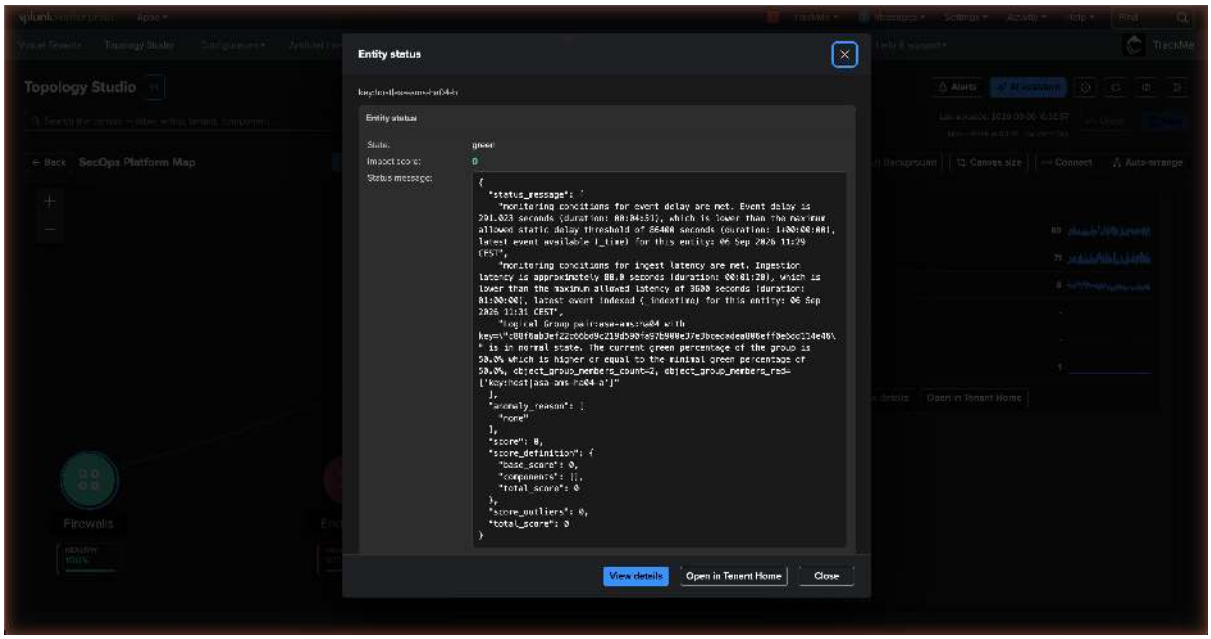
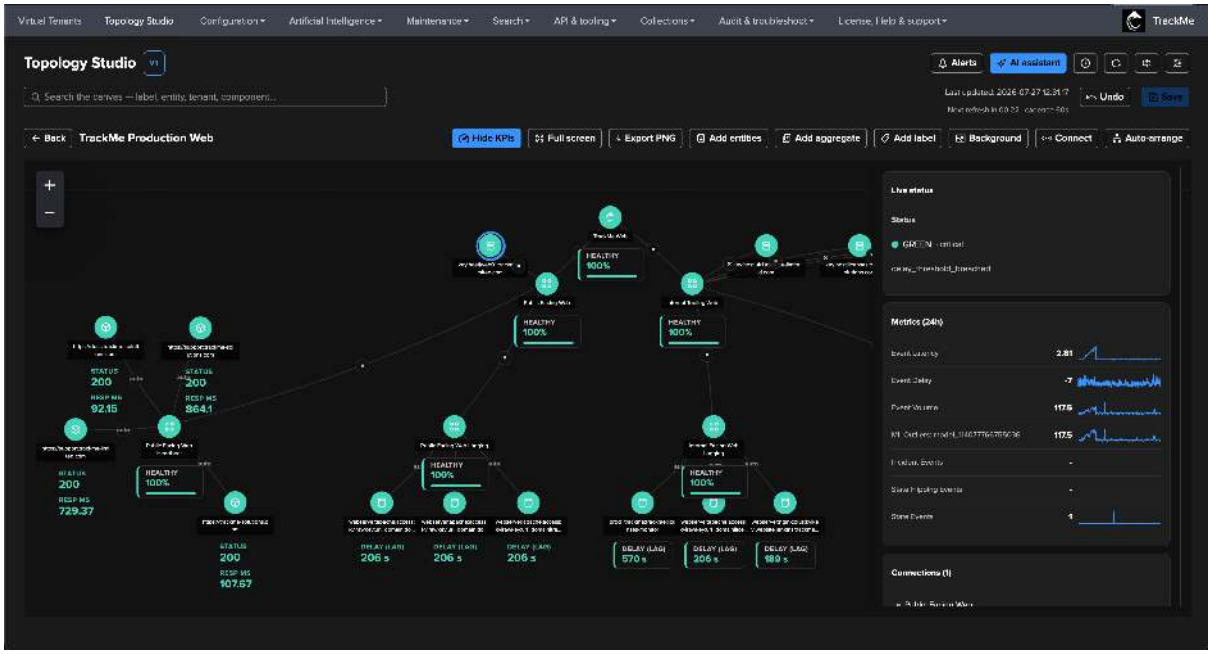
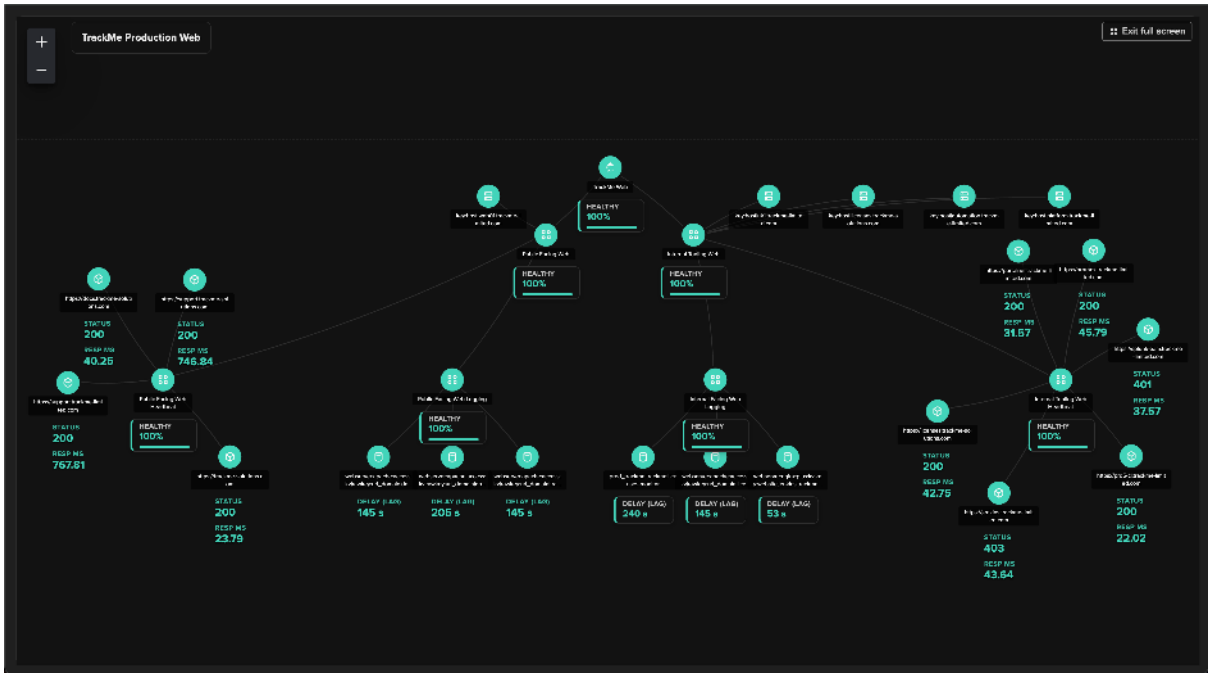
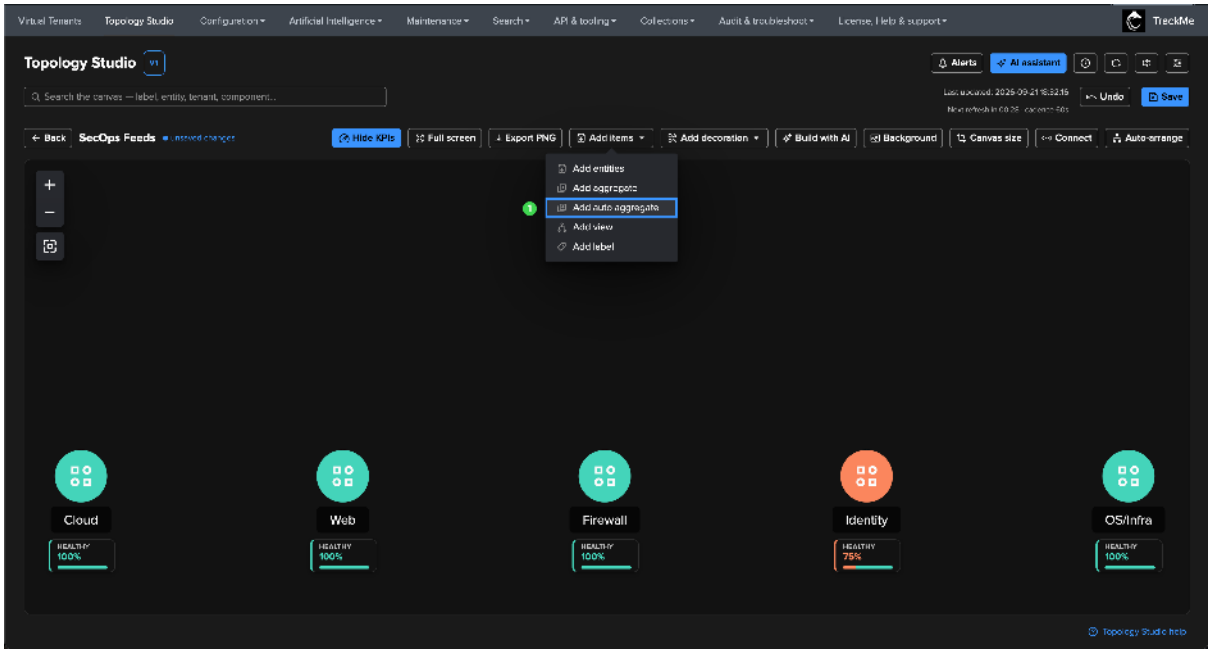
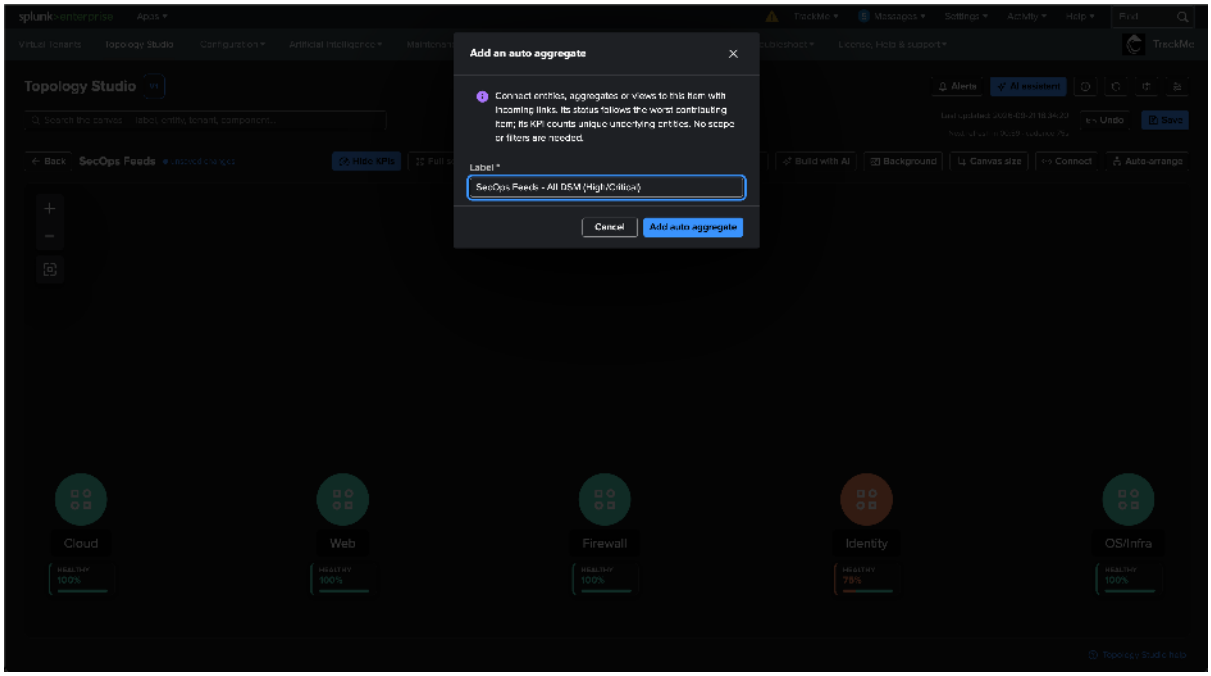
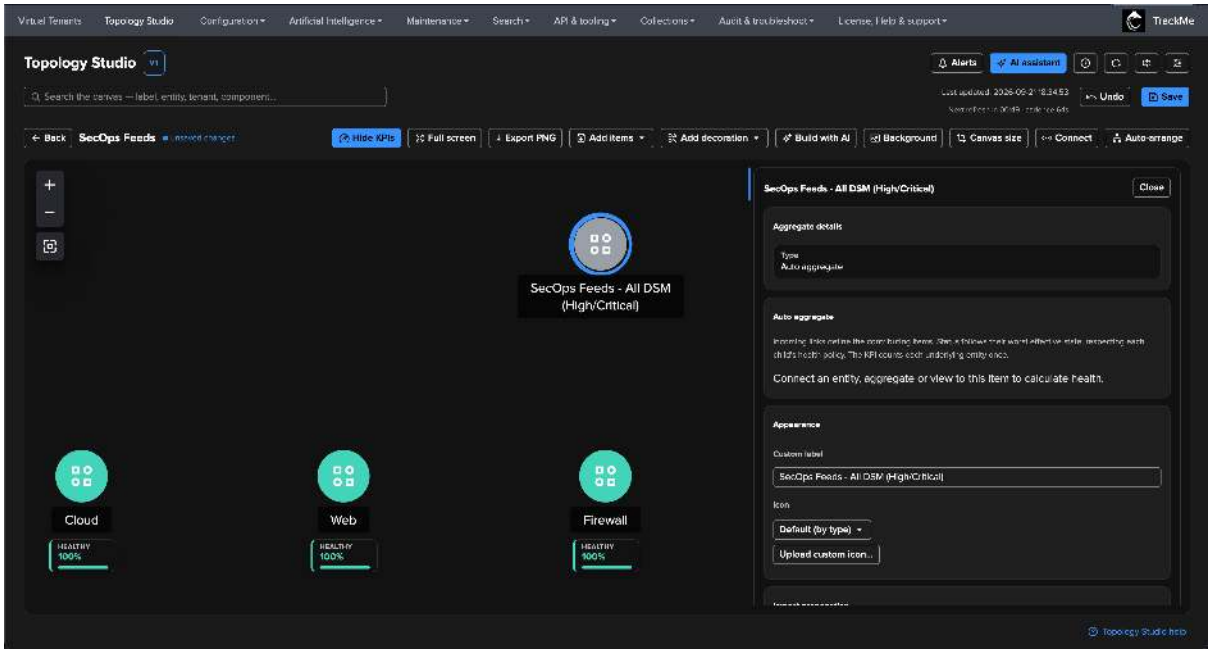
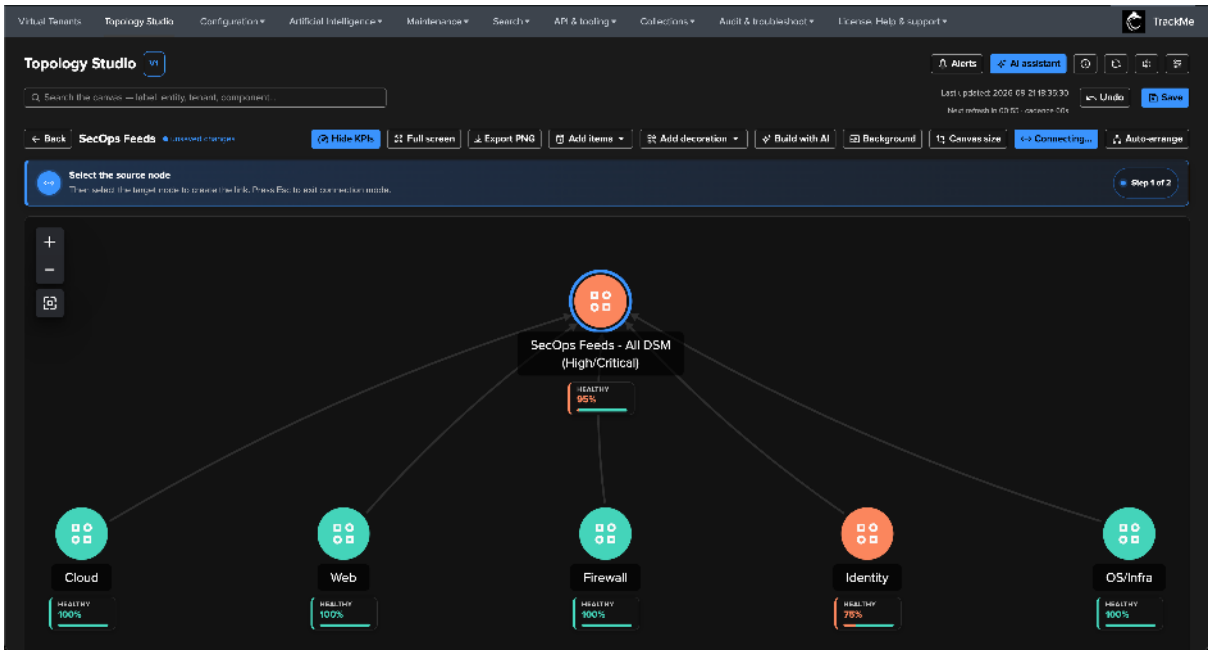
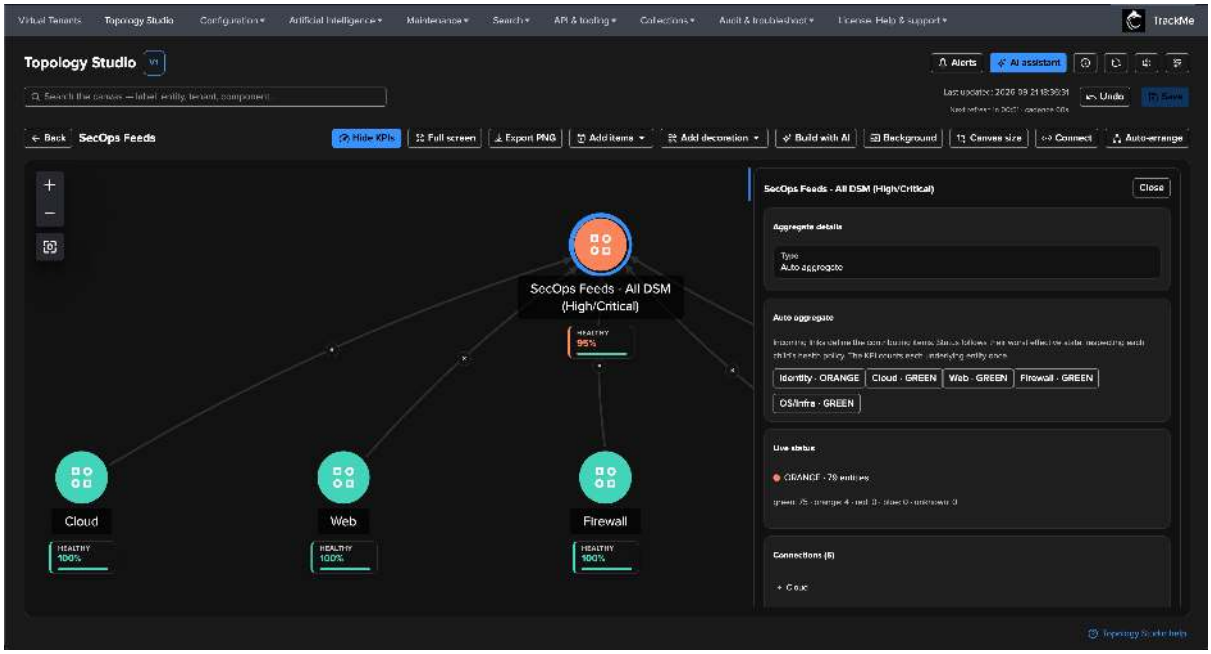
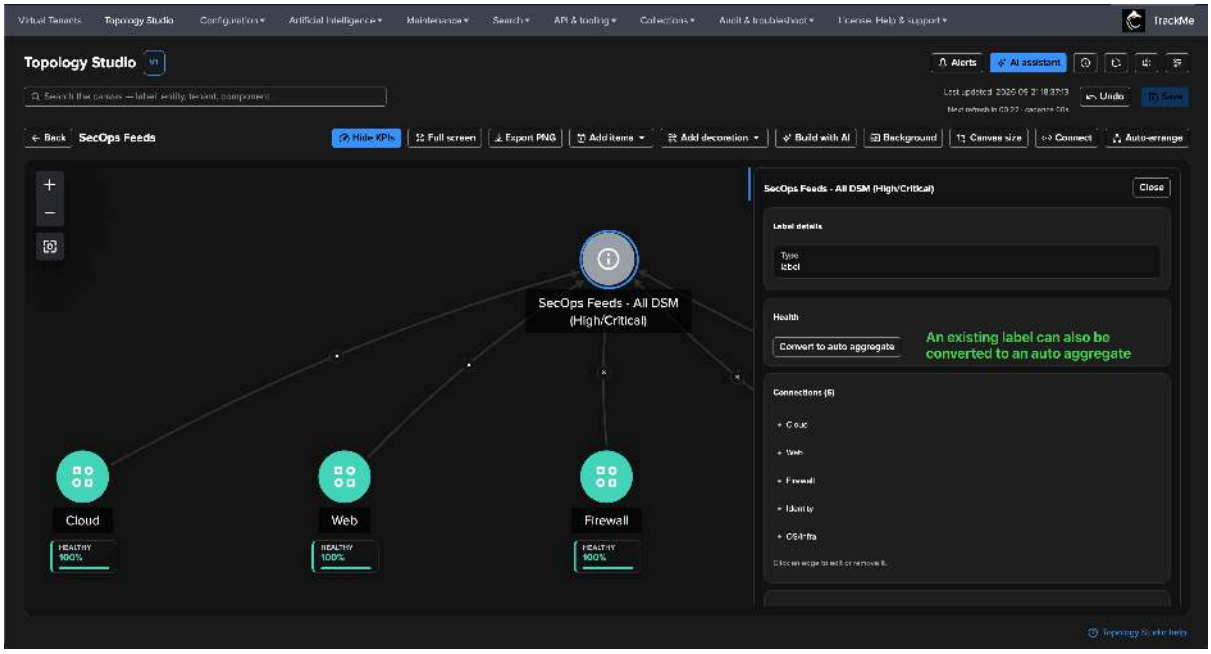
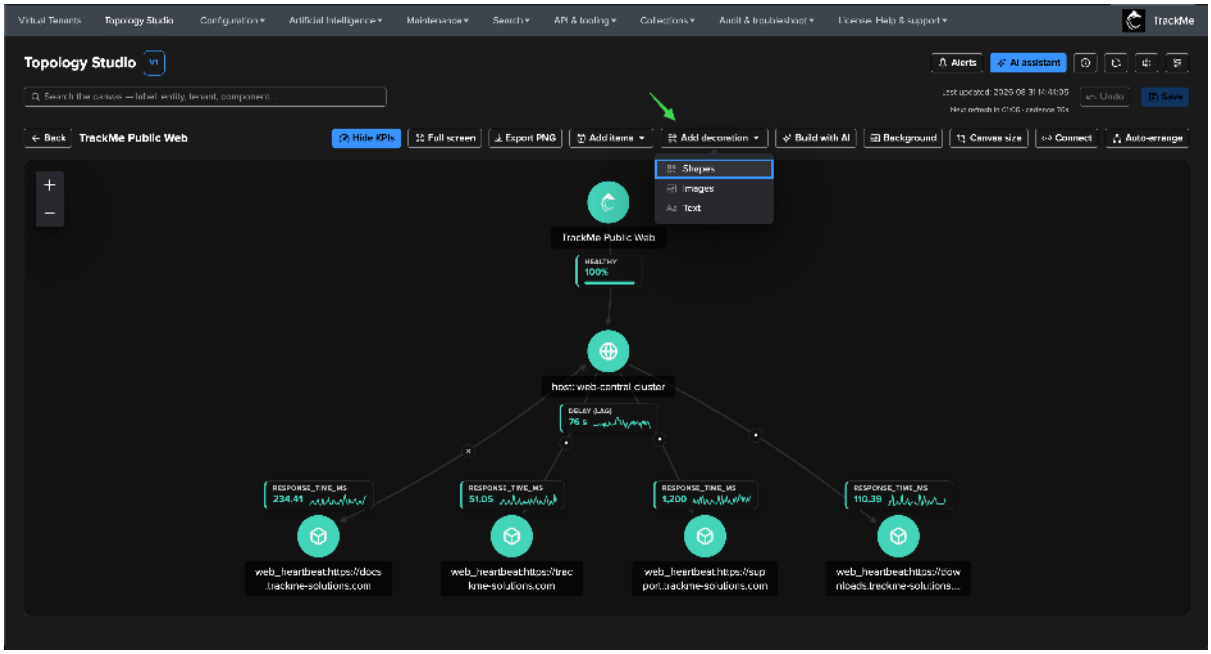
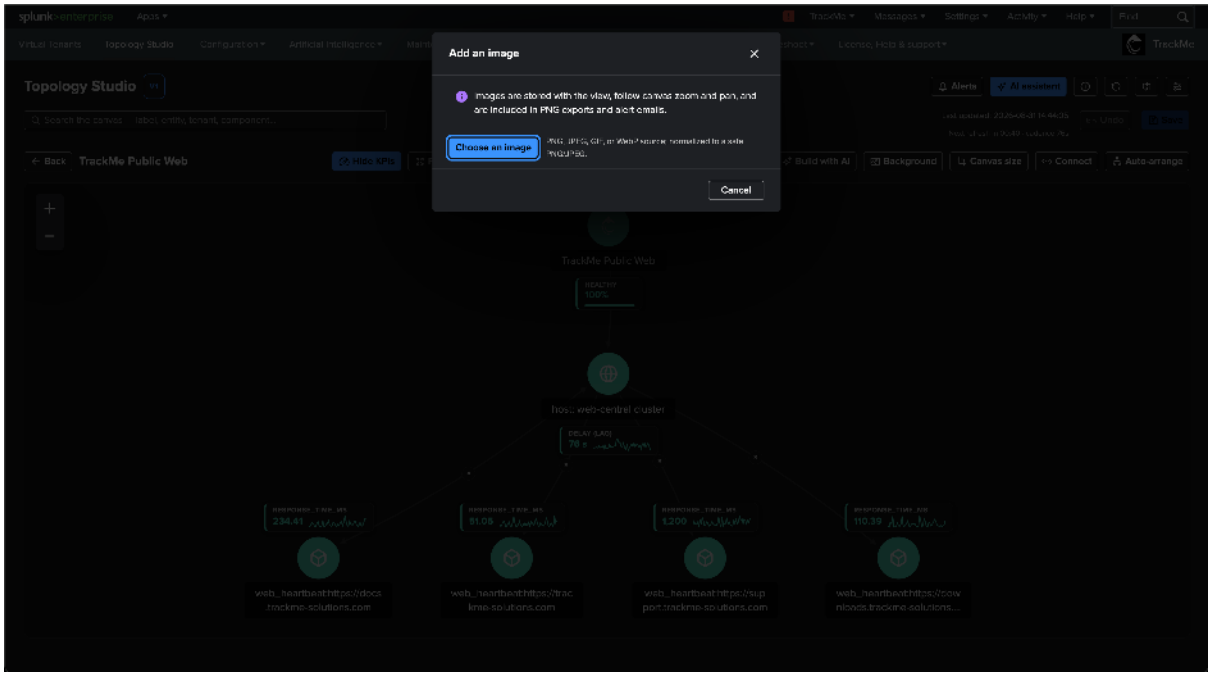
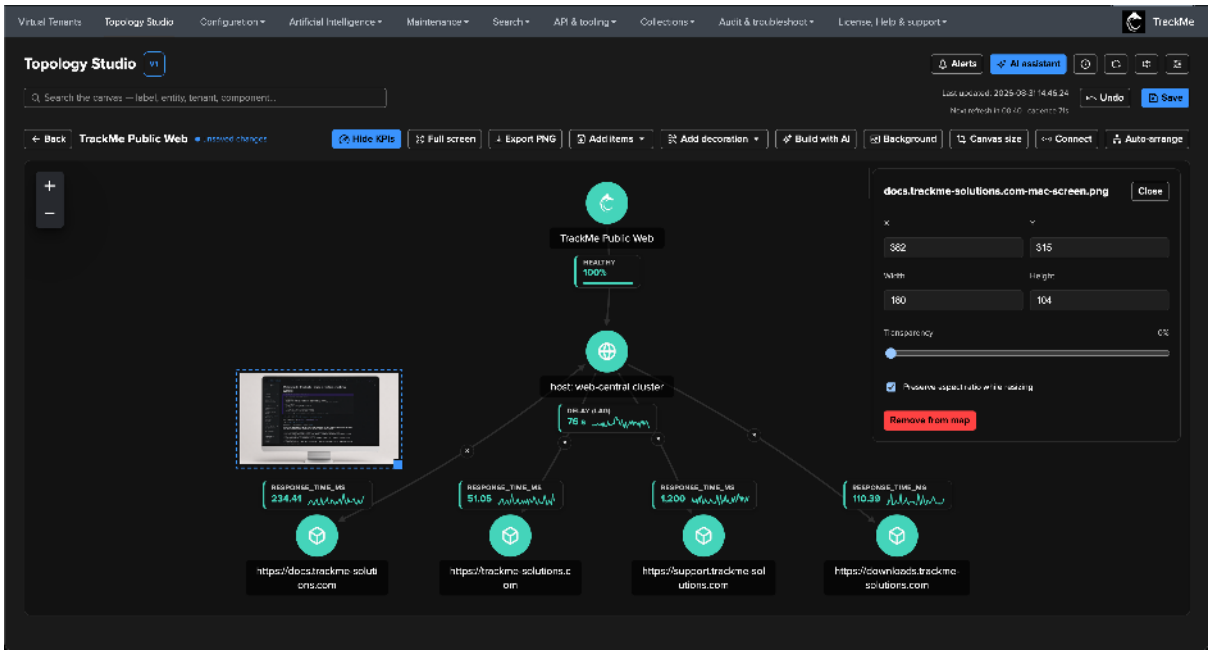
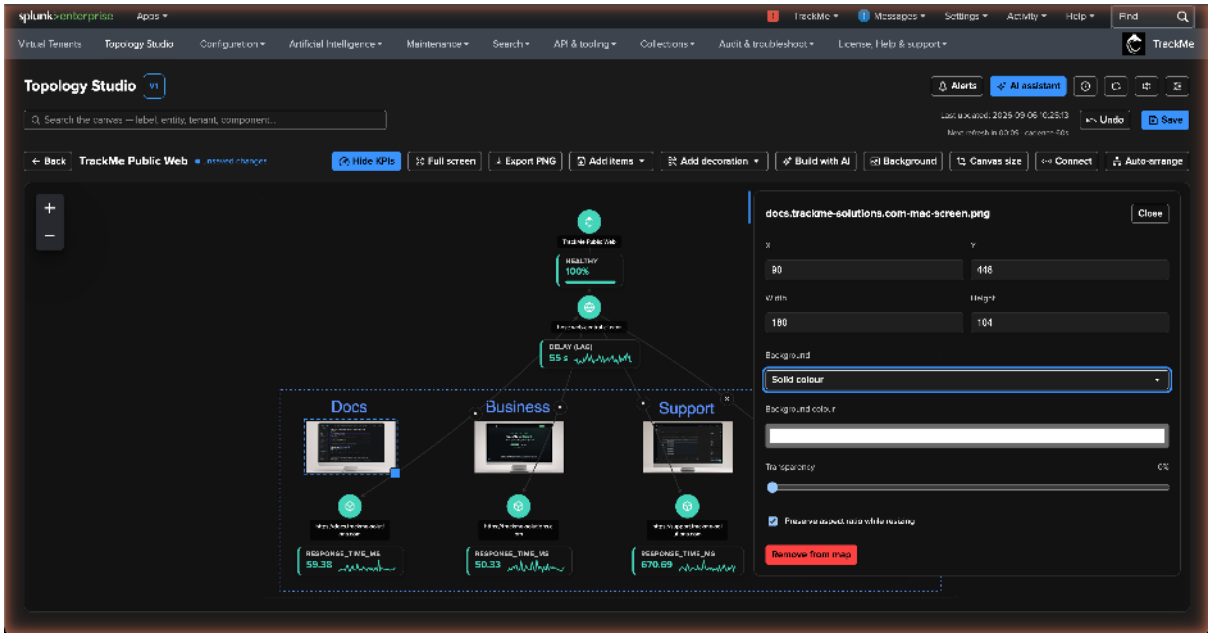
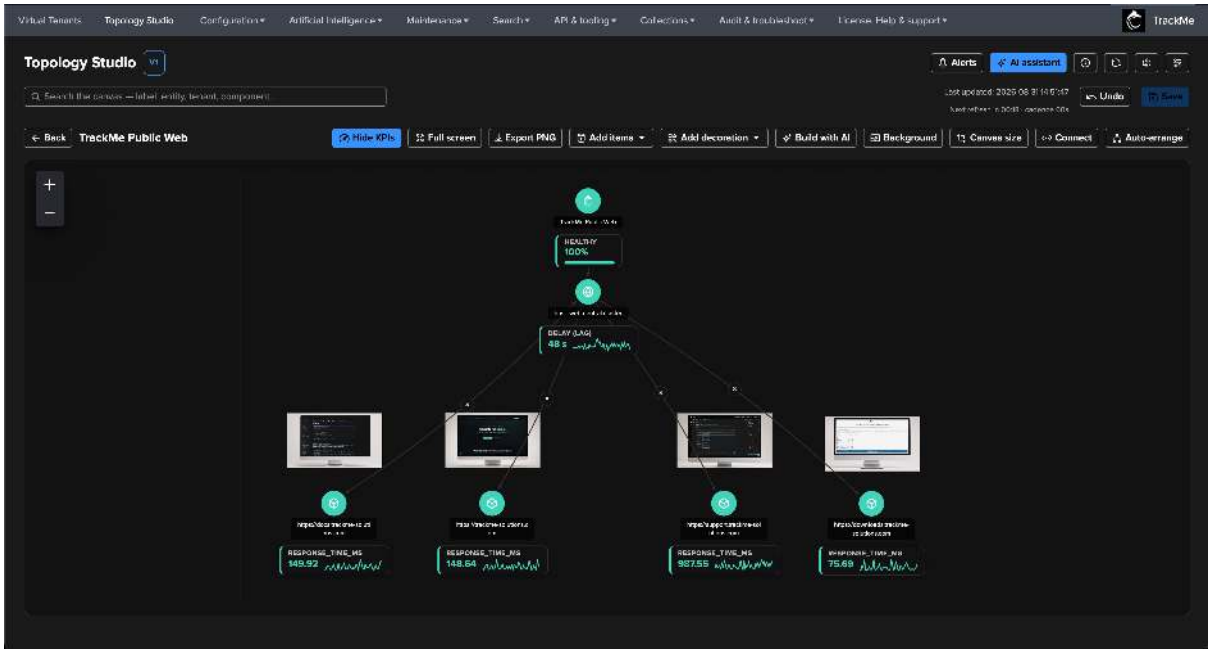
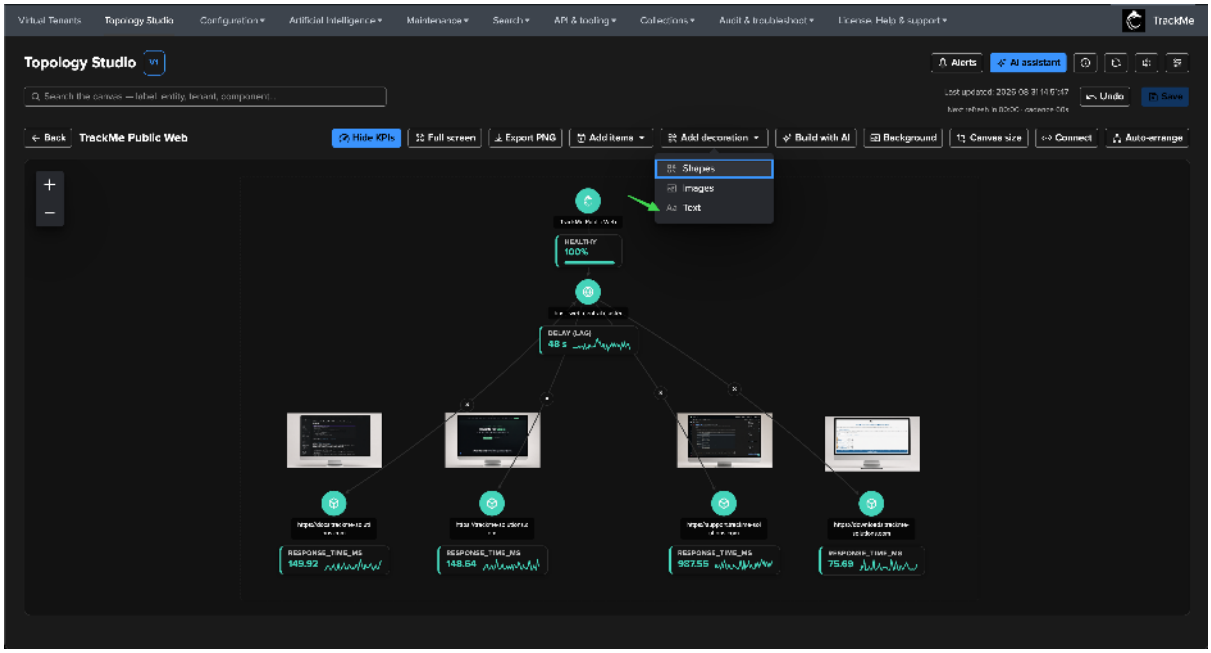
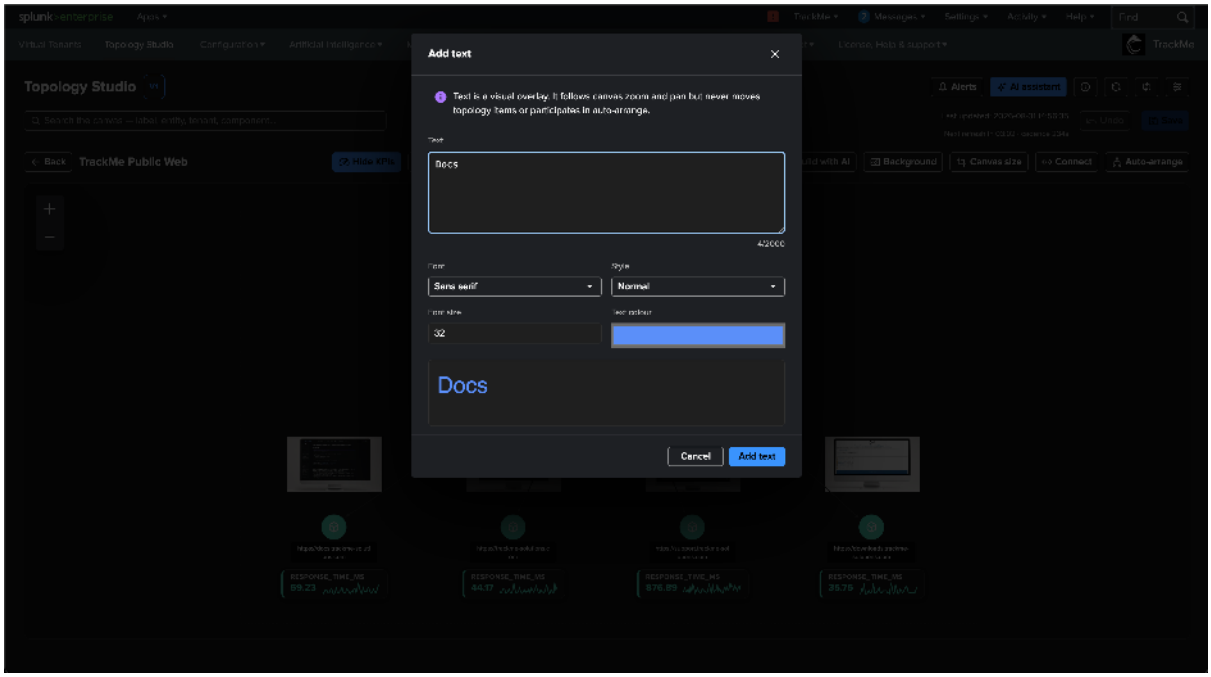
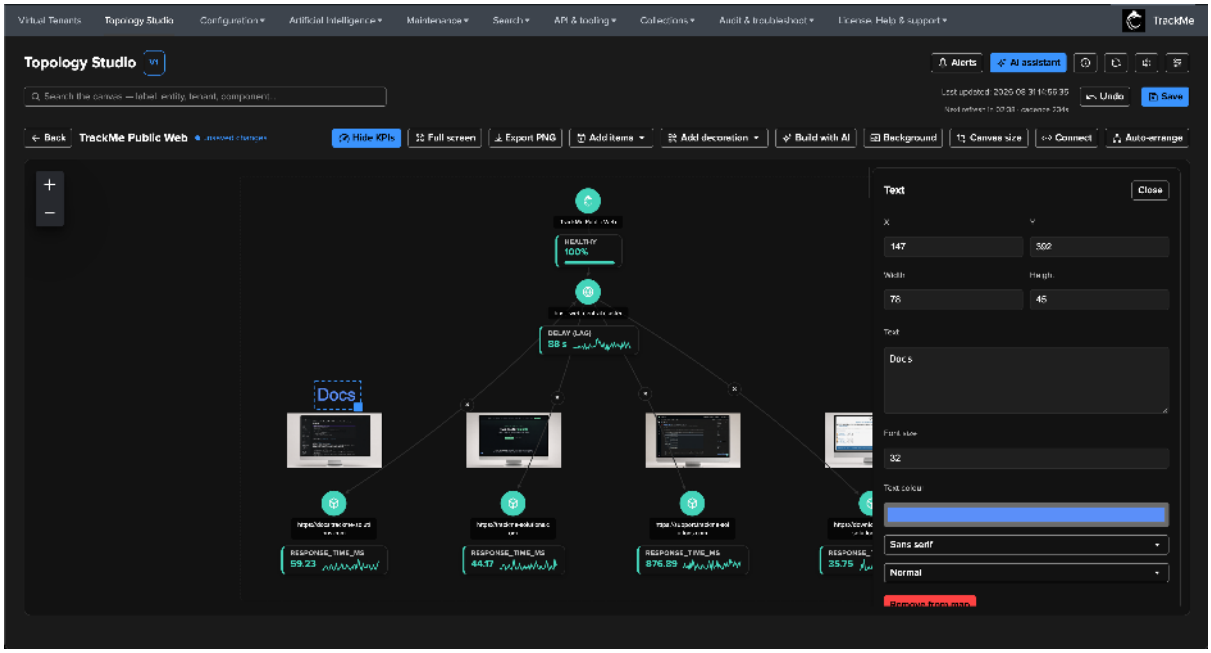
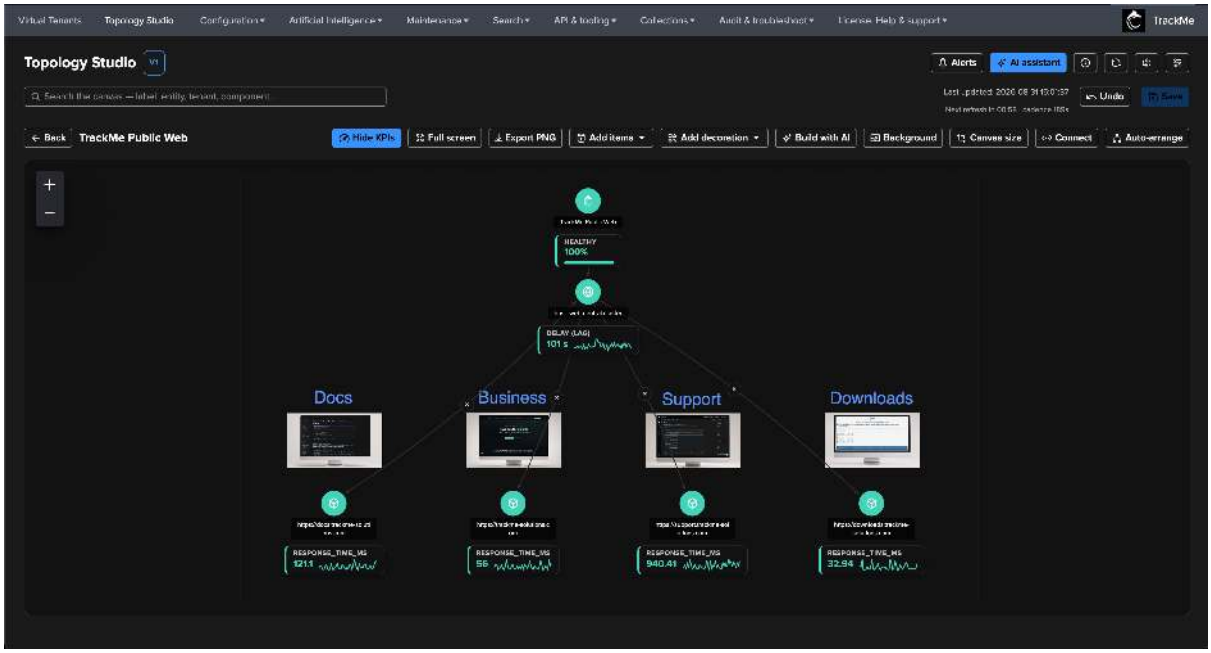
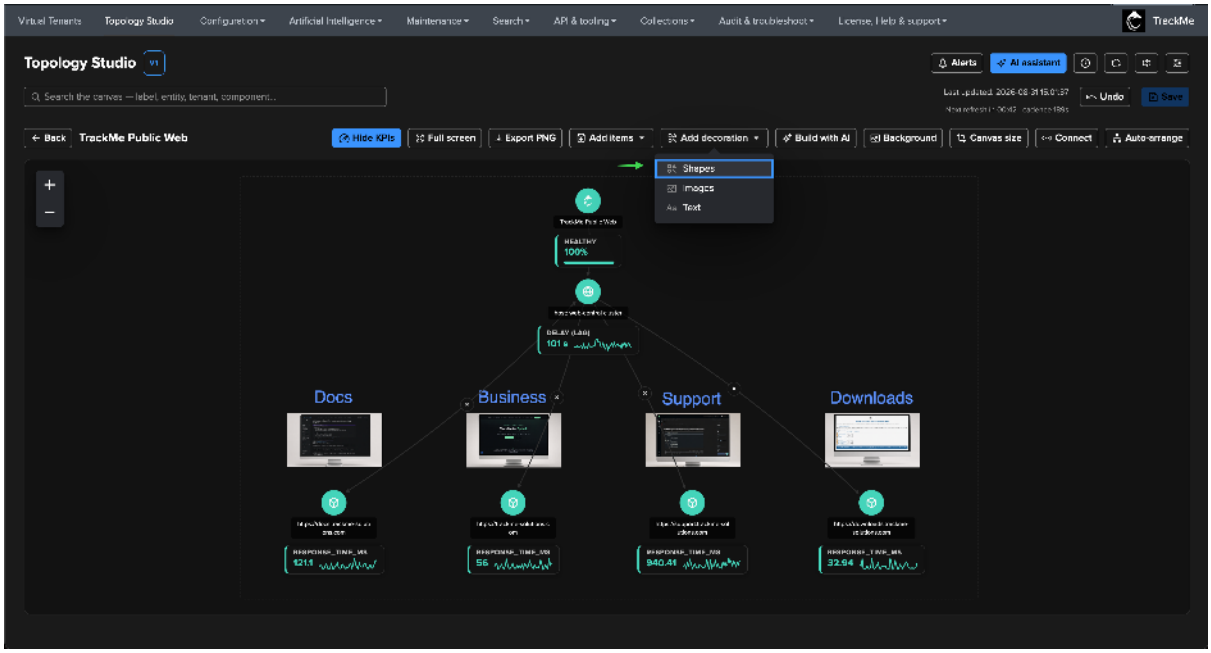
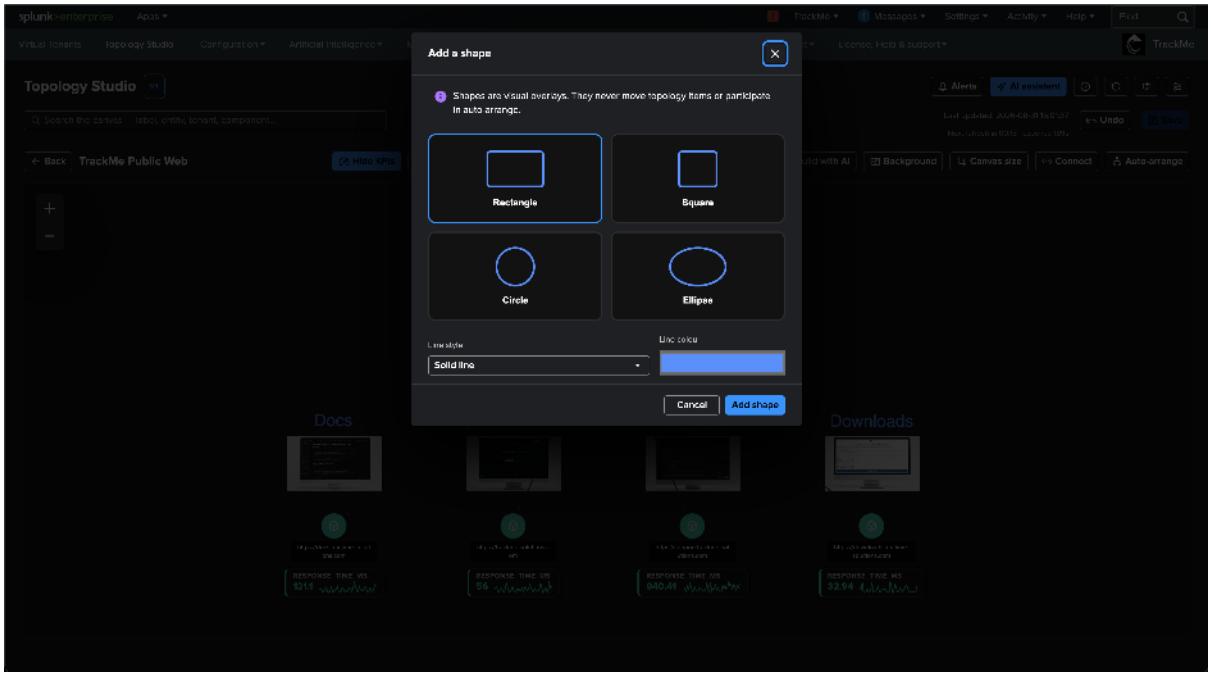
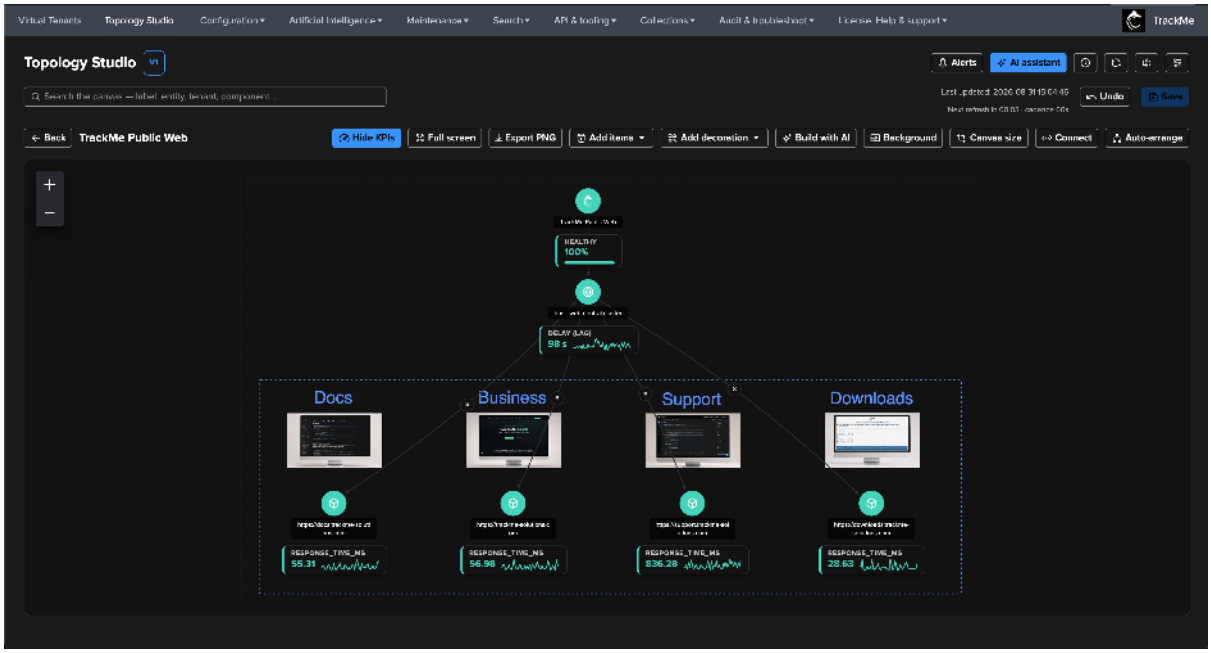
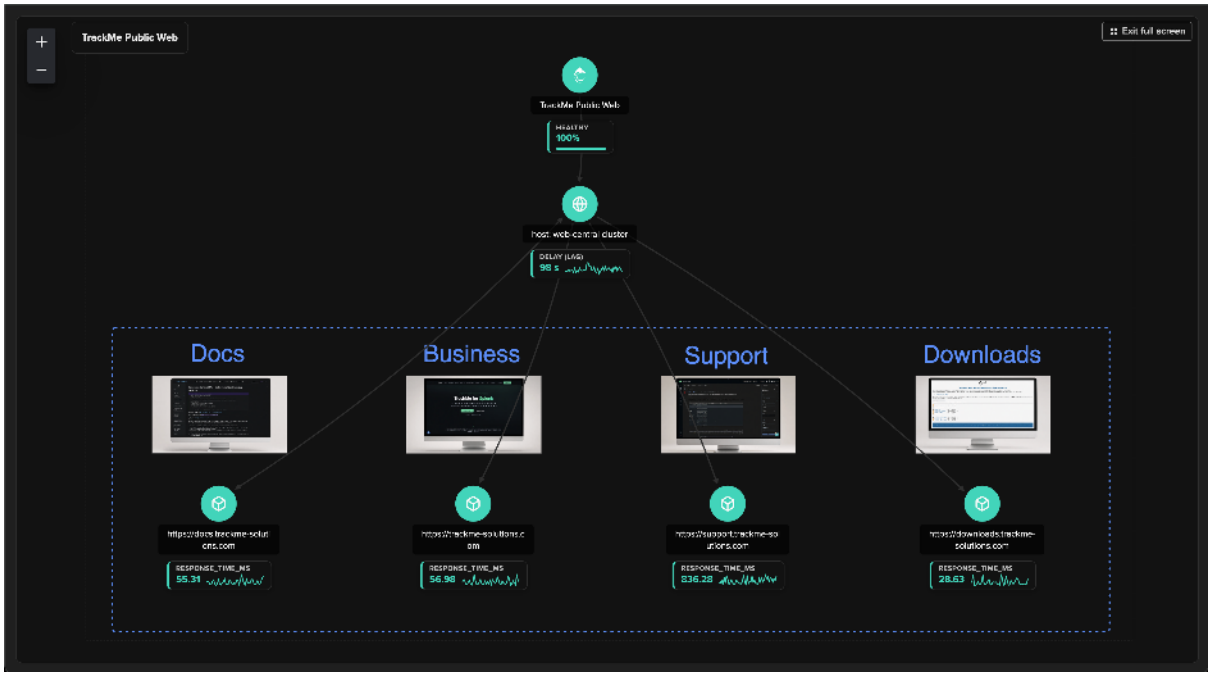
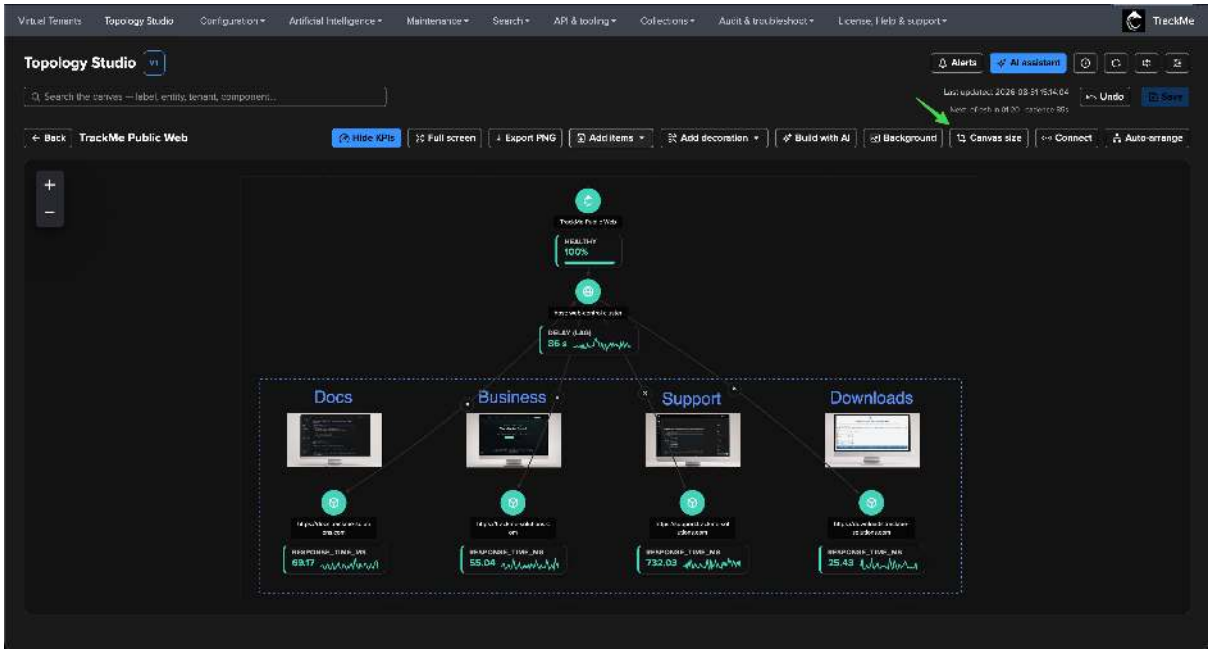
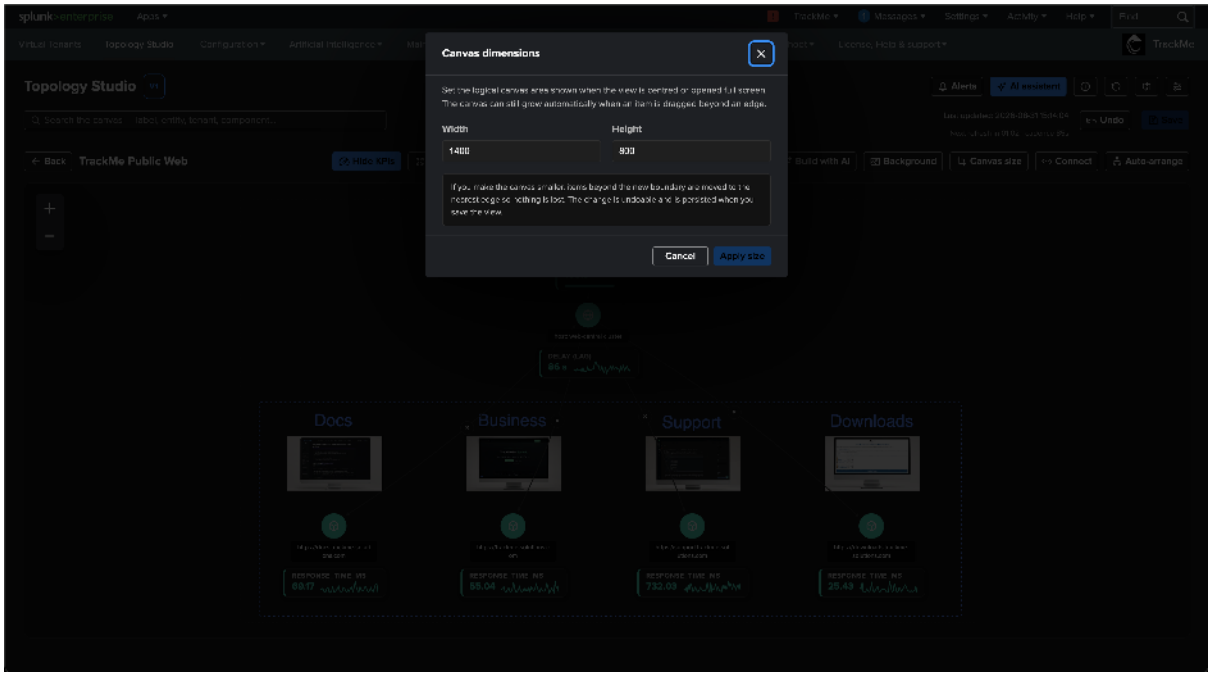
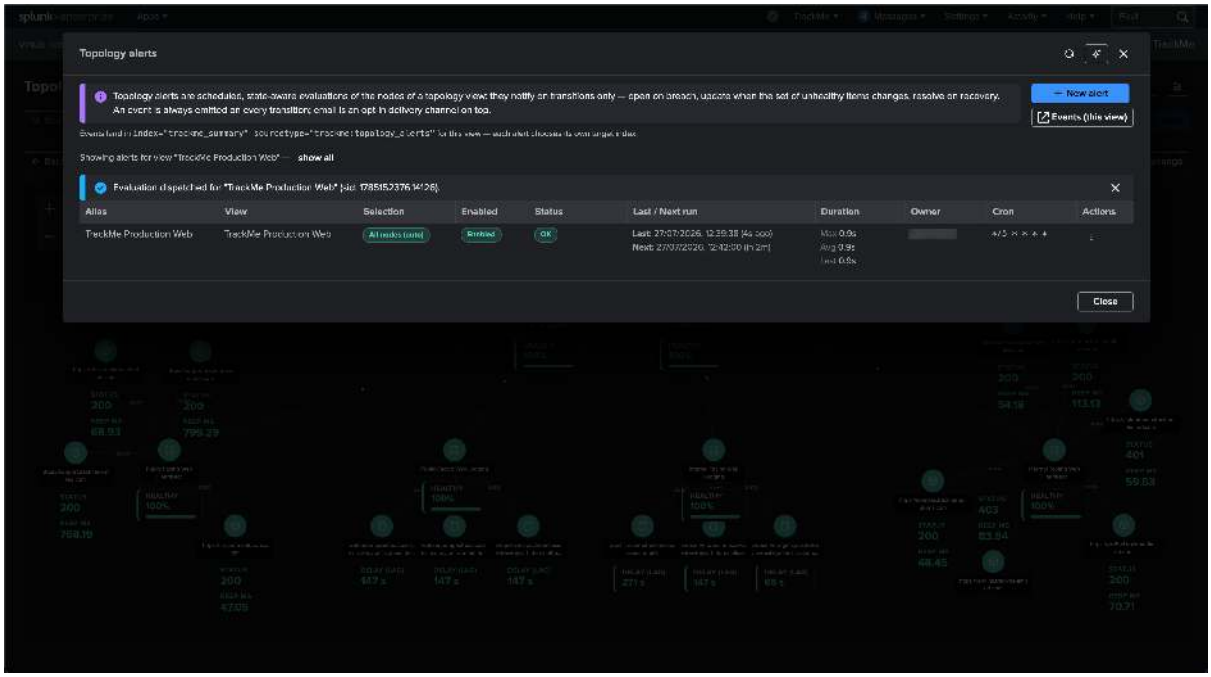
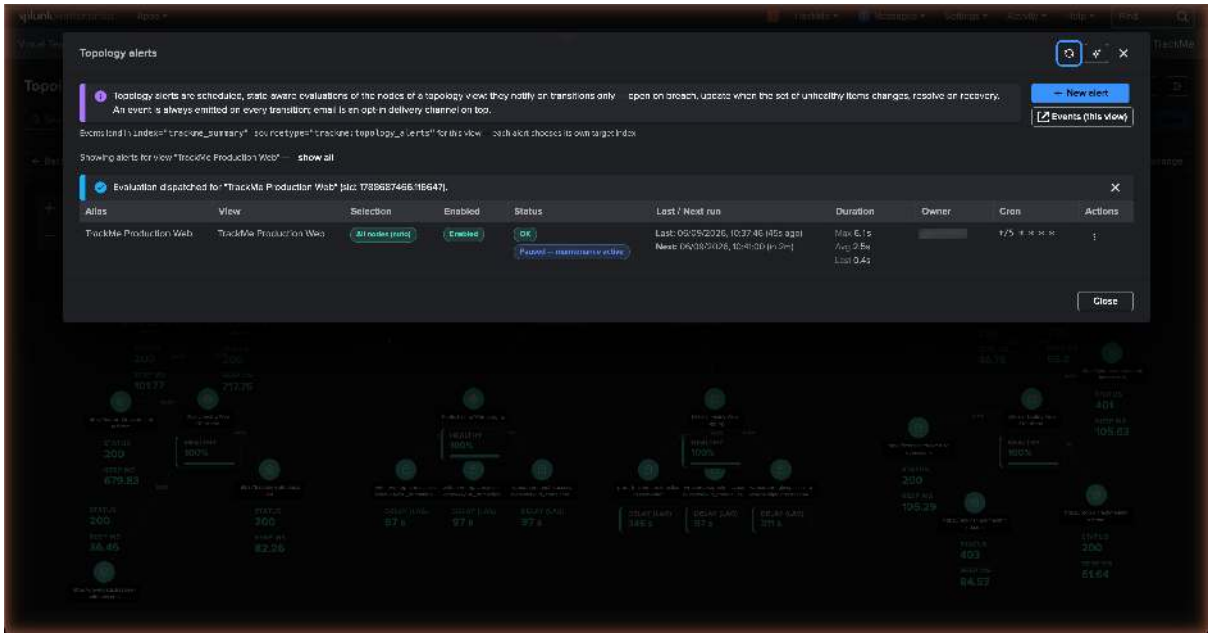
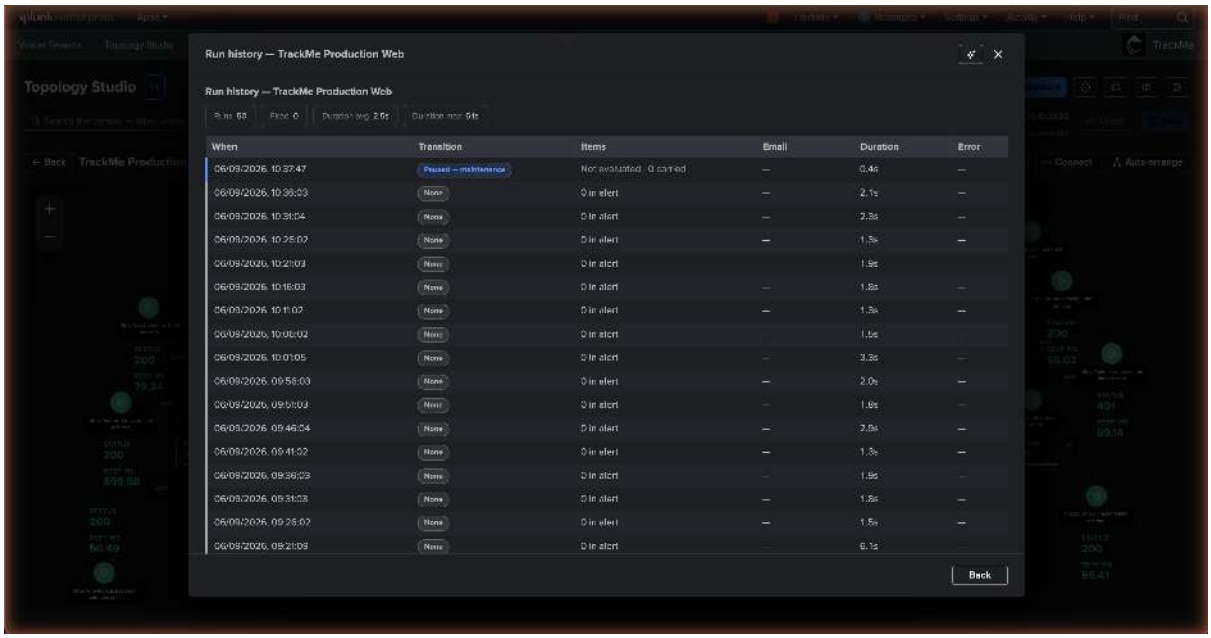
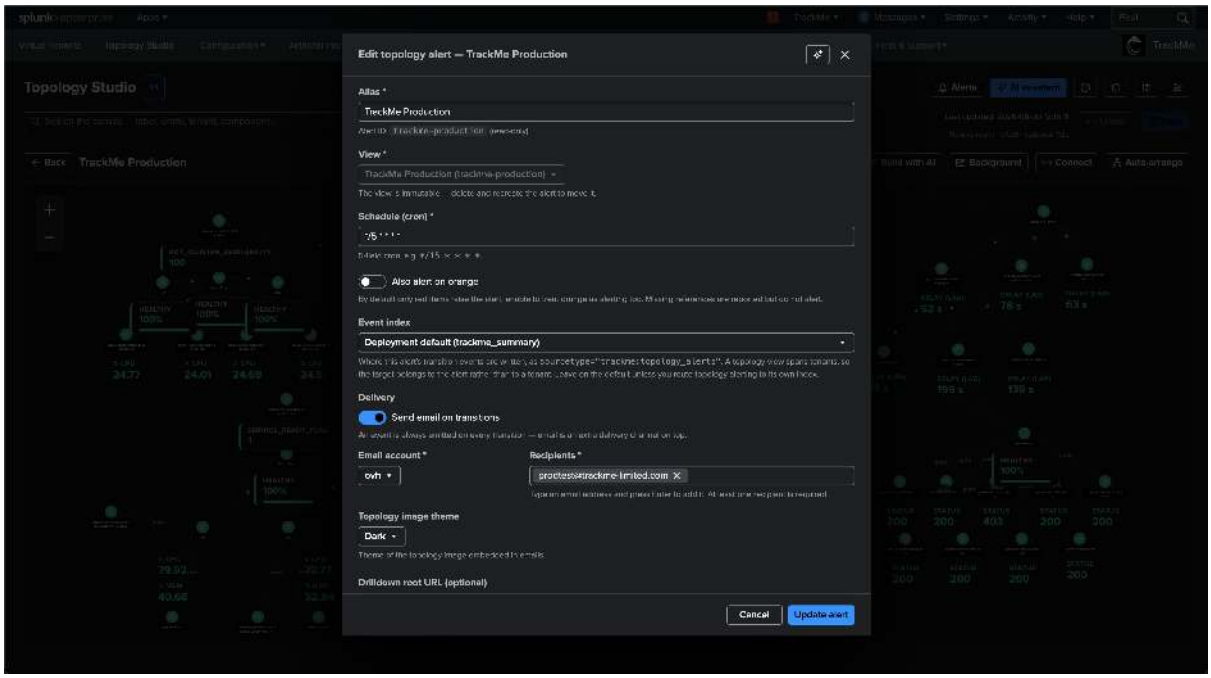
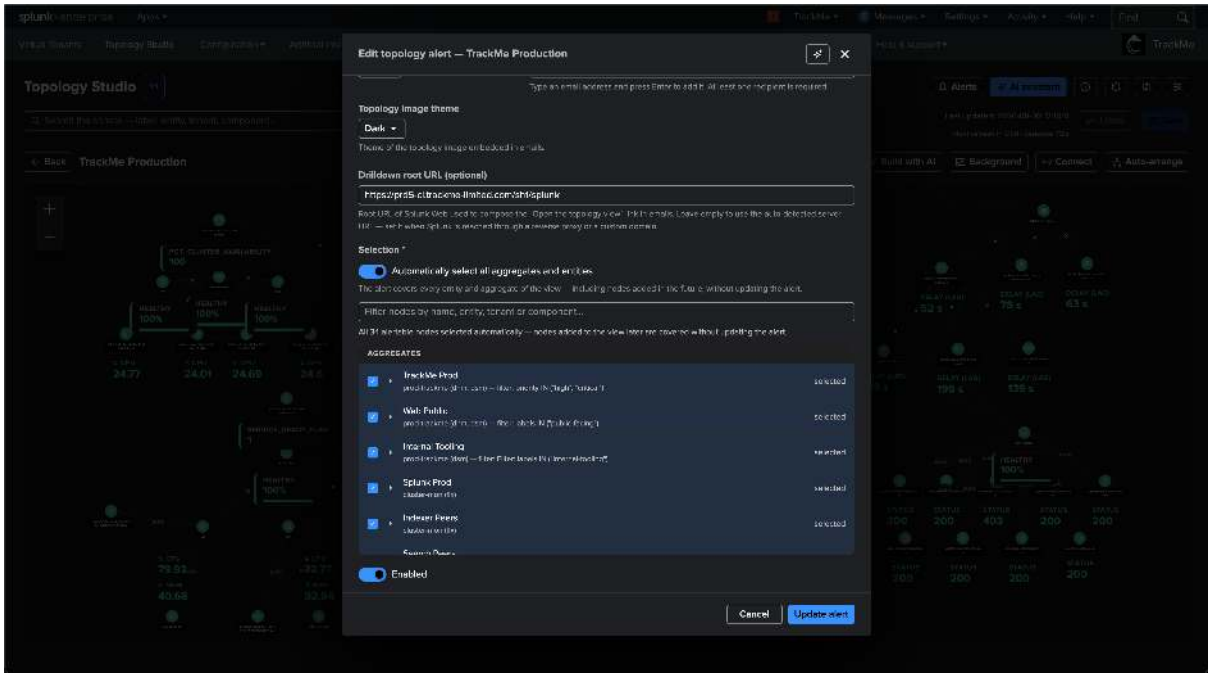
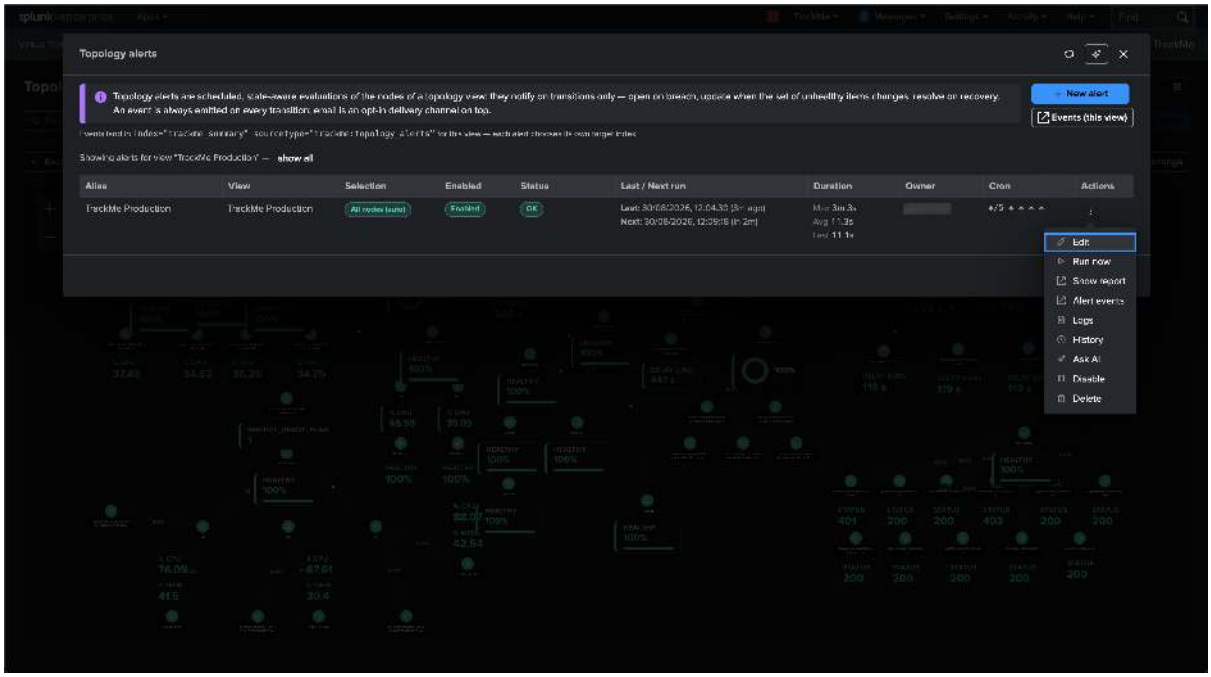
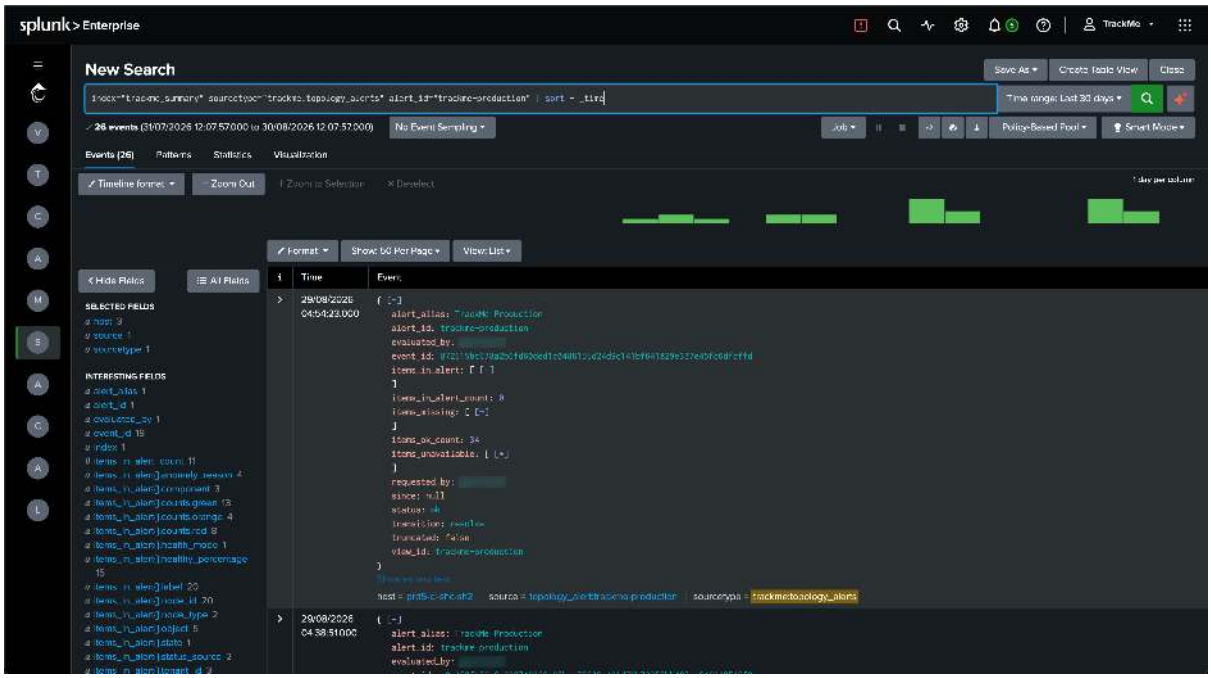
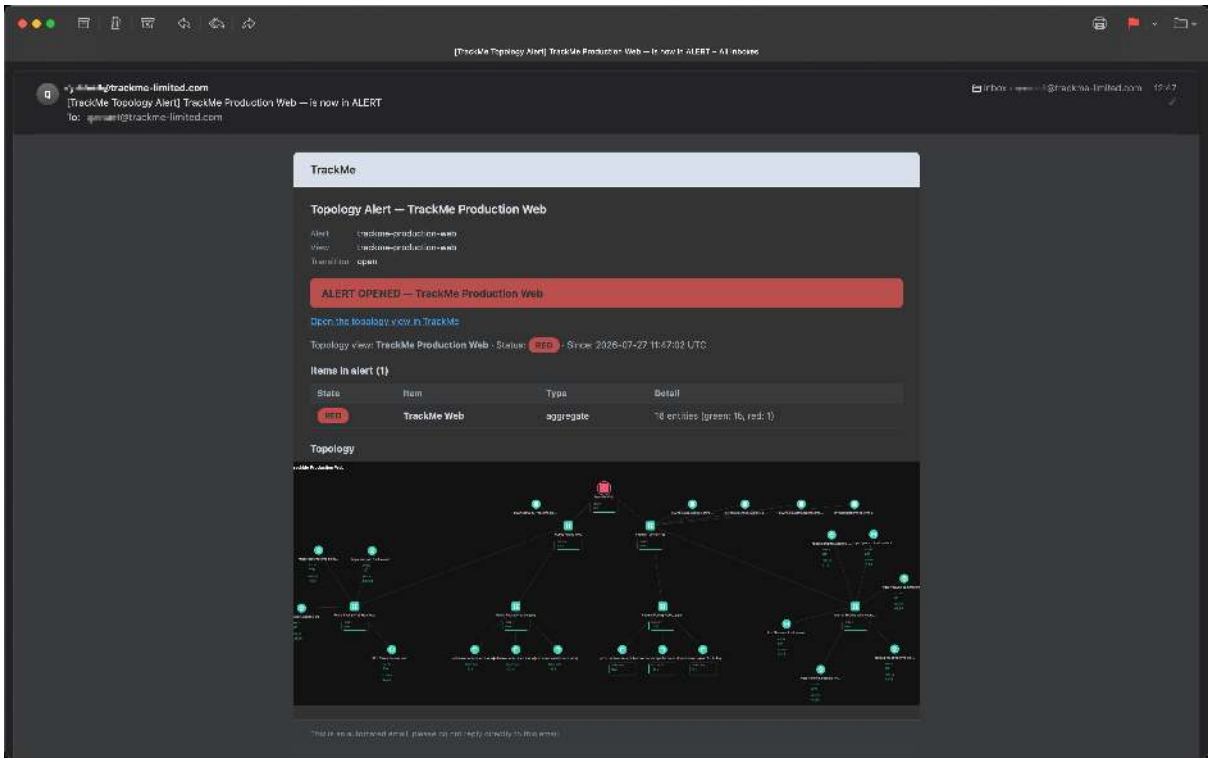
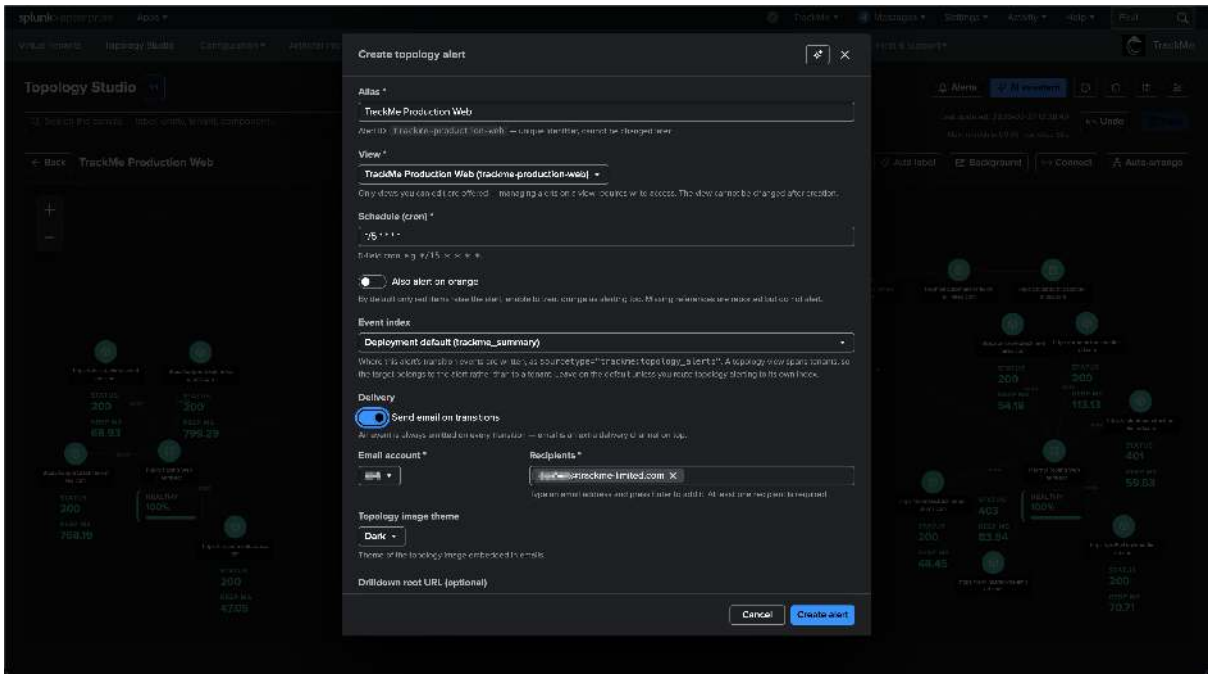
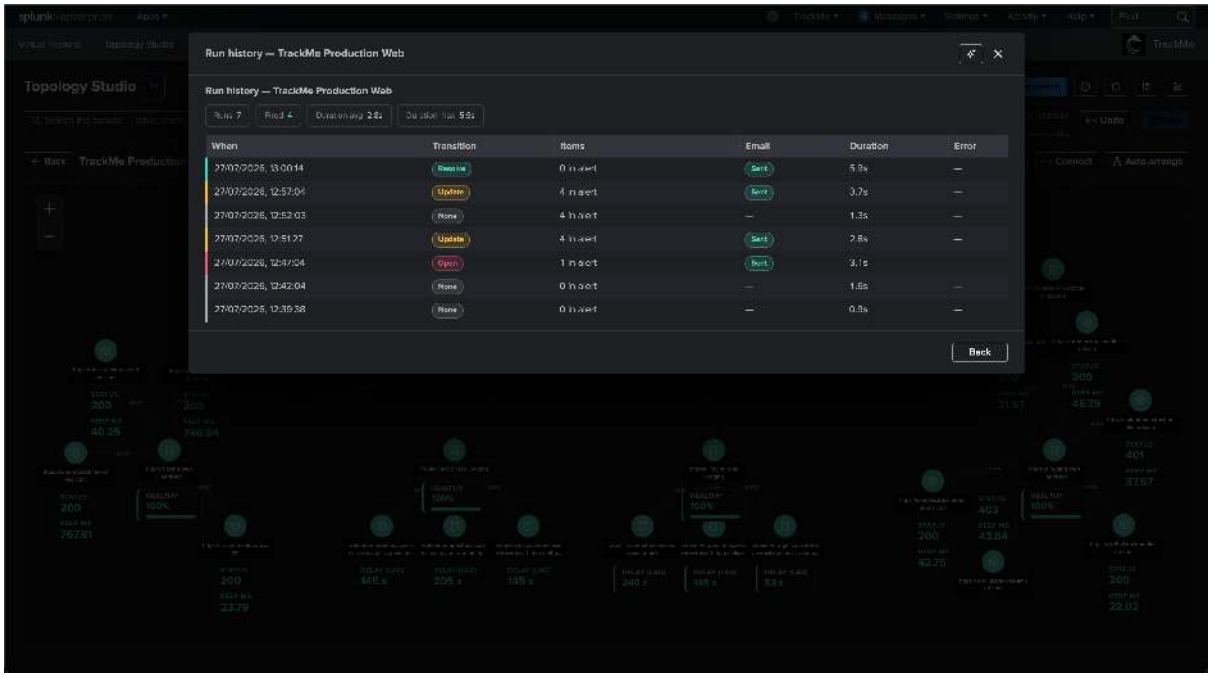
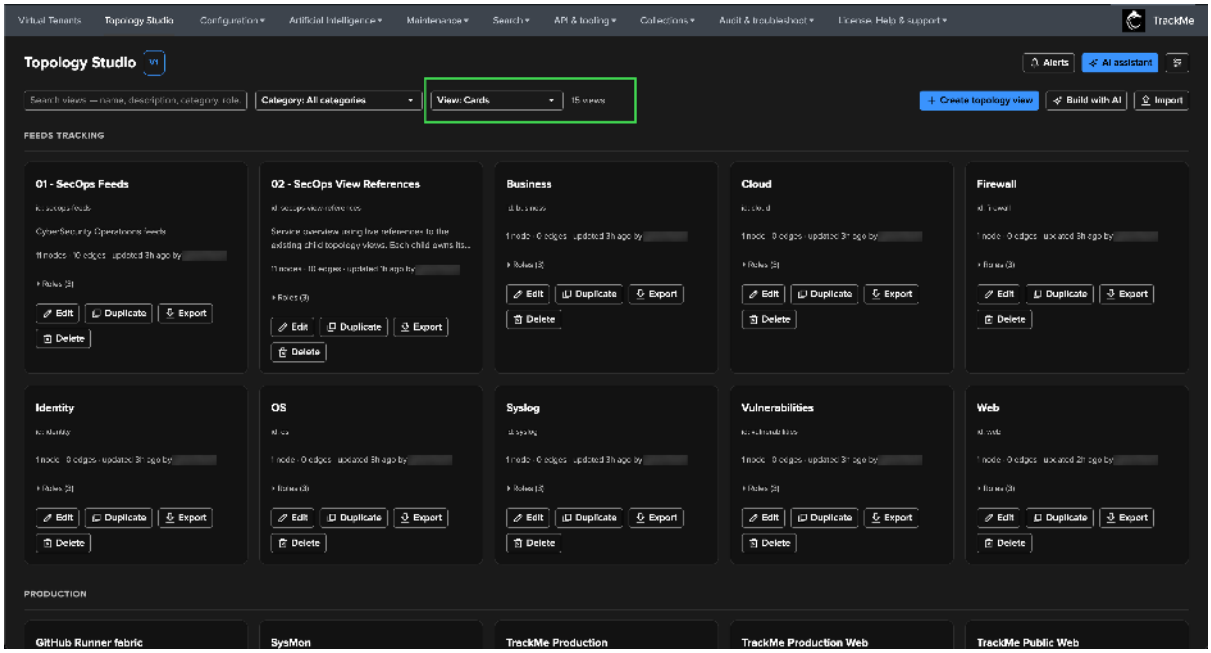
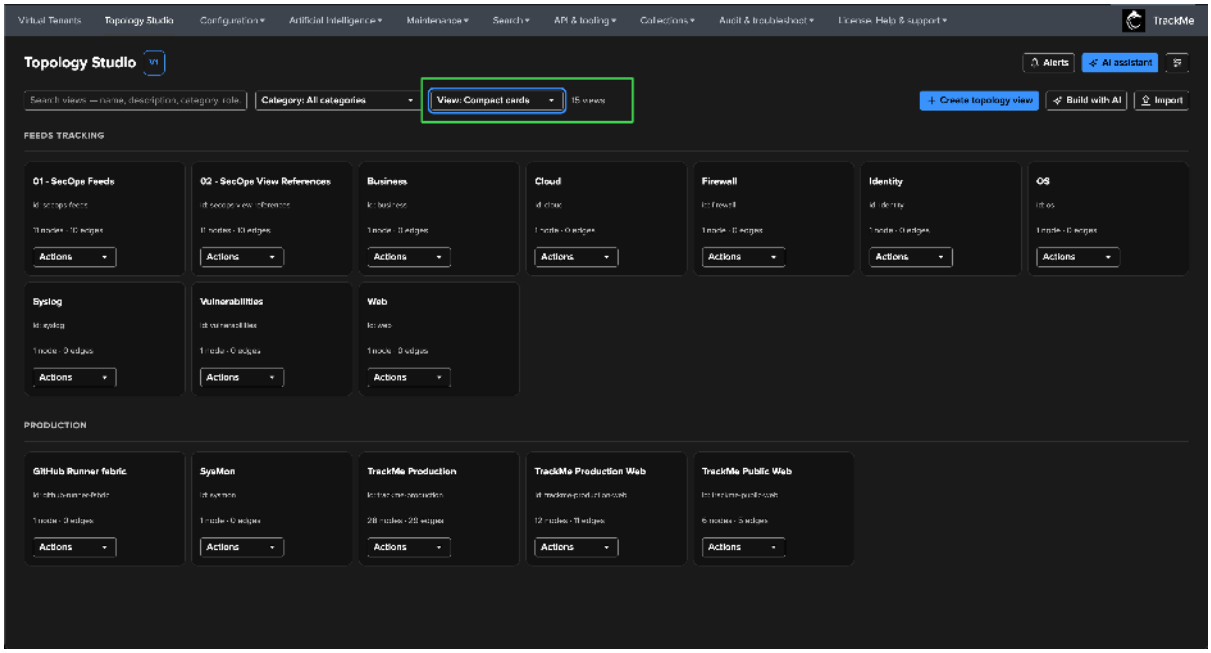
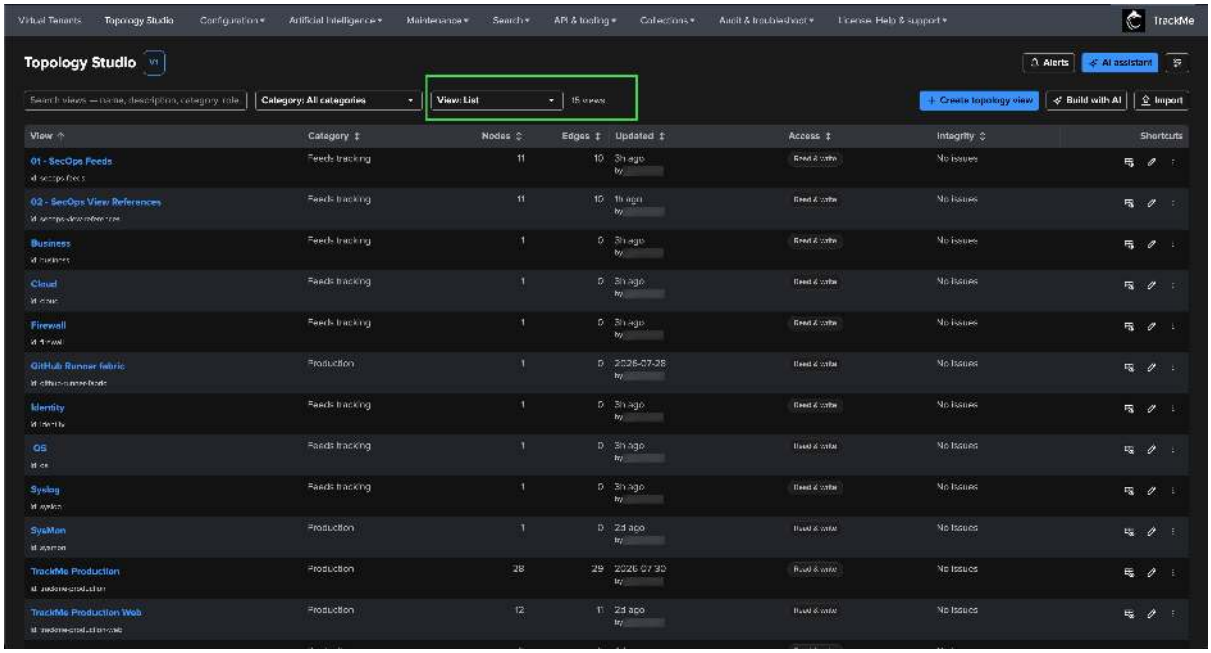
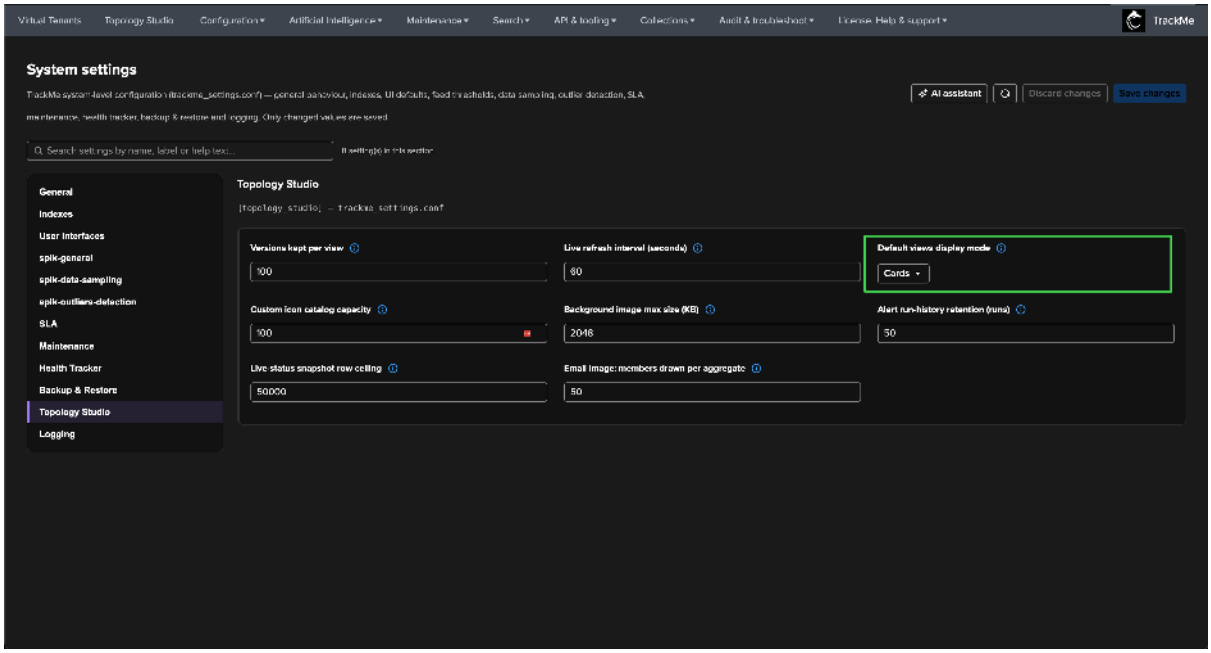
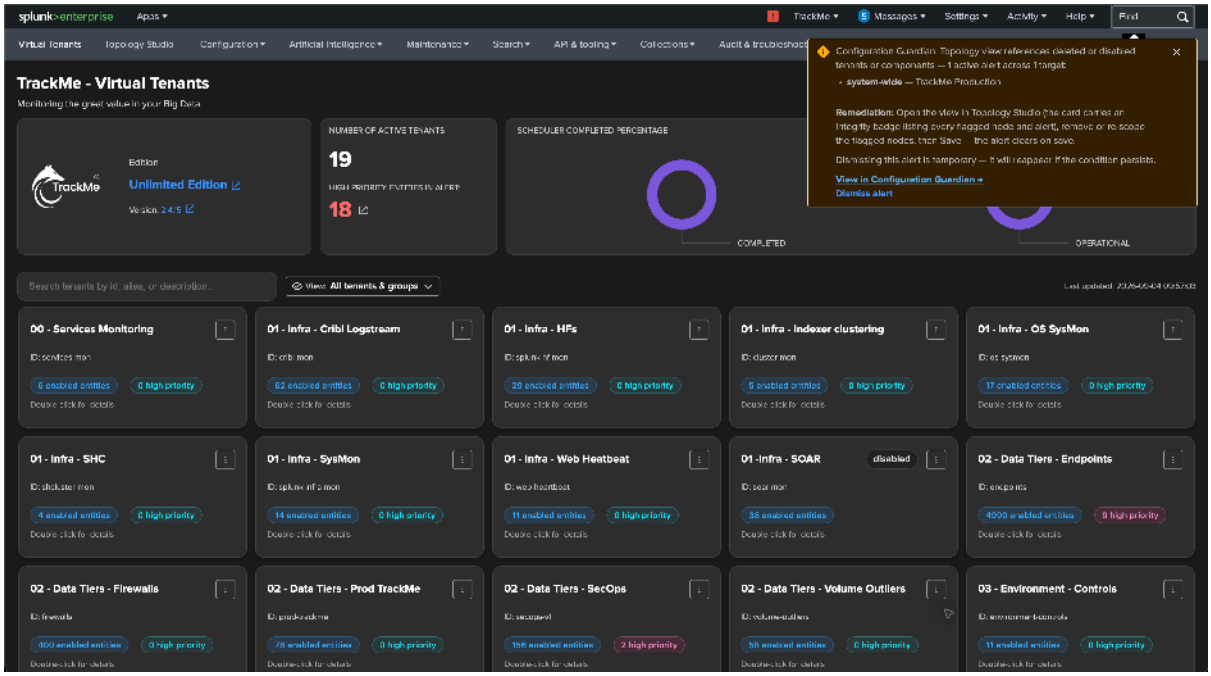
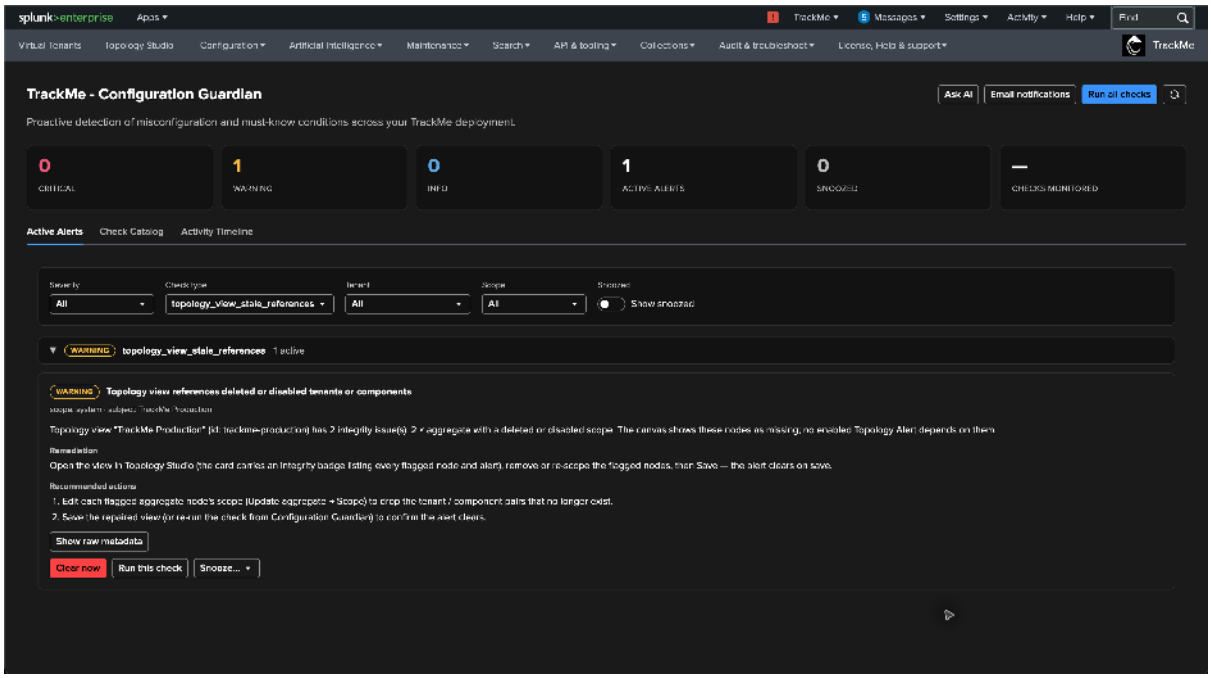
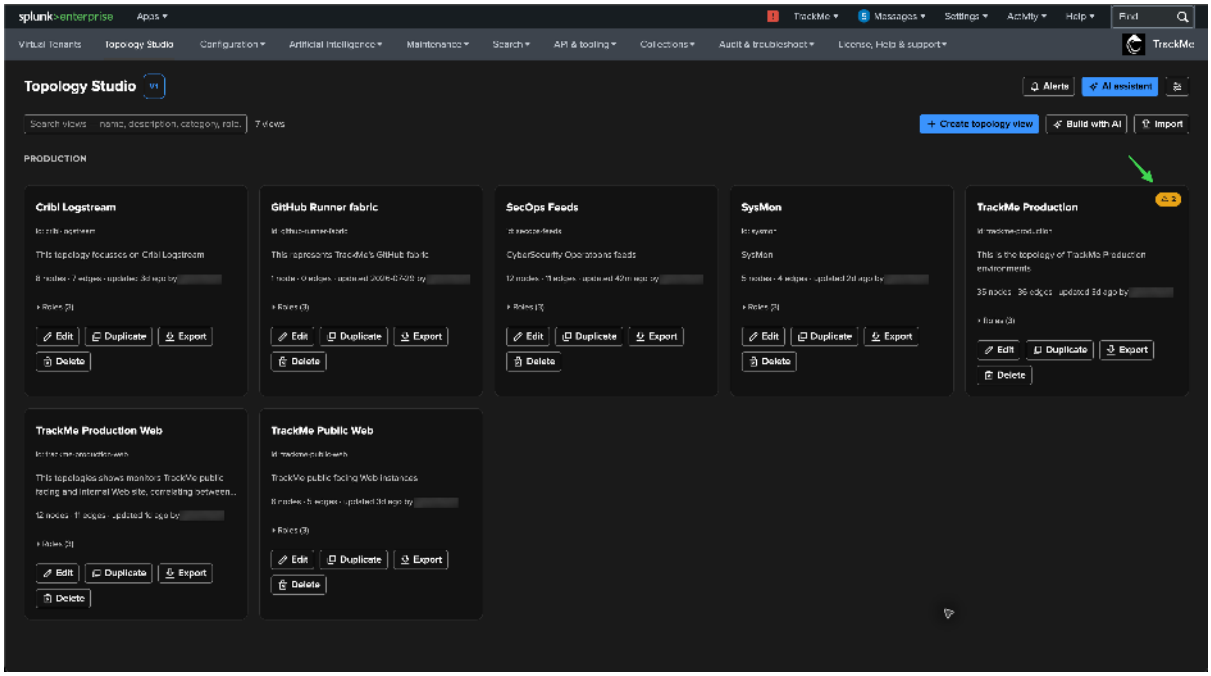
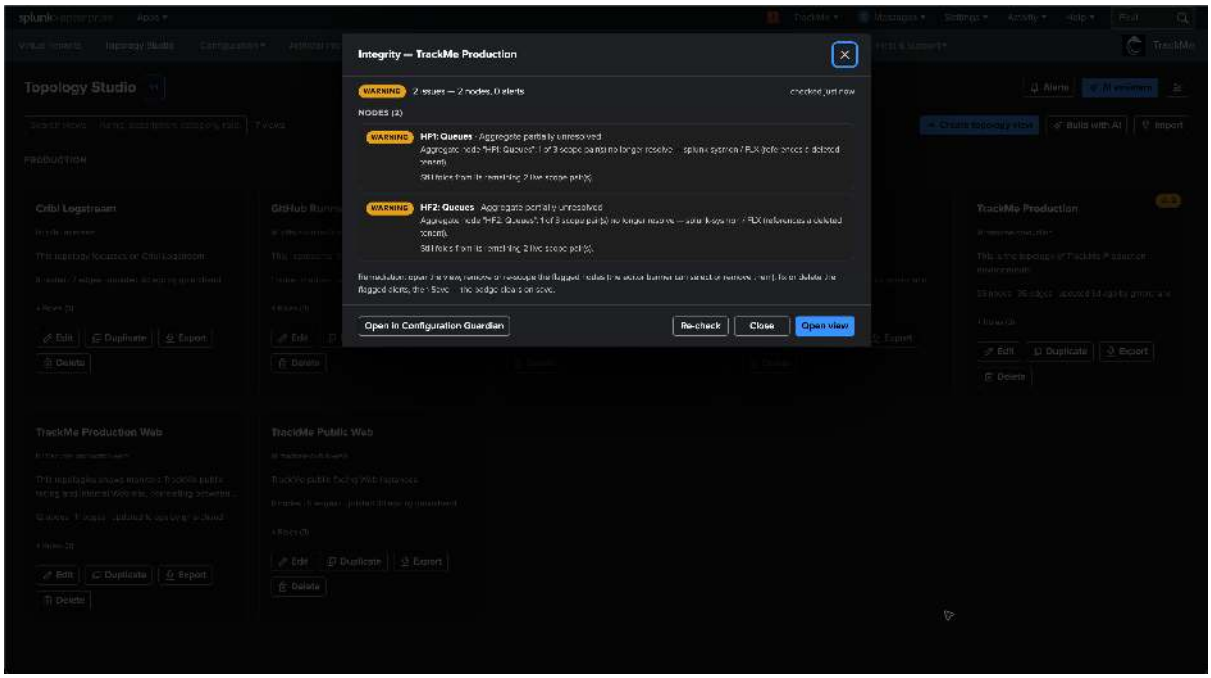
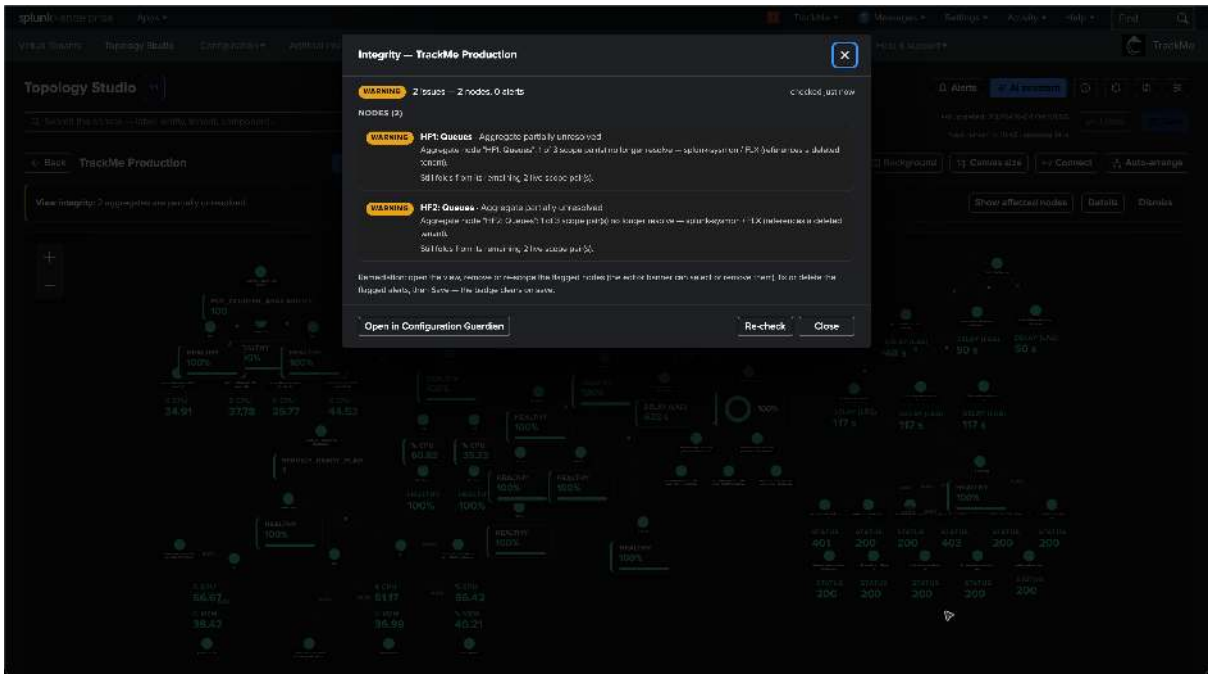
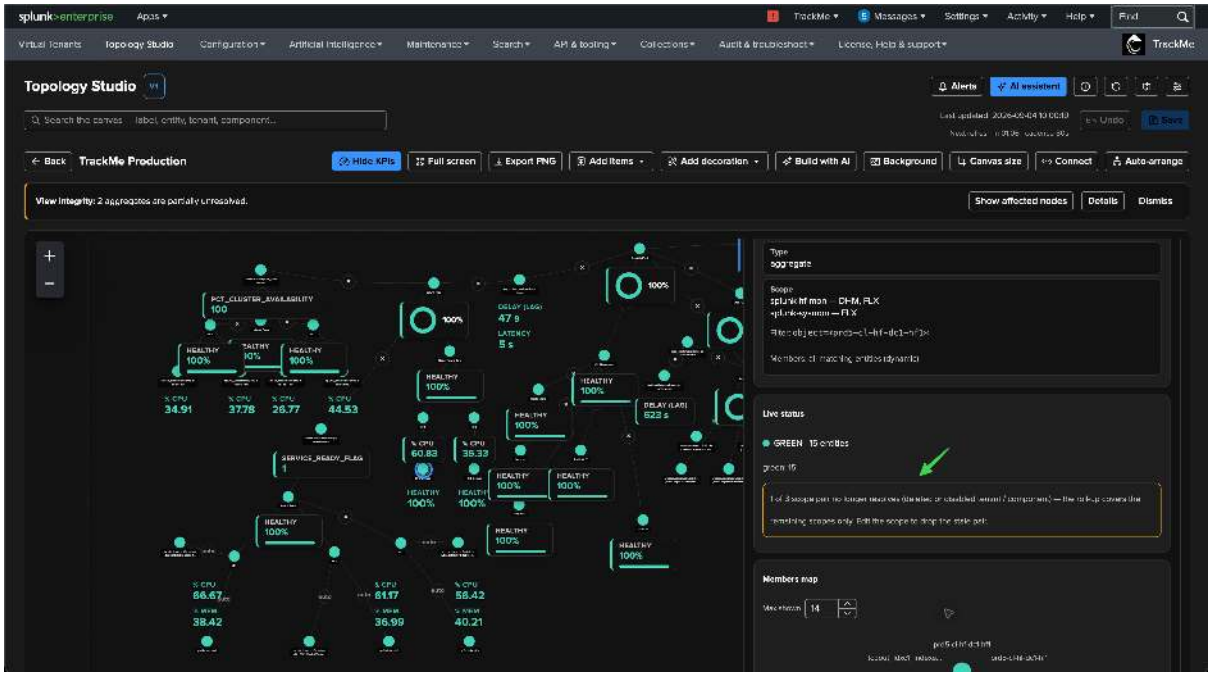
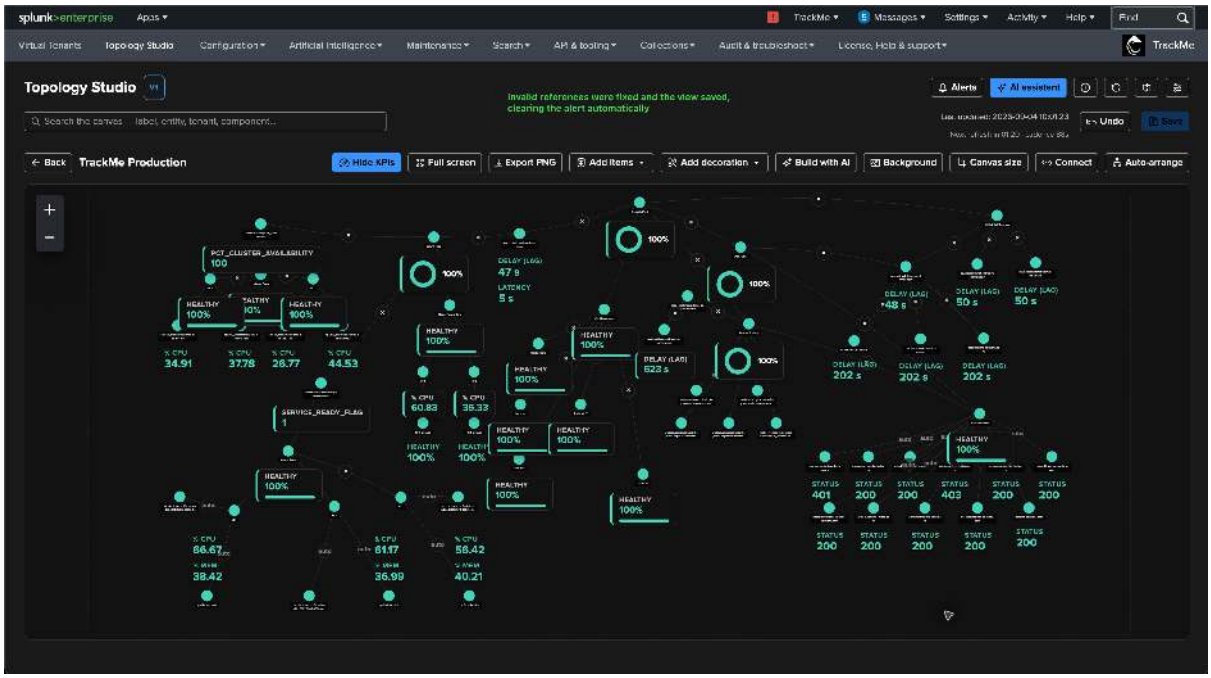
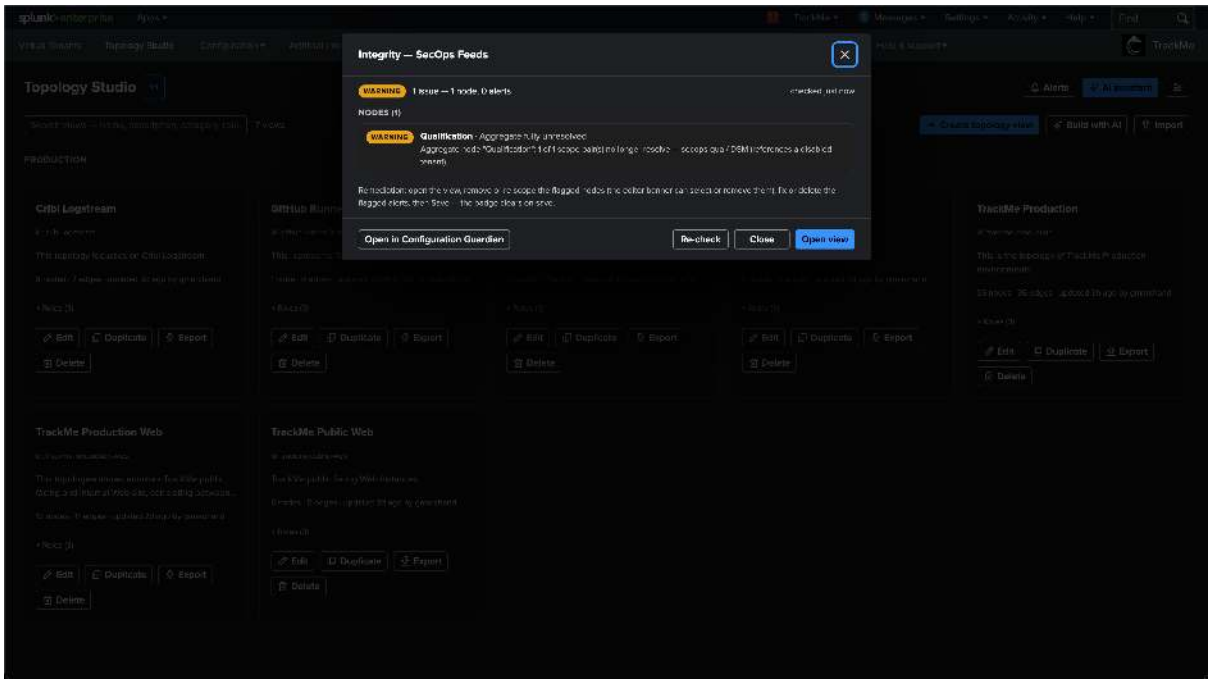
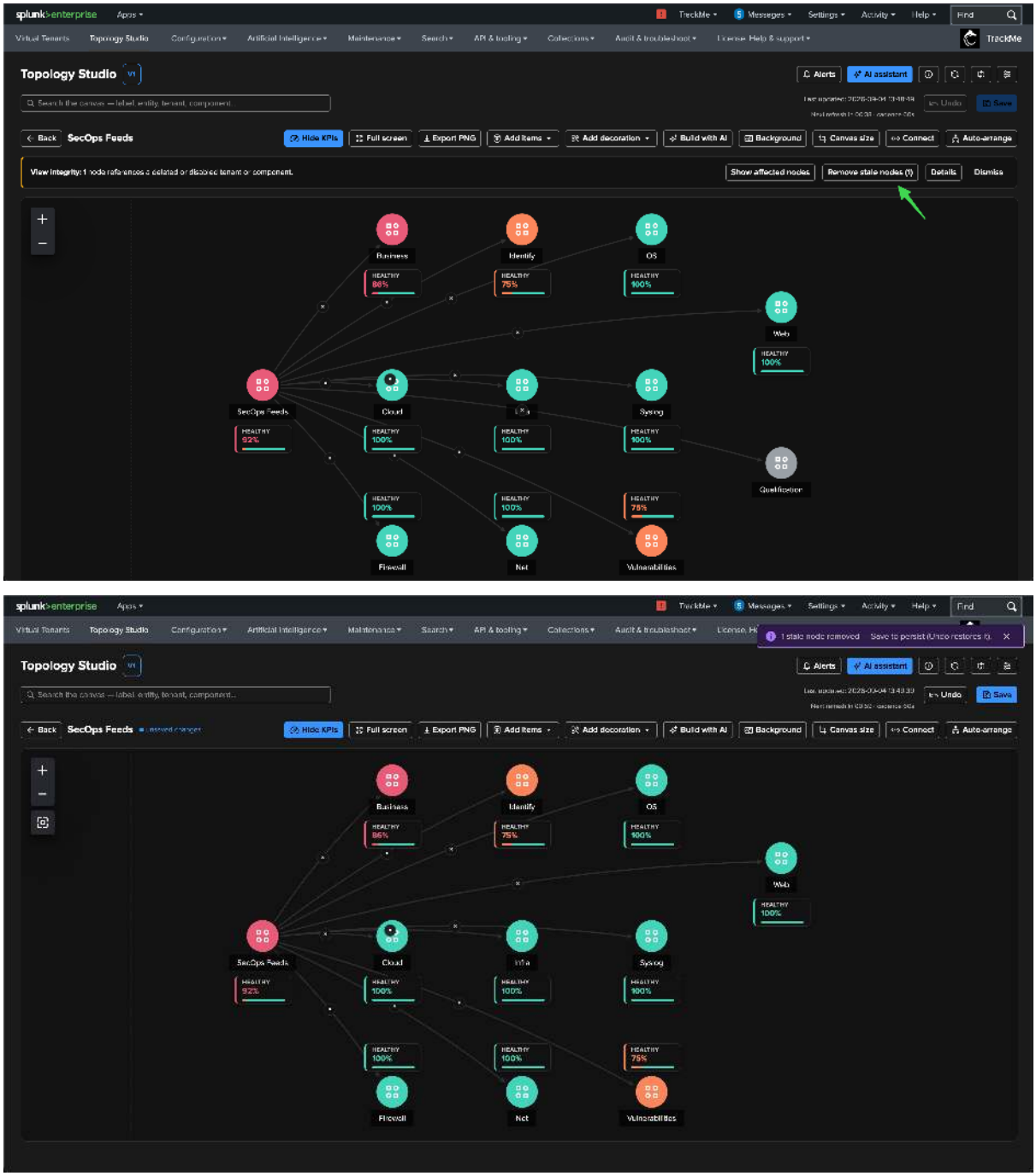
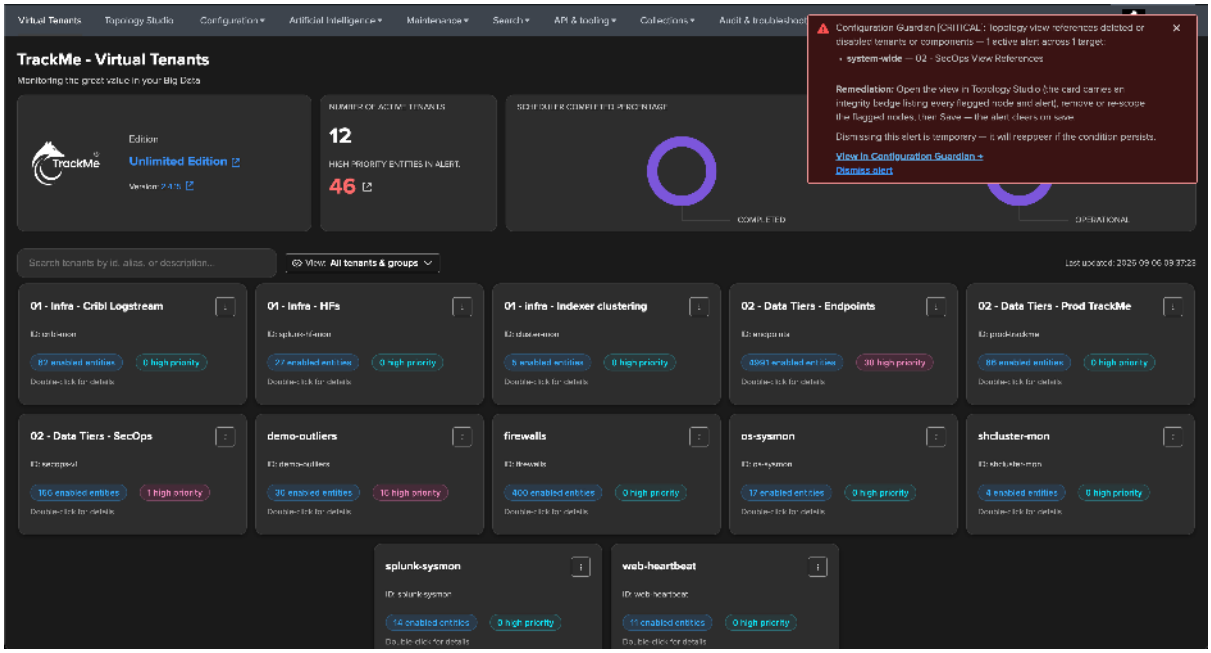
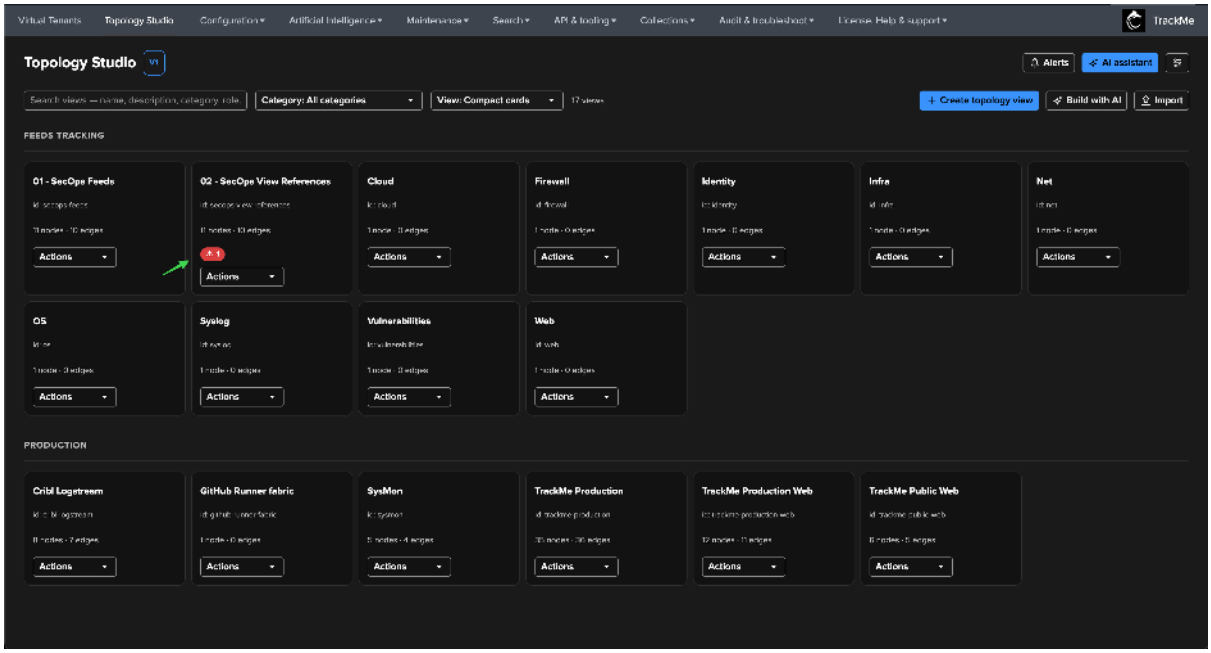
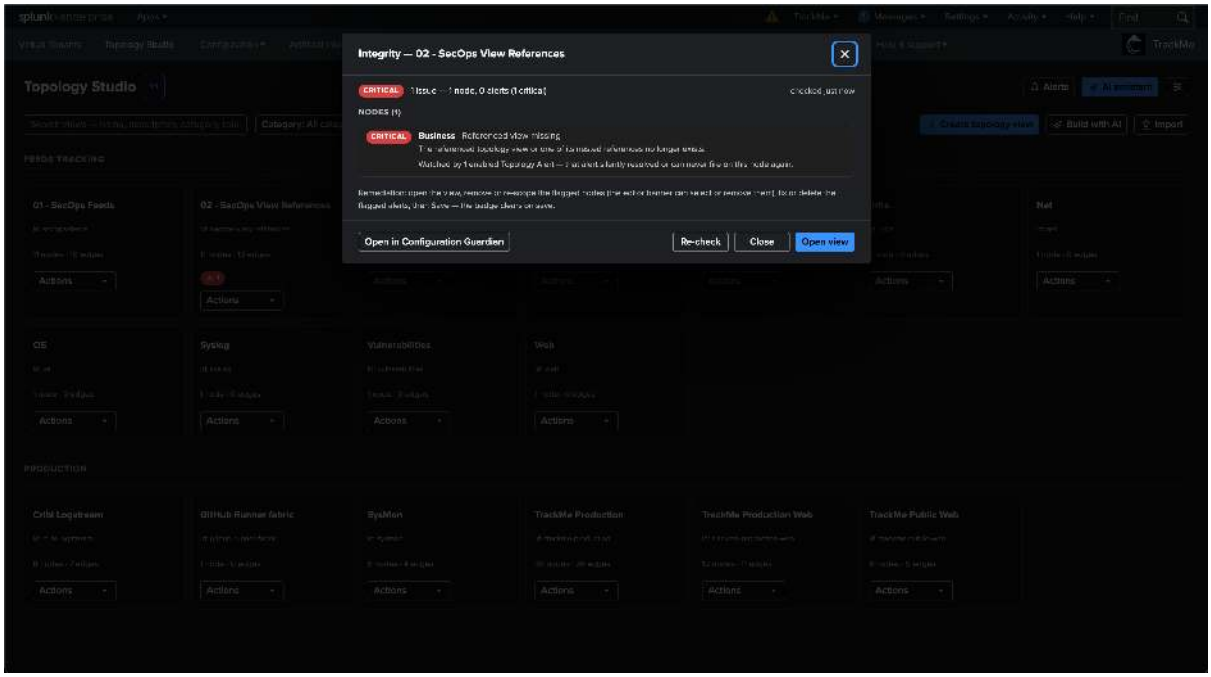
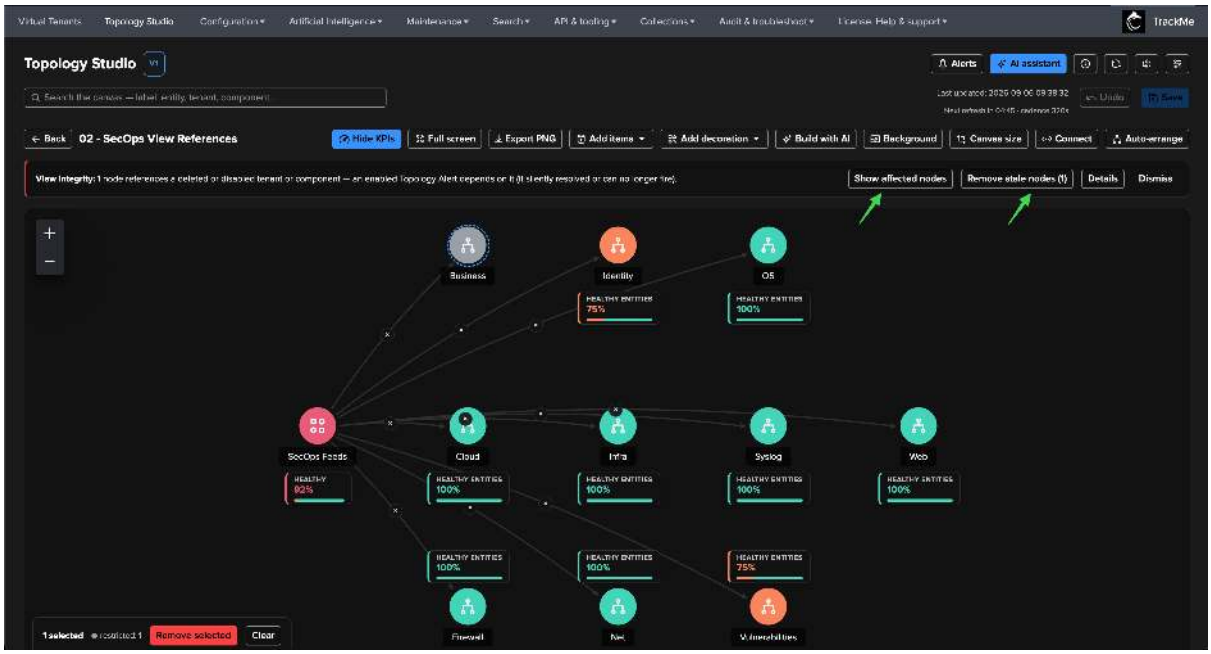
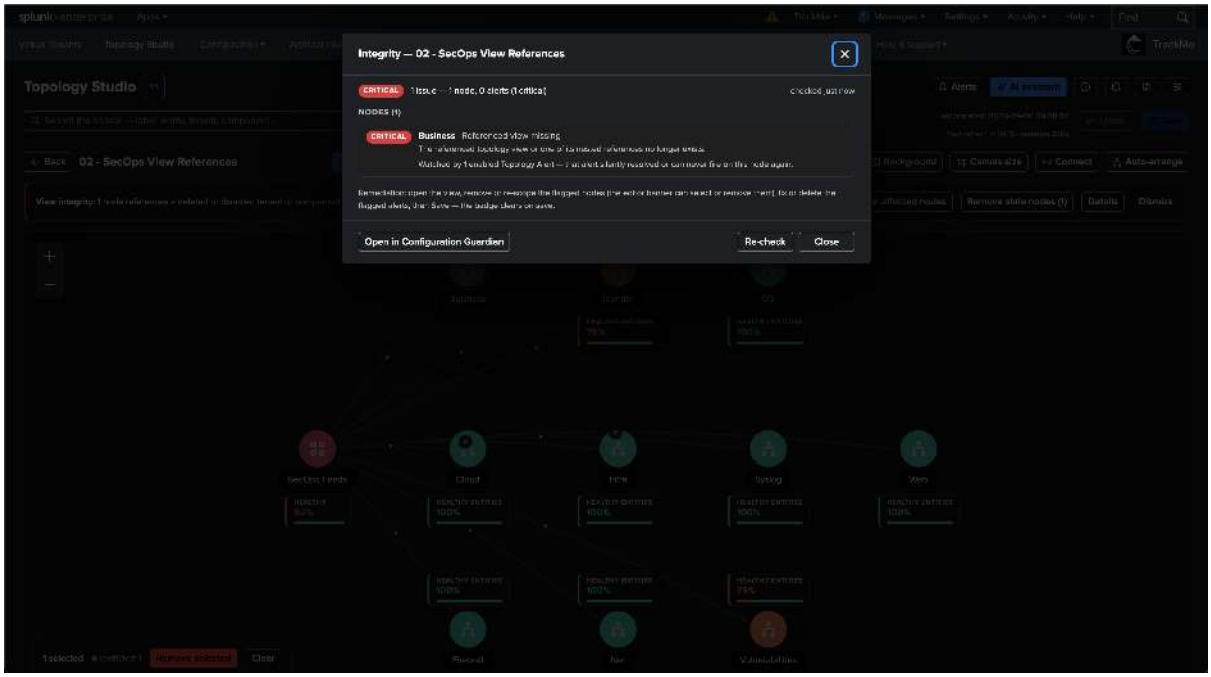
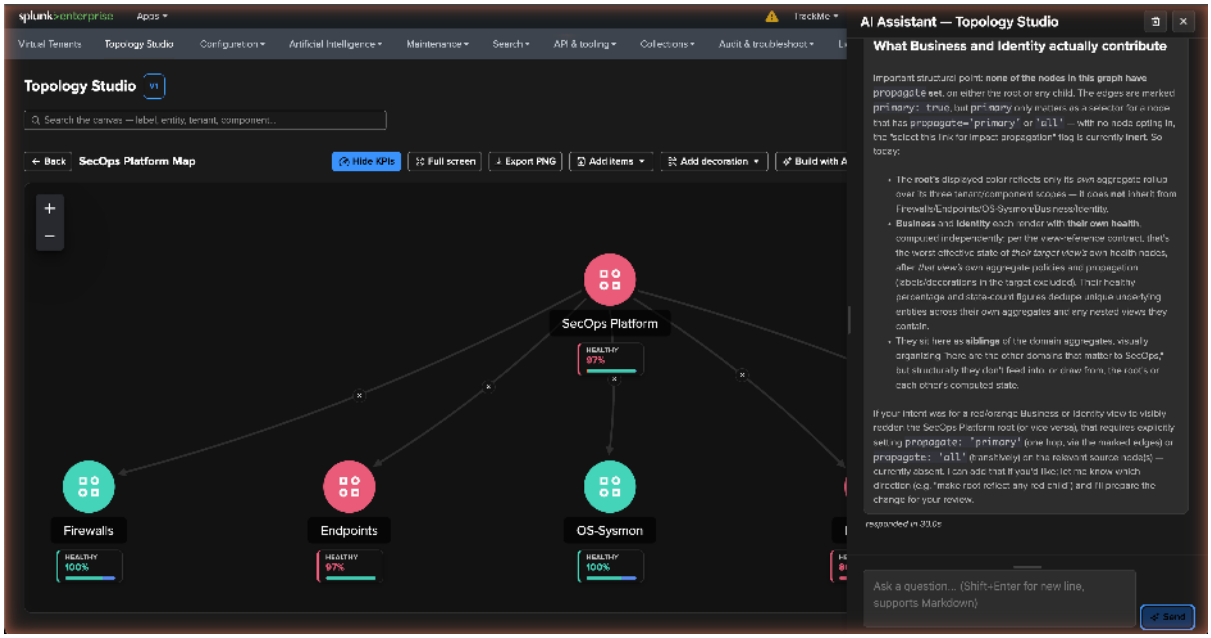
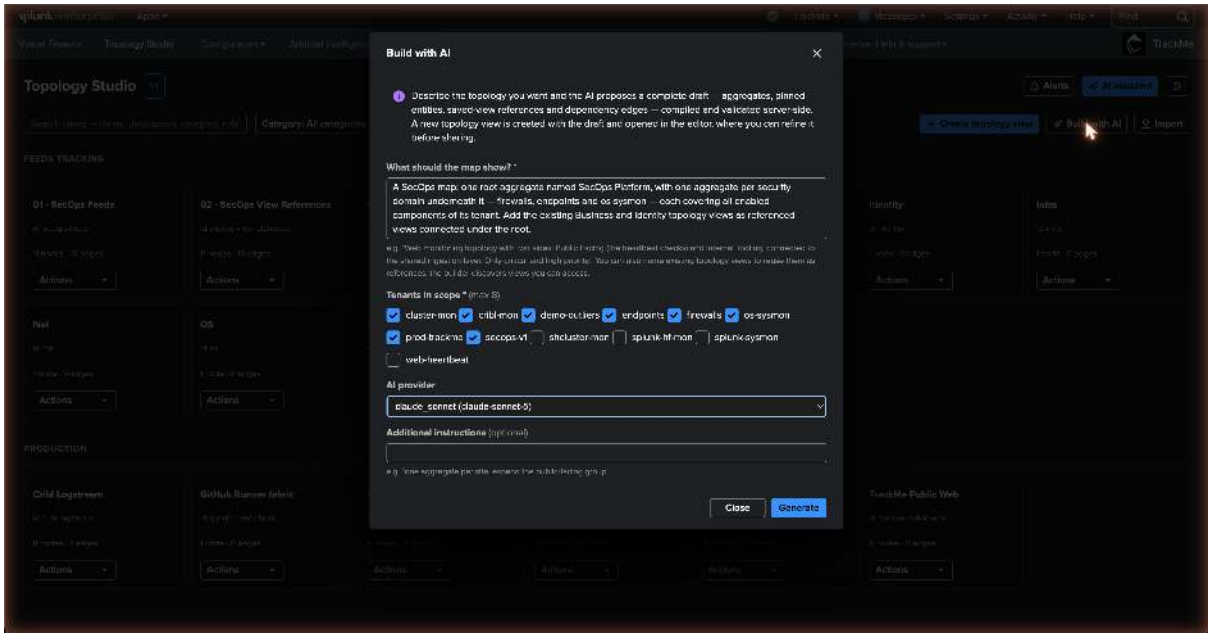
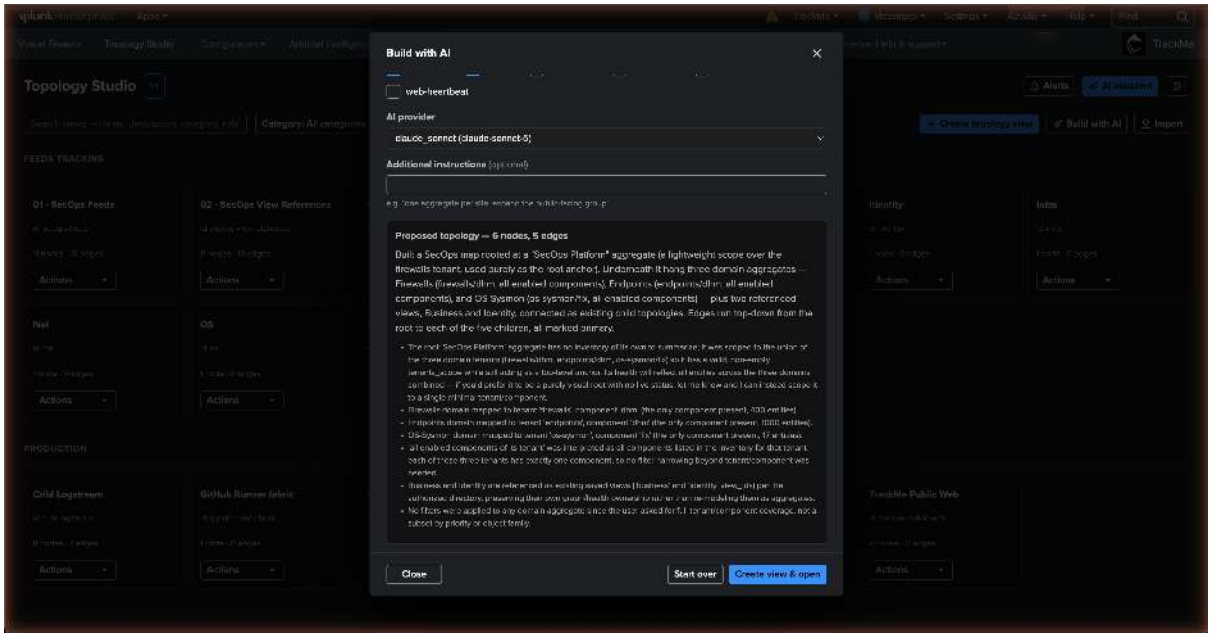
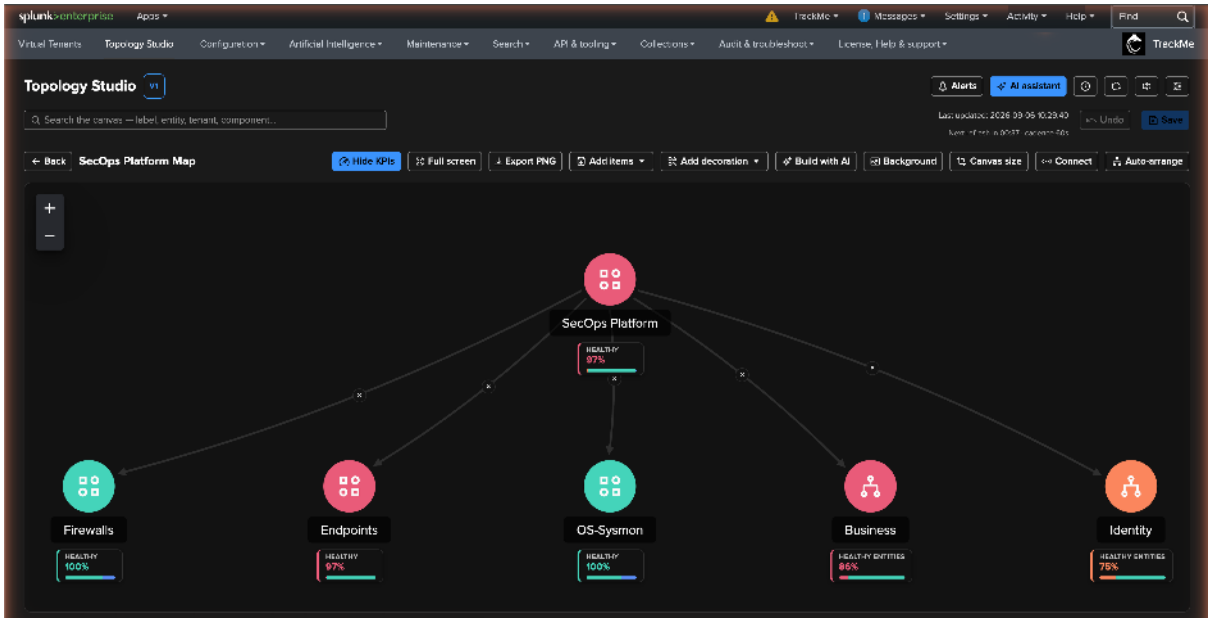
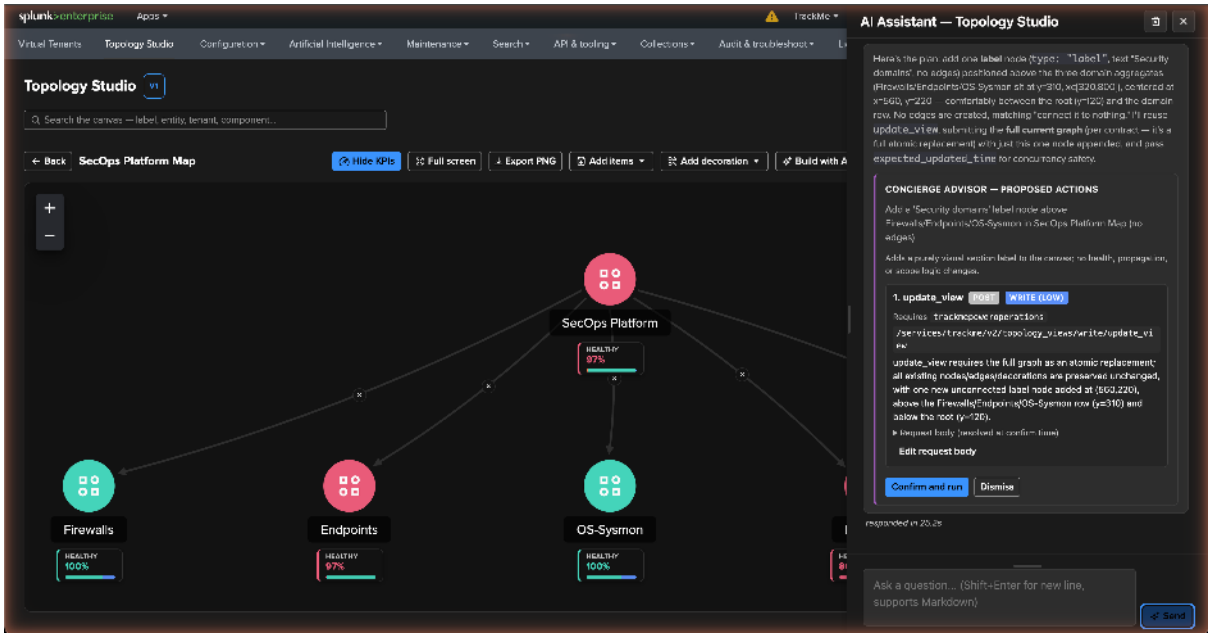
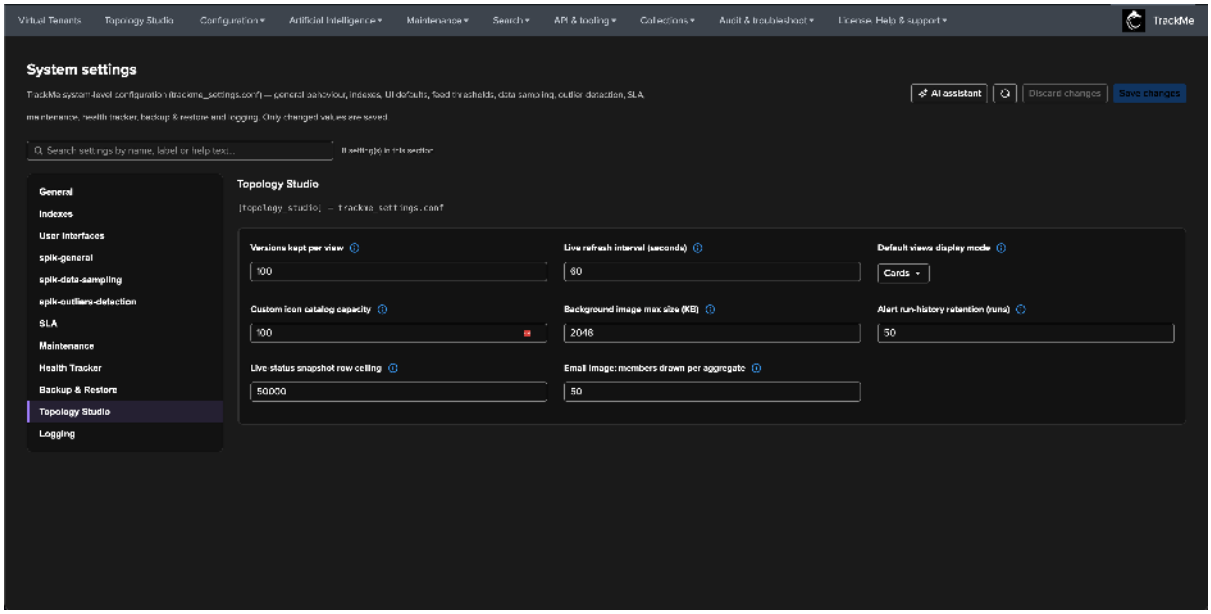
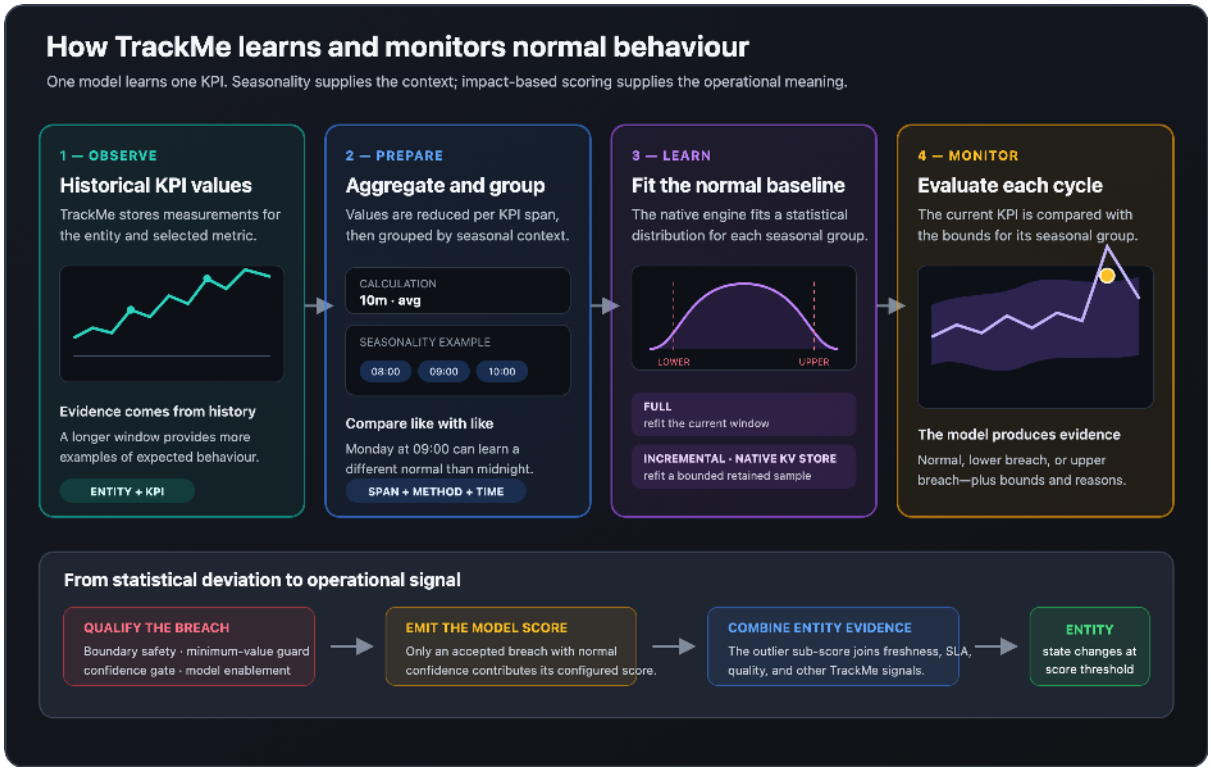
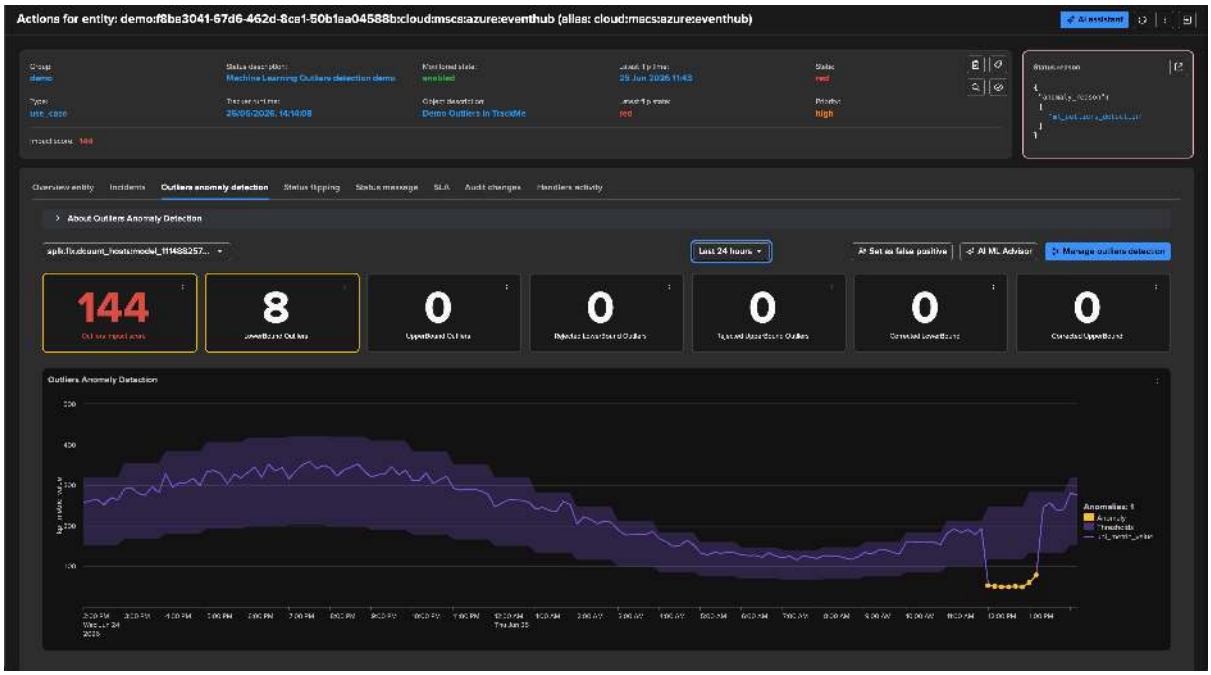
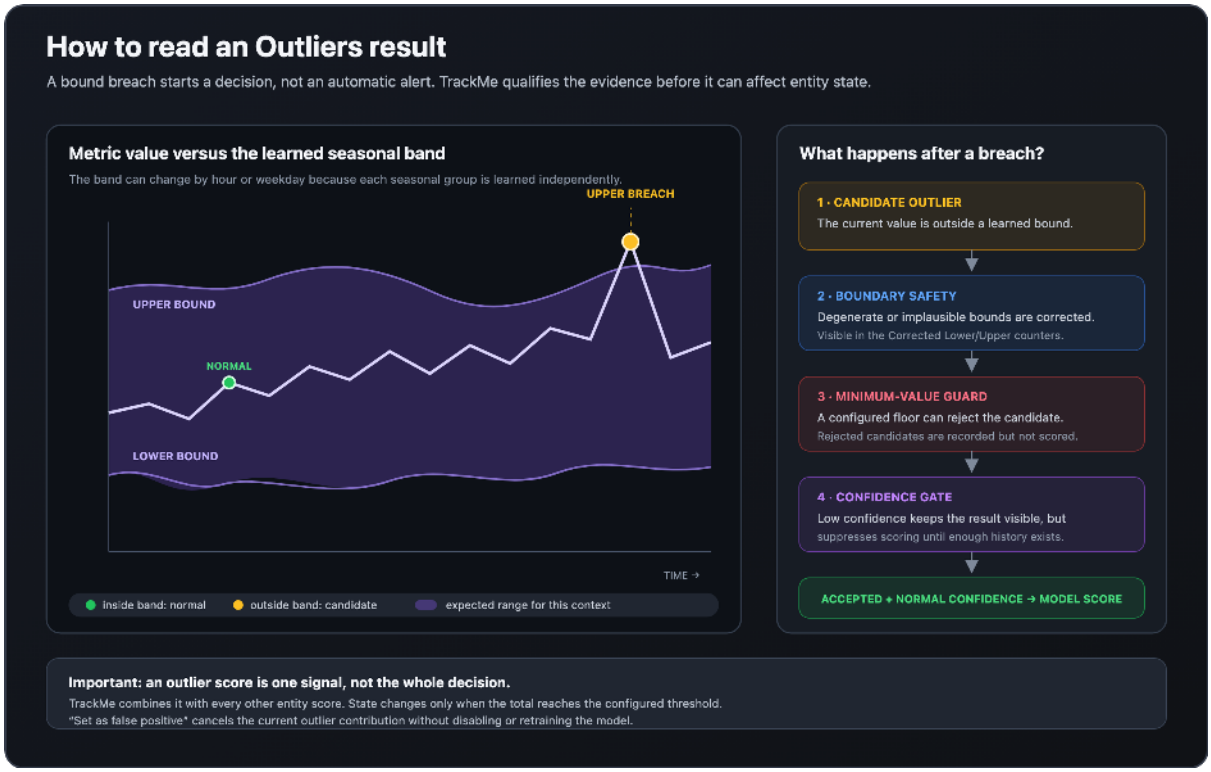
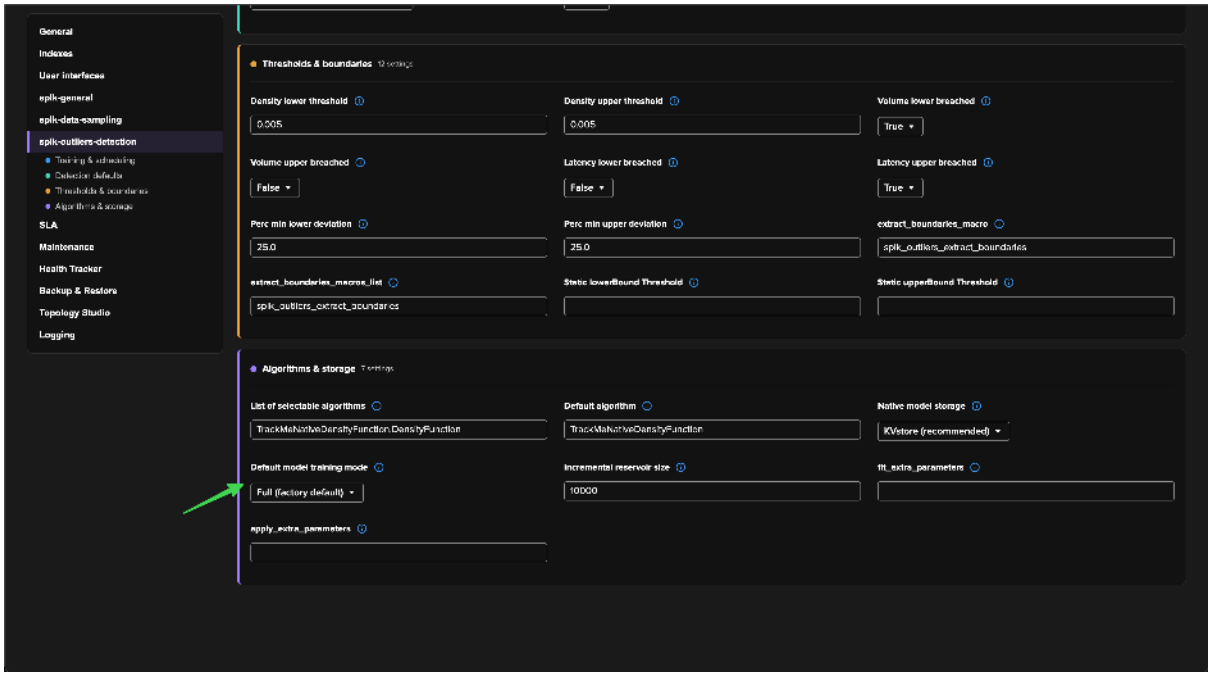
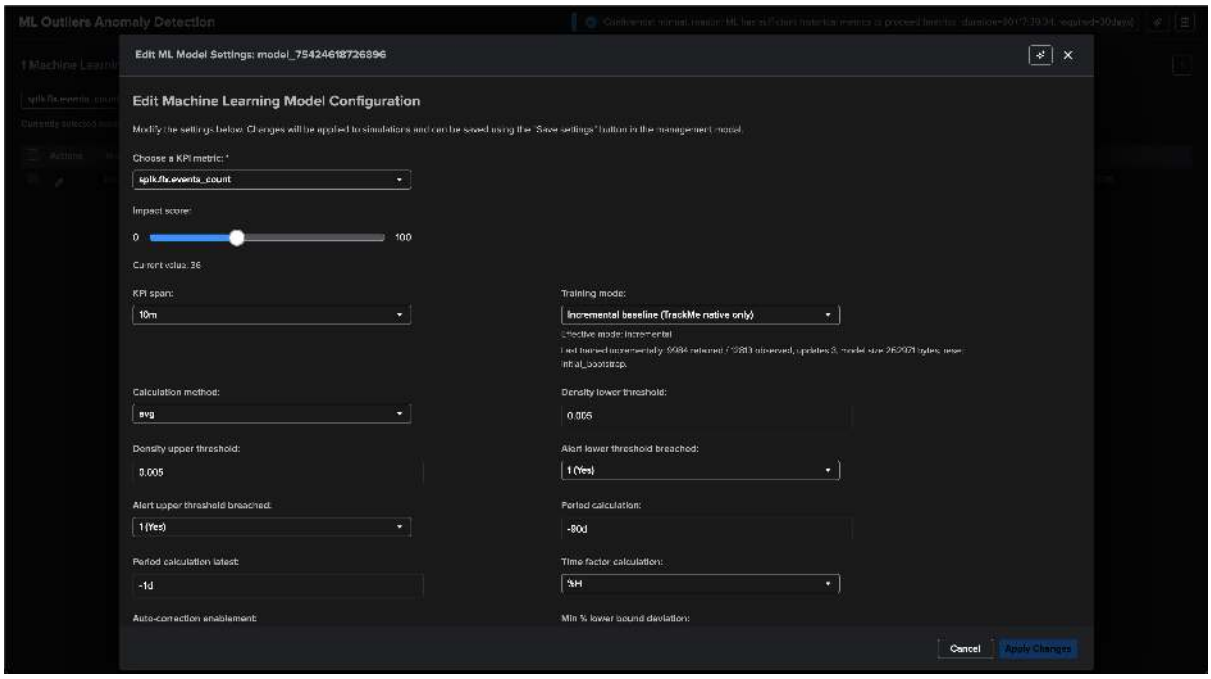
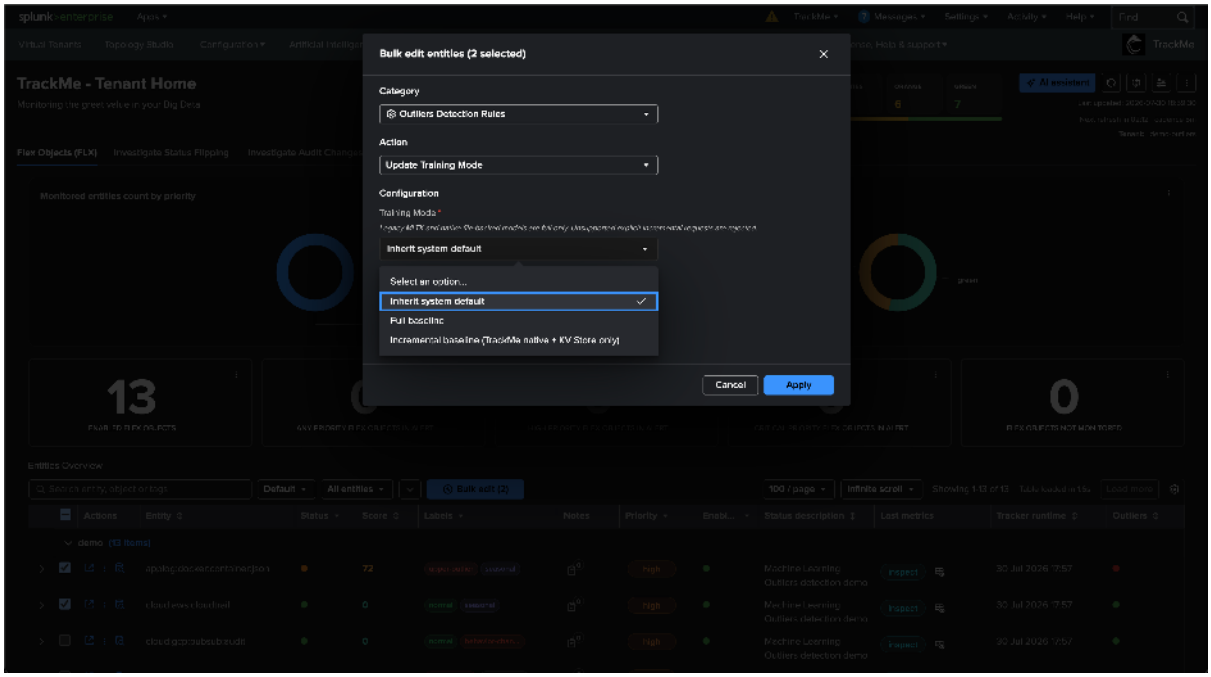
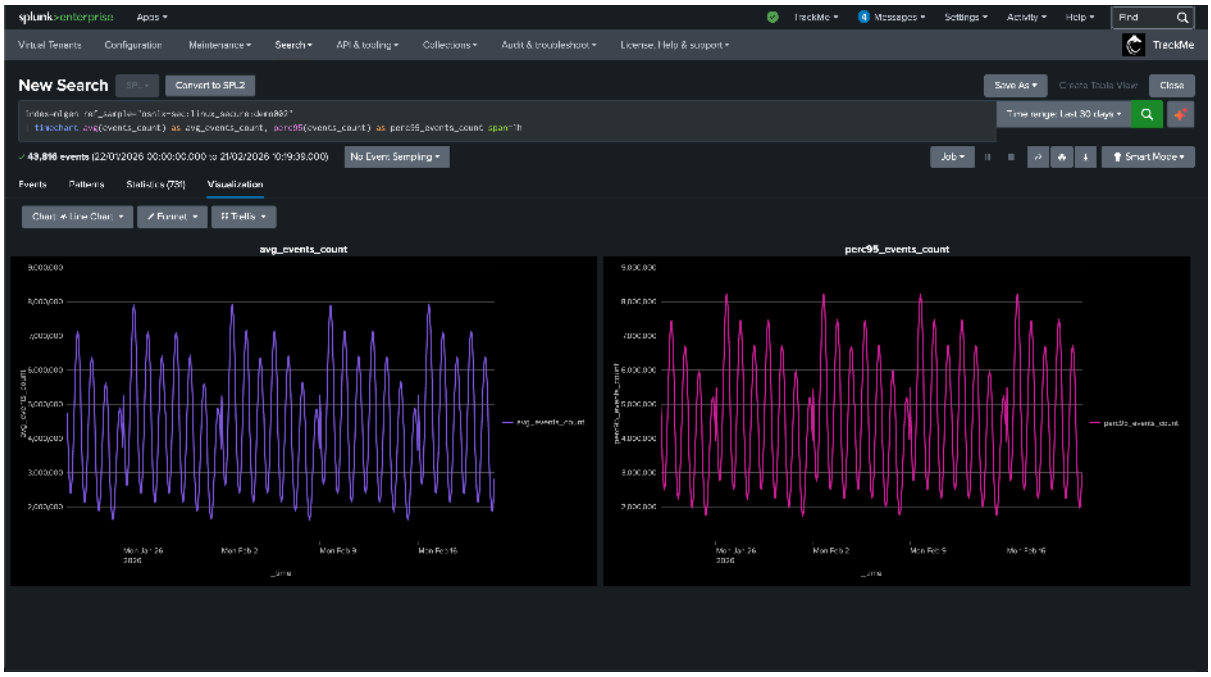
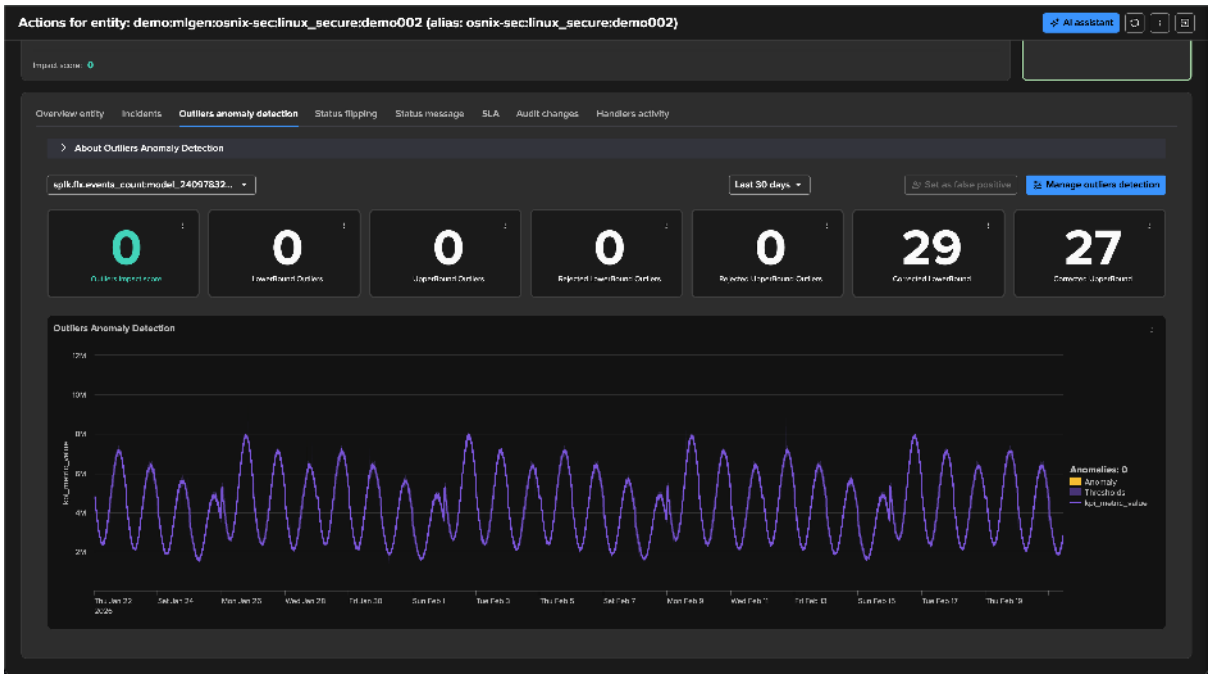
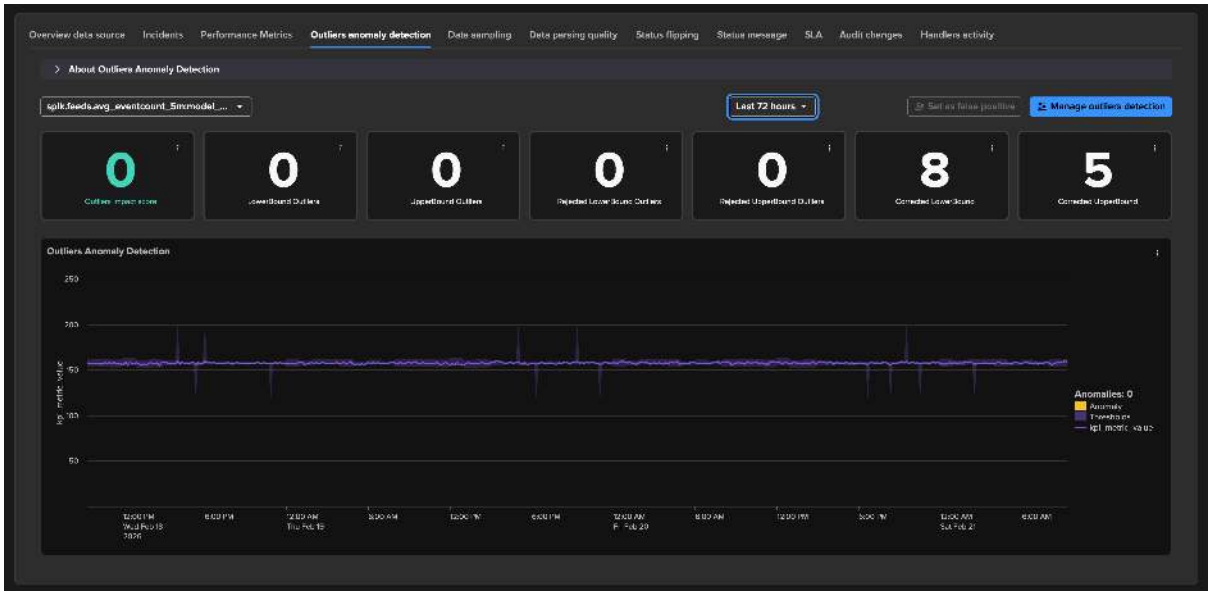
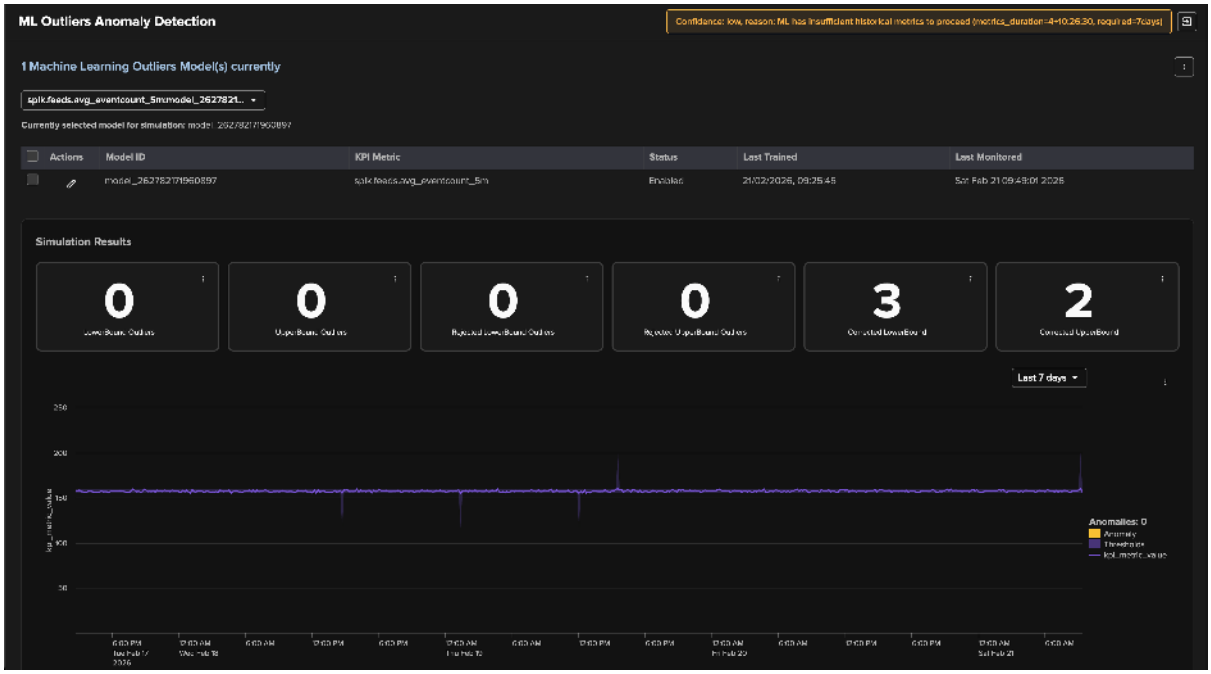
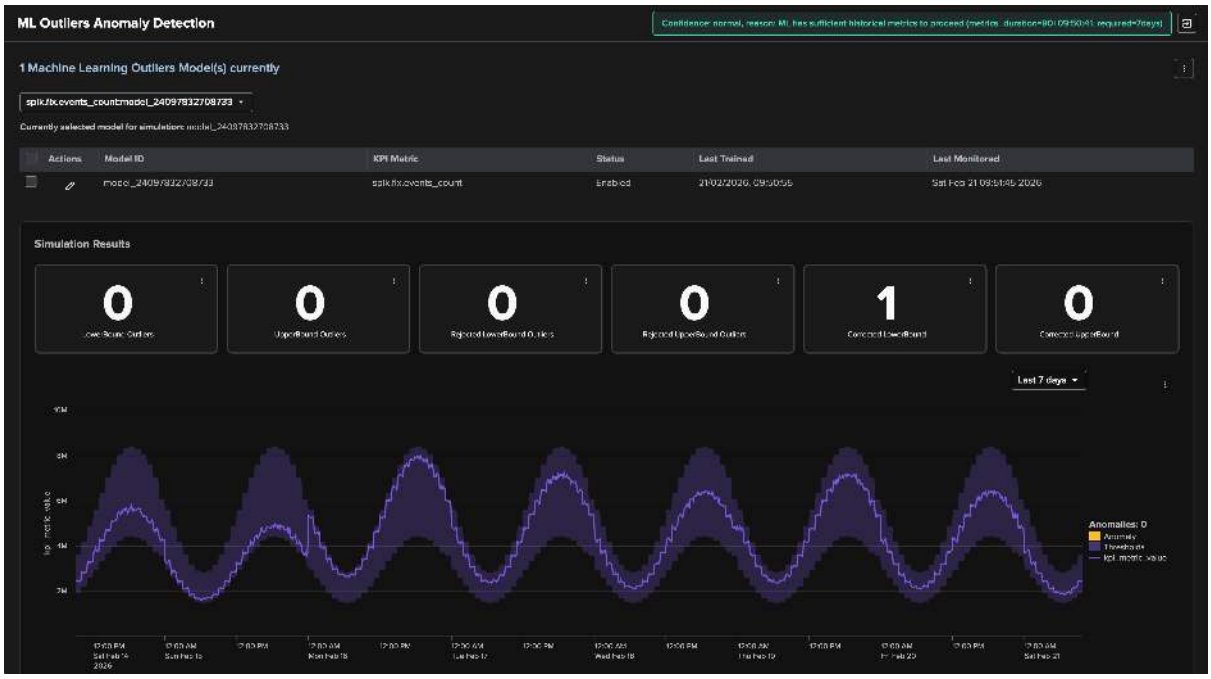
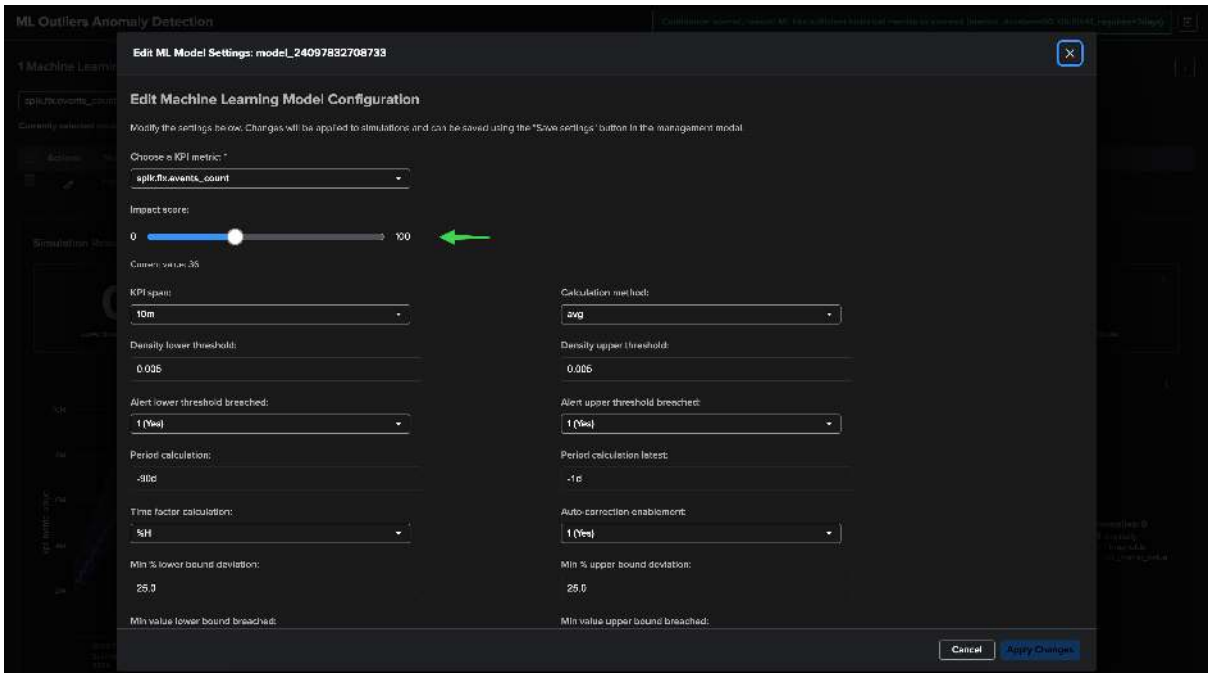
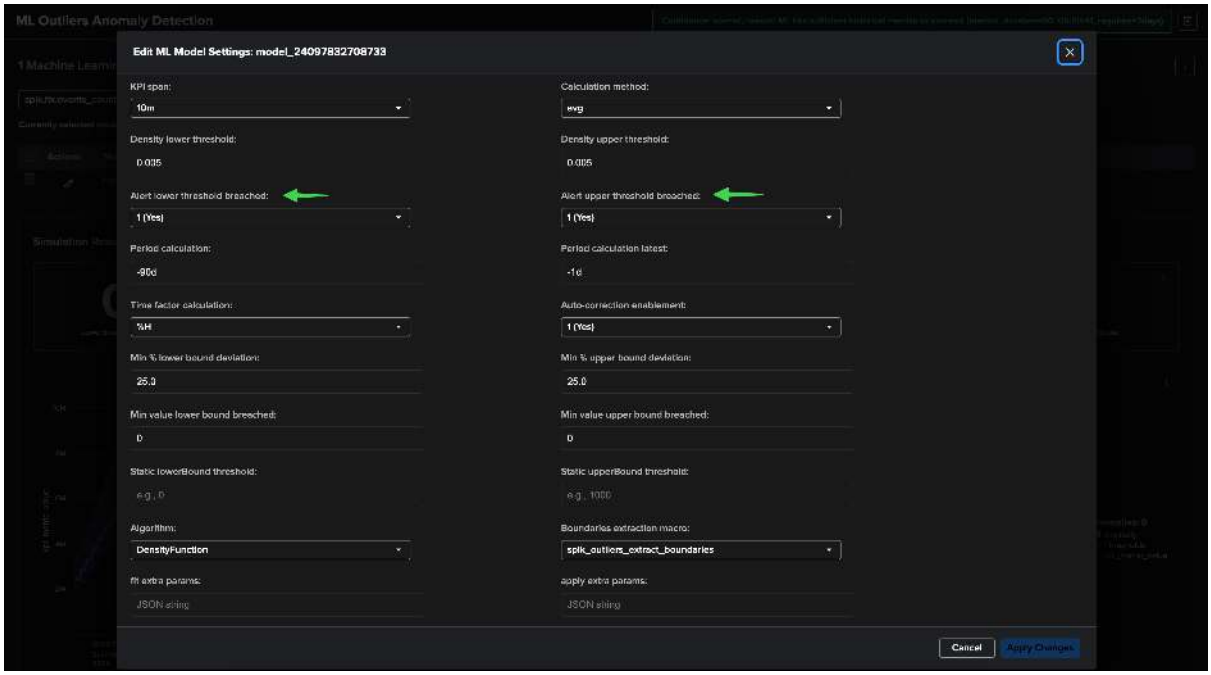
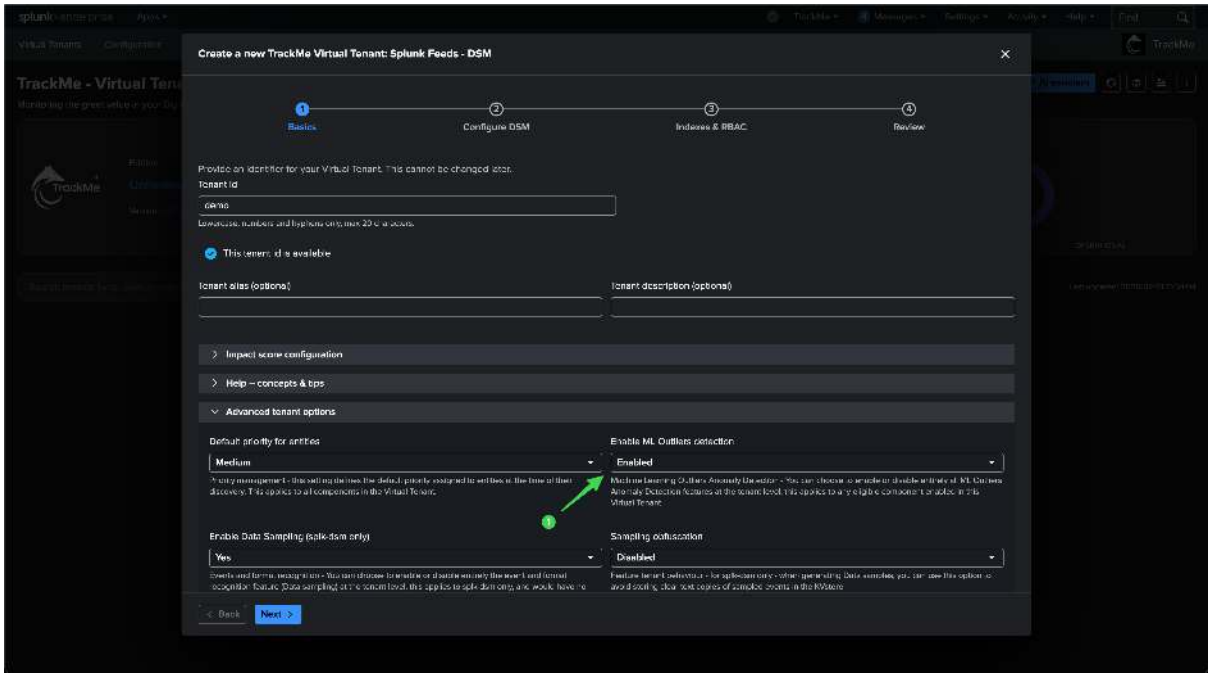
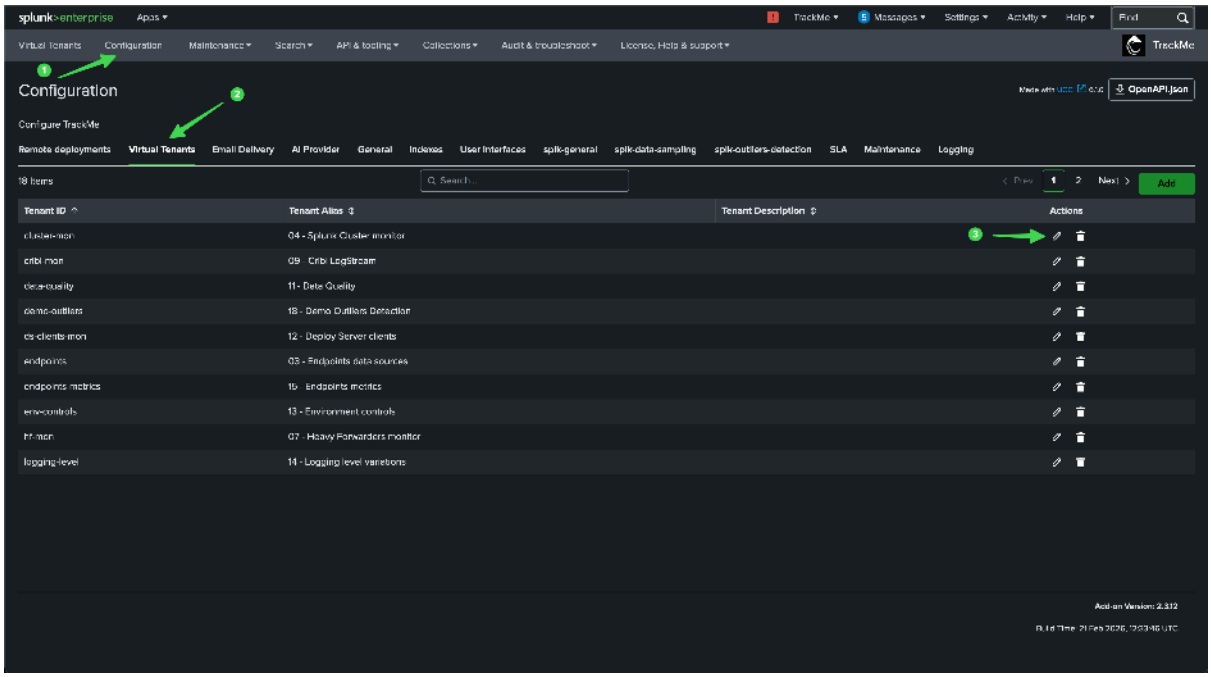
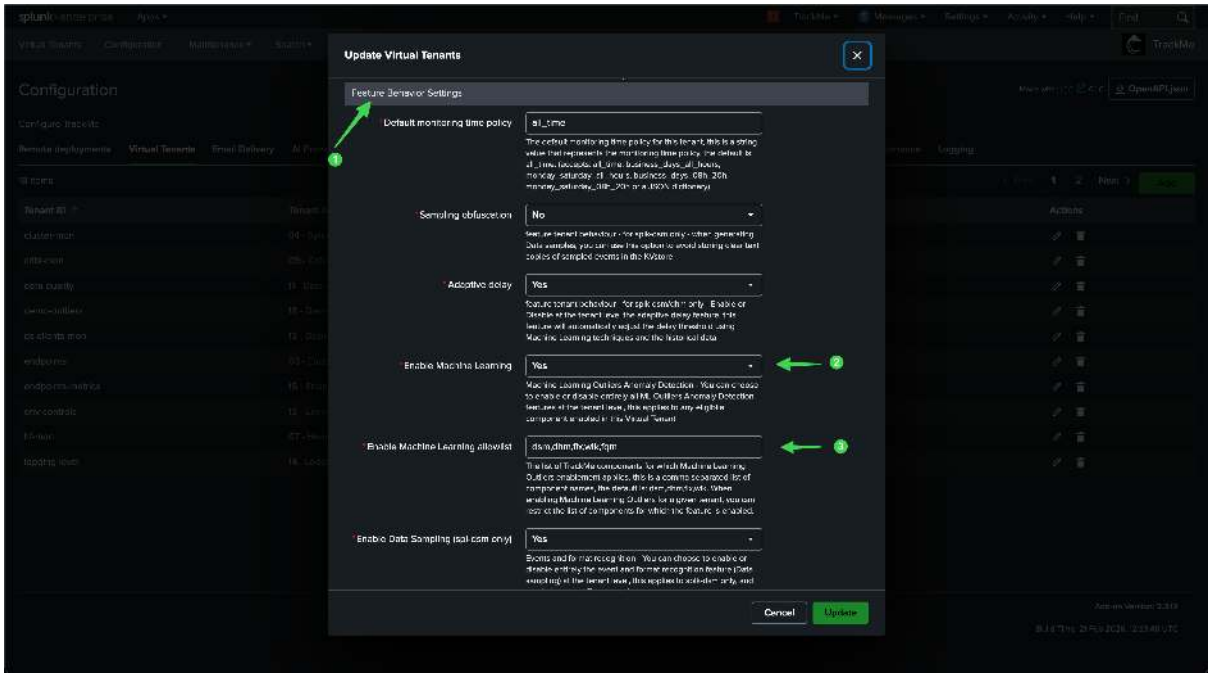
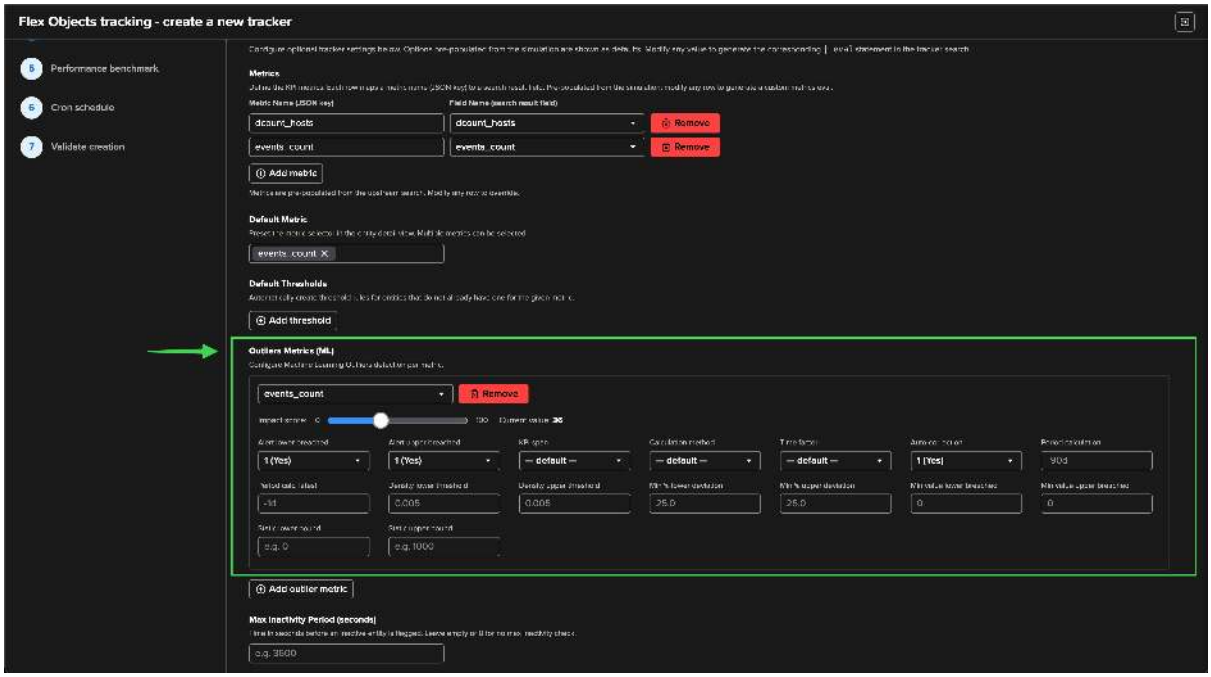
```
| inputlookup trackme_dhm_tenant_mytenant where monitored_state="enabled"
| fields object
| lookup my_cmdb_lookup host AS object OUTPUT pair_id
| eval object_group_name = "pair:" . pair_id
| where isnotnull(object_group_name) AND object_group_name!=""
| stats values(object) as object_group_members by object_group_name
| eventstats dc(object_group_members) as dcount_members by object_group_name
| where dcount_members>1
| trackmeautogroup tenant_id="mytenant" purge_single_member_grp=True
```

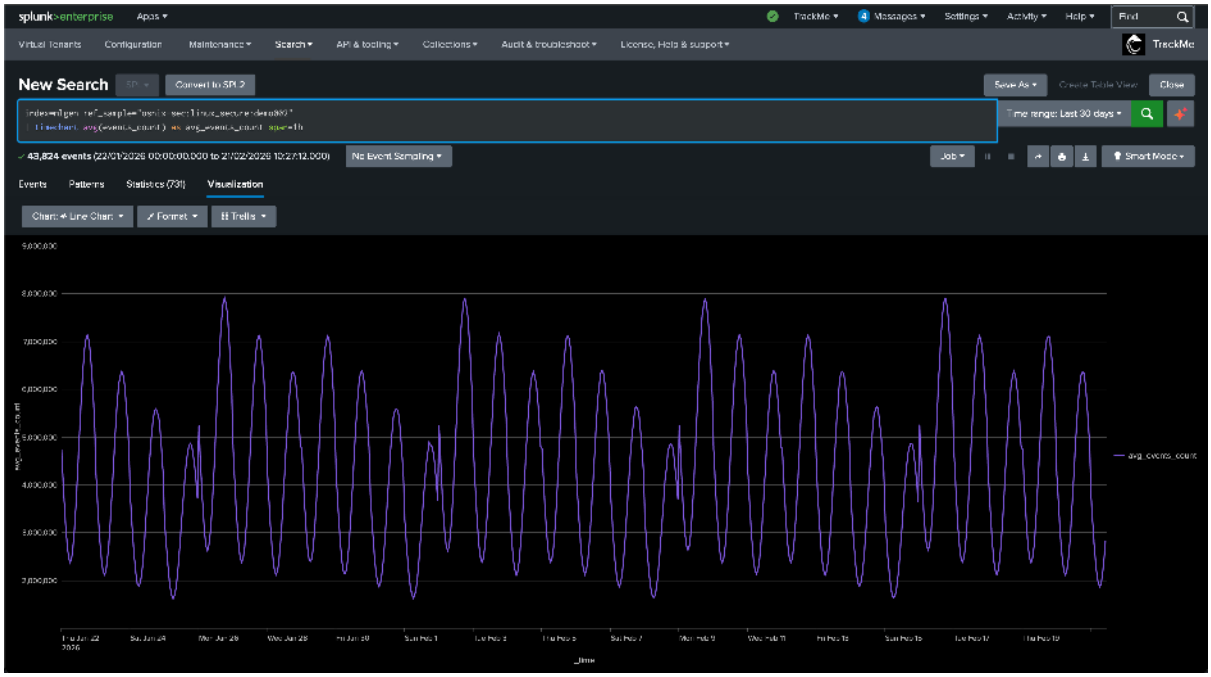
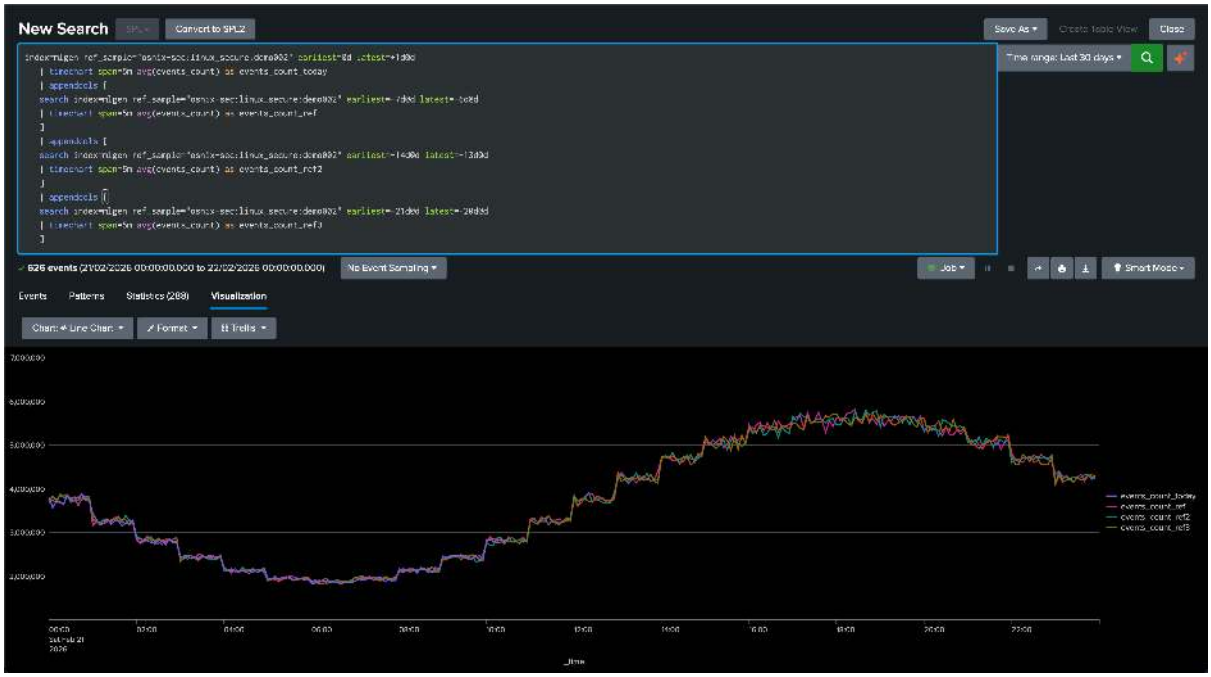
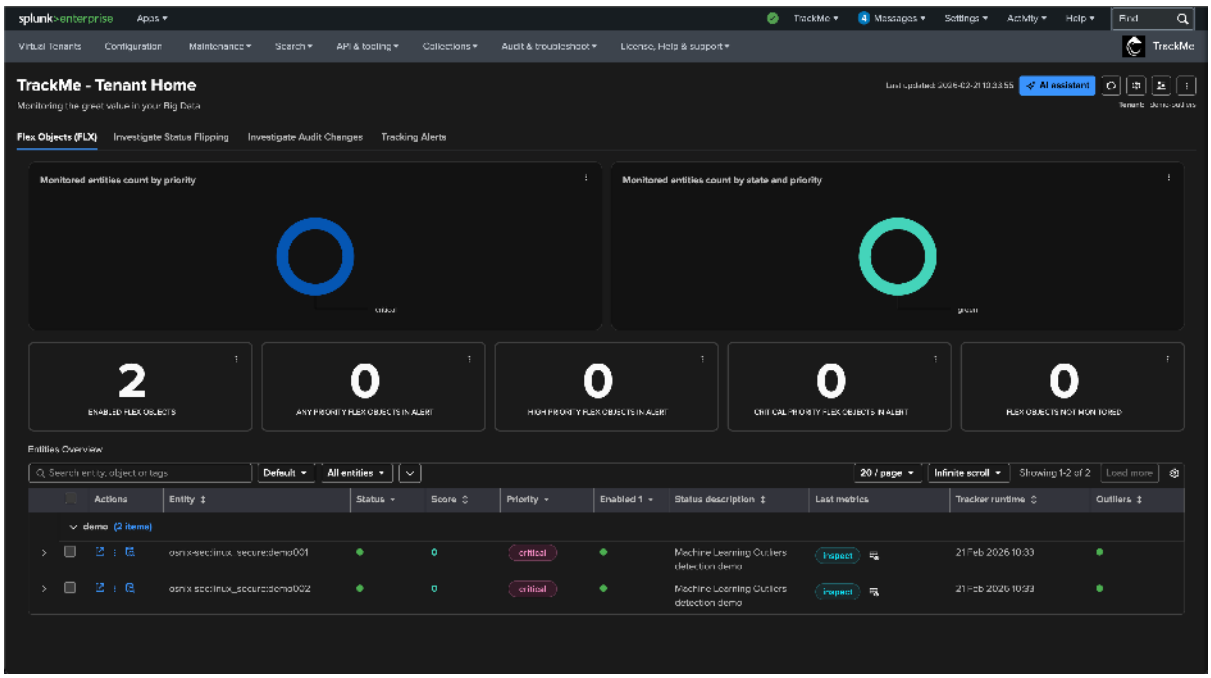
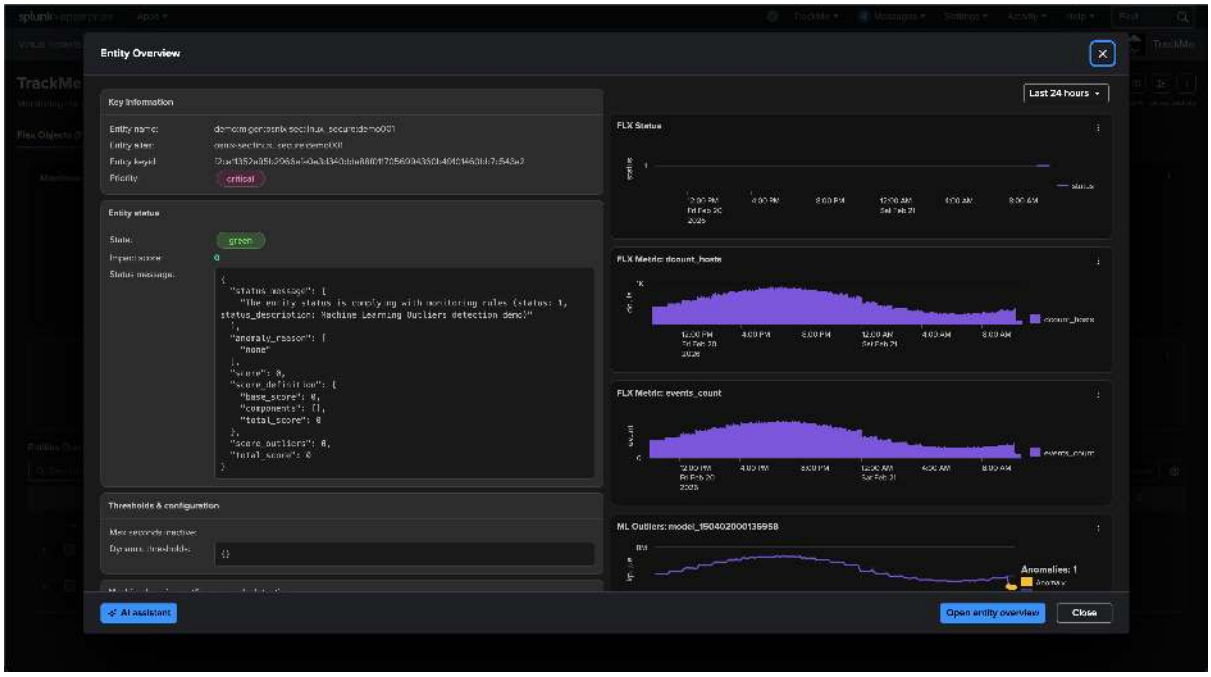
Scheduling it

Once the SPL is right, **save it as a scheduled report** to keep groups current as your estate changes (daily is usually enough). The command logs its actions to `index=_internal sourcetype="trackme:custom_commands:trackmeautogroup"`.

Important
```

This scheduled report is **yours**, not orchestrated by TrackMe — so if you later delete the tenant, remember to delete the report too.

See also

- *Disruption queue* — the other source of the blue state.
- *The entity state machine* — where group protection fits in the state machine.
- *REST API reference* — the REST API reference.
- *Virtual Groups* — *virtual* groups (cross-tenant views) are a different feature.

8.5.8 Disruption queue

Some entities dip into red for **short-lived, self-correcting reasons** — a two-minute indexing hiccup, a momentary source outage, a brief drop in host count. Paging on every such blip is pure noise. The disruption queue is the **safeguard against those false positives**: when an entity first goes red, TrackMe holds it in **blue** for a grace period and only lets it turn **red** (and alert) if it stays red *continuously* beyond that grace. A problem that clears within the window never alerts.

How it works

Each entity has a grace duration. When it first goes red, a timer starts and the entity is shown **blue** (suppressed). From there:

- if it is **still red** after the grace duration, it becomes **red** and can alert;
- if it **recovers** at any point, the timer resets — the grace measures *uninterrupted* disruption only, so the next red run starts the clock again.



While an entity is in the grace window its status message says exactly that — it has an active anomaly, but the disruption duration has not yet reached the minimal time:





Configuring the grace period

The grace duration cascades — the most specific wins:

1. a **system default** (typically 5 minutes);
2. a **tenant default** — set under **Configure tenant → Disruption Queue → Default disruption grace period**, applied to every entity in the tenant;
3. a **per-entity override** — set from the entity's *Disruption queue* action (0 seconds up to 48 hours).

For the **Flex Object (splk-flx)** component there is an additional level: a default disruption grace can be set at the *Flex Object tracker* level, applied to the entities that tracker discovers. Setting the grace to **0 disables it** — red is immediate, no grace.

Note

The disruption queue applies to all components and was introduced in TrackMe 2.1.18.





What blue buys you

While the entity is protected, it is **blue** — and blue is genuinely quiet, not just visually different:

- **No alert** opens, updates, or closes for it.
- It does **not** count as red against *SLA* — a protected entity keeps its compliance and accrues no critical-red time.
- It does **not** flap red ↔ green; the grace absorbs the churn (see *Status flapping*).





Note

- The disruption queue is one of the **two sources of the blue state**; the other is *logical-group protection*. If both apply, the group is named in the status message.
- It **defers** the red state — it is not an "I know about this, stop telling me" switch. For that, use *acknowledgment* (which is independent — an entity can be both acked and in the grace window).

Warning

Tune it to your environment. **Too long**, and an entity that is genuinely broken can sit in blue and never reach red. **Too short**, and entities flap blue/red. Match the grace to how long your real, transient blips actually last.

See also

- *The entity state machine* — where the grace fits in the state machine.
- *Status flapping* — using the grace window to tame flapping.
- *Logical groups* — the other source of the blue state.
- *Acknowledgments* — acknowledgment, the deliberate "silence me" tool.

8.5.9 Notes

Notes are free-form, Markdown annotations you attach to an entity — the running log of *what is going on with this thing and what we did about it*. They are perfect for investigation context, runbook and ticket links, workarounds, and hand-offs between shifts, without cluttering the entity's structured fields. Notes are available on **all seven components**.

What a note is

- **Free-form Markdown** — links, lists, tables, code fences, and inline formatting all render in the entity's Notes panel.
- **Per entity, multi-note** — an entity can carry **any number** of notes, shown newest first; they are never merged or deduplicated.
- **Authored** — every note records **who** created it and **when**.
- **Immutable** — there is no edit action. To change a note you delete it and create a new one; the audit trail records both. Each note card offers only **clone** and **delete**.

Anyone can **read** notes; **creating, cloning, and deleting** them requires the power role.



Markdown, links, and reports

Because notes are rendered as Markdown, they are far more than a comment box. Drop in a runbook or ticket link (`[INC-1234](https://…)`), a checklist, a table, or a fenced command. It is even a natural home for an *AI Advisor* investigation — save its report as a note and the whole team has the context:



Note

TrackMe renders your Markdown as-is and does not sanitise it — write Markdown, and do not paste untrusted HTML.

Creating, expiring, and cloning notes

When you add a note you choose a **validity period — Permanent**, or an expiry of up to 365 days. An expiring note shows an *"Expires on …"* badge and is **automatically purged** once it lapses, so time-boxed context ("waiting on the vendor until Friday") cleans itself up. Permanent notes are never purged.



A note can also be **cloned to many entities at once** — the picker copies the note's *text* to every entity you select, and each target gets its own independent note (so editing or deleting one never affects the others). This is the quick way to spread the same context across a set of related entities; cloned notes are created **permanent**.

The Global Note & documentation link

Above the note list, each entity also has a single **Global Note** and a **documentation link** — the home for the *one* canonical description and the *one* runbook or wiki URL for that entity. Unlike the timestamped notes below it, the Global Note is a single editable field, and the documentation link renders as a clickable link. (This is the evolution of the old "Documentation Notes & Links" / identity-card feature; an entity-specific value overrides the tenant-wide default.) Programmatically it is served by the `splk_identity_cards` resource group — reads require the `trackmeuseroperations` capability and `splk_identity_cards/write` requires `trackmepoweroperations`.

Where notes appear — and where they don't

- Notes live **on the entity**: open them from the entity, or from the **Notes** action in the entities table, which also shows a **note count** per entity.



- Notes are **searchable by substring** inside the panel, but — unlike *labels* and *tags* — they are **not** a filter axis or a group-by. Use notes for *context a human reads*; use labels and tags for *structured filtering*.
- Notes are **not** included in alerts or notable events, and they **never affect an entity's score or state**. (An acknowledgment carries its own separate comment — that is not a note.)

Tip

Notes vs the entity description. A note is a timestamped, authored, append-only log entry. If you instead want a *single, editable-in-place* description of the entity, edit its `object_description` field (or the Global Note above) rather than adding a note.

See also

- *Labels & auto-labels* and *Tags* — the structured annotation features.
- *Operations & Administration* — the REST API, including the notes endpoints.

8.6 Alerting

The decision maker computes an entity's *current* state. Alerting turns that into notifications, investigations, and traceable incident records — without drowning you in spam. This part covers the two alerting paths and the acknowledgment system that keeps them quiet.

- **Stateful alerting** — the modern path: one incident record per problem that opens, updates, and closes, driving email, ingest, and command channels.
- **Notable events** — the traditional path: normalised TrackMe notable events for correlation, third-party integration, and custom workflows.
- **Acknowledgments** — silence a known problem without hiding it.

A companion *in-depth reference* collects the full alerting detail — the stateful-alert wizard, active commands, incident-event anatomy, worked examples, and the third-party integration architectures.

Note

The two paths are **independent and composable** — a single alert can raise a stateful email incident *and* a notable event at the same time.

8.6.1 Stateful alerting

Stateful alerting is TrackMe's modern alert path. Instead of re-firing every cycle an entity is in trouble, it keeps **one incident record** per problem: it opens when the entity first goes into alert, updates while the problem persists, and closes when the entity recovers. The result is rich, context-aware notifications without the spam.

Note

This page is the **overview**. For the full reference — the stateful-alert wizard step by step, embedded charts, the AI status report, active commands, the anatomy of stateful incident events, worked examples per component, and the third-party integration architectures — see *Alerting & third-party integration — in depth*.

As an entity moves from a non-alerting state into alert and back, TrackMe maintains a single persistent **incident record** in KV-store, emits lifecycle events to its sourcetypes (`trackme:state`, `trackme:flip`, `trackme:notable`, `trackme:sla_breaches`), and drives the opened / updated / closed notifications and active commands from it.

Here is what a pair of those notifications looks like — the **opened** email when the entity first goes red, and the threaded **closed** email when it recovers, each carrying the entity's status, anomaly reason, and the embedded 24-hour incident, flipping and performance charts:

TrackMe Notification

Environment: https://prd5-cl.trackme-limited.com
Detection Time: Thu Jun 26 17:13:35 2025 UTC
Alert Status: opened
Tenant: secops
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Priority: critical
Drilldown Link

Status: red
Incident ID: 998bcb4f8cf2b9a1

Detailed Information:

The entity has transitioned to an alerting state:

- **Alias:** eventgen-firewall:netscreen:firewall
- **Object:** eventgen-firewall:netscreen:firewall
- **Object ID:** 822f63337de7e8a4ebd82d477d64c3053c4042d6d08a6ebb32c24e333d3fe3f2
- **Category:** splk-dsm
- **Anomaly Reason(s):**
  - delay_threshold_breached
- **Message(s):**
  - Monitoring conditions are not met due to delay issues. Event delay is 1551.0 seconds (duration: 00:25:51), which is higher than the maximum allowed delay of 900 seconds (duration: 00:15:00), latest event available (_time) for this entity: 26 Jun 2025 16:47, latest event ingested (_indextime) for this entity: 26 Jun 2025 16:47. This incidates that the source is receiving events with timestamps older than the threshold defined for this entity.
- **Message source:** trackme:state
- **Message source ID:** 1575326d50723a5a5832ee3b0179e5c5d88e6e598c120dc7d76b205817bccee5

Incidents, Flipping & State Charts (24h)



Performance Charts (24h)



This is an automated email, please do not reply directly to this email.

✅ TrackMe Notification

Environment: https://prd5-cl.trackme-limited.com
Detection Time: Thu Jun 26 18:43:19 2025 UTC
Alert Status: closed
Tenant: secops
Alias: eventgen-firewall:netscreen:firewall
Object: eventgen-firewall:netscreen:firewall
Priority: critical
Drilldown Link

Status: green
Incident ID: 998bcb4f8cf2b9a1

📋 Detailed Information:

✅ The entity has received an incident closure update and is now in a non-alerting state:

- **Alias:** eventgen-firewall:netscreen:firewall
- **Object:** eventgen-firewall:netscreen:firewall
- **Object ID:** 822f63337de7e8a4ebd82d477d64c3053c4042d6d08a6ebb32c24e333d3fe3f2
- **Category:** splk-dsm
- **Anomaly Reason(s):**
  - delay_threshold_breached
- **Message(s):**
  - 26/06/2025 18:43:19, object="eventgen-firewall:netscreen:firewall" has flipped from previous_state="red" to state="green" with anomaly_reason="none", previous_anomaly_reason="ml_outliers_detection", disruption_time="5705.75"
  - monitoring conditions for event delay are met. Event delay is -8.0 seconds (duration: -00:00:08), which is lower than the maximum allowed delay of 900 seconds (duration: 00:15:00), latest event available (_time) for this entity: 26 Jun 2025 18:43
  - monitoring conditions for ingest latency are met. Ingestion latency is approximately 2.439 seconds (duration: 00:00:02), which is lower than the maximum allowed latency of 900 seconds (duration: 00:15:00), latest event indexed (_indextime) for this entity: 26 Jun 2025 18:43
- **Message source:** trackme:flip
- **Message source ID:** b002a5a25c76ad3f26600f25ea83770b84072c3c6491c2c5aadd9967182f3d98

📈 Incidents, Flipping & State Charts (24h)



📈 Performance Charts (24h)





The incident lifecycle

Transition	What happens
Opened	The entity enters an alerting state (red — or orange, if you opt in). An *Alert Opened* notification is sent.
Ongoing	The entity is still in alert on a later cycle. The **same** record is updated and an optional *Update* notification is sent — no new incident is created.
Closed	The entity returns to a non-alerting state. An *Alert Closed* notification is sent.

Because the same rule on the same entity always updates the same record, email notifications **thread** into a single conversation in your mail client.

Note

Alerting *consumes* state — it does not compute it. The lifecycle is driven entirely by the entity's state and score (see *The entity state machine*). Acknowledgment suppresses notifications but never changes the underlying state.

Delivery channels

One alert can drive any combination of:

- **Email** — themed HTML notifications, threaded, with embedded 24-hour charts and an optional AI status report (see below).
- **Ingest** — a JSON event written to the TrackMe summary index, for ES correlation, historical analysis, or forwarding to a SIEM.
- **Commands** — a saved search dispatched on open / update / close, with the alert context injected as tokens — ideal for ticketing, paging, and custom automation.

The AI status report

When enabled, TrackMe builds the entity's full context — state, anomaly reason, priority, tags, SLA, recent ack and disruption history, and any visible Guardian alerts — and asks your configured AI provider for a short, 2–3 paragraph investigation summary, embedded in the email.

Important

The AI status report is **fail-open**: if no provider is configured, or it errors or times out, the email still sends without the AI section. Alerting is never blocked by AI. See the *Artificial Intelligence* section to configure a provider.

Settings you will reach for

Setting	Effect
orange_as_alerting_state	When on, orange states also open incidents (off by default). Any non-alerting state still closes the incident.
email_send_update_if_ack_active	When off (the default), acknowledged entities receive no update emails — the record still updates internally.
generate_charts / theme_charts	Embed the 24-hour charts, and choose dark or light.
ai_status_report / ai_provider_name	Enable the AI summary and optionally override the provider.
auto-ack on open	Automatically acknowledge an entity when an incident opens (see *Acknowledgments*).

Note

Embedded email charts require Python 3.9+ and are **not available on Splunk 9.1 / 9.2**. If charts cannot render, the email still sends (fail-open) with a placeholder. With `orange_as_alerting_state` on, an entity that oscillates green ↔ orange creates many short incidents — keep it off, or widen the *disruption grace*.

See also

- *Alerting & third-party integration — in depth* — the full reference: the stateful-alert wizard step by step, active commands, the anatomy of stateful incident events, worked examples per component, and the third-party integration architectures.
- *Notable events* — the notable-events / correlation path.
- *Acknowledgments* — quietening known problems.
- *The entity state machine* — the states that drive the lifecycle.

8.6.2 Notable events

The notable-events path generates TrackMe **notable events** from an entity's state — normalised JSON events written to a dedicated index whenever an alert fires. A *notable* here is TrackMe's own concept: a consistent, machine-readable record of *what fired and why*, ready for correlation searches, third-party integration, or any custom incident workflow. (Being ordinary Splunk events, they can be consumed by anything that reads the index — but they are not tied to any particular app.)

Notables are also the backbone of the **legacy third-party integration** design. Two architectures are possible: **scenario 1** — TrackMe alerts emit notable events into a notable index, and a single centralised alert watches that index and dispatches to Jira, ServiceNow, Slack, email, and so on; or **scenario 2** — each TrackMe alert runs the third-party action directly. Scenario 1 is preferred — one place to decide what becomes an incident, normalised events, and acknowledgment handled consistently.

How it works

A notable is generated **only when an alert fires** for an entity — **one event per fire**. There is no open/update/close lifecycle: each notable is an instant snapshot of the entity at the moment it fired, written as a JSON event to the tenant's **notable index** (default `trackme_notable`).

Each notable carries at least: a rule name, a severity / urgency (`low` / `medium` / `high` / `critical`), the entity identity (`object_id`, `tenant_id`, `component`), its `object_state`, `total_score`, and `anomaly_reason`, its priority / tags / SLA, timestamps, and a **drilldown URL** back to the entity. You can review them in the entity's **Notables** tab, or query them with the tenant notable macro, for example:

```
`trackme_notable_idx(mytenant)` tenant_id="mytenant" object="myobject"
```

Notables are generated by the `trackme_notable` **alert action**, which writes the JSON event to the tenant's notable index (default `trackme_notable`).

Tip

You can also generate notables directly from an ad-hoc search with the `trackmegennotable` command — useful for one-off correlation work outside the standard alert actions.

Notables vs stateful alerting

	Notable events	*Stateful alerting*
Lifecycle	One event per fire (no state machine)	Open / update / close, deduplicated
Channels	Index ingest only	Email, ingest, commands
Consumer	Correlation searches, third-party integration, custom workflows	Email-first teams, automation
Acknowledgment	Ignored	Honoured
Threading	No	Yes

Both can run on the **same** alert, so you can emit notables for correlation or integration *and* email a small team at once.

Note

- Notables are written to the tenant's **notable index** (default `trackme_notable`); the target index is set on the alert action.
- There is no de-duplication at the TrackMe layer: a flapping entity firing every cycle floods the index. Use a less frequent schedule or stateful alerting.
- **Notables ignore acknowledgment** — acking an entity does not suppress its notables.
- If you have **no** notable consumer, prefer stateful alerting; unread notables just pile up.

See also

- *Stateful alerting* — the modern, ack-aware path.
- *Acknowledgments* — why ack does not affect notables.
- *Operations & Administration* — index configuration and the REST API.

8.6.3 Acknowledgments

An **acknowledgment** lets you silence a problem you already know about — without hiding it. An acknowledged entity keeps its true state and score and is still monitored; stateful alerting simply stops re-notifying you. The key idea: **ack suppresses notifications, not state.**



The **Manage TrackMe Acknowledgments** dialog is the central place to review and set acks: the table lists each acknowledged entity with its type, expiration, source, captured anomaly reason and comment, and the lower panel creates or updates an ack — pick a **duration**, a **type** (sticky / unsticky), and add an optional **note** that is logged for auditing.

Sticky vs unsticky

Kind	Behaviour
sticky	Never auto-expires — it stays until you clear it (or it is revoked by a reason change). Means "known and accepted; don't tell me again."
unsticky	Auto-expires after `ack_period` seconds (default 24 hours). Means "I see it; give me time to investigate."

An ack records who set it, a comment, the kind, and its expiration. Re-acking an already acknowledged entity **extends** it.

Creating an ack

- **In the UI** — the entity's *Ack* action; set (and later extend) the expiration, add a comment, and remove it at any time. Acknowledged entities show an "eyes" icon next to their status.
- **Automatically** — the auto-ack alert action can acknowledge an entity whenever an incident opens, with a configurable period and kind (see *Stateful alerting*).
- **Programmatically** — through the acknowledgment REST endpoints.

A background sweep (every ~5 minutes) clears expired **unsticky** acks; sticky acks are never swept.

Reason-change revocation

By default, TrackMe revokes an ack automatically if the entity's **anomaly reason changes**. Each cycle it compares the current reason against the reason captured when you acknowledged; if they differ, the ack is revoked and a fresh alert can be raised. This prevents *silent escalation* — an ack for a delay problem will not quietly mute a new, unrelated latency problem on the same entity. This behaviour was introduced in TrackMe 2.0.97 and is tunable under **Configure → General** with three options: *Expire Ack on anomaly reason change behavior*, *…min time since*, and *…only for auto ack*. Ack activity is **audited** and reviewable in the entity's **Audit** tab.

What ack does and doesn't do

Ack suppresses	Ack does **not** change
Stateful-alert update emails	The entity's true state and score
(the record still updates internally)	Monitoring and visibility in the UI
	The disruption queue / grace behaviour
	Notable events — notables ignore ack entirely

Note

If acknowledged entities are still emailing you, set `email_send_update_if_ack_active` to off. If an ack expired sooner than expected, it was probably **unsticky** (24-hour default). To silence *everything* for an entity regardless of reason changes, disable reason-change revocation at the tenant level — but that is unusual.

See also

- *Stateful alerting* — where ack gates update emails.
- *Disruption queue* — ack is independent of the disruption grace.
- *Notable events* — why notables are not suppressed by ack.

8.6.4 Alerting & third-party integration — in depth

Tip

This is the **in-depth reference** for alerting. For a quick, readable introduction, start with *Stateful alerting*, *Notable events*, and *Acknowledgments*. This page collects the full detail: the stateful-alert wizard step by step, embedded charts, the AI status report, active commands, the anatomy of stateful incident events, worked examples per component, and the legacy notable-based and per-alert third-party integration architectures.

Introduction to Stateful Alerting in TrackMe

Hint

TrackMe Stateful alerting is TrackMe's core alerting model: **state-aware alerts** that follow the full life cycle of an incident, with advanced **email thread notifications** and **active command** execution.

- TrackMe automatically maintains cycles of incidents in a persistent, state-aware manner.
- It delivers rich **HTML email notifications** with automated, embedded metrics charts, and can include an **AI-generated entity status report** so recipients get immediate insight into an entity's health without opening the Splunk UI.
- It generates **stateful events** to trigger advanced third-party integrations — on incident creation, on updates, and on closure when the entity returns to a normal state.
- It can **execute active commands** (generating or streaming) depending on whether the incident is opened, updated or closed — for example, to create, update and close incidents in an ITSM.
- It can automatically **acknowledge** entities when an incident opens, and lets you select which **priority levels** generate emails and commands, at the handler level.
- The charts **time range window** is configurable per alert (default 24 hours).
- Together, these provide a complete, powerful and flexible state-aware alerting architecture.

- **Email charts are not available in Splunk 9.1 / 9.2 environments** (they require Python 3.8 or later).

Sophisticated Alerting Made Easy - TrackMe Alerting in a Nutshell

In short:

- Once enabled, when a given entity turns into an alerting state, a new incident is created.
- When new events are detected, such as Notable events or SLA breaches, the incident is updated.
- Eventually the entity returns to non alerting state, the incident is closed.
- Rich HTML emails notifications can be delivered by TrackMe, including automated and embedded metrics charts.
- When AI providers are configured, an **AI-generated status report** can be included in email notifications, providing contextual analysis adapted to the incident lifecycle (opened, updated, or closed).
- Events are also generated, in the `sourcetype=trackme:stateful_alerts`.
- Therefore, TrackMe can generate **Opening / Updating / Closing events**, and/or **rich Emails notifications**.
- **Active commands (generating or streaming commands)** can also be executed, depending on the incident status.

Creating a Stateful Alert in a Few Easy Steps

In the TrackMe Virtual Tenant, open the "Tracking Alerts" tab. From the Actions menu (the ⍰ button, top right), click "Create a new alert":



The "Alert tracking definition" wizard opens in assisted mode. Choose the type of alert to track — here `TRACKME STATEFUL ALERTS` (the other types are Notable events, splk-DSM data sources, and SLA breaches) — then work through the guided steps:



Step 1: Select the Stateful Alert Mode

The first step is to select the alert mode, available options are:

- `Emails and Ingest`: generate both emails deliveries and ingest events
- `Emails, Commands and Ingest`: generate emails deliveries, active commands and ingest events
- `Commands and Ingest`: generate active commands and ingest events
- `Commands and Emails`: generate active commands and emails deliveries
- `Ingest Only`: generate ingest events only
- `Emails Only` (not recommended): generate emails deliveries only
- `Commands Only`: generate active commands only

Ingest actions

- We recommend to always have the ingest actions enabled, unless you have a specific use case with multiple alerts.

Select the delivery mode from the `Mode` dropdown (in the wizard's "Stateful main features" step):



Step 2: Configure emails delivery, if wanted

When the selected mode includes emails, configure the delivery under `Email Configuration` — choose the delivery account (the default localhost MTA, or an SMTP account you configured), an optional environment name added to the email header, the list of recipients (type an email and press Enter to add each one), whether to keep sending update emails while an entity is acknowledged, and optionally the priority levels that trigger emails (leave empty for all):



For Splunk Cloud environments:

- You cannot configure an external SMTP destination in Splunk Cloud; the local MTA is mandatory.
- Therefore, for Splunk Cloud, you do not need any further configuration; TrackMe uses the local MTA as a default.

For Splunk Enterprise environments:

- You can choose to use the local MTA or configure an external SMTP destination.
- You can configure an external SMTP destination in Splunk Enterprise, and TrackMe will use it to deliver emails.

External SMTP destinations are managed from **Configuration → Manage Email delivery accounts.** The page lists every account with its SMTP server, security mode and last connectivity status, and offers per-row **Edit / Clone / Test connectivity / Delete** actions plus **Test all** and **Add account**. When no account exists yet, the page shows a short getting-started guide:



Click Add account, fill in the SMTP connection details and use Test SMTP connection to validate before saving. SSL or TLS is recommended, but not mandatory — an external SMTP server with no security is also supported:



Once saved, the account appears in the list with its live connectivity status, and can be selected in the stateful-alert email configuration:



Step 3: Charts generation

About Emails clients compatibility

- Emails charts use SVG images format, some limited email clients such as Web Gmail do not support this format and charts are not displayed.

In the `Charts Generation` section, choose whether to generate charts, the charts theme (light or dark) and the charts time window. By default, TrackMe generates dynamic charts using super fast TrackMe metrics queries and indexes, and embeds them in the emails notifications:



What does this look like? Very cool!:



TrackMe also stores the charts in base64 format in the persistent KVstore collection, so you can access them later on if needed notably for third party integrations.

Replace mytenant with the name of your Virtual Tenant:

```
| inputlookup trackme_stateful_alerting_charts_tenant_mytenant | eval key=_key, _
↪time=mtime
```

You could link stateful alerts events to the charts in the KVstore collection, in a Splunk search:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id=mytenant
| lookup trackme_stateful_alerting_charts_tenant_mytenant incident_id OUTPUT␣
↪chart_svg_base64
```

Step 4: AI Status Report in emails

About the AI Status Report

- The AI Status Report feature requires at least one AI provider to be configured in TrackMe. If no AI provider is available, this section will not appear in the wizard.
- For more information about configuring AI providers, consult the AI Assistant documentation.

When AI providers are configured in TrackMe, the stateful alert wizard offers the option to include an AI-generated entity status report in email notifications.

The AI Status Report is a concise, actionable summary generated by the configured AI provider (such as OpenAI,

Azure OpenAI, or Claude) that analyses the entity context and health status. It is embedded directly in the HTML email notification, between the detailed information section and the charts section.

Key features:

- **Context-aware analysis**: The AI prompt adapts to the incident lifecycle stage — it provides different insights depending on whether the incident is being **opened** (what went wrong and where to investigate), **updated** (whether the situation has changed), or **closed** (confirmation of recovery and what was previously wrong).
- **Automatic formatting**: The AI response is automatically converted to email-safe HTML, ensuring clean rendering in all email clients.
- **Fail-open behaviour**: If the AI provider is unavailable or returns an error, the email notification is delivered normally without the AI section — no email is ever blocked by an AI failure.
- **Cost-aware**: The AI report is only generated when email delivery is actually required. If the entity's priority level is filtered out from email notifications, no AI call is made.

Configuration options:

- `AI Status Report`: Enable or disable the AI status report in email notifications. When at least one AI provider is configured, this defaults to **enabled**.
- `AI Provider`: When multiple AI providers are configured, you can select which provider to use. If only one provider is configured or no specific selection is made, the first available provider is used automatically.





Example of an email notification with the AI Status Report:





Step 5: Active commands

In TrackMe, active commands run a custom SPL command depending on the incident status. Choose the `Commands mode` (Streaming or Generating), optionally restrict execution to specific priority levels, and provide the SPL command for each state — opened, updated and closed (each with an `Edit` / `SPL Preview` toggle):



Streaming commands example: Creating / Updating / Closing incidents in ServiceNow

This implementation example uses the Splunk TA for ServiceNow application and its built-in command to create, update, and close incidents.

For more details about the command usage, consult the ServiceNow TA documentation.

When creating the TrackMe alert, set up the active commands for each state of the alert (opened, updated, closed).

These are examples of commands that need to be adapted to your ServiceNow context:

commands_created:

```
| eval account="snow_account_name" | eval contact_type="email" | eval␣
↪ci_identifier=object | eval priority="1" | eval category="Software" | eval␣
↪subcategory="TrackMe" | eval custom_fields=
↪"u_affected_user=nobody||u_caller_id=12345" | eval short_description="The␣
↪TrackMe entity " . alias . " has transitioned to an alerting state: " .␣
↪object_state . ", detection_time: " . detection_time . ", messages: " .␣
↪mvjoin(messages, "|") | snowincidentstream
```

commands_updated:

```
| eval account="snow_account_name" | eval contact_type="email" | eval␣
↪ci_identifier=object | eval state="2" | eval category="Software" | eval␣
↪subcategory="TrackMe" | eval custom_fields=
↪"u_affected_user=nobody||u_caller_id=12345" | eval short_description="The␣
↪TrackMe entity " . alias . " The entity has received an update: " . object_state␣
↪. ", detection_time: " . detection_time . ", messages: " . mvjoin(messages, "|")␣
↪| snowincidentstream
```

commands_closed:

```
| eval account="snow_account_name" | eval contact_type="email" | eval␣
↪ci_identifier=object | eval state="7" | eval category="Software" | eval␣
↪subcategory="TrackMe" | eval custom_fields=
↪"u_affected_user=nobody||u_caller_id=12345" | eval short_description="The␣
↪TrackMe entity " . alias . " has received an incident closure update and is now␣
↪in a non-alerting state: " . object_state . ", detection_time: " .␣
↪detection_time . ", messages: " . mvjoin(messages, "|") | snowincidentstream
```

You can call tokens at any stage of the command, using the format:

- `field_name="$result.field_name$"`

Or:

- `field_name="$field_name$"`

Tokens are applicable against any field resulting from the alert, as well as fields added by TrackMe, you can consult an example of an incident event in the `Incidents` tab, see the anatomy of StateFul Incidents Events section for more details.

Generating commands example

TrackMe can also call a generating command instead, which you can leverage to tackle your specific use case.

The following example shows a simple usage for demonstration purposes using the Splunk `makeresults` command, where we would add a subset of the content to a lookup file, which could then be consumed by a third-party logic.

commands_created:

```
| makeresults | eval request_type="new", ci="$object$", ci_id="$object_id$",␣
↪alert_status="$alert_status$", event_id="$event_id$",description="The TrackMe␣
↪entity $alias$ has transitioned to an alerting state: $object_state$,␣
↪detection_time: $detection_time$" | outputlookup append=t trackme_incidents.csv
```

commands_updated:

```
| makeresults | eval request_type="update", ci="$object$", ci_id="$object_id$",␣
↪alert_status="$alert_status$", event_id="$event_id$", description="The TrackMe␣
↪entity $alias$ has received an update: $object_state$, detection_time:
↪$detection_time$" | outputlookup append=t trackme_incidents.csv
```

commands_closed:

```
| makeresults | eval request_type="close", ci="$object$", ci_id="$object_id$",␣
↪alert_status="$alert_status$", event_id="$event_id$", description="The TrackMe␣
↪entity $alias$ has received an incident closure update and is now in a␣
↪non-alerting state: $object_state$, detection_time: $detection_time$" |␣
↪outputlookup append=t trackme_incidents.csv
```

Troubleshooting active commands execution

Hint

About the user context

- It is important to remember that the command is executed in the context of the owner of the alert.
- When it comes to TrackMe, the owner of the alert is handled automatically by TrackMe to correspond to the owner of the Virtual Tenant.
- By default, the owner when creating Virtual Tenants in TrackMe is nobody. (splunk-system-user)
- Therefore, keep this in mind while troubleshooting the command execution, as a third party command might potentially require specific permissions and capabilities to be available for the user context.

Consult the logs of the Stateful alert action to review the execution of the active commands:

```
index=cim_modactions sourcetype=modular_alerts:trackme_stateful_alert "Command"
```

Especially, the command execution will be logged as:

```
index=cim_modactions sourcetype=modular_alerts:trackme_stateful_alert "Executing␣
↪command"
```

If the active command is executed, but does not return any results, the logs will show an error message, otherwise an informational message will be shown.

TrackMe processes active commands by converting the upstream results in a base64 encoded string, then executing the trackMe command trackmeyieldjson followed by the requested search logic.

This is a very powerful and flexible approach, you can easily re-execute the command at anytime with 100% of the original context and easily troubleshoot any issue:

```
| trackmeyieldjson json_value_b64="<encoded_base64_string"
```

Step 6: Various options

Hint

About the cron schedule: the cron schedule is not exposed as a configurable option, on purpose, because stateful alerts are meant to run on a frequent basis, ideally every 5 minutes:

- The stateful handlers tracks changes in the entities from different angles, and then decides if the notification means the beginning of an incident, an update or a closure.

- For this to work as expected, the alert must be execute as soon as possible and it then handles the incident life cycle.
- Therefore, it is **not recommeded to change the default cron schedule which should be set to every 5 minutes.**

Finally, there are various options that you can configure for the alert, such as:

- `Consider orange as an alerting state`: In TrackMe, orange is typically used to indicate a warning state; you can configure TrackMe to consider this as an alerting state.
- `Drilldown URI`: The drilldown URI is used to build the drilldown link in the email notifications.
- `Priority`: The priority of entities for triggering the alert; you may, for instance, want to trigger the alert only on high or critical priority entities.

In the current interface these options appear within the relevant blocks of the wizard rather than in a single "common options" screen — the priority filters sit inside the email and active-command blocks, and the remaining options under the wizard's *Various* step.

Additional information

TrackMe implements various dynamic actions, depending on the circumstances or the type of entities it is dealing with, to adapt automatically the behavior of Stateful alerts.

- **Time detection, deduping and backfilling:** TrackMe automatically identifies if events should be skipped according to the entity cycle, this means that it will skip events if necessary and will never send duplicated alerts, whatever the triggers of the alerts are. (so you can have an overlapped search approach safely and TrackMe does the hard work!)
- **Dynamic charting:** TrackMe automatically chooses which metrics should be charted, and how, based on the entity type and metrics available.
- For instance, for the splk-dsm component, it will automatically add component specific charts such as data sampling charts. (quality issues)
- For components like splk-flx and splk-wlk, TrackMe will automatically verify metrics availability for the entity, and will generate charts accordingly.
- For the splk-vol component (Volume Outliers), the notification embeds the rolling volume charts (24h / 12h / 4h / 60m over the alert's chart window), the ML outliers chart of each model and a daily licensed volume bar chart over the last 30 license days.
- **Status flexibility:** TrackMe has different statuses, you can influence for instance if orange should be considered as an alerting state or not. On the other hand, "blue" which is a special status (notably for logical grouping) is automatically considered as a non alerting state, eligible for incident closures.

Anatomy of Stateful Incidents Events

In addition with rich email notifications, TrackMe also generates incident events, which can easily be used to trigger third party integrations.

- alert_status: `opened` - This is the start of the incident
- alert_status: `updated` - This is an update of the incident
- alert_status: `closed` - This is the closure of the incident

Example of an opening incident event:

```
index=trackme_summary sourcetype="trackme:stateful_alerts" alert_status=opened␣
↪tenant_id=* object_category=* object=*
```



Example of an updating incident event:

```
index=trackme_summary sourcetype="trackme:stateful_alerts" alert_status=updated␣
↪tenant_id=* object_category=* object=*
```



Example of a closing incident event:

```
index=trackme_summary sourcetype="trackme:stateful_alerts" alert_status=closed␣
↪tenant_id=* object_category=* object=*
```



In TrackMe, incidents events can be accessed through the UI and the Incidents tab:



TrackMe Stateful alerts events are `JSON` formatted events, these contain the following key information:

Field	Description
alert_status	The status of the incident, one of `opened`, `updated` or `closed`.
alias	The entity alias, if defined.
cmdb	The matched CMDB record(s) for the entity — a single object, or a list of objects — from the tenant's configured CMDB lookup. Present only for ingest delivery and when a CMDB match is found.
ctime	The creation time of the incident (epoch)
delivery_type	The delivery mode(s) called for this incident, such as `email`, `ingest` and/or `commands`.
detection_time	The time of the detection of the alert in human readable format (strftime %c %Z)
drilldown_link	A link to the TrackMe Home user interface which filters on the tenant, component and entity, and automatically opens the entity overview screen.
event_id	The unique identifier for this event, the sha-256 hash of the event content.
incident_id	The unique identifier for this incident — a stable, shorter identifier common to every event and object related to the same incident (from `opened` through `updated` to `closed`).
labels	The entity's labels, as a list (present only when the entity has labels).
message_id	A unique identifier for this message, principally used for emails.
message_source	The sourcetype at the origin of the incident, such as `trackme:state` or `trackme:flip`.
message_source_i	The unique identifier (`event_id`) of the original event that triggered the incident (also used for de-duplication).
messages	The list of messages from TrackMe that occur during the incident life cycle, this related to the TrackMe status_message field
mtime	The last modification time of the incident (epoch)
object	The object name for this entity, this basically is the name of entities in TrackMe
object_category	The TrackMe component this incident relates to (splk-dsm, splk-dhm, splk-mhm, splk-vol, splk-flx, splk-fqm, splk-wlk, splk-cim).
object_id	The unique identifier for this object in the tenant / component central KVstore collection.
object_state	The current state of the object (blue / green / orange / red).
opened_anom	The anomaly reason captured when the incident was opened.
priority	The priority of the object (low / medium / high / critical).
reference_chain	The list of `message_id` values that relate to the same incident, used for email threading.
score	The entity's impact score (present when scoring applies to the entity).
score_definitic	The human-readable definition of the impact score (present alongside `score`).
tags	The entity's tags, as a list (present only when the entity has tags).
tenant_id	The tenant this incident is related to.
updated_anomal	The anomaly reason at the time of an update or closure (present on `updated` and `closed` events).

Example: Feed tracking entity (splk-dsm)

In this example, a sourcetype is affected by high latency and/or delay, a first notification is generated:

A first notification is generated, either through a notable event of a flip event:



An opening incident event is created:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="mytenant"␣
↪object="myobject" alert_status="opened"
```



Eventually, an update will be issued if a new event is generated:



An update incident event is created:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="mytenant"␣
↪object="myobject" alert_status="updated"
```



Later on, the SLA is breached, a new notification is generated:



Another update incident event is created:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="mytenant"␣
↪object="myobject" alert_status="updated"
```



Finally, the issue is resolved, an incident closure notification is generated:



An incident closure event is created:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="mytenant"␣
↪object="myobject" alert_status="closed"
```



Example: Splunk Cluster Monitoring with splk-flx

In this example, we monitor a Splunk cluster and Splunk indexers through the Flex object component.

Initial status: The cluster global status is healthy, all peers are up and running:





The cluster and its members can also be visualised as a topology — either as a hierarchical link graph or as a node-link network graph:





An incident affects an indexer, the global cluster status as well as the indexer turn into alerting state:





Drilling into the affected cluster entity shows the peer drop and the detailed cluster status message:





A first notification is generated for the cluster status. The rich HTML email, top to bottom: the incident header and detail, the AI status report with the 24-hour event charts, and the embedded performance charts:







The affected indexer peer raises its own notification too — the same rich format, scoped to that peer:







An opening incident event is created:

```
index=trackme_summary sourcetype=trackme:stateful_alerts tenant_id="cluster-mon"␣
↪object="infrastructure:splunk_cluster:idxc1" alert_status="opened"
```



Once the underlying issue is fixed and the peers rejoin the cluster, both entities return to a healthy (green) state and TrackMe closes their incidents — each recovery raises a closure notification and writes a closure event.

The affected indexer peer recovers first: it is back to green, and its incident is closed. The closure event appears both in the entity's Incidents tab and in the summary index:





The peer's closure notification — the same rich, threaded email, now delivering the all-clear:







With the peers back in service, the cluster status returns to green too, and its own incident is closed:





And the cluster status raises its own closure notification:







Example of a notification due to Machine Learning Outliers detection:

Below is an example of a notification due to Machine Learning Outliers detection:

Note: TrackMe will automatically add a data sampling related chart in the notification, to help you visualise the issue.





Example of a notification due to a feed data quality issue (data sampling):

Below is an example of a notification due to a feed data quality issue (data sampling):

Note: TrackMe will automatically add a data sampling related chart in the notification, to help you visualise the issue.



Example of a notification for the Workload component (Splunk scheduled searches and alerts monitoring):

Below is an example of a notification for the Workload component (splk-wlk):

Note: For this component, TrackMe will automatically generate charts per metrics, depending on metrics available in the entity.





Legacy Alerting with Notables or direct alerts

Legacy Alerting is based on Notables or direct alerts:

Creating an alert in TrackMe

You can create a new alert in the scope of any Virtual Tenant:



Available options differ per TrackMe component. Creating a `splk-dsm` (data source) alert opens the wizard on the component-specific steps — start by naming the alert:



Step 2 is the entity filters. For legacy alerting architecture, it is recommended to keep these filters at their default values (ANY priority, Red state, all objects, all tags) so that Notable events and acknowledgements are generated for all entities. Apply filtering at the Notable alert level instead:



Step 3 is where the component alert earns its keep: its `Alert Actions` trigger the TrackMe Auto Acknowledgment and the TrackMe Notable event. Keeping `Notable` enabled (strongly advised) means every triggering entity produces a Notable event:



Hint

Volume Outliers (splk-vol). The wizard offers a `SPLK-VOL` alert type for a Volume Outliers tenant. Its entity filters keep **Trigger on Outliers** on by default — the ML outliers *are* the state of that component (a drop or a spike of the licensed volume of an index) — and the same alert also covers an index turning red for **inactivity**. `splk-vol` is selectable in the stateful email component filter and in the SLA breaches alert. See *Volume Outliers — in depth*.

Generating the Notable events is the whole point of the component alert: once written to the tenant's notable index, they can be consumed in several ways — by creating a **Notable alert** (see the next section) that filters and forwards them to third parties, by any home-made Splunk search (including cross-tenant searches over several notable indexes), or by an external system reading the index directly. This decoupling between *generating* the notables and *acting* on them is what makes the pattern so flexible.

TrackMe modular alert actions

TrackMe has several alert actions that are leveraged when creating alerts:

- **TrackMe Notable event:** This alert action generates a JSON event in the Virtual Tenant notable index summarizing the status of an entity when the alert triggers
- **TrackMe Auto Acknowledgment:** This alert action automatically acknowledges the entity in TrackMe's workflow for a given period of time

These actions are attached to the underlying Splunk scheduled search — in the native alert view you can see them listed alongside the trigger condition and the trigger history:



The TrackMe Notable event is a mandatory alert action, others can be disabled during the creation of the alert if you wish to do so.

Alerts Triggering

When the alert fires, the Tracking Alerts tab summarises the enabled and recently-triggered alerts — with per-alert triggering charts and, in the Alert details table, the exact modular actions each alert runs (here `add_to_triggered`, `trackme_auto_ack` and `trackme_notable`):



Opening an alert's `Actions` view drills into its trigger history and the alert actions executed over time — one series per action, confirming the auto-acknowledge and notable-event actions ran on each trigger:



Third Parties Alert Integration

Introduction to Third Parties Alerting Notification

Any TrackMe alert can be extended to send notifications to a third party, typically:

- Sending emails to a recipient, such as the owner of the entities/service, or TrackMe administrators and analysts
- Opening an incident to your ITSM such as **ServiceNow** or **Atlassian Jira**
- Creating a message in a **Slack** or **Microsoft Teams channel**

Note

Two main alerting architecture design can be implemented:

- **Architecture scenario 1: Notable based design** TrackMe alerts generate notable events, a centralised Splunk alert monitors for TrackMe notables and run alert actions for the third party
- **Architecture scenario 2: Per alert design** TrackMe alerts are configured to run the third party alert action and perform the associated action directly

While the two options are possible, the Architecture scenario 1 based on Notables has multiple advantages:

- Each TrackMe alert handles all entities, ensuring the creation of TrackMe notable events and other features such as the acknowledgment
- Allows centralising the definition of what should lead to the creation of an incident, sending an email or sharing channel message
- Notable events are normalised and provide all necessary information in a consistent fashion
- For instance, you can have a single “TrackMe forward notable” alert which restricts to `high` priority entities and any additional logic of your own, easily in SPL
- This logic can easily be maintained over time, without having to modify multiple alerts and provides a deeper and more consistent control

This documentation covers both approaches in the next sections.

Architecture scenario 1: TrackMe Notable Events

Introduction: Anatomy of TrackMe Notable Events

Hint

Bringing TrackMe Notable Events into Splunk SOAR

- If you are looking to bring TrackMe Notable Events into Splunk SOAR, the recommended approach is the built-in ingestion of the TrackMe App on SOAR — see *Ingesting TrackMe alert events into SOAR*.

TrackMe Notable Events creation

TrackMe notable events are generated when a TrackMe alert triggers against one or more entities, relying on the built-in `TrackMe Notable` modular alert action (see *the TrackMe modular alert actions* covered earlier — the `TrackMe Notable event` action is what writes these events).

Note

Notable events are ONLY created when a TrackMe alert is configured and triggers; in this scenario, all eligible entities should therefore be covered by an alert per tenant and component.

TrackMe Notable Events structure

TrackMe Notable events are `JSON` formatted events, these contain the following key information:

Field	Description
alias	The entity alias (friendly name), when defined.
anomaly_r	A normalised list of the reason(s) behind the entity's status.
cmdb	The matched CMDB record(s) for the entity — from the tenant's configured CMDB lookup. Present only when a CMDB match is found.
drill-down_link	A link to the TrackMe Home user interface which filters on the tenant, component and entity, and automatically opens the entity overview screen.
event_id	The unique identifier for this notable event — the sha-256 hash of the notable event content.
keyid	The unique identifier for this object in the tenant / component central KVstore collection.
labels	The entity's labels, as a list (present only when the entity has labels).
object	The object name for this entity, this basically is the name of entities in TrackMe.
object_categ	The TrackMe component this notable relates to (splk-dsm, splk-dhm, splk-mhm, splk-vol, splk-flx, splk-fqm, splk-wlk, splk-cim).
priority	The priority of the object (low / medium / high / critical).
properties	The complete TrackMe entity record as nested JSON, as it was when the alert triggered. It is also field-extracted automatically, so you can address `properties.*` fields directly.
score	The entity's impact score (present when scoring applies to the entity).
score_defi	The human-readable definition of the impact score (present alongside `score`).
state	The status of the object (blue / green / orange / red).
status_messa	A human readable detailed description of the reason why TrackMe generated an alert for this entity.
tags	The entity's tags, as a list (present only when the entity has tags).
tenant_id	The tenant this Notable event is related to.
timeStr	The time of the event in human readable format.

Therefore, TrackMe Notable events contain all necessary information in a consistent and normalized manner, to interact with any third party.

Example of a TrackMe Notable event in Splunk search — the JSON payload with its extracted fields, including the nested `properties` *record (which carries the whole entity record, so you can address* `properties.*` *fields directly):*



The same Notable events are also surfaced in TrackMe's user interface, from an entity's Incidents view:



Step 1: Create TrackMe alerts for tenants and components in used

The first step is to create a TrackMe alert for every tenant and component that is in use.

From each tenant's **TRACKING ALERTS** tab, use the wizard to create an alert per component, leaving the entity filters at their defaults — the exact flow is covered in *Creating an alert in TrackMe*. At this stage there is no need to filter on priority: filtering is applied later, when monitoring the Notable events. Keep the `TrackMe Notable event` action enabled so every triggering entity produces a Notable.

Step 2: Create the alert monitoring TrackMe notables

The second step is to create the alert that consumes the Notable events and drives the third-party interactions.

TrackMe ships a built-in `Notable events` alert type for exactly this. From the **TRACKING ALERTS** tab create a new alert, choose **Notable events**, and (optionally) restrict it to the priority levels that matter — leave the filter empty to trigger on any priority:



For advanced cases — cross-tenant searches, custom normalisation or enrichment — you can also drive this from a plain Splunk alert over the notable index. As a basis:

```
index=trackme_notable tenant_id=* priority=*
| table tenant_id, keyid, event_id, object, object_category, state, priority,␣
↪anomaly_reason, status_message, drilldown_link
```

For instance, a typical use case would be to create incidents only for high priority entities in alert:

```
index=trackme_notable tenant_id=* priority="high"
| table tenant_id, keyid, event_id, object, object_category, state, priority,␣
↪anomaly_reason, status_message, drilldown_link
```

Note

In TrackMe, you can define different notable indexes per tenant, if you did so you will want to adapt your search.

When building such a custom Splunk alert, a good baseline is:

- Cron every 5 minutes: */5 * * * *
- Earliest time: -15m
- Trigger for each result
- Throttle against the keyid for 60m

Either way, you then enable the third party alert action on that alert. For example, to create an incident in JIRA, the following shows example content that uses token replacements:

```
TrackMe alert for a Splunk high priority entity:

- tenant: $result.tenant_id$
- object: $result.object$
- object_category: $result.object_category$
- state: $result.state$
- priority: $result.priority$
- anomaly_reason: $result.anomaly_reason$
- status_message: $result.status_message$
- drilldown_link: $result.drilldown_link$
```

Hint

Re-assigning to a service account

- As a good practice, you should re-assign the created Splunk alert to the TrackMe service account user, or nobody
- You should avoid running alerts on behalf of normal users

Adding third parties actions:

Atlasian Jira example:

Note: this integration relies on the following Add-on for Jira: https://splunkbase.splunk.com/app/4958





Email example:

Note: Sending emails is a builtin alert action in Splunk





Slack example:

Note: this integration relies on the following Add-on for Slack: https://apps.splunk.com/app/2878





Optional: Notable events enrichment

When using Notable events as the central means for third party alerting, further logic can easily be implemented such as custom normalisation and enrichment.

With enrichment for instance, you may want to leverage your CMDB knowledge to add context information such as the owner or contact address email for the notification, there are plenty of options and designs possible.

TrackMe allows to lookup easily in your CMDB to provide context information handy in the user interface, you can leverage the same content from the central Notable alert, for more information about the CMDB Lookup integrator

see: *CMDB integration*

For instance, we could enrich our Notable event alert by joining a CMDB lookup on the entity's index and sourcetype — taken from the `properties.*` *fields:*

```
index=trackme_notable tenant_id=* priority="high"
| lookup my_cmdb index as properties.data_index, sourcetype as properties.
↪data_sourcetype OUTPUT
| table tenant_id, keyid, event_id, object, object_category, state, priority,␣
↪anomaly_reason, status_message, drilldown_link
```

The enrichment fields are then available alongside the notable fields, and can be used as tokens in the alert action. For example, to set a recipient email address with a fallback for unknown entities:

```
index=trackme_notable tenant_id=* priority="high"
| lookup my_cmdb index as properties.data_index, sourcetype as properties.
↪data_sourcetype OUTPUT
| eval contact=if(isnull(contact), "support@trackme-solutions.com", contact)
| table tenant_id, keyid, event_id, object, object_category, state, priority,␣
↪anomaly_reason, status_message, drilldown_link, contact
```

Architecture scenario 2: Per TrackMe Alert Design

Another design choice is to configure third party alert actions on a per TrackMe alert instead of relying on the Notable events.

Note

This integration can also be considered as a valid choice, eventually easier and more logical, however it does not present some of the advantages in comparison with a TrackMe Notable based design:

- Each alert needs to be configured individually, which can be more challenging to maintain over time
- Entities that are not covered by any TrackMe alert will not lead to TrackMe alert actions being executed (Notable, Ack), for instance if you restrict alerts using entities priority or tags and some entities are out of the scope
- Normalisation, enrichment or the introduction of custom SPL logic is decentralised opposed to the central alert based on Notables
- Finally, you may need more alerts to be created (for instance to cover entities by priority and have alert actions executed for all entities)

For the purposes of this documentation, the following scenario will be implemented:

- A first alert is created which targets `high` priority entities, when triggering a notification is sent using a Splunk alert action to open an incident in our ITSM tool (we will use Jira but this is applicable to any Splunk supported third party)
- A second alert is created which targets `medium` priority entities, this time an email will be sent when the alert triggers
- Entities set as `low` are not subject to a third party integration, however an alert should also be active for notable events and other TrackMe alert actions to be triggered

Step 1: Create a TrackMe alert for high priority entities

We start by creating an alert in TrackMe for high priority entities — using the splk-dsm (or relevant component) alert type and setting the priority filter to `high`, exactly as shown in *Creating an alert in TrackMe*. Repeat for the other priority levels so that all entities are covered by alerts — in this example, three alerts in total.

Step 2: Enable alert actions per alert

Next, update each TrackMe alert to add the third-party alert action. From the alert's actions menu, open the Splunk alert-edit screen and add the action you need — for example the Jira alert action — mapping the TrackMe fields to the action's parameters using token replacements. These steps are identical to architecture scenario 1 (Notable based); consult that section for the details.

Context body example:

```
TrackMe alert for a Splunk high priority entity:

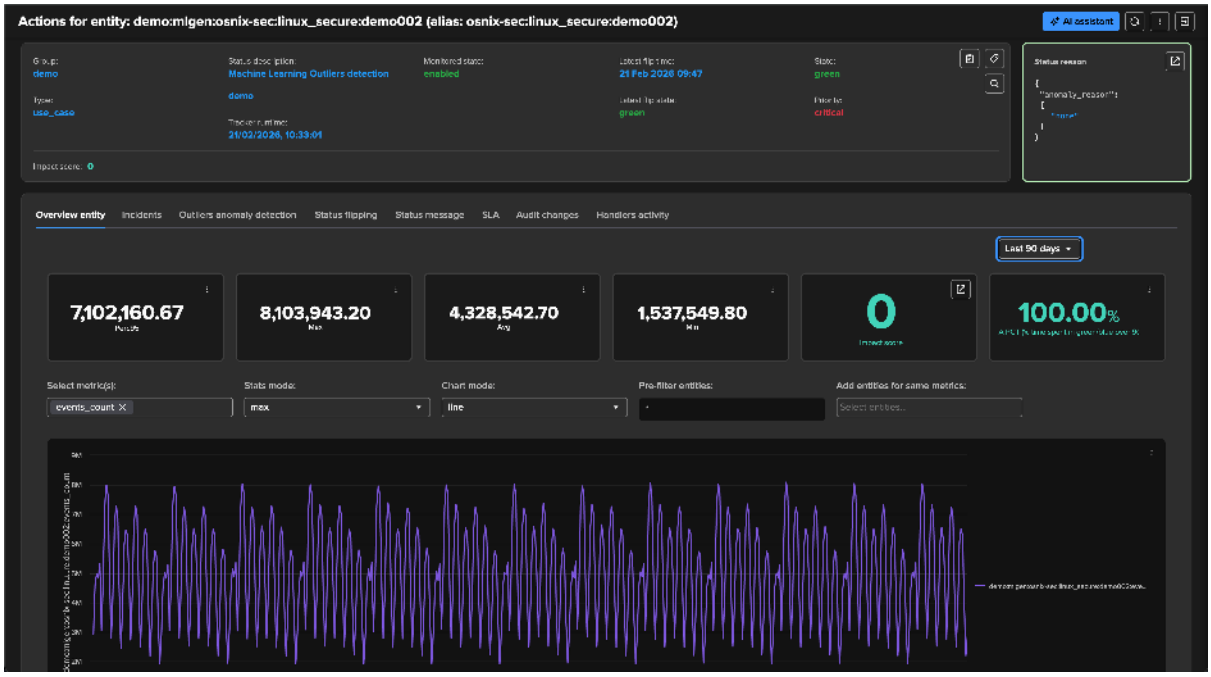
- tenant: $result.tenant_id$
- object: $result.object$
- object_category: $result.object_category$
- state: $result.state$
- priority: $result.priority$
- anomaly_reason: $result.anomaly_reason$
- status_message: $result.status_message$
- drilldown_link: $result.drilldown_link$
```

The same process needs to be achieved for every alert that should monitor TrackMe entities, per tenant and component.

When alerts trigger, Splunk automatically configured the alert actions according to your settings.

Deleting TrackMe alerts

Hint

We recommend deleting TrackMe alerts through TrackMe

- When creating a TrackMe alert, TrackMe preserves knowledge of it, which lets it automatically manage assignment and Role Based Access Control settings according to the Virtual Tenant setup.
- Deleting through TrackMe keeps that knowledge in sync. If an alert is removed directly in Splunk, TrackMe still verifies the existence of the alerts it created and updates its own knowledge accordingly.

Deleting an alert from the user interface

You can delete TrackMe alerts from the user interface — from the **TRACKING ALERTS** tab, open the alert's actions menu and choose **Remove alert**, then confirm. Because TrackMe maintains the knowledge of objects associated with a given Virtual Tenant, alerts should always be deleted through the user interface or the REST API.





Deleting an alert from the REST API

You can as well delete an alert using the REST API and the following endpoint:

```
| trackme mode=post url="/services/trackme/v2/alerting/admin/del_alert" body="{
↪'tenant_id':'mytenant', 'alert_name': 'TrackMe alert tenant_id:mytenant - Alert␣
↪custom on splk-dsm'}"
```

8.7 Topology Studio

Topology Studio is TrackMe's canvas for **your** view of the environment: build live topology maps from real TrackMe entities and from the objects of the findings tenants (Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) — place them, connect them, share them with your teams, and watch their

status update in real time. And because a map you cannot act on is just a picture, any topology can also **alert**: a state-aware engine correlates the health of everything the map covers and notifies you the moment the picture degrades.



- **Overview** — what Topology Studio is, its core concepts, and how it compares to the built-in topology views and Virtual Groups.
- **Creating dynamic views from Tenant Home** — generate a live canvas from any Virtual Tenant, choose a hierarchy, filter and inspect its members, then save it for continued authoring (TrackMe 2.4.16+). The findings tenants generate their own maps from their workspace — pools and endpoints, the data path, the fleet, Worker and Flow topologies — and save them to Studio the same way.
- **End-to-end example** — a complete worked example, screen by screen: a production Web platform monitored across its public-facing and internal-tooling sides, from the first aggregate to a real incident alerting and resolving.
- **Authoring topologies** — creating views, the canvas editor, node types, edges, visuals, and sharing.
- **Auto aggregates** — combine connected entities, aggregates and saved views into a live service-health parent, without duplicating scopes or filters (TrackMe 2.4.18+).
- **Decorations and canvas layout** — shapes, images and free text drawn directly on the canvas, and explicit canvas sizing for presentation-perfect views.
- **Topology Alerts** — smart, state-aware alerting and correlation on your maps.
- **Views browser and display modes** — the landing page: **Cards**, **Compact cards** and **List** layouts, search and category filtering, what is remembered per user, and the deployment-wide default administrators choose.
- **View integrity and consistency checks** — detect stale tenant/component references and alert configuration issues, then repair the affected views.
- **AI integration** — the AI Assistant understands your topologies, and can even build them for you.
- **Administration** — permissions, tenant isolation, licensing, export/import, and operational internals.

Note

Topology Studio was introduced in **TrackMe 2.4.9** as a major new product feature. It is available to **every edition** — Foundation, Enterprise and Unlimited, and active trials — including the complete authoring experience and Topology Alerts. See *Licensing* for the only gate, license read-only mode.

8.7.1 Overview

What Topology Studio is

TrackMe knows a great deal about your environment: every monitored entity, its health, its priority, its metrics. **Topology Studio** lets you arrange that knowledge the way *you* think about your environment — as a dependency map. A payment flow, a datacenter, an ingestion pipeline, a business service: you decide what belongs on the map and how it connects, and TrackMe keeps the map alive.



A **topology view** is:

- **Built from live health.** Health nodes represent monitored entities, the objects of the findings tenants (Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge), scoped aggregates, auto aggregates of connected nodes, or saved views rolled up as one node. Labels and decorations add context without health calculations. When an entity turns red, or an object has a critical finding, so does its node.
- **Authored or generated.** Place nodes, draw edges and choose the icons and layout yourself, or, since TrackMe 2.4.16, *generate a live preview from a Virtual Tenant* and save it as your starting point. A findings tenant generates its maps from its own workspace (HEC pools and endpoints, the S2S data path, the UF Fleet and Forwarder Collectors fleet maps, Cribl Worker, Process and Flow topologies) and saves them to Studio. Refine the map to reflect your understanding of dependencies.
- **Shared.** Views are stored centrally and shared under role-based access control, with separate read and write permissions — author once, consume everywhere, including full-screen on a wall of screens.
- **Alerting-capable.** Any view can carry *Topology Alerts* — a state-aware engine that correlates the health of the map and notifies on transitions, with the rendered topology image embedded in the notification email.
- **AI-integrated.** The *AI Assistant* understands your views and alerts in context — and can create or modify topologies for you through consented, audited actions.

Core concepts

Views

A view is one topology map: an identifier (`view_id`, lowercase, up to 40 characters), a display alias, an optional description and category (views sharing a category are grouped together in the list), the canvas content itself, and the role-based access lists. Views are global objects — they are not owned by a tenant, they *reference* entities across tenants.

Node types

Five kinds of items live on the canvas:

- **Entity nodes** — a single monitored entity, referenced by tenant, component and identity. The node carries the entity's live state color, and up to **five KPIs** displayed on the canvas (delay, latency, volume, skip rate… — or, for Flex Objects, FQM and WorkLoad, any of the entity's own metrics). An object of a findings tenant — a HEC endpoint, an S2S pool, a forwarding agent, a collector, a Cribl Destination, an Edge Node — is placed the same way: its node is red when one of its findings is critical, orange for another finding, blue when its findings are only snoozed, green when measured without a finding, and grey when not measured, with its KPIs and sparklines.
- **Scoped aggregate nodes** — a live roll-up of *many* entities: one or more tenant/component scopes plus an optional filter (the same filter grammar as Virtual Groups). The node folds to the **worst state** of everything it matches and shows per-state counts — or, by opt-in, classifies by a **healthy-percentage policy** with your own thresholds. Aggregates can also **auto-populate**: their current members appear as linked satellite nodes around them, following membership dynamically.
- **Auto aggregates** (TrackMe 2.4.18+) — health parents whose membership comes from incoming connections to entities, aggregates or view nodes. Children keep their scopes and policies; the parent follows their **worst effective state** and counts **unique underlying entities** for its KPI. No duplicated filters or independent parent policy are needed. See *Auto aggregates* for creation, label conversion and nesting.
- **View nodes** (TrackMe 2.4.15+) — a reference to another *saved* topology view, placed on this canvas as a single health node. The referenced view keeps ownership of its graph, its aggregate policies and its own alerts; the parent stores only the identifier, never a copy, so saving the child is reflected at the parent's next refresh. The node folds to the **worst effective state** of the child's health nodes — nested views and impact propagation included — while its KPI counts the **unique underlying entities** across the child's aggregates, direct entities and nested views (an orange child with 12 green and 4 orange entities reads *75% healthy, 16 entities* and stays orange). Open the referenced view in a new tab from the node's inspector: the parent canvas, viewport and unsaved edits stay put. Views inside views turn *a map* into *a map of maps* — an estate modelled as layered, independently owned views with one roll-up. A view node is a first-class alertable item, and it travels with history, duplication and export like any other node.
- **Label nodes** — free-text annotations with color control, for naming zones, tiers or flows on the map.

Live status

Open a view and it polls live status on a fixed cadence (with an honest countdown, a manual refresh, and a pause/resume toggle). Entity nodes colorize by `object_state`, and the nodes of findings-tenant objects by their findings; scoped aggregates fold their scopes server-side; auto aggregates combine their incoming contributors; view nodes resolve the referenced view server-side on every refresh, checking your read access to the view and to every tenant it reaches at each hop. Nodes whose underlying entity was deleted are flagged as **missing** (with a one-click cleanup); nodes referencing tenants outside your permissions render neutrally as **restricted** — see *tenant isolation*. A referenced view that is missing and one you are not allowed to read look the same, so a guessed identifier reveals nothing; and a reference whose health cannot be fully evaluated — a cycle, an incomplete child read, an exhausted nesting or size budget — never reports healthy. Auto aggregates likewise remain unavailable when their contributors are empty or incomplete, rather than showing a partial roll-up as healthy.

How it compares

Feature	Purpose
Topology Studio	Live maps across tenants and components, authored directly or generated from Tenant Home. Saved views are persistent, shared under RBAC and alerting-capable.
Tenant Home Studio preview (2.4.16+)	A temporary live canvas generated from the current selection, with configurable grouping, filtering, inspection and layout. Save explicitly to keep it as a normal Studio view.
Findings workspace maps	The maps a findings tenant generates in its own workspace (HEC pools and endpoints, the S2S data path, the UF Fleet and Forwarder Collectors fleet maps, Cribl Worker, Process and Flow topologies), live with their findings. Saved to Studio on request.
Link graph and Network graph	The existing per-component relationship graphs, available as alternate Tenant Home topology tabs with their detailed breakdowns.
Virtual Groups	Read-only aggregation *cards* in the Virtual Tenants grid — counts across tenants, no canvas, no edges, no alerting.

Hint

Use the Tenant Home preview to discover structure and start a saved Studio view, the Link graph and Network graph for their component breakdowns, and Virtual Groups for quick cross-tenant counters.

Where to find it

Click **Topology Studio** in the TrackMe navigation bar.

You can also start from the **topology icon** in any Virtual Tenant's Tenant Home and *save the generated preview* into Studio, or save a map generated in a findings tenant's workspace.

The landing page — the *views browser* — lists every view your roles can see, with a text search and a category filter, in the layout you prefer: **Cards** (grouped by category), **Compact cards** (denser tiles) or **List** (a sortable table). Your choice is remembered per browser; administrators set the deployment-wide default. From there you create, open, duplicate, export or import views. On a fresh install there is nothing to list yet, so the page invites you to create your first view — or to **Import** one exported from another instance. See *Authoring topologies* to build the first one.

8.7.2 Creating dynamic views from Tenant Home

Starting with **TrackMe 2.4.16**, you can generate a live Topology Studio canvas directly from **any Virtual Tenant**, using the topology action in Tenant Home. A findings tenant (Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream, Cribl Edge) has no Tenant Home: it generates its maps from its own workspace and saves them to Studio, as described on its own page. Start with the current tenant, monitoring component or entity selection, choose how to group it, and save the result when you want a view to share and refine.

The preview is temporary: opening it creates no saved view. It already shows live health, supports inspection and layout changes, and keeps the existing **Link graph** and **Network graph** available as alternate tabs.

This walkthrough uses a SecOps tenant with Data Source Monitoring (DSM), then shows a *Cribl Logstream example with a priority filter* using Flex Objects.

Step 1 — Open the tenant topology

Open the Virtual Tenant's **Tenant Home**, select the monitoring component you want to explore, and click the **topology icon** in the upper-right toolbar. The example starts from the `secops-v1` tenant's DSM overview:



The topology panel opens on the **Topology Studio** tab. Its source is the selection used to open it; the component-wide action in this example generates a map of the tenant's data sources.

Step 2 — Review the live preview and its scope

The initial canvas contains **aggregate nodes**, with a root for the tenant and groups chosen by the component's default hierarchy. For DSM, the default is **Index**. The root and each group show their own live aggregate health and KPI.



Use the controls above the canvas to decide what the view should represent:

- **Hierarchy** selects the grouping structure.
- **Include entities** adds individual entity nodes beneath the groups. It is off initially to keep the map compact; the aggregates still include their represented members and evaluate their live health.
- **Entity limit** bounds the number of distinct entities represented, including when individual entity nodes are hidden. Check the count beside **Temporary topology** before saving.

Important

In this capture, the preview represents **150 of 156 entities**. The aggregates describe those 150 entities, so their health may differ from the full Tenant Home summary. Raise **Entity limit** within the allowed range if you want to include more entities before saving. Turning off **Include entities** does not remove the membership limit.

Step 3 — Choose a hierarchy

Keep the component-aware default, choose an existing Tenant Home grouping preset, or use a custom hierarchy with **up to three distinct fields in order**. The available fields depend on the component and source records, and include **tags** and **labels**. For example, an ordered hierarchy using tags then priority creates a tag level followed by priority groups beneath it.

In the SecOps example, choosing **Tags** replaces the index groups with operational categories such as `business`, `cloud`, `firewall` and `vulnerabilities`:



Every intermediate group is a live aggregate of its descendants. Entities can belong to more than one tag or label group; overlapping membership is counted once in ancestor totals. Records without a value for a grouping field appear under an **(ungrouped)** branch.

Changing the hierarchy rebuilds the layout while retaining the applied condition and represented membership. If a hierarchy expands beyond the supported graph limits, the preview reports an error and prevents saving. Reduce the number of levels or the entity limit, then review the new scope.

Step 4 — Inspect and arrange the map

Click an aggregate to inspect its members. Here, the `vulnerabilities` group shows **75% healthy**; the inspector's **Members map** helps identify the member in red and the members in orange, without adding every entity to the main canvas.



While the view is temporary, you can:

- **Search the canvas** to locate nodes by label, entity, tenant or component.
- Show or hide **KPIs**, zoom, and inspect the groups and entities.
- Drag nodes, use **Undo**, or run **Auto-arrange** to organise the layout.
- Refresh live health, pause/resume refresh, or **Export PNG** for an image of the current map.

Use **Condition** and **Apply** when you want to filter the represented membership itself, as in the *Cribl example below*. Applying a condition rebuilds the layout, so apply filters and choose the hierarchy before making final layout adjustments.

Note

The generated connections describe the selected grouping hierarchy; they do not infer dependency impact propagation. Save the view to author membership, connections and other Studio settings. Switching between the topology tabs keeps the temporary preview; closing the panel discards an unsaved preview and its layout changes.

Step 5 — Save as a Topology Studio view

Click **Save as topology view**. In the **Create topology view** dialog, enter a unique **View ID** and a **Display name**, optionally add a description and category, and choose the roles that **Can view** and **Can edit** the result. The identifier cannot be changed later.

The example uses `secops-by-tag`, the display name `SecOps feeds - organised by tag`, and the category `Feeds tracking`:



Saving requires the usual Studio authoring permissions and a license that permits writes. A role in **Can edit** still needs TrackMe power permissions; see *Administration* for the access rules.

Click **Create view**. After the confirmation **Topology view saved**, click **Open in Topology Studio**:



The saved canvas opens with the generated groups and your layout, ready for continued authoring:



Important

Saving captures membership and layout; health remains live. The generated aggregates retain the explicit entity selection and any applied condition. New entities later added to the tenant do not automatically join this saved selection. You can change membership and aggregate settings in Studio.

From here, use *Authoring topologies* to add nodes, edit connections or refine aggregate policies, and *Decorations and canvas layout* to present the map. You can also configure *Topology Alerts* on the saved view to monitor its health.

To combine these groups with other items or saved views, add an *auto aggregate* as a parent in the saved view. Its incoming links compose the children's health without duplicating their selections or filters; the generated groups retain their own membership.

Example — Cribl Logstream with a priority filter

The same workflow works with other monitoring components. This example opens the **Flex use cases topology** for the `cribl-mon` tenant, containing *Cribl Stream monitoring entities*.

Keep **Hierarchy: Default (Group)** and enter the following in **Condition**:

```
priority IN ("high", "critical")
```

Click **Apply**. The condition selects the high- and critical-priority members from the source selection and applies to **all generated aggregates**, including the root. Use commas between values in an `IN` list.



The preview reports **20 of 20 matching entities represented**. Its groups cover destinations, source and destination health, infrastructure, pipeline traffic, and total traffic in and out. **Include entities** remains off, giving a compact health overview of the selected Cribl monitoring entities.

To reset the filter, clear **Condition** and click **Apply** again. If a condition is invalid or its results cannot be retrieved completely, resolve the displayed error before saving; edited condition text must be applied before creating the view. After applying a valid filter, check the matching-entity count and use **Save as topology view** to keep the result with its condition and layout.

8.7.3 End-to-end example — monitoring a production Web platform

This chapter builds a complete, production-grade topology from scratch — the exact workflow, screen by screen. The scenario is the monitoring of TrackMe's own Web platform, articulated around **two sides**:

- **Public Facing** — the websites customers reach: the main site, documentation, support portals, license services.
- **Internal Tooling** — the engineering-facing services behind them: CI, container management, virtualization, internal Splunk services.

Each side is observed from **two angles**, correlated on one canvas:

- **Active monitoring** — synthetic Web heartbeats (FLX entities) probing every URL: HTTP status, response time.
- **Trace logging** — the webserver access logs (DSM entities) and the hosts shipping them (DHM entities): event delay, lag, host activity.

By the end of the chapter the topology detects a real incident, opens an alert, sends lifecycle emails with the rendered map embedded, and resolves itself when the platform recovers. Every capability used here is described in depth in *Authoring topologies* and *Topology Alerts* — this page is the story of using them together.

Step 1 — Create the view

Open **Topology Studio** from the main navigation. On a fresh installation the landing page offers to create the first view, to have the *AI Builder* draft one from a plain-language description, or to import one:



Click **Create topology view** and describe the view: a view ID (the immutable `view_id` is derived from what you type and cannot be changed later), a display name, a description, a category — views sharing a category are grouped on the landing page, and can be filtered from the *views browser* — and the RBAC contract: which roles **can view** and which roles **can edit**.





The new view opens on an empty canvas. The **Start your map** helper summarises the node types — for a service map like this one, aggregates are the natural starting point:



Step 2 — The root aggregate

Tip

This example uses a **scoped root aggregate** with its own selection and health policy. Since TrackMe 2.4.18, an *auto aggregate* is another option when the root should derive its health entirely from connected children. Follow that guide for child-to-parent connections; the scoped-root workflow below remains valid.

The first node is the root of the map: one aggregate called **TrackMe Web** covering *everything* the platform depends on. Its scope spans two tenants — the Web-heartbeat tenant (FLX synthetic checks) and the production tenant (DSM data sources and DHM hosts) — and a **filter** narrows the membership to what matters here, combining the priority

and the labels that classify the two sides:

```
priority IN ("high", "critical") AND labels IN ("public-facing",
↪"internal-tooling")
```





Two health calculations are available. **Worst member** is strict — any red member makes the aggregate red. **Healthy percentage** classifies the healthy ratio through configurable bands — drag the two boundaries to decide how much of the 0–100% scale is red, orange, or green. The root node uses healthy percentage, so a single degraded URL does not paint the whole platform red:



Add the aggregate: the node lands on the canvas, already live — the inspector shows the resolved scope, the filter, the live status (16 entities, all green) and, further down, a members map of the matching entities around the aggregate:





Step 3 — Branding the node with a custom icon

Right-click any node for its context menu — inspect, rename, change icon, update the scope in place, connect from here, or remove. **Change icon** scrolls the inspector to the appearance section, where the icon catalog is available and **custom icons** can be uploaded (they are normalised client-side and stored with the deployment, so every viewer sees them):





Step 4 — The two sides: Public Facing and Internal Tooling

The second layer splits the platform into its two sides. Each side is an aggregate scoped to the production tenant (DSM, DHM and FLX) whose filter selects one label — first **Public Facing Web**:

```
priority IN ("high", "critical") AND labels IN ("public-facing")
```



Step 5 — Connecting the map

With two nodes on the canvas, enable **Connect** mode: click the **source** node, then the **target** node — the directed edge materialises immediately. Press Esc, click **Connecting…**, or simply save to leave the mode:


Topology Studio
TrackMe Production Web
Hide KPIs
Full screen
Export PNG
Add items
Add decoration
Build with AI
Background
Canvas size
Connect
Auto-arrange
TrackMe Web
Public Facing Web


Topology Studio
TrackMe Production Web
Hide KPIs
Full screen
Export PNG
Add items
Add decoration
Build with AI
Background
Canvas size
Connecting...
Auto-arrange
Select the source node
TrackMe Web
Public Facing Web





Repeat for the second side — **Internal Tooling Web**, same scope, filter on the `internal-tooling` label — and connect it under the root. The hierarchy takes shape:





Step 6 — Bulk selection and alignment

Hold **Command** (macOS) or **Control** (Windows/Linux) and click nodes to build a multi-selection — selected nodes carry a dashed blue ring, and the **bulk actions** bar appears: align in row, align in column, remove, clear. Aligning the two sides on one row keeps the layers visually clean:





Tip

Bulk selection covers auto-generated member entities too (see the next step) — group-drag, alignment and the shared actions apply to them exactly as to regular nodes.

Step 7 — Auto-generated member entities

So far every node is an aggregate. The next layer shows the *individual* URLs being probed — without placing them one by one. The **Public Facing Web Heartbeat** aggregate scopes the FLX component only and filters on the heartbeat entities of the public side:

```
object=web_heartbeat* AND priority IN ("high", "critical") AND labels IN␣
↪("public-facing")
```



Then enable **Automatically show and link member entities**: every entity matched by the aggregate renders as a linked node around it — membership stays live, so URLs added to the monitoring later appear on the map automatically, worst states first. The expansion is configurable:

- **Auto-entities filter** — an optional second filter that chooses *which* members are drawn, without changing what the aggregate measures.
- **Max entities shown** — beyond the cap (up to 500) the aggregate shows *+N more*.
- **Default icon** and the **"auto" marker** on generated links — one icon for every member, and whether the links show their non-removable *auto* tag.
- **Entity label field** — label the auto-generated entities with the **alias** (default, always available) or the raw **object** name; a per-entity rename always wins.
- **Default KPIs on member nodes** — up to five stacked KPIs applied to every member, picked from the scope's catalog or typed as dynamic metric names, each renamable inline (`http_status` becomes *status*, `response_time_ms` becomes *resp ms*), with a shared display format (stat card, coloured value, badge, sparkline).





Preview resolves the scope and filter live before anything is added — here the four public URLs, all green:





Repeat the pattern for the remaining branches — **Public Facing Web Logging** (DSM webserver access sources, delay/lag KPI), **Internal Facing Web Logging**, and **Internal Tooling Web Heartbeat**. Each branch correlates one side's active monitoring with its trace logging, and the map builds itself:



Note

Notice the `401` and `403` status values on some internal URLs — for authenticated services an unauthenticated probe legitimately answers 401/403, and the FLX use case treats those as the expected status. The topology shows the *value*; the entity's own thresholds decide what is healthy.

Step 8 — Adding individual entities

Specific entities can also be placed individually — here the DHM **hosts** shipping the webserver logs, so host-level activity is visible at the very top of the map. **Add entities** opens the cross-tenant picker: choose a tenant and a component, filter, select, and add:





Connect the hosts to the side they serve, and the design is complete:



Step 9 — Reading the map

Everything on the canvas is inspectable. Selecting an **entity** shows its live status, anomaly reason, and 24-hour metric sparklines (event latency, delay, volume, ML outliers, state events). Selecting an **aggregate** shows the state distribution and an interactive members map — click a member dot to preview it, then jump to its node or open it:





Entity details opens the same rich modal used across TrackMe — full status message, charts, and a jump to the entity's Tenant Home:



The canvas **search** matches labels, entities, tenants and components — matches are ring-highlighted and pre-selected, Enter cycles through them, Esc clears:



Step 10 — Alerting on the topology

A map you cannot act on is just a picture. **Alerts** (top right) opens the topology alerts manager — state-aware, scheduled evaluations of the view that notify on **transitions only**: open on breach, update when the set of unhealthy items changes, resolve on recovery. An event is emitted on every transition; email is an opt-in delivery channel on top:



New alert configures everything in one dialog: the alias, the view, the cron schedule (every 5 minutes here), whether orange also alerts, the target event index, and the delivery — enable **Send email on transitions**, pick the email account and recipients, choose the topology image theme for the embedded rendering, and optionally override the drilldown root URL (for reverse proxies or custom domains):





The **selection** decides what the alert covers. *Automatically select all aggregates and entities* covers every alertable node — including nodes added to the view later, without touching the alert. The alert list shows the live scheduling state, and each row's action menu gives the full lifecycle — edit, run now, alert events, logs, history, Ask AI, disable, delete:





Run history keeps every evaluation with its transition, item count, email delivery and duration — right after creation, runs settle at *None / 0 in alert*:



Step 11 — An incident, end to end

A real incident: the **license services website starts answering HTTP 503**, and the license-monitor data feed goes quiet. The canvas tells the story at a glance — the red states bubble up the branch: the failing URL and its feed go red, the side-aggregates degrade (67%, 83%), Internal Tooling drops to 78%, and the root re-classifies to 94%:



On the next scheduled evaluation the alert **opens** — the email carries the transition, the items in alert, and the rendered topology image:



As the incident propagates (the stale feed breaches its delay threshold, more aggregates classify red), the set of unhealthy items changes and the alert sends an **update** — never a duplicate:





Every transition is also an **event** — queryable, correlatable, and ready for downstream integrations:

```
index="trackme_summary" sourcetype="trackme:topology_alerts"␣
↪alert_id="trackme-production-web"
```



Full-screen mode turns the same live map into a wall display — the incident is readable across the room:



Step 12 — Recovery

When the license services recover, the map heals bottom-up, and the next evaluation **resolves** the alert — the run history closes the loop (open → update → resolve), and the resolution email reports how long the view stayed in alert:


Topology alerts
Topology alerts are scheduled, state-aware evaluations of the nodes of a topology view: they notify on transitions only — open on breach, update when the set of unhealthy items changes, resolve on recovery. An event is always emitted on every transition; email is an opt-in delivery channel on top.
New alert
Events (this view)
Showing alerts for view "TrackMe Production Web" — show all
Evaluation dispatched for "TrackMe Production Web" (id: 1785153607155146).
Alias
View
Selection
Enabled
Status
Last / Next run
Duration
Owner
Cron
Actions
TrackMe Production Web
TrackMe Production Web
All nodes (auto)
Enabled
OK
Close


Run history — TrackMe Production Web
Run history — TrackMe Production Web
When
Transition
Items
Email
Duration
Error
Back





Step 13 — Version history

Every save of the view was kept as a **version**. The history (clock icon in the header) previews each version's graph and restores any of them — restoring is **not destructive**: the rollback is written as a new save, so it is itself versioned and can be undone. The most recent 30 versions are kept:





Step 14 — Themes

The canvas follows the TrackMe theme — the same view in light mode, in the editor and as a full-screen wall display:





Where to go next

- *Views browser and display modes* — Cards, Compact cards and List layouts, filtering, and the deployment-wide default.
- *Authoring topologies* — every editor capability in depth: node types, edges, visuals, custom icons, backgrounds, import/export.
- *Topology Alerts* — the alerting engine: transitions, selection, delivery, events, escalation.
- *AI integration* — the AI Assistant understands your views, and can build topologies for you.
- *Administration* — permissions, tenant isolation, licensing, operational internals.

8.7.4 Authoring topologies

Creating a view

To start from existing monitoring data, you can also *generate a view from Tenant Home* (TrackMe 2.4.16+), choose its hierarchy and condition, then save it into Studio for authoring.

From the Topology Studio landing page, click **Create topology view**:

- **View ID** — type freely; the identifier is derived automatically from what you enter. It is unique and **cannot be changed later**, so it is worth a moment's thought — it is what deep links and exports refer to.
- **Display name** — the label shown on the card, the canvas and in alert emails.
- **Description** — optional, free text.
- **Category** — optional grouping label (for example `Production`). Views sharing a category are grouped under one heading in the Cards layouts of the *views browser*, and the browser's **Category** filter narrows the list to one of them.
- **Can view (roles)** — read-only access.
- **Can edit (roles)** — roles allowed to modify or delete the view.

Two guarantees are built into the role lists, so you cannot lock yourself out: administrators always have access regardless of what is listed, and the view's **creator always keeps edit access**. Roles with edit access can always view, so they need not be repeated in the view list.

Important

A role listed under **Can edit** still needs TrackMe *power* permissions to author. Granting edit to a user-tier role does not promote it — such a role remains read-only either way. See *Administration*.

The new view opens directly on the canvas editor.





The canvas editor

The editor is a direct-manipulation canvas: drag nodes to place them, click to inspect, right-click for context menus. The essential mechanics:

- **Explicit Save.** Nothing is persisted until you click **Save** — you can experiment freely. The editor warns before you navigate away with unsaved changes, and **Undo** (the button, or the universal **Ctrl/Cmd+Z**) reverts the last 25 structural operations.
- **Conflict detection.** If someone else saved the view while you were editing, Save is refused and you choose between reloading their version or overwriting it — no silent lost updates.
- **Multi-select and group moves.** Modifier-click (or select-all) selects several nodes at once — drag any one of them and the **whole group moves together**, with a live preview of where every companion will land. Bulk deletion works on the same selection.
- **Escape releases.** Grabbed a node (or a group) and changed your mind? **Escape** releases it back where it was — the drop never happens and no undo step is consumed. Undo remains the tool for a move that already landed.
- **Auto-arrange.** Compute a hierarchical layout from your edges, with configurable direction, spacing, wrapping and canvas sizing in **TrackMe 2.4.15**. Expanded aggregates' visible members participate too. Use it as a starting point, then refine by hand and **Save**. See *Auto-arrange* for the options and examples.
- **Large maps with bounded composition.** General canvas content is guarded by a storage-size backstop. Auto aggregates have a separate limit of **100 per graph** and **12 composition levels**; referenced views also have nesting and evaluation budgets. See *Auto aggregates*.

Version history and rollback

Every Save automatically snapshots the previous state of the view, so each view carries its own **version history**: open it from the canvas toolbar to list the versions (who saved, when, and from which action), preview any version's graph, and **restore** one. Restore is **non-destructive** — the state you are replacing is snapshotted too, so a rollback is itself reversible. Broke the map at 5 pm? Go back to how it was at noon in two clicks.





Auto-arrange

Auto-arrange organises the topology from its connections, placing roots and their descendants in a hierarchy. **TrackMe 2.4.15** extends this capability with four directions, three spacing presets, wrapping and control over canvas sizing. It also arranges the currently visible auto-generated members of expanded aggregates alongside the rest of the map.

Open **Auto-arrange** from the authoring toolbar, or right-click an empty area of the canvas and choose **Auto-arrange**. The options panel lets you decide how the map should flow before applying the layout:

1. Choose **Direction**, **Spacing** and **Canvas** behaviour.
2. Set **Wrap wide groups** and **Fit view after arranging** as needed.
3. Click **Arrange** to apply the layout. **Cancel** closes the panel without applying your choices.
4. Review the result, refine positions by dragging if needed, then **Save**. **Undo** reverts an arrangement as one change, including any canvas resize.

Choosing a direction

The **Direction** setting controls where the hierarchy starts and which way it flows. It changes node positions while preserving the existing connections and their source/target relationships.

The screenshots use the same **SecOps Feeds** map to compare all four choices, with **Balanced** spacing, **Grow if needed**, wrapping and viewport fitting enabled.

Top to bottom (the default) places the root above its descendants. **Bottom to top** places the root below them:





Left to right places the root on the left and flows towards the right. **Right to left** places the root on the right and flows towards the left:





Spacing and wrapping

Spacing controls the gaps between nodes and hierarchy levels:

- **Compact** reduces gaps for a denser map.
- **Balanced** is the default spacing.
- **Spacious** increases gaps to give branches more room.

The layout reserves space for the **KPI displays currently visible** on the canvas, including those on generated aggregate members. Hidden KPI displays do not reserve that space; arrange with KPIs visible if you intend to show them in the finished view.

Wrap wide groups is enabled by default. When a hierarchy level is too wide for the current canvas, its nodes wrap into additional rows. For a horizontal direction, the same option wraps tall groups into additional columns using the canvas height. Turn it off to keep each hierarchy level on a single row or column, allowing more canvas space if needed.

Canvas sizing and viewport fitting

The **Canvas** setting controls the logical dimensions of the map:

Option	Behaviour
Keep current size	Arrange within the existing canvas dimensions. A crowded map may have tighter gaps because the canvas cannot grow during this operation.
Grow if needed	The default and recommended option. Keep the existing dimensions, enlarging the canvas where the layout needs more room.
Fit to content	Resize the canvas around the arranged content, growing or shrinking it as needed within the supported canvas limits.

Fit view after arranging is a separate option, enabled by default: it adjusts the viewport zoom and framing to show the arranged map. Turn it off to keep your current viewport while applying the new layout. **Fit to content** changes the canvas dimensions; **Fit view after arranging** controls how you look at that canvas in the browser.

Shapes, images and text decorations **stay in their existing positions** during auto-arrange. Canvas fitting includes their extents so that resizing around the nodes does not leave those decorations outside the canvas. You may want to reposition a decorative frame or title after moving the nodes it describes. See *Decorations and canvas layout* for manual canvas sizing and decoration controls.

Expanded aggregates and saved layouts

In **TrackMe 2.4.15**, auto-arrange includes the **currently rendered members of expanded aggregates** and their links. Their arranged positions are retained relative to the parent aggregate and saved with the view. Collapsed aggregates, and expanded aggregates with no currently rendered members, keep their stored member layout. Expand the aggregates whose members you want to arrange first.

The last applied options are remembered **in your browser**, ready for the next arrangement. Those preferences are separate from the shared view: **Save** persists the resulting positions and canvas dimensions. The view is not saved merely by opening the panel or clicking **Arrange**.

Adding entity nodes

Add items → Add entities opens a cross-tenant picker: choose a tenant and a component, search, and multi-select the entities to place. Any entity from any tenant and any of the seven components (DSM, DHM, MHM, VOL, FLX, FQM, WLK) can join the same map, and so can the objects of the findings tenants (Cribl Stream and Cribl Edge objects, Splunk HEC pools, endpoints and tokens) — this is where cross-domain dependency maps come from (the data source *and* the host *and* the workload that produce a service, side by side). **Add items** also offers a template per findings tenant type (Cribl Stream, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors) that builds a block of its objects in one step.

New nodes land collision-aware near the canvas center, ready to be dragged into place.



Entity KPIs

Each entity node can display up to **five KPIs**, stacked as on-canvas displays next to the node — the numbers you want visible at a glance:

- **DSM / DHM / MHM / WLK** offer a fixed, typed KPI catalog (lag, latency, event volume, host counts, skip percentage, execution errors…).
- **FLX, FQM and WLK** additionally expose the entity's **own metrics**: whatever the entity reports in its `metrics` payload becomes selectable as a KPI — for Flex Objects this means your custom KPIs land on the map (WLK is hybrid: catalog first, metrics on top).

The node chooses one **display style** applied to its whole KPI stack: a themed stat **card**, a card-free colored **value**, a compact status **badge**, or a **sparkline** (label, value and a 24h mini trend line — see below); aggregate nodes additionally offer a health **donut** (their member-state distribution — a display only an aggregate can have). Every KPI in the stack can carry its own short **label** replacing the technical metric name — rename each one independently.





Sparkline KPIs — trends over time, at scale

The **Sparkline** display style (TrackMe 2.4.13+) turns each selected KPI into a label, its current value, and a **24-hour mini trend line** drawn right on the canvas — the same at-a-glance history the entity inspector's *Metrics (24h)* panel shows, now visible for every node you choose without opening anything.

Sparklines are designed to be **safe at any view size**: no matter how many nodes carry the style, the whole view's trend history is fetched by **one single batched metrics search**, refreshed every few minutes on its own schedule. The KPI *value* keeps updating with every live-status refresh — only the trend shape follows the slower cadence. A KPI without enough recorded history shows a dashed placeholder line instead of a trend, and PNG exports draw the real trend lines. In Topology Alert emails, sparkline-styled KPIs render as stat cards (the email has no live history at generation time).

The style is available for entity nodes and for the auto-generated members of an expanded aggregate (via the aggregate's member KPI defaults). Aggregate nodes keep their own display choices (card, value, badge, donut).





Adding aggregate nodes

A **scoped aggregate node** rolls up many entities into one node: pick one or more tenant/component scopes (up to 8 pairs) and optionally a **filter expression** — the same filter grammar used by Virtual Groups (`field=value` with wildcards, `OR` / `AND`, parentheses). By default the node folds to the **worst state** of everything it matches and shows per-state counts; alternatively, an aggregate can opt into a **healthy-percentage policy** — you set the orange and green thresholds, and the node classifies by the percentage of healthy members instead of the single worst one (the right semantics for large fleets where one red host should not paint a 500-strong aggregate red).

To combine existing canvas items instead, use an *auto aggregate*: incoming connections define its membership, without a second scope or filter.

The modal's **preview** shows the roll-up *and lists the matching entities themselves*, so you verify the filter selected the *right* entities — not just the right number — before you commit. Scoped aggregates have no separate per-view count limit; the normal graph storage-size guard still applies.

An existing aggregate's scope and filter can be **edited in place** at any time (right-click → *Update scope*): the node keeps its identity, position, edges, KPIs and member layout — only the membership definition changes.



Auto-populated aggregates

This expansion option belongs to **scoped aggregates**. It is distinct from an *auto aggregate*, which combines incoming health nodes and does not expand into generated members.

An aggregate can also **expand**: turn on **Automatically show and link member entities** and its current members appear as satellite nodes linked around it, worst states first. Membership is dynamic — entities that newly match the scope join the map on their own, entities that stop matching leave it — and the positions you give individual members are remembered. This is the "self-maintaining" corner of a map: author the aggregate once, and the topology follows reality.

The **Auto-generated entity options** section of the aggregate dialog shapes that expansion (the options marked 2.4.15 are new in TrackMe 2.4.15):

- **Auto-entities filter** (2.4.15) — a second filter, in the same grammar as the aggregate's own, applied *after* the scope, filter and exact selection. It only decides **which** members appear on the canvas: the aggregate's health, counts and alert evaluation still cover the full membership. Use it to draw the ten members that matter out of a 500-strong fleet without changing what the aggregate measures.
- **Max entities shown** — the ceiling on drawn members, up to **500** per aggregate (2.4.15; 50 before). Beyond it the node shows *+N more*.
- **Default icon** (2.4.15) — one icon for every generated member, from the catalog or your custom icons; a per-member icon override still wins.
- **Show "auto" on generated links** (2.4.15) — shows or hides the non-removable *auto* marker on the generated links in the editor.
- **Entity label field** — label members with their **alias** (the default) or the raw **object** name; a per-member rename always wins.
- **Default KPIs on member nodes** — up to five stacked KPIs, each renamable, applied to every member at once with a shared display style — one setting instead of fifty. The picker is pre-populated from the members the expansion actually produces, and the same values appear on the canvas, in the inspector and in the server-rendered alert emails.





Individual members can still carry their own **custom label and icon** from the inspector.

Auto-generated members can also join your dependency map directly: **connecting an edge to one pins it** — the member becomes a permanent entity node (keeping its name, icon and position, still linked to its aggregate), so membership changes no longer remove it and your edges stay valid. Search finds them, KPIs follow them, and the alert email renders them exactly once.



Adding view nodes — views inside views

Since TrackMe 2.4.15, a saved topology view can itself be placed on a canvas as a **view node**: one health node that stands for the whole referenced view. This is how a map of maps is built — an estate modelled as layered, independently owned views, each rolled up into the one above it. The referenced view keeps ownership of its graph, its aggregate policies and its own Topology Alerts; the parent stores its identifier only, never a copy, so anything saved in the child is reflected at the parent's next refresh.

Open **Add items → Add view** (the empty-canvas guide's **View** card and the canvas right-click menu lead to the same dialog). The dialog lists the views your roles can read, with a **category** filter and a search by name or identifier; the current view and views already on the canvas are excluded. Selecting a view shows its size (nodes and connections) before you add it. **Circular references cannot be saved** — a view can never reference itself, directly or through other views.





Once on the canvas, the node behaves like any other: move it, rename it, change its icon, connect it with edges, include it in bulk selections and, if you wish, let it *propagate* its state to what depends on it. What it *shows* comes from the child:

- **State** — the **worst effective state** of the child's health nodes (entities, aggregates and nested views), taken *after* the child's own aggregate policies and impact propagation. Labels and decorations contribute nothing.
- **KPI** — the **Healthy entities** card, enabled from the inspector's *KPI display* section, shows the percentage of healthy **unique underlying entities** across the child's aggregates, direct entities and nested views — members hidden by display limits included, overlaps deduplicated by tenant, component and key. Green and protected entities count as healthy. The percentage is deliberately separate from the policy-driven colour: a child that is red because one aggregate breached its policy can still report *86% healthy entities · 7 entities* — both are true.
- **Inspector** — the *Referenced view* section names the child, gives its state, its healthy percentage and per-state counts, and offers **Open view in new tab**: the child opens in its own tab while this canvas, its viewport and its unsaved edits stay exactly where they are.





To point a node at a different view, remove it and add the desired view — removing a view node removes the reference and its edges only; the referenced view is untouched.



A few rules keep views inside views safe at any depth:

- **Permissions at every hop.** Your read access to the referenced view *and* to every tenant it reaches, through any depth of nesting, is checked on every refresh. A child you are not allowed to read renders neutrally with the notice *This view or its contents are outside your permissions* — and a missing view looks exactly the same, so a guessed identifier reveals nothing. A parent that contains references you cannot read opens read-only.
- **Fail closed.** A cycle introduced after validation, an incomplete read of the child, or an exhausted budget makes the reference report *unavailable* health — never green. The bounds are generous (up to 12 nesting levels and 100 distinct views per evaluation) and no hand-authored estate approaches them.
- **Alerting and portability.** A view node is one alertable item in *Topology Alerts* — selected explicitly or through *all nodes* — and its transitive tenants count for scoped maintenance; the child's own alerts stay independent and are never executed by the parent. History, duplication and portable *export* keep the reference by identifier, but an archive does not bundle the child views: import the dependencies first, with the same identifiers, on the

destination.

- **Integrity.** A reference to a deleted view, at any depth, is reported by the *view integrity checks*.

Tip

Build with AI can compose view nodes too: it discovers the saved views you are allowed to reference and places them alongside aggregates and entities in its draft, and Save re-checks permissions and cycles like any other write.

Adding auto aggregates

Add items → Add auto aggregate creates a health parent with only a label. Connect each contributing entity, aggregate or view **to the parent**. Its colour follows the worst effective child state; its KPI counts each underlying entity once. There is no scope, filter or separate health policy on this parent.

You can also select an existing label and choose **Convert to auto aggregate** in its inspector. Identity, position, icon and links are preserved; verify the link direction before saving. Follow the *illustrated auto aggregate guide* for the full workflow and health rules.

Labels, icons and background

- **Label nodes** are free-text annotations with color control — name your zones, tiers and flows. They do not contribute health; convert a label to an *auto aggregate* when it should become a health parent for its connected items.
- **Node icons**: pick from a catalog of 74 icons with keyword search (databases, networks, clouds, applications, pipelines…). The same icons appear in the server-rendered alert emails.
- **Custom icons**: upload your own images — vendor logos, team emblems, product marks. TrackMe normalizes any picked image into a crisp square icon and stores it in a **deployment-wide catalog**: upload once, and every author sees it in the picker (a *Custom* section with live thumbnails) across every view. Uploads are validated server-side (PNG/JPEG, size and pixel-dimension caps), and a deleted custom icon degrades gracefully — nodes fall back to their default glyph, nothing breaks.
- **Background image**: upload a PNG or JPEG (up to 2 MB) as the canvas backdrop — a regional map, an architecture diagram, a site plan — with fit (cover/contain) and dim controls so node colors stay readable.

Beyond labels and backgrounds, the canvas can carry free-floating **decorations** — shapes, images and text drawn directly on the map (TrackMe 2.4.14+). They have their own page: *Decorations and canvas layout*.



Connecting nodes — edges

Toggle **Connect** mode, click a source node, then click a target: the link is drawn and saved with a one-way source → target relationship by default. Links are what turn a set of nodes into a *dependency map* — and they are also what auto-arrange, live impact propagation and alert rendering follow. For *auto aggregates*, those same links also define membership: select the **child as source** and the **auto parent as target**. Neither impact propagation nor link selection is required for a child to contribute; hiding the arrow does not remove the contribution.

Select a saved link on the canvas, right-click it, or open it from a node's **Connections** list to manage it in the link inspector. The inspector shows the stored **Source** and **Target** explicitly and lets you:

- choose **One way (→)** — show an arrow at the target and propagate from the stored source to the stored target;
- choose **Bidirectional (↔)** — show arrows at both ends and allow propagation in either direction;
- choose **No direction (—)** — hide the arrowheads while retaining the stored source-to-target propagation direction;
- **Reverse source and target** for a one-way or hidden-arrow link;
- select the link for impact propagation, add an optional label, or remove it.

Reversal is disabled while a link is bidirectional because both ends are already visually equivalent. Choose **One way** or **No direction** first when you really need to swap the stored endpoints. Reversal also clears **Select this link for impact propagation**: that selection belonged to the old source, so enable it again only if the new source should use the reversed relationship.

Automatically generated aggregate-member links are intentionally different: they are one-way, read-only links owned by aggregate expansion. Configure the aggregate rather than the generated link.



Impact propagation

A dependency map can do more than show colours side by side — it can **propagate any non-green status** onto the nodes that depend on it. Red is bad, orange is a warning, and blue is informational; only green means good and therefore does not propagate. The inherited status becomes the target's effective canvas and Topology Alert status, while its own native status is retained for explanation.

Choose the propagation mode in the source node's **Impact propagation** section:

- **Do not propagate** — the default; no dependency impact is sent. The node still contributes to any auto aggregate connected to receive it.

- **Propagate through selected links** — reflect the node's native non-green status one hop across every link whose **Select this link for impact propagation** switch is enabled. You may select several links.
- **Propagate through all downstream links** — follow every reachable relationship, transitively, for "if this fails, everything below it is affected" semantics. Link selection is not required in this mode.

Direction remains significant. One-way and hidden-arrow links propagate from their stored source to their stored target. A bidirectional link contributes both paths, so either endpoint can affect the other immediately when its node mode permits it. On a selected bidirectional link, the selection applies from either endpoint. Ordinary propagation-only cycles terminate automatically. Feedback involving auto-aggregate membership or impact is rejected to avoid a parent depending on its own health result.

Composition remains independent of propagation. An auto aggregate respects each child's effective state even with propagation disabled on the child. The parent's composed state is its native state and can itself propagate to other dependencies. See *Auto aggregates*.

An inherited status does not become a new source by itself. **Selected links** stays a one-hop operation even when the target also has propagation enabled; use **all downstream links** when the intended impact must cross the whole chain.

Status conflicts are resolved without hiding a more important local condition. For propagation precedence only, red outranks orange, orange outranks informational blue, and blue outranks good/green. An equal or higher native status therefore remains the target's own truth. Blue is still distinct from healthy green: a blue source can repaint a green target blue, but it does not open an alert.

Protected or maintenance-blue entities and nodes whose state is unknown are never repainted. A blue aggregate is the deliberate exception because its blue is only the derived roll-up of its members, not maintenance applied to the aggregate itself. A higher-precedence red or orange dependency may therefore repaint the aggregate while its native blue roll-up remains visible in the explanation.

Every repainted node identifies the cause in its inspector and reports its native status alongside the inherited one. The same effective status and attribution are used by Topology Alerts and their email details: red alerts, orange follows the alert's warning setting, and blue remains informational.



Consuming a view

Open a view and it is live: status polls on the deployment's cadence (60 seconds by default, stretched automatically on slow views — see *System-wide settings*), with the last-updated time and an honest countdown to the next refresh displayed under the header controls, plus a manual refresh and a pause toggle. Since TrackMe 2.4.15 a large view no longer sits greyed out while its first status loads: a compact **progress card** tracks graph preparation and status transfer while the canvas stays visible and usable.

Clicking a node opens the **inspector** — live state, per-metric sparklines for entities, a mini-map of members for aggregates, and a one-click jump to the entity's full Entity Overview in Tenant Home. The inspector opens at a size derived from your viewport and is **resizable**: drag its accent-coloured left edge (or use the arrow keys on it), and the width you pick is remembered in your browser.

For an **auto aggregate**, the inspector instead lists contributing nodes and their effective states; click a contributor to focus its node. The parent also shows its combined entity count and state distribution.

The scoped aggregate **mini-map is interactive**: set how many members it shows (1–50), and click any member dot — mouse or keyboard — to preview that member inline: state, name and tenant, its **24-hour metric sparklines**, and (TrackMe 2.4.15) the same destinations an entity node offers — **View status** opens a compact status card (state, impact score, status message) in place, **View details** the full entity modal, and **Open in Tenant Home** the entity's overview. On an expanded aggregate, **Open node** jumps straight to the member's own node and inspector.





For consumption at scale:

- **Canvas search** — a keyword box in the header finds nodes by label, entity name, tenant, component, or an aggregate's scope and filter — matches light up with halo rings on the canvas, **Enter** cycles through them (selecting each node and panning it into view), **Esc** clears. On a large map, this is how you find the node you

just added — or the one that's red.

- **Deep links** — every view has a bookmarkable URL; share the link, land on the map.
- **Full-screen mode** — a chrome-free rendering for a wall of screens, addressable directly by URL parameter so a kiosk browser can boot straight into it.
- **KPI visibility, also by URL** — the toolbar's **Show / Hide KPIs** toggle controls the on-canvas displays, and the choice is URL-addressable too (`&show_kpis=1` or `&show_kpis=0`): decide per screen what a kiosk shows, even though full-screen hides the toolbar. A complete wall-screen URL: `?view=<id>&fullscreen=1&show_kpis=0`.
- **PNG export** — one click captures the current canvas as an image for slides and incident timelines.





8.7.5 Auto aggregates — compose service health

An **auto aggregate** (TrackMe 2.4.18+) is a live health parent whose membership comes from **incoming connections**. Connect existing entities, aggregates or saved-view nodes to it, and it combines their health without another tenant scope or filter to maintain. Changes to the children's membership are reflected on the next live refresh.

Use it to combine the building blocks you already have: several infrastructure views under one service, domain aggregates under a platform, or a mixture of groups and individual entities. The children keep ownership of their selection and health policies.

Choosing the right item

Item	Membership and health	Use it for
Entity node	References one monitored entity and displays its live state and KPIs.	Showing an individual dependency.
Scoped aggregate	Selects entities through tenant/component scopes and a filter or an explicit selection; uses worst-state or healthy-percentage policy.	A live group of entities selected from monitoring data.
Auto aggregate	Combines incoming health nodes; follows their worst effective state.	A health parent over existing groups, entities or referenced views.
View node	References a separately saved topology view by its identifier.	Reusing an independently maintained map as one node.
Label	Free text and cosmetic colour; no health calculation.	Naming a zone or adding an annotation.

Note

An auto aggregate is different from an **auto-populated aggregate**. The latter expands a scoped aggregate's members into visible entity nodes. An auto aggregate combines nodes already connected to it; it has no independent scope, filter, health policy, member selection or expansion.

Add an auto aggregate

This example starts with five scoped aggregates: **Cloud**, **Web**, **Firewall**, **Identity** and **OS/Infra**. The parent will reflect these existing groups without repeating their filters.

1. Open a writable view and choose **Add items → Add auto aggregate**.
2. Enter a **Label** and click **Add auto aggregate**. No scope or filter is required.





The new parent starts **grey and unavailable** until it has contributors. Selecting it opens the inspector, which identifies its type as **Auto aggregate** and asks you to connect an entity, aggregate or view to calculate health. Placing it above the children does not create membership by itself.



Connect the contributors

1. Click **Connect** in the toolbar.
2. Select a **child first**, then select the **auto aggregate as the target**.
3. Repeat for each contributor. Press **Esc** to leave connection mode.
4. Inspect the parent to confirm the intended contributors, then **Save**.

The arrows point **child → parent**, even when the parent is positioned above the children. In this example all five groups contribute, and the parent's colour becomes orange because Identity is orange.



Connection rules

- **One way** contributes from the stored source to the target. Reversing the endpoints changes which auto aggregate receives the contribution.
- **Bidirectional** allows either endpoint to contribute to an auto aggregate at the other end, provided the result does not create a health feedback cycle.
- **No direction** hides the arrowheads but retains the stored source-to-target relationship. It does not disable membership.
- Multiple links to the same child count that child once. Labels and decorations do not contribute health.
- Removing a link updates membership. Removing the last contributor leaves the parent unavailable rather than healthy.

Read the health and KPI

The inspector's **Auto aggregate** section lists each contributor and its effective state. Select a contributor to focus its node and inspect it. **Live status** shows the combined state, entity count and distribution.



The parent colour and its healthy-entity percentage answer different questions:

- **Colour** follows the worst **effective child state**, after each child's own health policy and inherited impact. An orange child makes the parent orange even when most underlying entities are green.
- **Healthy entities** counts the union of underlying entities, deduplicated across overlapping children. Green and protected blue entities count as healthy. It does **not** average child percentages or count each child as one entity.

In the capture, **75 of 79 entities** are green and four are orange. The KPI therefore rounds to **95% healthy**, while the parent remains **orange**. The percentage does not apply a new health threshold to the parent.

Important

Impact propagation is a separate control. A child contributes to an auto aggregate through its incoming connection even when **Do not propagate** is selected. Enabling propagation is not required for composition. Keep using the propagation control when the node should also affect other dependencies; the auto parent's composed state can itself propagate downstream.

Convert an existing label

If a cosmetic label already sits above the right groups, select it and choose **Health → Convert to auto aggregate** in its inspector. Conversion preserves the node's identity, position, icon and connections, and replaces its cosmetic colour with calculated health. Check that the links point **towards** the new parent, then **Save**. **Undo** can revert the conversion before saving.



Nesting, permissions and alerts

An auto aggregate can contribute to another auto aggregate. A graph supports up to **100 auto aggregates** and **12 composition levels**. Feedback involving auto membership or impact is rejected when saving, so a parent cannot depend on its own result.

Contributors must be readable and fully evaluable. Empty, missing, restricted, unknown, truncated or partially stale contributors make the parent unavailable; it never reports a healthy subset as a complete result. Repair the contributing node or referenced view when its data is incomplete. An auto parent does not grant access to its children's tenants or views.

An auto aggregate is an alertable item in *Topology Alerts*. The canvas and alert evaluator use the same health rules, including dependency permissions and maintenance scope. Temporary unavailability freezes an existing alert rather than producing a false recovery. History, duplication, export and alert images retain the auto aggregate and its connections. Referenced views still need to exist on the destination when importing a map.

Generate or refine with AI

Both *AI workflows* understand auto aggregates. For example:

- In **Build with AI**: *"Create domain aggregates for these data sources and combine them under one auto aggregate, without duplicating their filters."*
- When **refining a draft**: *"Replace the root scoped aggregate with an auto aggregate fed by the existing children. Keep their policies and KPI settings."*
- In the **AI Assistant**: *"Convert this label into an auto aggregate and keep its incoming connections."*

Build with AI generates the child-to-parent links and validates the result; refinement preserves composition unless you ask to change it. Assistant edits use the existing confirmation flow. Review the contributors and live preview before saving the resulting view.

8.7.6 Decorations and canvas layout

A topology map often needs more than nodes and edges to tell its story: a title, a frame around a zone, a product logo, a screenshot of the service being watched. Since **TrackMe 2.4.14**, the Topology Studio canvas is also a **diagramming surface**: draw **shapes**, drop **images** and place **free text** anywhere on the map, and give the view an explicit **canvas size** so it presents exactly the same on every screen.

Decorations are deliberately *visual overlays*, not topology items:

- They **never move topology items**, and **auto-arrange leaves decorations in place**. Its canvas sizing accounts for their extents; decorations do not participate in the node hierarchy, statuses or alerting.
- They **follow canvas zoom and pan**, staying glued to the graph coordinates you gave them.
- They are **included in PNG exports and in Topology Alert emails** — the decorated map is the map everyone sees, everywhere.
- Like everything else on the canvas, they only persist when you **Save**, and **Undo** covers them.

Adding decorations

The authoring toolbar groups creation actions into two menus: **Add items** (entities, scoped aggregates, *auto aggregates*, *views* and labels — the nodes of the map) and **Add decoration** (**Shapes**, **Images**, **Text** — the overlays this page covers). A view can carry up to **100 decorations**; the menu disables itself at the cap.



The sections below build up a real example — the *TrackMe Public Web* view from the *end-to-end tutorial* — decorated with a screenshot per monitored site, a text title above each branch, and a frame around the whole fleet.

Images

Add decoration → Images embeds a picture into the view: a vendor logo, a team emblem, a screenshot of the application the branch monitors.

- **Any common source format works** — PNG, JPEG, GIF or WebP, up to 15 MB — and TrackMe **normalises** it into a safe, compact PNG or JPEG: the image is re-encoded, downscaled to at most 2048×2048 pixels, and compressed until it fits a small per-image budget, so even heavily decorated views stay fast to load and safe to store. (Animated GIF/WebP sources are flattened to a single frame.)
- **Images are stored with the view itself** — not as external references. Duplicating a view, restoring a version from history, or moving a view between deployments via portable export carries the images along; nothing can go missing.





Selecting a placed image opens its **inspector**: exact position and size, a **Transparency** slider (fade a backdrop image so node colours stay readable), a **Preserve aspect ratio while resizing** lock (on by default), and **Remove from map**.

Since **TrackMe 2.4.15** the inspector also offers a **Background** for the image: **None (transparent)**, the default, or **Solid colour** with a colour picker. A solid backing keeps transparent artwork — a logo, a line drawing — readable on a dark canvas or over a background image; it fills the image rectangle, the transparency slider fades image and backing together, and PNG exports and alert emails render it the same way.





Text

Add decoration → Text places free text anywhere on the canvas — view titles, zone names, annotations. Up to **2000 characters** per decoration, multi-line supported.

Styling is intentionally focused so that text renders **identically everywhere** — in the editor, in PNG exports and in the server-rendered alert emails:

- **Font**: Sans serif (default), Serif, or Monospace — generic families that every renderer resolves, rather than installed-font names that would differ between your browser and the email renderer.
- **Style**: Normal, Bold, Italic, or Bold italic.
- **Font size**: 8 to 240, with a live preview in the dialog.
- **Text colour**: free colour picker.





As with images, selecting placed text opens its inspector — position, size, and every styling control, editable in place with immediate effect on the canvas.





Shapes

Add decoration → Shapes draws outline shapes: **Rectangle**, **Square**, **Circle** or **Ellipse** — the classic tools for framing a zone, grouping a tier, or circling the part of the map a runbook talks about.

- **Line style**: solid, dashed or dotted.
- **Line colour**: free colour picker.
- **Line width**: 1 to 12, adjustable from the shape's inspector after placement.

Shapes are outlines by design — no fill — so they frame content without ever hiding a node or a status colour. Squares and circles keep their sides equal when resized; rectangles and ellipses resize freely.


Topology Studio
Shapes
Images
Text
Docs
Business
Support
Downloads


Add a shape
Shapes are visual overlays. They never move topology items or participate in auto arrange.
Rectangle
Square
Circle
Ellipse
Solid line
Cancel
Add shape



Working with decorations

Decorations handle like first-class canvas citizens:

- **Click** one to select it and open its inspector; **drag** to move; use the corner handle to **resize**. **Right-click** opens a context menu with the same styling controls plus *Remove from map*.
- **Bulk selection includes them**: modifier-click decorations and nodes into one selection, align them on shared rows or columns, or remove the whole selection at once.
- **Undo** (button or **Ctrl/Cmd+Z**) covers decoration changes on the same shared history as the rest of the canvas; slider-style edits such as transparency or colour coalesce into a single undo step per gesture.
- Decorations paint **beneath links and nodes**, so a frame or backdrop can never cover a status.
- Nothing persists until **Save** — decorations ride the same explicit save, version history and conflict detection as every other change.

The result, full screen — the decorated view as a wall display, exactly as it also renders in PNG exports and alert emails:



Canvas size

For automatic node placement, see *Auto-arrange*: **TrackMe 2.4.15** provides four layout directions, spacing and wrapping options, and a choice to keep, grow or fit the canvas when arranging. The controls below set the canvas dimensions manually.

By default the canvas simply fits its content — fine while a map grows, less fine when a view must present *identically* on the NOC wall, in full screen and on a laptop. **Canvas size** (TrackMe 2.4.14+) gives the view an explicit logical canvas area: once applied, centred and full-screen modes fit the **complete canvas boundary**, not just wherever nodes happen to be.





How it behaves:

- **Dimensions** are whole numbers from **400 to 8000** per axis; new views start at **1600×900**.
- **Existing views are untouched**: a view authored before 2.4.14 keeps its historical fit-to-content presentation until an author applies a size. For such a view the dialog **suggests a size derived from what you are currently seeing**, so accepting the suggestion preserves the presentation as-is while opting into fixed sizing.
- **Shrinking never loses anything**: items beyond a new, smaller boundary are moved to the nearest edge — nothing is deleted or cropped.
- **The canvas still grows on its own** when you drag an item near or beyond an edge, exactly as before — the explicit size is a floor for presentation, not a cage for authoring.
- The change is **one undoable operation** and, like everything else, persists when you **Save** the view.

Tip

Canvas size and decorations combine naturally: fix the canvas, frame it with a rectangle, title it with text — and the view becomes a stable, presentation-ready dashboard that renders the same in *full-screen kiosk mode*, PNG exports and *Topology Alert emails*.

8.7.7 Topology Alerts

A topology map answers *"what depends on what"*. **Topology Alerts** answers the next question: *"tell me the moment this picture degrades"*. Any topology view can carry one or more alerts — the topology itself becomes an alerting object, correlating the health of everything the map covers.



Why alert on a topology?

Classic per-entity alerting tells you *an* entity broke. A topology alert tells you *your service* broke — because you already encoded which entities constitute the service when you drew the map. One alert covers the payment flow end to end; the notification embeds the rendered map, so the recipient sees at a glance *which part* of the flow is red.

The state-aware engine

Topology Alerts are **state-aware**, not fire-and-forget. Each evaluation compares the current health of the covered nodes with the alert's stored state and only acts on **transitions**:

- **Open** — the covered scope degrades (red by default; orange optionally counts): a new alert cycle opens and notifies.
- **Update** — the situation *changes* while open (more nodes affected, fewer, different ones): an update is recorded. No change → no re-notification. **No spam by design.**
- **Resolve** — everything covered returns to health: the cycle closes, and the resolve notification reports **how long the incident lasted**.

Further semantics keep the engine honest:

- **Missing** nodes (the underlying entity was deleted, or, since TrackMe 2.4.15, its tenant was deleted or disabled or its component removed) are reported as such rather than treated as healthy or broken.
- **Unavailable** data (a tenant's collections temporarily unreachable) **freezes** the alert's state instead of transitioning — an evaluation that cannot see clearly never fabricates a resolve or an open.

- **Maintenance freezes too** (TrackMe 2.4.15). When TrackMe's global *maintenance mode* is active — immediate or scheduled, for every tenant or for a scope — and that scope covers **any** tenant the alert's selection touches, the evaluation is paused: no tenant is read, the alert's current state carries forward untouched, and no open, update or resolve transition, event or email is produced. Maintenance scoped to *other* tenants evaluates normally. The run history records the run as **paused, not failed**, and the alert list and the History view show a blue *Paused — maintenance* chip for as long as the window lasts. Treating maintenance as healthy would resolve every open alert the moment a window starts; evaluating anyway would open phantom incidents on the entities a window turns red — the freeze avoids both. Should the maintenance status itself be unreadable, evaluation proceeds normally rather than silencing alerting indefinitely.





If your map uses *impact propagation*, the alert engine honors it: a node repainted by an upstream failure **counts as degraded** — the alert fires on the same picture the canvas shows, and the notification names the upstream cause.

Note

From **TrackMe 2.4.15**, *view integrity checks* flag stale tenant/component references and alert configuration issues through Configuration Guardian and Topology Studio. An alert whose entire selection becomes missing can

resolve; the integrity check highlights that lost coverage. The **Alerts** modal also shows a per-view integrity chip for alert-level findings.

What an alert covers

Per alert, you choose:

- **the nodes** — an explicit selection of nodes on the view, or **all nodes** — which dynamically includes nodes added to the view later. Entity nodes, the nodes of findings-tenant objects (judged by their findings), scoped aggregates, auto aggregates and, since TrackMe 2.4.15, *view nodes* are all alertable: a view node counts as **one item**, judged by the referenced view's effective health, and the tenants it reaches through any depth of nesting count towards the maintenance scope described above. The referenced view's own alerts stay independent — the parent never runs them;
- **the trigger states** — red always alerts; **orange** optionally;
- **the schedule** — each alert runs on its own cron schedule (default: every 15 minutes), visible with next-run and run-now controls.

An *auto aggregate* counts as **one alertable item**, evaluated from its worst effective contributor state. The canvas and alert evaluator share the same composition rules. Its dependency tenants count towards permission and maintenance checks; temporary unavailability freezes an existing alert rather than reporting a false recovery. The healthy-entity KPI is an entity-level percentage, not the number of alertable nodes.

The create/edit panel gathers everything in one place: identity and schedule, the orange-as-alerting toggle, the event index, the delivery options — and the node selection, where **automatically select all** covers every alertable node of the view including nodes added later, while unticking it hands you a filterable per-node checklist:





Notifications

Notable events (always)

Every transition (open / update / resolve) emits a **notable event** with `sourcetype=trackme:topology_alerts` — searchable, dashboard-able, and available to any downstream pipeline you already run on TrackMe events. Each alert chooses its own **event index**; leaving it on the deployment default targets TrackMe's summary index.

Finding the events

You never have to guess where the events landed — the UI takes you straight to them:

- the **Alerts modal banner** states the exact `index` and `sourcetype` search scope for the current view, and the **Events (this view)** button opens a pre-scoped Splunk search across all of the view's alerts;
- per alert, the **Actions menu → Alert events** opens the same search narrowed to that alert (`alert_id="<alert>"`).





Each event is a structured JSON record carrying the full context of the transition: `transition` (open / update / resolve), `status`, `alert_id` and `alert_alias`, `view_id`, a stable `event_id`, and the per-node detail — `items_in_alert` (with component, state and anomaly reason per node), plus the `items_ok_count`, `items_missing` and `items_unavailable` breakdowns — everything a downstream correlation or ticketing pipeline needs without re-querying TrackMe.

Email (opt-in)

On top of the events, an alert can deliver **email** through any TrackMe email delivery account:

- branded HTML in dark or light theme, up to 50 recipients;
- **the topology itself, rendered server-side as an image** — with the same node icons as the canvas, colored by the state that triggered the notification, and including the **auto-populated members** of expanded aggregates with their KPI displays: the recipient sees the same map the canvas shows, not a table. Because an email is a fixed-size picture, the members drawn per expanded aggregate are capped by the deployment-wide *email*

image member ceiling (50 by default, worst state first) and the rest are summarised by an exact *+N more* note on the aggregate's KPI card; a view node is drawn as one node with its underlying entity count;

- a **deep link** back to the live view. The link root is auto-resolved from your Splunk Web configuration and can be **overridden per alert** — for reverse proxies and custom domains;
- resolve emails report the incident duration.





Managing alerts

The Alerts modal on each view is the management surface:

- create, edit, **enable/disable** and delete alerts;

- **run now** for immediate evaluation;
- **run history** — the recent runs with status, duration and outcome, so you can see the engine working (and prove a schedule is firing);
- **Ask AI** per alert — hand the alert and its recent runs to the *AI Assistant* for interpretation.

Hint

Creating and managing alerts on a view requires **write** access to that view (see *Administration*). If a view is deleted, its alerts are deactivated and remain inspectable by administrators and the alert's creator for cleanup.



8.7.8 Views browser and display modes

The Topology Studio landing page is the **views browser**: it lists every topology view your roles are allowed to read, and it is where you open, create, duplicate, export, import and delete views. Starting with **TrackMe 2.4.15**, the browser offers **three display modes — Cards**, **Compact cards** and **List** — a **category filter** next to the text search, and a deployment-wide **default display mode** that administrators choose in the System settings.



The three display modes

The **View** selector, next to the **Category** filter, switches between the three layouts. All three show the same views — the server filters the list against your roles before it reaches the browser — and all three honour the same permissions for every action; they differ in **density** and in **how you scan** the list.

Mode	What each view shows	Best for
Cards *(shipped default)*	Rich tiles grouped under **category headings**: display name and `view_id`, the description (two lines), node and edge counts, when and by whom the view was last updated, a collapsible **Roles** summary (view roles and edit roles), a **read-only** badge when you cannot edit the view, the *integrity badge* when findings exist, and the action buttons — **Edit**, **Duplicate**, **Export**, **Delete**.	A handful of views per team, where the description and the last-update attribution matter at a glance.
Compact cards	Narrower tiles, still grouped by category: name, `view_id`, node and edge counts, a `read-only` mention when applicable and the integrity badge. Descriptions, update attribution and role details are omitted; the actions fold into a single **Actions** menu.	Many views on one screen while keeping the visual, category-grouped layout — a wall of small tiles you recognise by name.
List	One flat, striped **table** — no category grouping — ordered by view name by default, with sortable **View**, **Category**, **Nodes**, **Edges**, **Updated**, **Access** and **Integrity** columns and a **Shortcuts** column with direct **Open / Edit** buttons plus an overflow menu for **Duplicate**, **Export** and **Delete**.	Large estates: scan dozens of views, sort by what was updated last, by integrity status or by access level, and open the right one in one click.

Cards

Cards is the layout Topology Studio has always used, refined in 2.4.15. Views are grouped under their **category** heading (the uncategorised bucket first, then categories alphabetically — the grouping is case-insensitive, so `Demo` and `demo` share one heading). Click anywhere on the card body to open the view; click **Roles** to expand the two role lists (**View** and **Edit**); the **read-only** badge tells you that you can consume the view but not modify it.

Compact cards

Compact cards keep the category grouping and shrink each tile to what identifies the view: its name, identifier, node and edge counts, and its integrity and read-only state. Everything you could do from a full card is still available from the **Actions** menu on the tile, filtered by your permissions on that view.



List

List renders the views as a single table you can **sort by any column** — click a column header to sort ascending, click again to reverse; ties are broken by view name. Category grouping is intentionally dropped in this mode: the **Category** column is sortable instead, and the **Category** filter above the table narrows the rows.

Column	Content
View	The display name (click it to open the view) with the `view_id` beneath.
Category	The view's category, or *Uncategorised*.
Nodes / Edges	The saved graph's node and edge counts.
Updated	How long ago the view was last saved, and by whom.
Access	**Read & write** or **Read only** — *your* effective access to that view. Hover the badge to see the view roles and, when you can write, the edit roles.
Integrity	The persisted *integrity verdict*: a badge with the number of *critical* or *warning* findings (click it to open the integrity panel), **No issues**, or **Not reported** when the view has not been checked yet.
Shortcuts	**Open** (the eye icon), **Edit** (the pencil — only when you can author and have write access to the view) and an overflow menu with **Duplicate**, **Export** and **Delete**, each shown only when permitted.



Hint

The List table has a minimum width and scrolls horizontally on narrow screens rather than collapsing its columns — the **View** column stays readable first.

Searching and filtering

Two filters sit above the views in every mode, and combine:

- **Search** — a free-text match against every human-meaningful field of the view metadata: display name, identifier, description, category, and the view/edit role lists. Typing `prod` finds the views categorised *Production* as well as the ones whose name or description mentions it.
- **Category** — a dropdown listing **All categories**, then **Uncategorised** and every category present in the views you can see, alphabetically. The choices come from the *full* list you have access to, so an active search never hides the other categories from the dropdown.

The counter next to the filters reports **N of M views** whenever a filter is active. When nothing matches, the browser says so and offers **Clear filters** to reset both the search and the category in one click. If a refresh or a deletion removes the category you had selected, the selection is kept — visibly — until you clear it, rather than silently showing unrelated views.

Note

Search and category selection are **session-local**: they reset when you leave the page. The **display mode**, on the other hand, is remembered — see below.

What is remembered, and where

Your **display mode** choice is a **personal preference**: the selector writes it into TrackMe's shared browser preferences cookie (`trackme_user_preferences`, the same cookie that carries your other UI preferences). It therefore:

- survives page reloads and reopening Topology Studio;
- is **per browser** — a different browser or workstation starts from the deployment default until you pick a layout there too;
- **wins over the deployment default** once set, even if an administrator changes the default later.

A missing, corrupt or unsupported cookie value is ignored and the browser falls back to the deployment default, then to **Cards**. Browsers that block cookies can still switch layout for the current visit — the choice simply is not persisted.

Deployment default — the administrator's choice

From TrackMe 2.4.15, administrators choose the **initial layout** for the whole deployment under **Configuration → System settings → Topology Studio → Default views display mode** (`topology_views_default_display_mode` in the `[topology_studio]` stanza of `trackme_settings.conf`): **Cards**, **Compact cards** or **List**.



The resolution order is simple:

1. the user's **own** saved choice, when the preferences cookie holds a valid one;
2. otherwise the **deployment default** from the System settings;
3. otherwise **Cards** (when the setting is unavailable or invalid).

Changing the deployment default affects **new users and users who never picked a layout**; it does **not** reset the personal choices already stored in other users' browsers. Pick **Compact cards** for deployments that expose many views but still benefit from a visual, category-grouped layout, and **List** when operators mostly scan, sort and open views from a dense table. See *System-wide settings* for the complete list of deployment-wide Topology Studio controls.

Permissions, in every mode

The display mode never changes *what* you can do — only how the controls are presented (buttons on a card, an **Actions** menu on a compact card, the **Shortcuts** column in the list). Every action is gated the same way in the three layouts, and re-enforced server-side:

Action	Requires
Open	Read access to the view — every view listed qualifies.
Export	Read access to the view — every view listed qualifies.
Duplicate	TrackMe *power* permissions (the copy belongs to you); read access to the source is enough.
Edit	Power permissions **and** write access to that view.
Delete	Power permissions **and** write access to that view.
Create, **Build with AI**, **Import**	Power permissions; the buttons are hidden from user-tier roles.

When the deployment is in license **read-only mode**, the authoring actions disappear in every layout while opening, exporting and reading stay available — see *Licensing*.

See also

- *Authoring topologies* — creating a view, choosing its category and its view/edit roles.
- *View integrity and consistency checks* — what the integrity badge and column report, and how to repair a view.
- *Administration* — the role model behind the **Access** column and the deployment-wide settings.

8.7.9 View integrity and consistency checks

Starting with **TrackMe 2.4.15**, Topology Studio includes a **view integrity linter**: a consistency checker that validates saved views against the current Virtual Tenants, enabled monitoring components, Topology Alert configuration and — since a view can be placed on another canvas as a *view node* — the saved views it references. As your environment changes, it identifies references that no longer resolve and explains how to repair them.

The linter is integrated with *Configuration Guardian* through the `topology_view_stale_references` check. Administrators receive Guardian findings, while users of a view see an issue badge, an integrity panel and an editor banner directly in Topology Studio.

This complements *Topology Alerts*: those alerts monitor the health of the entities covered by a map; integrity checks monitor whether the map's references and alert configuration still work. A green aggregate can still have a consistency warning if part of its original scope has disappeared.

For fully unresolved items — entity nodes on a deleted or disabled scope, aggregates with no live scope left, view nodes whose referenced view no longer exists — **Remove stale nodes** can remove all flagged stale nodes and their connected edges **in one click**, ready to review and save. Partially unresolved aggregates retain live coverage and are repaired by editing their scopes, as illustrated below.

Example: a view references a deleted Virtual Tenant

In this example, the **TrackMe Production** view contains two aggregates, **HF1: Queues** and **HF2: Queues**. Each originally spans three tenant/component scope pairs. The Virtual Tenant `splunk-sysmon` has been deleted, leaving its `FLX` scope unresolved in both aggregates.

Each aggregate still computes its status from the **two remaining live scope pairs**. The linter reports two **Aggregate partially unresolved** warnings so that this reduction in coverage is visible, even while the canvas shows healthy results.

1. Follow the Configuration Guardian notification

For an administrator, the Virtual Tenants page displays a Configuration Guardian toast naming the affected view. **View in Configuration Guardian** opens the corresponding check, where the finding includes the view name, issue count and recommended remediation.

Here, Guardian reports **one system-scoped alert for the view**, containing **two node findings**. The alert count and the number of integrity issues measure different things: one view can contain several affected nodes or Topology Alerts.





2. Open the view's integrity panel

On the Topology Studio landing page, the **TrackMe Production** card carries an amber badge with **2** issues. Click the badge to open **Integrity — TrackMe Production** without opening the canvas.

The panel checks the saved view against the current configuration and groups its findings into **nodes** and **Topology Alerts**. In this example it names both queue aggregates and explains that **1 of 3 scope pairs no longer resolves** because `splunk-sysmon / FLX` references a deleted tenant.

Use **Open view** to reach the canvas, **Open in Configuration Guardian** to inspect the check through the Guardian page, or **Re-check** to evaluate the saved view again. See *Permissions and refresh behaviour* for how re-checking updates the card badge.





3. Locate the affected aggregates on the canvas

Opening the view displays a **View integrity** banner above the canvas. In this case, it reports that two aggregates are partially unresolved.

- **Show affected nodes** selects the flagged nodes on the canvas so that you can locate and inspect them.
- **Details** opens the integrity panel from the editor.
- **Dismiss** hides the banner for the current opening of the view; it does not repair the references or clear the Guardian finding.





4. Repair the scopes and save the view

Select an affected aggregate to inspect it. Its **Live status** section shows the current entity results alongside a warning that **1 of 3 scope pairs no longer resolves**. The roll-up covers the remaining scopes only.

For **each** affected aggregate:

1. Open **Update aggregate → Scope**.
2. Remove the obsolete `splunk-sysmon / FLX` pair, keeping the live scope pairs. If monitoring moved to another tenant, select the intended replacement scope instead.
3. Apply the aggregate update, then **Save** the topology view after repairing both aggregates.

Saving refreshes the integrity check. Once all findings are resolved, the editor banner, card badge and corresponding Guardian alert clear automatically. The final screenshot shows the saved view with its aggregates preserved and the integrity banner gone.





Important

A **partially unresolved aggregate remains useful**: it still computes its status from its live scope pairs. It stays at warning severity, even when an enabled Topology Alert covers it, and is never removed by **Remove stale nodes**. Repair its scope to restore the intended coverage or remove obsolete references.

Example: remove fully unresolved items in one click

When one or more items are **fully unresolved**, Topology Studio can clean up all of those stale nodes at once. The editor's **Remove stale nodes (N)** button shows how many nodes it will remove; you do not need to select or delete them individually. The action is available to users who can edit the view.

In this second example, the **SecOps Feeds** view contains a **Qualification** aggregate whose only scope pair, `secops-qua / DSM`, references a disabled Virtual Tenant. The integrity panel reports **Aggregate fully unre-**

solved: **1 of 1 scope pairs** no longer resolves, leaving no live scope for this node.



To remove the obsolete items:

1. Choose **Open view** from the integrity panel. The canvas shows the unresolved **Qualification** node in grey and offers **Remove stale nodes (1)** in the integrity banner.
2. Click **Remove stale nodes (1)**. TrackMe removes the stale node and its connected edge immediately. With multiple fully stale nodes, the same action removes them all, together with their connected edges, as **one undoable change**.
3. Review the updated canvas. The notification confirms **1 stale node removed — Save to persist (Undo restores it)**, and the view shows **unsaved changes**. Use **Undo** if you want to restore the removed items.
4. Click **Save** to persist the cleanup and refresh the integrity result. Once all findings are resolved, the persisted card badge and Guardian alert clear.

The screenshots below show the canvas **before** and **immediately after** the single-click removal. In the second screenshot, the stale node is gone and the integrity banner has disappeared, but the cleanup is still waiting to be saved.



This cleanup removes obsolete items from the view. If the tenant or component should remain monitored, you can instead re-enable it or reconfigure the node's scope. **Partially unresolved aggregates are preserved** by the bulk action; edit their scopes as in the first example. Any separate Topology Alert findings still need to be reviewed and their configuration corrected.

Example: a view references a deleted view

A *view node* stores only the identifier of the view it references, so deleting that view leaves the parent with a reference that no longer resolves. In this third example, the **02 - SecOps View References** view is a map of maps: a **SecOps Feeds** aggregate connected to a dozen view nodes, one per security domain. The **Business** view has since been deleted, and an enabled Topology Alert selects the Business node — so the finding is **critical** rather than a warning.

The Configuration Guardian toast on the Virtual Tenants page is now **critical**, and the **02 - SecOps View References** card on the landing page carries a red badge with **1** issue.





The integrity panel reports one node finding, **Business · Referenced view missing**: the referenced topology view, or one of its nested references, no longer exists. A missing view *anywhere below* the node is reported on that node, whatever the depth. The panel also states that the node is **watched by one enabled Topology Alert**, which either silently resolved or can never fire on this node again — the reason for the critical severity.

Opening the view, the **Business** node renders grey, exactly like a view you are not allowed to read: live status deliberately shows a missing view and a denied one the same way, so that a guessed identifier reveals nothing. The linter, which evaluates the saved view under your own permissions, is what tells the two apart. The **View integrity** banner offers the usual controls: **Show affected nodes** selects the node, **Remove stale nodes (1)** is available because a missing reference is fully unresolved, and **Details** opens the same panel from the editor.


Integrity — 02 - SecOps View References
CRITICAL
NODES (1)
CRITICAL
Business
Open in Configuration Guardian
Re-check
Close
Open view
Topology Studio


Topology Studio
TrackMe
Alerts
AI assistant
Back
02 - SecOps View References
Hide KPIs
Full screen
Export PNG
Add items
Add decoration
Build with AI
Background
Canvas size
Connect
Auto-arrange
Show affected nodes
Remove stale nodes (1)
Details
Dismiss
Business
Identity
OS
SecOps Feeds
Cloud
Infra
Syslog
Web
Firewall
Net
Vulnerabilities
1 selected
Remove selected
Clear



To repair the view:

1. If the referenced view is gone for good, click **Remove stale nodes (1)** — the node and its edges are removed as one undoable change. If the view was recreated under a different identifier, remove the node, add the replacement with **Add items → Add view**, and reconnect its edges.
2. **Save** the view. The badge and the Guardian alert clear on save.
3. Review the Topology Alert that watched the node: its explicit selection now points at a removed node (**Alert selection drift**), so update the selection to the intended live nodes.

What the linter detects

Node and aggregate references

Finding	Meaning	Remediation
Deleted tenant	An entity node references a Virtual Tenant that no longer exists.	Remove the obsolete node or replace it with an entity from the intended tenant.
Disabled tenant	An entity node references a disabled Virtual Tenant.	Re-enable the tenant if it should still be monitored, or update the view.
Disabled component	An entity node references a monitoring component that is no longer enabled in its tenant.	Restore the component if required, or replace or remove the node.
Aggregate fully unresolved	None of the aggregate's tenant/component scope pairs remains active.	Reconfigure the scope or remove the obsolete aggregate.
Aggregate partially unresolved	Some scope pairs are inactive; the aggregate still computes from the remaining live pairs.	Edit the scope to remove or replace the stale pairs while retaining the aggregate.

An entity node on a deleted or disabled tenant, or a disabled component, is reported as **missing** on the canvas. An aggregate is missing when its entire scope is unresolved. These scope findings apply to **scoped aggregates**. An *auto aggregate* has no independent scope: repair its contributing nodes or referenced views when they become stale. An incomplete contributor makes the auto parent's health unavailable. A disabled tenant therefore no longer leaves nodes frozen on their last known colours.

View references

Finding	Meaning	Remediation
Referenced view missing	A view node references a topology view that no longer exists — or a view nested anywhere below it does. The node is fully unresolved.	Remove the node (**Remove stale nodes** includes it), or remove it and add the intended replacement view; then update any alert that selected it.
View reference cannot resolve	The reference forms a cycle through other views, or exceeds the nesting and size limits of the evaluator.	Break the cycle in the views involved, or flatten the nesting. This node keeps its place on the canvas and is not removed by the bulk action.

On the canvas, a view node whose target is missing renders grey, identically to a view outside your permissions; the finding is what distinguishes them. A referenced view you are **not allowed to read** is not a finding — it is a permission boundary, shown as a restricted node — and a transient failure to read a child view never proves it stale. Deleting a view queues a check across every view that references it, so parents are revisited promptly.

Topology Alert consistency

With alert checks enabled, the linter also detects:

Finding	Meaning	Remediation
Orphaned alert	A Topology Alert references a view that no longer exists.	Review and delete the obsolete alert, or recreate the required alert on an existing view.
Alert selection drift	An explicit node selection contains removed or stale nodes; an all-nodes selection has no live nodes left.	Repair the view and update the alert's node selection to cover the intended live nodes.
Alert not evaluating	An enabled alert's saved search is disabled or unscheduled, or its latest consecutive evaluations have failed.	Review the alert's schedule and run history, correct the reported errors, then verify evaluation succeeds.
Alert email account missing	An enabled alert has email delivery enabled but its configured email account is missing or empty.	Select an existing email delivery account, restore the required account, or turn off email delivery if it is no longer needed.

In the **Alerts** modal, the view column shows an **Integrity: N alert issue(s)** chip when that view has alert-level findings. This is a count for the view; open its integrity panel to identify the individual alerts and the reasons.

Note

The 2.4.15 linter checks tenant/component configuration, referenced views and alert consistency. It does not detect an individual entity deleted from an otherwise active tenant/component, or an aggregate whose valid scope currently matches zero entities.

Understanding severity and remediation

Amber / warning indicates inconsistent configuration, including a partially unresolved aggregate, a missing referenced view that no alert watches, or a missing alert email account. **Red / critical** indicates an enabled Topology Alert is affected: for example, it covers a stale node or a missing view reference, its view has disappeared, or it is no longer evaluating. The integrity panel explains which enabled alerts depend on a flagged node.

This matters because **missing nodes do not count as degraded entities** for Topology Alerts. An alert whose entire selection becomes missing can resolve; the Guardian finding identifies the lost monitoring coverage. A critical integrity finding calls for repairing that coverage even if the topology's operational alert has resolved.

For the bulk cleanup workflow, see *Remove fully unresolved items in one click*. Review any alert findings after removing nodes and update their selections separately.

Permissions and refresh behaviour

View access still applies. Read-only users can inspect integrity findings for views they can read. Tenant visibility is enforced: a node referencing a tenant outside the caller's visibility appears as **Restricted node**, with its details withheld. Authoring permissions and write access to the view are required to repair it or refresh its persisted verdict.

The card badge comes from the **persisted Guardian result**, while opening the integrity panel performs a **live check of the saved view**. **Re-check** repeats that live check for readers; for users who can author the view, it also requests a persisted refresh so the badge follows. Unsaved canvas edits must be saved before the live check can evaluate them.

This check's findings on shared **Configuration Guardian surfaces**, including Virtual Tenants toasts, are **administrator-only**. Other users access the findings through Topology Studio under the view's own permissions. See *Administration* for the read/write role model.

Detection and refresh happen at several points:

- View saves and other view writes, and Topology Alert configuration changes, refresh the affected view's result.
- Tenant deletion, enable/disable and component changes trigger a background check across views; allow that check to complete after a tenant change.
- Deleting a view triggers the same background check, so views that referenced it are revisited.
- TrackMe's general health manager runs the check daily as a safety net.
- Administrators can use **Run this check** in Configuration Guardian; view authors can use **Re-check** in the integrity panel.

If required configuration sources cannot be read, the panel identifies the incomplete check. An incomplete run does not clear or downgrade an existing Guardian alert. The panel also explains when a persisted refresh could not be performed, such as when the Guardian check is disabled.

Snoozing the Guardian alert also hides its persisted card badge for the snooze duration. Dismissing or snoozing a finding does not repair the view; a live integrity check can still report the underlying problem.

Guardian check settings

Administrators can find `topology_view_stale_references` in the Configuration Guardian **Check Catalog**, in the **Topology Studio** category. The check can be enabled or disabled and exposes two settings:

Setting	Default	Effect
`consecutive_run_errors`	3	Number of latest consecutive failed evaluations required to flag an enabled alert as not evaluating; configurable from 1 to 50.
`include_alert_checks`	true	Includes Topology Alert consistency checks. When disabled, checks cover node references only.

Guardian's existing snooze, notification and audit features apply to this check. See *Configuration Guardian* for their configuration.

8.7.10 AI integration

Topology Studio is a first-class citizen of TrackMe's *AI Assistant* — the assistant doesn't just *know about* your topologies, it can reason over them and, with your consent, **build and modify them for you**.

> **Note**
>
> Like every AI capability in TrackMe, this is **opt-in**: it requires the AI Assistant to be enabled and at least one AI provider configured. Without it, Topology Studio is fully functional — the AI entry points simply don't appear.

The topology chat context

The AI Assistant on the Topology Studio page runs in a **dedicated topology context**, scoped to where you open it:

- on the **views browser**, the badge reads *Topology Studio — all views* and the assistant sees every view your roles can read, with a bounded sample of recent graphs and alerts;
- on an **open view**, the badge names that view and the conversation is isolated to it: the assistant sees the view's full graph, its alerts and, since TrackMe 2.4.15, the saved views it may *reference*.

In both cases it sees only what your permissions allow and it understands the topology contracts — what a valid graph looks like, what aggregates and filters can express, how health rolls up, how alerts behave. The context is stored configuration, not live status: the assistant explains how a view is built and evaluated, it does not claim to know the current colour of a node. Pick the AI provider from the selector at the top of the panel.

Ask it things like:

- *"Which of my views cover the firewall data sources?"*
- *"Explain how this view determines aggregate health."*
- *"How would an auto aggregate combine these views without repeating their filters?"*
- *"What would be a good alert configuration for this map?"*



Build with AI — a topology from a description

Build with AI (TrackMe 2.4.13+) turns a plain-language description into a complete topology draft. It runs as a dedicated **AI Builder** agent rather than the chat assistant, which makes it fast and reliable across all supported providers, and it is available from three places:

- the **views browser** — preview and refine a draft before creating the view;
- the **editor toolbar** of a writable view — the draft lands on the open canvas as one pending change;
- the **empty-canvas guide** of a brand-new view.

The dialog asks for four things: **what the map should show**, in your own words; the **tenants in scope** (up to eight, pre-selected from those you can see); the **AI provider**; and optional **additional instructions** such as *"one aggregate per site, expand the public-facing group"*. Generation streams its progress, and the result is a **proposed topology** with a summary and the builder's notes — which tenant and component each aggregate was mapped to, what was left unfiltered and why.





What the builder can place:

- **scoped aggregates** with tenant scopes and filters — a scope-based building block, since a scope needs no entity identities;
- **auto aggregates** (TrackMe 2.4.18+) — health parents combining incoming entities, aggregates or view nodes without duplicated scopes and filters;
- **pinned entities**, drawn only from your real inventory — the builder never invents an identity;
- **dependency edges**, with auto-aggregate membership directed **child → parent** even when the parent is laid out above its children;
- since **TrackMe 2.4.15**, *view nodes* — it discovers the saved views you are allowed to reference and composes them alongside aggregates and entities; a draft that names a view outside that directory is rejected as a whole.

Every draft is **validated server-side** before it reaches the canvas, by the same validators as a hand-authored save. From the views browser, **Create view & open** saves the view and opens it in the editor; on an open view, the draft is applied as one **undoable pending change** — review it, adjust it, undo it — and nothing is stored until you **Save**, where the normal version history applies and permissions and cycles are re-checked. **Start over** discards the proposal

and returns to the description.



Note

Build with AI is interactive only and requires a configured AI provider. Automation users have a REST surface for it, documented in the REST API Reference.

AI-modified topologies through the assistant

The assistant can also change a view in conversation: *"add the linux hosts tenant's DHM entities as an aggregate under the ingestion tier"*, *"add a label above the three domain aggregates"* — and, for recovery, *"roll this view back to how it was before the last change"* through the view's version history. It knows your tenants and which components each has enabled, and on an open view it knows the real entities behind it.

Every write goes through the **consent-card flow**: the assistant presents exactly what it intends to do — the endpoint, the write level, the request body — and nothing touches your view until you **Confirm and run**. The proposal is validated by the same server-side validators as any hand-authored save, so the AI cannot produce a graph a human could not, and your tenant permissions bound what it can see, exactly as they bound the canvas. Since 2.4.15 the assistant also preserves existing view nodes when it edits a graph, and only references views from the directory it was given.



Auto aggregates in generation and refinement

Both the Assistant and Builder understand *auto aggregates*: membership follows incoming connections, health follows the worst effective child state, and the KPI counts unique underlying entities rather than averaging child percentages. This is separate from expanding a scoped aggregate into automatically displayed entity nodes.

For a new map, ask Build with AI to *"Create three domain aggregates and combine them under one auto aggregate; keep the filters on the children."* For existing saved maps, ask for *"An auto aggregate over these three saved views, using view references rather than copying their definitions."*

In the full-screen preview, **Refine this draft** applies follow-up instructions to the latest successful draft. For example: *"Replace the scoped root with an auto aggregate fed by the existing children. Preserve their health policies and member KPIs."* Review the incoming links, contributors and live health before **Create view & open**, or before applying to an existing editor. Failed or cancelled refinement keeps the previous draft.

For a small change to a saved view, ask the Assistant to *"Convert this label into an auto aggregate and retain its incoming connections."* The Assistant preserves existing membership, child policies and links during unrelated edits, and proposes the change through the usual consent card. It explains configured health rules; the canvas preview supplies current live health.

Generated and refined graphs pass the normal permission, cycle and composition checks. The Builder supports **40 scoped and auto aggregates combined** in a draft; this is separate from the manual canvas composition limits.

Per-alert Ask AI

Each Topology Alert carries an **Ask AI** action: the assistant receives the alert's definition and recent run history and helps you interpret it — why it opened, whether the schedule and scope make sense, what the run-history pattern suggests.

Hint

The assistant's knowledge of the topology contracts is kept in lock-step with the product by continuous integration — when Topology Studio evolves, the assistant's understanding evolves in the same release.

8.7.11 Administration

Permissions

Topology Studio follows TrackMe's standard three-role model:

Role	What it can do
`trackme_user`	Open and consume shared views (live status, inspector, full-screen, deep links).
`trackme_power`	Everything above, plus **author**: create views, edit views they have write access to, and manage Topology Alerts on those views.
`trackme_admin`	Everything, on every view.

Per-view access — read/write split

Each view carries **two role lists**:

- **Read roles** — who sees the view on the landing page and can open it.
- **Write roles** — who can edit the canvas, change the view's settings, and manage its alerts.

The view's **creator always retains access** to their own view, and administrators always see everything. Alert management is tied to **write** access — being able to read a map does not grant the right to change how it alerts.

The *views browser* reflects this split for every user: a **read-only** badge on cards, and an **Access** column (*Read & write* / *Read only*) in List mode, computed server-side for the caller — the actions each user sees (Edit, Delete…) follow the same verdict in all three display modes.

Tenant isolation

A topology view can reference entities from many tenants — but **viewing a map never extends anyone's tenant permissions**. Live status is enforced server-side, per node, against the caller's tenant visibility:

- Nodes referencing tenants the caller **can** see behave normally.
- Nodes referencing tenants the caller **cannot** see return a `restricted` marker only — **no entity identity, no state, no counts**. The UI renders them as neutral restricted nodes.
- Scoped aggregates spanning visible and non-visible tenants fold **only the visible scopes**, and report how many scopes were withheld.
- **Auto aggregates** depend on their contributing nodes and referenced views. Restricted or incomplete contributors do not produce a healthy partial roll-up; the parent is masked or unavailable as appropriate. Composition never grants access to another tenant or view.

Important

This makes cross-team maps safe to share broadly: everyone sees the same topology *shape*, while the live data each person receives is exactly what their tenant permissions allow — enforced in the backend, not the browser.

Licensing

Topology Studio is available to the **Foundation**, **Enterprise** and **Unlimited** editions alike (and to active trials) — the complete experience, including **authoring**: creating, editing, duplicating, importing and sharing views, and managing

Topology Alerts. Foundation access was opened in TrackMe 2.4.12; earlier 2.4.x releases mistakenly restricted authoring to Enterprise and Unlimited.

The only license gate is TrackMe's product-wide **license read-only mode**, entered when the license expires:

- **Creating, modifying and deleting** views and alerts is refused (the REST API answers `HTTP 402`).
- **Consuming** keeps working — existing views stay listed and open live, with the inspector, full-screen, deep links and exports available as usual.

Export, import and portability

Views are **portable**:

- **Export** produces a single archive containing the graph, the background image and the view's alert definitions — everything travels together.
- **Import** recreates the view on any TrackMe instance (staging → production, instance migrations, sharing between deployments), optionally under a new view id. Imports are size-capped and fully re-validated on arrival — an imported view obeys the same rules as one authored in place.
- **View nodes travel by identifier.** An archive keeps every *view node* as a reference to the child's `view_id` but does **not** bundle the child views themselves: import the referenced views first, with the same identifiers, on the destination — an import whose references cannot be resolved is refused.
- **Restricted exports stay honest.** When part of a view's tenants are outside the exporting user's visibility, the archive is marked as tenant-restricted and omits the alert definitions (they could name withheld entities); such a partial archive is rejected on import rather than silently recreating an incomplete view.
- **Auto aggregates retain their membership mode and connections.** Import them into a version supporting auto aggregates (2.4.18+), with any referenced views already available. Existing scoped aggregates keep their scope-based behavior.

Topology views are stored in global KV store collections, so they are automatically included in TrackMe's *backup & restore* cycle.

System-wide settings

Topology Studio controls are **admin-configurable, deployment-wide** — under **Configuration → System settings → Topology Studio** (the `[topology_studio]` stanza of `trackme_settings.conf`). Six numeric settings were introduced in TrackMe 2.4.9; **TrackMe 2.4.15** adds two more — the **default views display mode** and the **email image member ceiling** — for eight in total:



Setting	Default	Controls
`topology_versions_retention`	30	Restorable versions kept per view. Every save is a version; older versions beyond this count are pruned on save.
`topology_refresh_interval_seconds`	60	Base cadence of the canvas live-status refresh. This is the **floor**, not a fixed rate — the adaptive engine only ever *stretches* the interval on views that are slow to evaluate.
`topology_views_default_display_mo`	`cards`	**Default views display mode** (2.4.15): the initial layout of the *views browser* — `cards`, `compact` or `list` — for users who have not yet picked a layout themselves. A user's own choice, stored in their browser preferences, always takes precedence; changing this setting does not reset existing personal choices.
`topology_icon_max_count`	100	Capacity of the deployment-wide custom icon catalog.
`topology_background_max_kb`	2048	Maximum size of an uploaded canvas background image, in kilobytes.
`topology_alert_runs_retention`	50	Evaluations kept per topology alert in the run history, pruned after each run.
`topology_snapshot_max_rows`	50000	Maximum entity rows read per (tenant, component) snapshot when resolving live status and evaluating alerts. When a source is capped, aggregates report *truncated* rather than pretending the roll-up is exact. Raise only on very large estates with KV store capacity to match.
`topology_email_max_members_per_ag`	50	**Email image: members drawn per aggregate** (2.4.15). Maximum auto-generated member nodes drawn per expanded aggregate in the **topology alert email image**, worst state first (range 1–500). The canvas can expand an aggregate to 500 members because the viewer can pan and zoom; an email is a fixed-size picture, so the members beyond this ceiling are summarised by an exact *+N more* note on the aggregate's KPI card. Only the email rendering is bounded — the canvas, live status, aggregate health and alert decisions are unaffected, and an aggregate's own *Max entities shown* still applies when it is lower.

Every consumer reads these settings **fail-safe**: a missing or invalid value falls back to the shipped default, and values are clamped server-side to a sane range — a broken setting can never take Topology Studio down. Each setting also carries its own *learn more* documentation directly in the System settings page.

Hint

Which settings are read *when* differs: the refresh interval and the display mode are picked up the next time Topology Studio (or a view) is opened; the email member ceiling is resolved once per alert transition render; the retention and size caps apply on the next save, upload or run.

Operational internals

Auto-aggregate composition is bounded to **100 auto aggregates per graph** and **12 levels**. These are validation bounds rather than configurable system settings. Evaluation limits or incomplete dependencies produce unavailable health. See *Auto aggregates*.

For administrators who want to know what's running under the hood:

- **Storage** — views, background assets, version history, alert definitions, alert state and run history live in dedicated global KV store collections (`kv_trackme_topology_*`). Version snapshots are bounded per view and pruned automatically on save; deleting a view cascades its alerts, assets, version history and its *integrity* finding in Configuration Guardian, and queues a re-check of the views that referenced it.

- **Alert execution** — each Topology Alert materializes as its own scheduled saved search running the evaluation command on the alert's cron schedule. The saved search is owned by the system (`nobody`) and kept private, so end-user searches never gain direct access to the Topology KV store collections; the alert's **creator** remains the authorization principal — their power permissions, write access to the view and tenant visibility are re-verified on every run, and a creator who lost access skips the run (state untouched) rather than producing a false all-clear. Enable/disable in the UI schedules/unschedules it; per-alert run locks prevent overlapping evaluations (a *run now* and a scheduled run coordinate rather than collide).
- **Notable events** — every alert transition (open / update / resolve) is indexed under `sourcetype=trackme:topology_alerts` in the alert's own configured **event index**; when left on the deployment default, events land in TrackMe's summary index (`[index_settings] trackme_summary_idx`, `trackme_summary` unless changed). The TrackMe notable index is reserved for the notable alert action — Topology Alerts never write to it. See *Topology Alerts* for the event payload and the UI shortcuts that open the events.
- **Email rendering** — the emailed topology image is rendered server-side; no browser involved, no external service.

Hint

If an alert's saved search drifts (deleted or altered outside TrackMe), the scheduling layer detects and self-heals it on the next management-surface interaction.

8.8 Machine Learning

TrackMe ships its own native machine-learning engine — per-tenant statistical models that learn each entity's normal numeric behaviour and flag deviations, feeding the impact score rather than raising alerts on their own.

- **Outlier detection** — how the models work, seasonality, confidence, scoring, and which components benefit, with a full in-depth reference.
- **Incremental baseline training** — why a model may retain bounded evidence across training cycles, how that changes the statistical meaning of its baseline, and when to prefer a full rolling-window rebuild.

8.8.1 Outlier detection

TrackMe includes a built-in machine-learning engine that learns each entity's *normal* numeric behaviour from history and flags deviations. It runs natively inside TrackMe — no Splunk Machine Learning Toolkit dependency — with per-tenant models and impact-based scoring, so an anomaly rarely raises a false alert on its own.



The important distinction is between a **statistical deviation** and an **operational alert**. A model first decides whether the KPI is outside its expected range. TrackMe then qualifies that evidence and, when accepted, adds the model's score to the entity's other signals. The entity changes state only when the combined impact score reaches its configured threshold.



The entity's **Outliers anomaly detection** tab is where this surfaces: counters for the outlier impact score and lower/upper-bound (and rejected / corrected) outliers, and a chart of the metric over time against the model's learned thresholds, with anomalies flagged. When an outlier drives the state, the status reason reads `ML_outliers_detection`.

Note

This page is the **overview**. For the full reference — the density-function engine, true-context simulation, managing and tuning models, the backend, system-wide options, and troubleshooting — see *Outlier detection — in depth*. To understand why a model may retain history across training cycles, see *Incremental baseline training*.

How it works

Detection runs in two phases:

- **Training** (`trackmefit`) — TrackMe builds and trains a model from the entity's historical metrics and stores it. Models are created automatically when entities are discovered, and retrained on a regular cadence. Full training rebuilds the current calculation window; opt-in incremental training retains a bounded sample across cycles.
- **Monitoring** (`trackmeapply`) — each cycle, the trained model is applied to the current value, producing an *is-outlier* flag, the expected boundaries, and a confidence level.

The native engine auto-selects the best-fit statistical distribution per model (or you can pin one), and models are stored **per tenant** for clean lifecycle management and good search-head-cluster performance.



A bound breach is therefore a **candidate outlier**, not an automatic alert. Boundary safety can correct an unusable or implausible range, a configured minimum-value rule can reject the candidate, and low confidence keeps the result visible while suppressing its score. Only an accepted breach with normal confidence contributes the model's configured score.

Seasonality

Models can learn time-based patterns, so "normal" shifts by time of day or day of week. Choose a **time factor**:

- **none** — for steady, time-independent KPIs (a fixed-rate feed, a constant host count).
- **hour of day** — 24 slots.
- **day of week** — 7 slots.
- **day-of-week × hour-of-day** — for strongly weekly patterns.

Tip

Match the seasonality to the metric. Using *hour of day* on a metric that really varies by weekday misses the pattern; applying any seasonality to a non-seasonal KPI just adds noise — set it to *none*.

Confidence and scoring

- **Confidence** is **low** until the model has enough history (by default **30 days**) and **normal** thereafter. A **low-confidence** model still runs, but its score contribution is **suppressed** so an undertrained model cannot raise false alerts.
- A detected outlier contributes an **outlier sub-score** to the entity's total score (see *Impact scoring & thresholds*). By default that contribution sits **below** the critical threshold, so an outlier alone will not turn an entity red unless you raise its weight or it combines with other anomalies.

Note

Training data. Allow at least the minimum history (**30 days** by default, governed by `splk_outliers_min_days_history`) for normal confidence — for seasonal models such as day-of-week × hour-of-day this also gives each slot enough samples.

Handling false positives

The **Set as false positive** action on an entity's Outliers tab suppresses an outlier-driven alert without retraining or disabling the model — it cancels the current outlier score, and thanks to the immediate score cache it takes effect on the very next evaluation. It is the manual counterpart to the automatic low-confidence suppression, and works for a single outlier or a whole entity. Per-model threshold guards and deviation checks reject outliers outside sensible value bounds.

You can also **simulate** a model in a true-to-life preview before applying changes, and run train / monitor / enable / disable / reset in **bulk** from the Tenant Home. The **ML Outliers Anomaly Detection** view lists an entity's models with their status and last train / monitor times, shows the model's **confidence** (here *normal* — enough history to proceed), and runs a simulation over a chosen window so you can see exactly where the model would flag anomalies before committing:



Which components benefit

Component	Outlier fit
VOL	**The purpose of the component —** outliers on the rolling licensed volume of every index (drops and / or spikes) *are* its state. Always on; one model per index by default, trained on history backfilled at creation. See *VOL — Volume Outliers*.
FLX	Natural fit — continuous, often seasonal metrics. **Enabled by default.**
DSM	Available, **off by default —** for volume anomalies we recommend **VOL** in a separate tenant (available in all editions), which keeps outliers detection separate from feed monitoring. Enable it on DSM if you also want per-feed models.
DHM	Available but selective — per-host volume is noisy. Off by default.
FQM, **WLK**	Available but not recommended — field-quality and scheduler metrics are too slow-moving, discrete, or regime-dependent. Off by default.
MHM	**Not applicable —** outlier detection is not available for the Metric Hosts Monitoring component.

Where it applies, outlier detection is enabled or disabled per tenant and per component.

Choosing where behavioural detection runs

"Host outliers or not" is usually the wrong question. Host-level (DHM) outlier detection ships **off by default** because most environments have limited control over endpoints and are quickly overwhelmed by host count and granularity — but that is a default, not a prohibition, and the value depends entirely on *which* hosts:

- Detecting a suspicious change in volume on an **Active Directory / domain controller** is genuinely valuable.
- Trying to detect volume variation on an ordinary **laptop or workstation** feed has little to no value.

The real question is **scoping outlier detection to the entities where behavioural change is meaningful**. Rather than a blanket on/off, use the **Outliers scope**, entity **labels** and **tags**, and policy management to enable models only for the hosts and feeds that matter — critical servers, sensitive sources — and leave the long tail of low-value endpoints out. That preserves the value while cutting the cost. For broad behavioural detection on **volume**, the right level is the index, and the right source is the **license usage log**: that is exactly what the **Volume Outliers (VOL)** component does — one entity per index, one incremental search per tenant, the history backfilled at creation, drops and spikes per index — available on every edition, Foundation included:

Note

Before 2.4.18 the recommended index-level option was a dedicated tenant built on **Flex Objects over the licensing logs** (`splk_license_usage_per_index_*` templates). VOL supersedes it: same source, purpose-built collection and UI, no template to configure, and included in Foundation. The templates remain in the Flex library for custom needs; the *Volume Outliers white paper* walks the new component end to end.

How it is controlled, tenant-level (applied once):

- **Whole tenant on/off —** *Configuration → Virtual Tenants → edit the tenant →* **Enable Machine Learning**, under *Feature Behavior Settings*.
- **Which components run outliers —** the **Enable Machine Learning allowlist**, in the same editor. Dropping DHM from the allowlist is the surgical option: ML stays on for DSM / FLX.
- **Which entities carry models —** the ML Outliers management **Outliers scope**, plus entity labels and tags.

Note

The ML cost lever is scope, not cadence. Do not reduce the ML-monitor frequency to save cost — that opens scoring gaps and delays detection. Choose which entities carry models instead.

See also

- *Incremental baseline training* — when and why to retain bounded historical evidence, how the reservoir evolves, and how to verify it.
- *Outlier detection — in depth* — the full reference: the density-function engine, true-context simulation, managing models, backend, system-wide options, advanced tuning, and troubleshooting.
- *Impact scoring & thresholds* — how the outlier sub-score feeds the total score.
- *VOL — Volume Outliers*, *DSM — Data Source Monitoring* and *FLX — Flex Objects* — the primary consumers.

8.8.2 Incremental baseline training

Outlier detection depends on a definition of *normal*. With the factory-default **full** training mode, TrackMe rebuilds that definition from the model's current calculation window and replaces the previous baseline. **Incremental** training is an opt-in alternative that retains a bounded, representative sample across successful training cycles and refits the model from that accumulated evidence.

The purpose is continuity: an older but still valid operating pattern does not disappear merely because it fell outside the latest calculation window. At the same time, new completed observations enter the retained sample, so the baseline can evolve rather than remaining frozen.

Important

Incremental does **not** mean that TrackMe blindly appends every point forever, nor does it mean online `partial_fit`. TrackMe keeps a bounded, seasonality-aware **reservoir sample** and performs a complete density-function refit from that sample after each successful update.

Why choose incremental?

Incremental training is useful when:

- a rolling calculation window is too short to retain infrequent but legitimate operating regimes;
- replacing the complete baseline on every training run makes boundaries move more than the underlying service behavior warrants;
- you want a model to accumulate evidence across many training cycles while keeping memory, model size and fitting cost bounded;
- you want explicit operational evidence of what was retained, how many observations were seen, and whether a run bootstrapped, updated or did nothing.

Examples include business processes with rare month-end activity, low-frequency batch patterns, or stable services where a short-lived lull should not erase knowledge of normal high-volume periods.

It is not automatically the better choice. Full training remains preferable when the most recent window is deliberately the source of truth, when old regimes should be forgotten quickly, or when a KPI changes meaning often enough that long-lived historical evidence would be misleading.

Full and incremental behavior

The two modes change **training memory**, not monitoring:

Table 15: Full and incremental baseline behavior

Question	Full	Incremental
What data defines normal?	Every point in the current calculation window.	A bounded sample of all observations accepted since the last bootstrap.
What happens on the next train?	Refit the current window and replace the previous baseline.	Accept newer completed points, update the retained sample, then refit.
What happens to older behavior?	It is forgotten when it leaves the calculation window.	It remains eligible for retention, subject to reservoir sampling.
How quickly does a new regime dominate?	As quickly as it fills the rolling window.	Gradually, as the new regime becomes a larger share of all accepted observations.
Is detection different?	No. Monitoring applies the fitted boundaries to the current KPI.	No. Monitoring applies the fitted boundaries to the current KPI.
Default	Factory default.	Opt-in at system or model level.

How incremental changes your understanding of the data

The most important difference is the time horizon represented by the fitted distribution:

- A **full** model answers: *What has been normal inside this calculation window?*
- An **incremental** model answers: *What has been normal across the accepted history since the last bootstrap, represented by a bounded sample?*

This has several consequences:

- **More history is not the same as guaranteed improvement.** The model has more evidence about recurring valid regimes, but it can also retain unrepresentative incidents if they are allowed into training.
- **There is no recency weighting.** Within each seasonality group, every accepted observation has an equal chance of being represented in the reservoir. Recent data does not automatically receive more weight.
- **Adaptation becomes progressively steadier.** A few new points normally have limited influence on a long-lived model. A sustained change gains influence as more observations are accepted.
- **The model still learns distributions, not labels.** It does not know that a point was a confirmed incident simply because monitoring once classified it as an outlier.
- **Seasonality remains fundamental.** Retention is stratified by seasonality group, so Monday at 09:00 does not crowd out Saturday at 02:00.

What "continuously improve" means

TrackMe continuously refreshes its statistical estimate from a broader retained evidence set. It does not receive correctness labels and does not optimise toward a known answer. Whether the broader history is more useful depends on the KPI and on keeping known bad periods out of the training evidence.

One incremental update

The first incremental run **bootstraps** from the configured calculation window. Later runs can safely search overlapping windows because TrackMe:

1. excludes the newest, potentially incomplete KPI span;
2. deduplicates observations by seasonality group and `_time`;
3. accepts only timestamps newer than the watermark for that group;
4. updates a deterministic Algorithm-R reservoir for every active group;

5. refits all active groups and, with distribution mode `auto`, repeats distribution selection;
6. validates and applies the candidate in memory before atomically replacing the stable model.

An overlap-only run is a **no-op**: it does not refit, write the model, increment the update counter or age seasonality groups. Groups with too little data remain `insufficient_data` and mature automatically as later cycles add observations.

When the model starts over

Incremental memory is valid only while the meaning of the model remains the same. TrackMe computes a configuration signature and bootstraps again when a signature-bearing setting changes, including:

- KPI metric, aggregation method or KPI span;
- seasonality factor;
- calculation earliest or latest boundary;
- algorithm or storage backend;
- density thresholds and distribution parameters;
- period exclusions.

State version, feature or grouping changes also trigger a bootstrap. Corrupt retained state is discarded and safely rebuilt from the current window. The last reset reason is visible in model diagnostics, for example `initial_bootstrap`, `training_signature_changed` or `state_corrupt`.

Note

Changing a signature-bearing option is intentionally a reset, not an incremental update: observations retained under the old interpretation may no longer be comparable with the new one.

Configuration and precedence

The factory default remains `full`. Configure the deployment-wide behavior under **Configuration → System Options → splk-outliers-detection**:

Setting	Factory default	Meaning
`splk_outliers_training_mode_default`	`full`	Effective mode for models whose per-model mode is `inherit`.
`splk_outliers_incremental_reservoir_`	`10000`	Total retained observations per model; accepted range `1000` to `20000`.

Each model has a **Training mode**:

- `inherit` — resolve the current system default at training time;
- `full` — always rebuild this model from the calculation window;
- `incremental` — always retain bounded incremental state for this model.

An explicit model value wins over the system default. Changing the system default therefore changes only inherited models; it never overwrites a model-level exception. Legacy models without a `training_mode` field behave as `inherit`.

Configure it in the UI

Deployment-wide default. Go to **Configuration → System Options → splk-outliers-detection → Algorithms & storage**. Set **Default model training mode** and the **Incremental reservoir size**. The default applies at training time to models whose training mode is **Inherit system default**.



One model. Open **Manage outliers detection**, edit the model, and choose its **Training mode**. The editor displays both the configured choice and the resolved effective mode. For an incrementally trained model it also shows the retained and observed counts, update count, serialized size, and most recent reset reason.



Many entities. From the Tenant Home entity table, select the entities and choose **Bulk edit**. Select **Outliers Detection Rules → Update Training Mode**, then choose **Inherit system default**, **Full baseline**, or **Incremental baseline**. An explicit bulk choice updates the selected entities' outlier rules without changing other entities.



Tip

Start with the system default on `full` and select a small number of representative models explicitly as `incremental` canaries. Enable the system default only after their boundaries and anomaly rates have been observed across several business cycles.

Compatibility and reservoir sizing

Incremental training is supported only for:

- `TrackMeNativeDensityFunction`;
- KV Store model storage.

The legacy MLTK `DensityFunction` remains available for full training, but incremental and MLTK `partial_fit` are intentionally unsupported. Native file-backed models are also full-only. The UI disables an unsupported incremental choice, and REST clients receive an explicit validation error.

The reservoir must hold at least **10 samples for every possible seasonality group**. TrackMe validates the group cardinality against the KPI span before training:

Table 16: Example seasonality budgets

Seasonality	KPI span	Maximum groups	Minimum reservoir
hour of day (`%H`)	`10m`	24	240
weekday × hour (`%w%H`)	`1h`	168	1,680
weekday × hour/minute (`%w%H%M`)	`10m`	1,008	10,080 — above the factory 10,000
weekday × hour/minute (`%w%H%M`)	`5m`	2,016	20,160 — above the V1 maximum

An unsafe explicit `incremental` model is rejected. An inherited model whose system default resolves to incremental falls back safely to full and reports `seasonality_budget_fallback_full`.

Contamination, drift and late corrections

The reservoir contains **accepted training observations**, not observations proven to be clean. This makes operational discipline more important than in a short rolling window:

- Add period exclusions for incidents, maintenance, migrations and known abnormal regimes **before** those timestamps are accepted.
- Confirmed outliers are not automatically removed from the reservoir.
- Reservoir history does not decay automatically.
- Corrections arriving at timestamps already behind a group watermark are intentionally ignored.

To remove contaminated history or incorporate late corrections, add the appropriate exclusion and trigger a clean bootstrap by changing a signature-bearing setting, or switch to full and retrain before enabling incremental again.

If the true behavior has permanently changed, choose deliberately:

- retain incremental mode when old and new regimes are both valid and should coexist;
- bootstrap incremental state when the old regime is no longer meaningful;
- use full mode when normal should follow a recent rolling window quickly.

Safety and failure behavior

TrackMe never deletes the stable model before building its replacement. Incremental training uses a per-model, TTL-protected lock and validates the complete candidate before an atomic KV Store write. Lock contention, ownership loss, fit or boundary failure, corrupt state that cannot be rebuilt, excessive seasonality cardinality, oversized serialization, or persistence failure all leave the **last good model** active and fail the training search visibly.

The retained record is bounded by the configured reservoir and an **8 MiB serialized-record safety ceiling**, below Splunk's 16 MiB KV Store record limit.

Simulation remains stateless: it always performs full training under a simulation-only model key and never reads or changes live incremental state.

Verify a canary

The model editor shows the configured and effective modes together with persistent runtime diagnostics:

```
Effective mode: incremental
Last trained incrementally: 9984 retained / 12808 observed,
updates 2, model size 262952 bytes, reset: initial_bootstrap.
```

A healthy canary satisfies:

- runtime mode is `incremental`;
- observed count is greater than or equal to retained count;
- retained count is at or below the configured reservoir;
- update count increases only when completed, newer observations are accepted;
- model size remains below 8 MiB;
- overlapping history appears as rejected old/duplicate points rather than being counted again.

Use the dedicated `trackmefit` log to distinguish each training action:

```
index=_internal sourcetype=trackme:custom_commands:trackmefit
"Incremental fit result for model"
| rex field=_raw "Incremental fit result for model \"(?<model>[^\"]+)\":␣
↪(?<incremental_json>\{.*\})$"
| spath input=incremental_json
| table _time model action reset_reason accepted_count duplicate_or_old_count
```

(continues on next page)

(continued from previous page)

```
        retained_count total_seen serialized_size preserved_groups pruned_groups
| sort - _time
```

Expected actions are:

- `bootstrap` — initial state or an intentional/reset recovery;
- `incremental_update` — newer points were accepted and a candidate was persisted;
- `no_new_data` — a safe no-op because the searched timestamps were already known.

For a practical qualification, run a bootstrap, a later update with completed new buckets, and an immediate overlapping rerun. The later update should accept new points; the overlapping rerun should report `no_new_data` without changing the persistent update counter.

Rollback to full

Changing the configured mode selects how the **next training run** learns the baseline. To complete a rollback, change the mode and then retrain the affected model or entities:

- **One model:** edit the model, select **Full baseline**, apply the change, and train it.
- **Selected entities:** select them in the Tenant Home entity table, choose **Bulk edit → Outliers Detection Rules → Update Training Mode → Full baseline**, and apply the change. Then use the Outliers bulk-training action for those entities.
- **All inherited models:** set `splk_outliers_training_mode_default = full` and retrain the inherited entities. Models explicitly configured as `incremental` remain incremental until changed individually or through bulk edit.

On each successful retraining, the standard full candidate replaces the incremental model and removes its retained incremental state. Switching that model back to incremental later performs a fresh bootstrap.

See also

- *Outlier detection* — the readable introduction to outlier detection, seasonality, confidence and scoring.
- *Outlier detection — in depth* — model management, backend scheduling, all system and per-model options, and troubleshooting.

8.8.3 Outlier detection — in depth

Tip

This is the **in-depth reference** for outlier detection. For a quick, readable introduction — how it works, seasonality, confidence, and which components benefit — start with *Outlier detection*. This page collects the full detail: the density-function engine, true-context simulation, managing models, the backend, system-wide options, advanced tuning, and troubleshooting. For the purpose, statistical interpretation and rollout of retained baselines, see *Incremental baseline training*.

Machine Learning Outliers Anomaly Detection in TrackMe

TrackMe implements Machine Learning Outliers Anomaly detection across all components, from feeds tracking to the monitoring of scheduled activity in Splunk.

TrackMe implements its own **native ML density function engine** (`TrackMeNativeDensityFunction`), providing a built-in outlier detection system. Models are stored and managed in **per-tenant KVstore collections**, delivering better performance, easier management, and eliminating replication overhead in Search Head Cluster (SHC) environments. The `DensityFunction` algorithm from the Splunk AI Toolkit is also available and can be selected per model. The workflow operates in two phases:

- **ML training** (`mltrain`): models are generated and trained from historical metrics stored in TrackMe's metric store indexes, and persisted to KVstore (or optionally to file)
- **ML monitoring** (`mlmonitor`): TrackMe evaluates the anomaly detection status for each entity by applying trained models

Models are created automatically when entities are discovered and can be customized per entity to adjust behavior as needed.

Impact-Based Alerting and ML Outliers

TrackMe uses **Impact-Based Alerting (IBA)** to determine how ML Outliers affect entity status. When an outlier is detected, TrackMe generates **outlier score events** with a configurable impact score (default: 36). Only when the cumulative impact score reaches the alerting threshold (score >= 100) does the entity transition to a red alert state. This approach provides high flexibility and significantly reduces false positive risks.

- Outlier impact scores are configurable per tenant and per component
- Outliers alone (at default score 36) will not trigger a red alert unless combined with other anomalies or the score is tuned higher
- One-click false positive suppression is available for individual outliers or entire entities
- See *QUICK START - TrackMe Impact-Based Alerting (IBA)* for full details on Impact-Based Alerting

Key capabilities

- **Confidence levels**: ML models have a confidence level (`low` or `normal`) based on available historical data. Models with `low` confidence do not influence entity status until sufficient data has been collected (configurable, default: 30 days)
- **Seasonality control**: Models can include time-based seasonality factors or set the `time_factor` to `none` to exclude seasonal concepts for KPIs that are not driven by time-of-day or day-of-week patterns
- **Threshold guards**: Per-model minimal and maximal thresholds for LowerBound and UpperBound outliers allow rejecting outliers that do not meet defined value boundaries
- **Auto-correction**: Built-in deviation checks reject outliers where the variation is too small relative to the current KPI value, reducing false positives
- **True context simulation**: Simulation mode trains a dedicated simulation model that fully replicates live model behavior, ensuring accurate previews before applying changes
- **Native ML engine**: TrackMe's built-in `TrackMeNativeDensityFunction` engine uses scipy with automatic best-fit distribution selection (normal, exponential, Gaussian KDE, beta), with the MLTK `DensityFunction` also available
- **KVstore model storage**: ML models are stored in per-tenant KVstore collections by default, with file-based storage available as an alternative
- **Full or incremental training**: Keep the factory-default rolling-window rebuild, or opt native KV Store models into a bounded, seasonality-aware retained baseline at system or model level
- **Custom algorithms and parameters**: Define custom algorithms, extra `fit`/`apply` parameters, and custom boundaries extraction macros for advanced ML requirements
- **Selective enablement**: ML Outliers can be enabled or disabled per tenant and per component using the `mloutliers` and `mloutliers_allowlist` configuration options
- **Bulk operations**: Reset, enable/disable, train, and monitor operations can be performed in bulk via the UI

See also: *Use TrackMe to detect abnormal events count drop in Splunk feeds* for a practical use case around Machine Learning outlier detection.

Seasonality Concepts

Most data sources exhibit recurring patterns — higher activity during business hours, lower volumes on weekends, periodic batch ingestion cycles. TrackMe's ML Outliers engine leverages these patterns by training models that learn the expected behavior over time and flag deviations as potential anomalies.

Sample pattern over the past 30 days — seasonality by weekday with higher activity during working hours:





However, not all KPIs follow seasonal patterns. For metrics with steady, time-independent behavior (for example, a fixed-rate feed or a constant host count), applying seasonality to the model can introduce unnecessary noise. To handle these cases, ML models can be configured with the `time_factor` set to `none`, which excludes day-of-week and hour-of-day seasonality from the outlier calculations. This can be set on a per-model basis, or chosen as the default for new models via the `splk_outliers_detection_timefactor_default` configuration option.



Note

Generating samples for outliers detection

You can find, download, and use the following sample generator with no restrictions: https://github.com/trackme-limited/mlgen-python — we use it to generate data with seasonality concepts for development, qualification, and documentation purposes.

Confidence Level

TrackMe assigns a **confidence level** to each ML model during training, based on the amount of historical data available:

- **low**: The model is trained and rendered, but outlier results **do not influence entity status**. This prevents false positives while the model is still learning from limited data.
- **normal**: The model is fully trusted and outlier results **contribute to the entity's impact score**.

The minimum number of days of historical metrics required to reach `normal` confidence is controlled by the `splk_outliers_min_days_history` configuration option (default: 30 days). The confidence level and reason are visible in the **Manage Outliers screen** and stored in the rules KVstore collection.





Outliers Impact Score & triggering on Upper Bound/Lower Bound breaches

When TrackMe's ML engine detects an outlier, it generates **outlier score events** that feed into the entity's cumulative impact score. Whether an outlier actually influences the entity status depends on several factors:

Upper Bound and Lower Bound breach configuration:

Each ML model defines which breach directions are active:

- `alert_lower_breached`: When enabled, a **LowerBound** outlier (observed value falls below the predicted lower boundary) generates a score event
- `alert_upper_breached`: When enabled, an **UpperBound** outlier (observed value exceeds the predicted upper boundary) generates a score event

These settings are configured per model, giving you full control over which types of volume deviations are meaningful for each metric. For example, you may want to alert only on lower bound breaches for a critical feed (detecting data

drops) while ignoring upper bound breaches that represent harmless spikes.

Impact score behavior:

- The default outlier impact score is **36** (configurable per tenant and per component)
- At the default score, a single outlier detection alone will not trigger a red alert (which requires a cumulative score >= 100)
- Multiple concurrent outlier detections, or outliers combined with other anomalies (delay breach, latency breach, etc.), can accumulate to reach the alerting threshold
- Outlier score events are aggregated over a **24-hour rolling window** — older scores automatically expire

Threshold guards:

Per-model minimal and maximal value thresholds provide an additional layer of filtering:

- `min_value_for_lowerbound_breached`: If the observed value remains above this minimum threshold, the LowerBound breach is considered insignificant and rejected
- `min_value_for_upperbound_breached`: If the observed value remains below this minimum threshold, the UpperBound breach is considered insignificant and rejected

This filters out insignificant deviations — for example, a small dip in event count that technically breaches the lower boundary but represents a negligible variation. Rejected outliers do not generate score events and do not affect the entity's impact score.

False positive suppression:

When an outlier is identified as a false positive, a one-click action generates a **negative score event** that cancels out the outlier's positive score. The anomaly reason is preserved for audit purposes but no longer contributes to the entity status. See *QUICK START - TrackMe Impact-Based Alerting (IBA)* for details on false positive management.

Outliers impact score and breach configuration in the Manage Outliers screen:





Outliers per component

ML Outliers detection is available across TrackMe components, but the value and applicability varies significantly depending on the nature of the data being monitored.

Enabling Outliers at tenant creation

When creating a new Virtual Tenant, the **Advanced Options** section includes the Outliers enablement setting. TrackMe automatically pre-selects the recommended value based on the component type:

- **flx** tenants: Outliers is **enabled** by default
- **dsm** tenants: Outliers is **disabled** by default — for volume anomalies we recommend the **vol** (Volume Outliers) component in a separate tenant, available in all editions, which models the licensed volume of every index and keeps outliers detection separate from the feed monitoring of DSM. You can still enable Outliers on the DSM tenant if you want per-feed models as well
- **vol** tenants: Outliers is **always enabled** — the outliers are the state of the Volume Outliers component; the wizard sets the VOL scope (priorities, minimum history, default KPIs) instead of an on/off switch
- **dhm**, **wlk**, and **fqm** tenants: Outliers is **disabled** by default
- **mhm** tenants: Outliers is not applicable

This setting can be changed at any time after creation from the **Virtual Tenants account** configuration page, allowing you to enable or disable Outliers detection as your requirements evolve.

Applicability per component

Table 17: ML Outliers applicability per component

Component	Recommendation	Details
dsm (Data Sources)	Optional (off by default)	Can provide volume-based anomaly detection per data feed, but disabled by default at tenant creation. The recommended approach is the **vol** (Volume Outliers) component, available in all editions — including **Foundation Edition** — in a separate tenant: it detects volume anomalies at the index level and keeps outliers detection separate from the delay, latency and availability monitoring of DSM. Enable Outliers on DSM only if you also want per-feed models.
vol (Volume Outliers)	Core feature	Outliers on the rolling licensed volume of every index (drops and / or spikes, chosen per index) **are** the state of the component. Every new index gets one model per default KPI (the rolling 24 hours by default, `vol_outliers_kpis`); the history is backfilled at tenant creation so the models train on day one; VOL has its own scope (`vol_mloutliers_priority_filter`, every priority by default, and `vol_mloutliers_filter_expression`) and its own minimum-history override. See *Outliers*.
dhm (Data Hosts)	Not recommended	Host-level metrics tend to have low predictability and stability due to the dynamic nature of endpoints. Outliers detection on hosts typically generates excessive false positives.
mhm (Metric Hosts)	Not applicable	Outliers detection is not applicable to metric hosts.
wlk (Workload)	Not recommended	Workload monitoring generally has low predictability and limited value from outliers detection. TrackMe automatically disables Outliers by default when creating a Workload tenant.
flx (Flex Objects)	Key feature	Outliers is a core capability of Flex Objects, enabling precise anomaly detection on custom metrics with full control over model configuration. Can be disabled at the tenant level for use cases where outliers detection is not applicable (no metrics, or simply not wanted).
fqm (Fields Quality)	Not recommended	Fields quality metrics are generally not suitable for outliers detection due to their inherent variability.

Hint

The `mloutliers_allowlist` configuration option controls which components have ML Outliers enabled per tenant. By default, all applicable components are included (`dsm,dhm,flx,wlk,fqm,vol`). Adjust this list to match your environment — for example, set it to `flx,vol` to focus on the highest-value components. A tenant that only runs DHM / MHM (where ML is off at creation) keeps ML on for VOL alone (`vol`) when the Volume Outliers component is added.

Enabling/disabling Anomaly Outliers at the tenant level during tenant creation:



Enabling/disabling Anomaly Outliers at the tenant level from the Virtual Tenants account configuration page:





Flex Objects and Outliers

For **Flex Objects (flx)**, ML Outliers models are primarily driven by the tracker configuration. When creating or editing a Flex tracker through the UI, the **Outliers Metrics (ML)** section allows you to define outliers models directly as part of the tracker setup:

- Select which metrics should have outliers detection enabled
- Configure per-metric parameters at creation time: impact score, breach directions, density thresholds, time factor, auto-correction, period calculation, and static bounds
- These settings are applied automatically when entities are discovered by the tracker, creating pre-configured ML models without manual intervention

This approach is a key advantage of Flex Objects — the outliers configuration is part of the tracker definition itself, ensuring consistent model parameters across all entities discovered by that tracker.

For other components (`dsm`, `dhm`, etc.), ML models are created with system-wide defaults at entity discovery and must be tuned individually or via bulk actions afterward.



Hint

When a Flex tracker defines `outliers_metrics` in its search results, the ML models are automatically created with the specified parameters. This can also be set programmatically via the `outliers_metrics` field in the tracker search output — see *FLX — Flex Objects* for details.

Demonstrating Machine Learning Outliers Detection in TrackMe

How ML Outliers Detection Works in TrackMe

In short:

- Outliers rely on TrackMe-generated metrics only
- This allows running fast and efficient training and rendering searches, with minimal cost in terms of resources

For the purpose of this demonstration, we create a Flex Object TrackMe tenant that uses our ML generator:

- For more information about the Flex Objects component: *FLX — Flex Objects*

Our Flex tracker:

- Tracker name: "demo"
- Runs every 5 minutes (earliest: -5m, latest: now)

```
index=mlgen ref=* instance_id=*
| stats avg(dcount_hosts) as dcount_hosts, avg(events_count) as events_count,␣
↪values(instance_id) as instance_id by ref

| eval group = "demo"
| eval object = instance_id . ":" . ref, alias = ref
| eval object_description = "Demo Outliers in TrackMe"
| eval metrics = "{'dcount_hosts': " . dcount_hosts . ", 'events_count': " .␣
↪events_count . "}"
| eval outliers_metrics="{'dcount_hosts': {'alert_lower_breached': 1,
↪'alert_upper_breached': 1, 'time_factor': '%H', 'period_calculation': '-90d'},
↪'events_count': {'alert_lower_breached': 1, 'alert_upper_breached': 1,
```

(continues on next page)

(continued from previous page)

```
↪'time_factor': '%H', 'period_calculation': '-90d'}}"
| eval status=1
| eval status_description="Machine Learning Outliers detection demo"
| table group, object, alias, object_description, metrics, outliers_metrics,␣
↪status, status_description
``` default metric for the TrackMe UI to pick when opening the entity screen ```
| eval default_metric="events_count"
``` alert if inactive for more than 3600 sec```
| eval max_sec_inactive=3600
```

This Flex Tracker creates entities by monitoring the availability of data in our ML index; it also generates metrics and automates the definition of models which alert on both lower bound and upper bound outliers.

We have our ML generator running and having backfilled the past 90 days of data; it currently does not generate any outliers:

```
index=mlgen ref="security:linux_secure" earliest=-90d
| timechart avg(events_count) as avg_events_count span=1h
```



Our ML generator takes into account the weekdays; we can use the following search to compare the relative activity of the current weekdays against the past 4 previous same weekdays:

```
index=mlgen ref="security:linux_secure" earliest=@d latest=+1d@d
| timechart span=5m avg(events_count) as events_count_today
| appendcols [
search index=mlgen ref="security:linux_secure" earliest=-7d@d latest=-6d@d
| timechart span=5m avg(events_count) as events_count_ref
]
| appendcols [
search index=mlgen ref="security:linux_secure" earliest=-14d@d latest=-13d@d
| timechart span=5m avg(events_count) as events_count_ref2
]
| appendcols [
search index=mlgen ref="security:linux_secure" earliest=-21d@d latest=-20d@d
| timechart span=5m avg(events_count) as events_count_ref3
]
```

Results:



TrackMe automatically discovered the entity, let's take note of its internal identifier which we will use to manually backfill TrackMe metrics, as if we had been monitoring this entity since the beginning:





We use mcollect to force backfilling metrics, pay attention to replace with the valid tenant_id value:

```
index=mlgen ref=* instance_id=* earliest=-90d latest=now
| bucket _time span=5m
| stats avg(dcount_hosts) as trackme.splk.flx.dcount_hosts, avg(events_count) as␣
↪trackme.splk.flx.events_count by _time, instance_id, ref

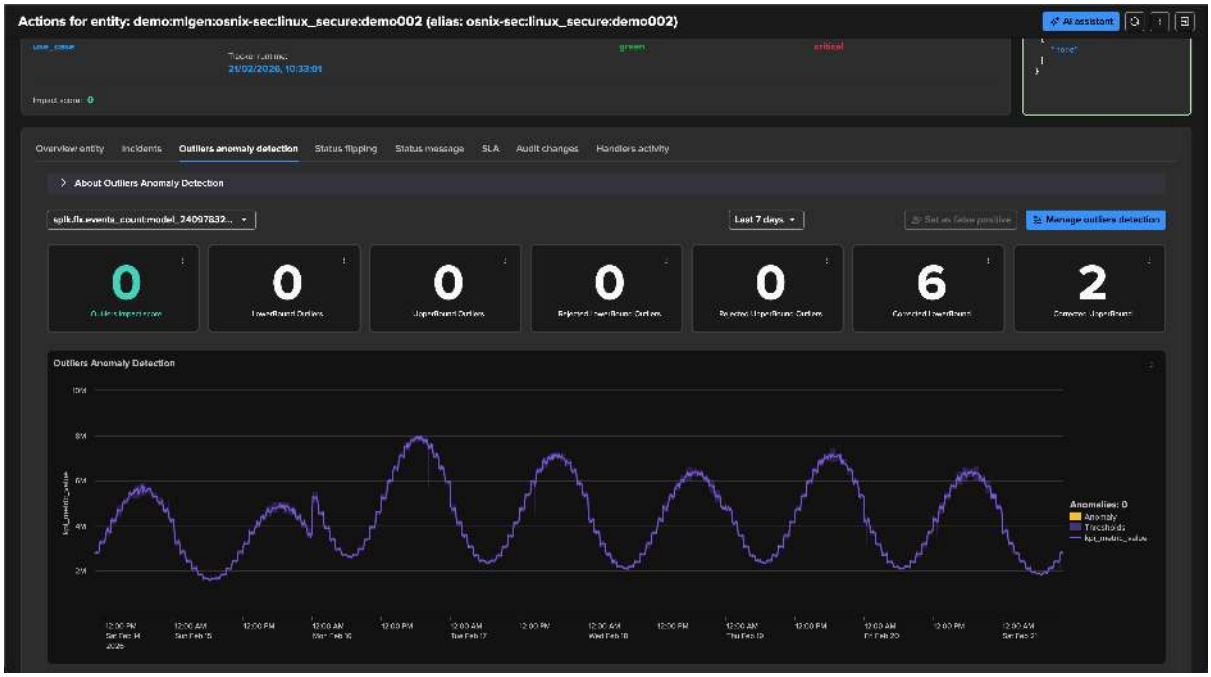
``` form the object ```
| eval object = "demo" . ":" . instance_id . ":" . ref

``` lookup the tenant/component ```
| lookup trackme_flx_tenant_demo-outliers object OUTPUT tenant_id, _key as␣
↪object_id, object_category
| where isnotnull(object_id)

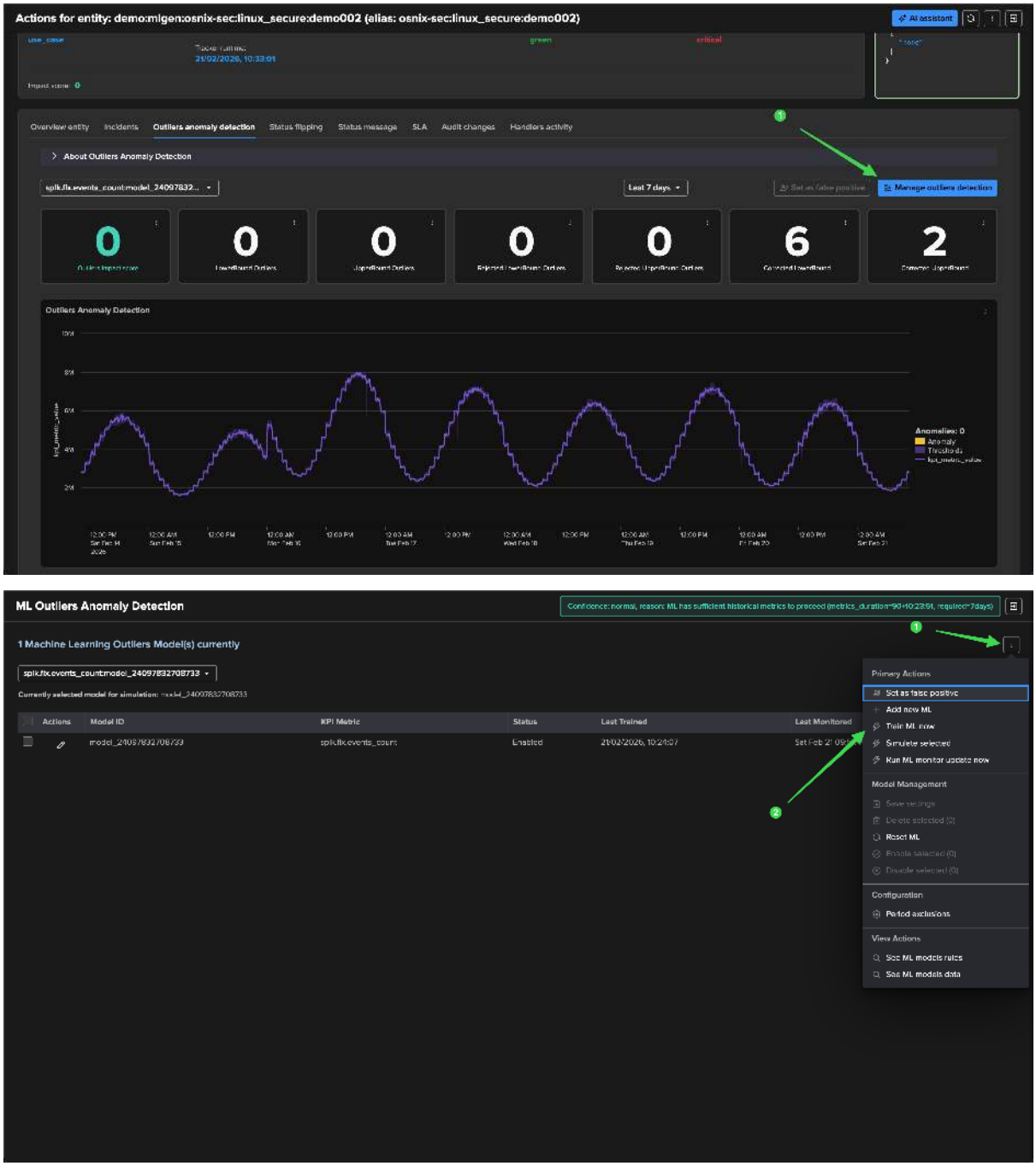
``` collect ```
| mcollect index=trackme_metrics split=t object, object_category, object_id,␣
↪tenant_id
```

Opening the entity shows we have backfilled metrics now:
```



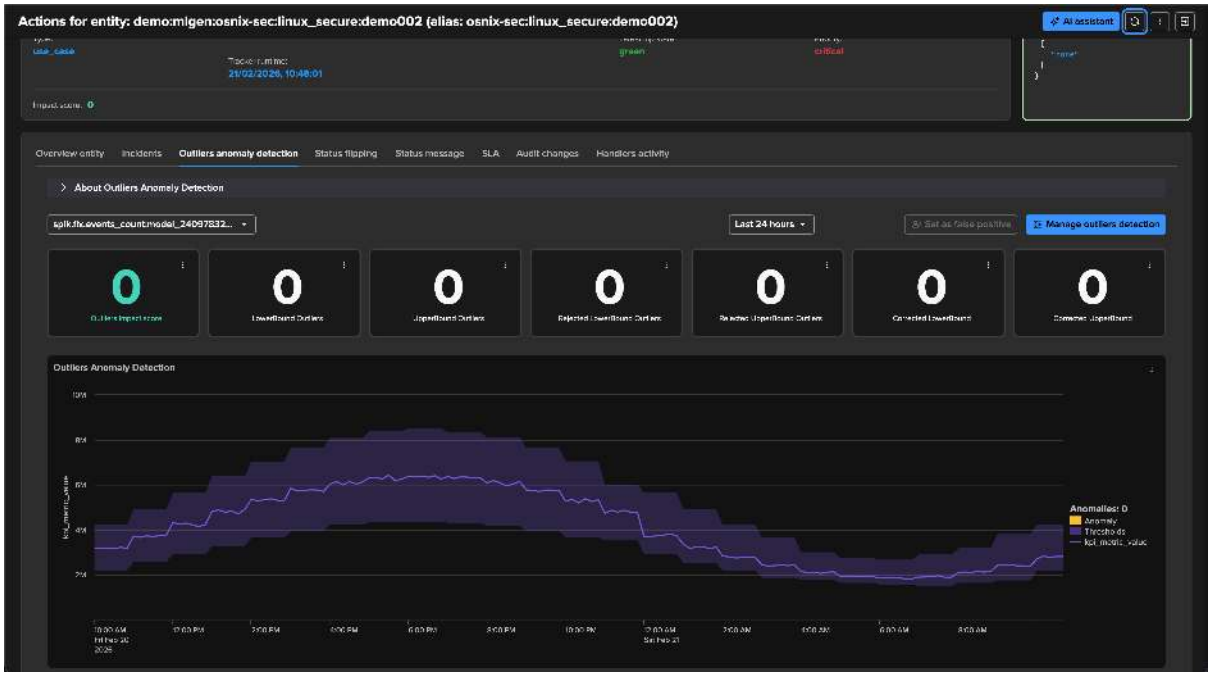
Depending on if TrackMe already ran or not the ML training job for the tenant, ML Outliers may not be ready yet:

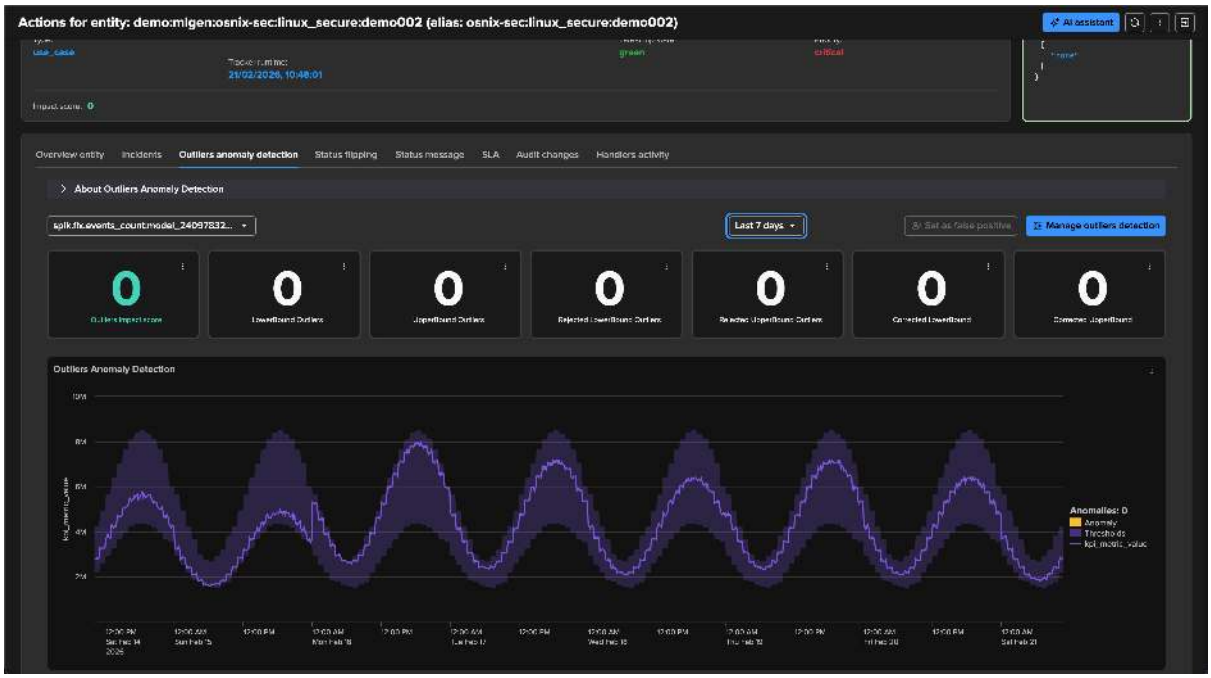


We can either run the mltrain job manually, or train the models via the UI:



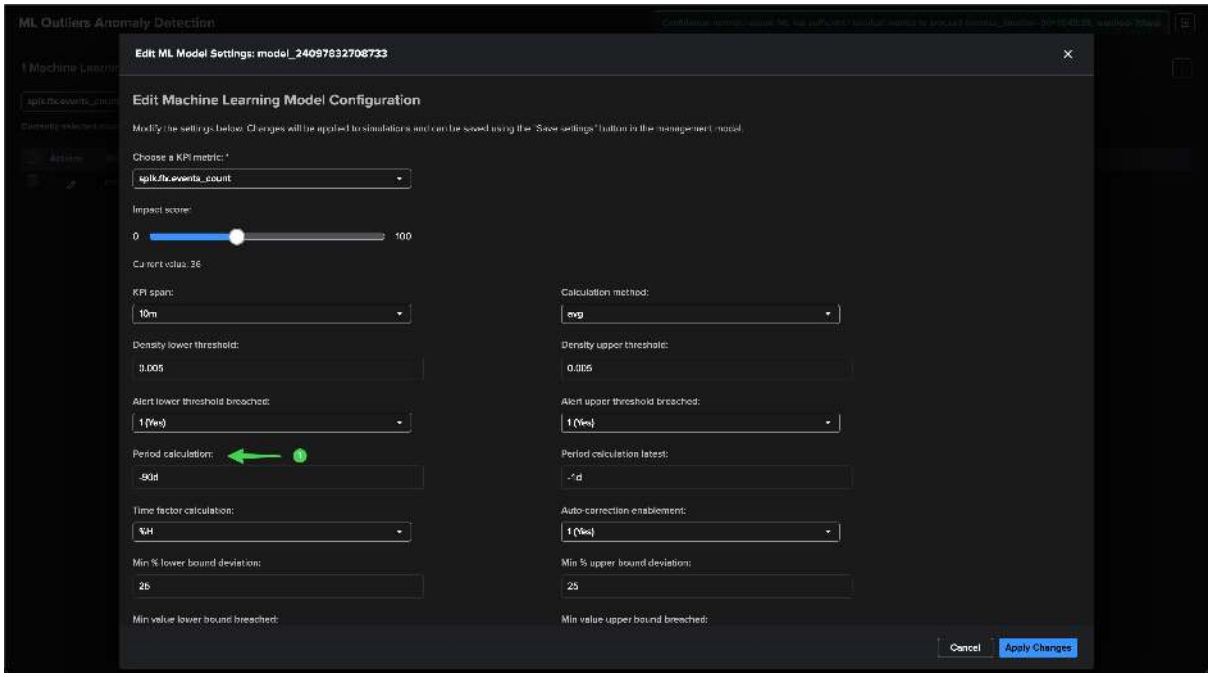
Machine Learning Outliers is ready: (we have no outliers yet)

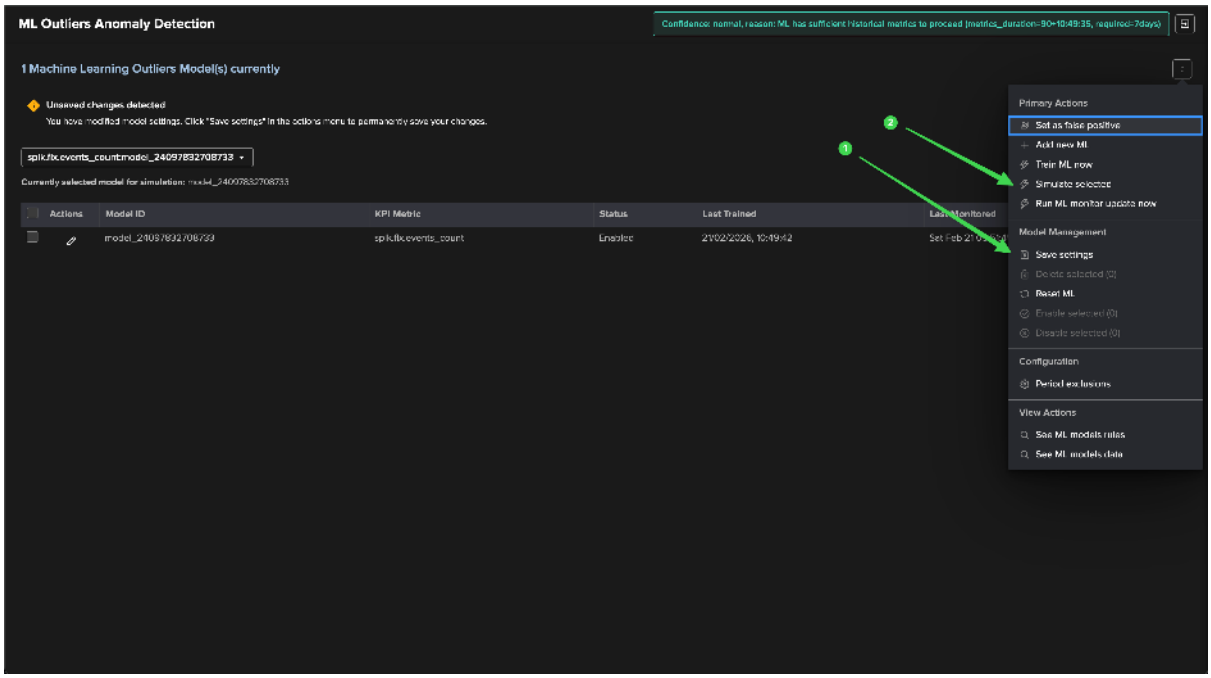


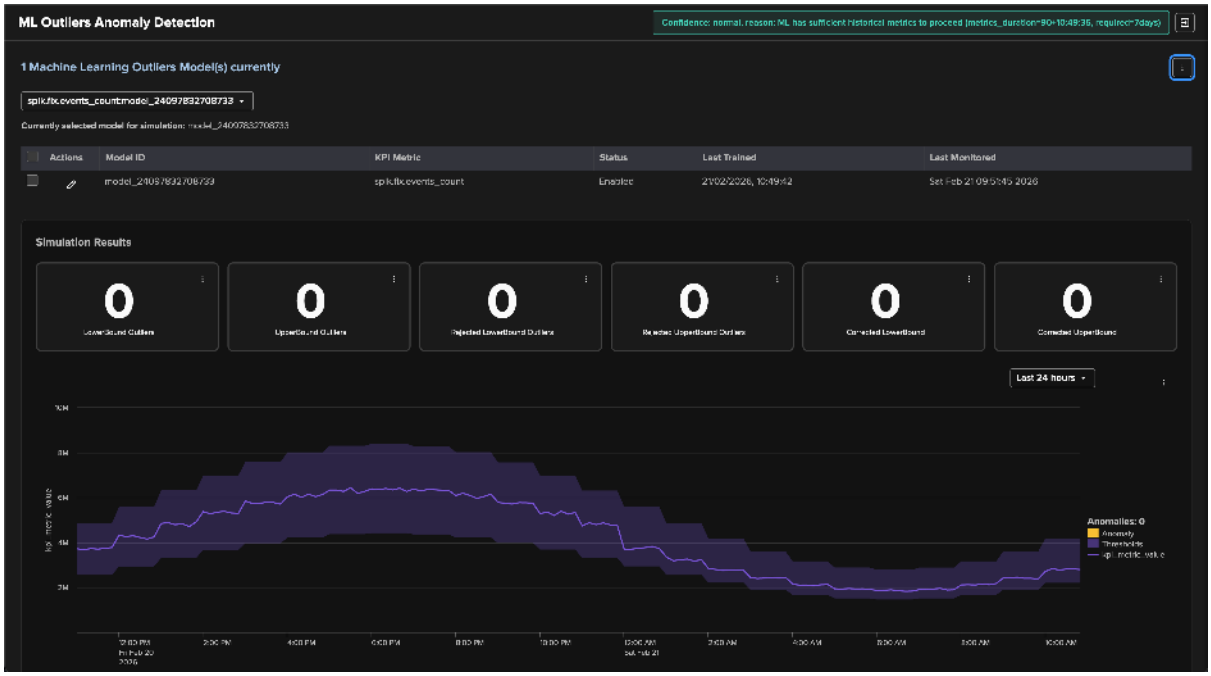


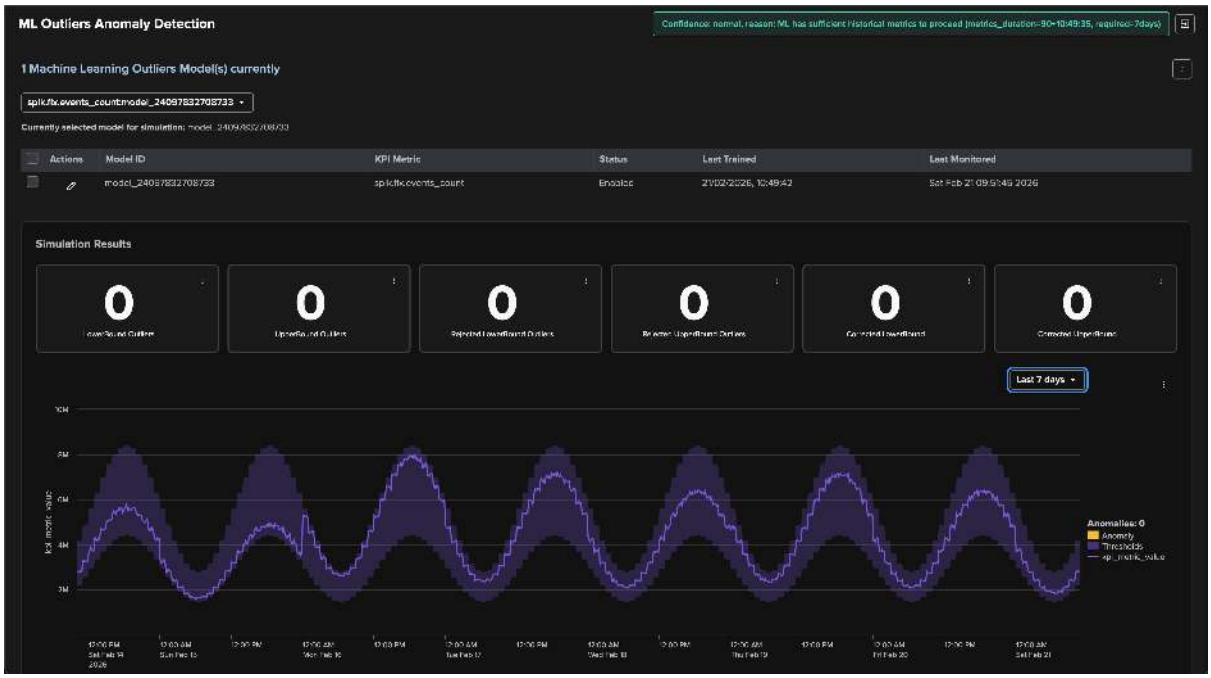
Let's access and review the models definition, for now we will only increase the training period to the past 90 days:

- Click on "Manage Outliers detection"
- Update the models to increase the time range for the calculation
- Manually run a training for each model
- Click on Simulate Selected to review the results (we have selected the event count model), this is looking great for now


Edit ML Model Settings: model_24097832708733
Edit Machine Learning Model Configuration
Choose a KPI metric: *
spk.fx.events_count
Impact score:
0
100
Current value: 36
KPI span:
10m
Calculation method:
avg
Density lower threshold:
0.005
Density upper threshold:
0.005
Alert lower threshold breached:
1 (Yes)
Alert upper threshold breached:
1 (Yes)
Period calculation:
-90d
Period calculation latest:
-7d
Time factor calculation:
%H
Auto-correction enablement:
1 (Yes)
Min % lower bound deviation:
25
Min % upper bound deviation:
25
Min value lower bound breached:
Min value upper bound breached:
Cancel
Apply Changes


ML Outliers Anomaly Detection
1 Machine Learning Outliers Model(s) currently
Unsaved changes detected
You have modified model settings. Click "Save settings" in the actions menu to permanently save your changes.
spk.fx.events_count:model_24097832708733
Actions
Model ID
KPI Metric
Status
Last Trained
Primary Actions
Set as false positive
Add new ML
Train ML now
Simulate selected
Run ML monitor update now
Model Management
Save settings
Reset ML
Configuration
Period exclusions
View Actions
See ML models rules
See ML models data





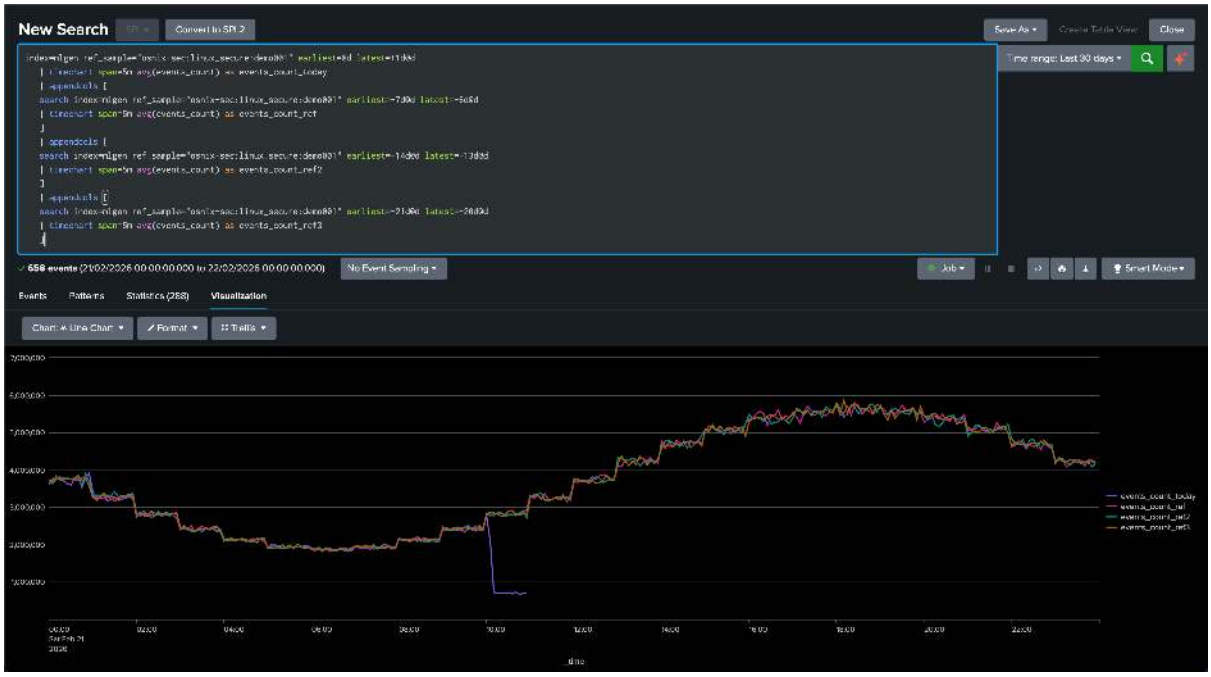
Scenario: Detecting a lower bound outlier

Although we know there is a weekday behavior in the data, for now we will stick with the default settings and we will start generating a lower bound outlier.

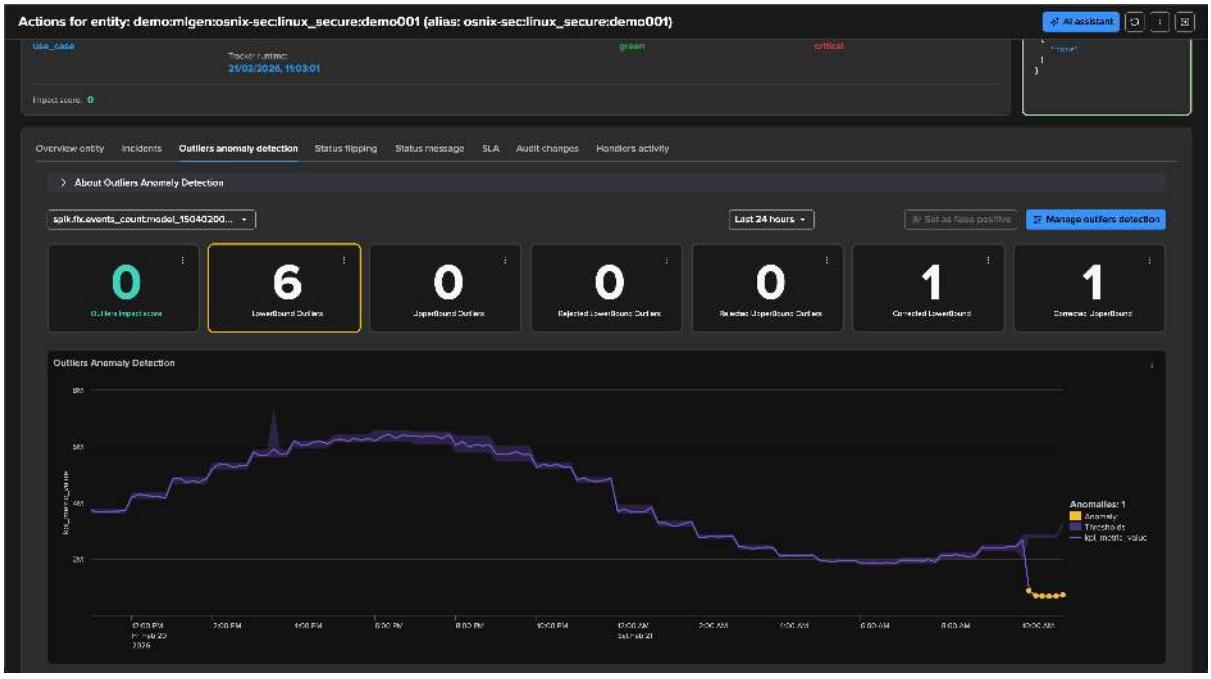
To achieve this, we stop the `run_backfill.sh` and we start `run_gen_lowerbound_outlier.sh`, this basically:

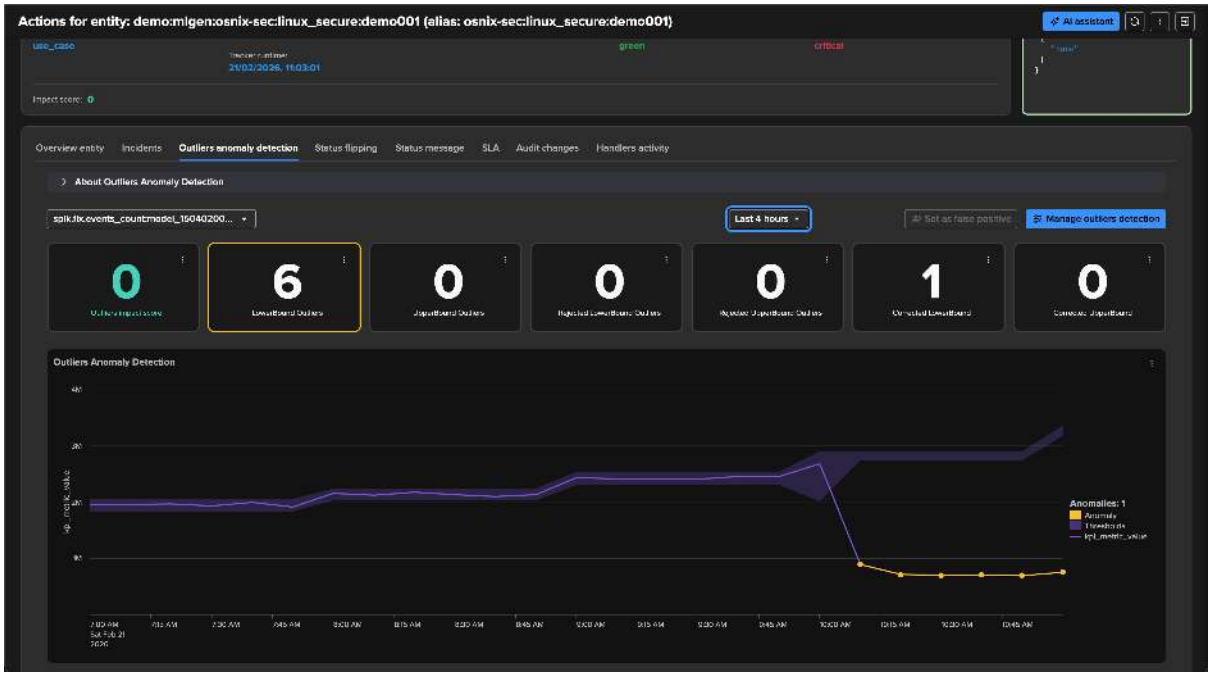
- Influences metrics with a large decrease of approximately 75%, according to the magnitude of the weekday/hour range

After a few minutes, we start to see a clear outlier using the previous days' comparison timechart search:



The outliers condition will also be reflected in TrackMe. It can take 5 to 10 minutes to be detected as an effective outlier:

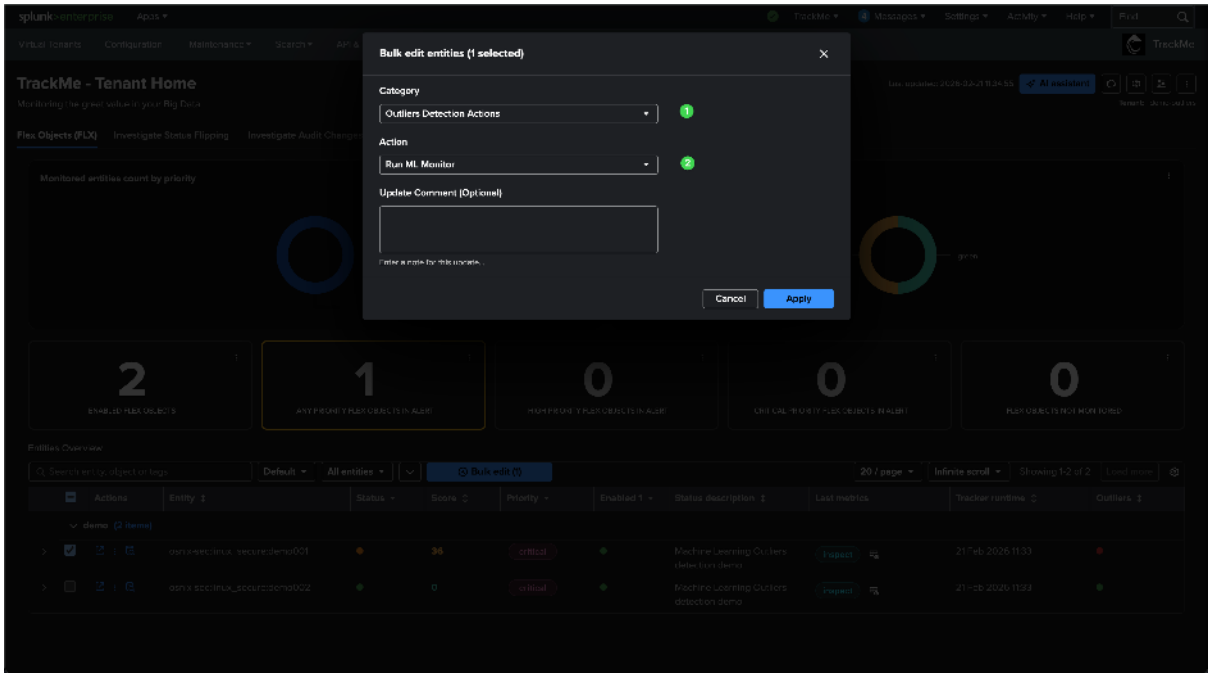


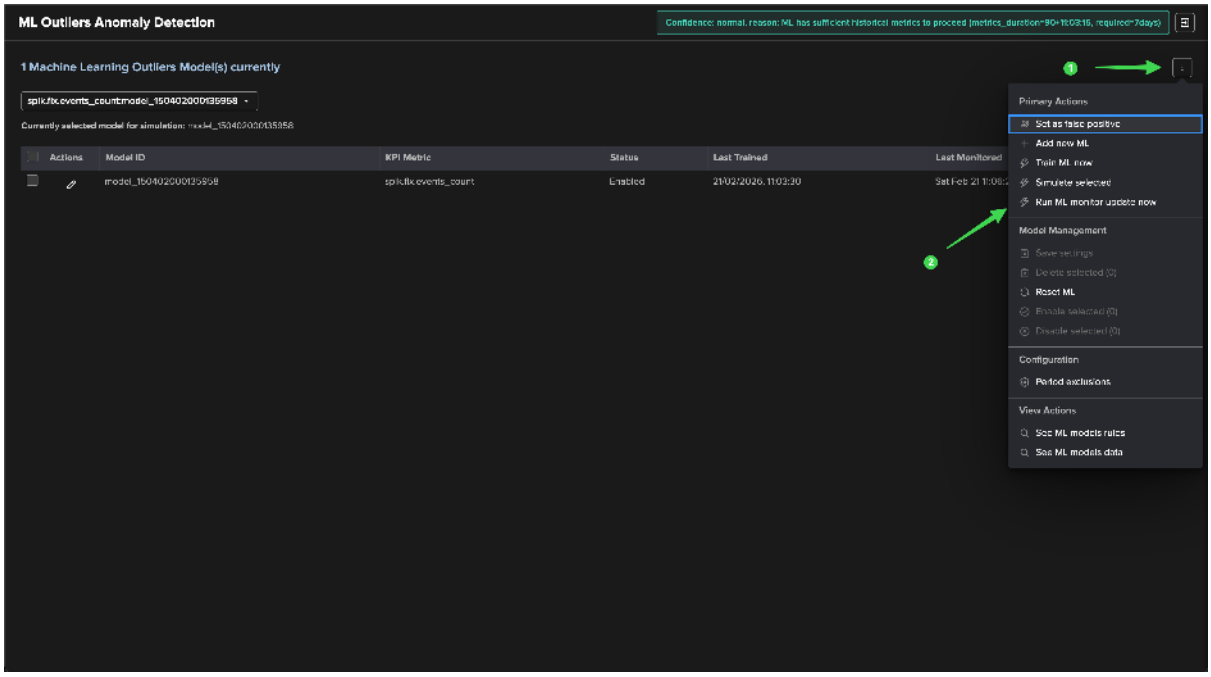


Excellent—the sudden decrease in activity has been detected successfully!

The next action is to run the ML rendering process, which can be achieved in different ways:

- Through the bulk actions for Outliers Detection
- Through the Outliers management screen

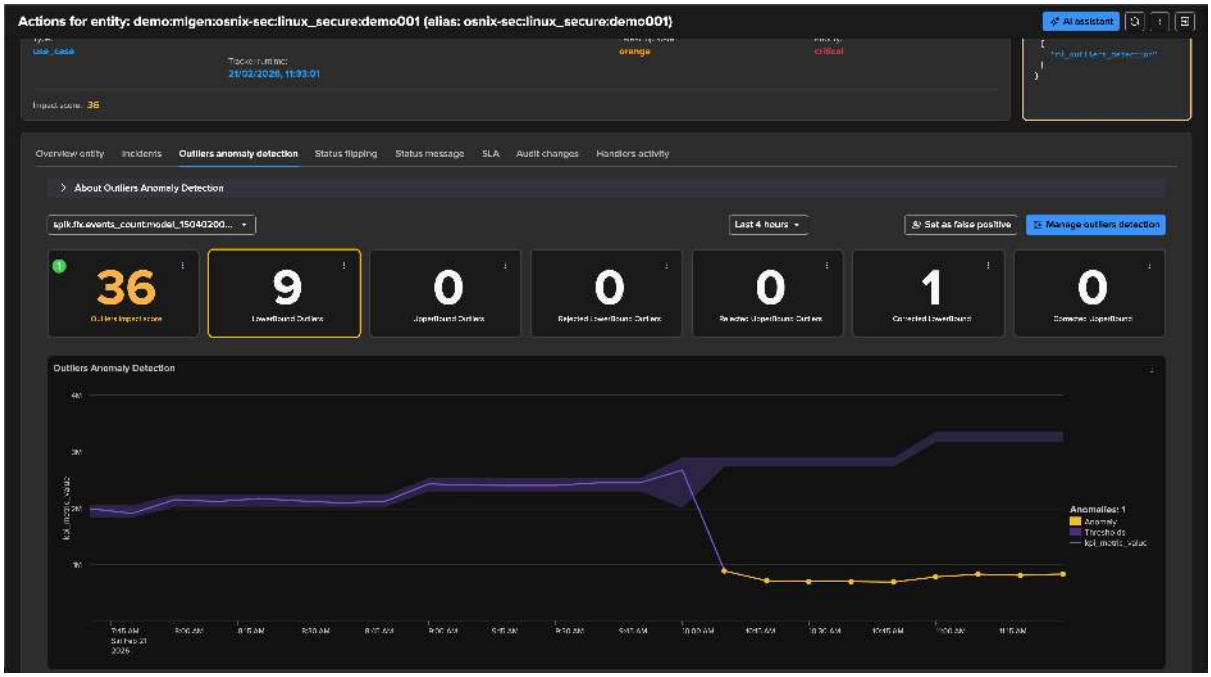


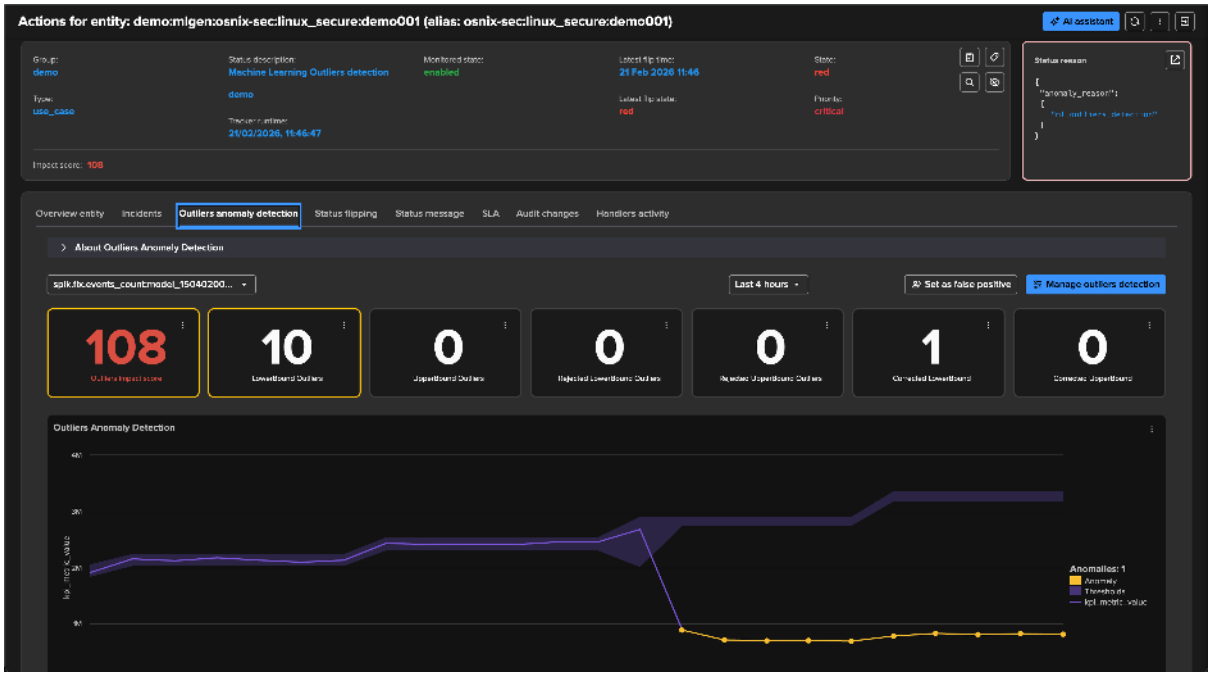


Note

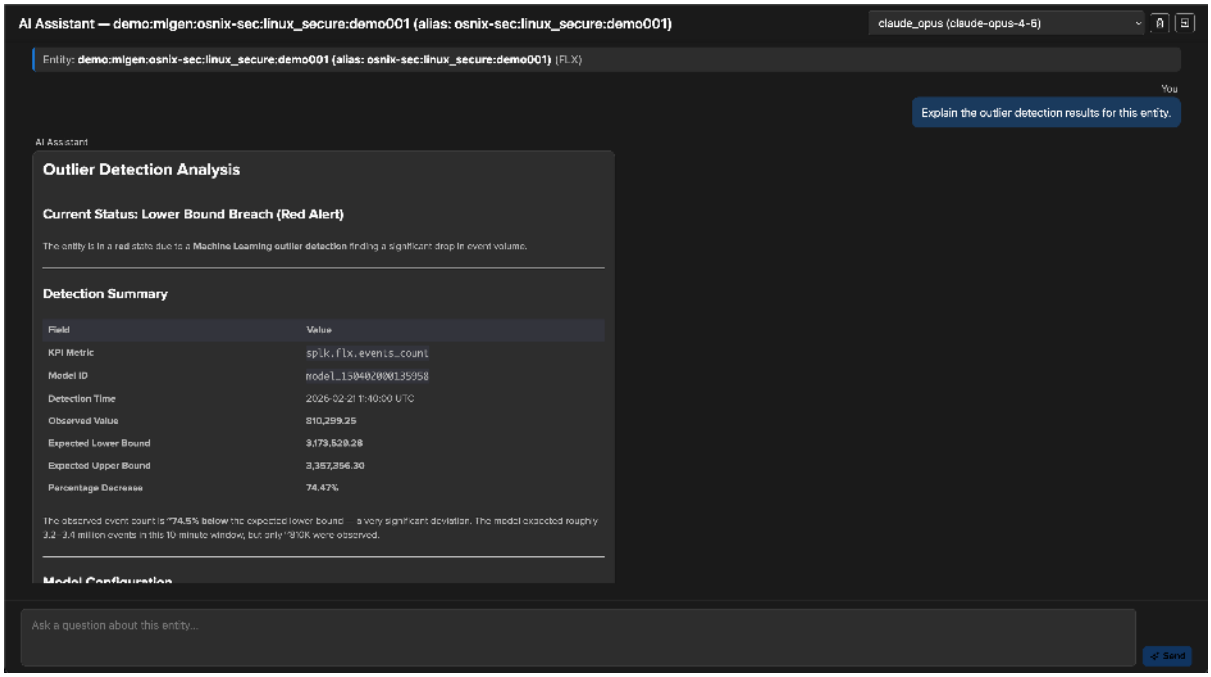
TrackMe monitors every entity's models by default once per hour at the earliest, which is configurable through the system-wide options: *System-wide configuration options*

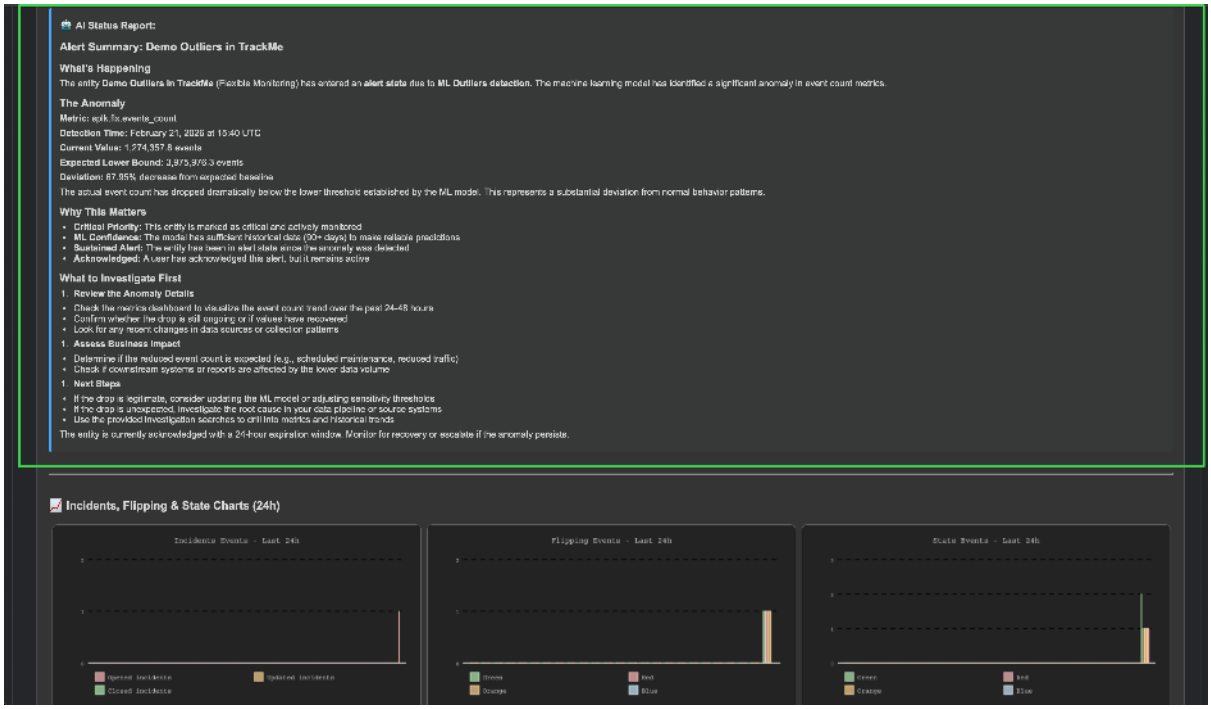
At some point, the Outliers condition is pertinent enough to generate a sufficient impact score to trigger a red alert:





If an AI provider is configured, the AI Assistant can be leveraged to investigate the Outliers condition and generate an AI status report in stateful email notifications:



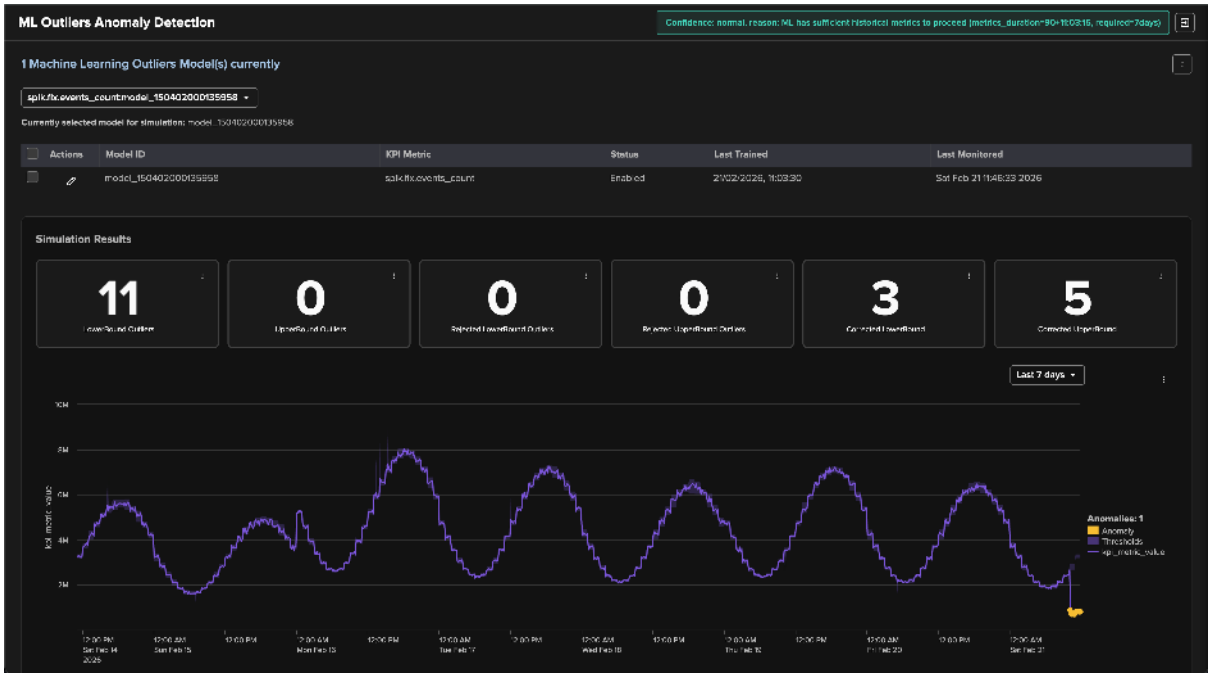


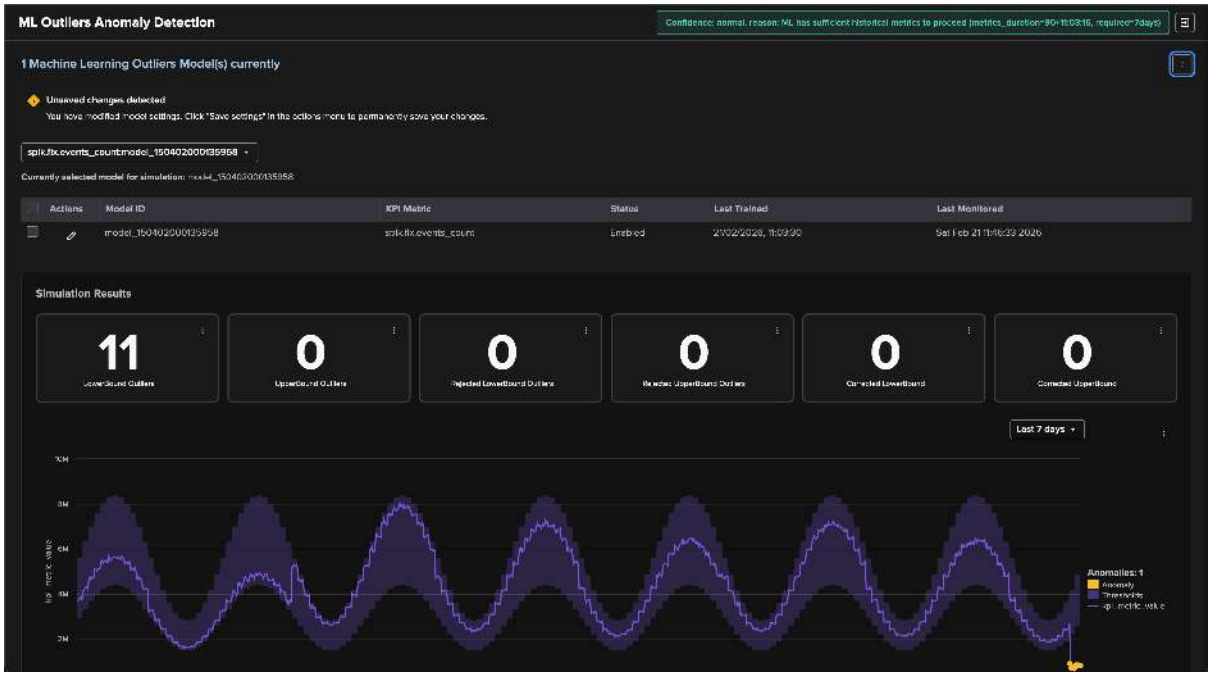
Fine tuning the ML models

True context simulation

TrackMe performs true context simulations — when a simulation is executed, TrackMe trains a dedicated model specifically for that simulation, allowing the outlier detection to fully reflect live behavior:

- All parameters, whether saved or not yet saved, are taken into account on the fly during the simulation
- You can add specific conditions such as periods of exclusion



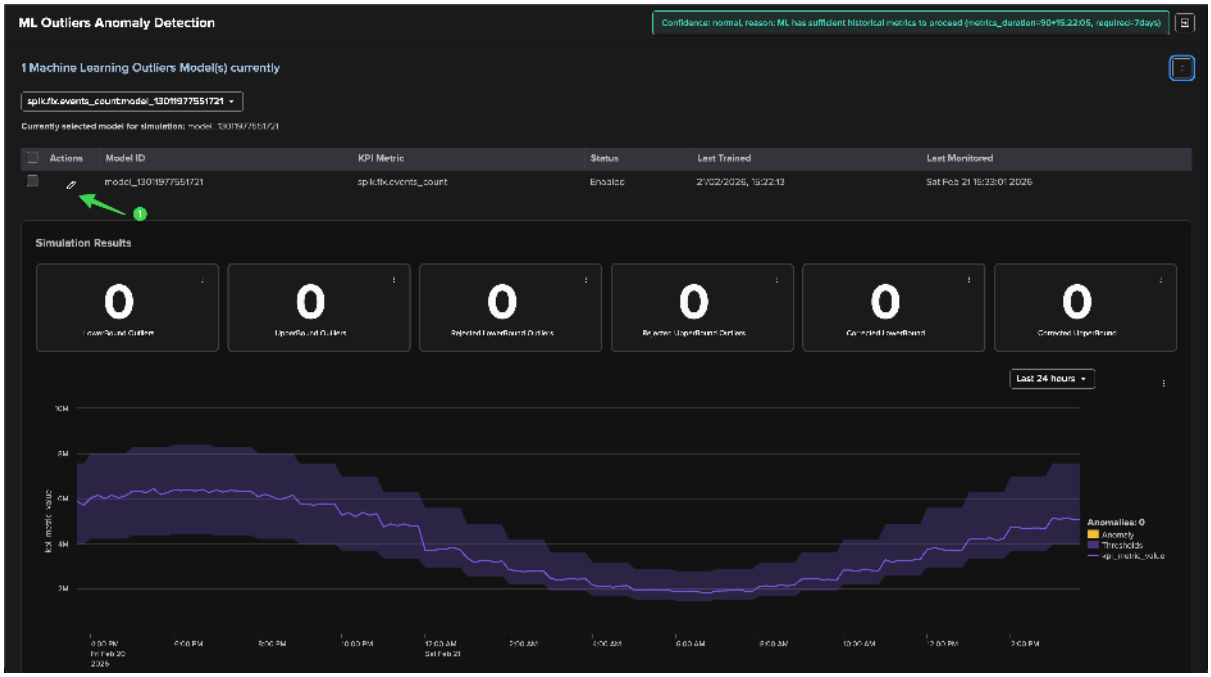


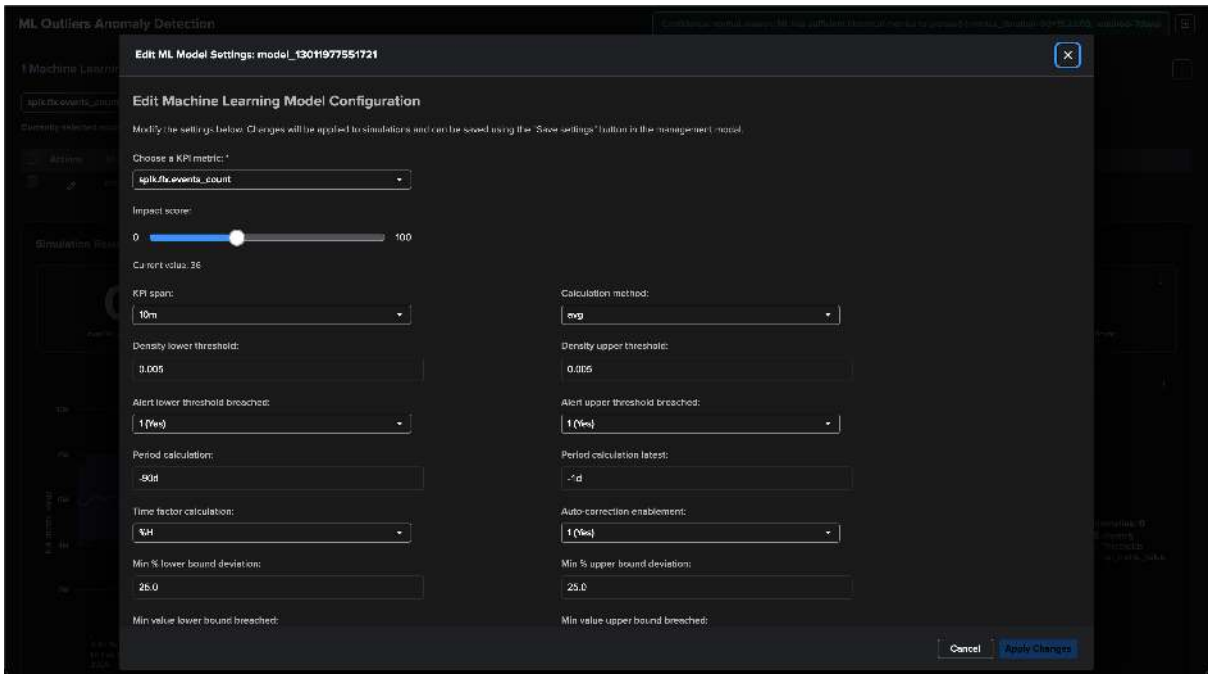
Adjusting model parameters

From the **Manage Outliers detection** screen, each model can be individually tuned to match the behavior of the underlying data. The key parameters are:

- **Outlier impact score** (`outlier_impact_score`): The score value generated when an outlier is detected. Default: 36. A single outlier at the default score will not trigger a red alert on its own (cumulative score >= 100 is required). Increase the value to make outlier detections more impactful, or decrease it to reduce their weight.
- **Training period** (`period_calculation`): Controls how far back in time the model looks during training. Longer periods capture more historical patterns but may dilute recent trends. Shorter periods react faster to changes.
- **Time factor** (`time_factor`): Determines the seasonality granularity. Common values include `%w%H` (weekday + hour), `%H` (hour only), `%w` (weekday only), or `none` to disable seasonality entirely.
- **Density thresholds** (`density_lowerthreshold` / `density_upperthreshold`): Control the sensitivity of the density function algorithm (both `TrackMeNativeDensityFunction` and `DensityFunction`). Lower values produce tighter boundaries (more sensitive), higher values produce wider boundaries (more tolerant).
- **Auto-correction** (`auto_correct`): When enabled, outliers where the deviation is below a configurable percentage of the current KPI value are automatically rejected as insignificant. The deviation thresholds are controlled by `perc_min_lowerbound_deviation` and `perc_min_upperbound_deviation`.

All changes can be previewed using the **Simulate** function before saving — see *True context simulation* above.





Excluding periods from training

When an incident or planned maintenance causes abnormal data patterns, the affected time period can be excluded from ML model training to prevent the model from learning incorrect baselines.

How to use:

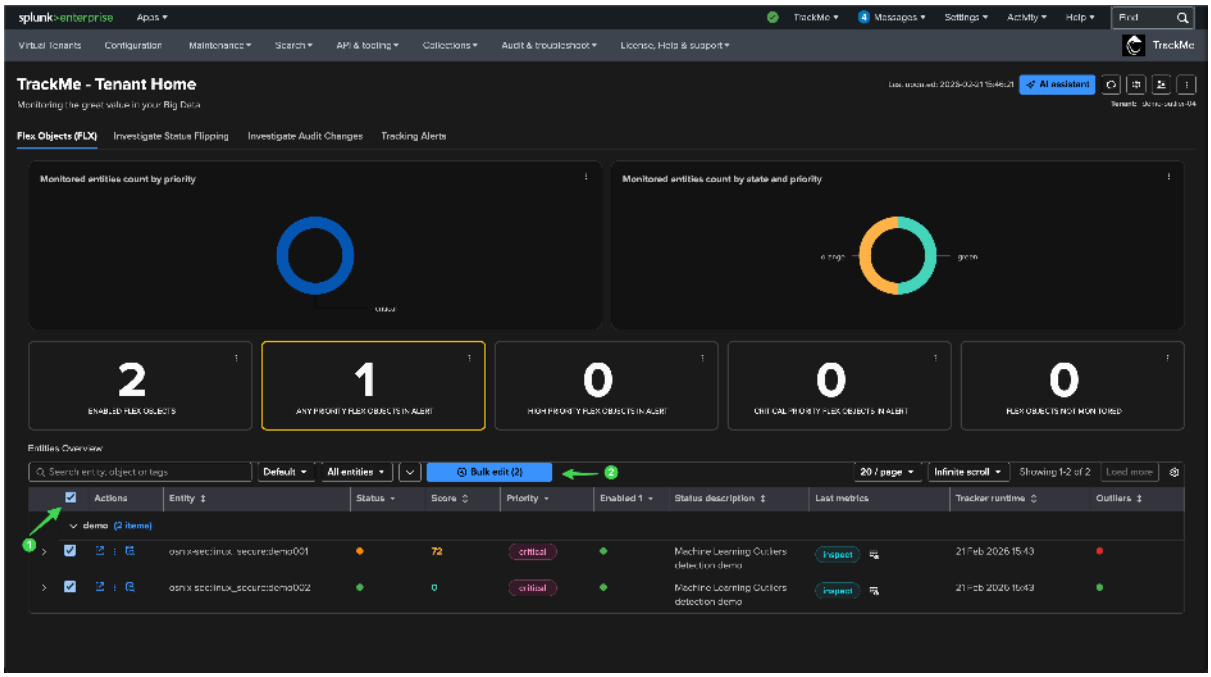
- Open the **Manage Outliers detection** screen for the entity
- Click **Period Exclusions** on the target model
- Define the start and end time for the exclusion window

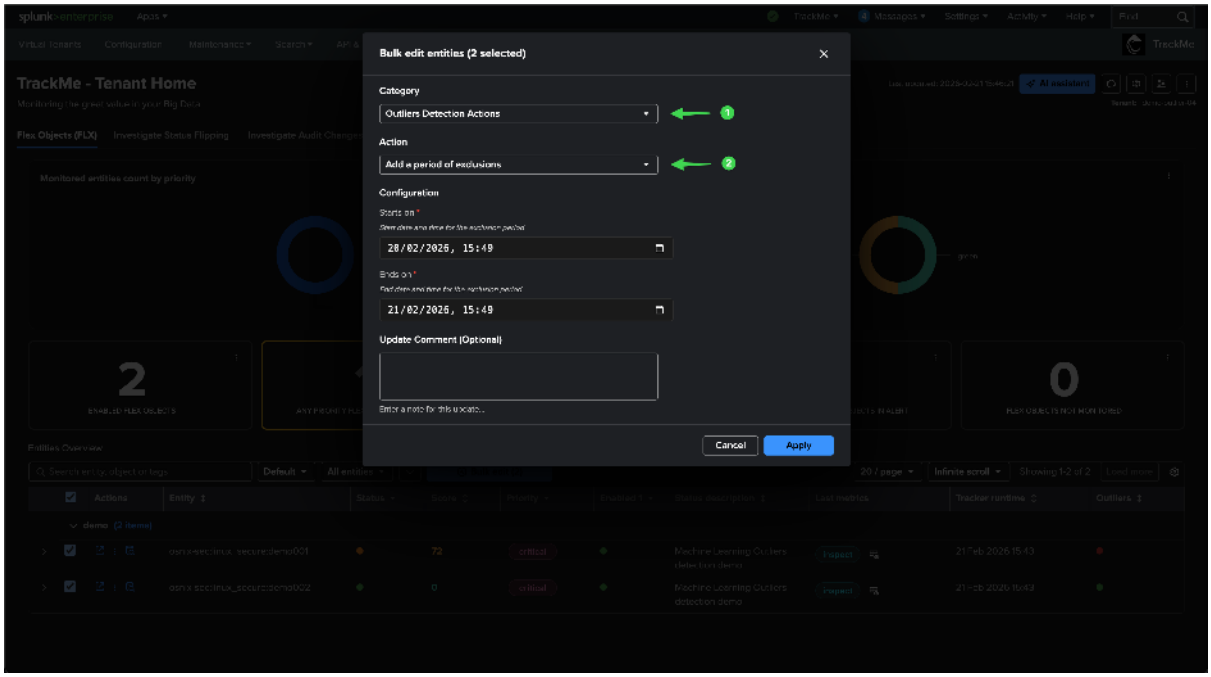
Behavior:

- Excluded periods are removed from the training dataset during the next `mltrain` cycle

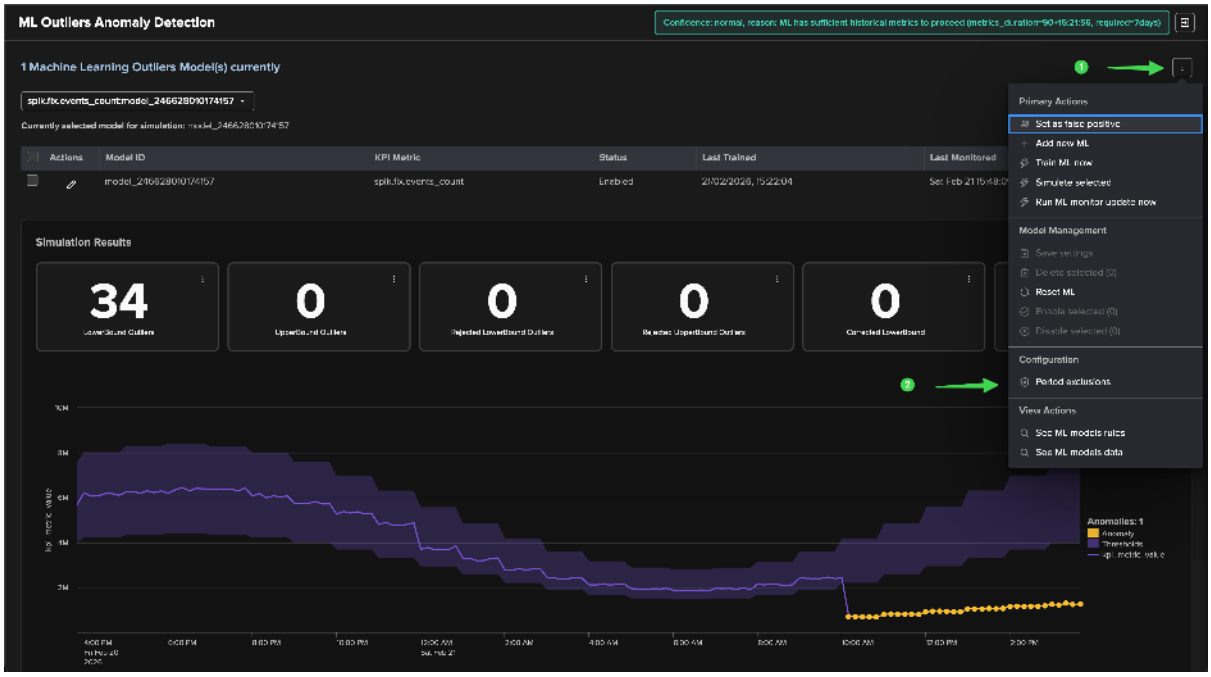
- When the exclusion window falls entirely outside the model's training period (e.g., the exclusion is older than the 30-day training window), TrackMe automatically removes the exclusion entry during training
- Exclusion events are logged in `index=_internal` with sourcetype `trackme:custom_commands:trackmesplkoutlierstrain` and the keyword `period exclusion`

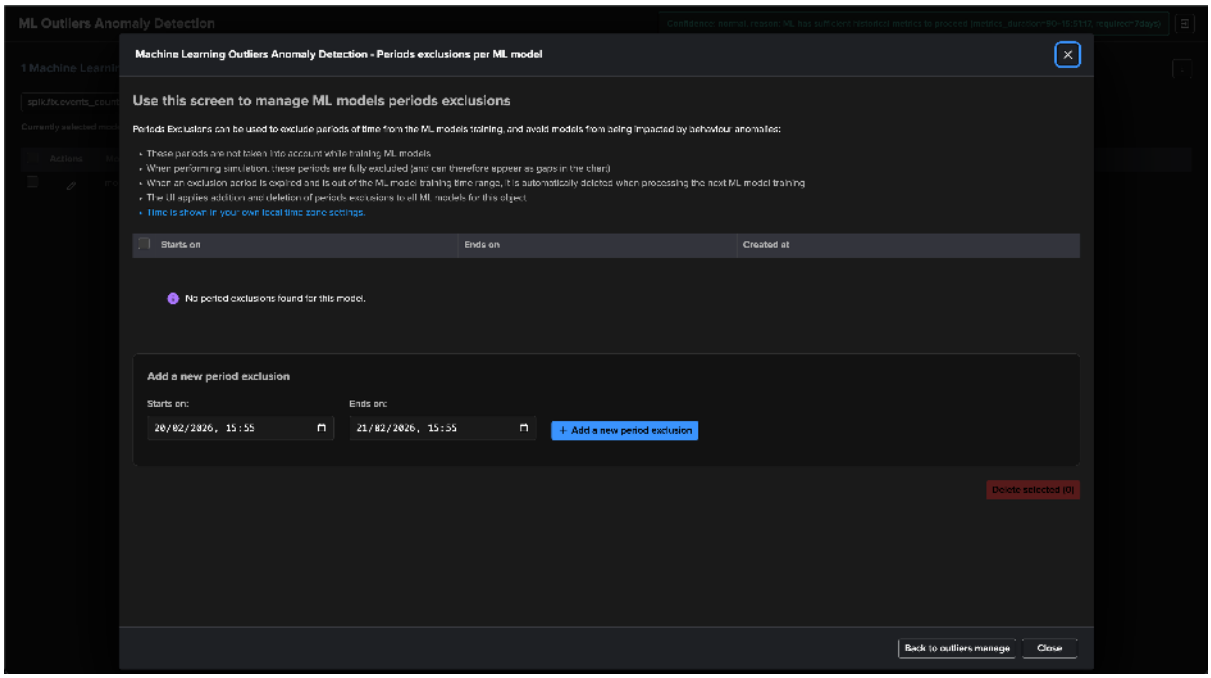
Periods of exclusion can be added through bulk actions:





Or added and managed through the Outliers management screen:





Managing Outliers

Manage Outliers screen

The **Manage Outliers detection** screen is the central interface for reviewing and configuring ML models for a given entity. It is accessible from the entity's detail view by clicking **Manage Outliers detection**.

The screen provides:

- An overview of all ML models defined for the entity, with their current status, confidence level, and last training/monitoring timestamps
- Per-model configuration: training period, time factor, density thresholds, breach direction settings, auto-correction, and threshold guards
- **Simulate Selected**: Runs a true context simulation for the selected model, training a dedicated simulation model to preview results before applying changes

- **Train Selected**: Triggers an immediate training cycle for the selected model
- **Save**: Persists configuration changes to the rules KVstore collection

Bulk actions

TrackMe provides bulk action capabilities for ML Outliers from the entity detail view. Two categories of bulk actions are available:

Outliers actions:

- **Train All Models**: Triggers an immediate training cycle for all models of the entity
- **Monitor All Models**: Triggers an immediate monitoring (rendering) cycle for all models of the entity
- **Reset All Models**: Resets all models to their default configuration and retrains them

Outliers rules:

- **Enable All Models**: Enables outlier detection for all models of the entity
- **Disable All Models**: Disables outlier detection for all models of the entity

These bulk actions are available directly from the entity screen, allowing efficient management without opening individual model configurations.

Enabling and disabling Outliers detection

ML Outliers detection can be controlled at multiple levels:

Per-tenant:

In the **Configuration** UI, the `mloutliers` setting controls whether ML Outliers is enabled for the entire tenant. When set to `0` (disabled), no ML training or monitoring jobs run for that tenant.

Per-component:

The `mloutliers_allowlist` setting defines which components have ML Outliers enabled. By default, all components are included (`dsm,dhm,flx,wlk,fqm,vol`). Remove a component from the list to disable ML Outliers for that component type across the tenant.

Per-entity:

Individual models can be disabled using the `is_disabled` flag in the Manage Outliers detection screen, or via the bulk **Disable All Models** action.

At discovery:

The `splk_outliers_detection_disable_default` system-wide option controls whether outlier detection is enabled or disabled by default when new entities are discovered.

Backend & Scheduling

ML training scheduled jobs

The `ml_train` scheduled job orchestrates the training of ML models across all enabled tenants and components.

- **Schedule**: Runs every hour
- **Max runtime**: 3600 seconds minus a 120-second safety margin
- **Behavior**: The orchestrator iterates through eligible entities sequentially, training models that have exceeded the `splk_outliers_time_train_mlmodels_default` interval since their last training
- **Orchestrator command**: `trackmesplkoutlierstrainhelper`
- **Per-entity command**: `trackmesplkoutlierstrain`

The training frequency per entity is governed by the `splk_outliers_time_train_mlmodels_default` system-wide option (default: 7 days). When the `ml_train` job runs, it selects entities whose models are due for retraining based on this interval — **an individual model is only retrained once every 7 days by default**, so most runs touch few models and exit quickly.

What one training run does:

Training schedules: current defaults vs long-lived tenants

- Since **release 2.3.0**, newly-created tenants schedule ML training **once per hour during the overnight window** (cron `0 22-23,0-6 * * *`) rather than around the clock, and TrackMe deliberately **randomises the exact schedule minute** so trackers stagger across tenants instead of co-firing.
- **TrackMe never rewrites an existing saved-search schedule on upgrade** — a deliberate policy, in case it was customised on purpose. A tenant created before 2.3.0 therefore keeps its original all-day pattern until you re-align its `mltrain` crons manually — a quick, worthwhile piece of scheduling hygiene (the cost is already negligible, since models only retrain every 7 days).

ML monitoring scheduled jobs

The `mlmonitor` scheduled job evaluates trained models against current data to detect outliers.

- **Schedule**: Runs every 20 minutes
- **Max runtime**: 900 seconds minus a 120-second safety margin
- **Behavior**: The orchestrator iterates through eligible entities sequentially, rendering models that have exceeded the `splk_outliers_time_monitor_mlmodels_default` interval since their last monitoring
- **Orchestrator command**: `trackmesplkoutlierstrackerhelper`
- **Per-entity command**: `trackmesplkoutliersrender`

The monitoring frequency per entity is governed by the `splk_outliers_time_monitor_mlmodels_default` system-wide option (default: 1 hour) — the tracker fires every 20 minutes, and each run re-scores the batch of entities whose last evaluation is more than an hour old.

Detection is where an entity is actually scored for anomalies. The compute engine is the **render** step — the same engine that powers the outlier charts in the UI. For each entity and its current seasonality slot:

Note

Both training and monitoring jobs process entities sequentially within their allocated runtime window. If the job reaches its max runtime before processing all entities, remaining entities are picked up in the next cycle — see *batches under a runtime budget*.

Important

Do not reduce the ML-monitor frequency to save cost — that opens scoring gaps and delays detection. The correct cost lever for ML is model *scope* (which entities carry models), not monitor cadence.

System-wide configuration options

The following options are available in **Configuration > System Options** and control the default behavior of ML Outliers across all tenants and components. These options are applied when new entities are discovered and new ML models are created.

Table 18: ML Outliers system-wide options

Option	Default	Description
splk_outliers_min_da	30	Minimal number of days of historical metrics required to reach `normal` confidence level for outlier detection. A tenant can override it in its configuration; the Volume Outliers wizard exposes the override (15 days on a VOL tenant) next to the backfill depth.
splk_outliers_time_t	604800	Interval in seconds between model training cycles per entity. Default: 7 days.
splk_outliers_traini	full	Effective training mode for models configured as `inherit`. `full` rebuilds the current calculation window; `incremental` retains bounded evidence across cycles for supported native KV Store models.
splk_outliers_increm	10000	Maximum observations retained per incremental model. Accepted range: 1,000 to 20,000. The value must provide at least 10 observations for every possible seasonality group.
splk_outliers_time_m	3600	Interval in seconds between model monitoring (rendering) cycles per entity. Default: 1 hour.
splk_outliers_max_ru	900	Maximum duration in seconds for the ML training job. Should align with the job's cron schedule. Default: 15 minutes.
splk_outliers_max_da	15	If a model has not been trained within this many days, it is automatically retrained before rendering. Default: 15 days.
splk_outliers_detect	0	When set to `1`, outlier detection is disabled by default for newly discovered entities. Can still be enabled per entity.
splk_outliers_calcul	stdev	Default calculation mode for anomaly detection. Can be updated per entity.
splk_outliers_densit	0.005	Default lower threshold for the density function algorithm. Lower values = tighter boundaries.
splk_outliers_densit	0.005	Default upper threshold for the density function algorithm. Lower values = tighter boundaries.
splk_outliers_alert_	1	Alert on lower bound breaches for volume-based KPIs. `1` = enabled, `0` = disabled.
splk_outliers_alert_	0	Alert on upper bound breaches for volume-based KPIs. `1` = enabled, `0` = disabled.
splk_outliers_alert_	0	Alert on lower bound breaches for latency-based KPIs. `1` = enabled, `0` = disabled.
splk_outliers_alert_	1	Alert on upper bound breaches for latency-based KPIs. `1` = enabled, `0` = disabled.
splk_outliers_detect	-30d	Default relative time period used for outlier calculations. Applied at discovery, can be updated per entity.
splk_outliers_detect	-1d	Default latest time quantifier for outlier calculations. Accepts Splunk relative time quantifiers (e.g., `-1h@h`).
splk_outliers_detect	%w%H	Default time factor for seasonality. Values: `%H` (hour), `%H%M` (hour/min), `%w%H` (weekday/hour), `%w%H%M` (weekday/hour/min), `%w` (weekday), `none` (no seasonality).
splk_outliers_detect	None	Default KPI metric for latency outlier detection. `None` disables latency outliers by default. Options: `splk.feeds.avg_latency_5m`, `splk.feeds.latest_latency_5m`, `splk.feeds.perc95_latency_5m`, `splk.feeds.stdev_latency_5m`.
splk_outliers_detect	splk.feeds	Default KPI metric for volume outlier detection.
splk_outliers_auto_c	1	Enable auto-correction by default. When enabled, outliers with insignificant deviations are rejected based on min deviation percentages.
splk_outliers_perc_m	5.0	Minimum percentage deviation required for a LowerBound outlier to be considered valid. Below this threshold, the outlier is auto-corrected.
splk_outliers_perc_m	5.0	Minimum percentage deviation required for an UpperBound outlier to be considered valid. Below this threshold, the outlier is auto-corrected.

continues on next page

Table 18 – continued from previous page

Option	Default	Description
splk_outliers_mltk_a	TrackMeNativeDensityFunction,Densi	Comma-separated list of selectable algorithms. TrackMeNativeDensityFunction is the built-in native engine; DensityFunction is the MLTK algorithm from the Splunk AI Toolkit. Custom algorithms added here become available in all Outliers configuration screens.
splk_outliers_mltk_a	TrackMeNativeDensityFunction	Default algorithm used when ML model rules are created (at entity discovery or model reset). Set to DensityFunction to use the MLTK algorithm instead.
splk_outliers_native	kvstore	Storage backend for native TrackMeNativeDensityFunction models. kvstore stores models in per-tenant KVstore collections (kv_trackme_native_ml_models_tenant_<tenant_id>); file stores models as JSON files in the Splunk lookups directory. KVstore is recommended for SHC environments.
splk_outliers_fit_ex	*(empty)*	Extra parameters appended to the fit command during training. For the native engine, example: exclude_dist="beta" to exclude Beta distributions from auto-selection. For MLTK, parameters are passed to the MLTK fit command.
splk_outliers_apply_	*(empty)*	Extra parameters appended to the apply command during rendering. For MLTK, example: sample="True".
splk_outliers_bounda	*(empty)*	Default Splunk macro used for boundaries extraction when defining ML model rules. Leave empty for standard behavior.
splk_outliers_bounda	*(empty)*	Comma-separated list of custom boundaries extraction macros. These become selectable in the Outliers management screens.
splk_outliers_static	*(empty)*	Static override for the calculated lowerBound. When set, replaces the dynamically computed lower boundary.
splk_outliers_static	*(empty)*	Static override for the calculated upperBound. When set, replaces the dynamically computed upper boundary.

Per-model options

Each ML model stores the following options in the rules KVstore collection. These are set from system-wide defaults at discovery and can be customized per model through the Manage Outliers detection screen.

Table 19: ML Outliers per-model options

Option	Description
kpi_metric	The KPI metric used for outlier detection (e.g., splk.feeds.avg_eventcount_5m).
kpi_span	The time span used for metric aggregation during training and rendering.
method_calculation	The algorithm used for this model. TrackMeNativeDensityFunction (native engine, default) or DensityFunction (MLTK).
model_storage	Storage backend for this model's trained data. kvstore (default for native engine) stores models in per-tenant KVstore collections; file stores models as files on disk.
training_mode	inherit (default) resolves the system training-mode default at training time; full or incremental explicitly overrides it for this model. Incremental requires TrackMeNativeDensityFunction with KV Store storage.
period_calculation	The relative time period used for training data (e.g., -30d).
time_factor	The seasonality factor applied to the model (e.g., %w%H, none).
density_lowerthreshold	The lower threshold for the density function algorithm.
density_upperthreshold	The upper threshold for the density function algorithm.
auto_correct	When enabled (1), outliers with insignificant deviations are auto-rejected.
perc_min_lowerbound_	Minimum deviation percentage for LowerBound outliers to be considered valid.
perc_min_upperbound_	Minimum deviation percentage for UpperBound outliers to be considered valid.
alert_lower_breached	When enabled (1), LowerBound breaches generate outlier score events.
alert_upper_breached	When enabled (1), UpperBound breaches generate outlier score events.
min_value_for_lowerb	Threshold guard for LowerBound outliers. If the observed value remains above this threshold, the breach is considered insignificant and rejected.
min_value_for_upperb	Threshold guard for UpperBound outliers. If the observed value remains below this threshold, the breach is considered insignificant and rejected.
outlier_impact_score	The impact score value generated when an outlier is detected. Default: 36. Higher values increase the weight of outlier detections in the entity's cumulative impact score.
is_disabled	When set to 1, outlier detection is disabled for this model.

Advanced Topics

Accessing ML model rules and results

Model rules (configuration) can be retrieved using the dedicated command or directly from the KVstore:

```
| trackmesplkoutliersgetrules tenant_id="<tenant_id>" component="<component>"␣
↪object="<entity_name>"
```

Or via the underlying KVstore lookup:

```
| inputlookup trackme_<component>_outliers_entity_rules_tenant_<tenant_id>
```

Model results (current outlier detection state) can be retrieved similarly:

```
| trackmesplkoutliersgetdata tenant_id="<tenant_id>" component="<component>"␣
↪object="<entity_name>"
```

Or via the underlying KVstore lookup:

```
| inputlookup trackme_<component>_outliers_entity_data_tenant_<tenant_id>
```

Algorithms and model storage

TrackMe provides two density function algorithms for outlier detection:

TrackMeNativeDensityFunction (default):

The native engine is TrackMe's built-in density function, powered by scipy. It automatically selects the best-fit distribution from four types — normal, exponential, Gaussian KDE, and beta — using Wasserstein distance. Models are stored in **per-tenant KVstore collections** (`kv_trackme_native_ml_models_tenant_<tenant_id>`) by default, with file-based storage available as an alternative via the `splk_outliers_native_model_storage` configuration option.

- Training uses the `trackmefit` streaming command
- Rendering uses the `trackmeapply` streaming command
- Distribution exclusion is supported via the `exclude_dist` parameter (e.g., `exclude_dist="beta"`)

DensityFunction (MLTK):

The `DensityFunction` algorithm from the Splunk AI Toolkit is also available and can be selected per model. This algorithm uses the Splunk AI Toolkit's `fit` and `apply` commands. Models are stored as `.mlmodel` files on disk.

Selecting the algorithm:

Add algorithms to the `splk_outliers_mltk_algorithms_list` system-wide option as a comma-separated list. These algorithms become selectable in all Outliers configuration screens. Set the default algorithm for new models via `splk_outliers_mltk_algorithms_default`.

Extra fit/apply parameters:

- `splk_outliers_fit_extra_parameters`: Additional parameters passed to the `fit` command during training (e.g., `exclude_dist="beta"` to exclude Beta distributions)
- `splk_outliers_apply_extra_parameters`: Additional parameters passed to the `apply` command during rendering (e.g., `sample="True"` for the MLTK `DensityFunction`)

These parameters are applied when ML model rules are defined (typically at entity discovery or model reset).

Custom boundaries extraction macros:

For algorithms that require custom boundary calculations, define Splunk macros and register them in `splk_outliers_boundaries_extraction_macros_list`. The default macro used for new models is set via `splk_outliers_boundaries_extraction_macro_default`.

KVstore model lifecycle management:

When using the native engine with KVstore storage, TrackMe automatically manages the model lifecycle:

- Models are stored in per-tenant KVstore collections, providing tenant isolation
- The general health manager periodically detects and removes orphan models — models whose associated entity or outlier rule no longer exists
- After upgrading from file-based to KVstore storage, the render command detects missing KVstore models and triggers automatic training on first access, ensuring zero downtime for ML outlier results

Expanding ML model data

The `trackmesplkoutliersexpand` streaming command expands the complex JSON dictionaries stored in the ML model KVstore collections into individual fields, making them easier to analyze and filter.

Expanding model results:

```
| trackmesplkoutliersgetdata tenant_id="<tenant_id>" component="<component>"␣
↪object="<entity_name>"
| trackmesplkoutliersexpand
```

Expanding model rules (definitions):

```
| trackmesplkoutliersgetrules tenant_id="<tenant_id>" component="<component>"␣
↪object="<entity_name>"
| rename entities_outliers as models_summary
| trackmesplkoutliersexpand
```

Extracting specific fields from outlier reasons:

After expanding results, use `rex` to extract details from the `isOutlierReason` field:

```
| rex field=isOutlierReason "time=(?<outlier_time>[^,]+)"
| rex field=isOutlierReason "pct_decrease=(?<pct_decrease>[^,\}]+)"
```

REST API endpoints

TrackMe exposes REST API endpoints for programmatic control of ML Outliers operations. These endpoints are used internally by the UI and can be called directly for automation.

Train models:

```
| trackme mode=post url="/
↪services/trackme/v2/splk_outliers_engine/write/outliers_train_models" body="{
↪'tenant_id': '<tenant_id>', 'component': '<component>', 'object': '<entity_name>
↪'}"
```

Reset models (restore to defaults and retrain):

```
| trackme mode=post url="/
↪services/trackme/v2/splk_outliers_engine/write/outliers_reset_models" body="{
↪'tenant_id': '<tenant_id>', 'component': '<component>', 'object': '<entity_name>
↪'}"
```

Delete models:

```
| trackme mode=post url="/
↪services/trackme/v2/splk_outliers_engine/write/outliers_delete_models" body="{
↪'tenant_id': '<tenant_id>', 'component': '<component>', 'object': '<entity_name>
↪'}"
```

Mass operations via REST API

For bulk operations across multiple entities, combine the `trackmesplkoutliersexpand` command with Splunk's `map` command to iterate over filtered results:

Example — mass delete models matching a specific KPI metric:

```
| trackmesplkoutliersgetrules tenant_id="<tenant_id>" component="<component>"
| rename entities_outliers as models_summary
| trackmesplkoutliersexpand
| search kpi_metric="<target_metric>"
| map search="| trackme mode=post url=\"/
↪services/trackme/v2/splk_outliers_engine/write/outliers_delete_models\" body=\"{
↪'tenant_id': '$tenant_id$', 'component': '$component$', 'object': '$object$'}\""
```

Warning

Mass delete operations are irreversible. Always verify the filter criteria by running the search without the `| map` portion first to confirm the targeted entities.

Troubleshooting

Understanding ML rendering decisions

To inspect the detailed decision-making process of the ML rendering engine, use the `trackmeprettyjson` command on the raw rendering results. This shows per-time-frame density function output, boundary calculations, and auto-correction decisions:

```
| trackmesplkoutliersgetdata tenant_id="<tenant_id>" component="<component>"␣
↪object="<entity_name>"
| fields _time, _raw
| trackmeprettyjson fields=_raw
```

The JSON output includes the rendering results for each time bucket, the calculated boundaries (lowerBound / upperBound), whether the value was flagged as an outlier, and whether auto-correction rejected the outlier.

Training logs

ML training activity is logged in `index=_internal` under two sourcetypes:

Orchestrator logs (job-level activity, entity selection, timing):

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkoutlierstrainhelper
```

Per-entity training logs (model training details, period exclusions, errors):

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkoutlierstrain
```

Monitoring logs

ML monitoring (rendering) activity is logged in `index=_internal` under two sourcetypes:

Orchestrator logs (job-level activity, entity selection, timing):

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkoutlierstrackerhelper
```

Per-entity rendering logs (outlier detection results, score events, errors):

```
index=_internal sourcetype=trackme:custom_commands:trackmesplkoutliersrender
```

8.9 Integrations & Remote Deployments

TrackMe can monitor remote Splunk deployments and exchange data with external systems. This part covers every connection TrackMe makes beyond the local search head.

- **Remote Splunk deployments** — monitor Enterprise or Cloud over an authenticated, token-rotated, failover-capable connection. This is the foundation several other integrations build on.
- **CMDB integration** — enrich entities with external ownership and asset metadata.
- **Lookup monitoring** — track the freshness of Splunk lookups as entities.
- **Splunk SOAR** — monitor SOAR's own health, and push TrackMe events into SOAR.
- **TrackMe App on SOAR** — drive TrackMe from SOAR playbooks.
- **Cribl** — monitor Cribl Stream pipelines with Flex Objects use cases. For a Cribl Stream deployment, the dedicated *Cribl Stream tenant* is the native path.
- **Elastic sources** — turn arbitrary SPL into tracked DSM entities.
- **TrackMe Broker** — reach what sits on your own network from Splunk Cloud: remote Splunk accounts, self-managed Cribl Leaders and webhook receivers, with every connection started on your side.

8.9.1 Remote Splunk deployments

A tenant can monitor a **remote Splunk deployment** — Enterprise or Cloud — instead of (or as well as) the local search head. One TrackMe instance then becomes a single pane of glass across many Splunk estates. Remote access is configured once as a **remote account** and referenced by tenants and trackers.

Which components need a remote account?

It depends less on the component than on **where its data is**. A component reads one of two things, and each has its own rule:

- **Indexed data** (`_internal`, `_introspection`, `_audit`, metrics, your own indexes). If the search head TrackMe runs on can search those indexes, the component uses `local` and needs no remote account. Otherwise, it uses a remote account to a search head that can.
- **A Splunk instance's own REST API** (its saved searches, users and roles, inputs, HEC tokens). The instance TrackMe runs on is `local`. Any other search head tier or instance needs a remote account, **even when it searches the same indexers**.

What each component's account points at

Component	Remote account	It points at
DSM, *DHM*, *MHM*	Only when the data is on another deployment.	A search head of that deployment.
FLX, *FQM*	Optional, chosen per tracker.	A search head that can search the tracker's data.
VOL	Rarely. VOL reads `license_usage.log` in `_internal` on the indexers the license manager forwards to, so `local` works wherever TrackMe searches those indexers.	A search head of another deployment. It does not need to reach the license manager.
WLK	One per search head tier other than TrackMe's own.	That tier: the saved searches' definitions and versions are read there, even over the same indexers.
UAM	One per search head tier other than the local one; remote accounts only for central monitoring.	Each tier. The indexing-layer searches go through the route account, `local` when it is selected.
Splunk HEC	Only when the endpoints' internal indexes are on another deployment. Optional accounts for the splunkd API enrichment.	That deployment's search tier, and the endpoints' management port for the enrichment.
Forwarder Collectors	Only when the collectors' internal indexes are on another deployment. Optional, per collector, for the input inventory.	That deployment's search tier, and each collector for the inventory.
Hybrid trackers, *lookups*, coverage gaps, *CMDB enrichment*	Optional.	The deployment that holds the data, lookup or CMDB.
Replica trackers	Always.	Another TrackMe instance.
Cribl Stream and Cribl Edge	No: Cribl tenants use their own Cribl connection.	The Cribl Leader.

Common setups

One search head, one indexing layer. No remote account. Every component reads locally; only the optional API reads of the Forwarder Collectors and Splunk HEC take an account per instance.

TrackMe next to Enterprise Security. TrackMe on the ad hoc search head, Enterprise Security on another tier over the same indexers: one remote account, to the ES tier, which WLK and UAM use to read its saved searches,

users and roles. DSM, DHM, MHM, FLX, FQM and VOL stay local.

A dedicated monitoring Splunk. TrackMe on its own Splunk, watching other deployments: one remote account per deployment, used by that deployment's tenants. VOL, UAM, Splunk HEC and Forwarder Collectors take one tenant per deployment.

Splunk Cloud, forwarders on your network. Your data is indexed on the stack, so the data components stay local. A premium search head (Enterprise Security) takes one account for WLK and UAM. Instances on your own network, read through their API, are reached through a *TrackMe Broker*, which only connects out.

Tip

Trackers run as the **tenant owner**, so the owner must hold one of the account's allowed roles (see *Access control*). The account itself needs read access to the indexes the component searches, `_internal` included.

Remote accounts

A remote account stores everything TrackMe needs to reach a remote search head:

- a **name** — the identifier tenants and trackers reference;
- one or more **Splunk REST URLs**;
- a **bearer token** (HTTPS only, stored encrypted in Splunk's credential store);
- the remote **app namespace** (defaults to `search`);
- the **allowed Splunk roles** (account-level RBAC).

The implicit account `local` represents on-box data.

High availability and resilience

- **Failover** — the URL field accepts a comma-separated list of REST endpoints. TrackMe tests them and picks a reachable one at random, so a search-head cluster with no load balancer still gets transparent failover.
- **Retry with backoff** — transient connection and `5xx` errors are retried with exponential backoff (configurable attempts, initial delay, and multiplier per account).
- **Connectivity test** — building or editing an account runs a connectivity and authentication check, and refuses to save on failure.

All remote searches flow through the `splunkremotesearch` command, which TrackMe uses transparently: it splits an SPL search into a remote part (run on the remote SH) and a local part, so trackers work against remote data without any special handling.

Automatic token rotation

TrackMe rotates each account's bearer token automatically — by default every **7 days**. Once a day, for each account, it creates a new token on the remote, verifies it, stores it locally, and revokes the old one. The remote service account therefore needs `edit_tokens_own` or `edit_tokens_all`, in addition to read access to its indexes (including `_internal`). `list_tokens_own` is not needed to rotate: TrackMe lists the account's tokens once the rotation is done. Without that capability the new token is in place and the old one revoked, but this last step fails, so the run reads as failed, including a rotation you force by hand.

Note

The **initial** token you create by hand is never revoked by TrackMe — disable it yourself once the first rotation has succeeded, per your token policy. A failed rotation is non-destructive: it is logged and the existing token stays in use.

You can also **force a rotation on demand** — for all accounts or a single one — from the Remote Accounts Overview. And for **Splunk Cloud** accounts, TrackMe can **auto-heal the target URL list** as the search-head-cluster members are recreated. Both are covered in *Remote deployments — in depth*.

Access control

Each account lists the local Splunk roles permitted to use it (`rbac_roles`) — by both membership and inheritance. A user not in a listed role is refused by `splunkremotesearch`. This prevents a tenant from being pointed at an unrelated Splunk to exfiltrate data. See *Roles & access control*.

A tenant selects its remote account at creation time (and trackers and Flex objects can each target a remote account too); the same connectivity check runs automatically.

A deployment the stack cannot reach

From Splunk Cloud, a Heavy Forwarder, a Deployment Server or a search head on your own network cannot be reached directly. Set **Connectivity** on the account to a *TrackMe Broker*: an add-on on a forwarder on your side then carries the account's calls, and every connection starts on your side. Nothing else about the account changes, and `splunkremotesearch` works the same way.

The account's token rotates automatically through the broker, as on a direct account and on the same condition: the account on the destination can edit its tokens (`edit_tokens_own` or `edit_tokens_all`). It is not the forwarder's own token: *two independent tokens* are at work, and rotating one never touches the other.

See also

- *Remote deployments — in depth* — cross-app RBAC, the REST endpoints, token-rotation internals, troubleshooting, and an SPL cookbook.
- *TrackMe Broker* — reaching a deployment on your own network from Splunk Cloud.
- *Creating a tenant* — choosing a deployment target when creating a tenant.
- *Roles & access control* — account-level access control.
- *Lookup monitoring*, *Hybrid trackers* and *Replica trackers* — features that also run against remotes.

8.9.2 Remote deployments — in depth

Tip

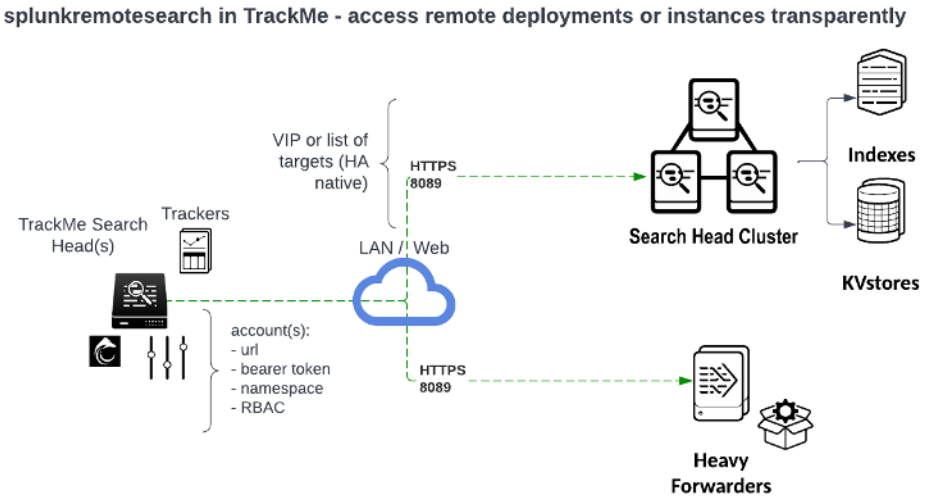
This is the **in-depth reference** for monitoring remote Splunk deployments with `splunkremotesearch`. For the readable overview, start with *Remote deployments*. This page covers the cross-app RBAC setup, the REST endpoints, token-rotation internals, troubleshooting recipes, and a ready-to-run SPL cookbook. For a deployment on your own network that Splunk Cloud cannot reach, see *TrackMe Broker*.

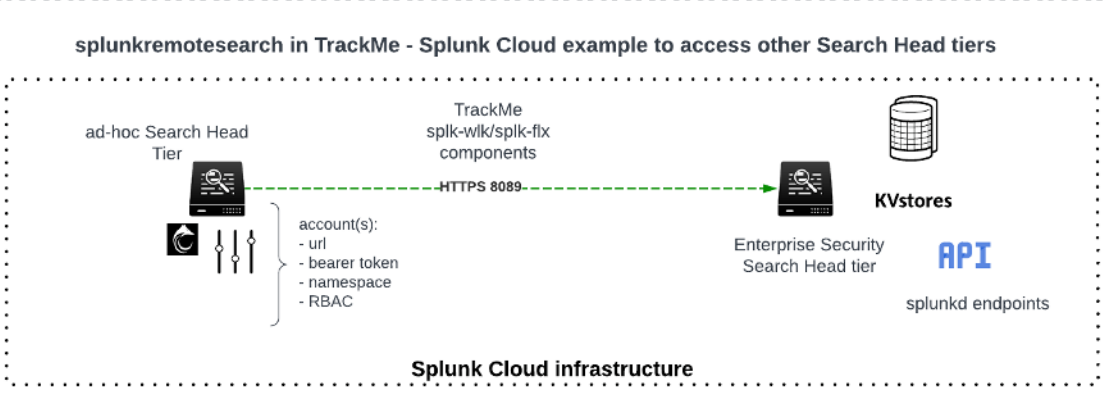
Overview of Splunk Remote Search Capabilities in TrackMe

TrackMe can manage locally available data, as well as data from remote Splunk deployments (Splunk Enterprise, Splunk Cloud) and remote independent instances such as utility nodes (Deployment Servers, Cluster Managers, License Manager, etc.) or Heavy Forwarders.

This is a key capability provided by TrackMe, which is used transparently when you configure trackers, allowing you to target either the local environment where TrackMe is hosted, or any other Splunk remote deployment.

There are plenty of use cases where this capability of TrackMe is game-changing, enabling TrackMe to become the single pane of glass for your monitoring and operations.





The Splunk remote search feature in TrackMe relies on multiple aspects:

- The `splunkremotesearch` TrackMe generating custom command, which relies on the Splunk Python SDK to perform searches remotely at scale
- The concept of **accounts** which allows TrackMe to store and access pre-configured configuration defining the Splunk remote environment
- Various backend-level features in TrackMe designed to identify and handle the remote entities accordingly
- The Remote deployment concept can be used for various purposes, from feeds tracking on remote deployments, to Flex objects, CIM compliance tracking, or Workload between different Search Head tiers

Note

Local service account user or SAML service account

- On the remote Search Heads tier counterpart, you can use a local user or a SAML user associated with the bearer token
- However, to use SAML, your SAML setup needs to support AQR and it needs to be configured
- Reference: https://docs.splunk.com/Documentation/Splunk/latest/Security/Setupauthenticationwithtokens
- Reference: https://docs.splunk.com/Documentation/SplunkCloud/latest/Security/SAMLConfigJWT

Hint

TrackMe supports multiple REST endpoints per account, with High Availability and Disaster Recovery capabilities: (from version 2.0.22)

- For each account, you can specify a comma-separated list of REST endpoints per account
- TrackMe will automatically verify the connectivity and randomly choose a REST endpoint target among any reachable endpoints
- Therefore, you can specify multiple endpoints to support high availability and distribution of the searches against multiple endpoints automatically
- This capability can be useful, for example, if you deal with a Search Head Cluster and cannot use a load balancer or VIP to access the cluster

Hint

TrackMe supports and requires Role-Based Access Control per Remote account defined: (from version 2.0.34)

- For each account, you need to specify a comma-separated list of Splunk roles that are allowed to use this Remote account
- For retro-compatibility purposes, if the roles for an existing account have not been set up yet, TrackMe will use built-in roles in addition to typical admin roles (admin, sc_admin, trackme_user, trackme_power, trackme_admin)
- If the user calling the `splunkremotesearch` command is not a member of the account-specified roles, access will be refused
- From version 2.0.61, both **direct membership and inheritance** are supported; users need to be a member of any of the provided roles, or a role which inherits from any of these roles

Hint

TrackMe automatically performs the rotation of Bearer tokens: (from version 2.1.8)

- Since TrackMe 2.1.8, we support automated rotation of the Splunk bearer tokens for remote accounts
- For every Splunk remote account, TrackMe will automatically try to rotate the bearer token based on the remote account's configured retention (every 7 days by default)
- This requires the Splunk service account on the remote side to own the capability `edit_tokens_own` or `edit_tokens_all`, so the token can be created and the previous token revoked
- The token renewal and revocation process is orchestrated by the general health tracker, which is executed once per day
- Note that the initial token you have manually created will not be revoked by TrackMe, and would be disabled automatically depending on your settings

Hint

TrackMe Audit dashboard: Splunk Remote Account Overview (from version 2.1.16)

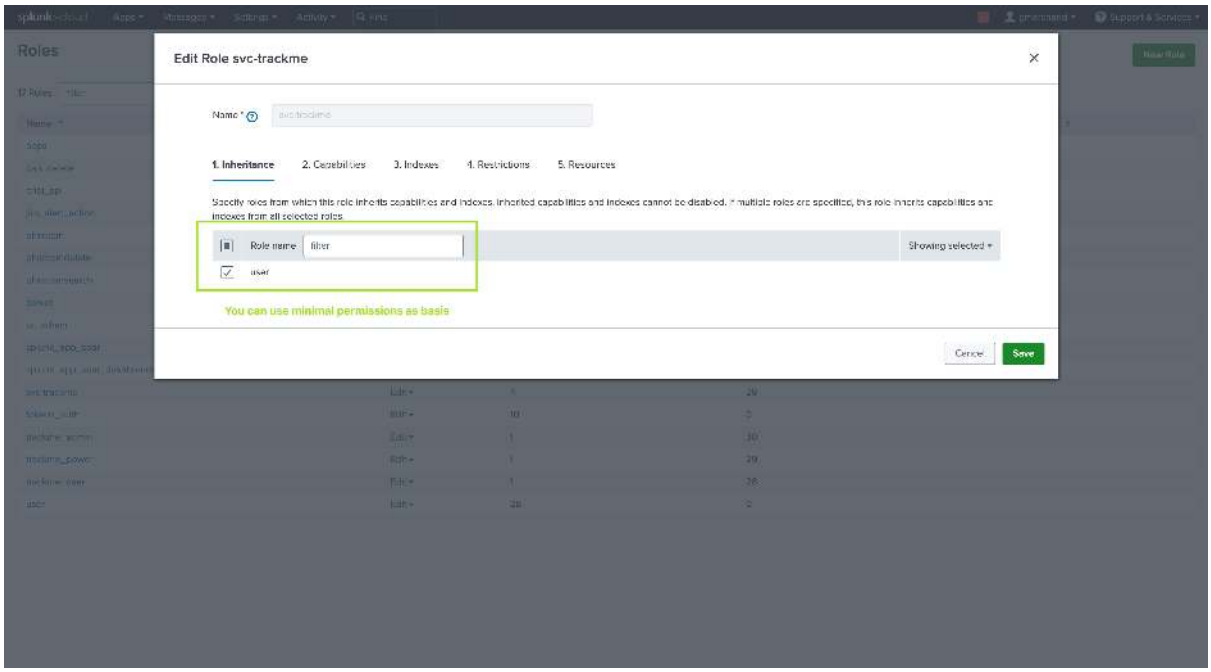
- TrackMe now includes an Audit dashboard to monitor the Splunk remote accounts
- This dashboard is available from the menu `Audit & Troubleshooting` -> `Audit - Splunk Remote Accounts Overview`

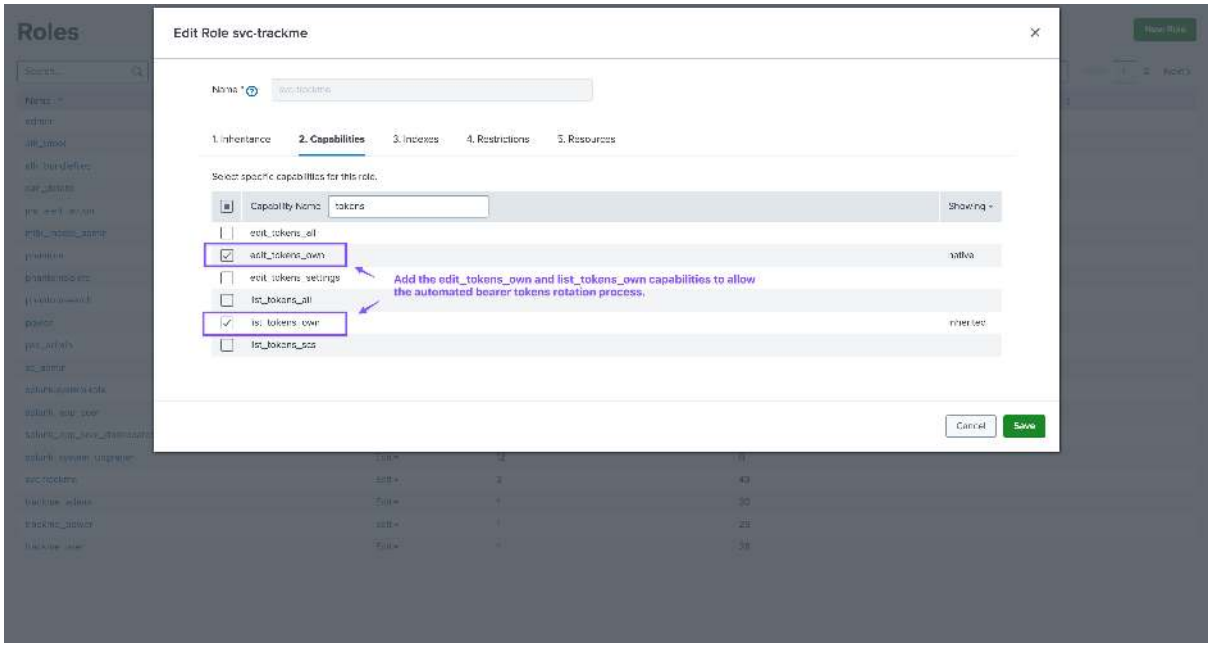
- This dashboard provides a comprehensive overview of the Splunk remote accounts, including the bearer token rotation process

Minimal RBAC Requirements for the Remote User Account

TrackMe remote capabilities rely on a Splunk bearer token authentication; this token is associated with a Splunk user on the remote side which itself is associated with specific roles, capabilities, permissions, and resource restrictions:

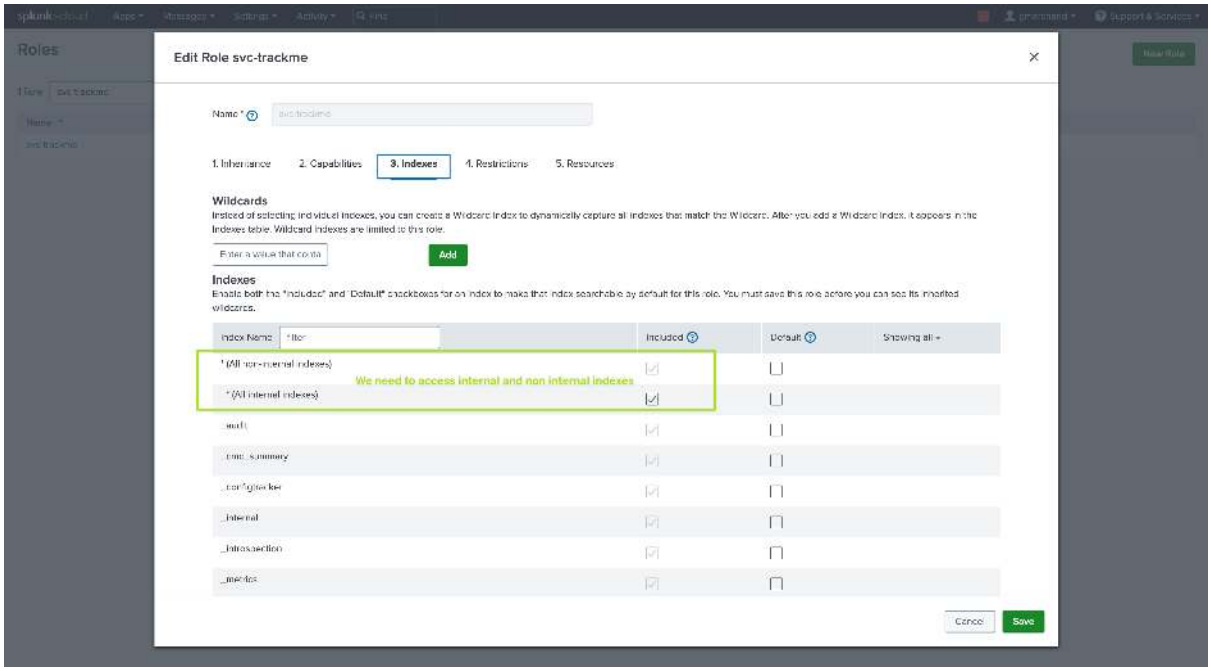
- **Roles and capabilities:** The user can be created with minimal permissions using the Splunk `user` role out of the box (You can inherit from user or a role providing the same capabilities as power)
- **Additional capabilities for tokens rotation:** To allow the automated bearer tokens rotation, the user needs to have the capability `edit_tokens_own` or `edit_tokens_all`. `list_tokens_own` is not needed to rotate: TrackMe lists the account's tokens once the rotation is done. Without that capability the new token is in place and the old one revoked, but this last step fails, so the run reads as failed, including a rotation you force by hand
- **Indexes:** Make sure the user can access both normal and `internal` indexes
- **Restrictions:** The user for TrackMe should not have any time limits restrictions, as there are use cases which require long-term searches
- **Resources:** It is recommended to give this user **enough concurrent searches** (unlike a very basic or minimal user) as well as a **sufficient quota** (5GB or 10GB for instance)

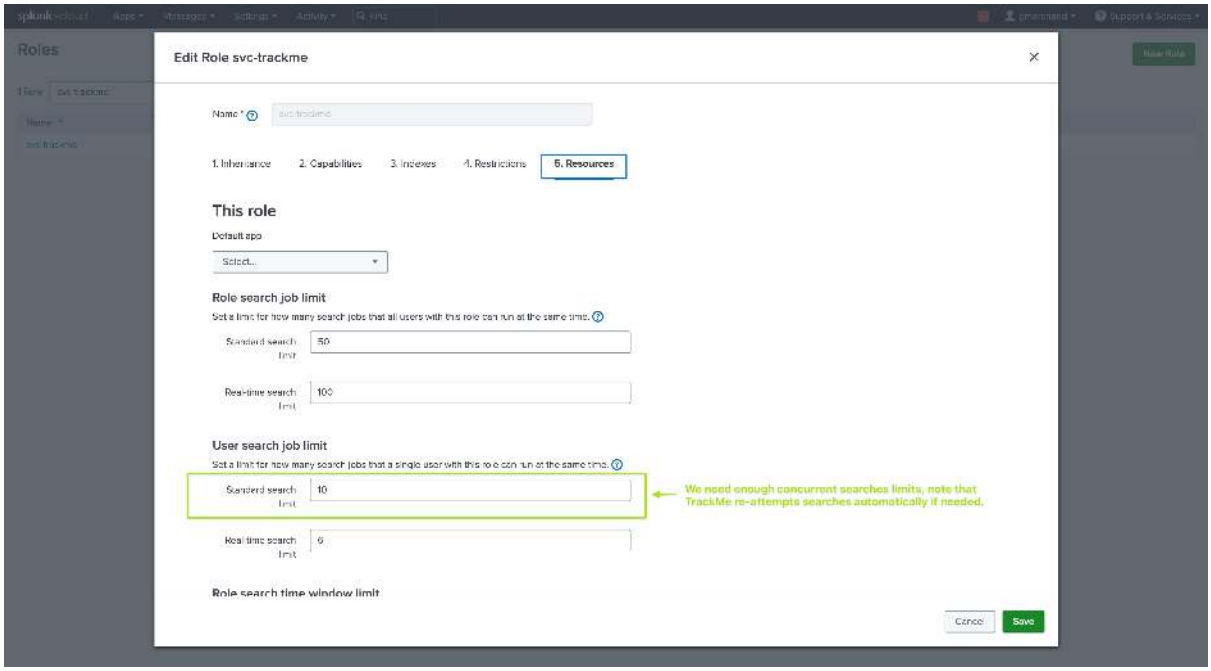


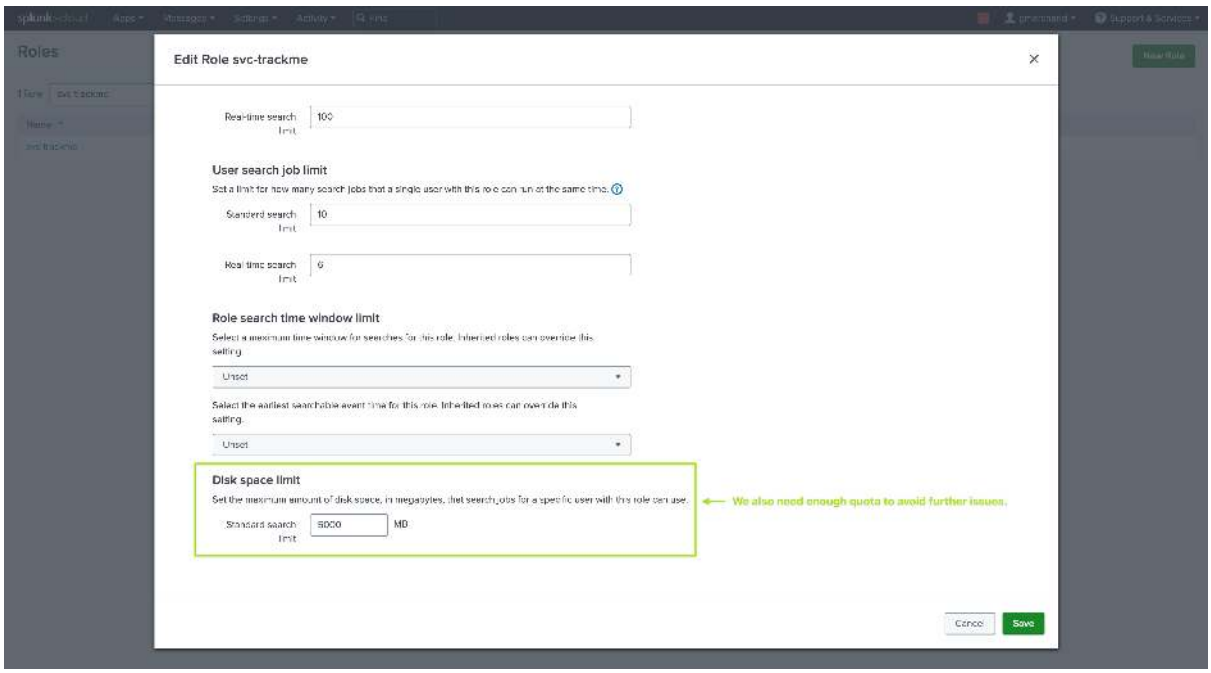


Note

This screen also selects `list_tokens_own`, and its caption asks for both capabilities. Rotation needs only `edit_tokens_own` or `edit_tokens_all`, and `list_tokens_own` keeps the run from reading as failed afterwards.







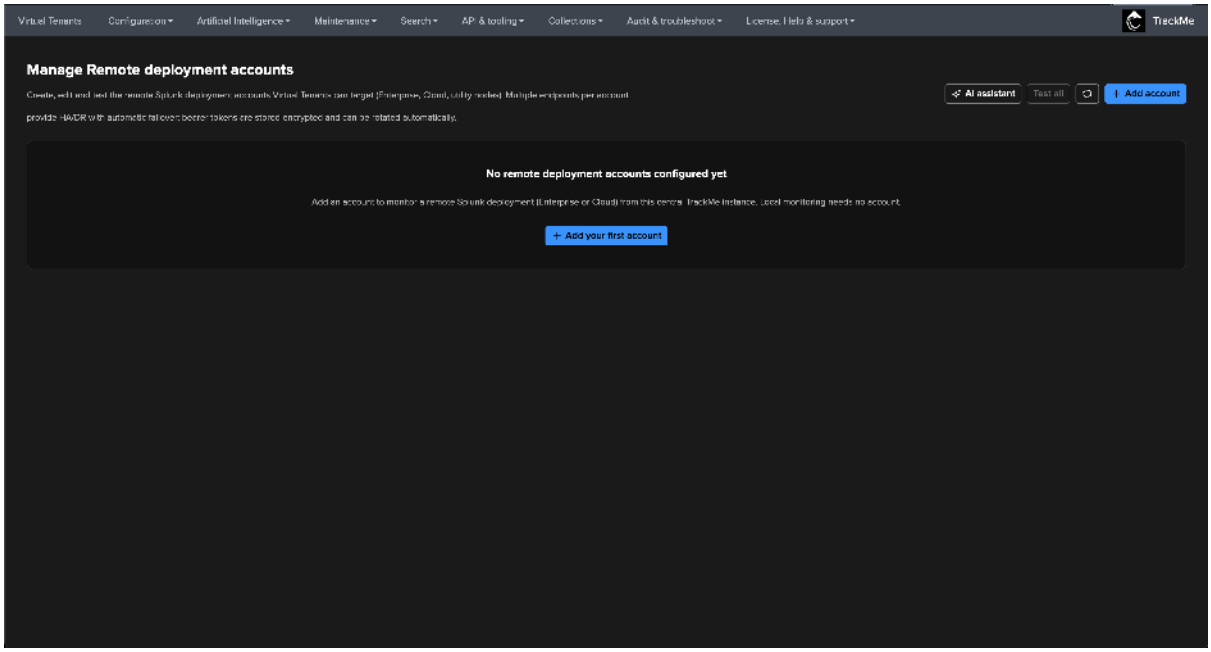
Configuring a New Remote Account

When configuring a Splunk Remote deployment, you will provide:

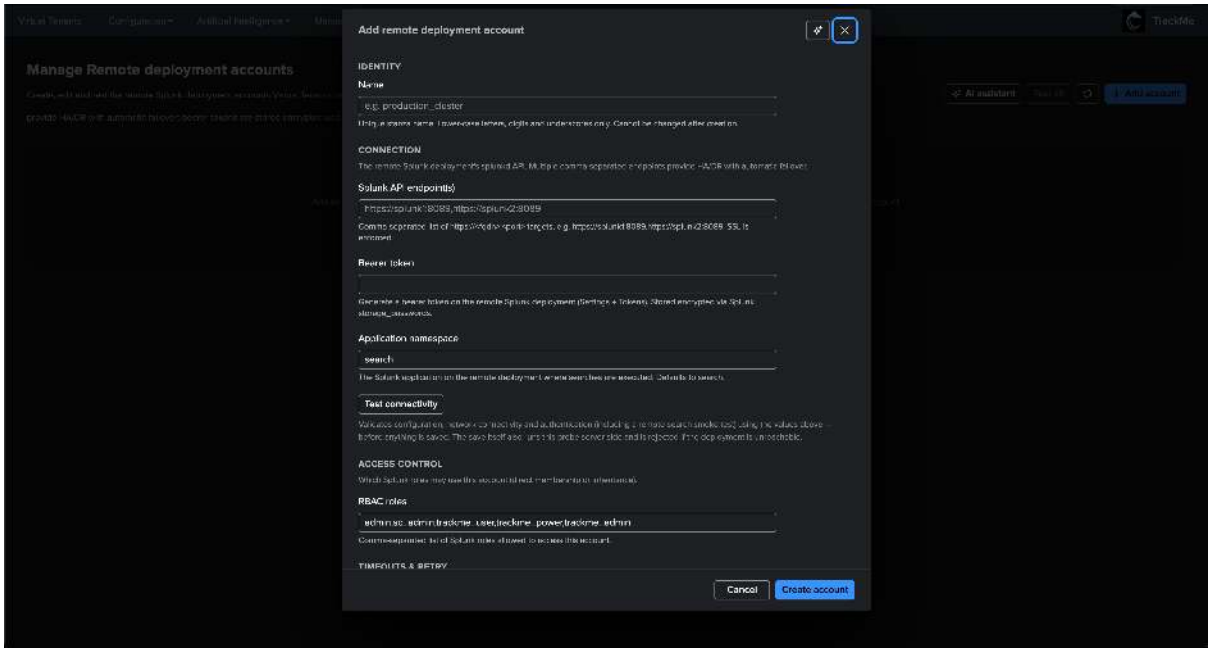
- A name which uniquely identifies the account (the account ID)
- The URL of the Splunk Search Head REST API, this can be a Fully Qualified Domain Name, a hostname, or an IP address (which itself can be a load balancer)
- Multiple REST endpoints can be submitted as a comma-separated list of targets
- The bearer token value, which is the secured credential to access the environment, is securely stored in the Splunk credential store
- The application namespace where remote searches are to be executed, defaults to the Search application on the remote target
- A comma-separated list of Splunk roles allowed to use this account (membership or inheritance)

- At the time of creation or later on if you edit the account, a connectivity check is automatically performed which validates the network connectivity and authentication. If the check fails, the interface will refuse the requested action

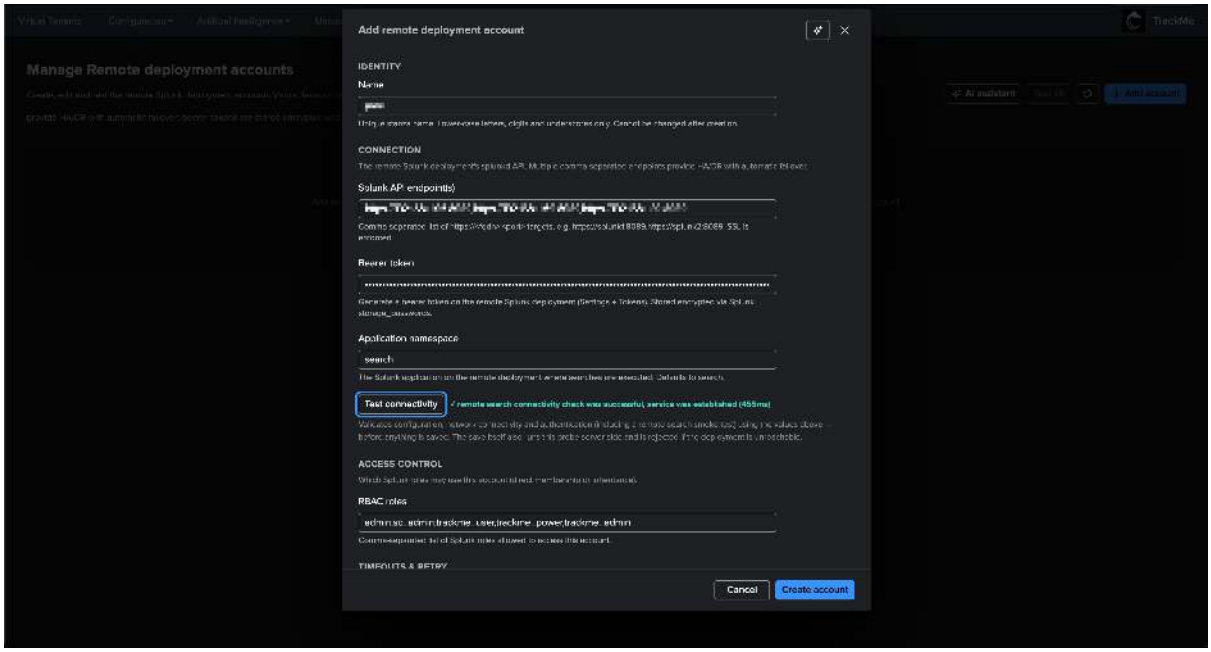
Remote accounts are managed from a dedicated page reachable from the top navigation: Configuration → Manage Remote deployment accounts. The page lists every account with its endpoints, token-rotation cadence and last connectivity status, and offers per-row **Edit / Clone / Test connectivity / Delete** actions plus **Test all** and **Add account**. When no account exists yet, the page shows a short getting-started guide:



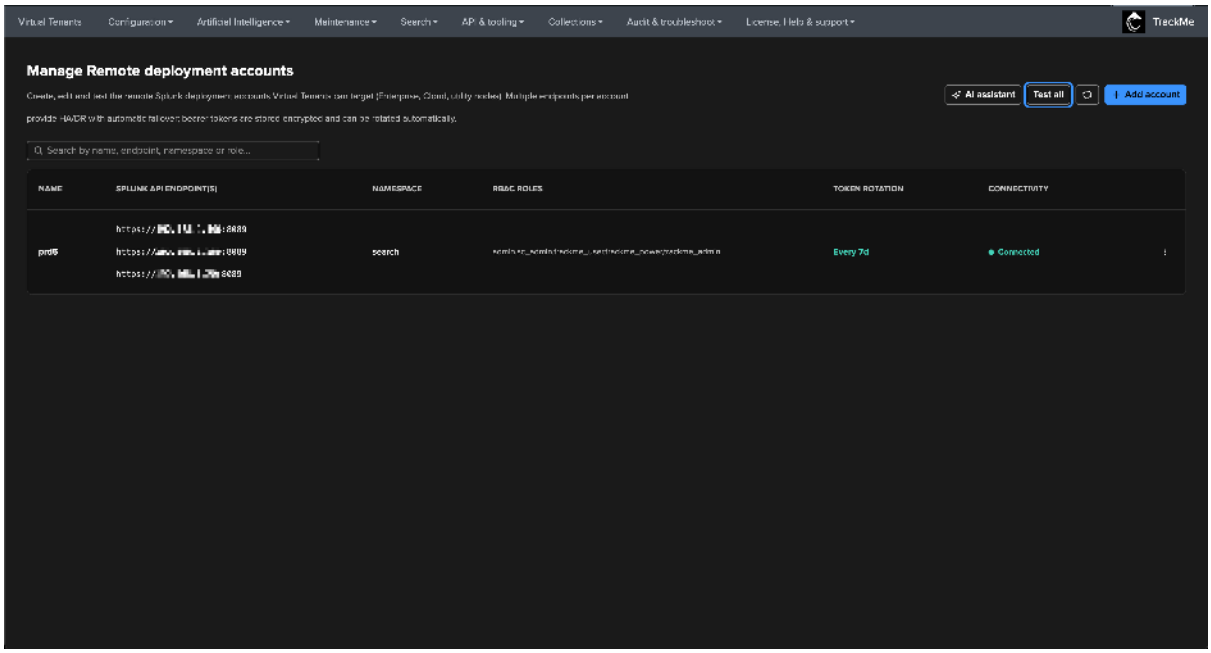
Click Add account and fill in the identity, connection, bearer token, namespace and RBAC fields:



Use Test connectivity in the modal to validate the account before saving — TrackMe runs a real remote-search smoke test and reports the round-trip time. The account cannot be saved until the check passes, so a broken account never reaches the list:

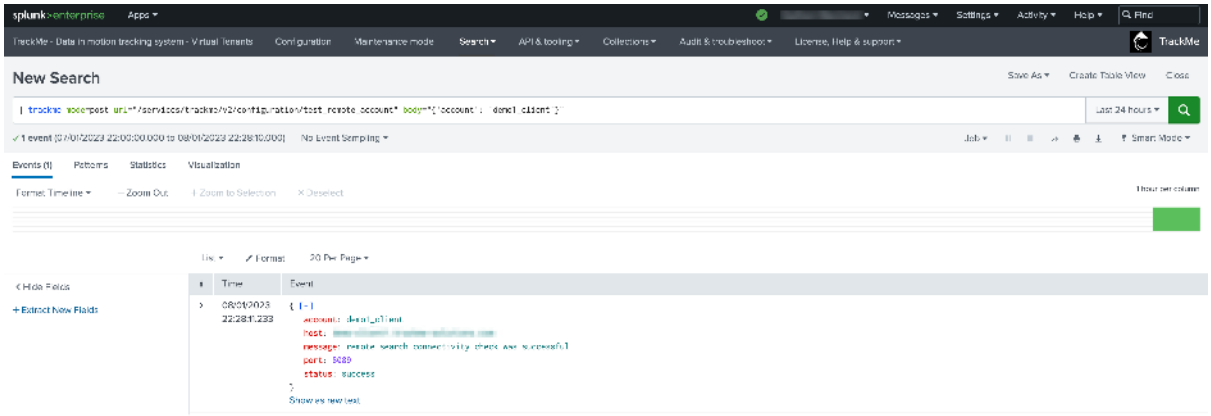


Once saved, the account appears in the list with its live connectivity status:



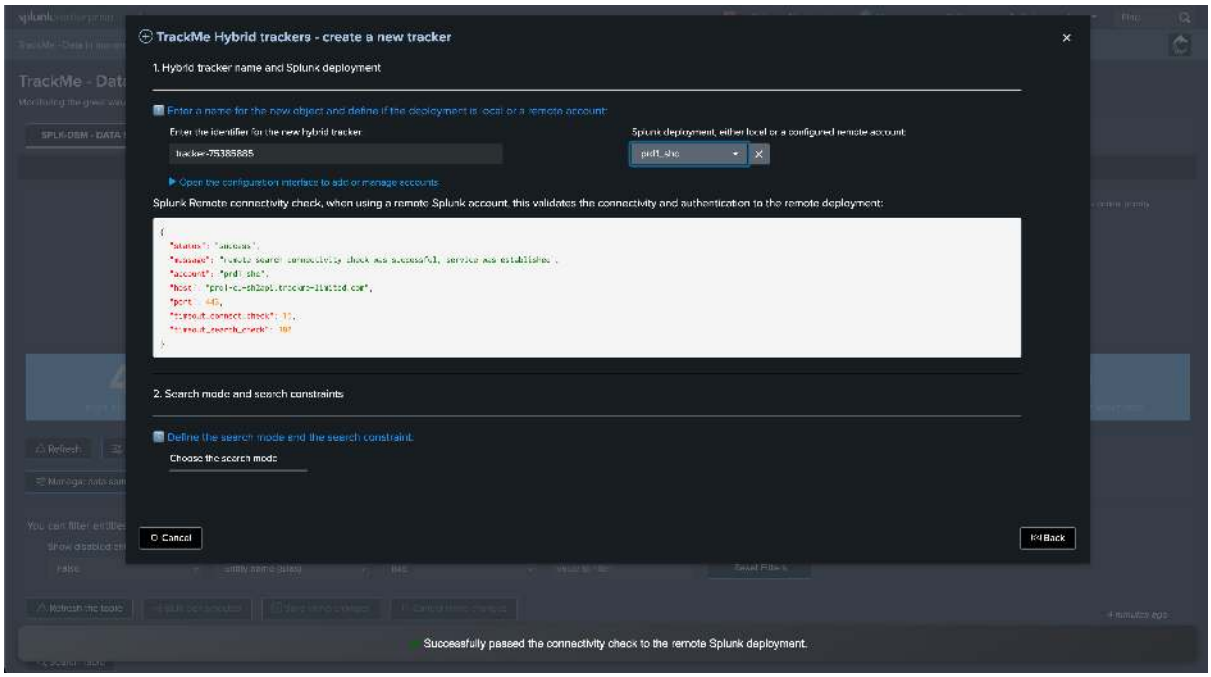
You can also test the connectivity at any time using the following TrackMe REST endpoint in SPL:

```
| trackme mode=post url="/services/trackme/v2/configuration/test_remote_account"␣
↪body="{'account': '<name of the account>'}"
```



The endpoint verifies that TrackMe can connect and authenticate successfully to the Splunk remote deployment.

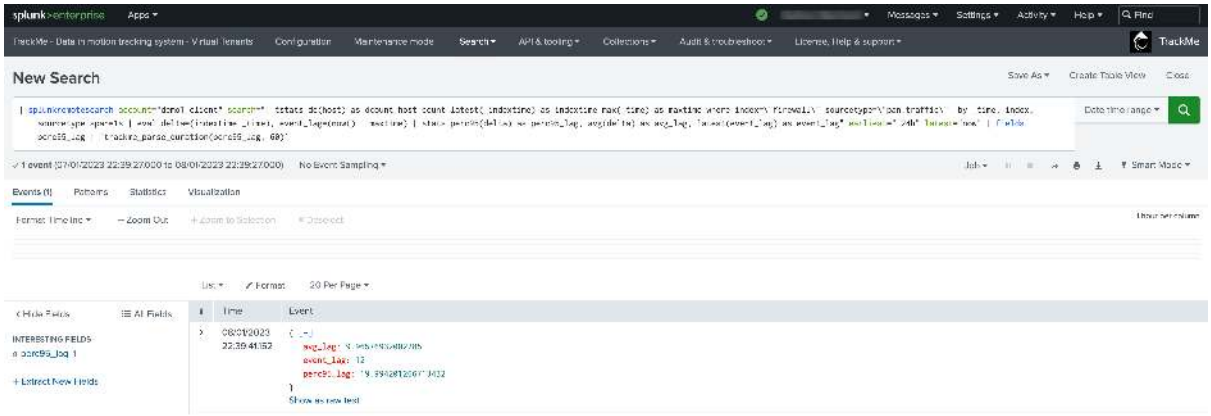
When creating a new Virtual Tenant or a new Hybrid object, TrackMe will perform the same connectivity verification automatically:



Example of a Splunk Remote Search Performed by TrackMe

When performing Remote Splunk searches, TrackMe automatically defines an optimized search to split the search logic between the part executed remotely, and the search logic handled locally.

Example:

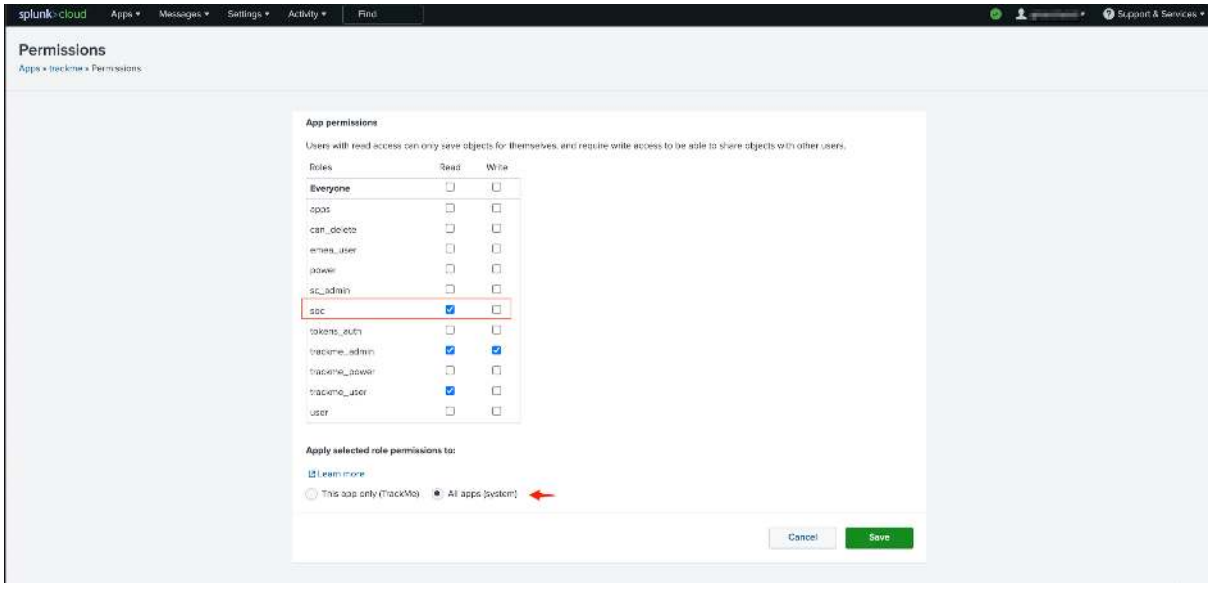


Using splunkremotesearch in a Different Application Namespace

To use the command `splunkremotesearch` from a different application than TrackMe, such as Search & Reporting, you need to perform a few steps as follows.

Share TrackMe at System Level

TrackMe by default is shared at the application level. You need to share at the **system** level as well as provide read access at least for your user roles ("soc" in our example):



Manage Capabilities

TrackMe requires capabilities to be able to access the command `splunkremotesearch`; the user roles need to have the capability:

- trackmeuseroperations

You can choose to inherit from the `trackme_user` role, or you can add the capability to your user roles as needed.

*Note: Access is granted by any of the 3 built-in trackme_* user roles**

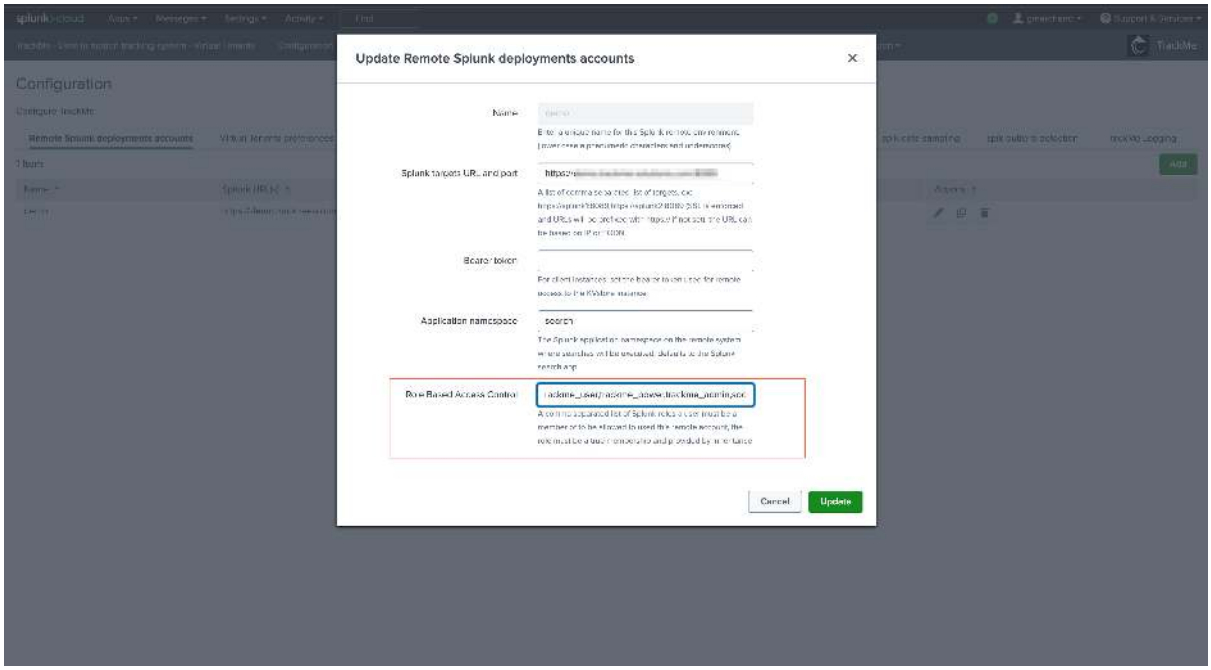
See: *Roles & access control* for more information about capabilities in TrackMe.

Manage Role-Based Access Control on the Account

User roles need to be listed in the account; by default when creating an account, the following roles are listed:

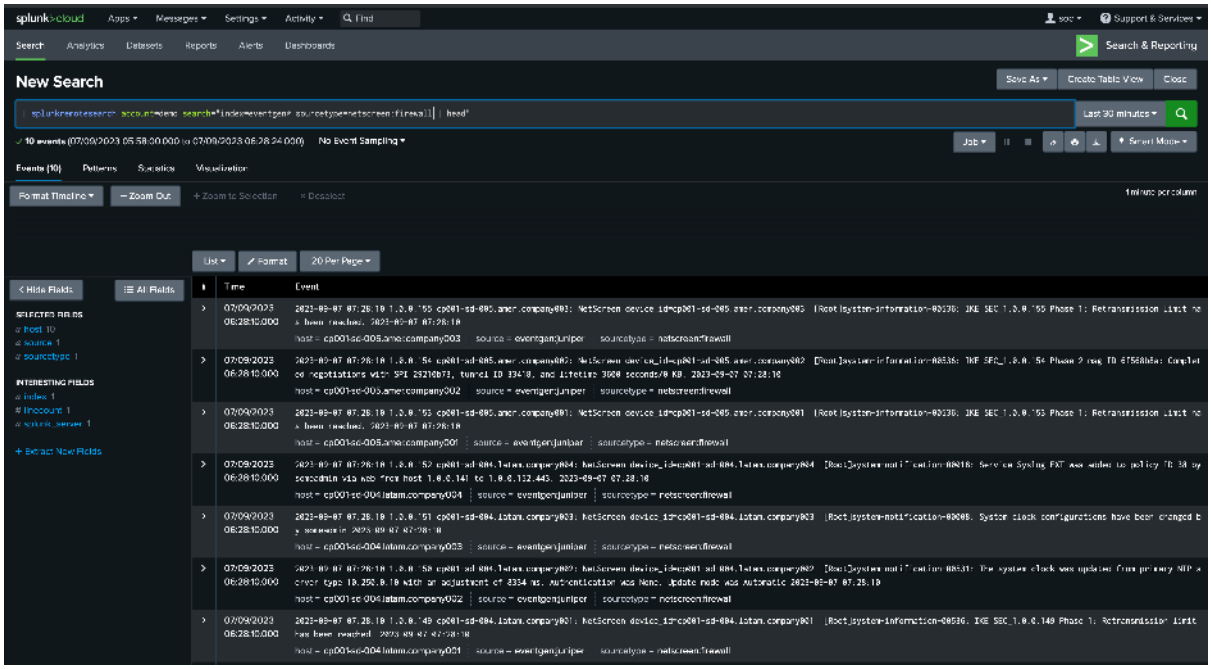
- admin, sc_admin, trackme_user, trackme_power, trackme_admin

This is an explicit membership requirement; in our example, we grant the user role "soc" to allow a member of this role to use the `splunkremotesearch` command against this account:



Use the Command

In our example, the "soc" user is a member of a role with the same name, and can now use the `splunkremotesearch` command as needed to perform searches against the remote deployment, from any other application namespace:



Accesses on the remote deployment rely on the service account which is associated with the bearer token configured in the account in TrackMe.

Troubleshooting Failure to Create or Update a Remote Account

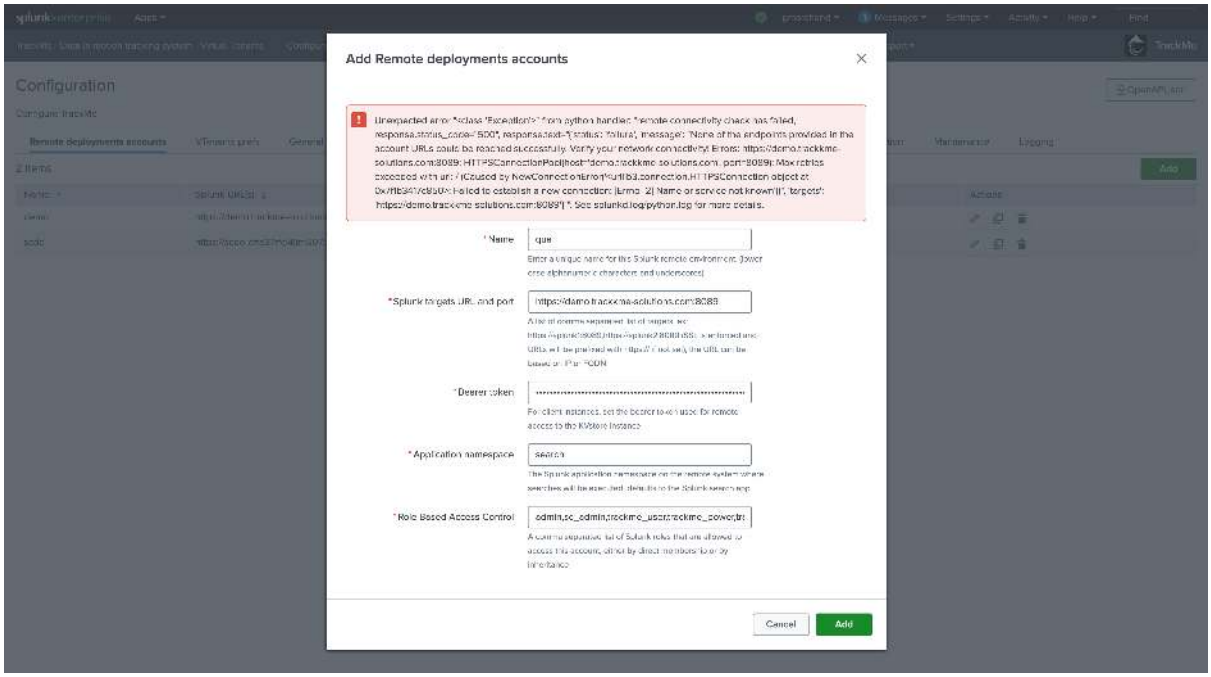
TrackMe Remote Search capabilities are an API-based integration with Splunk API relying on Bearer token authentication; you can refer to:

- https://docs.splunk.com/Documentation/Splunk/latest/Security/TroubleshootTokenAuth

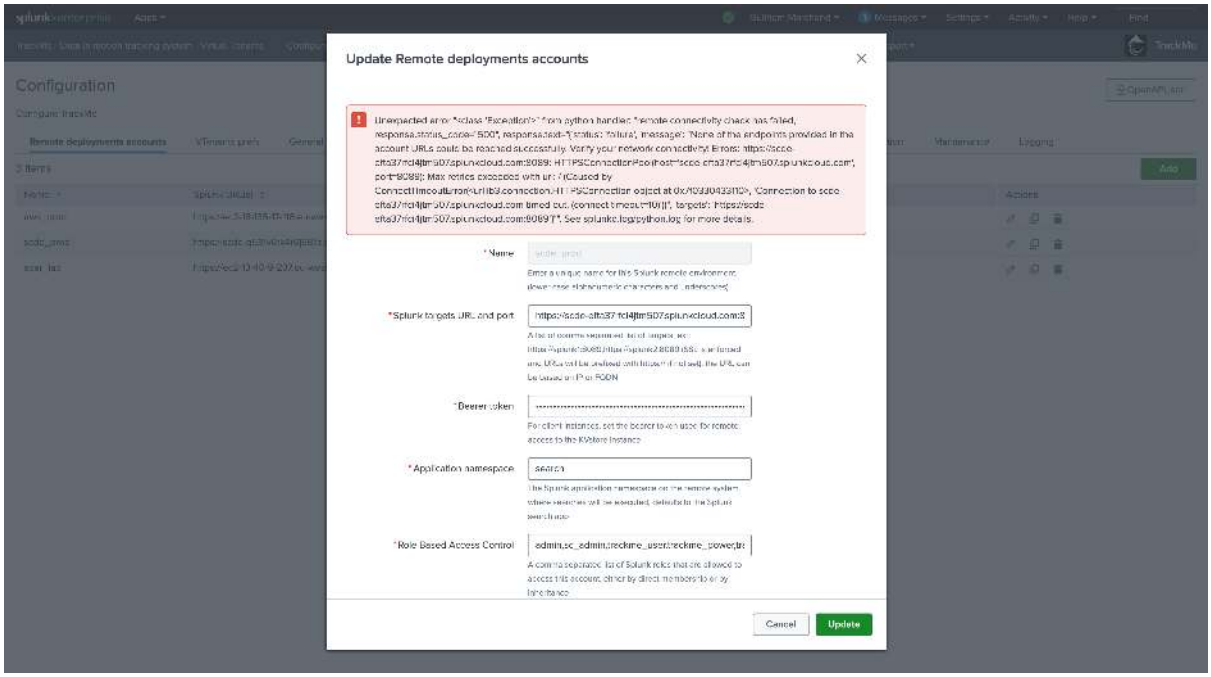
When attempting to configure a new remote account, or update an existing account, TrackMe will attempt to validate the connectivity, both from a network and authentication perspective.

If this step fails, the UI will clearly raise and show the exception encountered, for instance:

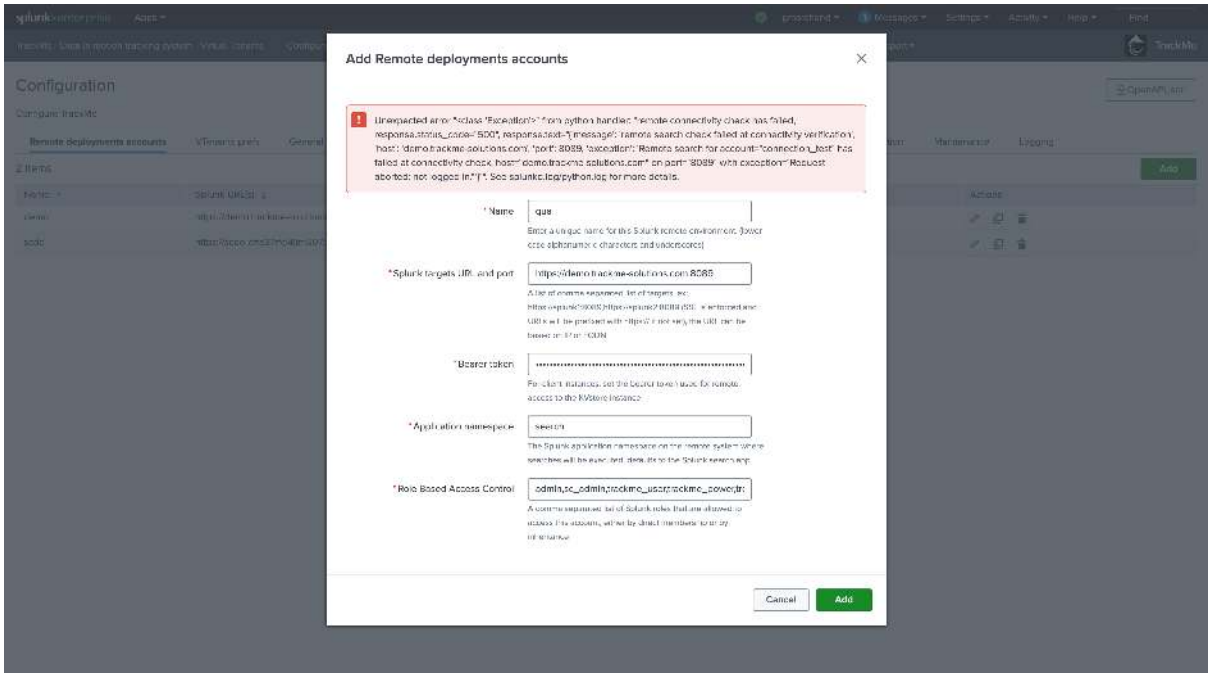
Example of a network connectivity failure: (wrong host)



Example of a network connectivity failure: (timeout)



Example of a refused authentication by the remote counterpart:



Note: You can also test the connectivity in pure SPL via the following TrackMe REST API endpoint:

```
| trackme mode=post url="/
↪services/trackme/v2/configuration/test_remote_connectivity" body="{
↪'target_endpoints': 'https://mysplunk.mydomain.com:8089', 'bearer_token': 'xxxx',
↪ 'app_namespace': 'search'}"
```

Inspect splunkd.log logs on the remote counterpart:

If the connectivity fails for authentication related reasons, such as a wrong token or lack of capabilities, verify the splunkd.log on the remote counterpart, you can use the following Splunk search:

```
index=_internal source=*/splunkd.log JsonWebToken
```

Messages will be logged by Splunk, eventually indicating the root cause of the failure.

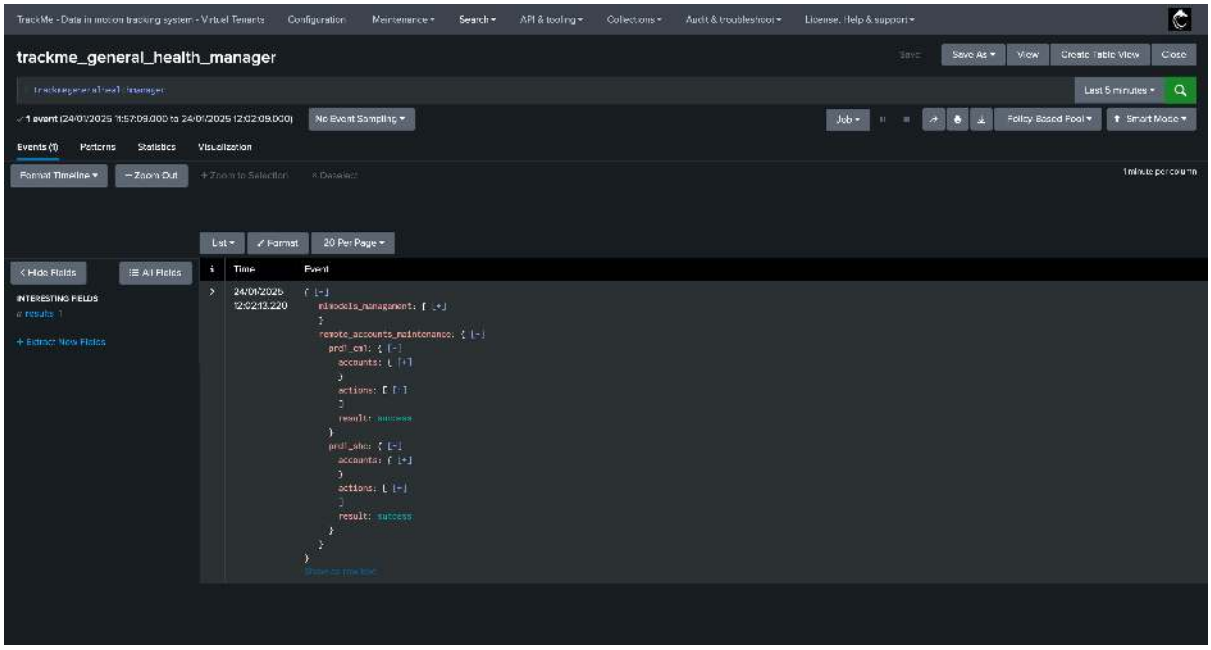
Example of a refused token:

```
02-05-2024 17:30:44.226 +0000 ERROR JsonWebToken [2843158 TcpChannelThread] -␣
↪JsonWebToken validation failed because: Token xxxx-xxxx has an invalid signature.
```

Bearer tokens automated rotation and revoking

Since TrackMe 2.1.8, we support automated rotation of the Splunk bearer tokens for remote accounts, the process works as follows:

- Once per day, a scheduled job called `trackme_general_health_manager` (General Health Manager tracker) is executed.
- **This job calls the endpoint `/services/trackme/v2/configuration/admin/maintain_remote_account` for each existing remote account, which does the following actions:**
  - Verify that the Remote account is up to date regarding available options, and update if necessary.
  - Verify if the Remote account has metadata available, and create/update if necessary. (stored in the KVstore `trackme_remote_account_token_expiration`)
  - If the Remote account bearer token has reached the rotation frequency, attempt to create a new bearer token, update the account and counters and finally revoke the previously generated token.



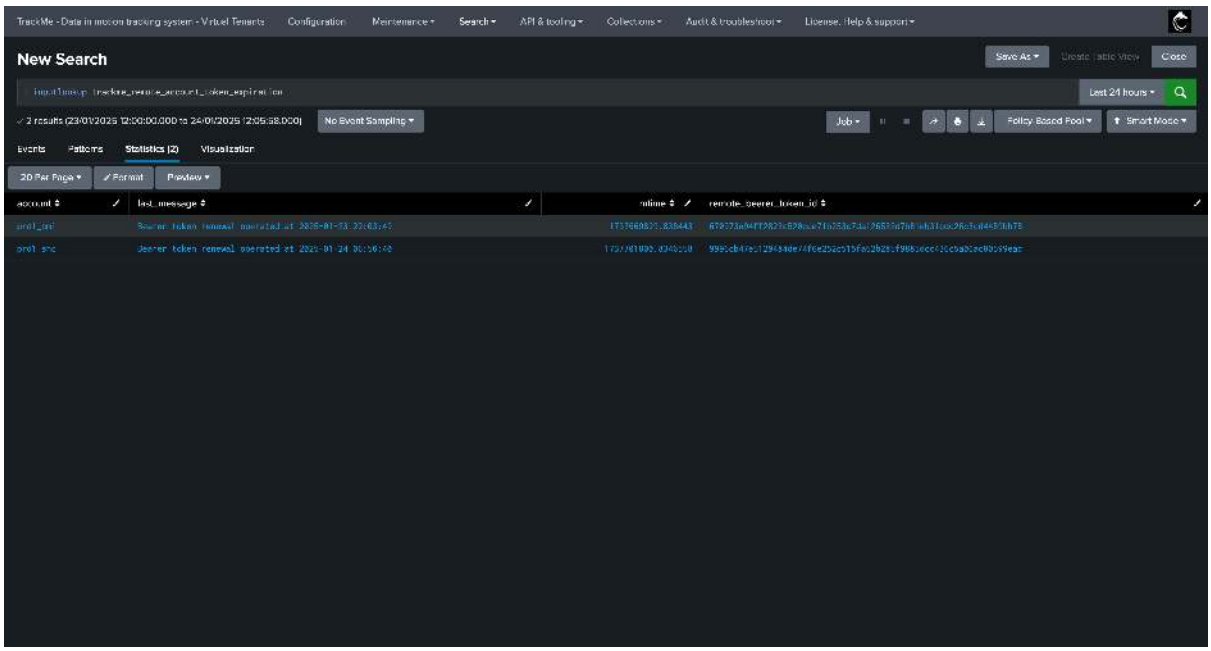
Notes about the initial token and rotation process:

- The initial token you have manually created will NOT be revoked by TrackMe, and would be disabled automatically depending on your settings.
- Once at least one rotation has been performed, and if it was successful, the initial token is not used anymore.
- From the stage of the first rotation, the previously generated and used token is automatically deleted by TrackMe, and the new token is automatically configured in the Splunk remote account.

Where to find bearer tokens rotation metadata?

TrackMe stores the metadata related to the bearer tokens rotation in the KVstore collection `trackme_remote_account_token_expiration`, you can access this collection using the following SPL search:

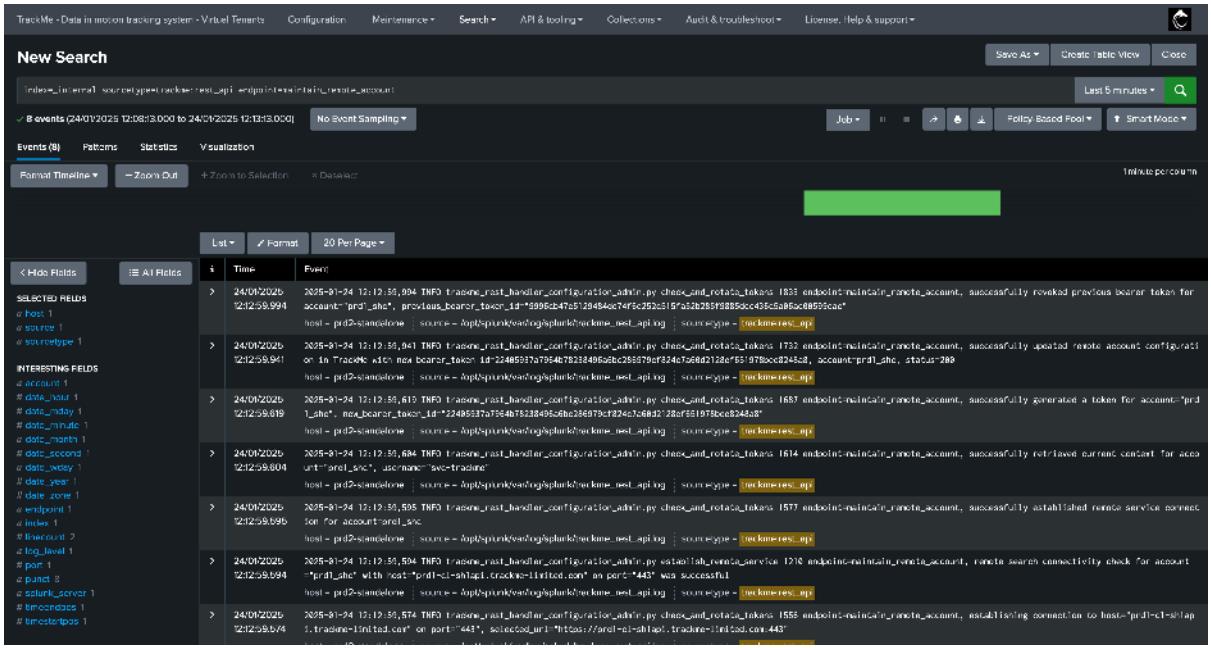
```
| inputlookup trackme_remote_account_token_expiration
```



How to access detailed logs about the tokens rotation process?

Logs are part of the TrackMe REST API logs:

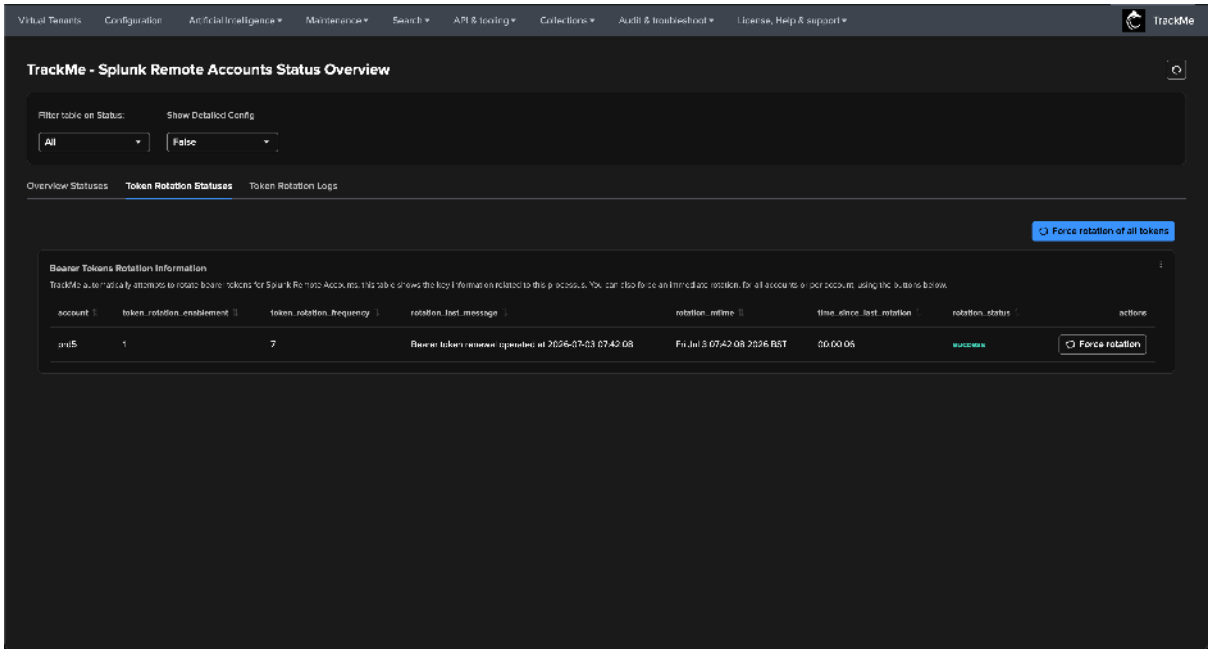
```
index=_internal sourcetype=trackme:rest_api endpoint=maintain_remote_account
```



How can I force the rotation of a bearer token?

From the user interface (since TrackMe 2.4.6):

You can force a rotation directly from the **Remote Accounts Overview** dashboard, in the **Token Rotation Statuses** tab — click **Force rotation of all tokens** to rotate every configured account at once, or use the per-account **Force rotation** action on a single row. A confirmation is requested; TrackMe then generates and validates a new token on the remote deployment, updates the account, revokes the previous token, refreshes the table, and reports the outcome (success, warning or failure). Forcing from the UI is equivalent to the REST call below with `force_tokens_rotation` set to `true` — it rotates regardless of the configured frequency.

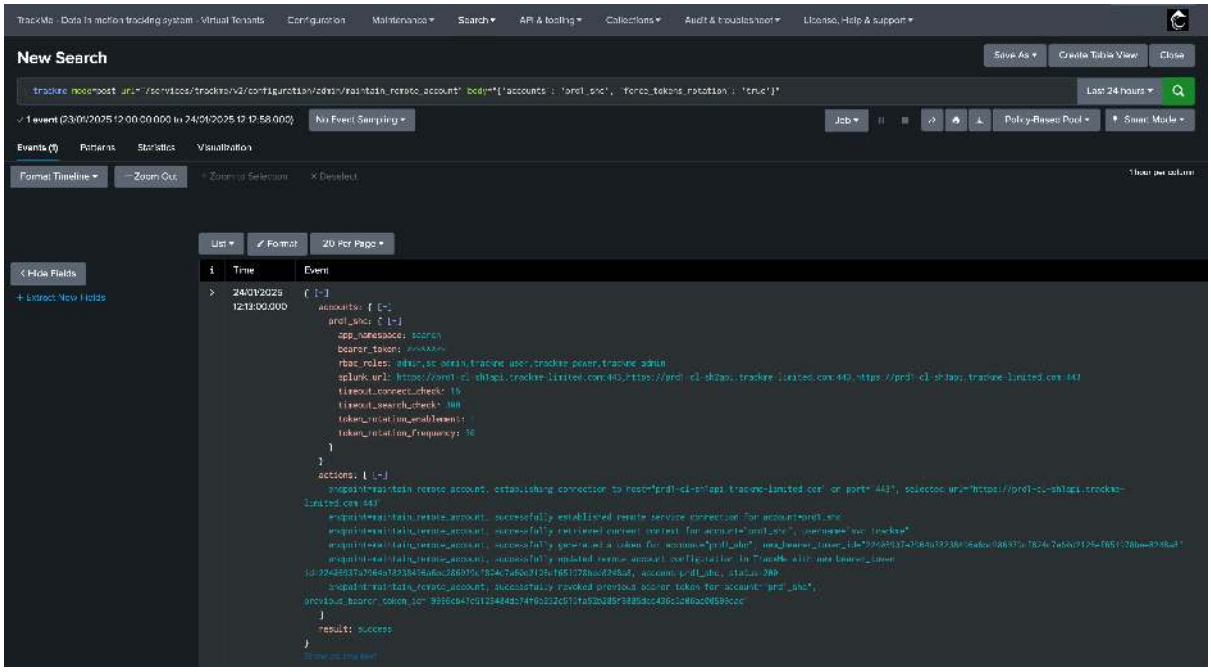


From the REST API (SPL):

The rotation of the bearer token can also be manually requested and performed by calling the TrackMe REST API endpoint, you can use the following SPL search:

```
| trackme mode=post url="/
↪services/trackme/v2/configuration/admin/maintain_remote_account" body="{'accounts
↪': '<account name>', 'force_tokens_rotation': 'true'}"
```

*Note: replace <account_name> with the actual name of the account, or use * to rotate all accounts.*



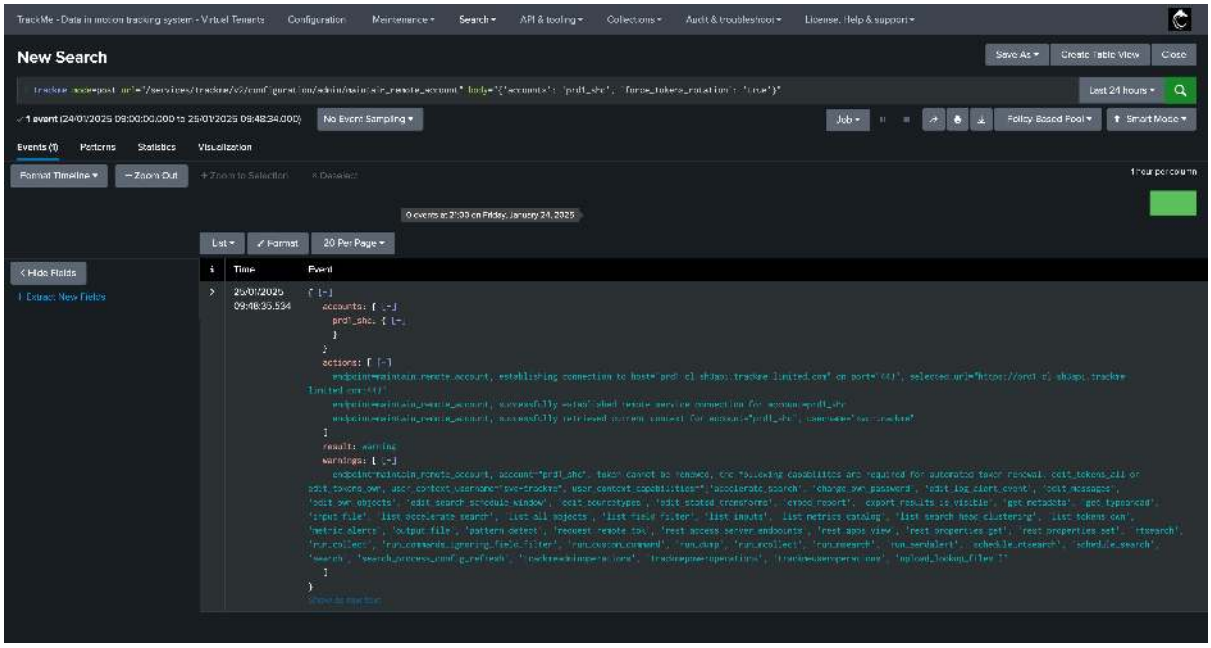
What if the token cannot be renewed?

If for some reasons, the token cannot be renewed, the process will be logged in the TrackMe REST API logs, and the token will not be revoked, the previous token will still be used:

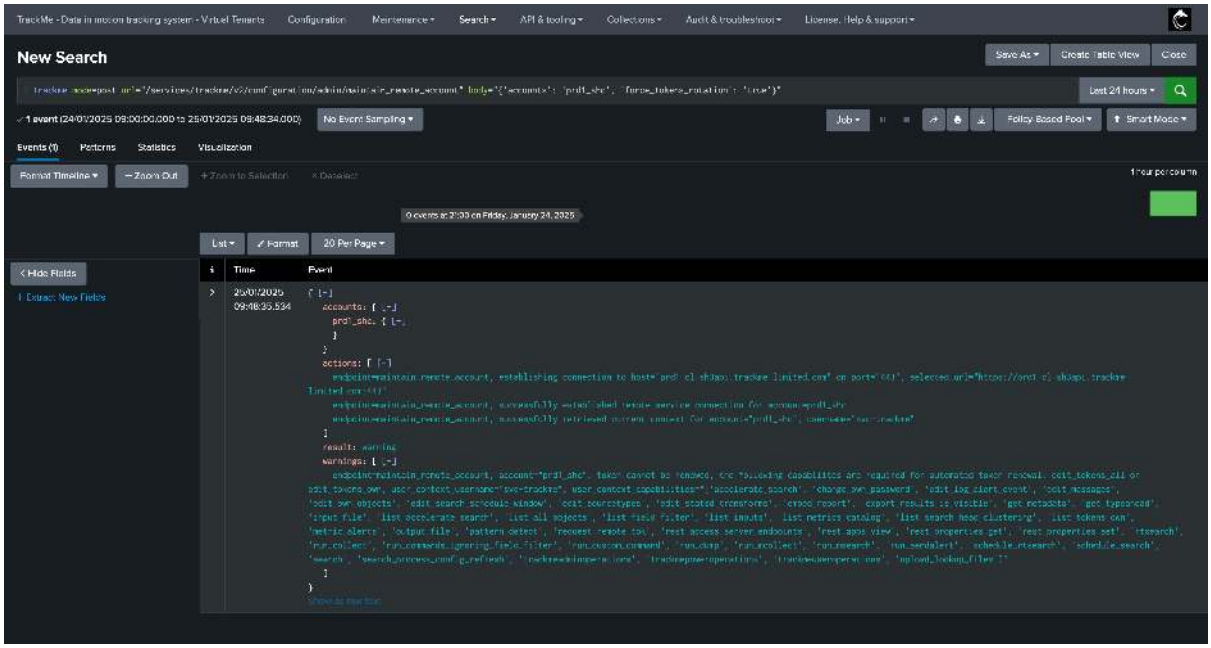
The following search will show errors or warnings related to the tokens rotation process:

```
index=_internal sourcetype=trackme:rest_api endpoint=maintain_remote_account␣
↪(log_level=ERROR OR log_level=WARN)
```

forced rotation is requested but failed:



In this example, the remote user account lacks the capability to create a new token, the rotation process is aborted.



Splunk Cloud remote members auto-healing

Since **TrackMe 2.4.6**, TrackMe can automatically keep the target URL list of a **Splunk Cloud** remote deployment account in sync with the live members of its search head cluster. Splunk Cloud periodically destroys and recreates search-head-cluster (or Enterprise Security) members, changing their hostnames — so the URLs configured on a remote account start pointing at hosts that no longer exist and the searches that rely on the account begin to fail. This capability detects that drift and, when enabled, repairs the account for you.

This feature is specific to Splunk Cloud

An account is eligible only if at least one of its members is a Splunk Cloud host — a hostname ending in `splunkcloud.com`. Accounts targeting Splunk Enterprise or any non-Cloud endpoint are ignored.

How it works

Once per day the **General Health Manager** (`trackme_general_health_manager`) evaluates every eligible remote account:

- **Local discovery.** It discovers the stack's current live members **locally** by searching the `_cmc_summary` index (Cloud Monitoring Console) — no remote call, which matters because the remote connection is precisely what breaks when members are recreated. The ad-hoc search head can see every search-head group of the same Cloud stack locally.
- **Group association.** It identifies which search head group the account belongs to (the `search_head_names` value from `_cmc_summary`) and persists the association in the KV Store collection `trackme_remote_account_member_monitoring`.
- **Reconciliation.** It compares the account's configured members with the live group: a live member missing from the account is **added**; a configured member that is no longer in the live group **and** is unreachable is **removed**; a member missing from the live group but still reachable is **kept** (to ride out a transient outage).
- **Auto-heal.** If auto-healing is enabled and a difference is found, TrackMe updates the account's target URL list — connectivity-tested before saving, so an account is never updated to an unreachable set of members.

Note

If TrackMe runs on **Splunk Enterprise** rather than inside a Cloud stack, `_cmc_summary` is not available locally, so the check simply skips the account and logs an informational message.

Enabling it (per account)

Auto-healing is controlled per account by **Splunk Cloud member auto-heal**, in the *Splunk Cloud self-healing* section of the account configuration (**Configuration → Remote deployments → edit the account**):

- **Enabled (default)** — TrackMe detects the drift and updates the target URL list automatically.
- **Disabled** — TrackMe still detects and reports the drift, but never modifies the account; you apply the change manually.

How drift and healing are surfaced

The outcome is surfaced through the *Configuration Guardian*:

- When an account is **auto-healed**, an **informational** (blue) notice summarises the members added and removed, and stays until you dismiss it.
- When drift is **detected but not healed** (auto-heal disabled, or the group could not be determined), a **warning** alert is raised — including the recommended target URL list to apply manually — and clears automatically once the account is back in sync.

Verifying and running the check

Review the live members of each search head group yourself with the same search TrackMe uses (evaluated over the last 24 hours):

```
search index=_cmc_summary search_head_names=* usage_source=* search_head_names!=
↪"idx"
| stats values(usage_source) as search_heads by search_head_names
```

Inspect the persisted account-to-group associations:

```
| inputlookup trackme_remote_account_member_monitoring
```

And trigger the check on demand (it otherwise runs once per day):

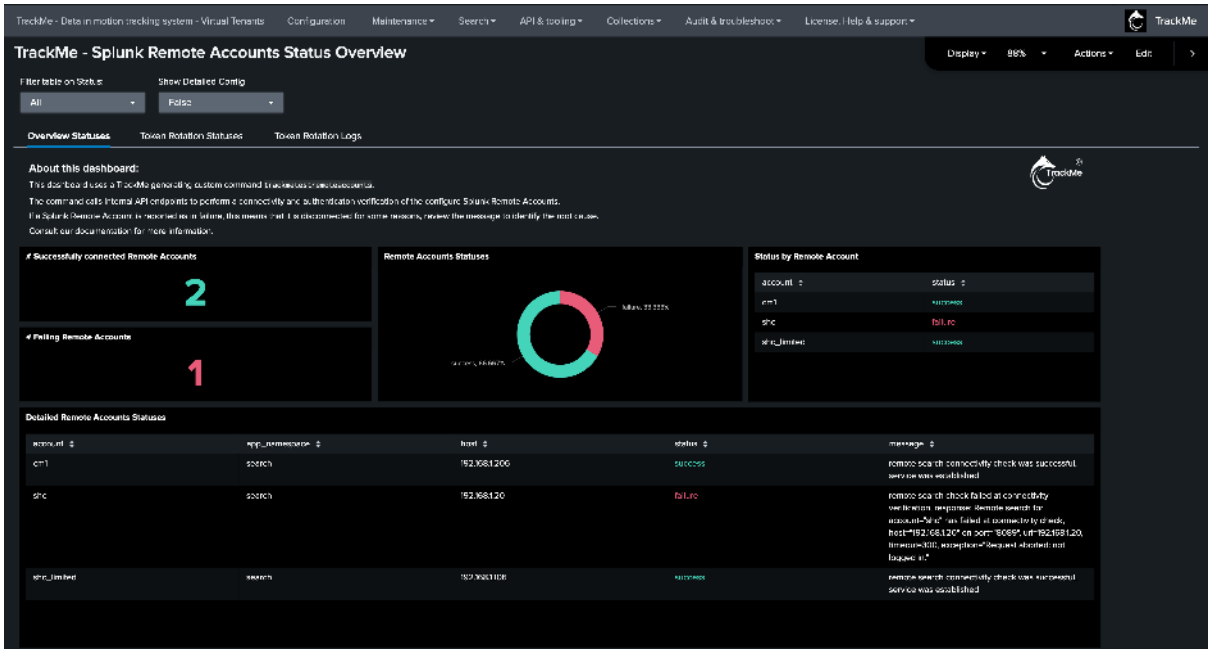
```
| trackme mode=post url="/
↪services/trackme/v2/configuration/admin/run_guardian_checks" body="{'check_type
↪': 'remote_account_member_drift'}"
```

Note

The exact `_cmc_summary` field names depend on your Splunk Cloud stack's schema; the search above reflects TrackMe's default — review `index=_cmc_summary` and adjust if your environment differs.

Audit Dashboard: Splunk Remote Accounts Overview

Access the Audit dashboard from the menu `Audit & Troubleshooting` -> `Audit - Splunk Remote Accounts Overview`.



Remote Accounts Commands Examples and References

Get the list of all configured remote accounts

This provides a list of all configured remote accounts, excluding the local account:

```
| trackme mode=get url="/services/trackme/v2/configuration/list_accounts"
| spath
| rename "accounts{}" as accounts
| table accounts
| mvexpand accounts
| where accounts!="local"
```

Test the Connectivity of all configured remote accounts

This command will test the connectivity of all configured remote accounts:

```
| trackme mode=get url="/services/trackme/v2/configuration/list_accounts" | spath
| rename "accounts{}" as accounts
| table accounts
| mvexpand accounts
| where accounts!="local"

| map maxsearches=100 search="| trackme mode=post␣
↪url=/services/trackme/v2/configuration/test_remote_account body=\"{'account': '
↪$accounts$'}\""
| trackmeprettyjson fields=_raw
| spath
```

Get the detailed including the bearer token for all configured remote accounts

This command will provide the detailed information about all configured remote accounts, including the bearer token:

```
| trackme mode=get url="/services/trackme/v2/configuration/list_accounts" | spath
| rename "accounts{}" as accounts
| table accounts
| mvexpand accounts
| where accounts!="local"

| map maxsearches=100 search="| trackme mode=post␣
↪url=/services/trackme/v2/configuration/get_remote_account body=\"{'account': '
↪$accounts$'}\""
| trackmeprettyjson fields=_raw
| spath
```

Force the rotation of the bearer token for all configured remote accounts

This command will force the rotation of the bearer token for all configured remote accounts in a single command:

```
| trackme mode=get url="/services/trackme/v2/configuration/list_accounts" | spath
| rename "accounts{}" as accounts
| table accounts | mvexpand accounts
| where accounts!="local"

| map maxsearches=100 search="| trackme mode=post␣
↪url=/services/trackme/v2/configuration/admin/maintain_remote_account body=\"{
↪'accounts': '$accounts$', 'force_tokens_rotation': 'true'}\""
| trackmeprettyjson fields=_raw
```

Get the token expiration metadata for all configured remote accounts and the time since the last rotation

This command will provide the token expiration metadata for all configured remote accounts and the time since the last rotation:

```
| inputlookup trackme_remote_account_token_expiration | eval keyid=_key
| eval _time=mtime
| eval time_since=tostring(round(now()-mtime), "duration")
```

8.9.3 TrackMe Broker

TrackMe's integrations make API calls: to a remote Splunk account on a Heavy Forwarder or a Deployment Server, to a self-managed Cribl Leader, to a webhook receiver on your network. From a Splunk Cloud stack those destinations cannot be reached, because the stack cannot connect into your network. A relay would not help: the connection would still start in the Cloud.

TrackMe Broker reverses the direction. A small Splunk add-on on a Heavy Forwarder on your side asks the stack for work, over the stack's REST API on port 8089, makes the HTTP call locally, and posts the answer back. Every connection starts on your side. Nothing connects in, and you open no inbound firewall rule.

An account simply says how it connects: directly, or through a broker. The code that makes the call does not change.

Note

TrackMe Broker needs TrackMe 2.4.24 or later on the stack and, on the forwarder, Splunk Enterprise 9.4.x or later with the TrackMe Broker add-on 1.1.1 or later, available on Splunkbase. It is compatible with Splunk *FIPS mode*. This page explains the concept and how to set it up. *TrackMe Broker — manage and monitor* goes through the two pages, everyday tasks and alerts. *TrackMe Broker — in depth* follows one call from end to end, with its timeouts, leases and failure codes. *TrackMe Broker — security and cryptography* covers pairing, key exchange, every cipher and the threat model.

How it works

There is no tunnel. splunkd gives an app request and response only: one reply per request, with no WebSocket and no way to push into an open connection. So work reaches a forwarder as the answer to its own request, and the two sides meet in the stack's KV store.

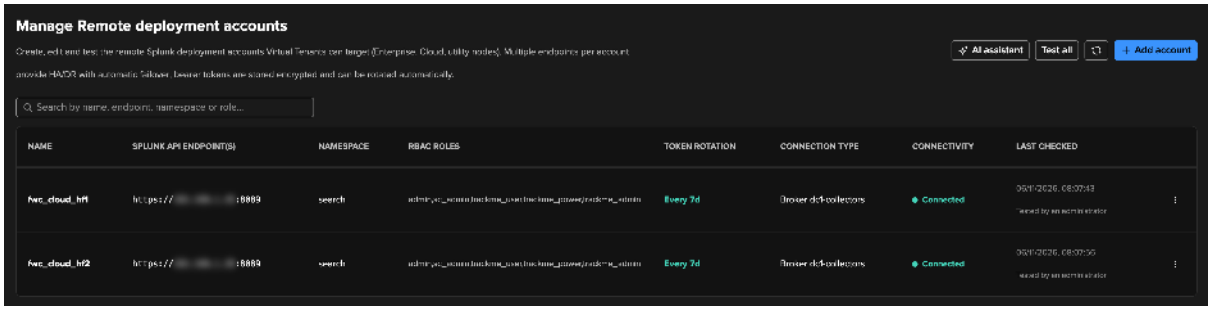
1. A TrackMe feature makes an HTTP call for an account whose **Connectivity** names a broker. The transport turns that call into an *envelope*, and TrackMe queues it, encrypted, in the KV store.
2. The add-on on the forwarder holds a long poll open on the stack, a signed `pull`. As soon as a request is queued, the stack answers that poll with it, encrypted for this forwarder alone.
3. The add-on checks that the destination is among the *targets* declared on this forwarder, then makes the call. While it runs, it renews its claim on the request.
4. The add-on sends the response back in encrypted chunks. TrackMe reads them, and the feature receives an ordinary HTTP response, or a failure of the kind a direct call would have raised.

The stack's side is a control plane: it queues, hands over and stores. The HTTP call itself only ever leaves from the forwarder, and only to a destination declared on that forwarder.

What can go through it

Destination	Set **Connectivity** in	What the broker may be asked to reach
Remote Splunk account (a Heavy Forwarder, a Deployment Server, an on-premises search head)	Configuration → Manage Remote deployment accounts	The account's Splunk URLs. A URL without a port stands for both 443 and 8089, as on a direct connection.
Self-managed Cribl Leader	Configuration → Manage Cribl connections, for a **Self-managed Cribl** connection	The connection's Leader URL. A Cribl Cloud connection stays direct.
Webhook receiver	Configuration → Manage Webhook accounts	The account's URL or secret URL, and its OAuth2 token endpoint.

See *Remote Splunk deployments*, *A self-managed Leader the stack cannot reach* and *A receiver the stack cannot reach* for these accounts. The three Manage pages named above also show a **Connection type** column, reading Direct or Broker and its name. **Connectivity** in those lists stays the result of the last test.



The pieces

On the stack:

Piece	What it does
The transport	Each account has a `connectivity` field: `direct`, the default, or `broker:<name>`. For a broker, one piece is swapped under the client (a `requests` adapter, a splunklib handler or the outbound client's transport). The calling code stays as it is. A value that cannot be read is refused. It never falls back to a direct connection.
The store	Seven KV collections hold the brokers, the forwarders, what they report, the queued requests, the claims, the response chunks and the pairings. Requests and responses are encrypted before they are written.
The channel	The operations a forwarder calls on port 8089: `register`, `pairing_status`, `pull`, `progress`, `push`, `fail` and `rotate_key`.
Manage TrackMe Brokers	Configuration → Manage TrackMe Brokers. Add broker from a pairing code, see each forwarder with its fingerprint, test a broker, remove a forwarder, and read the exchanges in flight and the recent failures.
Configuration Guardian	Two checks, `broker_offline` and `broker_degraded`, see *Configuration Guardian alerts*.

On the forwarder:

Piece	What it does
The modular input	One long-running process per connection. Saving a connection creates it, and removing the connection removes it. It is the loop that pulls, and it starts the calls.
The targets	What this forwarder may reach: deny by default. Each target also decides TLS verification for its destination. The stack cannot widen the list and never receives it, only a digest and two counts.
The executor	Makes one HTTPS call: no redirects, no ambient proxy, bounded in time and size. It checks the address it actually reached before anything is sent.
Identity and keys	The forwarder's own keys, and the stack's public keys it pinned, in this forwarder's `storage/passwords`. The forwarder's keys rotate on their own every 30 days. The stack's do not.
The add-on's page	Connections, the pairing code and fingerprint, targets, and a test for a target.

What you need

Requirement	Detail
A Heavy Forwarder	We recommend two dedicated instances, see *Where to run the broker*. Splunk Enterprise 9.4.x or later, on your side, able to reach both the stack and the destinations. The add-on refuses to start on a Splunk Cloud instance. It does not package the `cryptography` library: it uses the one in Splunk's own Python. Without it, the add-on's page says so, and the connection does not run.
HTTPS to the stack's management port	On Splunk Cloud, add the forwarder's egress address to the stack's `search-api` allow list, through the Admin Config Service (ACS). The stack URL is the management port: `https://<stack>.splunkcloud.com:8089`. Splunk Web is not a route to the channel: it requires its own session, so it answers a token alone with a redirect to its login page, which the add-on reads as a refusal.
A Splunk account on the stack	Holding only the role `trackme_broker`, with an authentication token. See below.
An HTTPS destination	Plain HTTP is refused.

The role `trackme_broker` holds one capability, `trackmebrokeroperations`. It imports no other role and has no index, and the rights Splunk grants every role by default that a broker has no use for are turned off. So the role is built to reach the channel and nothing else. Behind that capability, every operation is limited to the broker that the signed call names, and `register` and `pairing_status` are the only ones a token alone can use. The `admin` and `sc_admin` roles hold the capability too, but a broker should not run on them.

The token you create starts the connection, and TrackMe then renews it: see *The forwarder's token*. With a token the stack refuses, because it expired or was revoked, the add-on's page says the stack refused the token, its log reads `disconnected:http_401`, and in time Configuration Guardian raises `broker_offline`. To replace it, edit the connection on the add-on's page: a changed token applies within a minute, with no restart.

Where to run the broker

We recommend dedicated Heavy Forwarders: two instances that run the add-on and nothing else. A forwarder that already has another role, such as collecting data or serving as a Deployment Server, can host the add-on too.

In both cases the add-on runs in its own process and sends no events through Splunk's data pipeline. A blocked forwarding queue does not stop the broker, and a broker that stops does not affect forwarding. What the roles share is the host:

What it affects	Dedicated forwarders (recommended)	A forwarder shared with another role
Memory	The host holds Splunk and the broker. **Concurrent requests** times **Largest response (MB)** is what the broker may hold at once: about 1 GB with the defaults of 4 and 256.	The same amount comes on top of the other role. Check that the host has room, or lower these settings on the connection's edit dialog.
Restarts	Only your maintenance of the broker's own hosts stops it.	Anything that restarts Splunk for the other role, such as a configuration push, stops the broker as well.
Who controls it	Only the administrators of these two instances decide what the broker may reach.	Whoever administers the other role's instance also decides what the broker may reach.
Targets	Every target is another host.	A target can be this forwarder's own management port, which needs care on a pair: see below.

If you share a forwarder

- **Name hosts rather than `localhost` on a broker with two forwarders.** Either forwarder can take a call, so `https://localhost:8089` reaches whichever one pulled it, and the same account reaches a different instance from one call to the next. TrackMe does not detect this, because both target lists are identical. `localhost` only suits a broker with one forwarder.
- **On a Deployment Server,** install the add-on directly in `$SPLUNK_HOME/etc/apps`, as on any instance. The `deployment-apps` directory only holds what the server sends to its clients.

Whichever you choose

- **Use two forwarders.** One forwarder is a single point of failure for every account behind the broker, not only for its own role. See *Two forwarders, one broker*.
- **Each forwarder must reach every target,** including the other forwarder's management port if an account targets it.
- **If a Deployment Server delivers the add-on, protect its `local` directory.** The connection, its keys and its token are stored there, in `inputs.conf`, `trackme_broker_stacks.conf` and `passwords.conf`, and the targets in `trackme_broker_targets.conf`. A Deployment Server update deletes or overwrites what an app generated locally, so the forwarder would lose its connection and have to be set up and accepted again. Exclude the directory in `serverclass.conf`, as described in Protect content during app updates:

```
[serverClass:<your_class>:app:trackme_broker]
excludeFromUpdate = $app_root$/local
```

- **Forward its logs to the stack,** on a dedicated forwarder too. That is how `sourcetype=trackme:broker` reaches the `_internal` index you search when a broker has a problem.
- **Splunk Web must be on while you set up or change the connection,** because the add-on's page is served by it.
- **A universal forwarder cannot run the add-on,** which needs Splunk Enterprise. We have not tested a search head cluster member or an indexer, and do not recommend them.

Set it up

1. **On the stack,** allow the forwarder's address on `search-api`, create the Splunk account holding only `trackme_broker`, and create a token for it.

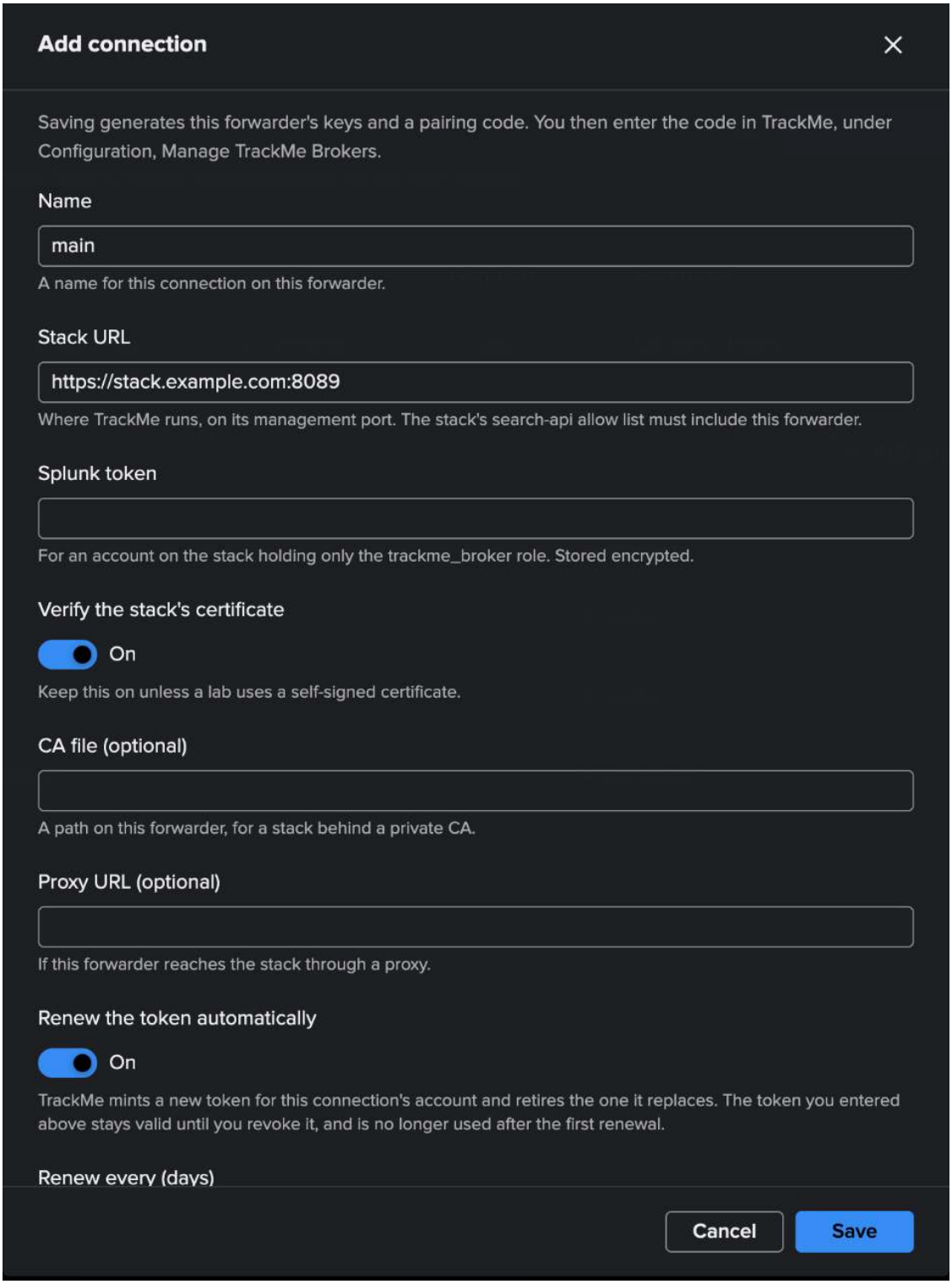
Warning

In Splunk's New User form, untick **Create a role for this user**. Left ticked, the user gets a role of the same name with five extra capabilities, instead of `trackme_broker`. Deleting that role afterwards can remove the user as well, and a token made for that user does not work again on a new user of the same name.

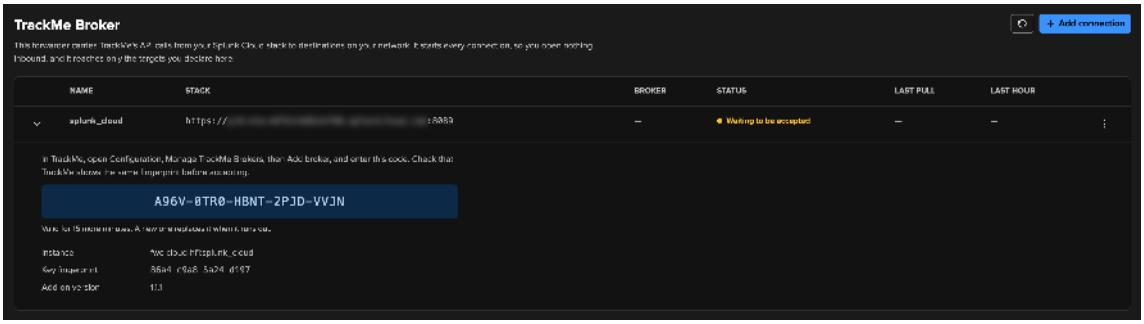
2. **On the forwarder,** install the TrackMe Broker add-on and open its page. It is published on Splunkbase, so you can install it from **Apps > Find More Apps** on a forwarder with internet access. Its releases are also on TrackMe Solutions Downloads, and *its release notes* give each package and its SHA-256.

3. Select **Add connection** and fill in:

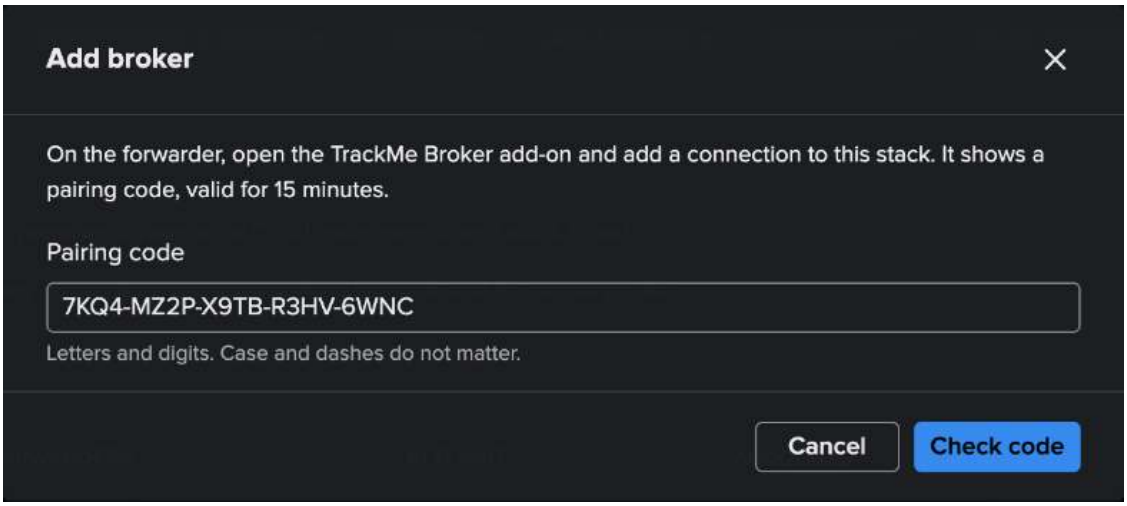
Field	What to enter
Name	A name for this connection on this forwarder
Stack URL	`https://<stack>.splunkcloud.com:8089`
Splunk token	The token you created. It is stored encrypted
Verify the stack's certificate	On by default. Keep it on
CA file (optional)	A path on this forwarder, for a stack behind a private CA or a TLS-inspecting proxy
Proxy URL (optional)	See *Through an egress proxy*
Renew the token automatically	On by default. See *The forwarder's token*
Renew every (days)	7 by default, from 1 to 365



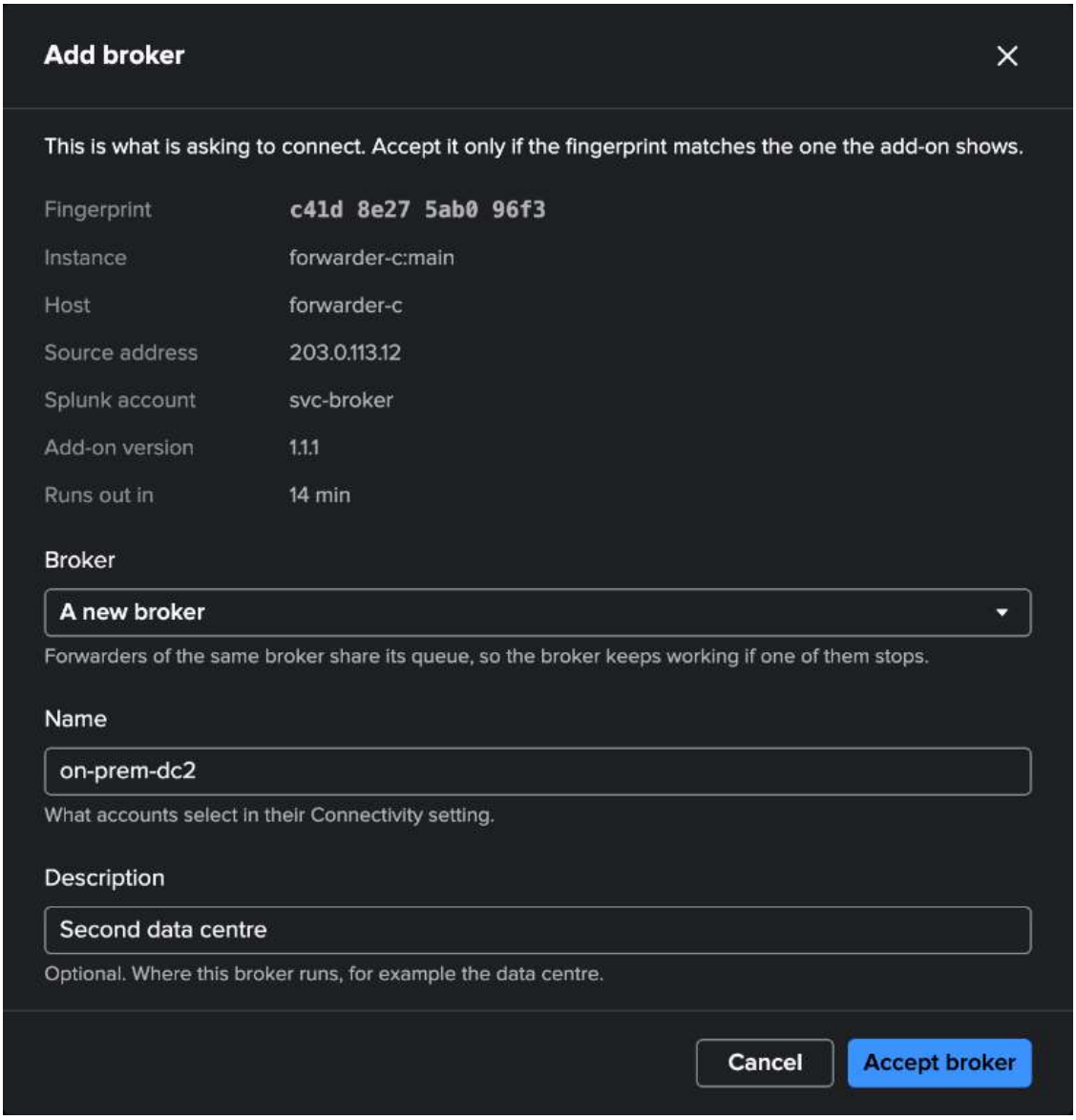
Saving creates the input. It generates the keys, registers them with the stack, and the page shows a pairing code, valid for 15 minutes, and a key fingerprint. A new code replaces the old one when it runs out.



4. **In TrackMe,** open **Configuration → Manage TrackMe Brokers** and select **Add broker**. Enter the code and select **Check code**. Case and dashes do not matter.



5. TrackMe shows what is asking to connect: the fingerprint, the instance, the host, the source address, the Splunk account and the add-on's version. **Compare the fingerprint with the add-on's page.** If it differs, stop. Nothing has changed yet. On Splunk Cloud, the source address is the load balancer's, so do not read it as your forwarder's.

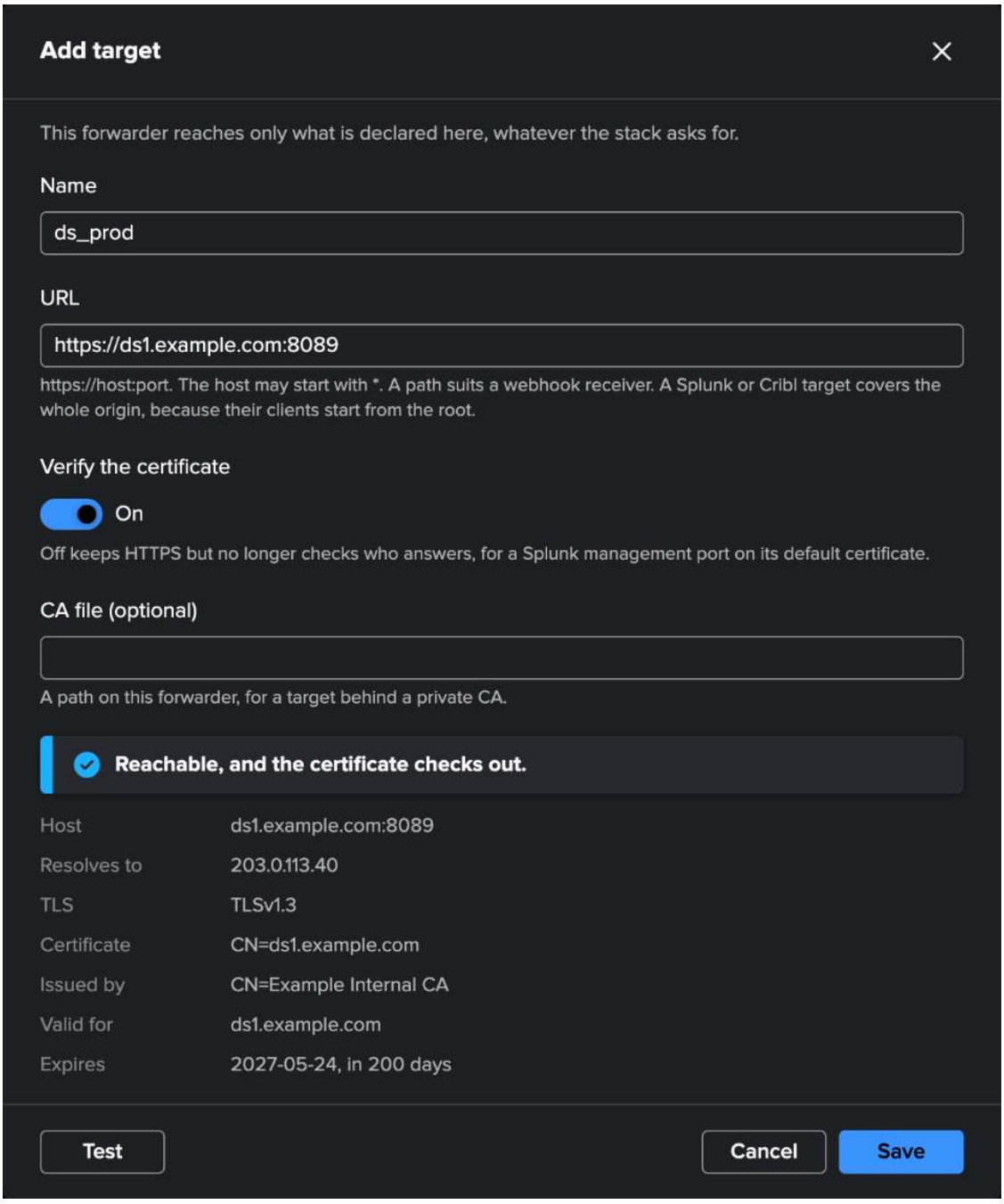


6. Choose **A new broker** and name it, or **Add to** an existing broker to make this forwarder part of a pool. Select **Accept broker**. The add-on reads Connected at its next check, which comes every 5 seconds. It then renews its token by itself, so the token you created is no longer used: see *The forwarder's token*.

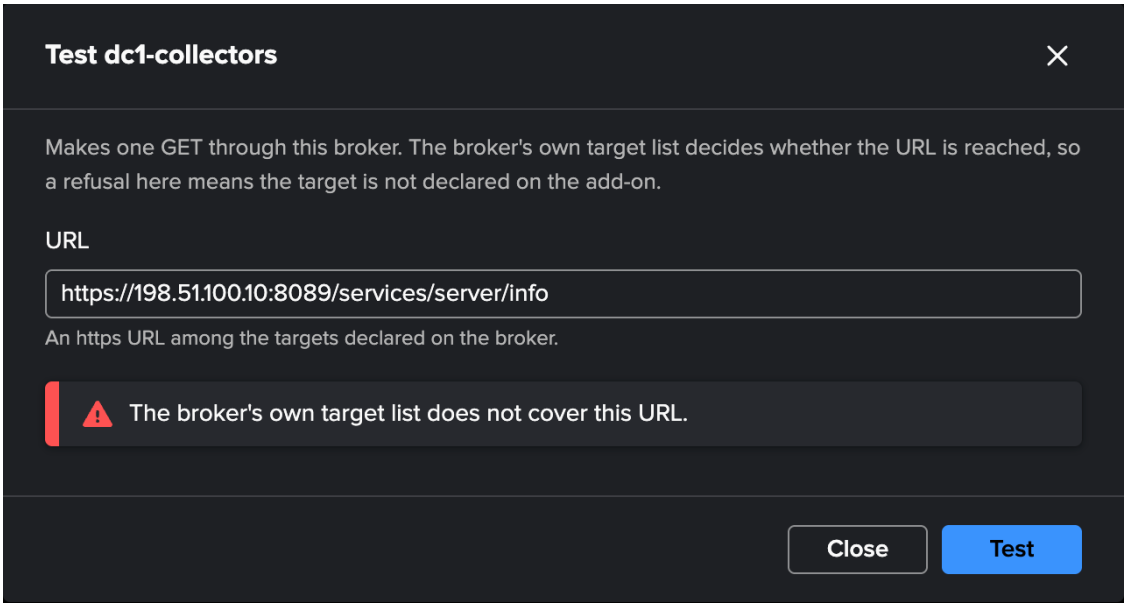
7. **On the forwarder,** select **Add target** for every destination this broker may reach. Nothing else is reachable, whatever the stack asks:

Field	What to enter
Name	A name for this destination
URL	`https://host:port`, optionally with a path. The host may start with `*.` for any subdomain. A path suits a webhook receiver, such as `https://hook.example.com:8443/api/hooks`. A Splunk or Cribl target covers the whole origin, because their clients start from the root. `https://localhost:8089` is this forwarder's own management port, which only suits a broker with one forwarder
Verify the certificate	On by default. Off keeps HTTPS and no longer checks who answers, which is for a Splunk management port on its default certificate
CA file (optional)	A path to a PEM file on this forwarder, for a private CA. The Splunk user must be able to read it

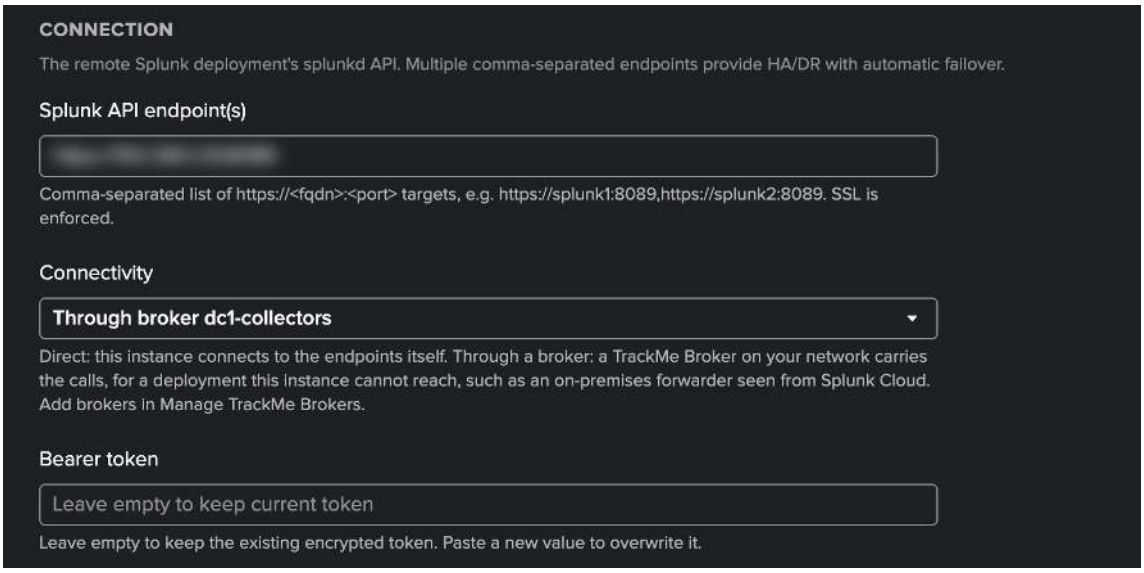
Test in the same dialog connects from the forwarder and shows what the destination presents and whether its certificate verifies. It sends nothing but a TLS handshake. A new target applies within a minute, and during that minute a call can read `target_not_allowed`.



8. **In TrackMe,** on the broker's menu, select **Test** and enter a URL among the targets. TrackMe makes one GET through the broker, and shows the target's status, time and size, or a sentence saying why it failed.



9. Set **Connectivity** on the account, and save it. A remote account is tested through the broker as it saves. A Cribl connection has **Save and test** beside **Save**. A webhook account is tested as it saves, when it is enabled.



If a call is refused because no target covers it, the add-on's page lists the destination under **Refused in the last hour**, by host and port, and offers **Add as target** for you to confirm.

Important

TLS verification lives on the target. Through a broker, the account's own TLS, proxy and allowed-destination options are not used, and those pages hide them: only the forwarder sees that destination. The forwarder reports a count of its targets without verification, never the list. Manage TrackMe Brokers shows it on the forwarder's row, as a note, since verification off is a deliberate choice for a Splunk management port on its default certificate.

Through an egress proxy

Proxy URL on a connection, `http://host:port`, sends the forwarder's calls to the stack through that proxy. Proxies from the environment are never used. The proxy concerns the way to the stack only: calls to targets go straight from the forwarder.

If that proxy inspects TLS, the forwarder sees the proxy's certificate instead of the stack's. Put the proxy's CA

in the connection's **CA file** and keep verification on. With the wrong CA the connection fails and does not fall back: the add-on logs `disconnected:unreachable:SSLError`. A proxy that is down reads as `disconnected:unreachable:ProxyError`, until it returns.

An inspecting proxy reads the outer call and not its content: see *what a TLS-inspecting proxy sees*.

Two forwarders, one broker

Repeat the forwarder's steps on a second forwarder, and choose **Add to** the same broker. Both pull from the same queue, so both share the work, and when one stops the other carries on with no failover step. An account binds to the broker's name and never to a host.

Give them the same targets. Each forwarder holds its own list, so a call could otherwise work or fail depending on which one pulled it. TrackMe compares a digest of the lists, and flags the difference. Name hosts rather than `localhost`, which means a different machine on each forwarder.

Keep **Largest response (MB)** the same too, for the same reason: the lower of the stack's limit and the forwarder's wins, and the caller sees `response_too_large`. TrackMe does not compare this setting.

Manage TrackMe Brokers shows the state of the pool: **Online**, or **Online, 1 of 2** while one is down. Configuration Guardian warns as well, so a pair does not run on one forwarder unnoticed until the second one fails.

The forwarder's token

The forwarder's Splunk token renews itself. TrackMe on the stack mints a new token for the same account, at the signed request of this forwarder, and retires the one it replaces. The role needs no extra capability, and a stolen token alone cannot do this, because renewal needs the forwarder's signing key.

When it renews	Where you set it
Right after pairing, so each forwarder holds a token of its own	**Renew the token automatically**, on the connection
Every 7 days by default	**Renew every (days)**, on the connection, from 1 to 365
When a token that states an expiry has under 30% of its life left	**Renew the token automatically**
When you ask	**Renew token now**, in the connection's menu

Token renewed, in the connection's details, says when it last did. A failure at any step leaves the token in use working. It is tried again every hour, and **Token renewal** says why it has not worked yet, without the connection reading as failed. TrackMe mints on the stack's system token. On Splunk Cloud this works with no capability added to the role: `trackme_broker` alone is enough. If Splunk ever refuses, only renewal stops.

Important

You can revoke the token you created after the first renewal. TrackMe never revokes a token you entered: nobody here knows its id, and two forwarders may share it. It stays valid until you revoke it, and it is no longer used once **Token renewed** shows a time on every forwarder that used it.

Remove on a forwarder's row in Manage TrackMe Brokers, or deleting its broker, revokes what TrackMe minted for that forwarder, and so does the stack when it drops a forwarder silent for 30 days. Bringing it back then takes a new token in its connection, and then a new code. A forwarder that never renewed still gets a new code by itself. Removing the connection on the add-on's page does not reach the stack, so when you retire a forwarder, select Remove on the stack as well.

A token that cannot be revoked at that moment stays valid. Manage TrackMe Brokers warns you, and TrackMe keeps trying. If that forwarder may be compromised, revoke its tokens in Splunk without waiting: their audience starts with `TrackMe Broker`.

The security page gives each step and what it proves.

Two independent tokens

A remote account through a broker involves two Splunk tokens, on two deployments. Each renews on its own schedule, and rotating one never touches the other.

	The forwarder's token	The remote account's token
What it is for	The forwarder reaching the stack's channel	TrackMe reaching the destination
Issued by	The stack, for the account holding `trackme_broker`	The destination, for the account there
Kept by	The forwarder, in its connection	TrackMe, in the stack's credential store
On the way	In the `Authorization` header of each channel call, so a TLS-inspecting proxy can read it	Sealed inside the calls made over the first token, so that proxy cannot. The forwarder holds it in memory while it carries the call
Renewed	By TrackMe on the stack, at the forwarder's signed request. The role needs nothing more	By TrackMe, through the broker, as on a direct account. The account on the destination needs `edit_tokens_own` or `edit_tokens_all`
Set on	The connection, on the add-on's page	The remote account, see *automatic token rotation*
The first one	Never revoked by TrackMe, yours to revoke	Never revoked by TrackMe, yours to revoke

What stays where

- The destination's credentials stay in TrackMe, and pass through the broker for each call.
- A queued request and its response are encrypted at rest with the broker's own key. What stays readable in the KV store is what an operator reads and what runs the queue: broker, method, destination host, status, caller, claim token, failure message.
- A stolen token is not enough. Every call is signed with the forwarder's key, accepted once by an administrator. Requests and responses are encrypted between the stack and the forwarder, including through a TLS-inspecting proxy.
- The forwarder's token renews itself, see *The forwarder's token*. The token you entered stays valid until you revoke it.
- Who may use an account does not change: the same roles apply as on a direct connection.
- The forwarder logs every call in `$SPLUNK_HOME/var/log/splunk/trackme_broker.log`, with its id, method, destination host and port, status and time, never a header, a query string or a body. The forwarder sends that file to your `_internal` index like its own logs, as `sourcetype=trackme:broker`. The times in the file are UTC, whatever the host's time zone, and searches in Splunk still show each user's own zone.

What to expect when something fails

What you see	What it means
`broker_offline`	No forwarder has been seen in the last 90 seconds. Calls fail at once rather than waiting. Configuration Guardian raises `broker_offline` after 15 minutes by default.
`target_not_allowed`	The forwarder has no target covering this destination. A target just added applies within a minute.
`target_unreachable`	The connection to the destination never opened, so nothing was sent.
`tls_verification_failed`	The forwarder refused the destination's certificate, at the handshake, so nothing was sent. Use **Test** on the target.
`broker_busy`	The queue of 500 requests is full.
`response_too_large`	The response is above its limit, see *Limits*.
`broker_lost`	A forwarder stopped during a call that is not safe to repeat. The call may have reached the destination.
`key_replicating`	A TrackMe key has not reached the search head that answered yet. Nothing changed, and a call asks again for as long as its own time lasts. If this never clears, see *keys on a search head cluster*.
`token_mint_failed`	On the add-on's page: TrackMe could not mint a new token on the stack. The token in use keeps working, and this is tried again every hour.

A call that is safe to repeat goes back to the queue when a forwarder stops, and another forwarder runs it. Disabling a connection stops it within a minute. The complete list is in *Failure codes*.

Limits

Limit	Value
Largest response	256 MiB for a remote search, whose results come back as one HTTP response, and 64 MiB for any other call. The forwarder's own setting can lower that. A response also stops at 256 chunks of 768 KiB of gzip, 192 MiB compressed, which search results normally fit
Largest request body	8 MiB, and it is not chunked. A larger one is refused at once with `request_too_large`
Longest exchange	One hour
Queued requests per broker	500
Concurrent calls per connection, by default	4

A direct account has no size limit on the one response that holds a search's results, so a broker adds this ceiling.

Concurrent requests and **Largest response (MB)** are on the connection's edit dialog. A change applies by itself within about three minutes: the connection stops taking work, gives running calls up to 60 seconds to finish, and restarts.

These are limits, not performance figures. A brokered call makes more round trips than a direct one, and a remote search is several calls, so a call through a broker takes longer than the same call made directly.

The limits section of the in-depth page lists them all.

See also

- *TrackMe Broker — manage and monitor*: the two pages, everyday tasks and alerts.
- *TrackMe Broker — in depth*: the life of one call, timeouts, pools, leases, every failure code.
- *TrackMe Broker — security and cryptography*: pairing, key exchange, ciphers, FIPS, the threat model.

- *Remote Splunk deployments*, *Cribl connections* and *Webhook accounts*: the accounts that can use a broker.
- *Configuration Guardian*: the checks that watch a broker.
- *TrackMe Broker compatibility*: the Splunk versions and FIPS mode.

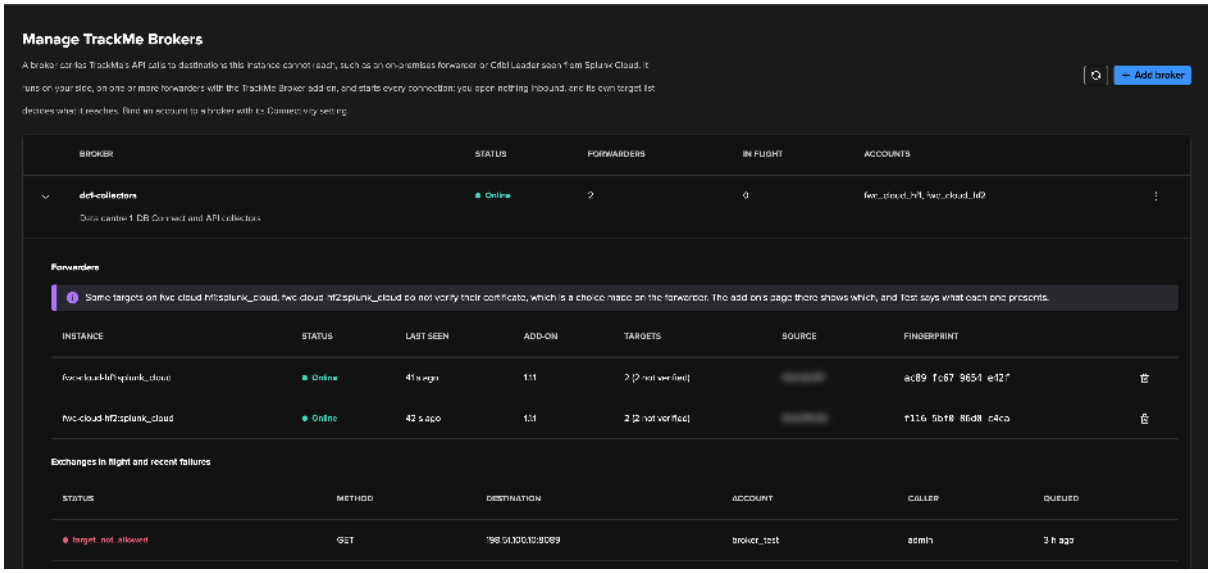
8.9.4 TrackMe Broker — manage and monitor

Once a forwarder is *set up*, you run TrackMe Broker from two pages: **Manage TrackMe Brokers** in TrackMe, on the stack, and the add-on's own page, on each forwarder. Configuration Guardian watches the brokers for you.

This page goes through both pages, the everyday tasks, the alerts, and where to read what happened.

Manage TrackMe Brokers

In TrackMe, open **Configuration → Manage TrackMe Brokers**. The page's operations need the `trackmeadmin-operations` capability.



Each broker has one row:

Column	What it shows
Broker	Its name, which is what accounts select in **Connectivity**, and its description if it has one
Status	How many of its forwarders are up, see below
Forwarders	How many forwarders are paired to it
In flight	The exchanges under way
Accounts	The accounts bound to this broker

Status reads as follows:

State	Meaning
Online	Every forwarder of this broker was seen in the last 90 seconds
Online, 1 of 2	Some were, and the others were not
Offline	None was, and at least one has pulled before
Not connected yet	Forwarders were accepted, and none has pulled yet
No forwarder	The broker has no forwarder left
Disabled	An administrator disabled it

Forwarders and exchanges

Expand a broker to see its forwarders and what it is carrying.

Forwarders column	What it shows
Instance	The forwarder's instance id, made of its host name and its connection's name
Status	**Online** if it was seen in the last 90 seconds, **Offline** otherwise
Last seen	When it last pulled, or **never**
Add-on	The version reported by its running broker process
Targets	How many targets it declares, and how many of those do not verify their certificate. The stack gets these counts and never the list
Source	The address the stack's splunkd sees. On Splunk Cloud, that is the load balancer's address and not the forwarder's, so do not read it as your forwarder's. The fingerprint is what identifies a forwarder
Fingerprint	The fingerprint of its public keys, which its add-on's page also shows

Two messages can appear above this table:

- *These forwarders do not reach the same targets.* A call may succeed on one and be refused on another. Make their target lists the same.
- *Some targets do not verify their certificate.* This is a note and not a fault, because that is a choice made on the forwarder, and the usual one for a Splunk management port on its default certificate. The add-on's page there shows which targets, and **Test** says what each one presents.

Exchanges in flight and recent failures lists calls under way, completed responses waiting for their caller, and recent failures. A completed response is removed when its caller releases it, or after an hour without release. Failures are kept for up to a day unless an administrator clears them.

Since TrackMe 2.4.25, the broker list and expanded exchanges refresh quietly about every 30 seconds. Refreshing pauses while an Add, Edit, Test or confirmation dialog is open, and while the browser tab is hidden.

After an add-on upgrade, distinguish the **installed** version in the forwarder's Apps page from the **running** version reported here. Replacing package files does not reload Python modules already held by a long-running broker input. A recent **Last seen** beside the old version means that process is still reporting the old version; waiting for more page refreshes will not upgrade it. Complete the required process restart after installation, one HF at a time, allowing calls in flight to finish and preserving the other HF's availability. Verify the new running version and native destination calls before proceeding to the next HF. Version information travels with each pull, with heartbeat writes limited to once every 15 seconds; it is not held in a long-lived version cache on the stack.

Status reads `queued` or `claimed` while a call is under way, the HTTP status for a completed response, and a plain-language explanation alongside the *failure code* when it fails. The other columns are **Method**, **Destination**, **Account**, **Caller** and **Time**. A failure shows when it failed; a call under way shows when it was queued. If an older failure has no completion time, the page says the time is unknown.

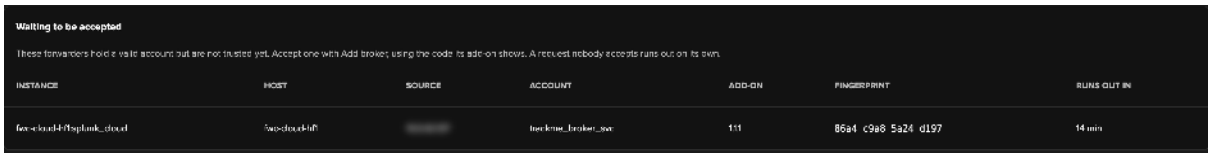
A failure older than the broker's recorded last successful exchange is dimmed and marked **Before the last success**. The recorded time is sampled at most once a minute. A success here means that an exchange returned a complete HTTP response, including an error status; it does not prove that every destination has recovered.

Clear failures appears beside the panel title when finished failures are listed. It asks for confirmation, then removes only this broker's `failed` and `expired` exchanges. Queued requests, calls in flight, unread completed responses and other brokers' exchanges are kept. The change is audited with the administrator, broker and number removed. It changes neither the broker nor its targets.

Waiting to be accepted

This panel appears when forwarders have registered and nobody has accepted them. It lists each one's instance, host, source address, Splunk account, add-on version, fingerprint and the time it has left. On Splunk Cloud, the source address is the load balancer's, so compare the fingerprint.

These forwarders hold a valid account and are not trusted yet. You cannot accept one from this list: only the code its add-on shows does that, through **Add broker**. A request nobody accepts runs out on its own.



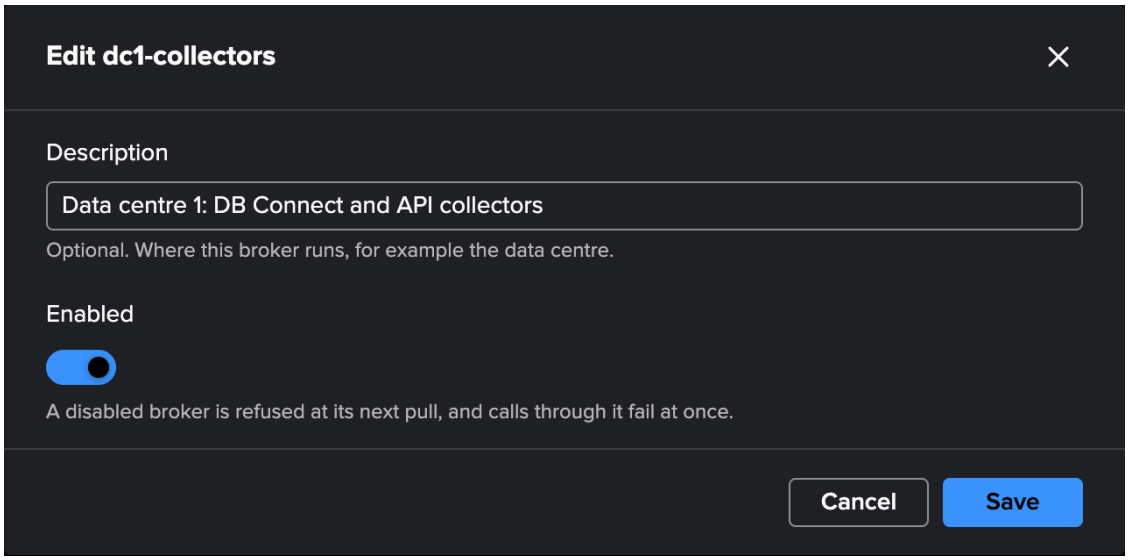
What you can do on a broker

The menu on a broker's row holds these actions:

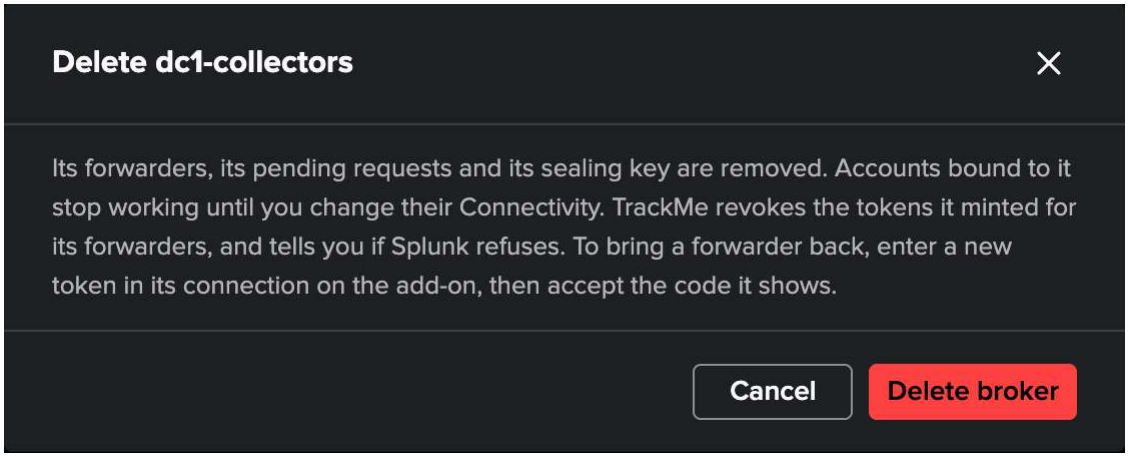
Action	What it does
Test	Makes one GET through this broker, to a URL you enter among its targets. It shows the target's status, time and size, and the round trip, or a sentence saying why it failed
Edit	Changes the broker's **Description** and **Enabled**
Replace sealing key	Writes a new key for what this broker keeps at rest on the stack. Requests in flight fail and their callers retry, and nothing changes on the add-on. See *Keys on a search head cluster* for when you need it
Delete	Removes its forwarders, its queue and its sealing key. TrackMe revokes the tokens it minted for those forwarders, and tells you if Splunk refuses. Accounts bound to it fail with `unknown_broker` until you change their **Connectivity**. To bring a forwarder back, enter a new token in its connection on the add-on, then accept the code it shows. A forwarder that never renewed its token gets a new code by itself

Enabled, when off, makes the broker refuse its forwarders at their next pull, and calls through it fail at once. Configuration Guardian raises nothing for a disabled broker, because that is your choice and not an outage.

To restrict which addresses may connect, use the stack's `search-api` allow list. On Splunk Cloud, port 8089 is closed by default, and that list says who may reach it.

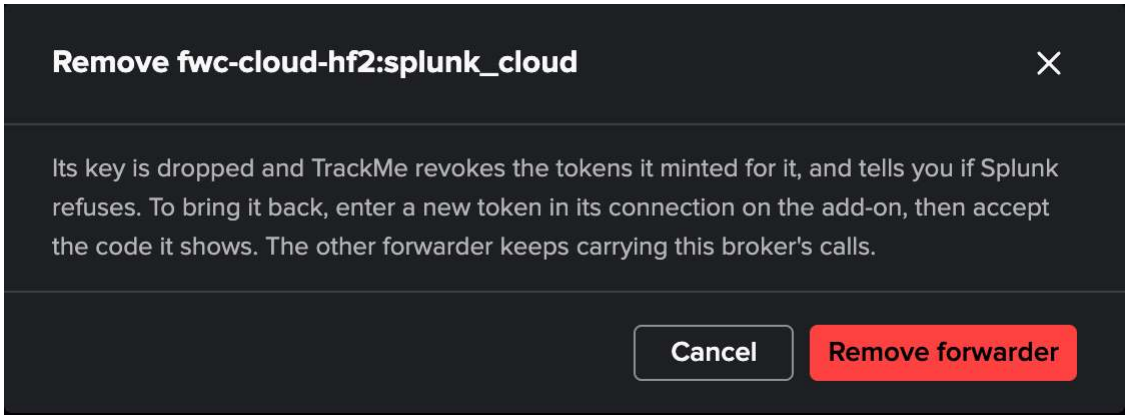


Delete asks you to confirm, and says what will happen:



Removing a forwarder

The bin icon on a forwarder's row removes that forwarder from the broker. Its key is dropped, TrackMe revokes the tokens it minted for it, and its next call is refused, including a call held open. The other forwarders of the broker carry on. If it was the only one, the dialog says so, because accounts bound to this broker then stop working until another one is added.

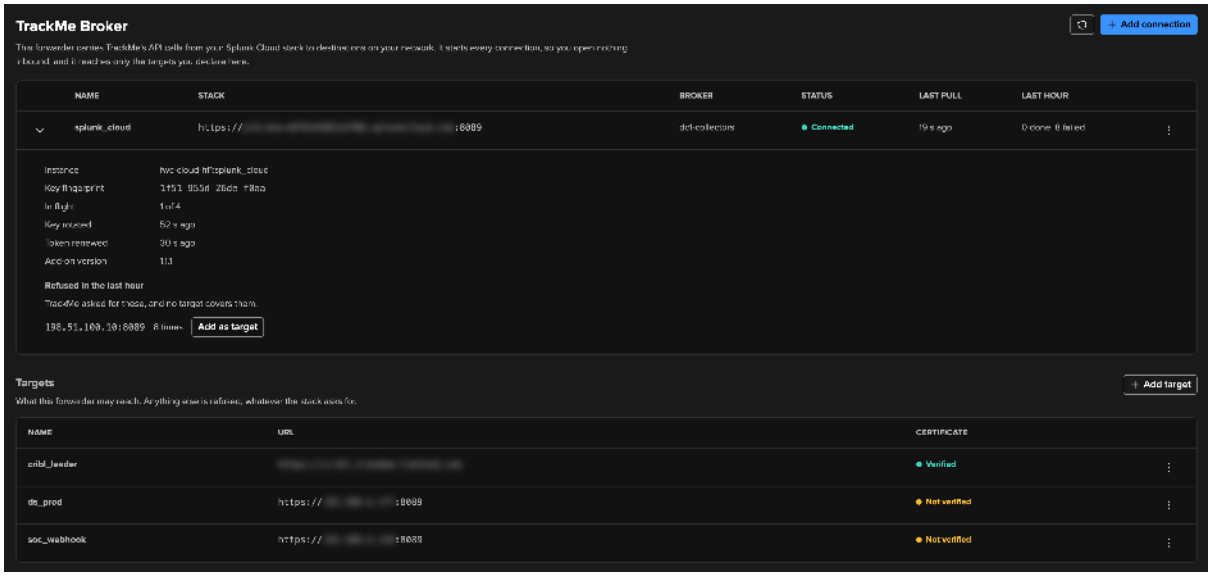


Bringing it back takes a new token in its connection, then the new code it shows: see *Bring a forwarder back*. A forwarder that never renewed its token gets a new code by itself.

A token that cannot be revoked at that moment is still valid. The page warns you, and that warning stays until the page reloads. TrackMe keeps trying, and *the security page* says how.

The add-on's page

On the forwarder, open the **TrackMe Broker** app. Its page holds this forwarder's connections and its targets.



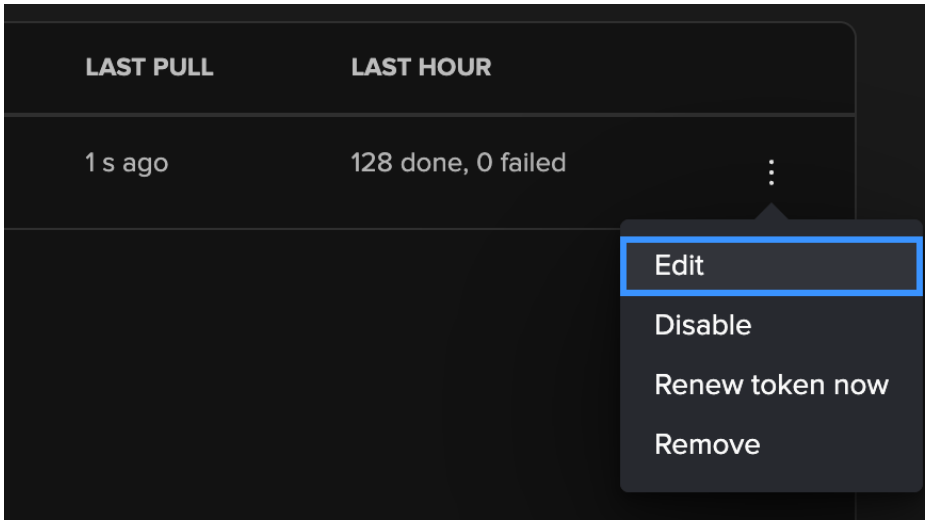
Connections

Column	What it shows
Name	The connection's name on this forwarder
Stack	The stack's URL
Broker	The broker it is paired to
Status	Starting, Waiting to be accepted, Connected, Disconnected, Disabled, Stopped, Not running or Unavailable. See *States and reasons*
Last pull	When it last asked the stack for work
Last hour	How many calls it finished in the last hour, and how many failed

Expand a connection for its details:

Detail	What it shows
Pairing code	While it waits to be accepted, with how long it stays valid. A new one replaces it when it runs out
Instance	Its instance id, as Manage TrackMe Brokers shows it
Key fingerprint	What you compare with TrackMe when you accept it
In flight	How many requests it is carrying, out of **Concurrent requests**
Key rotated	When its keys last rotated. They rotate by themselves every 30 days
Token renewed	When its token last renewed, or **Not yet**
Token renewal	Only when a renewal has not worked: why, as a sentence. See *Token renewal*
Add-on version	The version running here
Refused in the last hour	What TrackMe asked for and no target covers, by host and port, each with **Add as target** for you to confirm

The menu on a connection holds **Edit**, **Disable** or **Enable**, **Renew token now** once it is paired, and **Remove**.



Targets

Targets is what this forwarder may reach. Anything else is refused, whatever the stack asks for, and a forwarder with no target refuses every call. A new target applies within a minute, and during that minute a call can read `target_not_allowed`.

Each target shows its **Name**, its **URL** and its **Certificate**: **Verified** or **Not verified**. Its menu holds **Test**, **Edit** and **Remove**.

Test connects from this forwarder and sends nothing but a TLS handshake. It shows the host, what the name resolves to, the connection, the TLS version, and the certificate's subject, issuer, names and expiry. When a certificate does not pass, it says why and what to do:

Reason	What to do
The certificate is self-signed	Copy it, or its CA, to this forwarder, and name that file as the **CA file**
This forwarder does not trust who issued it	Copy the issuing CA to this forwarder, and name that file as the **CA file**
It is not for this host	Use a name the certificate lists, or have it reissued for this one
It has expired	It needs renewing at the destination
It is not valid yet	Check the clocks here and at the destination

With verification off on a target, the same test reads what would happen if it were on, and never acts on it.

Everyday tasks

Add a forwarder to a broker

1. On the new forwarder, install the add-on and add a connection, as in *Set it up*.
2. In TrackMe, select **Add broker**, enter its code and compare the fingerprint.
3. Under **Broker**, choose **Add to** and the existing broker, then select **Accept broker**.
4. Give it the same targets and the same **Largest response (MB)** as the others.

The status then reads **Online** when both are up, and **Online, 1 of 2** while one is down. See *Two forwarders, one broker*.

Replace a forwarder

A forwarder rebuilt with no keys generates new ones and a new code. It needs a token in its connection, like any new connection. When you enter that code, **Add broker** warns you that this replaces the forwarder paired under the same name, and gives its broker and fingerprint. If that is another machine, it stops working as soon as you accept, so compare the fingerprint with care.

Accepting moves what TrackMe minted for the earlier pairing to the new record, so it is revoked at the forwarder's next retire call. See *The forwarder's Splunk token*.

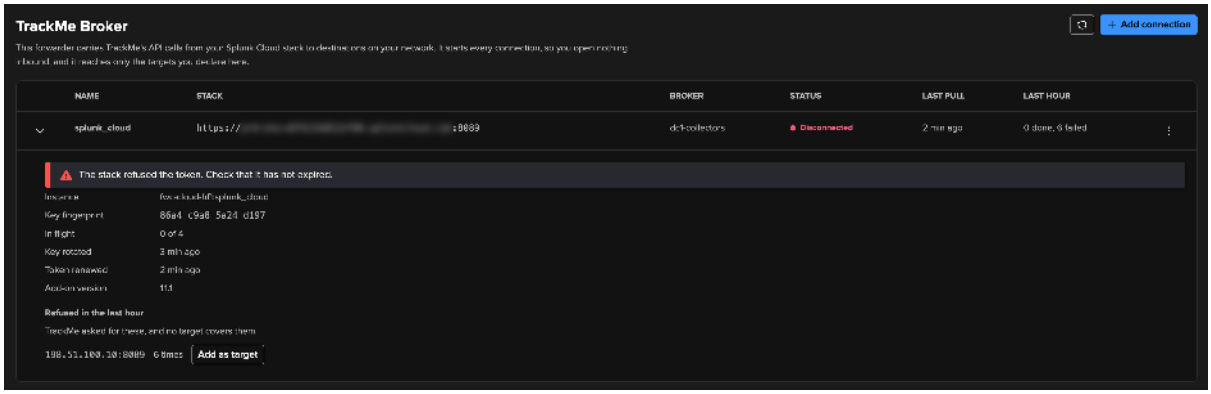
Retire a forwarder

1. In TrackMe, expand the broker and remove the forwarder. This revokes what TrackMe minted for it.
2. On the forwarder, select **Remove** on the connection. This deletes its keys and stops it serving that stack.
3. If you created a token for it and nothing else uses it, revoke that token in Splunk: TrackMe never revokes a token you entered.

Do the first step in any case. Removing the connection on the add-on's page does not reach the stack, so what TrackMe minted for that forwarder would stay valid until the stack drops it after 30 days of silence.

Bring a forwarder back

A forwarder removed in TrackMe, one whose broker was deleted, or one silent for over 30 days has had what TrackMe minted for it revoked, so the stack refuses its token: **Disconnected**, with `http_401` in its log. If a call it was holding open heard of the removal first, its page also says that TrackMe no longer knows this forwarder, `removed_from_stack` in its log, and it has made new keys. Either way, it needs a new token before it can show a new code.



1. On the stack, create a new token for its account, the one holding `trackme_broker`.
2. On the add-on's page, select **Edit** on the connection and paste the token into **Splunk token**. This applies within a minute, with no restart.
3. The connection then shows a new pairing code.
4. In TrackMe, select **Add broker**, enter the code, compare the fingerprint and accept.

Replace a token by hand

When the stack refuses the token in use, because it expired or was revoked, the add-on's page says so. Select **Edit** on the connection and paste a new token into **Splunk token**. Leave it empty to keep the stored token. A changed token applies within a minute, with no restart.

Renew token now, in the connection's menu, asks for a renewal at once and needs the current token to work.

Take a broker or a forwarder out of service

To stop	Do this
A whole broker	In TrackMe, select **Edit** on the broker and turn **Enabled** off. Calls through it fail at once with `broker_disabled`, and Configuration Guardian raises nothing
One forwarder	On its add-on's page, select **Disable** on the connection. Its process reads its input every minute, and stops. Calls it was running go back to the queue if they are safe to repeat, and fail otherwise. `broker_degraded` warns while it is away, once it has been silent for the check's time

Move an account off a broker

Set **Connectivity** on the account to **Direct** or to another broker, and save it. The broker's **Accounts** column no longer lists it, and the account's **Connection type** column follows. Do this before you delete a broker, since accounts still bound to it fail with `unknown_broker`.

Upgrade the add-on

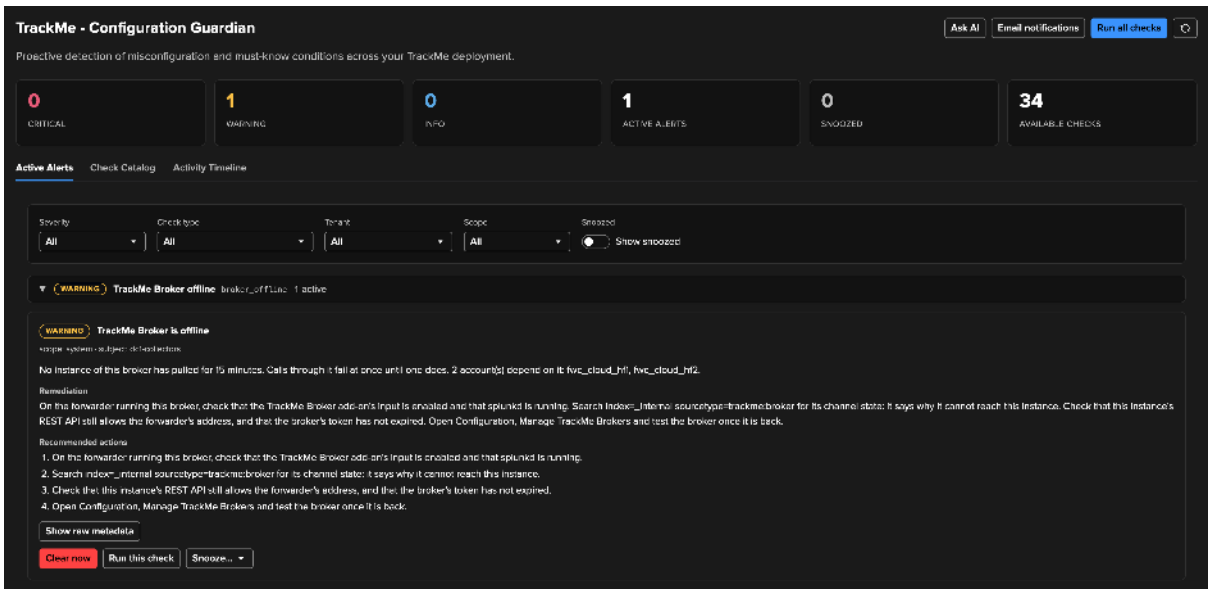
Install the new package on each forwarder. Its *release notes* say what changed and give the package and its SHA-256. The **Add-on** column on Manage TrackMe Brokers shows the version each forwarder reports.

Configuration Guardian alerts

Configuration Guardian has two checks for brokers, because a broker's failure otherwise only shows when something tries to use it. Each raises one alert per broker, on the system scope, and clears by itself.

The report `trackme_broker_watch` runs both every 15 minutes. It is scheduled only while a broker is declared, so a deployment without one runs nothing. The daily health manager and **Run now** run them too. If the broker list cannot be read, the checks are skipped and existing alerts stay.

TrackMe Broker is offline



Check	`broker_offline`
Raised when	No forwarder of the broker has pulled for `offline_after_minutes`. A forwarder just accepted counts from when it was paired, so one that never pulls raises this too, even with no account on the broker. A broker left with no forwarder raises it only if accounts depend on it
Severity	Warning, then critical once it has lasted `critical_after_hours`
What it says	How long no forwarder has pulled, that calls through it fail at once until one does, and the accounts that depend on it
Clears when	A forwarder pulls again. A forwarder just accepted is not enough to end an outage already reported. Disabling or deleting the broker clears it too, with no "resolved" notification, since that is not a recovery
Settings	`offline_after_minutes`: 15 by default, from 2 to 1440. `critical_after_hours`: 1 by default, from 1 to 168

What the alert recommends:

1. On the forwarder running this broker, check that the add-on's input is enabled and that splunkd is running.
2. Search `index=_internal sourcetype=trackme:broker` for its channel state, which says why it cannot reach the stack.
3. Check that the stack's REST API still allows the forwarder's address, and that the broker's token has not expired.
4. Open Manage TrackMe Brokers, and test the broker once it is back.

Every account bound to the broker stops working while this lasts: its trackers, its connectivity tests and its token rotation. If the channel state reads `http_401`, see *Bring a forwarder back*.

TrackMe Broker is degraded

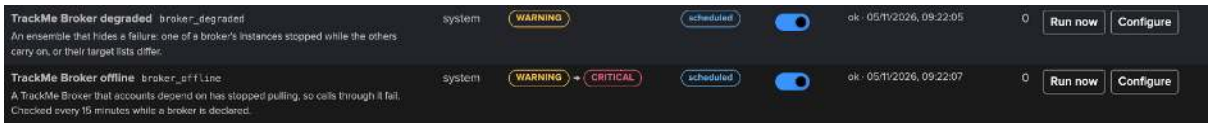
Check	`broker_degraded`
Raised when	A forwarder stopped pulling while others carry on, or the forwarders seen within that same time report different target lists
Severity	Warning
What it says	How many forwarders stopped and which, and that the rest carry the calls with no spare if they fail too, or that none is pulling now. Or that its forwarders do not declare the same targets, so a call can work or fail depending on which one pulled it
Clears when	The forwarder is back or removed, and the lists match. When none is left and `broker_offline` has raised its alert, that alert says it, and this one clears with no "resolved" notification
Settings	`silent_after_minutes`: 15 by default, from 2 to 1440

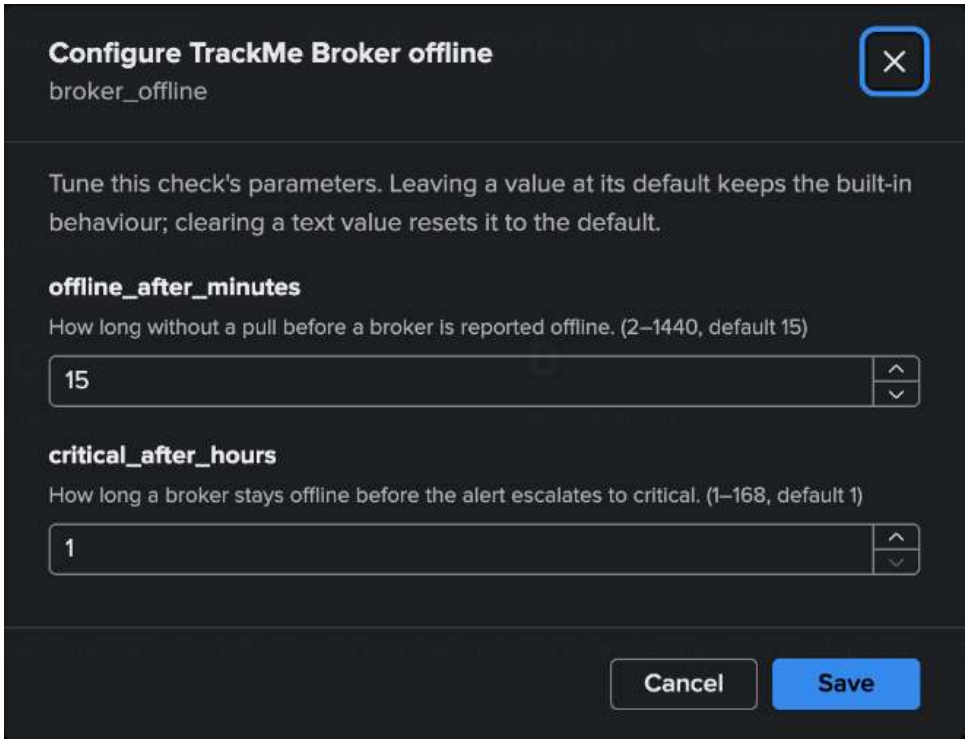
The broker still works during this alert, which is why nothing else shows it. What it recommends:

1. Manage TrackMe Brokers shows which forwarders stopped. On each of them, check that the add-on's connection is enabled and that splunkd is running.
2. If a forwarder was retired on purpose, remove it in Manage TrackMe Brokers.
3. If the lists differ, declare the same targets on every forwarder of this broker, on the add-on's page.

Tuning or turning them off

In Configuration Guardian's **Check Catalog**, **TrackMe Broker offline** and **TrackMe Broker degraded** each have their own toggle, **Run now** and **Configure**.





These settings are separate from **Online** on Manage TrackMe Brokers, which means seen in the last 90 seconds, so the page shows a stopped forwarder before the alert does.

Where to read what happened

What	Holds	Search
The add-on's log, `trackme_broker.log`, on the forwarder	Every call, with its id, method, destination host, status and time, and the channel's state lines. Never a header, a query string or a body. The times in the file are UTC, whatever the host's time zone, and searches in Splunk still show each user's own zone	`index=_internal sourcetype=trackme:broker`
TrackMe's broker log, `trackme_rest_api_broker.log`, on the stack	Each token minted, with its instance, account and id and never the token, and each Remove or Delete that could not revoke everything	`index=_internal sourcetype=trackme:rest_api source=*trackme_rest_api_broker*`
TrackMe's audit trail, on the stack	An administrator's changes: `broker_pairing_accepted`, `broker_updated`, `broker_instance_removed`, `broker_deleted`, `broker_key_rotated` and (since 2.4.25) `broker_failures_cleared`	Audit events of the `broker` object category
Exchanges in flight and recent failures, on Manage TrackMe Brokers	Each call under way, unread completed responses, and failures for up to a day unless cleared	Expand the broker

The forwarder sends its log to your `_internal` index like its own logs, so you can read it from the stack. *The failure codes* say what each code means and what to do.

See also

- *TrackMe Broker*: the concept, and how to set it up.
- *TrackMe Broker — in depth*: the life of one call, timeouts, pools, leases, every failure code and the limits.
- *TrackMe Broker — security and cryptography*: pairing, key exchange, ciphers, FIPS, the threat model and what can be revoked.
- *Configuration Guardian*: Configuration Guardian.
- *Release notes: trackme-broker*: what each release of the add-on changed.

8.9.5 TrackMe Broker — in depth

Tip

This is the **in-depth reference** for how a call travels through a broker. For the concept and the setup, start with *TrackMe Broker*. For pairing, keys, ciphers and the threat model, see *TrackMe Broker — security and cryptography*.

The transport

An account's `connectivity` field holds `direct`, the default, or `broker:<name>`. An absent or empty field is `direct`, and that path is unchanged. Anything else that is not `broker:<name>` raises `invalid_connectivity`: a mistyped field never falls back to a direct connection that nobody asked for.

For a broker, one piece is swapped under the client and the calling code stays as it is:

Client	What is swapped	A failure raises
`requests` (Cribl connections, a remote account's REST calls, token rotation, account tests)	A transport adapter, mounted on the session for `https://` and `http://`. A plain URL then fails at the envelope, and never leaves.	`Timeout` for `deadline_exceeded` and `target_timeout`, `SSLError` for `tls_verification_failed`, `ConnectionError` for the others
splunklib (`splunkremotesearch`, other SDK calls)	The `handler` of the connect call	`ConnectionError`
Outbound HTTP client (webhook accounts)	Its `transport`	The client's own outcomes: delivery unconfirmed, not sent, or a final refusal

The failure is what the client would have raised on a direct connection, with the broker's code in its text. A caller's retries, its failover between endpoints and its error reporting keep working unchanged.

Who may send a call

The exchange endpoints, `POST /services/trackme/v2/broker/exchange/<enqueue|result|result_chunk|release|cancel>`, require the capability `trackmeuseroperations`. TrackMe's code calls them on the local splunkd, and they are not exposed through Splunk Web. `enqueue` then checks the call against the account it is made for. The rights come first: the caller must be allowed to use the account, as a direct call would require, so someone without those rights learns nothing about its broker or destination from the answer. Then the account must be bound to this broker, and the destination must be one the account declares, by host and port. *The known limits* say what that leaves to the caller.

Account type	Who may use it
`remote_account`	A user holding one of the account's roles
`cribl_connection`	TrackMe administrators, as on the direct path
`webhook_account`	A power user or an administrator the account allows. The rights are checked before the destination, so nobody can test a guess at a secret URL's host
`account_test`	TrackMe administrators, for a test before an account exists. The forwarder's targets still decide what is reached

The users `splunk-system-user` and `admin` pass each of these rights checks. Only the user who queued an exchange can read it. For anyone else it reads as absent.

Where things live on the stack

The KV store is where the process that wants a call made meets the process that answers the forwarder, which can be another search head. Handlers write with the system token, after their own checks. The KV store only replaces whole documents, so what changes often sits apart from what must never be overwritten.

Collection	Key	What it holds
`kv_trackme_brokers`	Broker name	Description, enabled
`kv_trackme_broker_instance`	`<broker>:<inst`	What pairing pinned: both public keys, fingerprint, Splunk account. Written only by an acceptance or a rotation
`kv_trackme_broker_heartbea`	`<broker>:<inst`	What each pull reports: last seen, host, version, protocol, concurrency, target digest and counts, uptime, source address
`kv_trackme_broker_requests`	Request id	`sealed_envelope`, `sealed_meta`, status, deadline, attempts, claim token. In clear for an operator: broker, method, destination host and port, account, feature, caller, HTTP status, time at the target, failure code
`kv_trackme_broker_claims`	`<id>:<attempt>`	Who claimed it, and its lease. Also `<id>:lost:<attempt>`, the decision on a lost lease. Since 2.4.25, separate `success:<digest>` rows retain sampled broker completion times for one day; they contain no request or response content
`kv_trackme_broker_response`	`<id>:<attempt>`	`sealed_body`: one chunk. By attempt, so a late push of an old attempt can never mix with the next one's chunks
`kv_trackme_broker_pairings`	Registration id, `key:<name>`, `token:<broker>:<` or `token_asked:<bro`	Pending and accepted registrations, see *pairing*. The row of each key, which names its credential, see *keys on a search head cluster*. The ids of the tokens TrackMe minted for an instance, never a token, see *the forwarder's token*. A short-lived row for each `renew_token`, where the first insert wins, so that a copy of the call mints nothing

The channel

Every call comes from the forwarder, as `POST /services/trackme/v2/broker/channel/<operation>` on the stack's management port, with a bearer token for an account holding `trackmebrokeroperations`. Bodies are JSON. The add-on reads anything but a 200 as a refusal, redirects included, and never follows one.

Operation	Signed	What it does
`register`	By its own new key	Offers this instance's public keys, with a binding to the pairing code. Stores a pending row, and nothing else
`pairing_status`	By its own new key	Asks where a registration stands. Answers `pending`, `expired` or `accepted`, the last with the stack's keys
`pull`	Yes	A heartbeat, and a long poll, which Splunk Cloud's load balancer does not cut. Answers with the requests it claimed for this instance
`progress`	Yes	Extends the lease of a request still running. A renewal is also a sign of life
`push`	Yes	Delivers one chunk of a response. The last chunk completes the exchange
`fail`	Yes	Reports that a request produced no HTTP response, with a fixed code
`rotate_key`	Yes	Replaces this instance's keys, see *rotation*
`renew_token`	Yes, current key only	Asks for a new Splunk token for the account pinned at pairing, see *the forwarder's token*
`confirm_token`	Yes	Made with the new token, which proves that it works. Made again naming the token it replaces. TrackMe revokes only what it minted and still tracks for this instance: any other id is left alone and reads as retired

A signed call wraps its body as a string, so that nothing has to be canonicalised on the way:

```
{"broker": "onprem", "instance_id": "forwarder-a:main", "ts": 1793776421,
 "body": "{\"max_requests\":4,\"wait_seconds\":20,\"nonce\":\"...\"}",
↪"signature": "..."}
```

An instance id is the forwarder's host name and the connection's name.

Protocol versions

Both sides speak protocol 1. A `pull` carries the add-on's protocol under `instance`, and a stack that does not support it refuses with `protocol_unsupported` (HTTP 426). The two sides ship separately, so anything in this contract changes only with a new protocol number.

The life of one call

Queued on the stack

1. The transport builds the *envelope*.

Field	Meaning
`method`, `url`	As the caller built them. HTTPS only, with no credentials and no fragment in the URL. The method is one of GET, HEAD, POST, PUT, PATCH, DELETE or OPTIONS
`headers`	A list of `[name, value]` pairs. Hop-by-hop headers and `Host` are dropped on both sides: what the forwarder's own client decides cannot be contradicted, so a caller cannot reach an undeclared virtual host behind a shared front end
`body_b64`	The body, base64, at most 8 MiB decoded. It is not chunked: a larger body is refused at once with `request_too_large`
`timeout_seconds`, `connect_timeout`, `read_timeout`	The budget, see *Timeouts*
`retry_safe`	Whether repeating the call is harmless. It decides what a lost lease does. By default, true for GET, HEAD and OPTIONS. A caller may say otherwise
`response_limit`	Bytes the forwarder may return: 64 MiB by default, and 256 MiB for an SDK call, because a remote search's results are one HTTP response and a direct connection has no limit on it. The stack refuses more than 256 MiB. The add-on applies the lower of this and its own limit

TLS is not in the envelope. Verification and a private CA belong to the target declared on the forwarder, because only the forwarder sees that destination.

2. `enqueue` checks the envelope and the caller's rights on the account. It then refuses at once what cannot be served: `broker_disabled`, `broker_offline` when no instance has been seen for 90 seconds, `broker_busy` when 500 requests are queued.

3. The envelope is encrypted with the broker's key and written as a `queued` row, with its deadline. The caller receives a request id.

Handed over at the forwarder's pull

1. The add-on holds one signed `pull` open at a time, with a fresh nonce, while it has room for more work:

```
{"max_requests": 4, "wait_seconds": 20, "nonce": "32 hex characters, fresh for␣
↪this pull",
 "instance": {"host": "forwarder-a", "add_on_version": "1.1.1", "protocol": 1,
↪"max_in_flight": 4,
              "allowlist_digest": "...", "targets": 3, "targets_unverified": 1,
↪ "uptime_seconds": 120}}
```

The stack holds it for up to `wait_seconds`, at most 25, and answers as soon as work exists. An empty answer ends the hold, and the add-on pulls again. If a stack answers an empty pull at once, the add-on still waits a second before the next one. While it holds a call, the stack looks at the queue every 0.25 seconds, and every 15 seconds it checks that this instance is still paired, looks for ended leases and records a heartbeat.

2. When work exists, the stack first reads everything a claim depends on: that the instance is still paired, and the stack's keys. If the credential store does not answer, no request is lost.

3. **A claim is an insert** of `<request id>:<attempt>` in `kv_trackme_broker_claims`: the first inserter wins, on any search head. The row holds a lease of 60 seconds.

4. The stack opens the envelope with the broker's key, and hands it over **sealed for the key of the instance that claimed it,** with the request id and a new claim token as associated data. Each instance of a pool gets its own copy. The stack **signs** the seal, the claim, the attempt, the seconds left and the nonce of this very pull:

```
{"protocol_supported": [1], "requests": [
  {"request_id": "...", "claim_token": "...", "lease_until": 1793776481,
↪"remaining_seconds": 58, "attempt": 0,
   "sealed": "...", "signature": "..."}]}
```

One pull claims at most 20 requests, and stops adding once it has handed over 16 MiB.

Verified and carried out by the add-on

1. **The signature is checked first,** for the nonce it sent, before the request takes a slot or a lease renewal. One that does not verify gets no answer, because a `fail` signed for it would end whatever claim it names: its lease runs out, and the add-on's page says that requests arrived that TrackMe's key did not sign. A claim runs once, whatever an answer holds.
2. The seal is opened. One the stack signed and that nothing here can open answers `invalid_envelope` at once.
3. **The targets decide.** The URL must pass the checks below, and a declared target must cover it. Otherwise the call ends as `target_not_allowed` with nothing sent.
4. **The executor connects,** and checks the address it actually reached, before TLS and before anything is sent. The target's certificate settings apply. Redirects are not followed: a 3xx goes back to the caller as the answer.
5. **Every 20 seconds while it runs,** the add-on sends `progress`, which extends the lease by 60 seconds. A saturated instance stops pulling, and these renewals also say that it is alive. Since TrackMe 2.4.25, a late `progress` on a `done` or `failed` request returns HTTP 200 with `{"finished": true}` only when the claim token and instance still match. It changes neither the lease nor the request. The add-on already ignores that response body, so no add-on update is needed. Other mismatches, cancelled or expired requests, requests already released, and a `push` on an ended claim are refused with `claim_mismatch`. The call already sent to the target is not recalled.

Which destinations are allowed

The add-on applies these rules, and the stack has no say in them.

- A target is `https://host[:port][/path]`. The port defaults to 443 and must match. A wildcard `*.example.com` covers subdomains and not `example.com` itself. A path covers itself and what sits under it, on a segment boundary.
- A URL with credentials, a backslash or a control character is refused. Two parsers must agree on the host and port: the one that checks and the one that connects.
- A path cannot be walked out of. It is decoded for as long as decoding changes it, up to five rounds, and a `.` or `..` segment is refused, as is a path that still decodes after that.
- Private addresses are allowed, since a target declares them, and so is loopback: `https://localhost:8089` is the forwarder's own management port, which only suits a broker with one forwarder.
- Some addresses are always refused, whatever a target says, for what the name answers and again for the address a connection reaches: link-local (`169.254.0.0/16`, `fe80::/10`), the instance metadata addresses (`fd00:ec2::254`, `100.100.100.200`, `168.63.129.16`), multicast, unspecified and reserved addresses. An IPv4-mapped or NAT64 address counts as the IPv4 address it stands for. The names `metadata.google.internal`, `metadata.goog`, `instance-data` and `instance-data.ec2.internal` are refused by name.
- Ambient proxies and `.netrc` credentials are never used for a target.

Answered in sealed chunks

1. The add-on reads the response and gzips it as it reads. A body above the limit stops there with `response_too_large`, and so does one that would need more than 256 chunks. Hop-by-hop headers, `Content-Encoding` and `Content-Length` are dropped, since they no longer hold.

2. The gzipped body is cut into pieces of 768 KiB, each 1 MiB once in base64. Each piece is sealed for the stack's key, with the request id, claim token and chunk number as associated data. Chunk 0 also holds the reason and the headers, in the same seal, and carries the status code beside it.
3. `push` delivers the chunks in order. The stack opens each one, encrypts it again with the broker's key, and writes it. A repeated chunk replaces itself, so a push can be retried: up to 5 attempts for each report through a short outage. The last chunk marks the request `done`, if all the chunks and the headers are present. Otherwise the stack answers `incomplete_response`.
4. If there was no HTTP response, the add-on sends `fail`, with a fixed code, and the request ends as `failed`.

Read back by the caller

1. The caller polls `result`, each poll held up to 5 seconds. A poll is also how an ended lease or a queue nothing can serve is noticed, when no pull is doing it.
2. Once the status is `done`, the caller reads each chunk through `result_chunk`. A chunk stays readable until the release, so a read can be repeated: a target that already ran the call must not read as a failure because an answer was lost on its way back.
3. The caller joins the chunks, gunzips them under the same limit, and rebuilds what its client expects.
4. `release` removes the request, its claims and its chunks.

Timeouts

A call has one budget. It is stamped once, at `enqueue`, as `deadline = now + timeout_seconds`, on the stack's clock. **Time in the queue counts.**

How a budget gets into the envelope

What the client asked for	`timeout_seconds`	`connect_timeout`	`read_timeout`
`requests` with `timeout=(connect, read)`	connect + read	connect	read
`requests` with `timeout=t`	2 × t	t	t
`requests` with no timeout	60 s	30 s	60 s
A splunklib call, through `client.connect`	3600 s	30 s	3600 s
An SDK transport built with its own limit `b`	b	the lower of b and 30 s	b
The webhook client	connect + read, or what its own deadline leaves if that is less	connect	read
Test in Manage TrackMe Brokers	30 s	30 s	30 s

Whatever the source, the transport keeps `timeout_seconds` between 1 and 3600 seconds, and the stack refuses a value above 3600 seconds in any of the three fields.

The search timeout of a remote account

A remote account's `timeout_search_check`, 600 seconds by default, is passed to splunklib's connect call as `timeout`. **splunklib ignores that argument, on a direct account too**: on a direct connection, one SDK call has no time limit.

A brokered SDK call does have one, because an exchange must. It takes the channel's maximum, 3600 seconds, so that a long export does not fail only because it goes through a broker. `timeout_search_check` does not shorten

it. A TrackMe component that needs a tighter bound builds its own SDK transport with that bound, and the same bound then travels through the broker.

`timeout_connect_check`, 15 seconds by default, picks a reachable URL among an account's endpoints, through `requests` with a single value. Through a broker it therefore gives `connect_timeout` and `read_timeout` of 15 seconds, and `timeout_seconds` of 30.

How the forwarder uses it

The forwarder budgets from how long is left, not from a time of day. A claim hands over `remaining_seconds` as an integer that the stack's signature covers, and the forwarder adds it to its own clock, because the two clocks need not agree. A hand-over also carries `lease_until`, an absolute time on the stack's clock: it is not signed, and the add-on ignores it. Then:

- Both timeouts are computed once, from what remains when the call starts. The connection gets the lower of `connect_timeout` and what remains, and each read gets the lower of `read_timeout` and what remains. The read timeout is the longest silence between two reads, not a limit on the whole body.
- So a short budget caps both, and the code follows the stage rather than which limit was reached: a budget that runs out while connecting reads as `target_unreachable`, and one that runs out while waiting for the answer reads as `target_timeout`.
- While the body arrives, the deadline is also checked between blocks of the body. Past it, the call ends as `deadline_exceeded`.
- If nothing remains when the call starts, it ends as `deadline_exceeded` without connecting.

The caller waits until the deadline plus 5 seconds, because an answer in flight when its time ends costs nothing to wait for, and the alternative is to report a failure for a call that worked. Then it cancels, and raises `deadline_exceeded`. The deadline is on the row as well, so the stack expires it even if the caller has gone.

Which code means what

Code	When	Could the call have reached the target?
`target_unreachable`	The connection never opened: refused, no route, a name that does not resolve, or `connect_timeout` reached, or the budget if that is less. **This is a promise that nothing was sent**, and TrackMe may send the call again on that word.	No
`tls_verification_fail`	The certificate was refused, at the handshake.	No
`target_timeout`	The target did not answer within `read_timeout`, or within the budget if that is less.	Yes
`target_aborted`	The connection was lost after the request may have been written, or the body was cut short. A `read_timeout` reached while the body arrives reads like this, and so does a TLS error other than a refused certificate.	Yes
`deadline_exceeded`	The budget ran out anywhere else: before the call started, while the body arrives, at a claim, at the caller's poll, or at the caller's own limit.	Yes

Pools, leases and failure

A broker on the stack is an ensemble. An account binds to the broker's name and never to a host, and each forwarder paired to that broker is one instance. Every instance pulls from the same queue. There is nothing to elect and no failover step: this is active/active, with no delay on new calls, and both forwarders share the load.

A request is `queued`, then `claimed`, then one of `done`, `failed`, `expired` or `cancelled`. The one way back is a lost lease on a call that is safe to repeat.

When a forwarder stops

A lease ends 60 seconds after the claim or the last `progress`. The stack looks for ended leases at the start of a `pull`, every 15 seconds while it holds one, and when a caller polls. **The decision is taken once**, through an insert of `<id>:lost:<attempt>`, because a sweep on one search head and a poll on another can both see the same ended lease, and a second write would undo the claim that followed the first.

The call	What happens
New	The surviving instance takes it at its next pull.
Running, safe to repeat	Back in the queue when its lease ends, with the same deadline, and the survivor runs it as a new attempt with its own claim token. At most 3 attempts. The target may see it more than once, by design.
Running, not safe to repeat	Fails with `broker_lost`, because the target may have acted on it. The caller keeps its own retry rules.
Safe to repeat, with its 3 attempts used	Fails with `broker_lost` as well.
Past its deadline	Whichever comes first decides. A sweep gives `broker_lost`, even for a call that is safe to repeat. A caller's poll gives `deadline_exceeded`, even for one that is not.

A late push from an attempt that lost its lease is refused with `claim_mismatch`. Chunks are stored by attempt, so they can never mix.

When nothing can serve the queue

A pull writes a heartbeat at most every 15 seconds, and a `progress` does as well, so a saturated instance reads as alive. An instance not seen for 90 seconds is offline. With no instance online:

- `enqueue` refuses at once with `broker_offline`;
- a request already queued fails with `broker_offline` at the caller's next poll, rather than at a deadline that can be an hour away.

A disabled broker behaves the same way with `broker_disabled`, and also refuses its forwarders' calls, including one held open. An `enqueue` on a deleted broker is refused with `unknown_broker`. Its queued requests went with it, so a caller polling one gets `unknown_request`.

What Configuration Guardian raises

`broker_offline` warns when no forwarder of a broker has pulled for a while, then turns critical. `broker_degraded` warns when a forwarder stopped while others carry on, or when their target lists differ. *Configuration Guardian alerts* gives what each says, what it recommends, when it clears and its settings.

These checks use their own settings. **Online** on Manage TrackMe Brokers means seen in the last 90 seconds, so the page shows a stopped forwarder before the alert does.

Stopping and removing

- Disable a connection on the add-on's page: its process reads its input every minute, and stops. Calls it was running go back to the queue if they are safe to repeat, and fail otherwise.
- Remove a connection on the add-on's page: its keys go, its process notices within one pull and stops, and its line leaves the page. Calls it was running end the same way. Added back under the same name, it shows a new code within about a minute, or at splunkd's next interval if that is later.
- Remove a forwarder in Manage TrackMe Brokers: its key is dropped, TrackMe revokes what it minted for that forwarder at once, and its next call is refused, including a call held open, which reads the row again. With its token revoked, it reads `http_401`, so it needs a new token in its connection before it can show a new code. If a call held open hears the stack's refusal first, the add-on also verifies it, forgets its keys and reads

`removed_from_stack`. A forwarder that never renewed shows a new code directly. Other forwarders of the broker carry on.

- An instance silent for 30 days is removed from the broker in the same way, and has to pair again.
- Delete a broker: its forwarders, its queue and its key go, and TrackMe revokes what it minted for those forwarders. Accounts bound to it fail with `unknown_broker` until you change their **Connectivity**.

Failure codes

Fixed strings, never a value, a body or a secret.

Code	Set by	Meaning
`target_not_allowed`	Add-on	The URL is not among the forwarder's targets, or the address it reached is one always refused
`target_unreachable`	Add-on	The connection never opened, so nothing was sent
`target_aborted`	Add-on	The connection was lost after the request may have been written, or the body was cut short
`target_timeout`	Add-on	The target did not answer in time
`tls_verification_failed`	Add-on	The target's certificate was refused, at the handshake, so nothing was sent
`response_too_large`	Add-on	The response is above the limit, or would need more than 256 chunks
`request_too_large`	Stack	The request body is above 8 MiB. Refused at `enqueue`, so nothing was queued
`deadline_exceeded`	Either	The budget ran out
`invalid_envelope`	Either	The stack refused the envelope at `enqueue`. Or the stack signed this request and the add-on cannot open it, or its executor cannot use what it holds
`broker_error`	Add-on	An unexpected failure in the add-on
`broker_lost`	Stack	An instance lost its lease on a call that cannot be repeated
`broker_offline`	Stack	No instance seen in the last 90 seconds
`broker_busy`	Stack	The queue is full: 500 requests
`broker_disabled`, `unknown_broker`	Stack	As named
`cancelled`	Stack	The caller cancelled the exchange
`sealed_unavailable`	Stack	The broker's key cannot open the row: it was replaced, or it is gone. A credential store that does not answer is not this, and nor is a key still on its way (`key_replicating`, below): the row stays queued for the next pull
`claim_mismatch`	Stack	This claim no longer owns the request
`broker_user_refused`	Stack	The token's account is not the one pinned when this forwarder was accepted
`instance_not_paired`, `invalid_signature`, `clock_skew`	Stack	See *the signed channel*
`pairing_code_ambiguous`, `pairing_changed`, `pairing_code_not_found`	Stack	More than one registration claims this code; the row previewed is no longer the one this code finds; none does
`account_not_bound_to_broke` `destination_not_of_account`, `account_access_denied`	Stack	The caller's account does not allow this exchange

What the endpoints answer

These are refusals of one call to an endpoint, rather than how an exchange ends.

TrackMe's caller, on the exchange endpoints	Meaning
`invalid_request`, `invalid_broker_name`, `invalid_request_id`	The call itself is malformed
`invalid_url`	Not an HTTPS URL, or it holds credentials or a fragment
`invalid_connectivity`	The account's `connectivity` is neither `direct` nor `broker:<name>`
`unknown_account`, `unknown_account_type`	No such account, or no such type of account
`unknown_request`	No such exchange, one queued by another user, or one whose broker was deleted
`response_not_ready`	A chunk was asked for before the exchange was done
`invalid_chunk`, `unknown_chunk`	A chunk number out of range, or a chunk not stored
`key_replicating`	A key decided on another search head has not reached this one, see *keys on a search head cluster*. Nothing changed. A poll tries again by itself. An `enqueue` was not queued, and asks again for as long as its own time lasts
`broker_unavailable`	A passing failure. A poll tries again by itself. An `enqueue` that gets this may have been queued

A forwarder, on the channel	Meaning
`invalid_request`, `invalid_instance_id`, `invalid_request_id`	The call itself is malformed
`too_many_pending_pairings`	This account already holds 20 pending registrations
`protocol_unsupported`	The stack does not support the protocol this add-on speaks
`invalid_chunk`, `chunk_too_large`, `invalid_status_code`	A `push` out of range, with a chunk above the limits, or with a status outside 100 to 599
`incomplete_response`	The last chunk arrived with chunks or headers missing
`key_replicating`	A key decided on another search head has not reached this one. Nothing changed, and the add-on backs off. A `push` or a `fail` is retried until it lands or its claim ends
`token_mint_failed`	Splunk refused to mint a token on the system token. The token in use stands
`token_changed`	The token named is neither the pending one nor the current one. The token in use stands
`stale_call`	A `renew_token` played again: its `ts` is not newer than the last one, or a copy arrived alongside the original. Nothing is minted
`broker_channel_unavailable`	A passing failure on the stack. The add-on tries again

An administrator, on Manage TrackMe Brokers	Meaning
`invalid_code`	This cannot be a pairing code: wrong length, or a character outside the alphabet
`invalid_enabled`	A value that cannot be read
`broker_exists`	A broker of that name was created at the same moment
`unknown_instance`	That forwarder is no longer on this broker
`broker_list_truncated`	More than 1000 rows, so the page refuses to show part of a list
`key_replicating`	TrackMe's keys have not reached this search head yet, and nothing has changed. **Add broker** asks again by itself and says so
`broker_admin_unavailable`	A passing failure. Try again

What a failure proves

A caller that retries a POST on the word "not sent" must be able to trust it. So that word follows the stage of the exchange, and not the code alone. By default it is false: not knowing is not the same as not sent.

What happened	Known not to have left?
The stack answered the `enqueue` with its own refusal	Yes
The `enqueue` got no answer, or a proxy's error page	No: it may have been queued
A poll failed after the `enqueue`, whatever its code	No
The exchange ended `failed` with `target_not_allowed`, `target_unreachable`, `tls_verification_failed`, `invalid_envelope`, `broker_offline`, `broker_disabled`, `unknown_broker` or `sealed_unavailable`	Yes
The exchange ended any other way	No

The webhook client turns this into its own outcomes: *not sent* is worth another try, and anything not proven reads as *delivery unconfirmed*.

Operating a broker

Where to look

```
index=_internal sourcetype=trackme:broker
```

This is the add-on's log, from the forwarder. A state line says why it cannot reach the stack, such as `broker channel state="disconnected:http_401"`, with one line when the state changes and not one per attempt. Each call has one line, with its id, method, destination host and port, status or failure code, time, bytes and chunks. On the stack, the handlers log to `trackme_rest_api_broker.log`.

Manage TrackMe Brokers shows each broker's **Exchanges in flight and recent failures**. Failures stay readable for a day.

States on Manage TrackMe Brokers

Manage TrackMe Brokers gives each state and what a forwarder's row shows.

States and reasons on the add-on's page

Status reads Starting, Waiting to be accepted, Connected, Disconnected, Disabled, Stopped, Not running or Unavailable. A connection also says why, as a sentence. These are the reasons behind those sentences, as the add-on's status file and log name them. In the log, `unreachable` comes with the exception's type, such as `unreachable:SSLError`, and any other refusal without a code of its own as `http_<status>`.

Reason	What to do
`http_401`	The stack refused the token. Check that it has not expired or been revoked. A forwarder removed on the stack, or silent for 30 days, had what TrackMe minted for it revoked: enter a new token in its connection, and it then shows a new code
`http_403`	The account behind the token may not use the channel. It needs the `trackme_broker` role
`http_404`	The stack has no TrackMe channel. Check the URL, and TrackMe's version there
`unreachable`	Check the URL, the network, and that the `search-api` allow list includes this forwarder
`broker_disabled`	The broker is disabled in TrackMe
`broker_user_refused`	TrackMe paired this forwarder with another Splunk account. Use that account's token, or pair again
`invalid_signature`	TrackMe did not recognise this forwarder's key. Remove the instance in TrackMe, and pair again
`protocol_unsupported`	The TrackMe version on this stack does not support this add-on version
`removed_from_stack`	TrackMe no longer knows this forwarder, so it made new keys. Add it in TrackMe with the code shown. If the connection also reads `http_401`, its token was revoked too: enter a new one first
`not_paired_unverified`	Something answered that this forwarder is not paired, without the stack's signature, so the keys are kept. If the stack was rebuilt, remove this connection and add it again
`request_signature_invalid`	Requests arrived that TrackMe's key did not sign, and none was carried out. If this goes on, remove the instance in TrackMe and pair again
`rotation_not_confirmed`	Something answered a key rotation without the stack's signature, so the keys did not change
`key_replicating`	The search head this reached has not received TrackMe's keys from the others yet. This clears by itself once it does. If it never does, see *keys on a search head cluster*
`acceptance_not_bound_to_code`	An acceptance did not match this pairing code, and was refused. Check who accepted this broker
`cryptography_missing`	This Splunk's Python has no `cryptography` library, so nothing can be signed or sealed

A renewal that does not finish has its own messages, under **Token renewal** in the connection's details. It never reads as the connection failing, because calls go on with the token in use.

Reason	What it means
`token_mint_failed`	TrackMe could not mint a new token on the stack. The token in use keeps working, and this is tried again every hour
`token_not_confirmed`	Something answered a token renewal without the stack's signature, so the token was not changed
`token_not_retired`	The new token is in use, but TrackMe could not revoke the one it replaces yet. This is tried again every hour
`token_changed`	The stack no longer knows the new token, so the token in use stays. This is tried again every hour
`stale_call`	The stack took this renewal for a repeat of an earlier one, so nothing changed. This is tried again every hour
`http_401`	The stack refused a token during this renewal. The token in use stays, and this is tried again every hour
`key_replicating`	The search head this reached has not received TrackMe's keys from the others yet. This is tried again every hour

Any other reason reads as this: the token could not be renewed, the token in use keeps working, this is tried again every hour, and the add-on's log has the reason.

The page reads a status file, `$SPLUNK_HOME/var/run/trackme_broker/<input>.json`, mode 600, because a forwarder may not run a KV store. While a connection waits to be accepted that file holds the pairing code, which never reaches a log.

What the target test says

Test on a target sends nothing but a TLS handshake. It resolves the name once, screens the answer as a call would, and connects to the addresses it screened. It uses the same trust a call would use. It judges the certificate even with verification off, because whether turning it on would work is what you need to know. It needs an administrator on the forwarder. The page shows a sentence for each result. These are the results behind them, as the add-on's `/trackme_broker/test_target` endpoint answers.

Result	Meaning
`ok`	Reachable, and the certificate checks out
`reachable_unverified`	Reachable, and verification is off on this target. The page says whether this certificate would pass
`self_signed`, `certificate_untrusted`	Copy the certificate or its issuing CA to this forwarder, and name that file as the CA file
`hostname_mismatch`	Use a name the certificate lists, or have it reissued
`certificate_expired`, `certificate_not_yet_valid`	Renew it at the destination, or check the clocks
`ca_file_unreadable`	The CA file cannot be read, or holds no certificate. It must be a PEM file the Splunk user can read
`name_not_resolved`, `network_unreachable`	This forwarder cannot resolve the name, or has no route to the address
`connection_refused`, `connect_timed_out`	The host refused the connection, or nothing answered within 8 seconds. Check the port and any firewall
`not_tls`, `tls_handshake_failed`, `tls_timed_out`	The port does not speak TLS, TLS could not be negotiated, or the handshake took more than 8 seconds. A plain HTTP port reads as `not_tls`
`target_refused`	An instance metadata or special-purpose address, always refused. It is not probed
`wildcard_host`	A wildcard has no single host to test. Enter a host it covers
`invalid_url`	It needs `https://host`, with an optional port and path

Limits and constants

How large a call can be

The lowest of these applies to a response. Above any of them, the call fails with `response_too_large`.

Ceiling	Value
What the stack asks for on a remote Splunk SDK call, such as a remote search	256 MiB. A search's results come back as one HTTP response
What the stack asks for on every other call: `requests`-based calls, Cribl connections, webhook accounts	64 MiB
The forwarder's own setting, **Largest response (MB)** on the connection	256 by default, and at most 256. It can only lower the stack's limit
The chunks	256 chunks of 768 KiB of gzip: 192 MiB compressed. Search results normally compress well enough for 256 MiB to fit

A direct account has no size limit on that one response, so a broker adds this ceiling. A search that returns more than this through a broker fails, where the same search on a direct account would not.

Keep **Largest response (MB)** the same on every forwarder of a broker. The lower of the stack's limit and the forwarder's wins, so forwarders with different values can give a call different outcomes, depending on which one pulled it. TrackMe compares the forwarders' target lists, and not this setting, so nothing flags a difference.

A request body stays at 8 MiB and is not chunked: it is one KV row and one `pull`, and a larger one is refused at once with `request_too_large`. The KV store's own limit of 16 MB per record does not change. Responses fit under it because each chunk is its own row.

Every limit

What	Value
Queued requests per broker	500
Request body	8 MiB decoded, not chunked
Request headers	128 headers, 64 KiB in all
URL length	8192 characters
`timeout_seconds`, and each of the connect and read timeouts	At most 3600 s. By default 60 s, and 30 s to connect
`response_limit` in the envelope	64 MiB by default, 256 MiB for an SDK call. The stack refuses more than 256 MiB
Largest response (MB), `response_limit_mb`	256 by default, from 1 to 256, in MiB. The add-on applies the lower of this and the envelope's limit, so this setting can only lower a limit
Concurrent requests, `max_in_flight`	4 by default, from 1 to 64
Hold of a `pull`	The add-on asks for 20 s. The stack holds at most 25 s
Requests per `pull`	At most 20, and 16 MiB of sealed envelopes
Chunks	At most 256. The add-on cuts at 768 KiB of gzip, 1 MiB of base64, so a response that compresses badly stops at 192 MiB of gzip. The stack accepts up to 4 MiB of base64 and a 6 MiB seal
Lease, and `progress`	60 s, renewed every 20 s
Attempts on a request	3
Sweep for ended leases, and heartbeat writes	Every 15 s
Offline after no heartbeat	90 s
Time window of a signed call	300 s
A connection, its targets and whether its input is still enabled are read again	Every 60 s
A change to **Largest response (MB)** or **Concurrent requests**	Applies within about three minutes. Running calls get up to 60 s to finish
Retries of a `push` or a `fail`	5, each with a timeout of 120 s. A key still on its way does not count: those go on until the report lands or its claim ends
Retries of the channel	Backs off up to 60 s, and straight to 60 s for a refusal a retry cannot change
A response nobody released	Removed after 1 hour
Rows nobody came back for, failures included	Removed after 1 day, checked at most every 10 minutes
A silent instance	Removed after 30 days, and what TrackMe minted for it is revoked
The forwarder's token	Renewed right after pairing, then every 7 days by default, from 1 to 365, and with under 30% of a stated life left
A token renewal that failed	Tried again every hour
Broker name	Starts with a letter or a digit, at most 64 characters: letters, digits, dash and underscore

Concurrent requests and **Largest response (MB)** are on the connection's edit dialog. Together they are what the forwarder may hold in memory at once. The process is built with them when it starts, so a change restarts it by itself. It stops taking work, goes on renewing its leases, gives what is running up to 60 seconds to finish, and stops. splunkd then starts it again at the next interval. In total, the change applies within about three minutes. A call still running after those 60 seconds is cut, with its normal outcome: a call that is safe to repeat goes back to the queue, and any other fails with `broker_lost`. The connection's other fields and the targets apply within a minute, with no restart.

On the dialog, only a limit you changed is validated and saved. If the connection saves and its limits do not, the dialog says so.

See also

- *TrackMe Broker*: the concept and the setup.
- *TrackMe Broker — security and cryptography*: pairing, key exchange, ciphers, FIPS, the threat model.
- *TrackMe Broker — manage and monitor*: the two pages, everyday tasks and alerts.
- *Configuration Guardian*: Configuration Guardian.

8.9.6 TrackMe Broker — security and cryptography

Tip

This page is for whoever needs to assess the security of TrackMe Broker: how a forwarder and a stack come to trust each other, every key and cipher, and what holds against what. For the concept and the setup, start with *TrackMe Broker*. For how a call travels, see *TrackMe Broker — in depth*.

The short version

- The customer's side starts every connection. Nothing listens on the forwarder for the stack, and no inbound rule is needed.
- A Splunk token alone cannot pull or push: it can ask to be accepted, and only an administrator entering a short-lived code accepts it.
- From then on every call the forwarder makes is signed, and every request it receives is signed by the stack for that very pull.
- The forwarder's Splunk token renews itself, at the forwarder's signed request, so a stolen token alone cannot do it: see *the forwarder's Splunk token*.
- Requests and responses are encrypted between the stack and the forwarder, for the other side's key, on top of TLS.
- In the KV store, requests and responses are encrypted with the broker's own key.
- The forwarder reaches only the targets its administrator declared. The stack cannot widen that list, and never receives it.
- Every primitive is FIPS-approved, on one curve. On a forwarder they run under Splunk's FIPS module. On the stack TrackMe uses the library it packages, rather than Splunk's module: see *FIPS*.

What it does	Primitive	Sizes
Signs channel calls, hand-overs, refusals, and answers to a rotation or a token renewal	ECDSA on P-256 with SHA-256, DER-encoded, then base64	256-bit key
Agrees a key for one seal	Ephemeral-static ECDH on P-256	32-byte shared secret, 65-byte uncompressed points
Derives the key of one seal	HKDF-SHA256	No salt, 32-byte output
Encrypts between the two sides	AES-256-GCM	32-byte key, 12-byte random nonce, 16-byte tag
Binds keys to the pairing code	HMAC-SHA256	32-byte key derived from the code, 64 hex characters of output
Encrypts rows at rest on the stack	Fernet: AES-128-CBC with HMAC-SHA256	32-byte key: 16 to encrypt, 16 to authenticate
Fingerprint, digests	SHA-256	64 bits shown for a fingerprint

The library is `cryptography`. On the forwarder it comes from Splunk's own Python: the add-on does not package it. On the stack, TrackMe packages its own, with its own OpenSSL. The stack and the add-on hold the same implementation of pairing, signing and sealing. Fernet is on the stack only.

The keys

Key	Held by	Purpose
Instance signing key, ECDSA P-256	The forwarder's `storage/passwords`, realm `trackme_broker_pairing`, one per connection	Signs every channel call
Instance sealing key, ECDH P-256	The same	Opens what the stack seals for this instance
Stack signing key, ECDSA P-256	The stack's `storage/passwords`, realm `trackme_broker`, one per deployment	Signs what it hands to an instance, its refusals, and its answers to a rotation or a token renewal
Stack sealing key, ECDH P-256	The same	Opens responses sealed for the stack
Broker key, Fernet	The stack's `storage/passwords`, realm `trackme_broker`, one per broker	Seals rows at rest in the KV store

A key is never used for both signing and sealing. Private keys never leave their side. The stack creates its keys at the first acceptance, so a deployment with no broker holds none, and every search head reads the same ones.

Keys on a search head cluster

TrackMe Broker works on a search head cluster, on Splunk Cloud too: any member can serve a forwarder's call. A cluster replicates credentials when it can, and nothing says how long that takes on a given deployment. So no step here waits for a time or assumes one. On the stack, a key is in two places, and each decides one thing.

- The credential store holds the secret, under a name made for this one creation: `<name>.<key_id>`, with a random id. So search heads never overwrite each other.
- A row of `kv_trackme_broker_pairings`, `key:<name>`, says which credential is the key. One insert wins, on any member, and it is written only after the credential exists. Whoever loses removes their own credential.

No row means no key, and the next write makes one. A row whose credential is not on this search head yet means the key is on its way. The caller hears `key_replicating`, a 503, and nothing else changes, for however long that lasts. Each caller adapts, with no limit of its own:

Who is asking	What it does
A queued request	It stays queued. The stack opens a row before it claims it, so a key on its way leaves no claim behind.
A call being queued	Nothing was queued. TrackMe asks again for as long as the call's own time lasts, and then fails it as not sent.
A caller reading a result or a chunk	It treats this as passing, like any other failed poll.
A forwarder's `pull` or `push`	It gets a 503 and backs off, and its page says why. A `push` or a `fail` is retried until it lands or its claim ends.
Add broker	It asks again by itself and says it is waiting, until the keys arrive, the code runs out or the administrator closes the dialog.

A credential that is truly gone, deleted by hand or lost with a member destroyed before it replicated, reads as `key_replicating` for good, because nothing can tell it from a slow replication. For a broker, **Replace sealing key** writes a new credential and points the row at it. For the stack's keys, see *keys that live longer*.

Only a conflict proves that a row was lost, so only a conflict removes a credential. A KV write whose answer is lost may still have been written, so it removes nothing, and a **Replace sealing key** that fails removes neither credential. A crash or a lost answer can therefore leave a credential that nothing names. It seals nothing and stands in nobody's way.

Replacing a broker's sealing key can fail what is in flight for up to a minute, the time a process caches a key: a search head can go on sealing with the old key until its copy refreshes, and what it sealed then ends as `sealed_unavailable`.

Pairing and key exchange

splunkd authenticates every call to the stack's REST API, so the Splunk token stays, and pairing does not replace it. Pairing adds that **a token alone gets nothing**.

The steps

1. **Keys.** Saving a connection on the add-on's page creates the input. It generates two P-256 key pairs, one to sign and one to seal, and stores them in the forwarder's credential store.
2. **The code.** It generates a pairing code: 20 characters of Crockford base32, so 100 bits from a cryptographic random source, grouped for reading as `7KQ4-MZ2P-X9TB-R3HV-6WNC`. It is read case-insensitively, with O as 0, and I and L as 1. It lasts 15 minutes, and a new one replaces it.
3. **The binding.** The code never travels on the channel. What travels is `binding = HMAC-SHA256(key, canonical registration)`, where the registration holds the add-on's version, the host, the instance id, both public keys, a random nonce and the protocol. A row on the stack therefore holds a binding, not a code.
4. **Register.** The add-on sends the registration and the binding, with a signature by the new signing key over both, so a captured registration cannot be sent again under another code. The stack checks the signature and the sealing key's point, and stores a pending row with the token's account and the source address. It refuses a registration above 4 KB, and an account can hold at most 20 pending rows, so a token cannot fill the collection.
5. **Carried by hand.** The add-on's page shows the code and the fingerprint. An administrator enters the code in **Add broker** on Manage TrackMe Brokers.
6. **Preview.** The stack looks for the pending row whose binding matches the code, comparing in constant time. So the keys it finds are the ones generated next to that code, and something else holding the same token cannot substitute its own. A code that more than one row claims is refused, `pairing_code_ambiguous`, rather than guessed: someone else holds it. The preview shows the fingerprint, the instance, the host, the source address, the Splunk account and the add-on's version, and warns if this replaces a forwarder already paired under the

same instance id. On Splunk Cloud, the source address is the load balancer's and not your forwarder's. Nothing has changed yet.

7. **The fingerprint.** The administrator compares it with the add-on's page. It is the first 16 hex characters of the SHA-256 of both public keys, shown in four groups, on both sides. This is the check that a person makes between the two sides: it shows that what is about to be trusted is what the forwarder generated.
8. **Accept.** The acceptance names the row the preview showed, `pairing_changed` otherwise. One acceptance per row, on any search head: the first insert of a lock row wins, and a holder that died leaves it for two minutes. The stack pins the instance's public keys and the **Splunk account** that registered them, and removes any earlier keys of that instance.
9. **The answer.** The stack builds `answer = {broker, instance_id, stack: {signing, sealing}}` and `HMAC-SHA256(key, canonical answer)`, under the same code. It keeps the answer for seven days, so an acceptance is not lost to a code running out or a forwarder being away. The add-on always reads where its registration stands before registering again.
10. **Pinning.** The add-on polls `pairing_status` every 5 seconds, signed by the registration's key. On `accepted`, it checks the binding under the code it generated, **before** pinning anything. So the stack's keys are bound to the same code, whatever sits between the two, including a TLS-inspecting proxy. No match reads `acceptance_not_bound_to_code`, and nothing is pinned. It then forgets the code and starts pulling.

A second forwarder repeats the same steps, and joins the existing broker. **Waiting to be accepted** on Manage TrackMe Brokers lists pending registrations with no way to accept one from the list: only the code does.

How the code becomes a key

The HMAC key is `SHA-256("trackme-broker-pairing-v1" || "\n" || code)`, with the code normalised first. The code holds 100 bits, and a FIPS mode can refuse an HMAC key under 112 bits, so it is hashed. *Canonical* means JSON with sorted keys, no spaces, and ASCII only, the same on both sides.

Why a token alone only registers

Every operation other than `register` and `pairing_status` goes through the same check, in this order:

1. The instance is paired, its broker exists, and its broker is enabled. Otherwise `instance_not_paired` or `broker_disabled`.
2. The token's account is the one pinned at acceptance. Otherwise `broker_user_refused`.
3. `ts` is within 300 seconds of the stack's clock. Otherwise `clock_skew`.
4. The pinned signing key verifies the call. Otherwise `invalid_signature`.

Only then does the handler act, on the system token, on the rows of the broker that the call names. `register` and `pairing_status` have their own checks.

The signed channel

What each signature covers

Each message starts with its own label, so a signature for one purpose can never serve for another. The last row is the exception: its message is the JSON of the new keys, and only the new key signs it.

Signature	Key	Message
A registration	Its own new signing key	`trackme-broker-registration-v1\n` + canonical `{binding, registration}`
`pairing_status`	Its own new signing key	`trackme-broker-status-v1\n<registration_id>`
A channel call	Instance signing key	`trackme-broker-call-v1`, operation, broker, instance id, `ts`, and the SHA-256 of the exact body string, joined by newlines
A request handed over	Stack signing key	`trackme-broker-request-v2`, request id, claim token, the pull's nonce, attempt, remaining seconds, and the sealed value, joined by newlines
A refusal	Stack signing key	`trackme-broker-refusal-v1\n` + canonical `{broker, call, instance_id, operation, reason, server_time, ts}`
An answer to a rotation	Stack signing key	`trackme-broker-rotation-v1\n` + canonical `{broker, call, instance_id, keys, ts}`
An answer about a token	Stack signing key	`trackme-broker-token-v1\n` + canonical `{broker, call, detail, instance_id, token_id, ts}`. `detail` is the sealed token when handing one over, and then `confirmed`, `retired` or `kept`
The proof of new keys	The new signing key	Canonical `{instance_id, keys}`

In a refusal, and in an answer to a rotation or about a token, `call` is the SHA-256 of the signature of the call being answered, which nobody can know before the forwarder makes that call.

A request is bound to one pull

Every `pull` carries a fresh nonce: 16 random bytes, 32 hex characters. The stack signs each request it hands over for that nonce, with the seal, the claim, the attempt and the time left. The add-on refuses a request whose signature does not verify for the nonce it sent. So an answer captured on the way cannot be played back at a later pull, a day later, to run a POST again, and the budget cannot be changed on the way.

Signed refusals and clock correction

Two answers change what a forwarder holds: `instance_not_paired` makes it forget its keys, and `clock_skew` changes its clock offset. Anything on the way could otherwise do either with one forged answer. So each carries a `proof`: the stack's signature over the reason, the operation, the broker, the instance, `ts`, the time it gives, and the digest of the call's own signature.

The add-on acts only on a proof that verifies against the stack key it pinned, in answer to the call it just made. Anything else changes nothing: the keys stay, the page reads `not_paired_unverified`, and it tries again. A rebuilt stack holds other keys, so its refusals never verify, and the page says to remove the connection and add it again.

A clock that disagrees must not lock a broker out. On a proven `clock_skew`, HTTP 409, the add-on keeps the difference from the `server_time` in the answer, and sends the call once more. This is the same reason the add-on budgets a call from `remaining_seconds`. A hand-over also carries `lease_until`, an absolute time that is not signed, and the add-on ignores it.

Key rotation and revocation

- An instance's keys rotate every 30 days, without anyone acting. New keys are stored before the stack hears of them, so keys that could not be stored are never offered.
- `rotate_key` is signed by the current key, and carries the new public keys with a proof: the new signing key's signature over the instance and the keys. So the stack knows the sender holds both.
- The stack keeps honouring the previous signing key for an hour, so a call already on its way is not refused. Only the current key can start a rotation: the previous one can only repeat the rotation that retired it, or a leaked old key would take the instance over.

- A forwarder moves to new keys only on the stack's signed answer, for this very call. One forged 200 would otherwise leave it signing with keys the stack has never seen, for good. A 200 with no valid proof reads `rotation_not_confirmed`, and it keeps the keys it has. A rotation that is refused, or that cannot reach the stack, keeps its own code, and is tried again 5 minutes later.
- If the answer was lost, the add-on keeps the new keys as its next keys and repeats. A repeat changes nothing on the stack, and answers the same way. Once its old key is refused, more than an hour later, it signs the repeat with the new key, moves on the signed answer, and sends the call again.
- The add-on tries its current, next and previous sealing keys when it opens a request, because one sealed just before the rotation may still arrive.

Keys that live longer

The stack's own keys do not rotate, and nothing on Manage TrackMe Brokers replaces them. To replace them, because they leaked or because their credential is truly gone, delete the row `key:__stack__` of `kv_trackme_broker_pairings`, then pair every forwarder again: forwarders pin those keys, so on each one remove the connection and add it again, which takes a token for each. The next acceptance makes new keys. After pairing again, each forwarder keeps its earlier token until its next renewal, and then holds one.

The broker's Fernet key changes when an administrator selects **Replace sealing key** on Manage TrackMe Brokers. Queued requests and results not yet read then fail with `sealed_unavailable`, and a request already claimed carries on. Whether a failed call is tried again is up to its caller. Nothing changes on the add-on. The same action brings back a broker whose credential is truly gone.

Saving keys safely on the forwarder

- A save is all or nothing. Splunk's credential store takes a long value in 255-character pieces, one call each, so a save that stops part way would leave a document nobody can read, or one that reads and holds pieces of two. Each save writes a complete copy aside with its digest, then the document, then drops the copy.
- One process per input. splunkd can start a second process while the first still runs, and two would overwrite each other's keys. A file lock taken before the keys are read makes the second stand aside, and the system drops the lock however the process ends. Windows has no such lock, and there the next check is what ends an old process.
- Removing a connection deletes its status file, and a running process notices that within one pull and stops. It also reads its input and its keys every minute, and stops when its input has gone or what is stored is no longer its own. So a connection added again under the same name starts with new keys.
- A failure to read the keys never generates new ones: that would mean pairing again over a passing outage.

Revoking

- Remove on a forwarder's row, in Manage TrackMe Brokers, drops its keys and revokes the tokens TrackMe minted for it. Deleting a broker does both for each of its forwarders. The next call is refused, including a call held open, which reads the row again before it claims anything and every 15 seconds.
- A forwarder refused with a proven `instance_not_paired` (removed, its broker deleted, or silent for 30 days) forgets its keys, generates new ones, and shows a new code. Its page reads `removed_from_stack` until it is paired again. A forwarder whose tokens were revoked is also refused with `http_401`, so it cannot register those keys until its connection holds a new token. One whose next call arrives after the revocation reads `http_401` first, and hears this only then.
- A forwarder rebuilt with no keys generates new ones and a new code. Accepting it replaces the instance's row.
- Removing a connection on the add-on's page removes its private keys and does not reach the stack. What TrackMe minted for that forwarder stays valid until you select Remove on the stack, or until the stack drops that forwarder after 30 days of silence.
- A token you entered yourself is yours to revoke, as with any Splunk token: TrackMe never revokes it.

- Administrator actions are audited: an acceptance with its instance, fingerprint, account and source address, a removal with its instance, fingerprint, account and whether every minted token was revoked, a deletion with that same answer, and a change to a broker or a replaced sealing key. Each token TrackMe mints is logged in `trackme_rest_api_broker.log`, with its instance, account and id, and never the token.

The forwarder's Splunk token

The stack issues this token, so TrackMe on the stack mints the next one, at the signed request of the paired forwarder. The role `trackme_broker` needs no extra capability, and a stolen token alone cannot renew itself, because renewal needs the forwarder's signing key.

The order is the promise:

1. **Mint.** `renew_token` is signed by the forwarder's current key: the previous key is refused. TrackMe calls `POST /services/authorization/tokens` on the system token, for the account pinned at pairing, never one the caller names, and it sets no expiry. The new token goes back sealed for this forwarder's key, with the stack's signature over this very call. The stack keeps token ids, never a token, in a row of their own, `token:<broker>:<instance>` in `kv_trackme_broker_pairings`, and not on the instance's row, which only an acceptance or a rotation writes.
2. **Prove.** The forwarder makes `confirm_token` with the new token, so Splunk has just accepted it. It keeps the token in its key store from the moment it arrives, so a restart goes on from here.
3. **Store.** It writes the token into the connection, in the credential store on its side. Every thread reads it from there.
4. **Retire.** It makes `confirm_token` again, naming the token it replaces, and TrackMe revokes that one, along with anything else left on this forwarder's record. It revokes only tokens this stack minted and still tracks for this forwarder, each under the account it was minted for. Any other id is left alone and reads as retired, so a signed call cannot revoke the token entered at setup or another forwarder's.

A failure at any step leaves the token in use working. The next pass goes on from where this one stopped, an hour later, or at once on **Renew token now**. The connection shows the reason under **Token renewal**, and never reads as failing for it, because calls go on with the token in use. A token handed over and never confirmed is revoked by the next `renew_token`, unless the forwarder says that is the one it uses.

TrackMe mints on the stack's system token. On Splunk Cloud this works with no capability added to the role: `trackme_broker` alone is enough. If Splunk ever refuses, the page reads `token_mint_failed`, the token in use keeps working, and it asks again every hour. Nothing else depends on renewal.

A minted token has the rights of the account pinned at pairing, which is one more reason for that account to hold `trackme_broker` and nothing else.

When it renews

When	Set by
Right after pairing, so each forwarder holds a token of its own	**Renew the token automatically**, on the connection
Every 7 days by default	**Renew every (days)**, on the connection, from 1 to 365
When a token that states an expiry has under 30% of its life left	**Renew the token automatically**
On request	**Renew token now**, in the connection's menu, once paired

Renew the token automatically turns off the first three, and **Renew token now** still works. With it off from the start, the forwarder goes on calling with the token you entered. Each forwarder of a pool renews on its own, and retires only tokens minted for it.

The token you entered

TrackMe never revokes a token you entered. Nobody here knows its id, and two forwarders may share it. It stays valid until you revoke it, and after the first renewal it is no longer used. So once **Token renewed** shows a time on every forwarder that used it, you can revoke it, and your token policy may well ask you to.

Removing a forwarder

Remove on the stack, or deleting its broker, revokes at once what TrackMe minted for that forwarder: the token in use, a token replaced and not yet retired, and tokens left from an earlier pairing. A call it holds open may hear first that it is no longer paired, and then its page also reads `removed_from_stack`. Any new call is refused with `http_401`, because its token is gone, so bringing it back takes a new token in its connection, and then a new code. A forwarder that never renewed still gets a new code by itself. After a Remove, the other forwarders of that broker keep their own tokens and carry on.

A forwarder silent for 30 days is dropped the same way, and what TrackMe minted for it is revoked, because a minted token has no expiry and nothing would name it again. So a forwarder that comes back after that also needs a new token in its connection.

A token that cannot be revoked at that moment stays on record, and the stack's purge tries again until it is gone, because a minted token has no expiry of its own. Remove and Delete say whether everything was revoked, in the `tokens_revoked` field of their answer. Delete goes through every token record of that broker, so its answer also covers what an earlier Remove could not revoke.

When something is left, Manage TrackMe Brokers warns that a token is still valid and that TrackMe keeps trying, and the warning stays until the page reloads. The answer is audited, and `trackme_rest_api_broker.log` holds a warning. If that forwarder may be compromised, revoke its tokens in Splunk without waiting: their audience starts with `TrackMe Broker`.

Pairing a forwarder again moves the tokens of its earlier pairing to the new record, because one may be what it still calls with. The new record is written first and the old rows are deleted after it, so a write that fails loses sight of nothing. Each token keeps the account it was minted for, so one from a pairing under another Splunk account can still be revoked, and new tokens use the newly accepted account. These tokens go at its next retire call, and Remove and Delete revoke them too.

What the seal protects

The seal protects the hand-over. Once the forwarder calls with the new token, that token sits in the `Authorization` header like any other, and a *TLS-inspecting proxy* reads it. A token read that way still cannot pull, push or renew, and it goes at the next renewal.

This token is not a remote account's token. See *two independent tokens*.

Cryptography in full

Why P-256 and not Curve25519

A forwarder with FIPS on runs `cryptography` under Splunk's FIPS module, which refuses X25519, so sealing cannot use it. Ed25519 and ChaCha20 are refused the same way, and some Splunk Python builds refuse all three even without FIPS. Both sides must agree, so P-256 serves signing and sealing on both, with one curve throughout.

The sealing construction

Between the stack and a forwarder, each direction is sealed for the other side's pinned key. It is an integrated encryption scheme: a fresh key on the sender's side for every seal, against the recipient's long-term key.

1. The sender generates an ephemeral P-256 key pair for this seal.
2. It runs ECDH between the ephemeral private key and the recipient's public sealing key, which gives a 32-byte shared secret.

3. It derives a 32-byte key: HKDF-SHA256 with no salt and `info = "trackme-broker-seal-v1\n" || ephemeral point || recipient point`. Each point is 65 bytes, uncompressed. The info string binds the key to both public keys.
4. It generates a 12-byte random nonce, and encrypts with AES-256-GCM, using the associated data below. That gives the ciphertext and a 16-byte tag.
5. The sealed value is the base64 of `ephemeral point || nonce || ciphertext || tag`.

The recipient reads the ephemeral point, checks that it lies on the curve, and repeats the same ECDH and HKDF with its own private key. A public key that is not exactly 65 bytes, or not on the curve, is refused. Each seal has its own AES key, used once.

Seal	Plaintext	Associated data
A request, from the stack to the instance that claimed it	The envelope, as JSON	`<request_id>\n<claim_token>`: it cannot be replayed for another claim
A response chunk, from the add-on to the stack	`{body_b64}`, and `reason` and `headers` on chunk 0, as JSON	`<request_id>\n<claim_token>\n<chunk>`: it cannot be moved to another claim or position
A new Splunk token, from the stack to the instance that asked for it	The token	`trackme-broker-token-v1\n<broker>\n<instance_id>\n<ts>`: it opens only for that call of that instance

The associated data travels in clear beside the seal, and the receiver computes it from the fields of the call. So the seal proves that it belongs there.

A push wraps these layers: the body is gzipped, cut, base64-encoded, sealed as above, then placed in a JSON body, which the call's signature covers, inside TLS. So the sealed value is what the signature covers: the order is seal, then sign.

Encryption at rest on the stack

Before anything reaches the KV store, the envelope and the response, its headers and body, are sealed with the broker's own key. That is Fernet: AES-128-CBC with a random IV and HMAC-SHA256, through `cryptography`. The key lives in Splunk's credential store on the stack. Each process caches it for a minute.

Row	Sealed
A request	`sealed_envelope`, removed when the request ends, and `sealed_meta`: the response's reason and headers
A response chunk	`sealed_body`

In clear is what an operator reads and what runs the queue: broker, method, destination host and port, status, account, feature, caller, timing, failure code and message, deadline, attempts, claim token and `retry_safe`. A claim row holds nothing sealed. Field names never hold data.

The two seals are different on purpose. The channel's seal is for the other side's key, so the stack and the forwarder each read only what is for them. The at-rest seal is for the stack itself. The stack opens a request from its at-rest seal and seals it again for the instance at the claim, and opens a response chunk from the forwarder and seals it again at rest.

FIPS

TrackMe Broker is compatible with Splunk FIPS mode, on Splunk 9.4.x and later.

Every primitive here is FIPS-approved: ECDSA and ECDH on P-256, HKDF with SHA-256, AES-GCM, AES-CBC with HMAC-SHA256 for Fernet, and SHA-256. The broker's own code asks for nothing else, on either side, and the add-on's tests fail if its code asks for a primitive that FIPS does not approve. That is a property of the design

and of the code. A FIPS validation belongs to a cryptographic module, and TrackMe Broker does not hold one of its own. What runs these algorithms differs on each side:

Side	What runs the algorithms	With FIPS on
Forwarder	`cryptography` from Splunk's own Python, on Splunk's OpenSSL	It runs under Splunk's FIPS module, which is why X25519 is refused there.
Stack	The `cryptography` that TrackMe packages, with its own OpenSSL, which TrackMe's handlers import before Splunk's	**It does not run under Splunk's FIPS module.** TrackMe uses the library it packages rather than Splunk's module, with FIPS-approved algorithms only. No claim is made that this library is FIPS-validated. The choice of P-256 there is the design's.

What the code does because of FIPS:

What it does	Why
One curve, P-256, for signing and sealing	X25519 and Ed25519 are refused on a forwarder with FIPS on.
Hashes the pairing code before it keys an HMAC	The code holds 100 bits, and a FIPS mode can refuse an HMAC key under 112 bits.
Uses `Cipher(AES, GCM)`, never the `AESGCM` class	That class copies a prepared cipher context for each call, and the FIPS provider of Splunk's OpenSSL refuses the copy (`EVP_CIPHER_CTX_copy: not able to copy ctx`), for encryption and decryption alike. It is the same algorithm and the same bytes.
Guards splunklib's list of TLS key-exchange groups, in the builds of TrackMe and of the add-on	On a connection that does not verify certificates, a FIPS provider can refuse that list, and every such connection would then fail before a packet is sent. A refused list leaves OpenSSL's default groups.

Where a call can be read

Read from left to right: in the KV store a call is sealed with the broker's key, and only what an operator reads and what runs the queue is in clear. TrackMe's handlers hold it in clear in memory: the account's credentials are TrackMe's own. On the way to the forwarder it is sealed for the other side's key and signed, inside TLS. The add-on holds it in clear in memory for as long as it carries it, and logs only its id, method, destination host and port, status and time. The target receives an HTTPS request from the forwarder's address.

Destinations' credentials stay in TrackMe and pass through the broker for each call. Nothing keeps a copy on the forwarder.

What a TLS-inspecting proxy sees

An inspecting proxy on the forwarder's way out decrypts TLS, so it reads what sits inside it. Sealing is there for that case.

It can read	It cannot read
The Splunk token, in the `Authorization` header, a renewed one included as soon as the forwarder calls with it. See the stolen token below	The envelope: method, URL, headers, body
The channel's path, the broker's name and the instance id	The accounts' credentials
What `pull` reports: the forwarder's host, the add-on's version, the digest and counts of its targets, its uptime	The response's reason, headers and body
Request ids and claim tokens, attempts and seconds left, and the ids of tokens being renewed	The target list itself
The target's HTTP status and time on a `push`, and the code and message of a `fail`. A `fail` with `target_not_allowed` says the host and port that no target covers	The private keys

It cannot substitute keys at pairing, because of the bindings. It cannot replay an answer, hand a forwarder another token, make a forwarder forget its keys or change its clock, because those are signed for one call. Check the certificate with the connection's **CA file**: with verification on and the wrong CA, the connection fails and does not fall back.

TLS on each leg

Leg	TLS
Forwarder to stack	HTTPS. **Verify the stack's certificate** is on by default, with an optional **CA file** on the forwarder. Off keeps HTTPS and no longer checks who answers. Signatures and seals still hold, since both sides pin their keys at pairing, but whoever answers receives the Splunk token.
Forwarder to target	HTTPS only: plain HTTP is refused. **Verify the certificate** and **CA file** belong to the target, on the forwarder, because only the forwarder sees that destination. The account's own TLS options are not used. The forwarder reports a count of targets without verification, never the list.
TrackMe to its local splunkd	The exchange endpoints, on the stack's own splunkd, never through a proxy. TrackMe calls them without verifying the certificate, carrying the session key.

Threat model

Threat	Outcome
A stolen Splunk token	It cannot pull or push, and a token cannot act for a forwarder without that forwarder's signing key. It can register pending rows, which an administrator would have to accept, up to 20 for its account. It can tell brokers and instances apart from the refusals, read the stack's time, and make a code ambiguous: see the limits below. Its role is built to reach the channel and nothing else. It cannot renew itself, because `renew_token` and `confirm_token` are signed channel calls. A token TrackMe minted goes at the next renewal. A token you entered is yours to revoke.
A registration by someone holding a token	It needs its own code, so an administrator must be given that code. The fingerprint, source address and account are shown first. On Splunk Cloud, the source address is the load balancer's and not your forwarder's, so the fingerprint is what to compare.
A stolen pairing code alone	Useless without a token, and gone in 15 minutes. A row holds a binding, not the code.
A stolen pairing code and a token	Registering under it makes the code ambiguous, which accepts nobody and tells the administrator. If the forwarder's own row has run out, the preview shows the thief's fingerprint, which does not match the add-on's page, and an acceptance lands only on the row previewed.
TLS inspection on the way out	It sees signed, sealed values and the metadata above. It cannot substitute keys, replay an answer, or change what a forwarder holds.
A replayed answer to a `pull`	Refused: the signature covers the nonce of the pull that asked.
A replayed call to the stack	Bound by the 300-second window. See the limits below.
A forged refusal, or a forged answer to a rotation or a renewal	Ignored: only the pinned stack key, signing for this very call, changes what a forwarder holds.
A replayed `renew_token`	Refused as `stale_call`, and nothing is minted: `ts` must grow per instance, and a short-lived row where the first insert wins stops a copy arriving alongside the original. The new token travels sealed for that forwarder's key, signed for that very call.
Renewal turned on another account	It cannot be: TrackMe mints for the account pinned at pairing, never one the caller names, and the call must come from that account's token.
A signed `confirm_token` naming a token that is not its own	Left alone, and it reads as retired. TrackMe revokes only tokens this stack minted and still tracks for that forwarder, each under the account it was minted for. So a signed call cannot revoke the token entered at setup or another forwarder's.
A compromised stack, or a stack administrator acting against you	The forwarder's target list bounds them. They can ask a declared target for anything its origin or path allows, with any method, and read the answer. They cannot reach an address no target declares, change a target, turn verification on or off, read a private key of the forwarder, or read the target list, though a call to a guessed destination shows whether a target covers it. No code from the stack runs on the forwarder: it makes HTTP calls. Every call is logged on the forwarder itself, with its destination, in `trackme_broker.log`. A path on a target narrows what it allows. Splunk and Cribl clients need the whole origin.
Someone reading KV rows	They read ciphertext, and what an operator reads and what runs the queue. A key's row and a token's row hold ids, never a secret. Whoever can also read the stack's credential store holds the keys: this does not protect against that.
Someone reading a TrackMe backup	Backups hold the list of brokers and no key. They leave out instances, requests, claims, responses, pairings and heartbeats. So after a restore onto a rebuilt deployment, broker definitions come back with no keys, pairings or queues, and calls are refused with `broker_offline` until a forwarder is paired. Each forwarder keeps its keys, and its page says to remove the connection and add it again (`not_paired_unverified`). Accepting the new code into the restored broker's name works, and the first `enqueue` after that creates a new sealing key.
A forwarder's private key stolen	Full compromise of that instance, including what it carries: requests hold the destination's credentials. With a token for its account, it can also ask for new tokens, for that account only. Remove it on the stack, which revokes those too. The target list still bounds what it can reach, and the keys of the other instances are their own.
DNS changing its answer	The targets screen what a name answers, and the connection resolves it again. So the add-on checks the address the connection actually reached, before TLS and before anything is sent: a name that answers differently once connected

Known limits

What the design does not cover, stated plainly.

1. **Replays towards the stack stay inside the 300-second window.** Apart from `renew_token`, the stack does not remember the calls it has seen. A captured, signed `pull` replayed within that window claims real requests for an answer that the real instance never sees. Whoever replays it cannot open them, since they are sealed for the instance. Those requests end as a lost lease: a call safe to repeat goes back to the queue, and any other fails with `broker_lost`. An actor on that path could stop the traffic anyway. A replayed `push`, `progress` or `fail` names a claim, and is idempotent or refused. A replayed `confirm_token` answers the same way, and a replayed `renew_token` is refused and mints nothing.
2. **A token holder can block a pairing.** Repeating a registration, or registering under a stolen code, makes that code ambiguous. Nobody is accepted on the wrong keys, and the administrator sees the refusal. The add-on's next code, within 15 minutes, starts clean, though the same holder can repeat this while they hold the token. Revoke the token.
3. **A seal has no forward secrecy against its recipient's key.** The sender's side is ephemeral and the recipient's is long-lived. Whoever records traffic, and later steals a recipient's sealing key, can open what was sealed for that key. An instance's sealing key rotates every 30 days, and the add-on does not keep retired keys in storage. The stack's sealing key does not rotate. TLS underneath, if nothing inspects it, normally has its own forward secrecy.
4. **At-rest sealing protects KV rows, not the stack.** The broker's key and the stack's keys sit in the stack's credential store. An account that can read it can open what is sealed there.
5. **The stack's keys do not rotate, and replacing them means pairing again.** Nothing on Manage TrackMe Brokers does it. A leaked stack key is answered by deleting its row and pairing every forwarder again: see *keys that live longer*.
6. **A call's checks stop at the host and port.** `enqueue` compares the destination's host and port with what the account declares. The method, path, headers and body are the caller's. Anyone holding `trackmeuseroperations` and the rights on an account can call it directly, and send anything to that destination that the forwarder's targets allow.
7. **Two users pass every rights check.** `splunk-system-user` and `admin` pass the rights check of every account.
8. **A token alone is not quite nothing.** From the refusals it can tell brokers and instances apart. With the account pinned for a forwarder, it can read the stack's time from a `clock_skew`. It can fill the 20 pending rows of its account, which stops new registrations for that account until they run out.
9. **The target digest can be tested offline.** It is an unsalted SHA-256 of the sorted URLs, truncated to 64 bits, so whoever reads it can test a candidate list. It covers URLs only: forwarders with the same URLs and different certificate settings report the same digest.
10. **A caller on the stack can follow a redirect.** The forwarder does not follow one: a 3xx goes back as the answer. A `requests` caller on the stack that follows redirects then does so itself, as a new exchange under the same checks.
11. **TrackMe trusts its local splunkd.** It calls the exchange endpoints without verifying the certificate, carrying the session key.
12. **The fingerprint is for a person to compare.** It shows 64 bits of a SHA-256. A person who does not compare it adds nothing. The binding, not the fingerprint, is what stops keys being substituted on the way.
13. **The keys on the forwarder are as safe as the forwarder.** An administrator of that Splunk instance can read its credential store.
14. **A rotation can race an acceptance.** A rotation reads its row again just before writing, so a re-pairing that landed in between is not undone. The KV store has no compare-and-swap, so one landing within that very round trip still could be. A rotation comes every 30 days, and an acceptance is an administrator's act, so this is accepted.
15. **A connection removed on the add-on's page is not told to the stack.** What TrackMe minted for that forwarder stays valid until you select Remove on the stack, or until the stack drops that forwarder after 30 days

of silence and revokes it. Its audience names the forwarder, `TrackMe Broker <broker> <instance>`, so an administrator can revoke it sooner in Splunk.

16. **A token you entered, and renewal that does not run, stay yours to manage.** TrackMe never revokes a token you entered. With renewal off, or refused by Splunk, the token in use goes on until it expires or you replace it.

See also

- *TrackMe Broker*: the concept and the setup.
- *TrackMe Broker — manage and monitor*: the two pages, everyday tasks and alerts.
- *TrackMe Broker — in depth*: the life of one call, timeouts, pools, leases, every failure code.

8.9.7 Broker client API

Important

The client API is disabled by default. Installing or upgrading TrackMe leaves it off, and nothing changes until an administrator enables it under **Configuration → System settings → TrackMe Broker → Enable the broker client API**, or with `POST /services/trackme/v2/broker/client/admin/settings` and `{"enabled": true}`.

An independent Splunk app can submit an HTTPS request to TrackMe on its search head. A paired broker inside the destination network pulls the request, calls the API, and returns the response. The app needs no TrackMe Python imports and never connects directly to the private API. Broker wire protocol 1 is unchanged.

The example app's search command demonstrates this flow:

```
| apiexample account="example_api" operation="list"
| apiexample account="example_api" operation="echo" message="hello"
```

See *Use the example app*.

Ownership and authorization

The integrating app owns destination accounts, secrets, rotation and application permissions. For example, the example app stores its API bearer token in its own Splunk credential store, protected by object ACLs. TrackMe transports those credentials inside the sealed request; it does not create a TrackMe destination credential account for them. Never put secrets in SPL, logs or source control.

The API authenticates the caller's Splunk session and requires `trackmebrokerclient`. The `trackme_broker_client` role grants that capability without TrackMe tenant access, search roles or index permissions. Each client account additionally names the caller roles allowed to use it. An `application` label is audit metadata, not authorization. Administrative names do not bypass account policy.

Every result, chunk, cancellation and release checks the current account roles and original caller. Revocation takes effect on subsequent calls, subject to Splunk role replication. Cross-caller or inaccessible references return `unknown_request` without disclosing whether the exchange exists.

The client API needs no TrackMe licence.

Configure a separate client pool

Use the splunkd-only administration endpoints under `/services/trackme/v2/broker/client/admin/` with `trackmeadminoperations`:

Table 20: Administrative operations (POST, JSON)

Operation	Body
`create_pool`	`{"broker":"example-client"}`
`settings`	`{"enabled":true}`, or false to stop accepting new work. Writes `[broker] client_api_enabled` in `trackme_settings.conf`, the same switch as **Configuration → System settings → TrackMe Broker**, and skips the write when the value is already the one requested.
`save_account`	Account policy, as below

```
{
  "account": "example_api",
  "broker": "example-client",
  "origin": "https://api.example.com:8444",
  "path_prefix": "/v1",
  "methods": ["GET", "POST"],
  "roles": ["broker_api_example"],
  "enabled": true
}
```

The new pool has immutable `usage=third_party`. Native TrackMe accounts cannot submit into it. Add and pair a separate connection on each participating HF using the normal pairing code and fingerprint check. Preserve existing native connections. Declare the destination independently in each HF's target allowlist, with certificate verification and a trusted CA.

Separate connections on existing HFs support functional testing. They share CPU, memory and restart failure domains; they do not demonstrate isolation under load. Use small bounded test loads and verify native calls before, during and after.

The stack checks the HTTPS origin, path-prefix boundary and allowed methods. It rejects encoded paths, dot segments, repeated slashes, routing/method-override headers and hop-by-hop headers. Query strings pass through. Destination credential permissions remain necessary even when a path is allowed.

Turn the client API on

Turn it on last, once the broker is paired, the client pool and its accounts exist, and every forwarder of the pool declares the API as a target.

1. Open **Configuration → System settings**, section **TrackMe Broker**, card **Client API**.
2. Set **Enable the broker client API** to **Yes**.
3. Select **Save changes**. As for every setting on this page, the change stays pending until it is saved.

This writes `[broker] client_api_enabled = 1` in `trackme_settings.conf`, which a search head cluster replicates with the rest of that file. The `settings` operation of the administrative API changes the same switch. The `info` operation then answers `"accepting": true`.

Set it back to **No** to stop new calls. A caller can still collect a call that was already accepted, and pools and accounts are kept.

Use the example app

TrackMe provides an example app on request: the Broker API Example app. Use version 0.2.4 or later. Its **Accounts** and **Try the API** pages demonstrate account ownership and the round trip without requiring users to put credentials in a search. The app uses the neutral UCC default icons. Account registration remains available from its navigation; opening the app does not require completing Splunk's setup gate.

First configure and pair the separate client pool, declare the API target on each HF, and enable the client API. The example app does not enable the client API or modify native broker connections for you.

Open **Broker API Example → Accounts** as an administrator:

1. Enter an account name, such as `example_api`, and the private API's HTTPS origin, such as `https://api.example.com:8444`. Omit a path: the reference command uses fixed endpoints under `/v1`.
2. Select a third-party broker pool. Native pools do not appear in this selector.
3. Enter the API's bearer token. The app stores it in its own Splunk credential store and clears it from the form after submission. It is not returned by the account-list endpoint.
4. Expand **Splunk connection** and check this stack's HTTPS management URL, retained for standalone client configuration. The SPL command uses the local SDK connection and caller session supplied by Splunk; it does not send the search session through the public management address.
5. Select **Register account**. A new configuration refers to its own credential identity and a TrackMe policy permitting GET/POST under `/v1`.

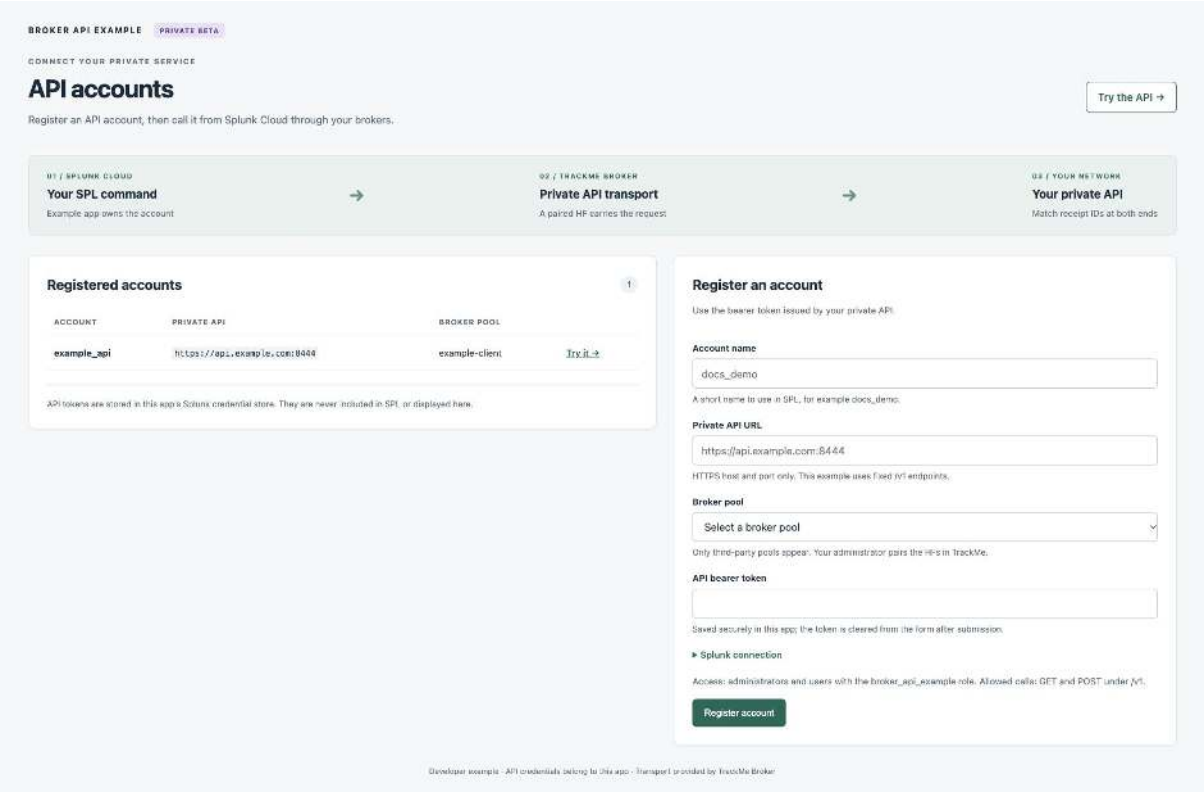


Fig. 1: Accounts page rendered with synthetic documentation values. The surrounding Splunk navigation is omitted from this preview.

Registration uses the administrator's current Splunk session. It requires both `brokerapiexampleadmin` and `trackmeadminoperations`; the example handler does not receive a system authentication token. The app grants its own setup/use capabilities and `trackmebrokerclient` to `admin` and `sc_admin`. It does not grant TrackMe administrator or tenant permissions. New account policies allow administrators and the `broker_api_example` role. Assign that role to users who should run the example command, and retain restrictive credential object ACLs.

The Accounts page registers new accounts; it does not overwrite existing ones. If registration fails, refresh the list before retrying: an ambiguous server reply may have saved the account. Partial setup can leave an unused credential or policy for administrator cleanup. Do not infer that a failed UI response rolled back every configuration write.

Existing accounts remain readable when optional administrator pool discovery is unavailable. In that case, the Accounts page explains the problem and disables new registration until discovery succeeds; the demo does not request pool discovery.

Open **Try the API**, choose an account and select **List example items**, **Echo a message**, or **Show API receipts**. The page displays the exact SPL before running it through Splunk's search-job SDK. The search executes on the search

head and calls the private API only through the broker transport. Results include the HTTP status, broker request ID and API receipt ID. Use **Open in Search** to inspect normal Splunk search messages when no rows are returned.

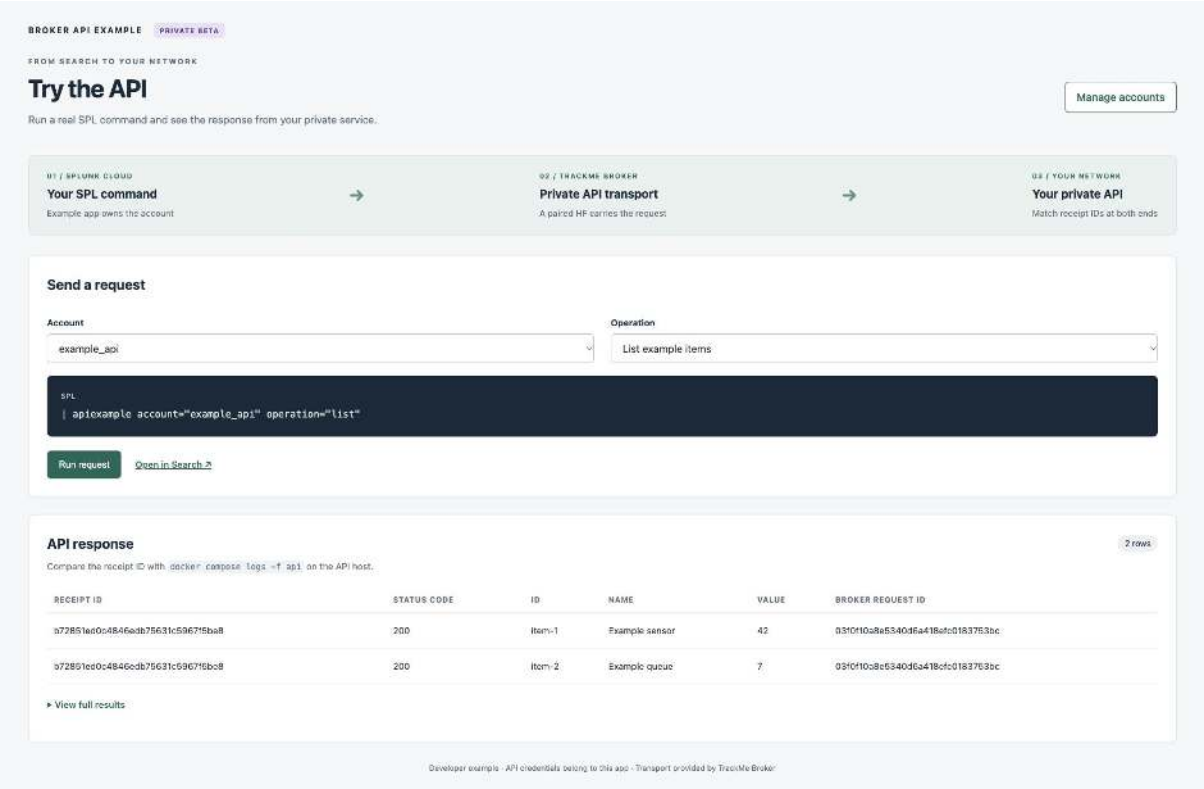


Fig. 2: Demonstration of the result layout using synthetic response data.

On the private API host, observe the other end with:

```
docker compose logs -f api
```

Each response produces a JSON stdout event containing UTC time, a fixed operation name, method, HTTP status, duration in milliseconds and `receipt_id`. Match the receipt ID to the SPL result. Health checks and authentication refusals are logged too. Raw URLs, query values, headers, tokens and bodies are excluded; unknown paths appear only as `unknown`. A recorded HTTP status is the server's chosen response, not proof that the caller received it. Docker retains three log files of up to 10 MiB each. The API's separate receipt ring retains at most 256 entries in memory and resets when the container restarts.

Exchange contract v1

Call `/services/trackme/v2/broker/client/v1/` over verified HTTPS to splunkd. Use the caller's Splunk session token or an appropriately scoped bearer token. The client endpoint is not exposed through Splunk Web's proxy.

Table 21: Client operations

Method	Operation	Behavior
GET	`info`	Returns the contract version, admission state and limits.
POST	`enqueue`	Accepts account, optional application label and envelope; HTTP 202 returns the request ID and deadline.
POST	`result`	Nonblocking state lookup using account and request ID.
POST	`result_chunk`	Reads one numbered chunk of a completed exchange.
POST	`cancel`	Marks the exchange cancelled; it cannot interrupt a destination request already executing.
POST	`release`	Deletes a terminal exchange's retained payload and frees admission capacity.

Example enqueue:

```
{
  "account": "example_api",
  "application": "example_app",
  "envelope": {
    "method": "GET",
    "url": "https://api.example.com:8444/v1/items",
    "headers": [],
    "body_b64": "",
    "timeout_seconds": 30,
    "response_limit": 8388608,
    "retry_safe": false
  }
}
```

The app adds its destination authentication headers from its protected credential store. Header values and request bodies are never required in diagnostic output. Unknown envelope fields are rejected. Subsequent operations use `{"account":"example_api","request_id":"<returned-id>"}`; a chunk read also supplies integer `chunk`, starting at zero.

States are `queued`, `claimed`, `done`, `failed`, `expired` and `cancelled`. A completed destination HTTP 401 or 500 is still `done`: inspect `status_code`. A done result also includes response headers, chunk count and elapsed time. Each chunk contains base64 bytes of the gzip stream. Concatenate decoded chunks in order, then decompress with the declared response limit; an empty stream represents an empty response. Reject excess chunks, oversized decoded data, invalid base64 and trailing compressed streams. Release only after the complete response is safely received.

Successful handler responses contain `contract_version=1` and `success=true`. Refusals contain `success=false` and a fixed `reason`. Common categories are invalid input (400), account/destination permission (403), inaccessible request (404), active/not-ready state (409), oversized request (413), capacity/rate limits (429), and disabled/offline/unavailable service (503). Splunk can reject a request before the handler; platform responses need not use this JSON format.

Retry and side effects

Never automatically enqueue again after a lost reply. The destination may have received the first request, even if the caller did not receive its ID. Default `retry_safe` is false. Only choose true when the application can tolerate a worker retry after a lost lease. It is not an exactly-once guarantee.

Poll the same request ID at the advertised interval, adding jitter. Back off on client-busy or rate-limit responses without extending the original deadline. Retrying a result or chunk read does not submit another destination request. Cancellation, an expired deadline or failed cleanup does not prove that a remote side effect did not happen. Treat `not_sent` conservatively; false or an absent field gives no such proof. The reference client never retries enqueue or falls back to direct networking.

Bounds and operational limits

Table 22: Limits

Resource	Bound
Configured client accounts	32
Retained exchanges	32 globally; four per account
Decoded request body / response body	1 MiB / 8 MiB
Incoming JSON HTTP body	1,500,000 bytes, enforced by splunkd before Python
Request headers	32 headers and 16 KiB combined
Exchange timeout	At most 120 seconds
Response chunks	At most 11; each base64 chunk at most 1 MiB
Compressed response bytes	Response limit plus 64 KiB, at most 8 MiB plus 64 KiB
Enqueue calls per second	Eight globally; two per authenticated principal
Result/chunk/cancel/release calls per second	32 globally; eight per authenticated principal

Rate state has bounded principal maps in two fixed rows. Account authorization precedes accounting; `info` does not consume the exchange budgets. Enqueue traffic cannot consume draining capacity. Multiple authorized callers can still collectively exhaust a global budget.

Client payloads live in separate KV collections. Native pulls do not scan them. A non-expiring KV mutex serializes client writers and cleanup across search heads. Ambiguous writes keep the mutex and admission capacity occupied. This deliberately halts client traffic instead of reclaiming space behind a potentially delayed writer. It does not stop the native broker queues.

Do not remove an old lock based on its age. Releasing it is an administrator's decision: `client/admin/status` reports the held lock (owner, age, poisoned), and `client/admin/reset_writer` with `{"expected_owner": "<owner>"}` removes it once no writer is running, then releases the reservations no request backs. The lock is never released automatically. Normal retention cleanup removes exchanges after their deadline plus 15 minutes; a poisoned mutex stops that cleanup too.

Limits

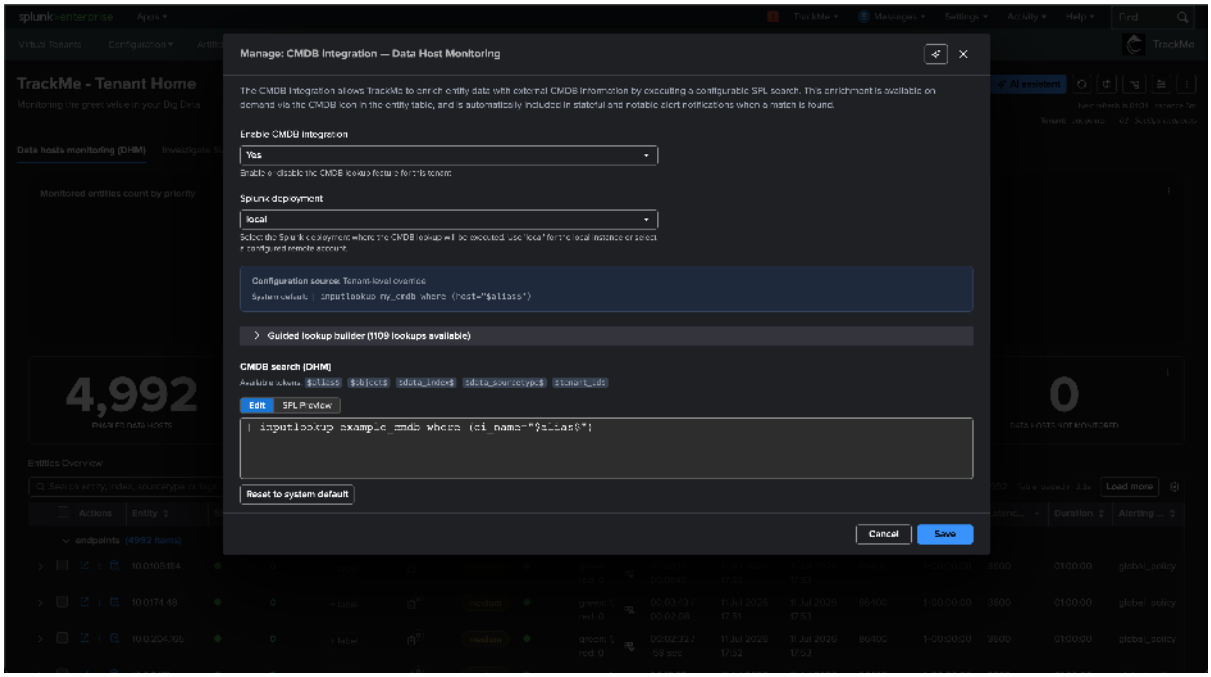
- Client accounts and pools are deleted, and a pool's sealing key replaced, through the administrative API: `delete_account`, `delete_pool` and `rotate_pool_key`. Both deletions are refused while one of the exchanges is active; cancel it first.
- A held client writer lock is released by an administrator with `reset_writer`, never automatically.
- TrackMe has no client management page yet: client accounts and pools are managed through the administrative REST API. On the **Manage TrackMe Brokers** page, **Delete** and **Rotate key** also work on a client pool.

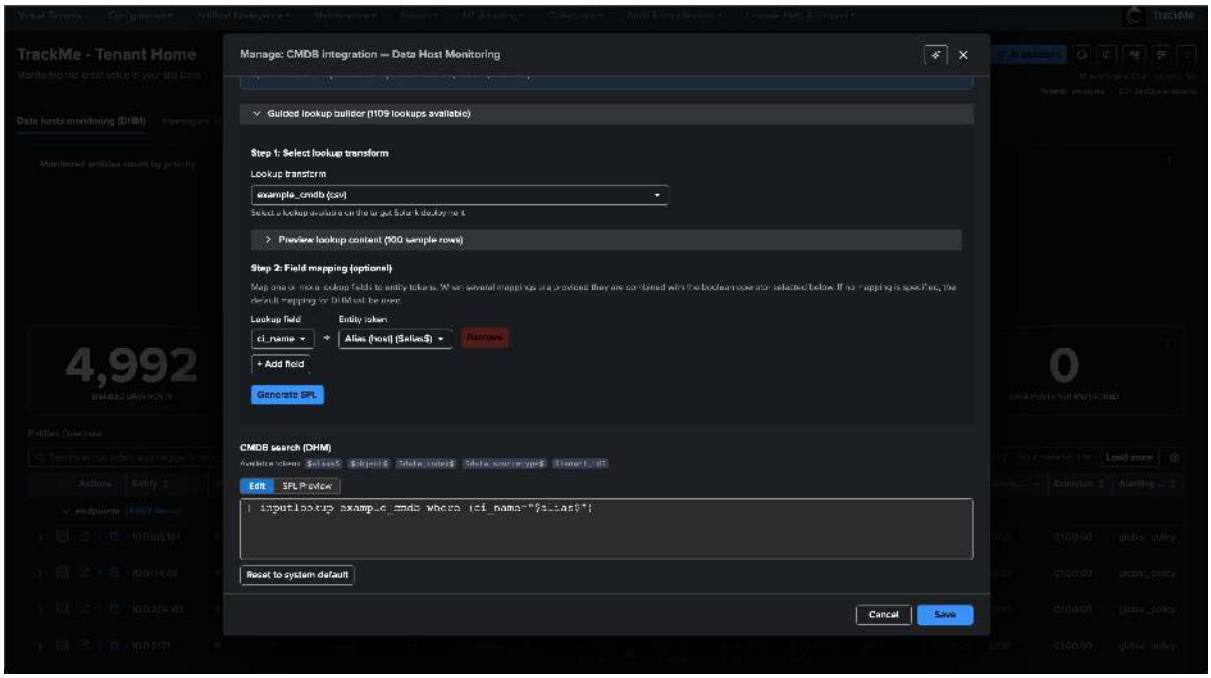
8.9.8 CMDB integration

The CMDB integration enriches any entity with external context — owner, contacts, asset metadata — pulled from a search you define. It surfaces **on demand** behind a CMDB icon on the entity, and is embedded **automatically** in stateful and notable alert notifications when a match is found. It is read-only: it never changes an entity's state or score.

How it works

The integration is configured **per component**. TrackMe ships a system-wide default template (under **Configure → splk-general → CMDB integration searches**), and each tenant can review or override it from the **Manage: CMDB integration** screen. From there you enable or disable the integration for the tenant, choose the **Splunk deployment** the lookup runs on (`local`, or a configured remote account for a remote CMDB), and edit the CMDB search template:





The search template is **any** SPL, with entity tokens that are auto-filled at run time — `$alias$`, `$object$`, `$data_index$`, `$data_sourcetype$` and `$tenant_id$`. For example, the Data Host Monitoring default looks up a CMDB lookup keyed on the host alias:

```
| inputlookup my_cmdb where (host="$alias$")
```

You do not have to write the SPL by hand: the **guided lookup builder** lets you pick a lookup transform, preview its content, and map one or more lookup fields to entity tokens — TrackMe generates the SPL for you (shown above). Because the template is arbitrary SPL, it can query a CSV or KV Store lookup, call a custom command, or reach an external ITSM such as ServiceNow or Jira.

Note

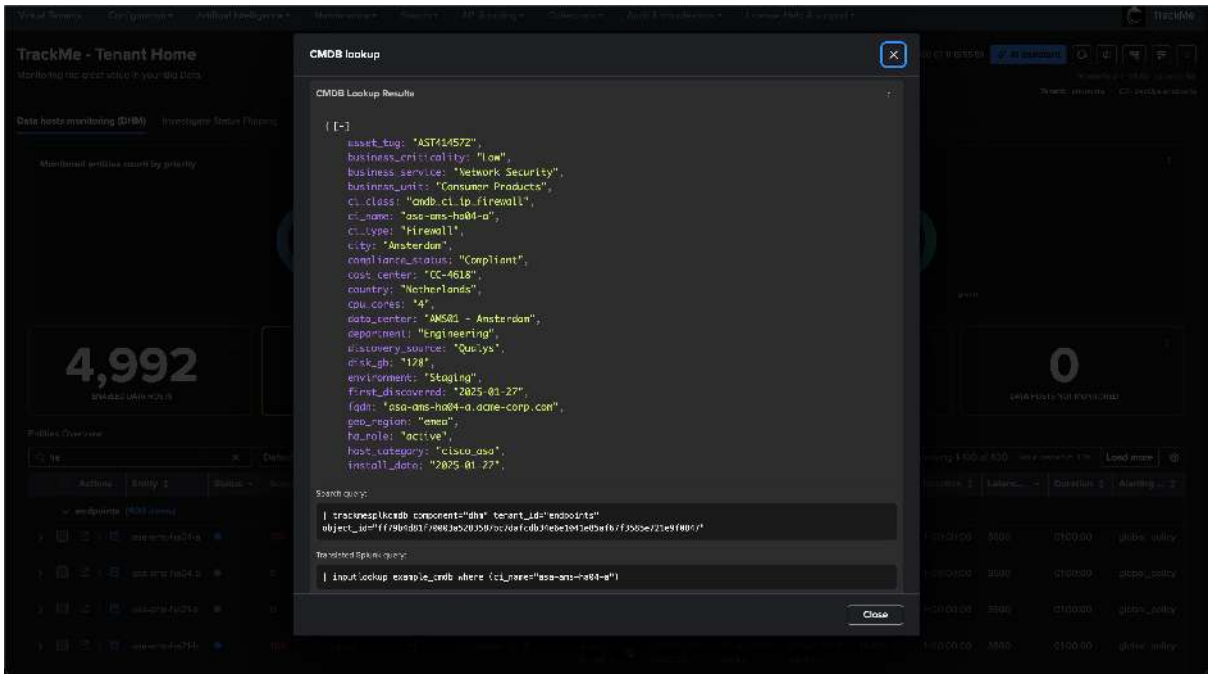
The lookup or transform the template uses must be visible in the TrackMe app context (or on the selected remote

account). Enrichment is read-only — it does not affect monitoring.

Under the hood, enrichment is performed by the `trackmesplkcmdb` command, which takes `tenant_id`, `component` (matching `^(?:dsm|dhm|mhm|cim|flx|wlk)$`) and either `object` or `object_id` — the latter being the KV Store key; pass one, not both. To troubleshoot a template, run it manually and inspect `index=_internal sourcetype=trackme:custom_commands:trackmesplkcmdb`.

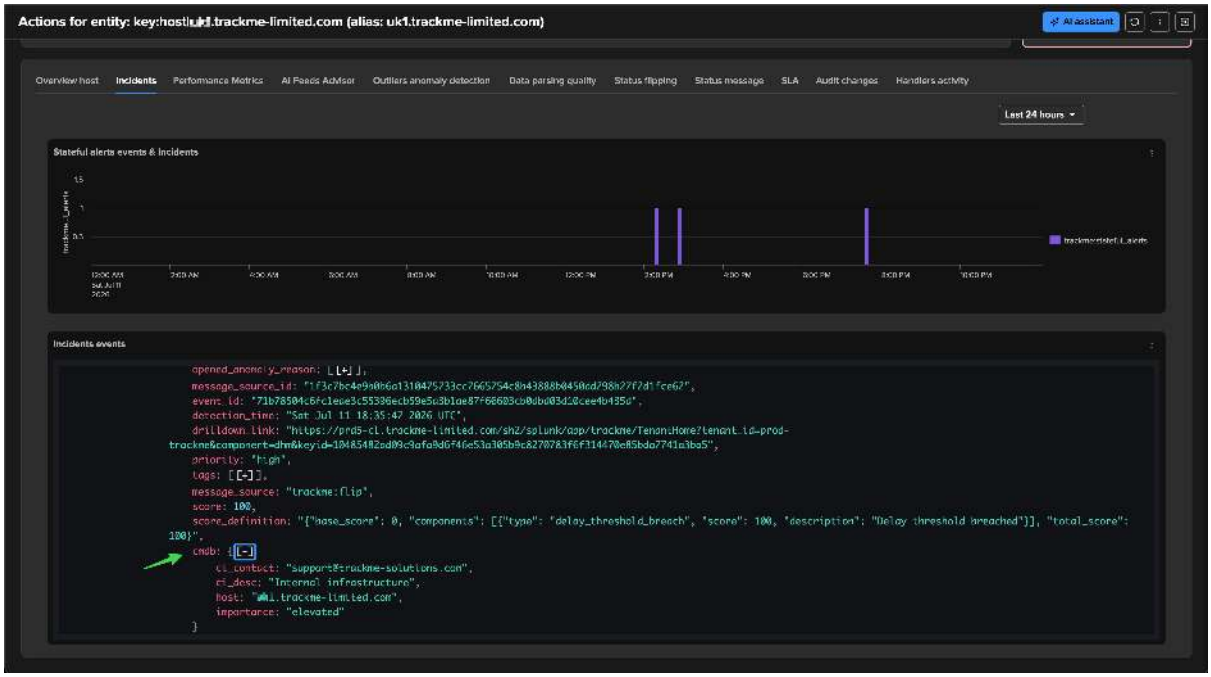
On demand, from the entity

Clicking an entity's **CMDB icon** runs the template and renders the result inline — the full CMDB record, together with the exact `trackmesplkcmdb` command and the translated Splunk query it resolved to, which makes troubleshooting straightforward. An empty result shows a clear "no results" message:



Enrichment at alert time

The same per-component templates also enrich alerts **automatically when they fire**: the *stateful* and *notable* alert actions run the entity's CMDB search at firing time and embed the result under the event's `cmdb` field, so downstream consumers (email, ES, a SOAR playbook, a ticket) receive the owner and asset context inline — no separate enrichment step:



It is controlled per tenant, from the same **Manage: CMDB integration** screen: the *Enable CMDB integration* toggle (on by default) turns it on, and the *Splunk deployment* setting selects whether the template runs locally or against a remote account. Enrichment is best-effort — if the lookup fails the alert still fires, just without the `cmdb` field.

See also

- *Monitoring Components* — the components that supply the entity tokens.
- *Lookup monitoring* — monitor the freshness of the CMDB source lookup itself.

8.9.9 Lookup monitoring

Lookup monitoring turns Splunk **lookups** — CSV files and KV Store collections — into first-class *DSM* entities, so a lookup that silently stops being updated (a stale threat-intel feed, an abandoned asset list) becomes visible as a problem. It is a dedicated DSM search mode.

How it works

- Choose **lookups (CSV file & KV Store monitoring)** in the Hybrid Trackers wizard. Instead of an SPL editor you set the **lookup type** (`csv`, `kvstore` or both), the **app namespace(s)** (a name, a glob like `TA-*`, or a comma list) and a **lookup name pattern** (regex) — or let the built-in **Populate** pickers discover the namespaces and lookups available on the deployment and fill those in for you.
- Each matching lookup becomes one entity, under the special `lookups` index.
- **Freshness** is determined differently per type:
  - **CSV** — from the file's real filesystem modification time (not the unreliable knowledge-object timestamp), with a streaming, gzip-aware row count.
  - **KV Store** — splunkd only tracks the key and user, so you provide a **preference-ordered list of candidate time fields** (`_time`, `mtime`, `updated_at`, …); the first one that parses wins. If none match, freshness cannot be determined (volume is still trackable).
- **Alerting on staleness and record count** — a lookup has no index time, so latency is neutralised. TrackMe alerts on **staleness** ("not updated in the past N seconds", default 24 hours, set in the wizard and editable later) and on the **record count**: by default an **empty lookup (0 records) flips red**, and you can require a minimum of *N* records per entity (2.4.8).

Standard DSM machinery still applies — outlier detection on the row count works out of the box (a 50,000-row lookup dropping to 200 is flagged automatically), and the full set of entity tabs is available. Lookup entities also get a **lookup-native presentation**: a *last-known records* KPI tile, with status messages, charts and alert emails framed as *records and update frequency* rather than feed latency.

Note

- Lookup monitoring is powered by the `trackmelookupsmonitor` command, shipped in a small companion add-on, **TA-trackme-lookupmonitor**. For lookups on a **remote** search head, install that add-on on the remote SH (the full app is not required there) and choose the remote account — TrackMe wraps the command in `splunkremotesearch` automatically (see *Remote Splunk deployments*).
- Discovery runs as the tenant owner — lookups that user cannot see are not found.
- This supersedes the older elastic-source `from lookup:` approach.

See also

- *Lookups monitoring — in depth* — the dedicated TA, the KV Store mtime REST probe, the emitted fields, and the limitations.
- *DSM — Data Source Monitoring* — the DSM entity model these lookups use.
- *Remote Splunk deployments* — monitoring lookups on a remote search head.

8.9.10 Lookups monitoring — in depth

Tip

This is the **in-depth reference** for monitoring Splunk lookups (CSV and KV Store) as DSM entities. For the readable overview, start with *Lookups monitoring*. This page covers the dedicated TA, the KV Store mtime REST probe, the emitted entity fields, and the limitations and edge cases.

Introduction

Lookups monitoring is a dedicated search mode of the Hybrid Trackers wizard that turns Splunk lookups — CSV files and KVstore collections — into first-class TrackMe DSM entities.

Lookups are everywhere in a mature Splunk deployment: threat-intel feeds, asset databases, enrichment tables, pre-calculated aggregations, exclusion lists. When a lookup silently stops being updated, anything that relies on it degrades quietly — detections miss new IOCs, dashboards show stale enrichment, downstream alerts misfire. Detecting that early is exactly the same problem TrackMe solves for indexed data, and the **lookups** search mode applies the same DSM machinery to lookup objects.

What the tracker does on every run:

- Enumerates the lookups visible to the dispatching user (CSV, `.csv.gz`, KVstore collections) and filters them by app namespace and a name regex.
- For each CSV file: reads the **filesystem modification time** (`os.stat`) and streams the file to count rows — not the unreliable `updated` mtime returned by `/services/data/lookup-table-files` (which reflects the knowledge-object mtime and changes whenever permissions or props are touched, not when the file actually changed).
- For each KVstore collection: reads the document count (cursor-paginated, uncapped by default) and probes an operator-supplied, preference-ordered list of candidate timestamp fields to derive the modification time.
- Emits one DSM entity per lookup, with `data_index = "lookups"`, `data_sourcetype = "<app>:<name>"` and `object = "lookups:<app>:<name>"`.

- Alerts when a lookup has not been updated within the configured **delay threshold** (`data_max_delay_allowed`) — for example `86400` seconds for "must be updated at least once every 24 hours".

Note

The previous approach using an Elastic Source with `| from lookup:<name> | eval _time=...` (see *Elastic sources*, *Elastic source example 2*) is **superseded** by this search mode. The lookups search mode is significantly faster on large lookups, does not require the lookup itself to maintain a `_time` field, and reads the **actual filesystem mtime** for CSV files rather than a per-row event time.

Why a dedicated Splunk add-on?

Lookups monitoring is powered by the `trackmelookupsmonitor` generative custom search command, shipped as a small dedicated Splunk Technology Add-on: TA-trackme-lookupmonitor.

The TA is split out from the main TrackMe app for three reasons:

1. **AppInspect / Splunk Cloud certification** — filesystem read access stays narrowly declared in a single-purpose TA. TrackMe itself does not gain any new sensitive permissions.
2. **Lightweight remote search-head footprint** — when monitoring lookups that live on a remote SH (separately-managed cluster, governance restrictions, etc.), only this small TA needs to be installed there. TrackMe itself is **not** required on the remote SH.
3. **Standalone usability** — the `| trackmelookupsmonitor` command can be invoked directly for ad-hoc lookup inventory, debugging or one-off audits.

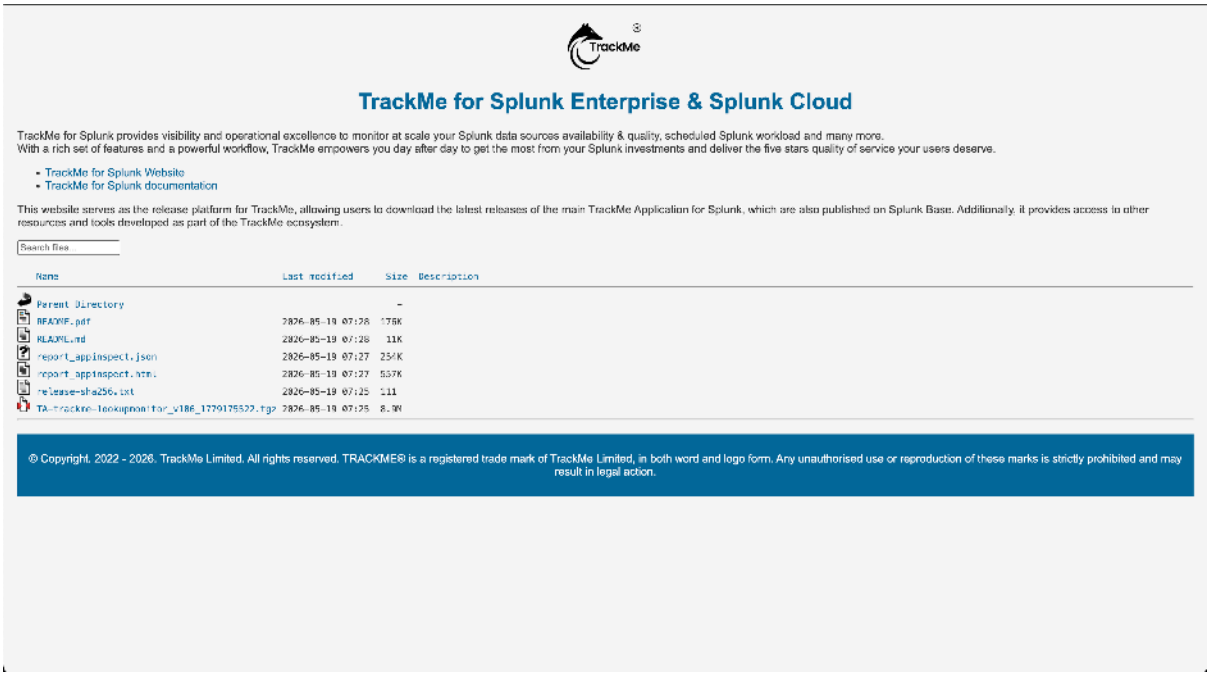
Prerequisites

Install the `TA-trackme-lookupmonitor` add-on on every search head that will run a lookups tracker:

- The **local** search head where TrackMe is installed (mandatory — the tracker dispatches from there).
- **Every remote** search head referenced by a Splunk Remote Deployment account that you want to monitor lookups on (see *Remote Splunk deployments*). The command is wrapped by `splunkremotesearch` so it executes on the remote SH; the TA is what makes that wrapped call resolvable.

Download the TA from the TrackMe distribution site:

- https://downloads.trackme-solutions.com/TA-trackme-lookupmonitor/



The TA is built with the Splunk UCC framework and bundles `splunklib` — no Splunk SDK install is required. It exposes a single configuration page with logging-level controls; logs land in `$SPLUNK_HOME/var/log/splunk/trackmelookupsmonitor.log`.

Tip

The TA can be tested in isolation. Once installed, run a standalone search like this on the target search head — no TrackMe required:

```
| trackmelookupsmonitor app_namespace="search" lookup_type="csv"
```

Creating a lookups Hybrid Tracker — CSV example

The lookups search mode is selected from the standard Hybrid Trackers wizard. The first few steps (tracker name, target Splunk deployment) are identical to the other search modes — see *Hybrid trackers* for the wizard entry points and shared fields.

On Step 2 (Search mode and constraints), choose `lookups (CSV file & KVstore monitoring)` **from the dropdown.** A banner reminds you that the mode requires the `TA-trackme-lookupmonitor` add-on on the dispatching search head (and on the remote search head when a remote account is selected).

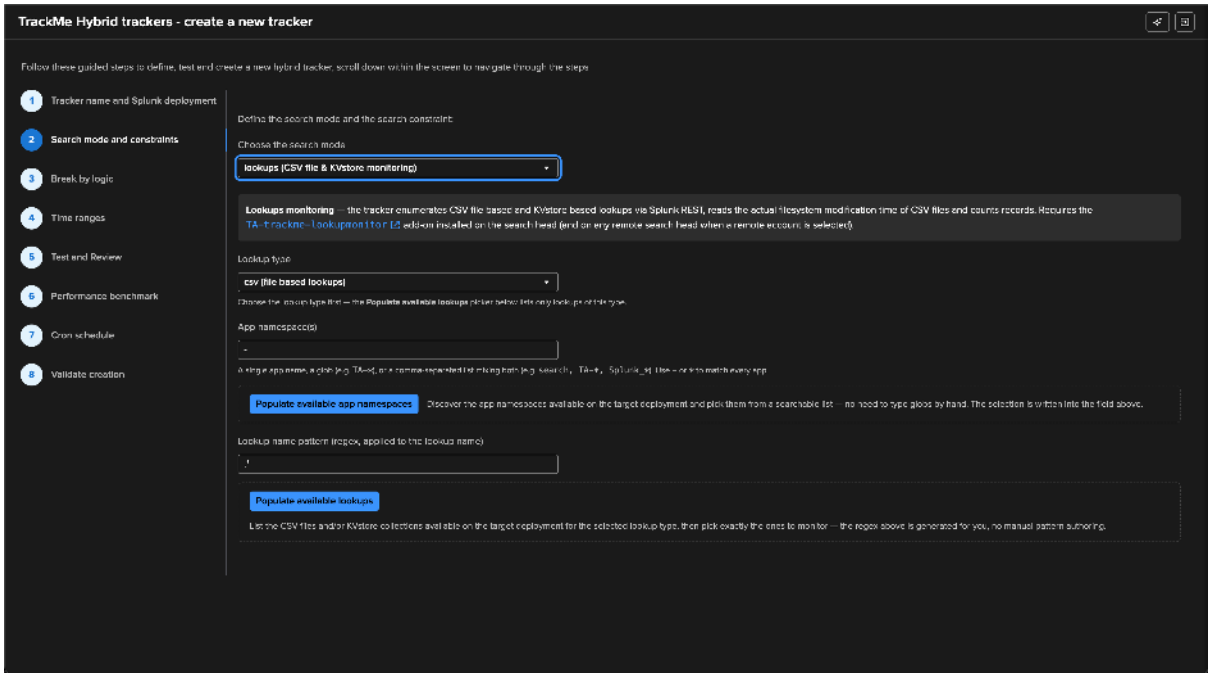
The wizard then hides the SPL constraint editor and replaces it with the lookups filter inputs:

- **Lookup type** — chosen **first**: `csv (file based lookups)`, `kvstore (collections)` or `both`. It comes first because the *Populate available lookups* picker below only lists lookups of the selected type.
- **App namespace(s)** — a single app name, a glob (e.g. `TA-*`), or a **comma-separated list mixing both** (e.g. `search, TA-*, Splunk_*`). The dash `-` or asterisk `*` means "every app the dispatching user can see". Whitespace around list items is ignored.
- **Lookup name pattern** — a regular expression applied (`re.search`) to the lookup name. Backslash escapes such as `\d`, `\w`, `\.` are preserved.

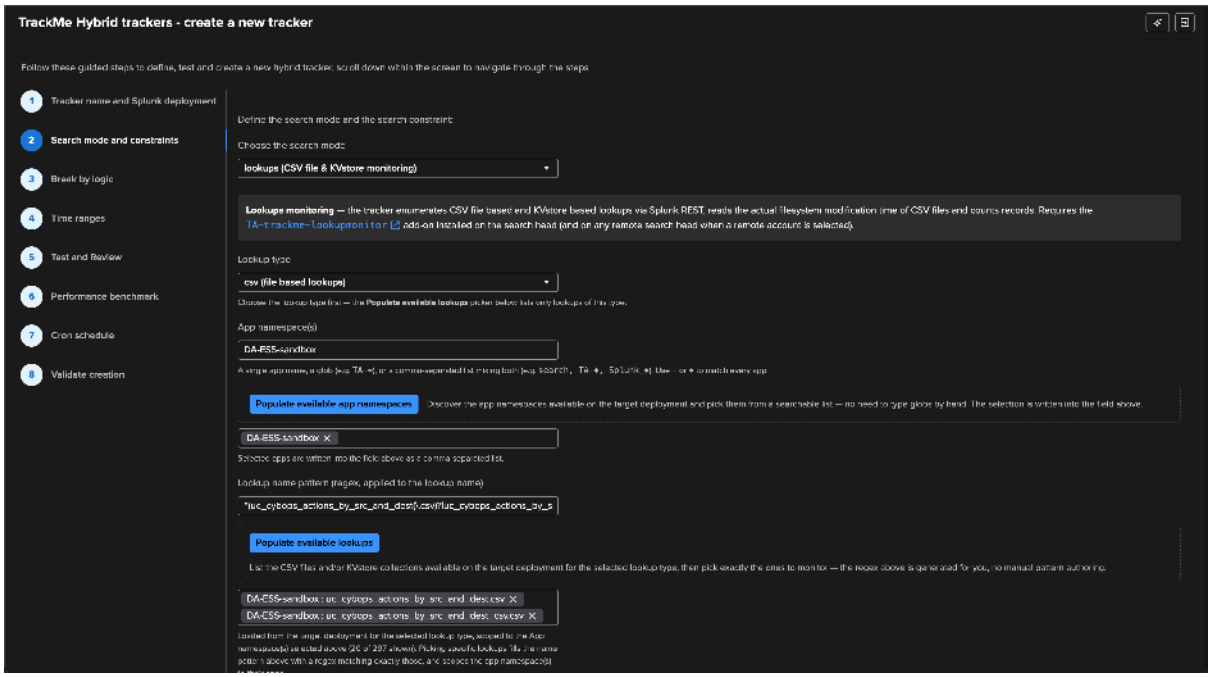
Dynamic pickers *(2.4.8)*

You no longer have to hand-write the namespace globs or the name regex. Two buttons discover what is actually available on the target deployment — local or remote — and translate your selection back into the fields above:

- **Populate available app namespaces** — lists the app namespaces present on the deployment; pick them from a searchable list and the selection is written into *App namespace(s)*, so there is no need to type globs by hand.
- **Populate available lookups** — lists the CSV files and/or KVstore collections of the selected lookup type; pick exactly the ones to monitor and TrackMe generates the matching *Lookup name pattern* for you — no manual pattern authoring.



Once you pick from the two lists, the chosen apps appear as chips and are written into *App namespace(s)* as a comma-separated list, and the selected lookups fill *Lookup name pattern* with an **anchored regex matching exactly those lookups** (the picker also narrows the app namespaces to just the apps of your selection). The list shows a capped preview — e.g. *20 of 297 shown* — but the selection itself is exact:



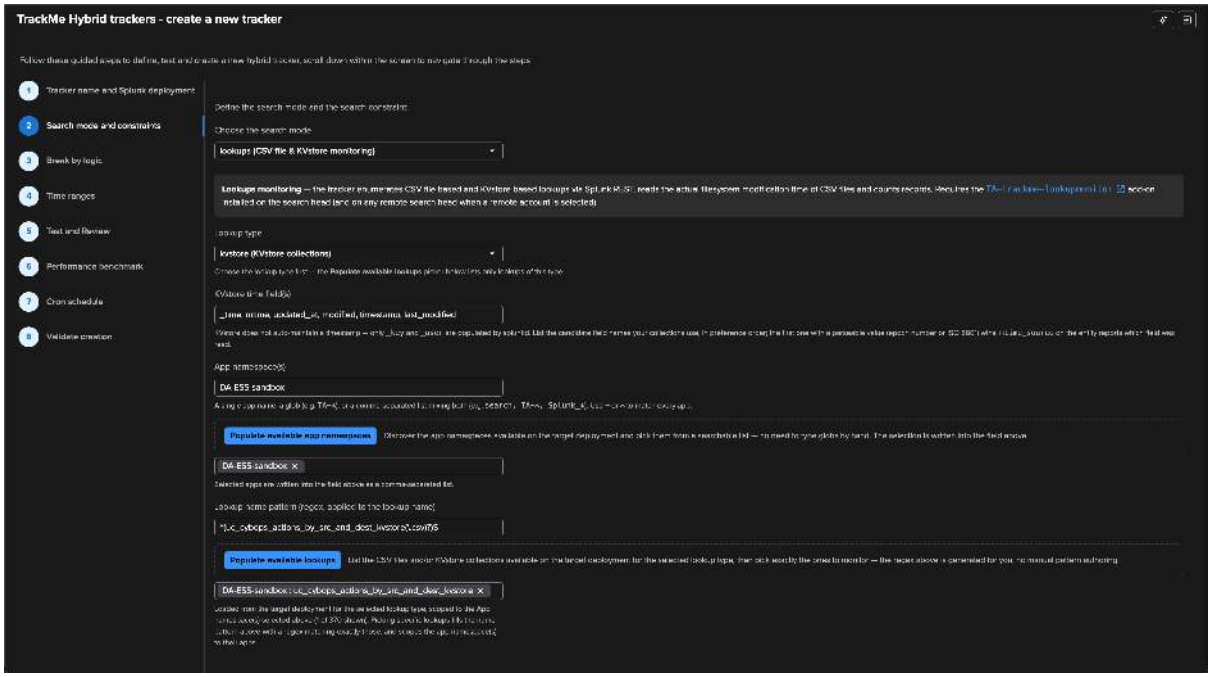
Steps 3 (Break by logic) and 4 (Time ranges) are repurposed in lookups mode:

- **Break by logic** is disabled with an explanatory banner — there is no break-by in lookups mode, the tracker creates one entity per discovered lookup.
- **Time ranges** are replaced with a single numeric field, *"Alert if the lookup has not been updated in the past (seconds)"*. The default is `86400` (24 hours). This value is stored on the entity as `data_max_delay_allowed` and is the primary alerting knob. Latency-based alerting does not apply to lookups (the tracker reads the actual filesystem mtime, not an ingestion timestamp) so the Splunk earliest/latest pickers are deliberately hidden.

Creating a lookups Hybrid Tracker — KVstore example

The KVstore flow is identical to the CSV one — including the two **Populate** pickers — with one extra input: `KVstore time field(s)`. Set **Lookup type** to `kvstore (KVstore collections)` and the *Populate available lookups* picker lists collections instead of files.

Why does KVstore need an extra input? `splunkd` only maintains `_key` and `_user` automatically on KVstore documents. There is no automatic `_time` field — it is only present if the storing application explicitly sets it. Conventions vary widely (`_time`, `mtime`, `updated_at`, `modified`, `timestamp`, `last_modified`…), so the tracker needs to know which field(s) carry the per-record modification time of your collections.



The default candidate list — `_time, mtime, updated_at, modified, timestamp, last_modified` — covers the most common conventions. The probe is **preference-ordered**: the tracker tries each candidate in order, runs

```
GET /servicesNS/-/<app>/storage/collections/data/<collection>
    ?fields=<field>&sort=<field>:-1&limit=1
```

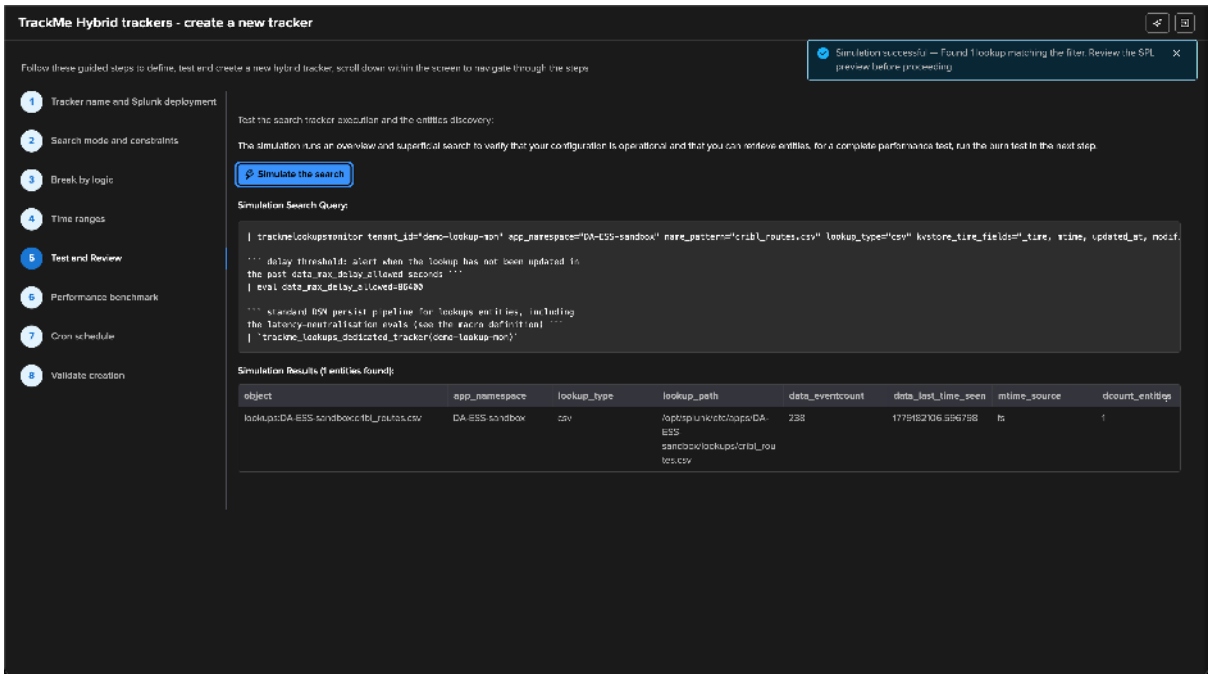
and stops at the first candidate that returns a parseable value (epoch number, numeric string, or ISO 8601 timestamp). The chosen field is recorded as `mtime_source = "kvstore:<field>"` on the emitted entity, so operators can confirm which field was read.

When none of the candidates work — i.e. the collection has no time-bearing field that the tracker can parse — `mtime_source` is set to `"unavailable"` and `data_last_time_seen` is null. The entity is still created (so its volume can be tracked) but the delay-state evaluation cannot trigger.

Step 5 (Test and Review) — simulating the discovery

The simulation step runs the discovery pipeline in preview mode and renders a results table with up to 100 entities so you can verify the filter scope and the resolved attributes before committing. The total in the success notification reflects the **true** number of matched lookups, even when more than 100 are present (the table itself is capped to keep the wizard responsive).

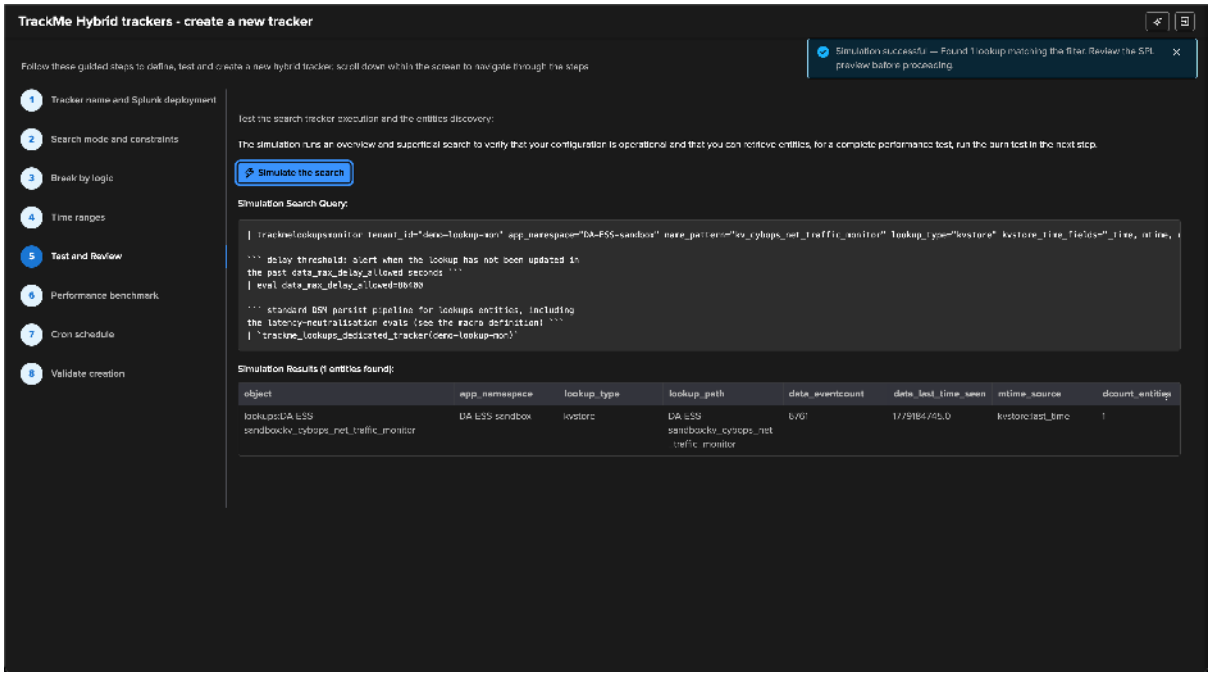
CSV example — a single lookup matching the filter:



What to look at on the simulation row:

- `object` — the entity identifier (`lookups:<app>:<name>`). This is what will appear in the DSM entity inventory.
- `app_namespace` — the Splunk app the lookup lives in.
- `lookup_type` — `csv`, `csv_gz`, `kvstore`, or `other` for non-CSV file payloads (`.kmz`, `.geo`).
- `lookup_path` — the absolute filesystem path for CSV; `<app>:<collection>` for KVstore.
- `data_eventcount` — the record count (rows for CSV, documents for KVstore).
- `lookup_size_bytes` — the on-disk size of the CSV file (`os.stat` size); not applicable to KVstore collections.
- `data_last_time_seen` — the resolved modification time (epoch seconds).
- `mtime_source` — `fs` for CSV (filesystem `os.stat` mtime), `kvstore:<field>` for KVstore (the candidate field that resolved), or `unavailable`.

KVstore example — single collection, mtime resolved from a candidate field:



In the KVstore example, `mtime_source = "kvstore:last_time"` — the tracker probed the candidate list and `last_time` was the first candidate to yield a parseable value.

Tip

If your simulation returns zero rows, the most common causes are an over-restrictive `app_namespace` glob, a `name_pattern` that does not match any visible lookup, or — for KVstore — a candidate list that does not include the field your collections actually use. The simulation SPL is printed in the wizard so you can run it directly in the search bar and adjust.

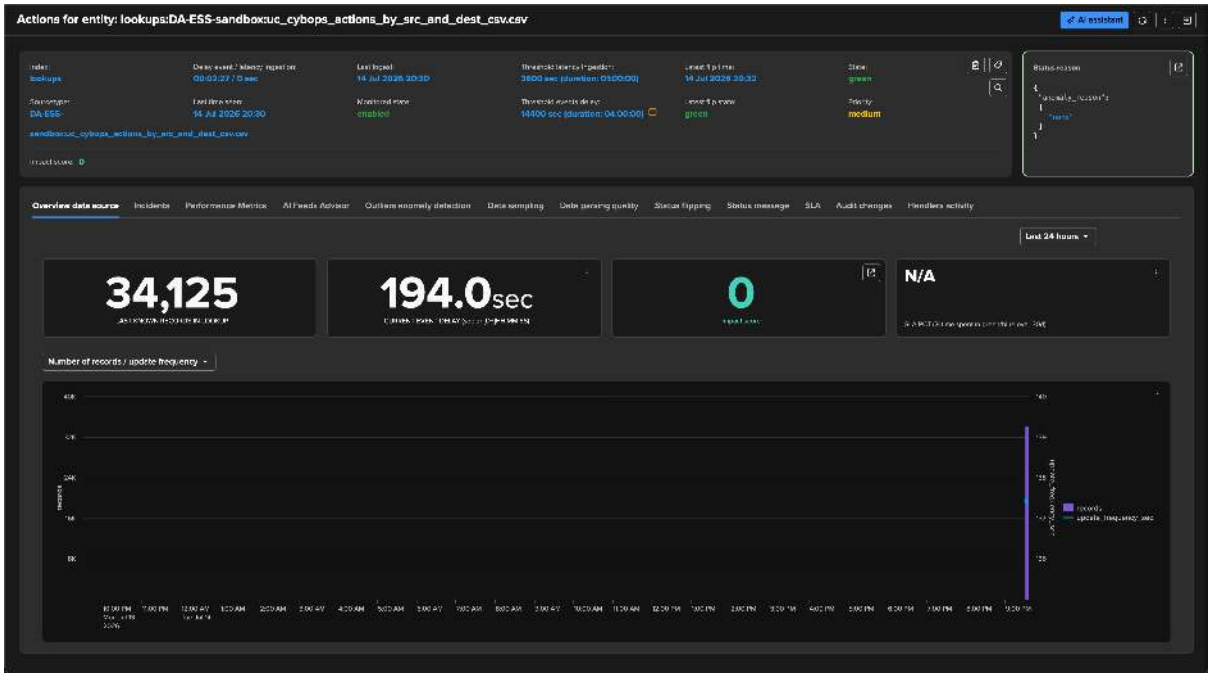
The resulting DSM entities

Each discovered lookup becomes a standard DSM entity, with `data_index = "lookups"` (a pseudo-index that classifies these entities in the inventory) and `data_sourcetype = "<app>:<name>"` (unique per lookup).

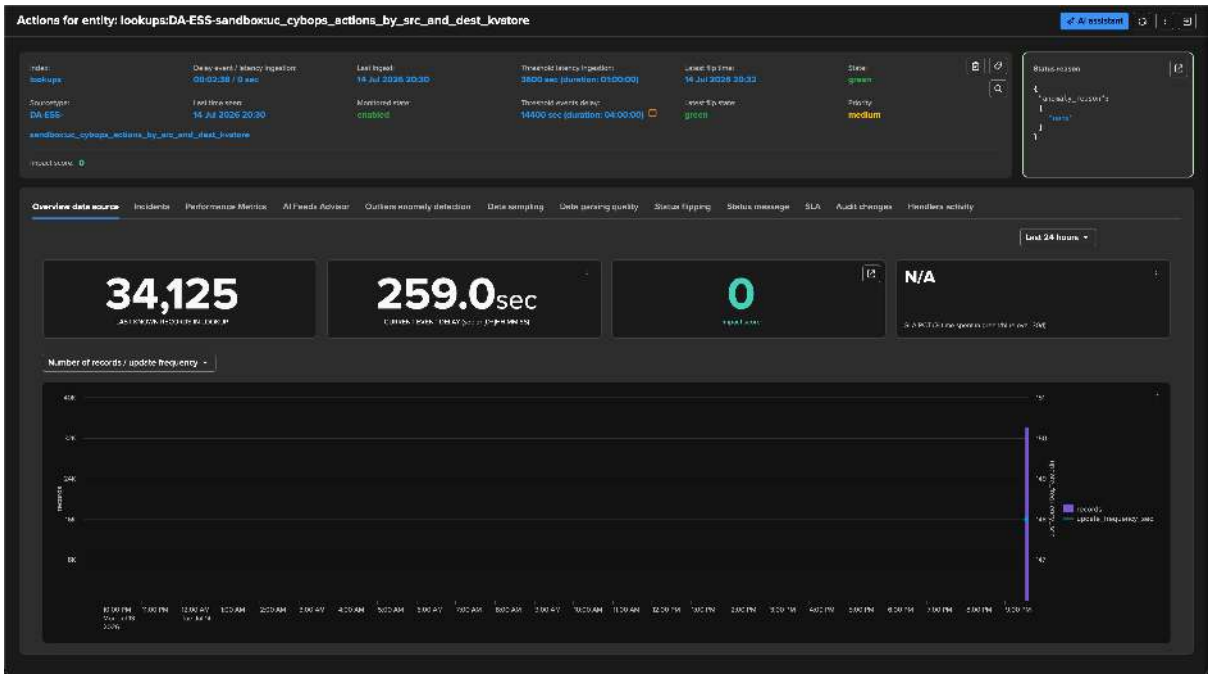
Because lookups have no `_indextime` concept, the **ingestion latency** sub-system of DSM is neutralised on lookup entities — there is no meaningful latency to measure between "the file was modified" and "Splunk indexed it". Lookup entities therefore get a **normalised, lookup-native presentation** *(2.4.8)*: the Overview leads with a **LAST KNOWN RECORDS IN LOOKUP** count tile, the latency tiles are hidden, and the data-source chart plots *number of records / update frequency* rather than volume/latency. The same normalisation carries through the status messages, the entity charts and the stateful-alert emails, which all speak *records and update frequency* instead of feed metrics.

The primary alerting signal is the **delay** since the last modification, gated by the `data_max_delay_allowed` threshold configured at tracker creation time (or edited later from the entity's *Update frequency policy* panel).

CSV-based lookup entity — note the "LAST KNOWN RECORDS IN LOOKUP" tile:



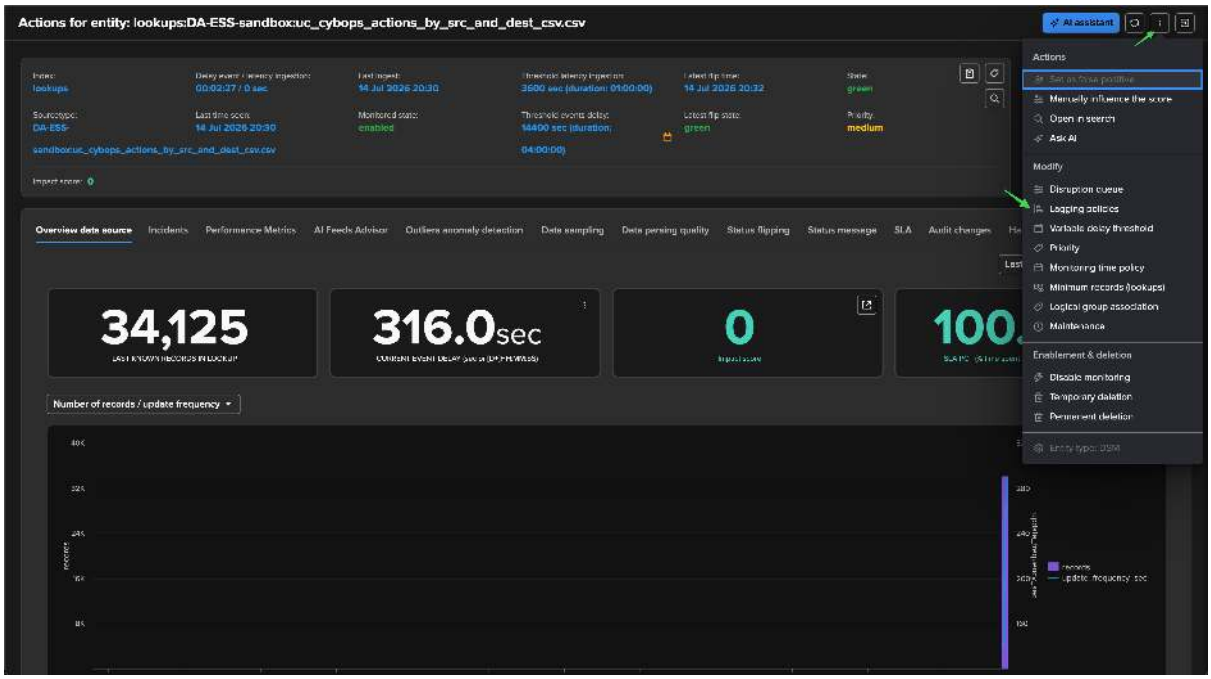
KVstore-based lookup entity:



Both entity views expose the standard DSM tabs: Overview, Incidents, Performance Metrics, Outliers Anomaly Detection, Data Sampling, Status Flipping, Status Message, SLA, Audit Changes, Handlers Activity. The Outliers detection on `data_eventcount` works out of the box — if a lookup that normally has ~50,000 rows drops to 200 (or balloons to 5,000,000) the standard outliers engine flags it without any extra configuration.

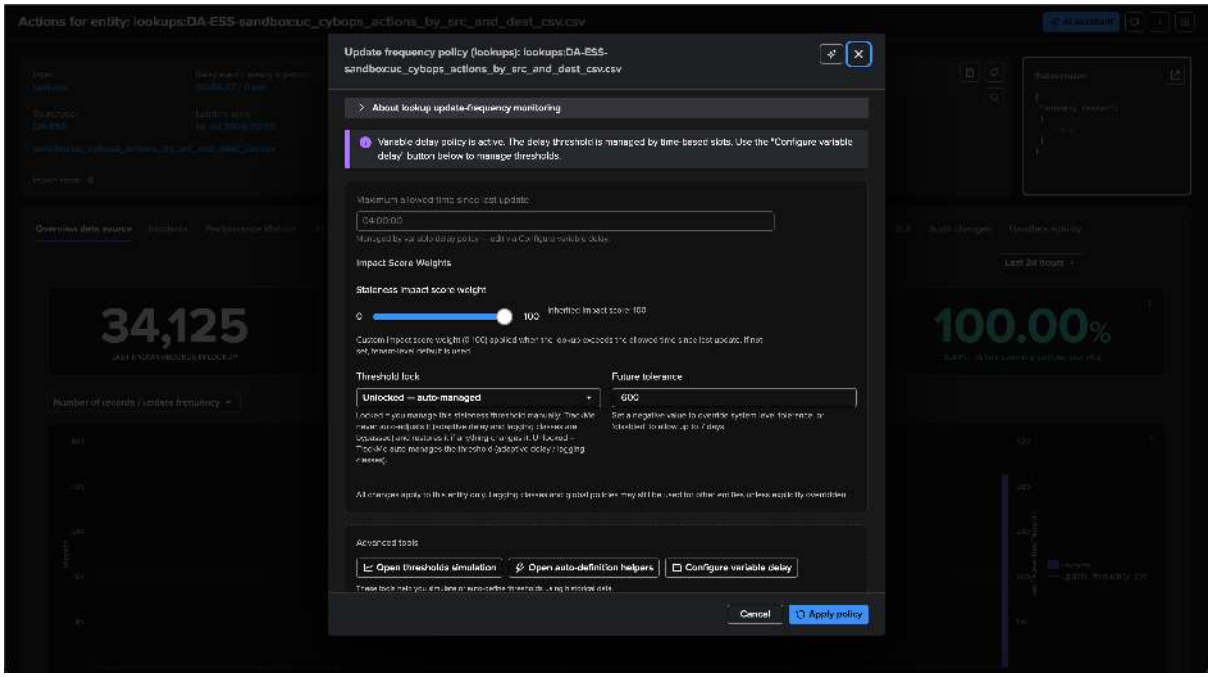
Configuring staleness alerting — the Update frequency policy

The staleness threshold and its impact-score weight are edited from the entity's **Actions** menu → **Lagging policies**:

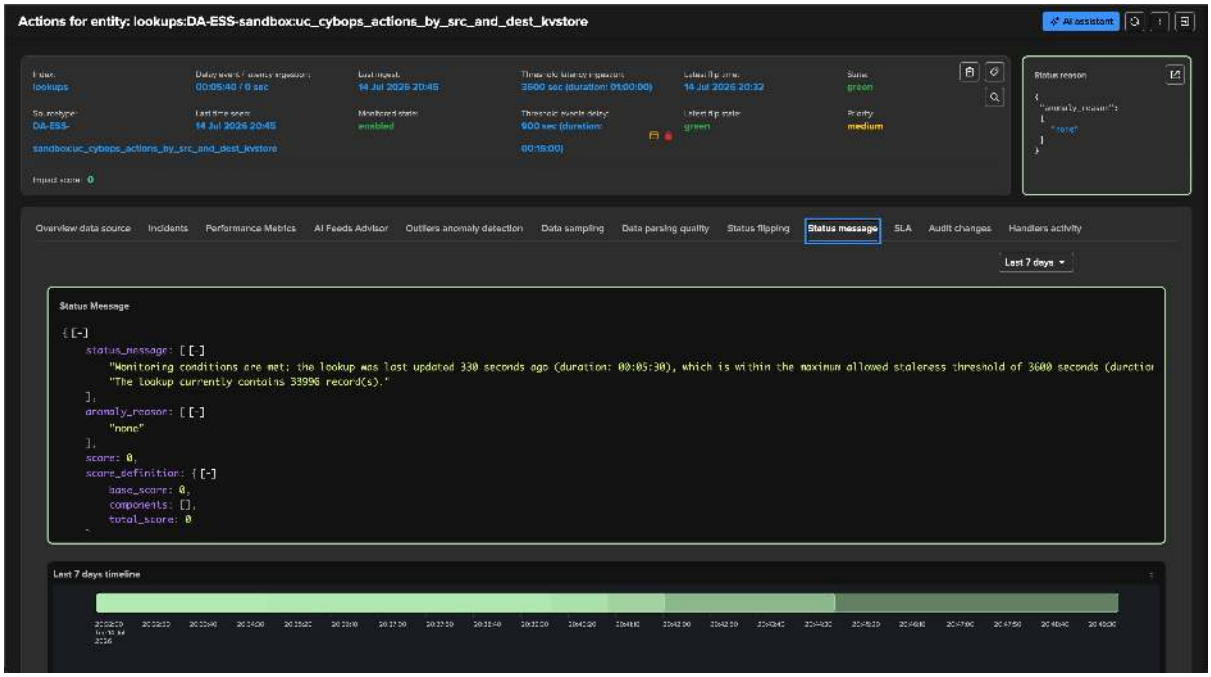


For lookup entities the screen is normalised to lookup semantics and titled **Update frequency policy (lookups)** — the latency inputs that only apply to indexed feeds are removed, and the delay controls are reframed as *update frequency / staleness*:

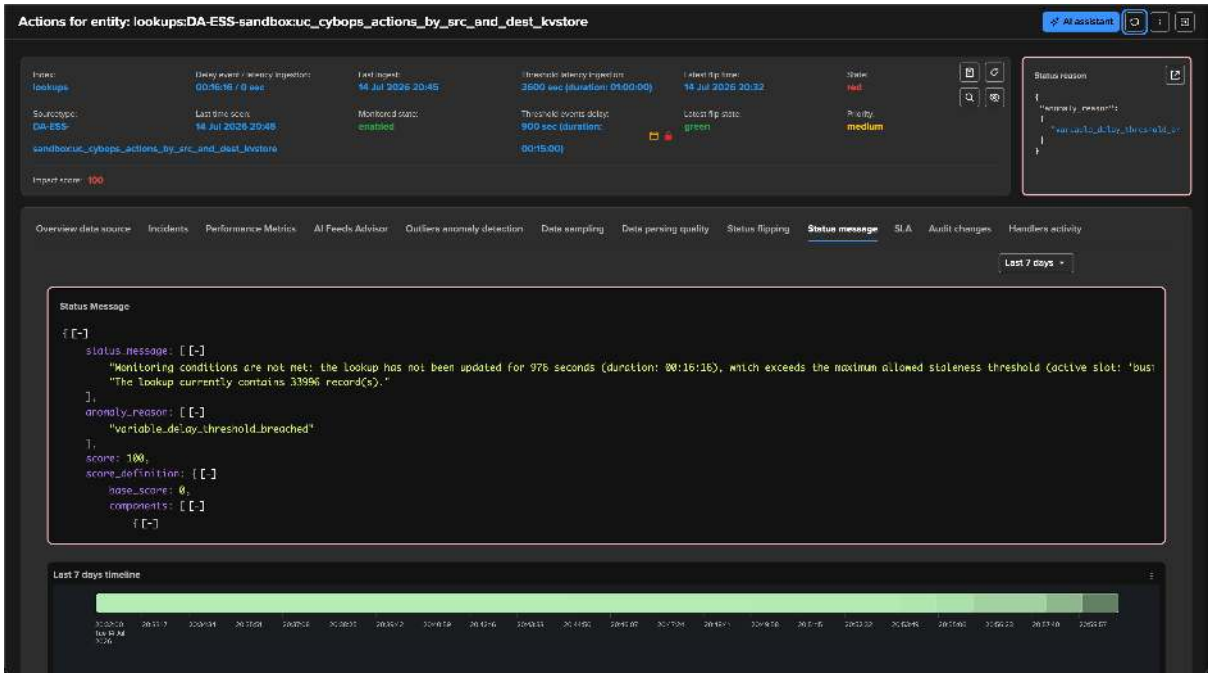
- **Maximum allowed time since last update** — the staleness threshold (the same value as `data_max_delay_allowed`). When a variable-delay policy is active it is managed by time-based slots; use *Configure variable delay* to edit those.
- **Staleness impact score weight** — a custom impact-score weight (0–100) applied when the lookup exceeds the allowed time since last update. This is the direct equivalent of the *delay* weight on feed entities; if left unset the tenant-level default is used (here it inherits `100`).
- **Threshold lock** — *Unlocked* lets TrackMe auto-manage the threshold (adaptive delay / lagging classes); *Locked* means you manage it manually and TrackMe never auto-adjusts it.
- **Future tolerance** and the **Advanced tools** (thresholds simulation, auto-definition helpers, configure variable delay) behave exactly as they do for feed entities.



When a lookup is refreshed within its allowed window the entity stays **green**: the Status message confirms the lookup was updated within the maximum allowed staleness threshold and reports the current record count, and the impact score stays at `0` with an empty score-components list:



When the lookup goes past its allowed window, the entity flips **red** and the staleness impact-score weight is applied. The Status message states how long it has been since the last update and by how much it exceeds the threshold. Here the anomaly reason is `variable_delay_threshold_breached` because a variable-delay policy with time-based slots is active; with a fixed threshold it reports `delay_threshold_breached`:



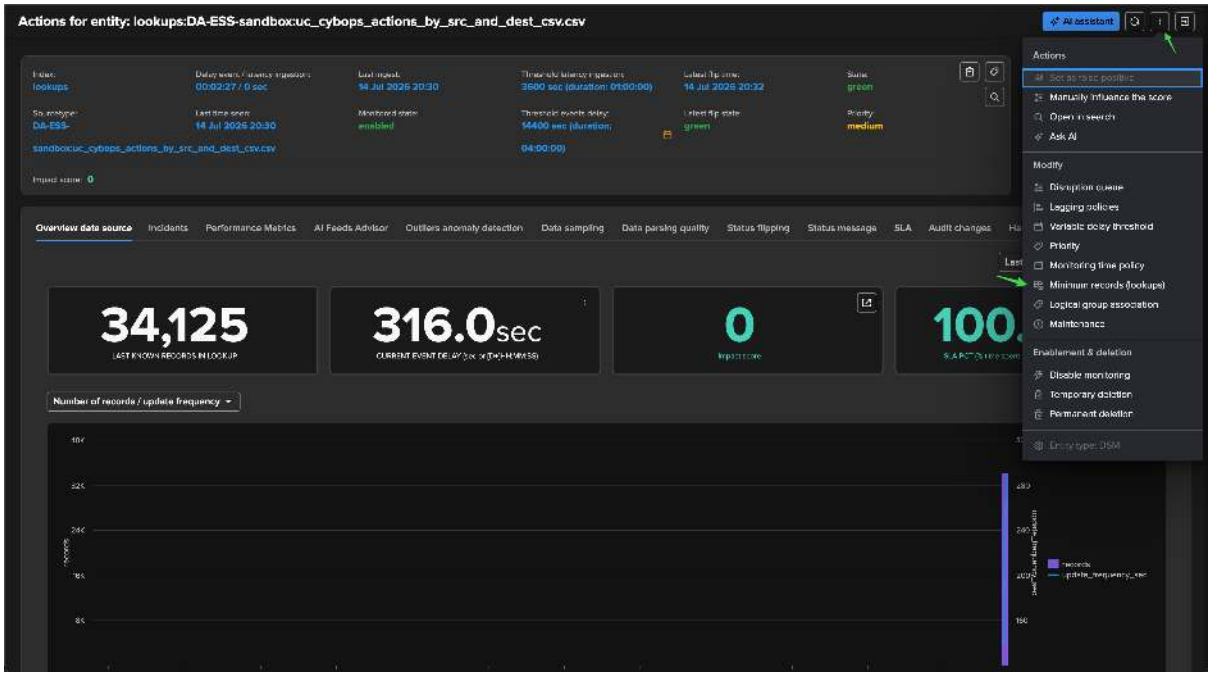
Minimum records — alerting on empty or under-filled lookups

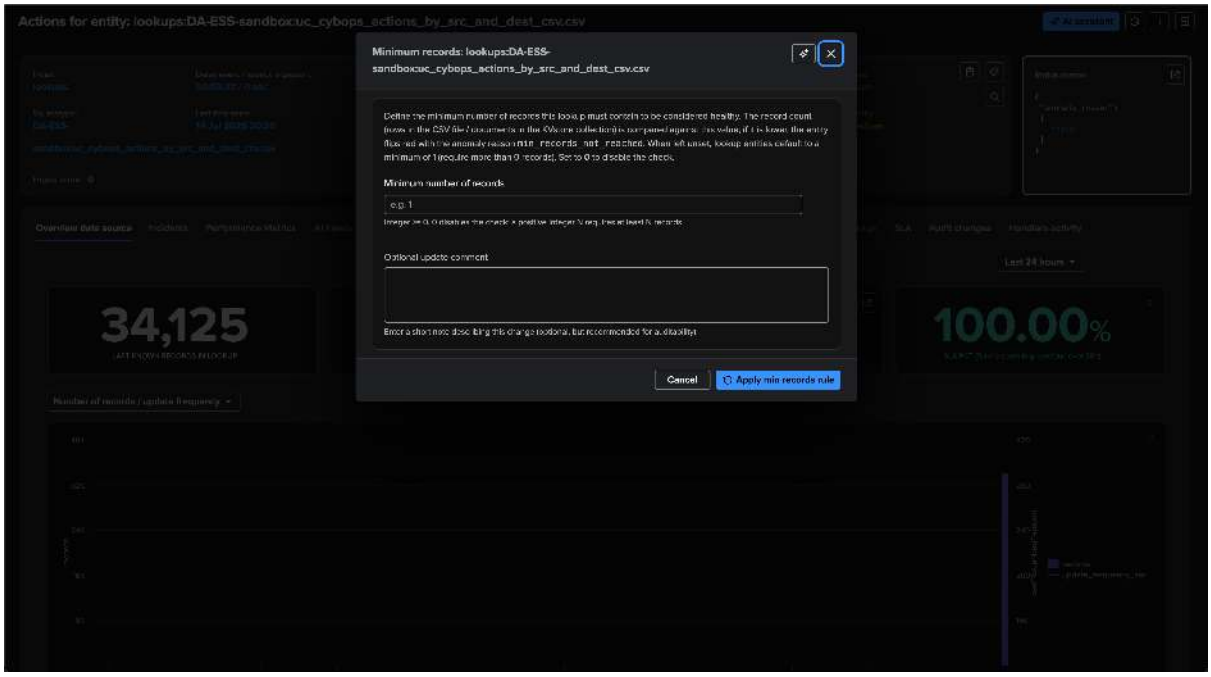
(2.4.8) Beyond staleness, TrackMe can alert when a lookup **runs empty, or drops below a minimum record count** — catching the case where a lookup is still updated on schedule but has silently lost its content (a rebuild that produced no rows, a truncated export, a broken upstream job).

Note

This is on by default: lookup entities require at least one record. When the minimum is left unset it defaults to `1`, so an **empty lookup (0 records) flips red out of the box** — no configuration needed. The record count (rows for CSV, documents for KVstore) is compared against the threshold on every run; when it is lower, the entity turns red with the anomaly reason `min_records_not_reached`.

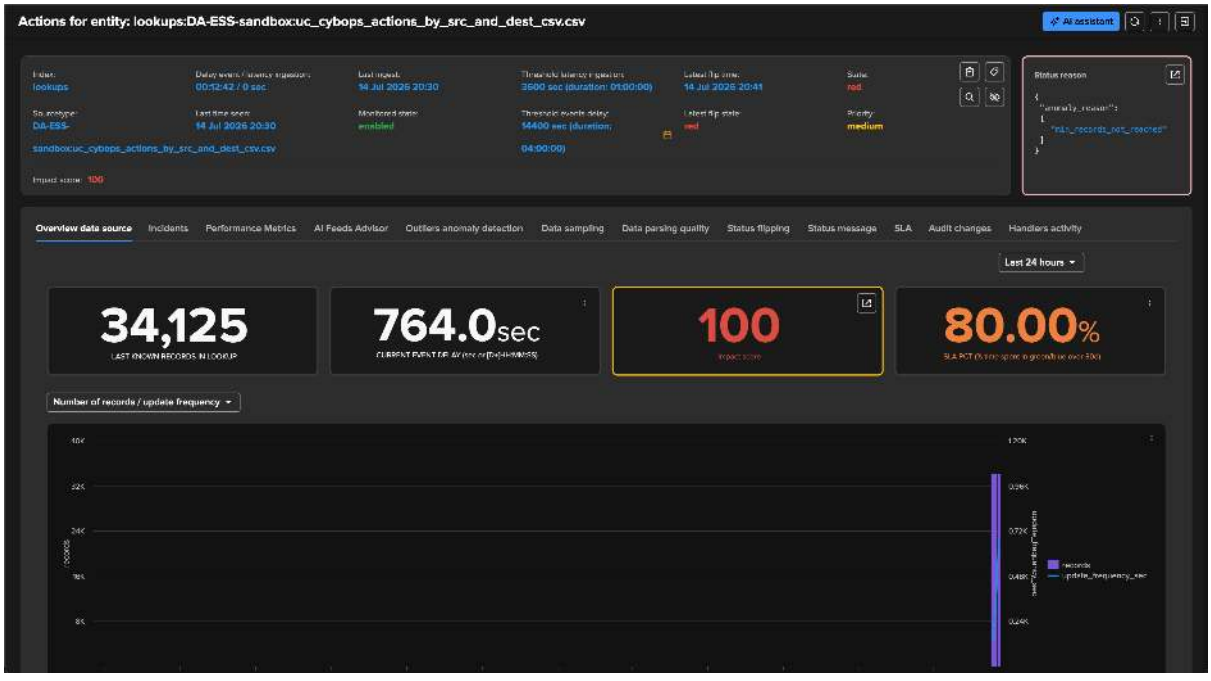
Set the threshold per entity from the **Actions** menu → **Minimum records (lookups)**:



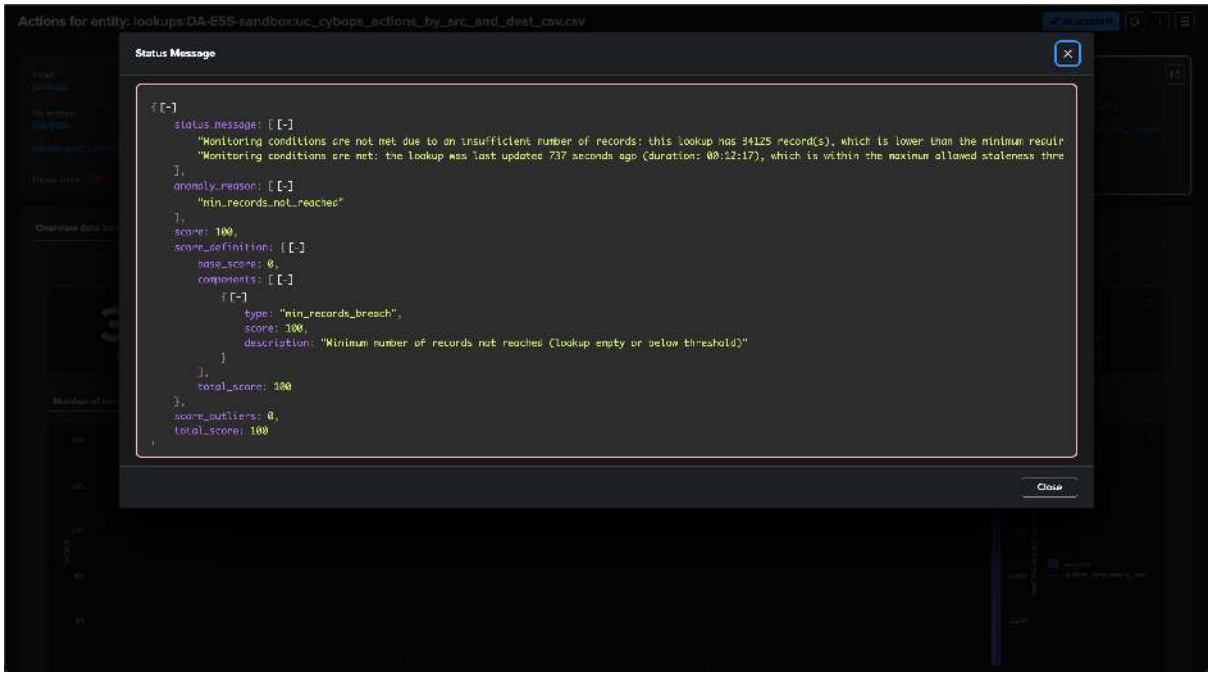


- **A positive integer N** requires at least `N` records.
- **``0``** disables the check entirely (for lookups that are legitimately allowed to be empty).
- **Unset** falls back to the default of `1` — i.e. alert on zero records.

When the condition is not met, the entity flips red, `min_records_not_reached` is added to `anomaly_reason`, and a `min_records_breach` component is added to the impact score (weight `100` by default, tunable per tenant via `impact_score_dsm_min_records_breach`):



The minimum-records and staleness checks are **independent** — the Status message spells out both, and the impact-score breakdown shows the `min_records_breach` component:



How modification times are resolved

CSV lookups

The tracker calls `/services/data/lookup-table-files` to enumerate visible lookups and resolve each one's on-disk path (`content.eai:data`), then calls `os.stat(path)` to read the filesystem mtime. That is **deliberately not** the `updated` mtime that the REST endpoint exposes — that one reflects the *knowledge-object* mtime (touched whenever permissions, ACLs, or `transforms.conf` properties change) and is not a reliable proxy for "when did the file last change?".

`mtime_source = "fs"` on the emitted entity signals this resolution path.

For `.csv.gz` files the same path is used; the row counter handles the gzip wrapper natively.

KVstore collections

For each KVstore collection that matches the filter, the tracker takes the comma-separated, preference-ordered `KV-store time field(s)` list and probes each candidate in order. The first candidate that returns a parseable value (epoch number, numeric string, or ISO 8601 — with `Z`, `+HH:MM`, `+HHMM` suffixes; naive timestamps treated as UTC) wins.

The emitted entity carries `mtime_source = "kvstore:<field>"` so the resolution is fully traceable. When no candidate works, `mtime_source = "unavailable"` and `data_last_time_seen` is null.

Document counts are obtained by cursor-walking the collection with `fields=_key` and summing the page lengths. By default the walk is **uncapped** — the `kvstore_count_cap` option defaults to `0` (no limit), so the full collection is counted. An operator can set a positive cap to bound the count on pathologically large collections; only then, when the cap is reached, is `kvstore_count_truncated = 1` set on the emitted row. The cursor page size is tunable via `kvstore_page_size` (default `10000`).

Remote search heads

The `| trackmelookupsmonitor` command is forced-local (`local = true` in `commands.conf`) so it always executes on the search head that dispatches it. To monitor lookups that live on a **remote** search head:

1. Install `TA-trackme-lookupmonitor` on the remote SH (TrackMe itself is *not* required there — that is the entire point of the TA split).
2. Configure a Splunk Remote Deployment account for that SH in TrackMe (see *Remote Splunk deployments*).

3. Select that account in Step 1 of the wizard. TrackMe will automatically wrap the lookups command with `splunkremotesearch` so the discovery runs on the remote SH and the resulting rows are streamed back for persistence.

The wrapping is transparent — you do not need to write SPL yourself. The wizard's Test and Review step renders the final SPL (with the `splunkremotesearch` wrapper applied when relevant) so you can review it before saving.

Notes and limitations

- **Permissions** — the saved search runs as the tenant owner (the user configured on the Virtual Tenant). Lookups that are not visible to that user under the configured app namespace will not be discovered.
- **Definition-only lookups** — a lookup defined in `transforms.conf` with no on-disk CSV file yet (`content.eai:data = None`) is skipped with a warn-level audit message. This is rare but not impossible (test fixtures, broken installs).
- **KVstore mtime requires an operator-maintained field** — splunkd does not auto-maintain a timestamp on KVstore documents. If your collections have no time-bearing field, the entity will still be created (so its volume can be tracked via outliers / threshold rules on `data_eventcount`) but the delay-state evaluation cannot fire — `mtime_source` will be `unavailable`. This is documented in the tracker output so it is never silent.
- **Large CSV row count cost** — the row counter is an O(filesize) streaming scan. A 10 GB lookup on local NVMe completes in 10-20 seconds; this is acceptable on minute-scale schedules but worth keeping in mind on multi-GB `.csv.gz` lookups (decompression cost dominates). Lookups are inspected **in parallel** to keep discovery fast over many objects — the concurrency is controlled by `max_workers` (default `min(cpu*2, 8)`), which you can lower to cap the load on a busy search head.
- **Stat / count race** — modification time is read first, row count second. Under a concurrent write, the worst-case result is "new mtime + previous-cycle count" for a single tick. This is acceptable and self-corrects on the next run.

See also

- *Hybrid trackers* — Hybrid Trackers wizard entry points and shared fields.
- *Remote Splunk deployments* — configuring remote Splunk accounts (required for remote-SH lookups monitoring).
- *Elastic sources* — the *Elastic source example 2: tracking lookups update and number of records* section is now superseded by the lookups search mode documented here.
- TA-trackme-lookupmonitor download page — dedicated add-on distribution site.

8.9.11 Splunk SOAR

TrackMe integrates with Splunk SOAR in two directions. This page covers **monitoring SOAR's own health** and **pushing TrackMe events into SOAR**. Driving TrackMe *from* SOAR is covered separately in *TrackMe App on SOAR*.

Monitoring SOAR health

You can monitor a Splunk SOAR deployment the same way you monitor anything else — with *Flex Objects*. A library of ready-made SOAR use cases covers asset health, automation-broker high availability, and infrastructure load (CPU, memory, services). Load a use case in a Flex-enabled tenant, adapt the SPL if needed, simulate, and create — TrackMe then tracks SOAR as tracked entities with the full state, scoring, and alerting model.

Pushing TrackMe events to SOAR

To drive SOAR playbooks from TrackMe, push TrackMe's signals into SOAR:

- **Forward notable events** (the recommended path) — TrackMe writes *notable events* to its notable index; the *Splunk App for SOAR* forwards them into SOAR as containers, where a playbook can act on them.

- **Push directly** — the `trackmesplksoar` command builds a SOAR container from an entity's context and posts it; it can be driven automatically from a *stateful alert* via the alert's *command* delivery channel.
- **Enrich back** — the `trackmesplksoarlookup` command pulls live SOAR object metadata (container, case, playbook, broker…) into a search, to decorate TrackMe entities with SOAR context.

Note

The push direction is **one-way and on-demand** — TrackMe does not poll SOAR on a schedule. Automate the enrichment lookup with a saved search if a dashboard must stay current, and spread large pushes (for example with `| head`) to avoid SOAR rate limits.

See also

- *SOAR monitoring — in depth* — the full use-case catalogue, the Automation Broker HA design, and the SOAR command references.
- *TrackMe App on SOAR* — the reverse direction: control TrackMe from SOAR.
- *FLX — Flex Objects* — the Flex use cases that monitor SOAR.
- *Notable events* — the notable events forwarded to SOAR.

8.9.12 SOAR monitoring — in depth

Tip

This is the **in-depth reference** for monitoring Splunk SOAR with TrackMe. For the readable overview, start with *SOAR monitoring*. This page covers the full use-case catalogue and entity grouping, the Automation Broker HA design, and the `trackmesplksoar` / `trackmesplksoarlookup` command references.

Hint

Version 2.0.46 and later

- The SOAR integration requires TrackMe version 2.0.46 and later
- SOAR High Availability management for Automation Brokers is available from TrackMe version 2.0.50 and later

Hint

New since TrackMe 2.1.10: REST lookup capabilities and review of the adhoc/playbook failure detections use cases

- In TrackMe 2.1.10, we have introduced a new REST lookup capability with the command `trackmesplksoarlookup` and fully refactored the adhoc and playbook failure detections use cases to leverage this new capability.
- With this command, TrackMe performs REST based live lookups to SOAR, allowing to retrieve the metadata associated with the SOAR objects such as the playbooks, assets, applications and brokers associated with the detection.
- These metadata information are then added to the extra_attributes field of the resulting entities, allowing to have a full context of the detection and to be able to review the details of the detection directly from the TrackMe UI.

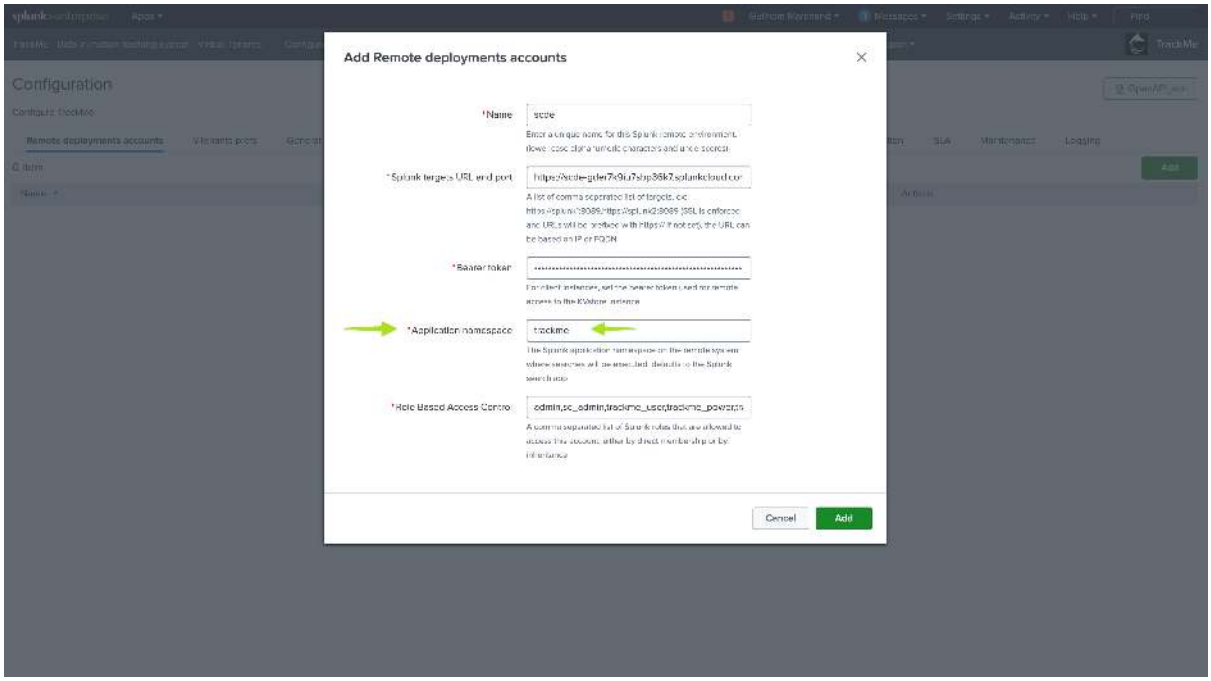
Hint

New since TrackMe 2.1.11: REST based SOAR monitoring of playbooks run concurrency and statuses

- In TrackMe 2.1.11, we have introduced a new REST based SOAR monitoring of playbooks run concurrency and statuses, leveraging the REST lookup capabilities.
- This allows to monitor the concurrency of playbooks run and their statuses, and to be alerted in case of issues.
- This is available out of the box with the use case `splk_soar_concurrent_playbooks`.

Splunk Remote Account: You must use the trackme namespace

- If you use a remote account to run the SOAR active monitoring (Automation Brokers High Availability), you must use the TrackMe namespace when configuring the Splunk remote account.
- This is required because TrackMe by default is shared at the application level, so unless you share TrackMe at the system level, your remote account would not be able to access to TrackMe custom commands on the remote partner.



Minimal permissions for TrackMe SOAR monitoring

- On the Splunk side, when setting up a dedicated Splunk user account for the purposes of SOAR monitoring, notably active monitoring such as the Automation Broker high availability management, you must honor some minimal permissions.
- This is applicable to both local monitoring, and **remote** account via TrackMe.
- The user account must be a member of the `trackme_admin` role (or have the `trackmeadminoperations` capability), the user should also be a member of the Splunk builtin `user` account, or have equivalent capabilities.
- Regarding access to Splunk indexes, you must grant access to the service account to all `phantom_*` indexes, and eventually to the `_internal` indexes depending on any custom Flex Object tracker you would create in addition with TrackMe builtin use cases.

- Finally, resources should also be sufficient to avoid hitting a max concurrent search quota, and storage quota, for instance up to 5 concurrent search and 1GB file storage quota.

1. Introduction to SOAR monitoring

TrackMe provides builtin use cases to actively and efficiently monitor at scale one or more Splunk SOAR environments.

SOAR Cloud & on-premise monitoring is performed via the TrackMe Flex Object component (splk-flx) which is a restricted component not available with the Foundation Edition trial.

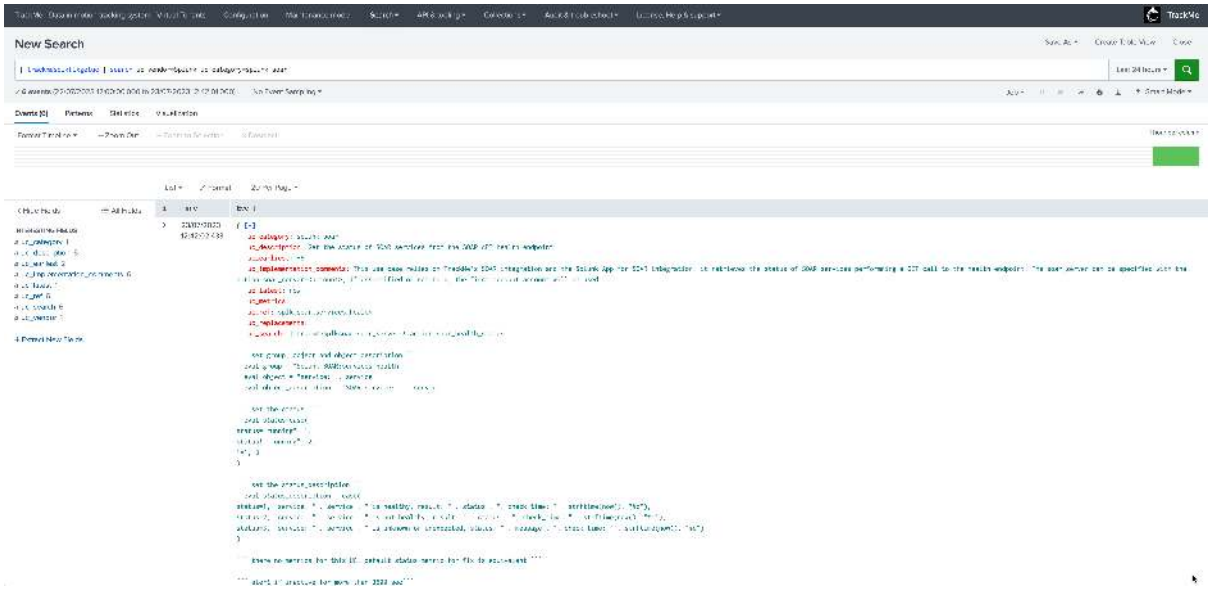
The monitoring relies on TrackMe's SOAR integration and the Splunk Application for SOAR integration, associated with TrackMe's Flex Object concepts, and provides the following use cases:

uc_ref	uc_description	uc_implementation_comments
splk_	Monitor the concurrency of playbooks run	This use case relies on SOAR events indexed in Splunk through the Splunk Application for SOAR integration. It monitors for playbooks run concurrency and triggers an alert if the concurrency limit is exceeded. This use case also leverages a live REST API lookup capability to retrieve the SOAR object context metadata.
splk_	Monitor for SOAR adhoc actions failures	This use case relies on SOAR events indexed in Splunk through the Splunk Application for SOAR integration. It monitors for app run action failures and triggers shall there be any failures, the alert will clear itself once failures cannot be found in the time range. You may want to adapt the search time range according to your preferences to keep the alert active for a certain amount of time after detection happened. This use case also leverages a live REST API lookup capability to retrieve the SOAR object context metadata.
splk_	Monitor for SOAR playbook actions failures	This use case relies on SOAR events indexed in Splunk through the Splunk Application for SOAR integration. It monitors for playbook action failures and triggers shall there be any failures, the alert will clear itself once failures cannot be found in the time range. You may want to adapt the search time range according to your preferences to keep the alert active for a certain amount of time after detection happened. This use case also leverages a live REST API lookup capability to retrieve the SOAR object context metadata.
splk_	Runs an active assets health check, parse and render results	This use case relies on TrackMe's SOAR integration and the Splunk App for SOAR integration, if active_check is set to True, it performs an active asset health check (connectivity test) by discovering assets, running a POST call to request the connectivity test, then parses and renders results for each asset. If active_check is set to False, it simply parses assets and checks the latest connectivity test achieved by SOAR and renders results. Running the active check requires the automation user in SOAR to have the edit asset permissions, if you do not want this or cannot set it, you can disable the active check and rely on SOAR doing it automatically once per day, therefore a failure detection can take up to 24 hours, enable active check is recommended. The soar_server can be specified with the option soar_server=<account>, if unspecified or set to *, the first account will be used. You can restrict the list of assets to be taken into account using the assets_allow_list option, and/or avoid some assets to be taken into account using assets_block_list option. The SOAR automation used for the SOAR server account configuration needs to be able to update assets if active check is enabled as it performs an active test request against the SOAR REST API.
splk_	Monitor SOAR CPU load from the SOAR API health endpoint	This use case relies on TrackMe's SOAR integration and the Splunk App for SOAR integration, it retrieves the 1/5/15 minutes CPU load performing a GET call to the health endpoint. The soar_server can be specified with the option soar_server=<account>, if unspecified or set to *, the first account will be used.
splk_	Monitor SOAR CPU load from the SOAR API health endpoint	This use case relies on TrackMe's SOAR integration and the Splunk App for SOAR integration, it retrieves and calculates the percentage of memory usage of the SOAR instance performing a GET call to the health endpoint. The soar_server can be specified with the option soar_server=<account>, if unspecified or set to *, the first account will be used.
splk_	Get the status of SOAR services from the SOAR API health endpoint	This use case relies on TrackMe's SOAR integration and the Splunk App for SOAR integration, retrieving the status of SOAR services performing a GET call to the health endpoint. The soar_server can be specified with the option soar_server=<account>, if unspecified or set to *, the first account will be used.
splk_	Monitor the status of SOAR Automation Brokers and optionally update SOAR Assets for High Availability and Failover man-	This use case relies on TrackMe's SOAR integration and the Splunk App for SOAR integration, it monitors the status of Automation Brokers and can automatically update SOAR Assets as needed when Automation Brokers are detected as inactive, this provides automated High Availability and failover capabilities for SOAR Automation Brokers via TrackMe's integration.

These use cases are provided via the Flex Object use cases library, but not that you can also manually implement new use cases, or customise builtin use cases as needed.

You can review the use case details ahead of their creation in TrackMe with the following command:

```
| trackmesplkflxgetuc | search uc_vendor=Splunk uc_category=splunk_soar
```



2. Requirements for SOAR monitoring

SOAR Cloud & SOAR on-premise

TrackMe's SOAR integration is compatible with both SOAR Cloud and SOAR on-premise deployments.

Splunk Application for SOAR integration

TrackMe's integration for SOAR relies on the Splunk SOAR integration for Splunk provided by the Splunk App for SOAR:

- https://splunkbase.splunk.com/app/6361

Hint

Splunk App for SOAR

- TrackMe performs active and bi-directional interactions with the SOAR API
- TrackMe relies on the Splunk App for SOAR integration and connectivity definition to achieve this job
- You do not need to have any additional configuration in TrackMe as long as the Splunk App for SOAR is installed and configured
- Some of the SOAR use cases rely on the SOAR events indexed in Splunk (index-phantom_*) which is also part of the Splunk App for SOAR integration
- TrackMe leverages the Splunk App for SOAR and extends its capabilities even further

TrackMe tenant with the Flex Object component for SOAR

You need a TrackMe tenant with the Flex Object component enabled, you can decide to create a dedicated tenant for the monitoring of SOAR, and use any existing tenant of your choice.

Once the Flex trackers have been created, TrackMe automatically groups the resulting entities for SOAR into the following groups:

uc_ref	grouping
splk_soar_actions_apprun_failures, splk_soar_actions_playbooks_failu	Splunk_SOAR:actions_health
splk_soar_assets_health	Splunk_SOAR:asset_health
splk_soar_infra_load, splk_soar_services_health	Splunk_SOAR:infrastructure
splk_soar_services_health	Splunk_SOAR:services_health
splk_soar_automation_brokers_manage	Splunk_SOAR:automation_broker_manage

TrackMe deployment target

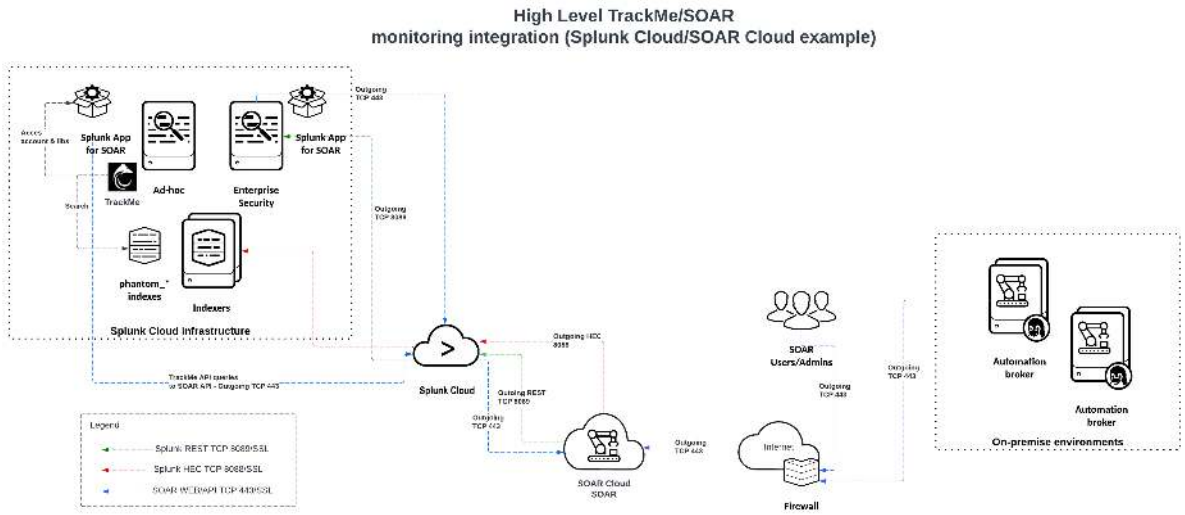
In TrackMe, you can choose where the SOAR monitoring Flex Objects use cases are executed, this can be either "local" or a remote deployment.

Note that some of the use cases would require TrackMe to be deployed on the remote target, as well as the Splunk App for SOAR configured on the remote target.

3. Implementation of SOAR monitoring

3.1 Integration architecture overview

The following diagram represents the integration from an high level perspective:



As a brief summary:

- REST API related use cases in TrackMe imply an active interaction with SOAR API
- TrackMe retrieves and loads the SOAR account configuration from the Splunk App for SOAR, you do not need to define anything in TrackMe
- TrackMe implements its own REST queries against the SOAR API as needed, in some cases such as the Assets connectivity check, this interaction may imply a bi-directional integration with POST and GET calls performed by TrackMe
- Some other use cases only deal with the SOAR data indexed in Splunk, and do not imply an API interaction

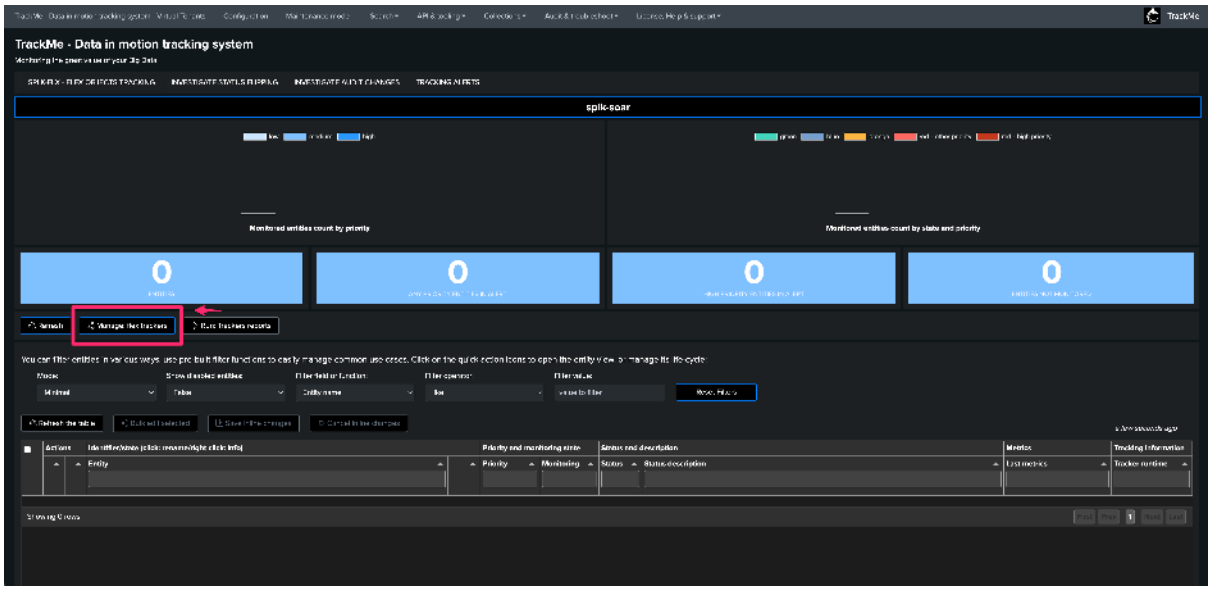
3.2 Creating SOAR Flex Object trackers

Hint

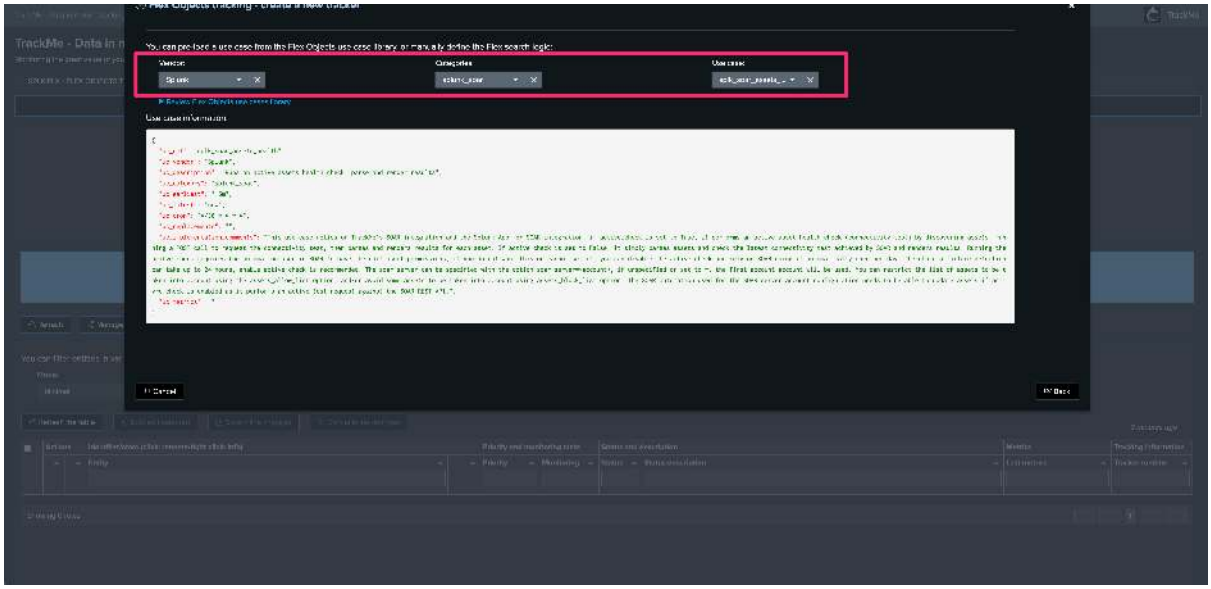
Running actions on a remote Splunk Search Head tier

- You can leverage TrackMe's splunkremotesearch to run these actions transparently on a remote Splunk Search Head tier
- In this case, the Splunk App for SOAR would be installed and configured on the target (not on TrackMe's Search Head)
- However, TrackMe needs to be be installed on the target Splunk Search Head tier, although it is unconfigured and not actively running anything (if you run in Splunk Cloud Victoria, TrackMe is installed on all Search Heads automatically)
- You need to create a TrackMe remote account which uses a token related a Splunk user on the remote Search Head
- TrackMe leverages a minimalist least privileges approach, the user on the remote Search Head tier only needs to be a member of the power role and trackme_admin role (or have equivalent capabilities)
- This requires TrackMe version 2.0.48 and later as we have addressed some issues to allow minimal permissions to be used

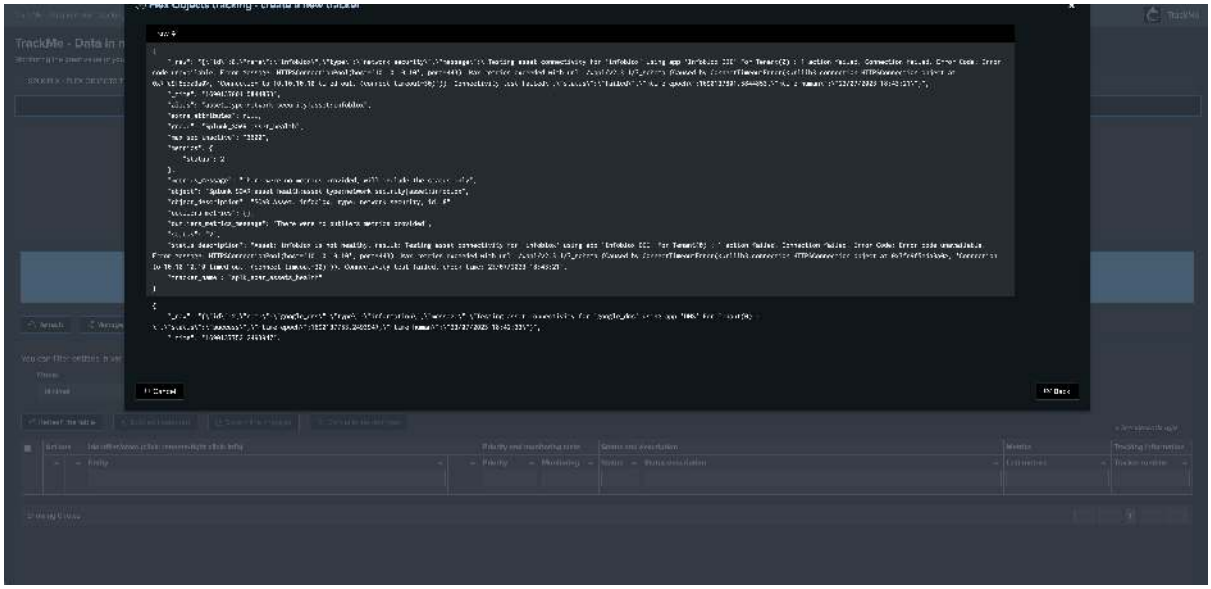
Once a TrackMe Virtual Tenant with the Flex Object (splk-flx) has been created, the setup is really straightforward and done via the UI:



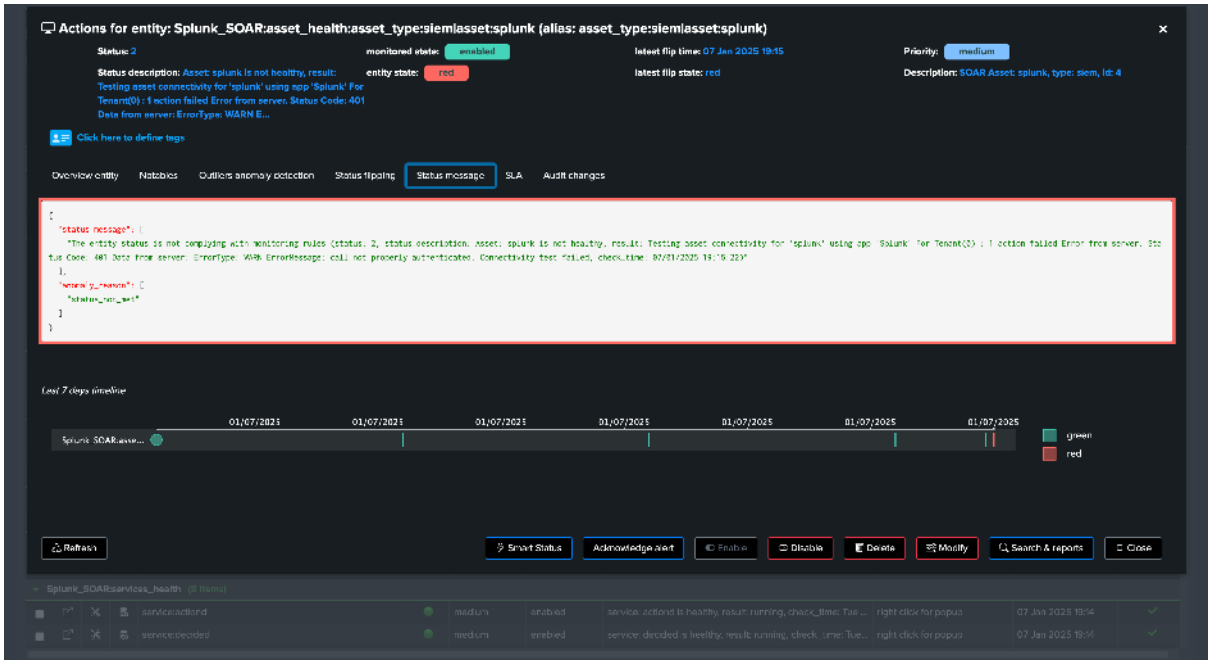
Then click on Create a new Flex Tracker, select Splunk as the vendor and splunk_soar as the category, the next step is to review and validate the results, this is all:



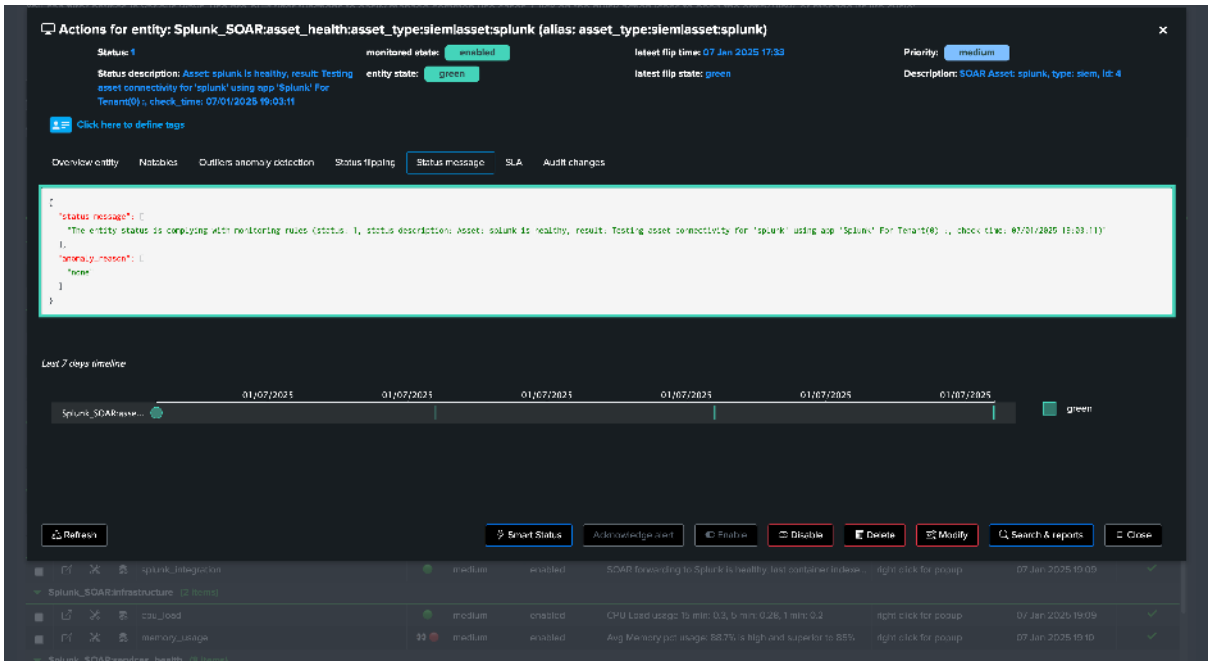
Example of results with the Assets active health check:



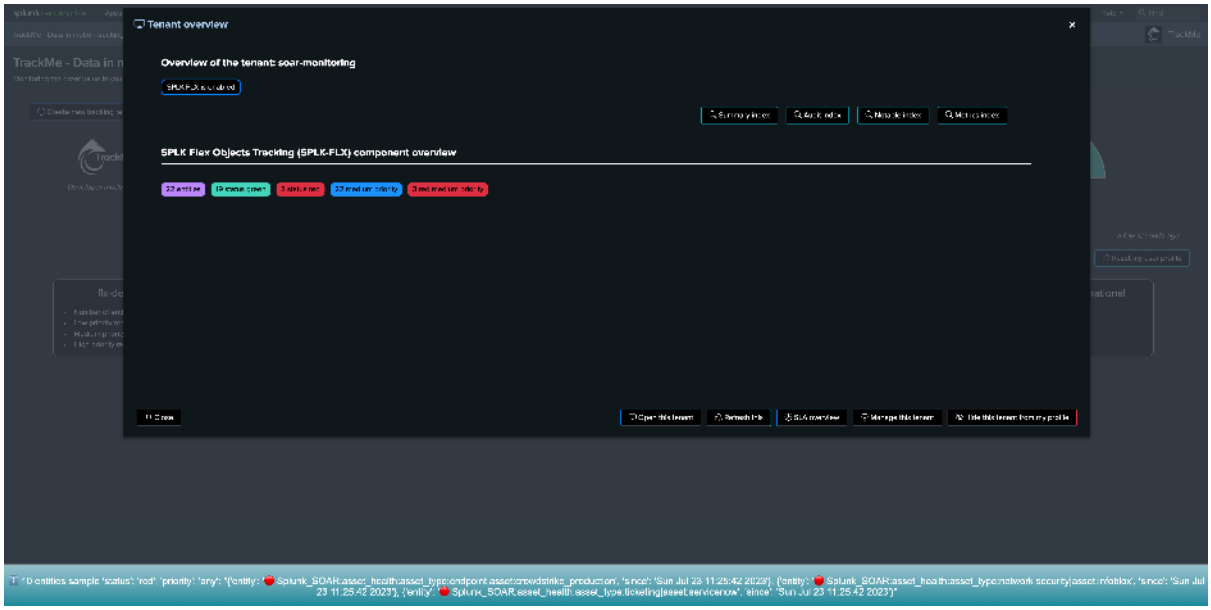
This asset is failing the connectivity check:

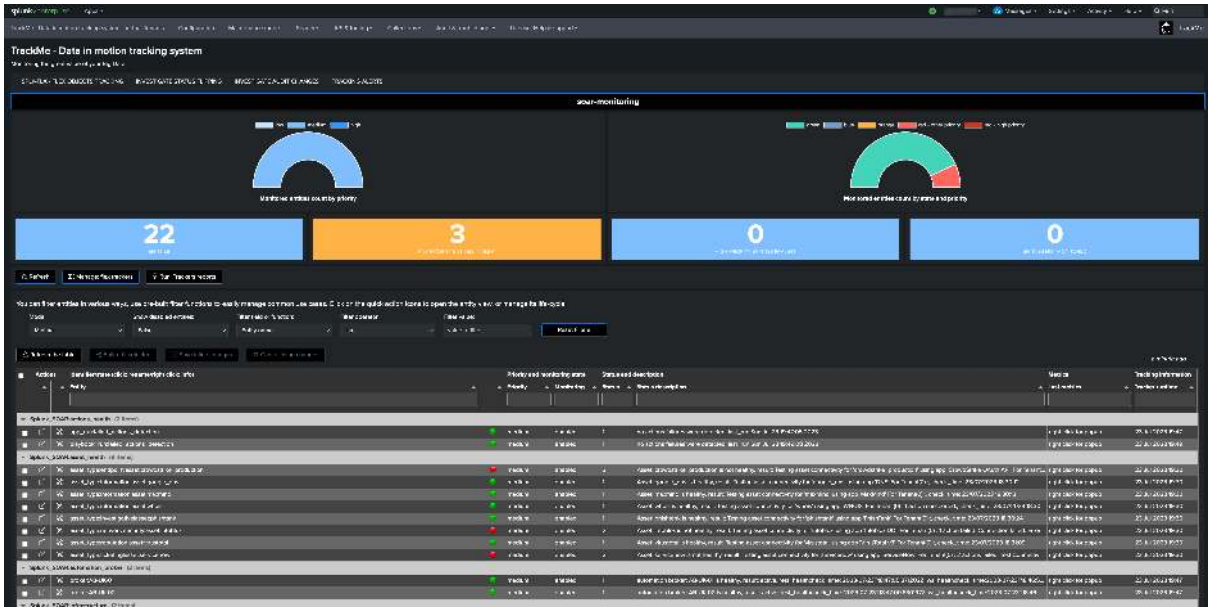


This asset passes successfully:



Setup is completed, the number of resulting entities depends on your environment:





4. SOAR Connectors Assets status active health check

A key monitoring capability for SOAR is to be able to detect as soon as possible when an application and its corresponding asset is experiencing an issue.

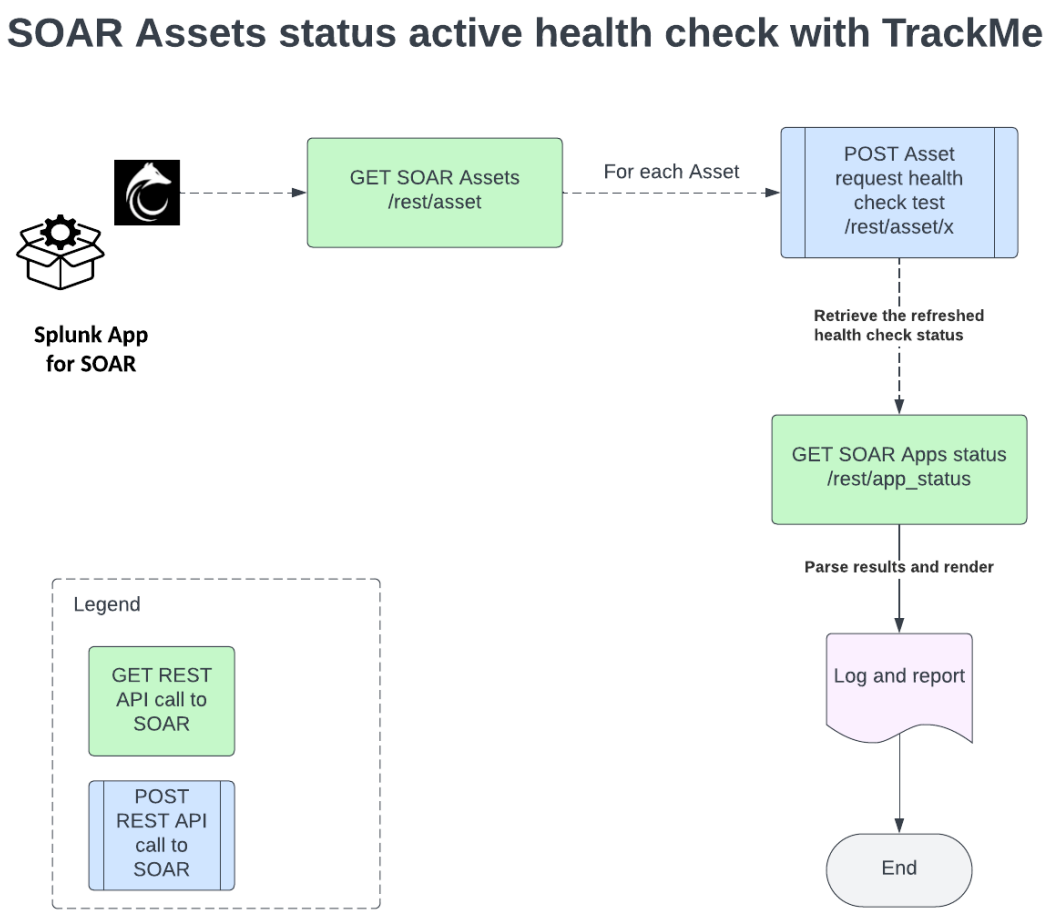
An asset that fails the status health check means that actions cannot be performed anymore, either ad-hocs actions requested by analysts or automated actions in the context of SOAR Playbooks.

Failures can happen for a variety of reasons, such as service accounts credential expiration, firewall rules changes or loss, API changes, SOAR Application upgrades, and many more.

This is an highly critical aspect of SOAR monitoring, especially true as Playbooks will only execute an action when needed, so the Application integration failure could remain undetected before it's too late.

SOAR has a builtin and mandatory connectivity feature per application, and per asset, however this health check is not continuously performed by SOAR, so only monitoring the `/rest/app_status` endpoint would result in monitoring outdated information.

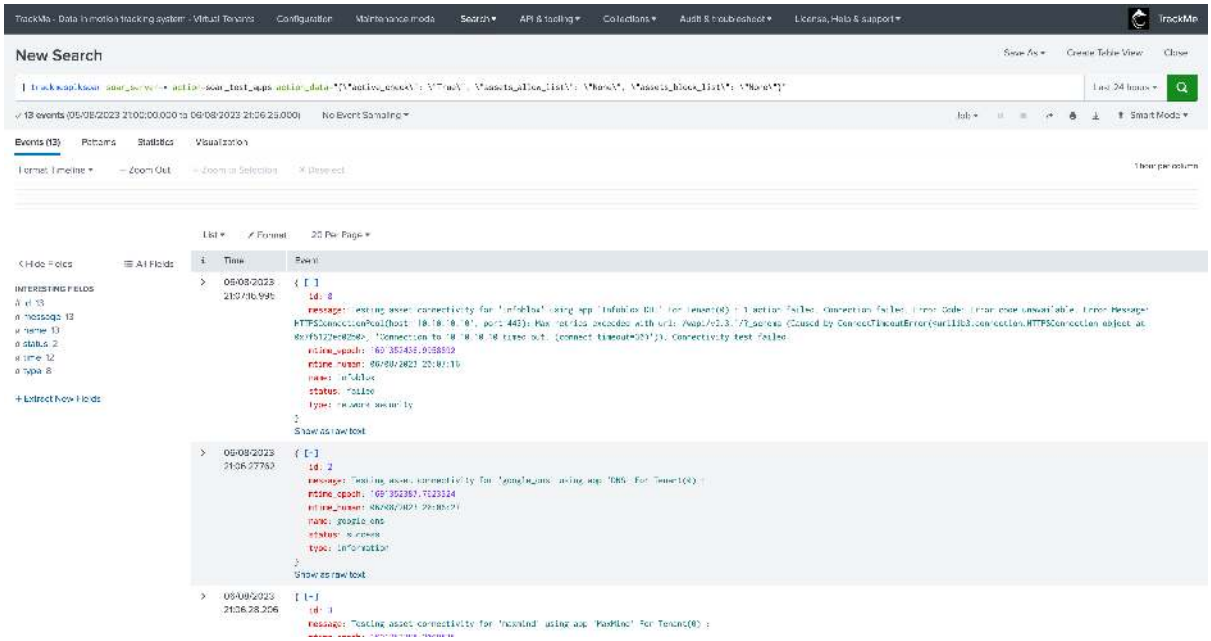
TrackMe handles this task by performing a multi-steps bi-directional integration with the SOAR REST API, summarised in the following diagram:



Advanced pre-built bi-directional interactions with Assets active check

The SOAR Assets active check in TrackMe relies on some multi-steps and bi-directional actions performed by TrackMe, the default usage of the command is the following:

```
| trackmesplksoar soar_server=* action=soar_test_apps action_data="{\"active_check\
↪": \"True\", \"assets_allow_list\": \"None\", \"assets_block_list\": \"None\"}"
```



In summary, the command does the following:

- Retrieve the list of assets eligible for a connection test
- For each asset, perform a POST call against the Asset API endpoint which requests an immediate application health check
- Record the response and store for later usage in the process
- Retrieve the Asset app_status endpoint result, parse and render the final results

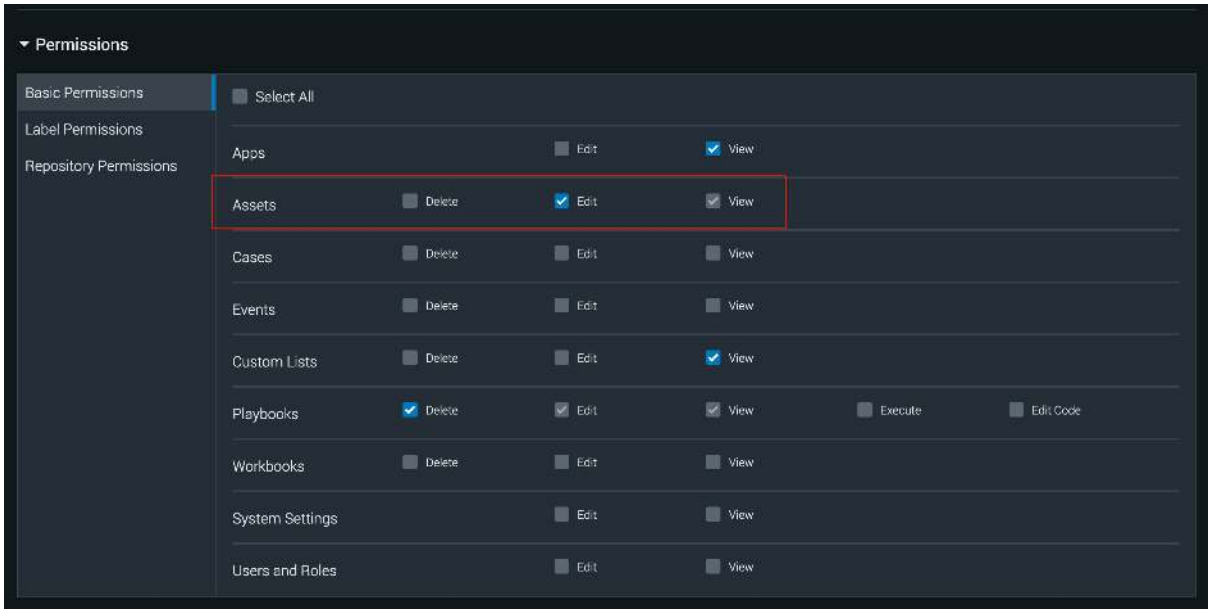
The command can accept several options:

Active checks

The default behaviour with `active_check: True` means TrackMe will actively perform the check bu running the POST call as described above:

If disabled with `active_check: False`, TrackMe does not perform the POST call and instead relies on SOAR doing it once per day, therefore it can take up to 24 hours before a failing application could be detected.

Note that `active_check: True` requires the SOAR automation user to have the edit Assets permissions:



Allow list and Block list:

You can mix the usage of asset_allow_list and asset_block_list to restrict and/or avoid some assets from being taken into account, for instance to avoid including the SMTP asset "internal_smtp":

```
| trackmesplksoar soar_server=* action=soar_test_apps action_data="{\"active_check\
↪": \"True\", \"assets_allow_list\": \"None\", \"assets_block_list\": \
↪"internal_smtp\"}"
```

5. SOAR Automation Brokers Monitoring, High Availability and Failover with TrackMe

Hint

Automation Broker Pools: Latest updates since TrackMe 2.1.8

- This feature was first introduced in TrackMe Version 2.0.50.
- Since TrackMe 2.1.8, these capabilities were enhanced so we can support independent pools of brokers that can be defined and managed in TrackMe.
- This allows you to have full flexibility, and configure multiple groups of brokers where Assets can be moved transparently depending on the brokers availability.

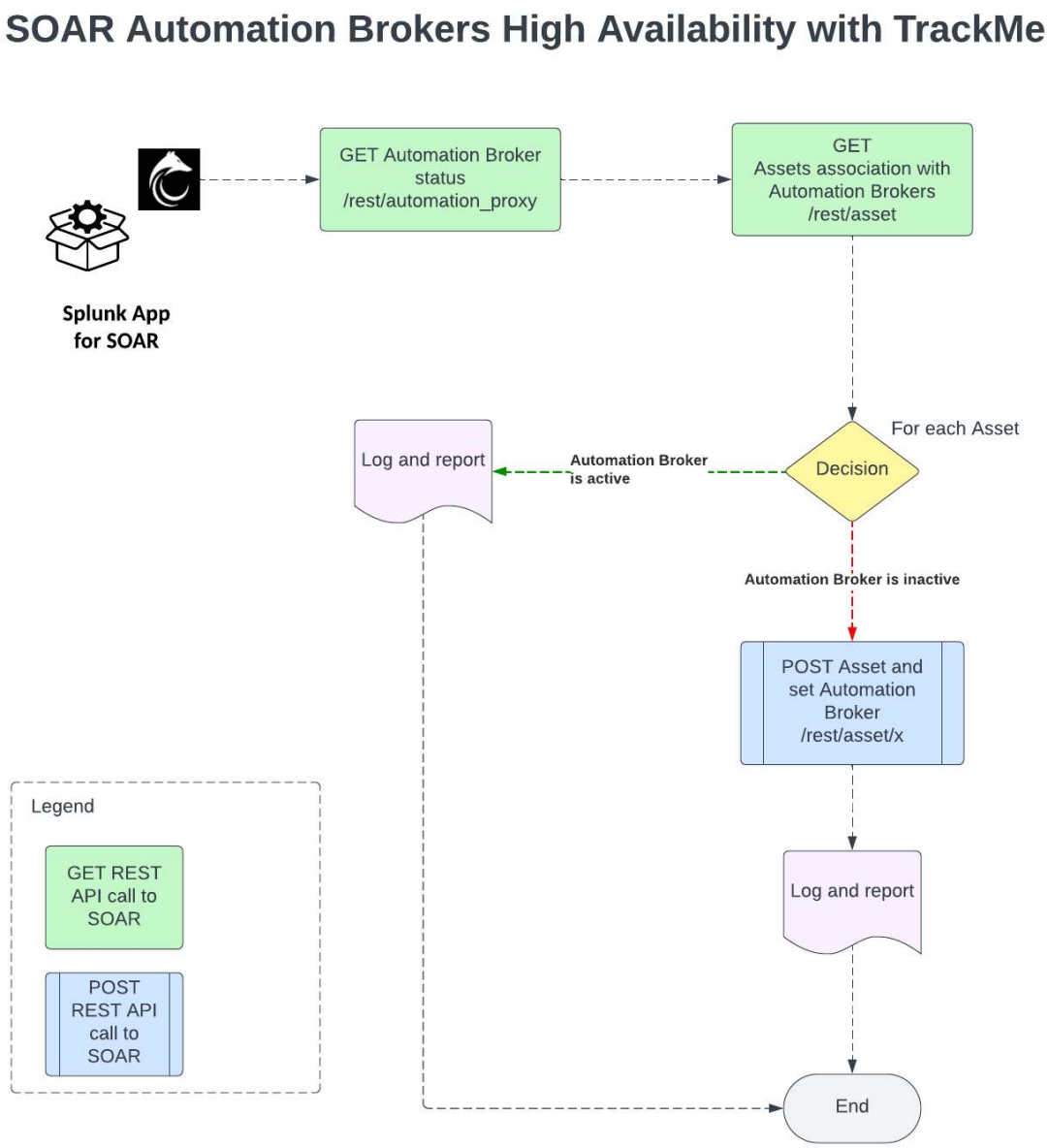
The builtin use case `splk_soar_manage_automation_brokers` allows managing SOAR Automation Brokers **High Availability** using TrackMe, based on the following workflow:

- Retrieve the list of Automation Brokers and their status from the SOAR API
- Retrieve the list of SOAR Assets and the association with Automation Brokers
- For each Automation Broker, if the broker status is inactive, perform an update of associated Assets to the next available and online Automation Broker

Automation broker status: the status is provided by the SOAR API and is based on the health check performed automatically by the SOAR Automation Broker.

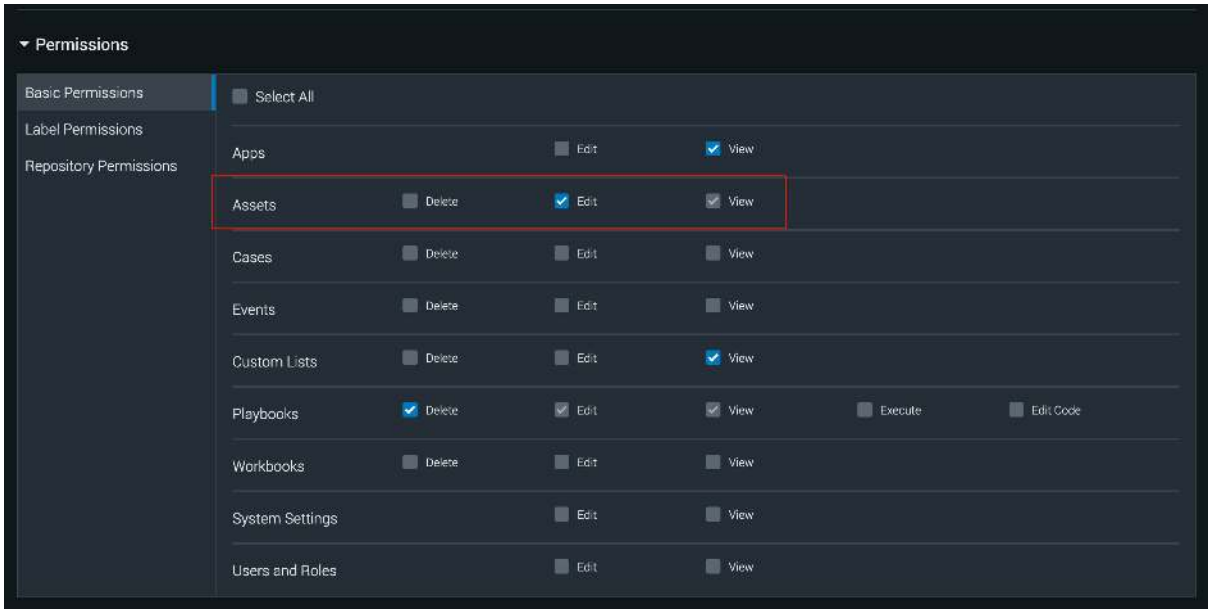
Additional options can be used to control and restrict the High Availability features, consult the examples below.

High level workflow diagram:



5.1 Requirements

The SOAR Automation User requires Assets management permissions:



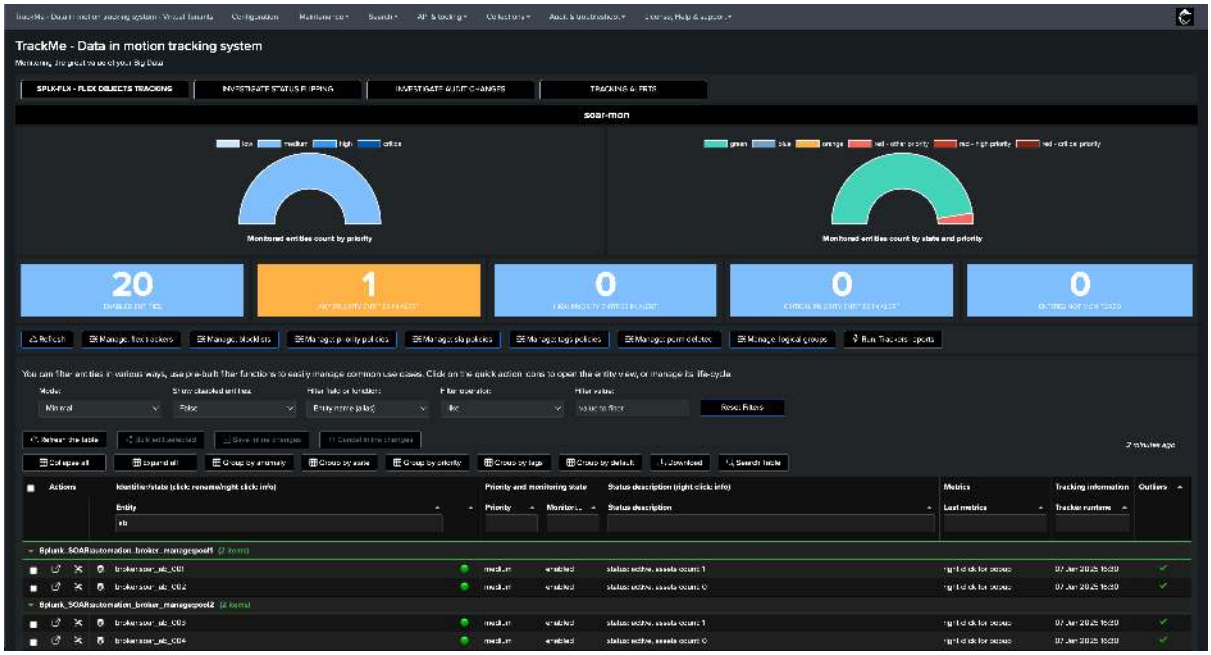
5.2 SOAR Automation Broker High Availability - failure detection and Assets updates

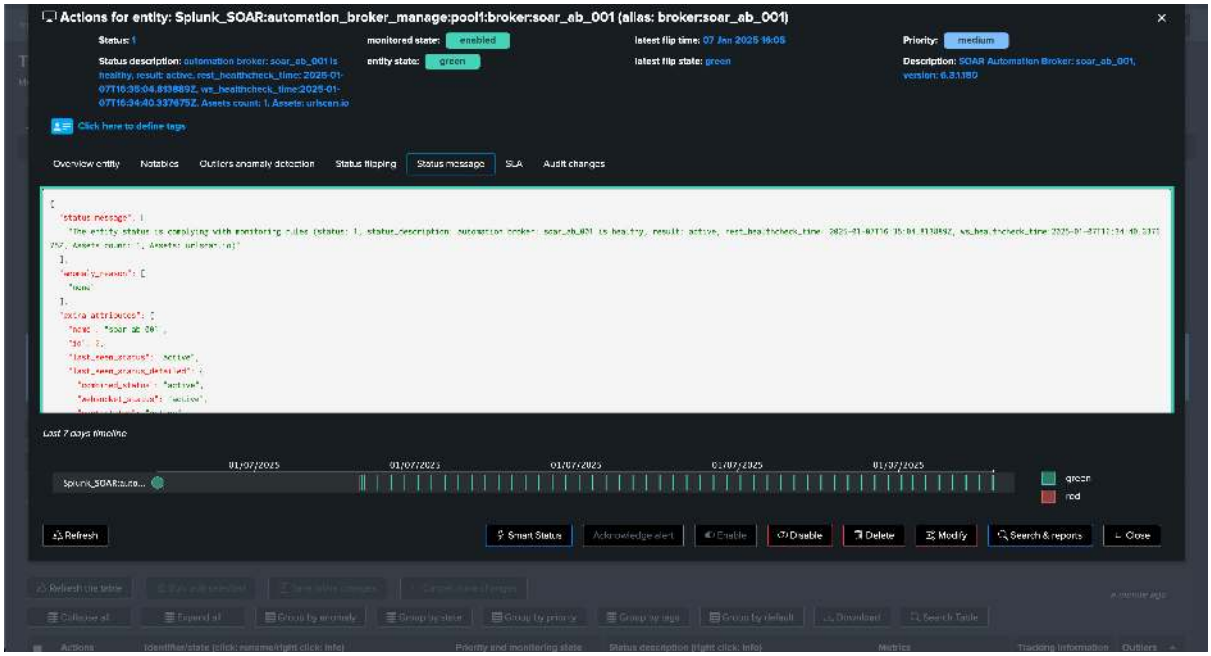
The tracking and management of SOAR Automation Brokers is achieved by running the following command through the builtin use case splk_soar_manage_automation_brokers:

```
| trackmesplksoar soar_server=* action=soar_automation_broker_manage
```

The command calls the TrackMe REST API endpoint `/services/trackme/v2/splk_soar/admin/soar_automation_broker_manage`, the following options are available:

argument	description
soar_se:	The SOAR server account as defined in the Splunk App for SOAR, if unspecified or set to *, the first server in the Splunk application for SOAR configuration will be used
mode	Optional, the run mode, valid options are readonly | live. In read only mode, the Asset is not updated and only the message of the action to be performed is registered. In Live mode, assets are updated if the associated broker is not healthy, and if at least one active broker is available in the pool, defaults to live
automation_br(	The comma separated list of Automation Brokers for the pool, if unspecified TrackMe will consider all registered SOAR Automation Brokers. When specified, the Flex Object tracker instance is tight to a certain list of Automation Brokers, and only these will be considered for the High Availability management. When creating multiple Trackers to manage different pools, you can update the group definition to clearly separate the resulting TrackMe entities, for easier management in the TrackMe UI.





Configuration example: Two pools of Automation Brokers

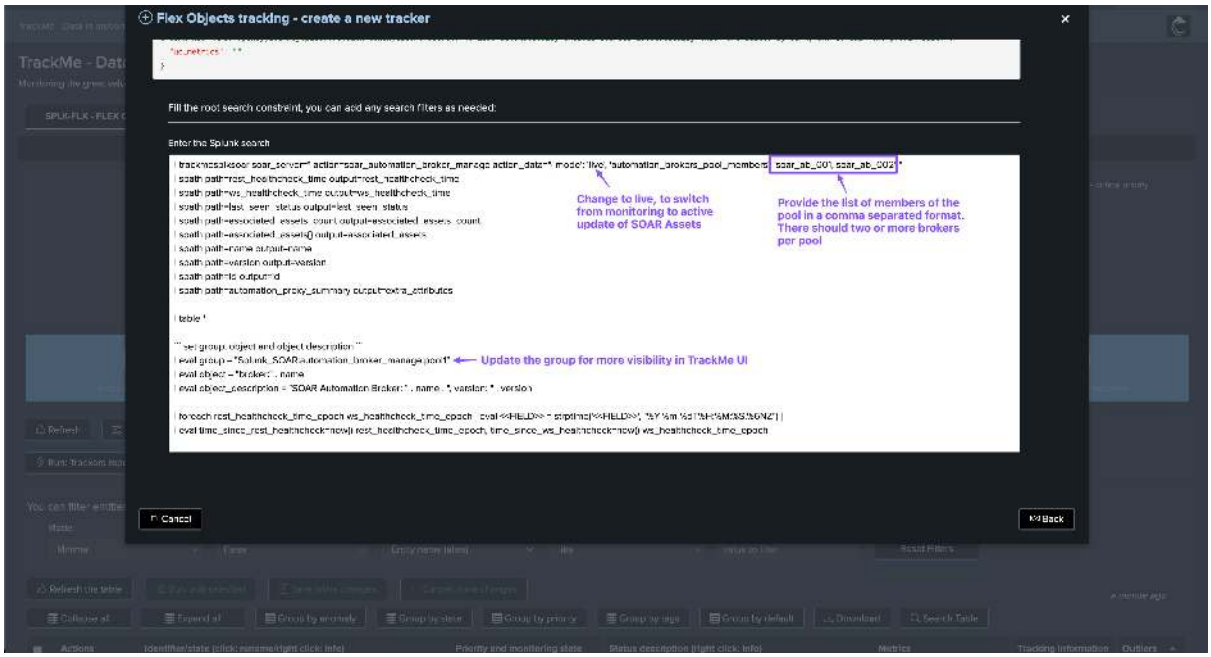
Consider the following Architecture scenario:

- Our SOAR environment has 4 Automation Brokers in total.
- Due to Network or any other technical constraints, certain assets can only use specific Automation Brokers.
- Therefore, we have 2 groups of Automation Brokers, in each of these groups, Assets cab be configured to use either broker from that pool.

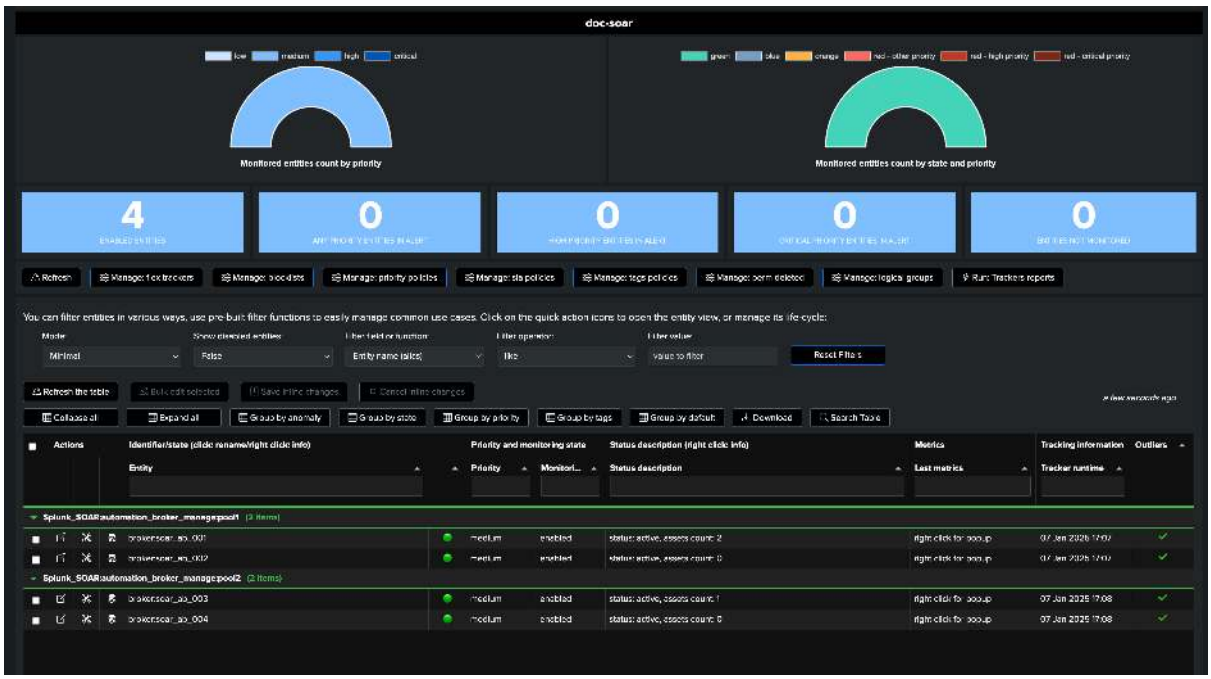
Automation Broker	Pool
soar_ab_001	Pool1
soar_ab_002	Pool1
soar_ab_003	Pool2
soar_ab_004	Pool2

Create two Flex Object trackers to manage each pool:

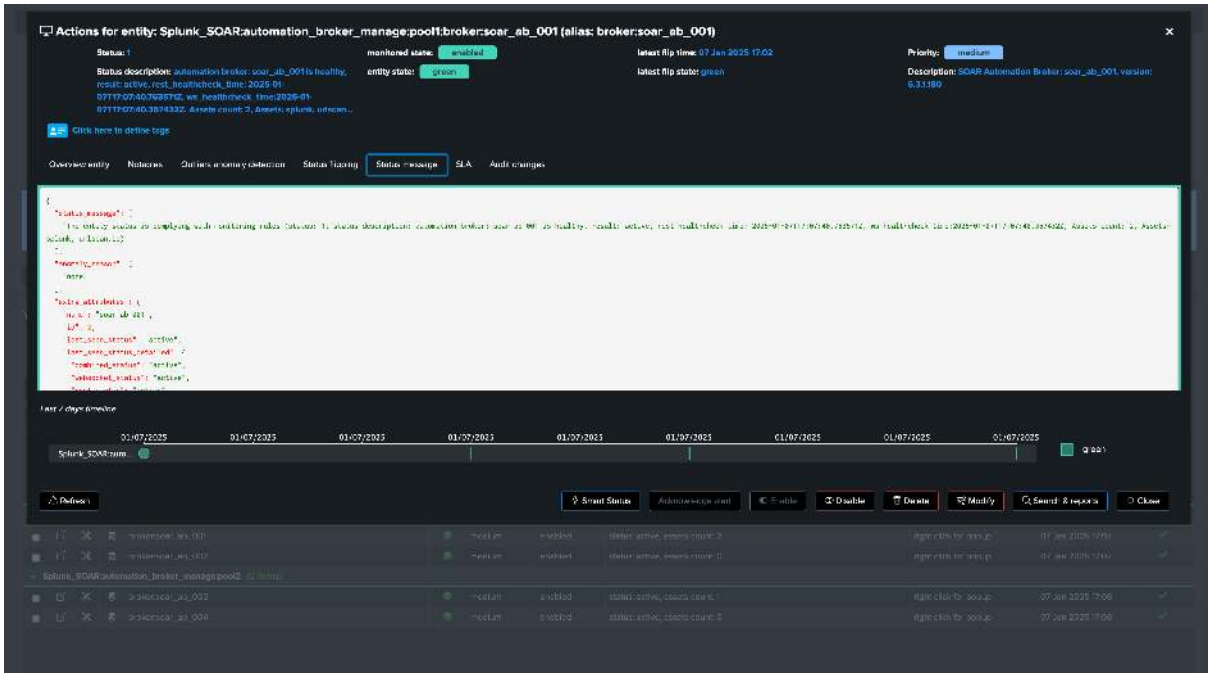
Create the first Tracker as following, and repeat for the second pool:

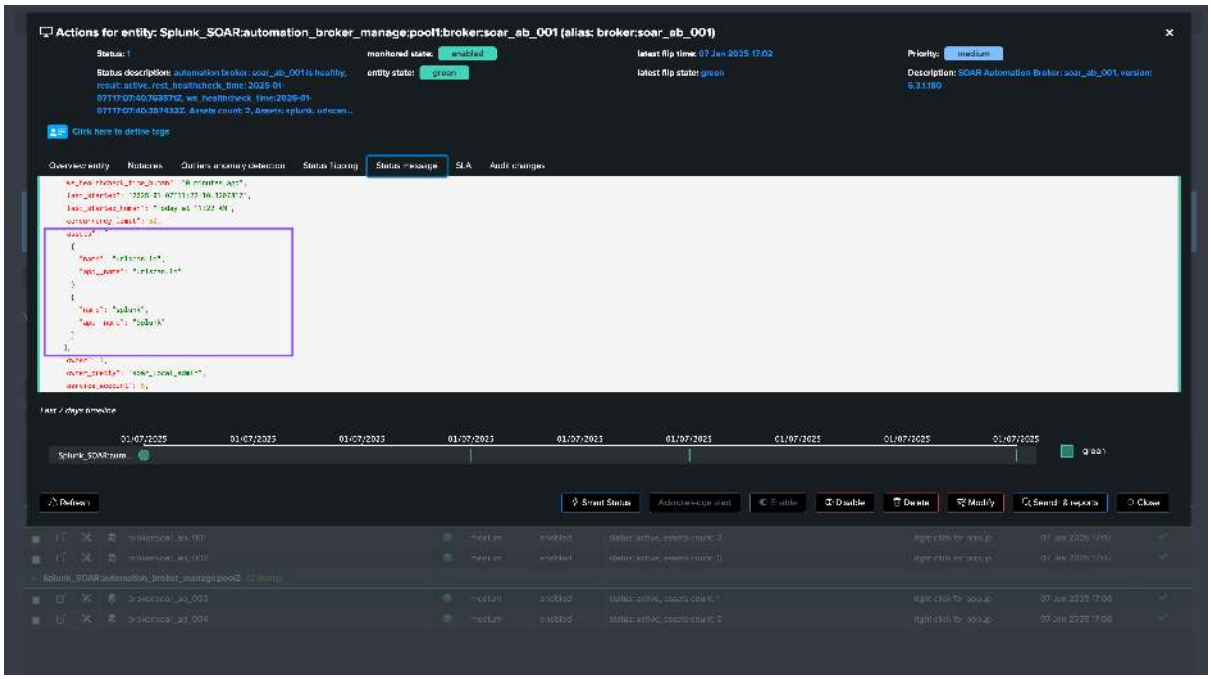


Once the trackers have been executed at least once, our setup is ready and we have 4 entities representing the 2 pools of brokers:



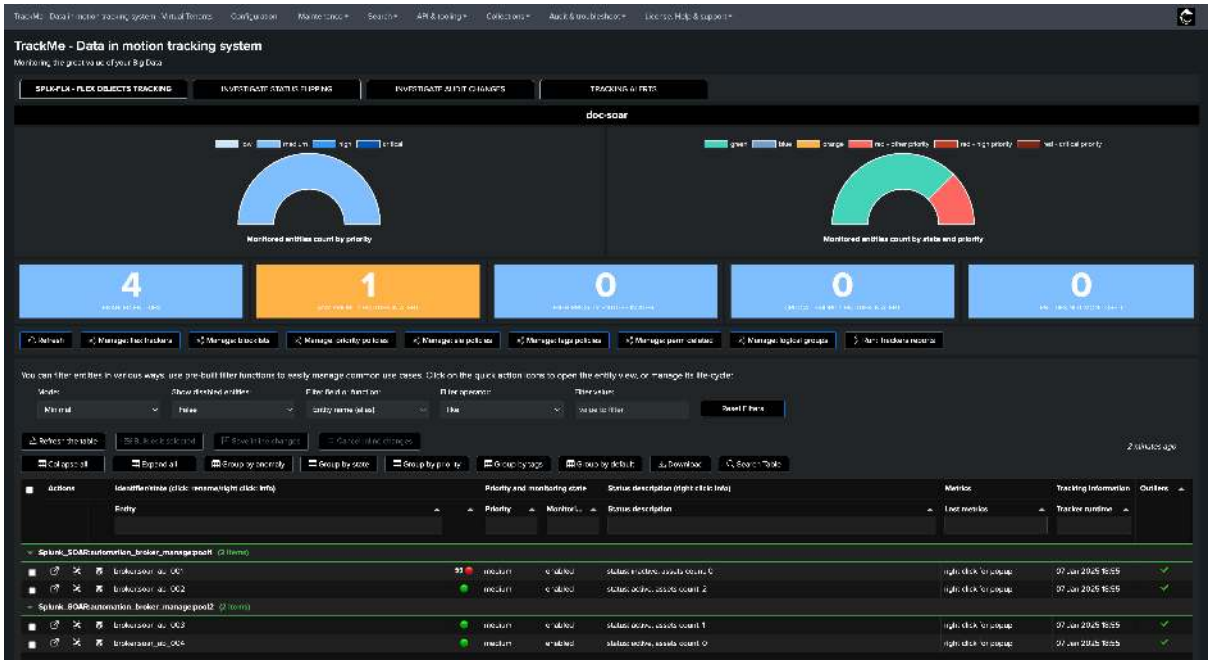
The status message shows the extended attributes, which contains all Metadata from the brokers, and various information such as the list of assets currently linked to this asset:

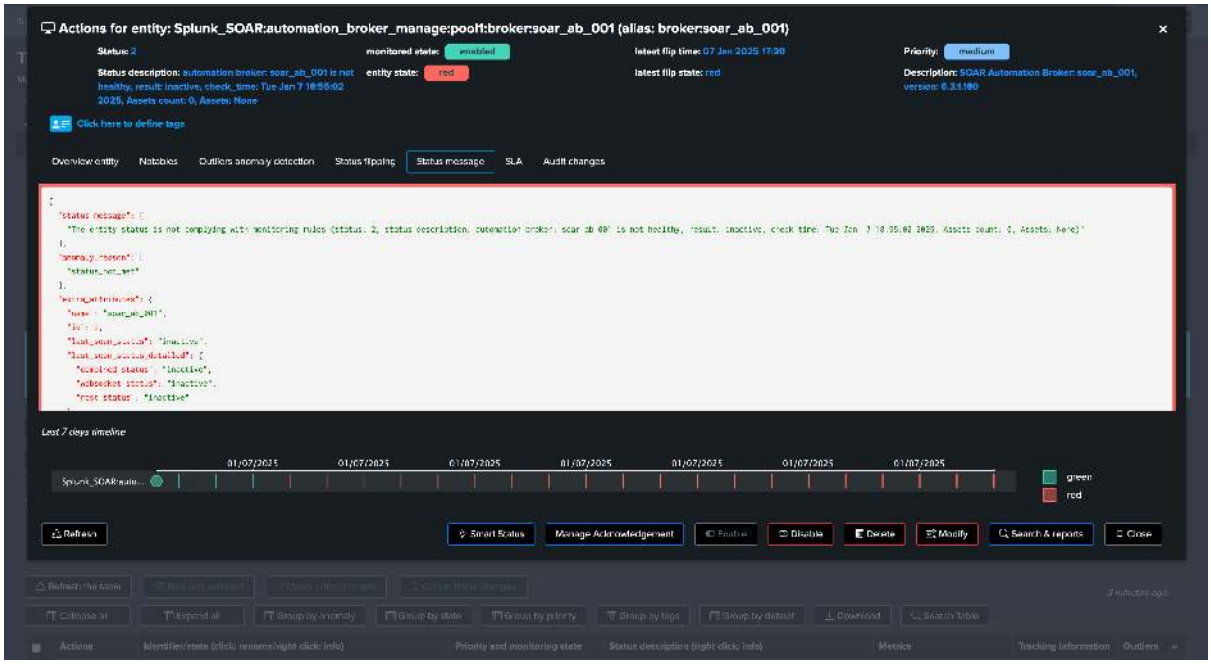




What happens when an Automation Broker is detected as inactive?

If an issue affects an Automation Broker, TrackMe turns the entity in red state, and will automatically update Assets so these are associated with any randomly chosen active broker from the same pool:



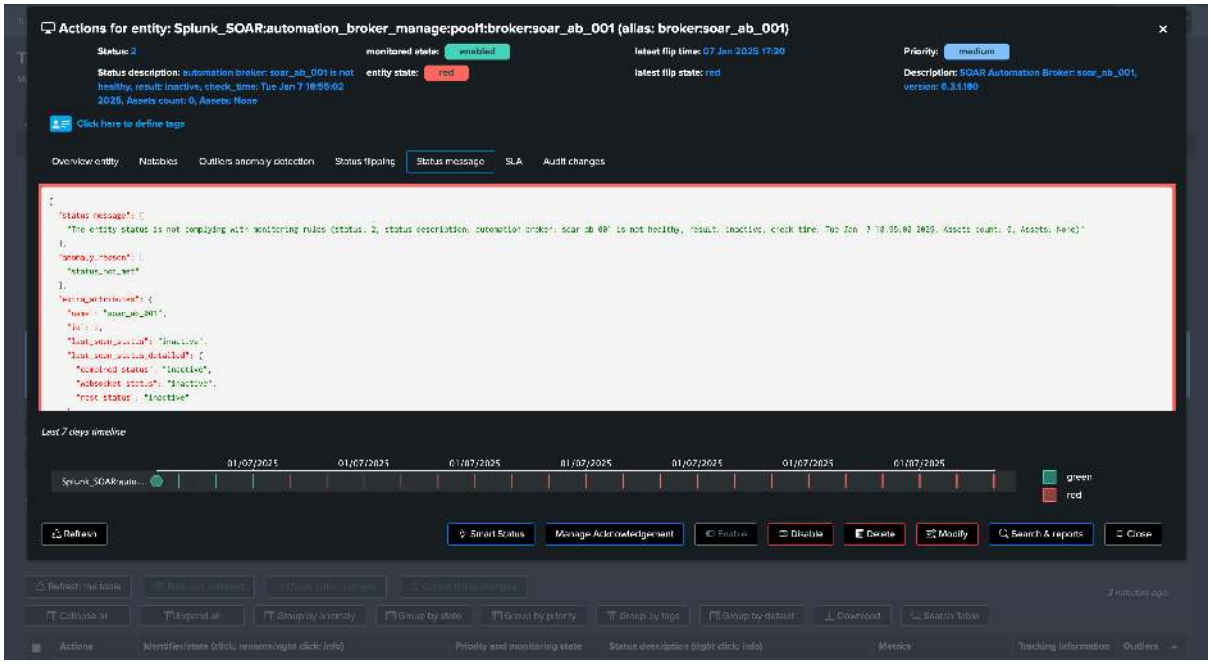


Logs can be found at:

```
index=_internal sourcetype=trackme:rest_api post_soar_automation_broker_manage
```

When updates of SOAR Assets are performed, the following search can be used to review the logs and executed actions:

```
index=_internal sourcetype=trackme:rest_api post_soar_automation_broker_manage␣
↪updated
```



A typical trace for an Asset update will look the following example:

A first warning message shows that the Automation Broker associated with the Asset is currently offline, followed by the update result message from the action performed by TrackMe.

```
2025-01-07 17:20:06,002 WARNING trackme_rest_handler_splk_soar_admin.py␣
↪post_soar_automation_broker_manage 1081 asset=urlscan.io, id=3 is associated␣
↪with automation_broker=soar_ab_001, id=2, status=inactive, asset will be updated␣
↪now!
2025-01-07 17:20:06,104 INFO trackme_rest_handler_splk_soar_admin.py␣
↪post_soar_automation_broker_manage 1138 asset=urlscan.io, id=3, asset automation␣
↪broker configuration was successfully updated from␣
↪automation_broker=soar_ab_001, id=2, status=inactive to␣
↪automation_broker=soar_ab_002, id=3, status=active, response="{'success': True,
↪'id': 3}"
```

What happens when the Automation Broker is back online?

When the Automation Broker is back online, the entity will turn back to green. However, note that the Asset definition will not be updated back to the original Automation Broker, this is a design choice to avoid a ping-pong effect, and to ensure that the Asset is always associated with an active Automation Broker.

6. Interacting with the SOAR API with GET and POST calls in pure SPL

The following sections show the root searches and technical details of each TrackMe's SOAR use cases.

Refer to the SOAR API reference documentation as needed:

- https://docs.splunk.com/Documentation/SOAR/current/PlatformAPI/Using

6.1 Running GET calls to SOAR API

trackmesplksoar:

TrackMe comes with a custom command (generating custom command) called `trackmesplksoar`, this SPL command does the interface with the Splunk App for SOAR and the SOAR API itself, usage:

```
| trackmesplksoar soar_server=<soar_server> action=<action> action_data=<json␣
↪action data>
```

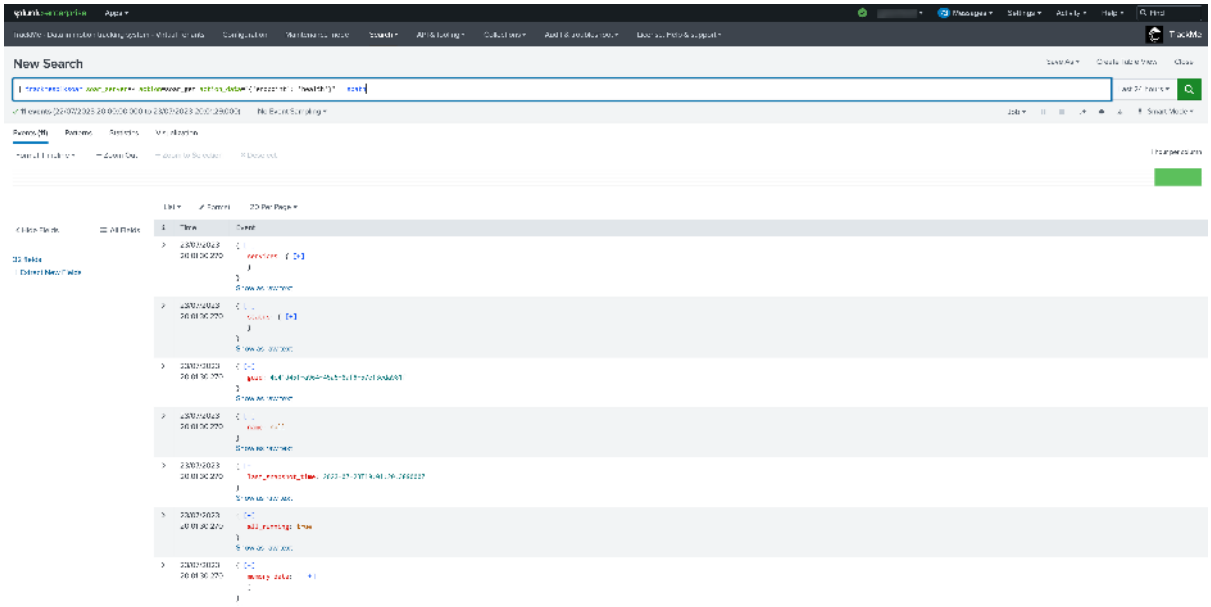
The command handles different options:

Syntax:

- soar_server: the name of the SOAR server as configured in the Splunk App for SOAR
- action: an action in the following support list: `soar_get|soar_post|soar_test_apps|soar_health_status|soar_hea`
- action_data: a JSON formatted object, either used by specific actions or used to perform a POST query to a SOAR endpoint

Example: the following command allows retrieving the health information from the SOAR API, it targets the endpoint rest/health_status:

```
| trackmesplksoar soar_server=* action=soar_get action_data="{'endpoint': 'health'}
↪" | spath
```



This action allows calling any SOAR API endpoint performing a GET call only.

Another example, to retrieve the configuration of all SOAR Assets, you would:

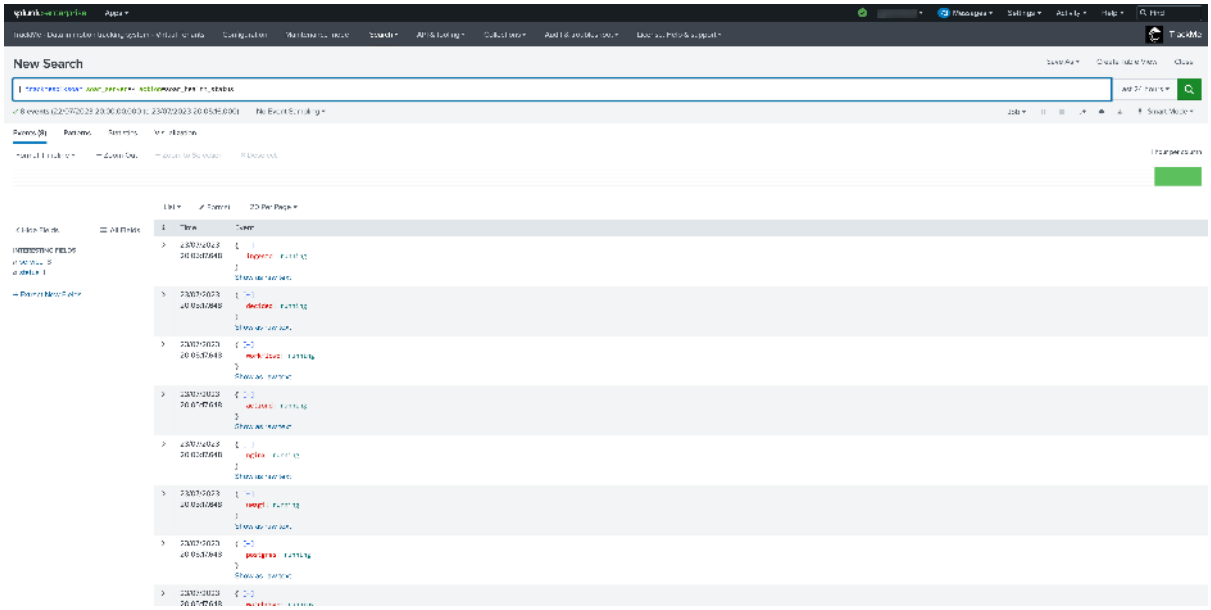
```
| trackmesplksoar soar_server=* action=soar_get action_data="{'endpoint': 'asset'}
↪" | spath
```

6.2 Pre-built actions and advanced parsing

In some cases, such as the one above, it may not be very straightforward to parse and use the API endpoints results, for instance when there are nested JSON structures as part of the response.

For instance, for the ease of the integration, the custom command provides builtin use cases which will parse the JSON results as needed and render the results properly, for instance the following relies on the health endpoint and extracts the status of SOAR services:

```
| trackmesplksoar soar_server=* action="soar_health_status"
```



6.3 Performing a POST call to a SOAR API endpoint via TrackMe

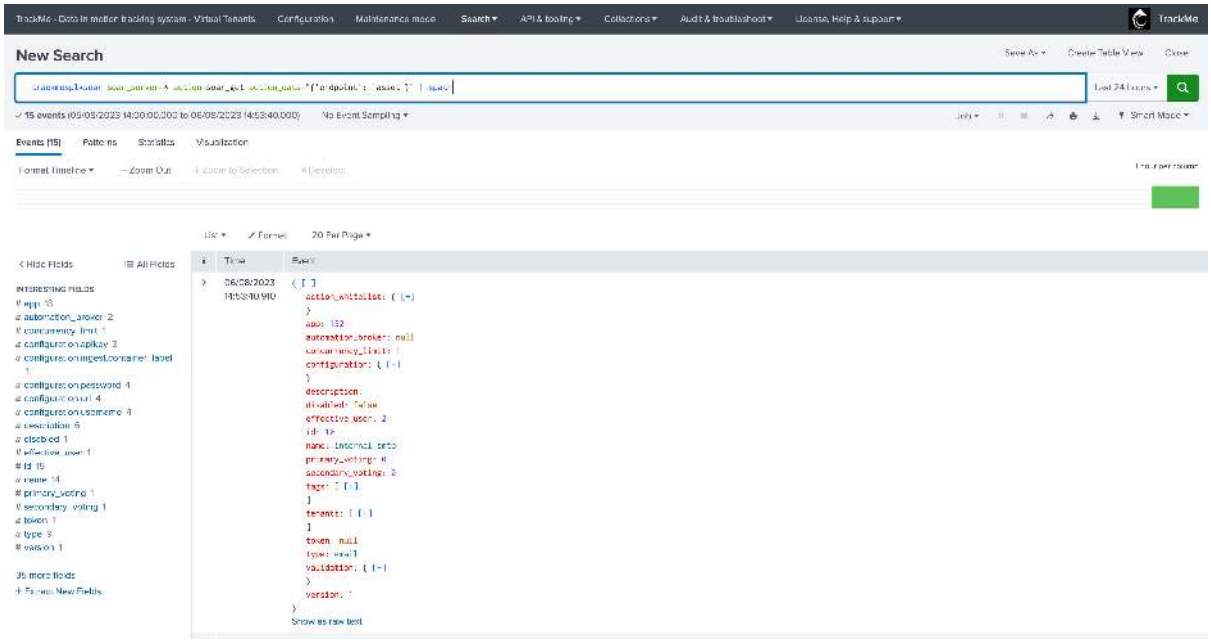
In fact, you can perform a POST to a SOAR API endpoint using TrackMe's integration, which the Splunk App for SOAR does not provide.

For instance, to run a POST against an Asset and request an immediate health check, you would first retrieve the list of assets to identify the ID of the Asset:

See:

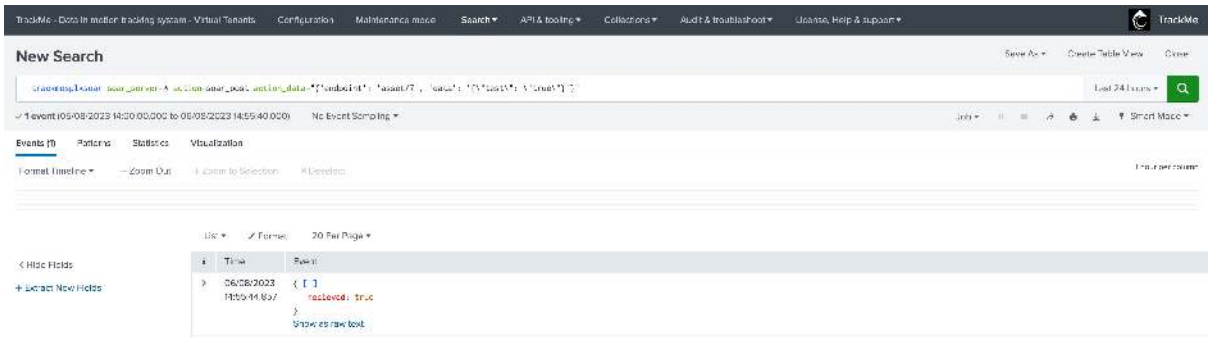
- https://docs.splunk.com/Documentation/SOAR/current/PlatformAPI/RESTInfo
- https://docs.splunk.com/Documentation/SOAR/current/PlatformAPI/RESTAssets

```
| trackmesplksoar soar_server=* action=soar_get action_data="{'endpoint': 'asset'}
↪" | spath
```



Then, you would:

```
| trackmesplksoar soar_server=* action=soar_post action_data="{'endpoint': 'asset/7
↪', 'data': '{\"test\": \"true\"}'}"
```



Note: "received: true" is the actual response from SOAR, this is not a typo from TrackMe but a typo in the SOAR API currently for this endpoint!

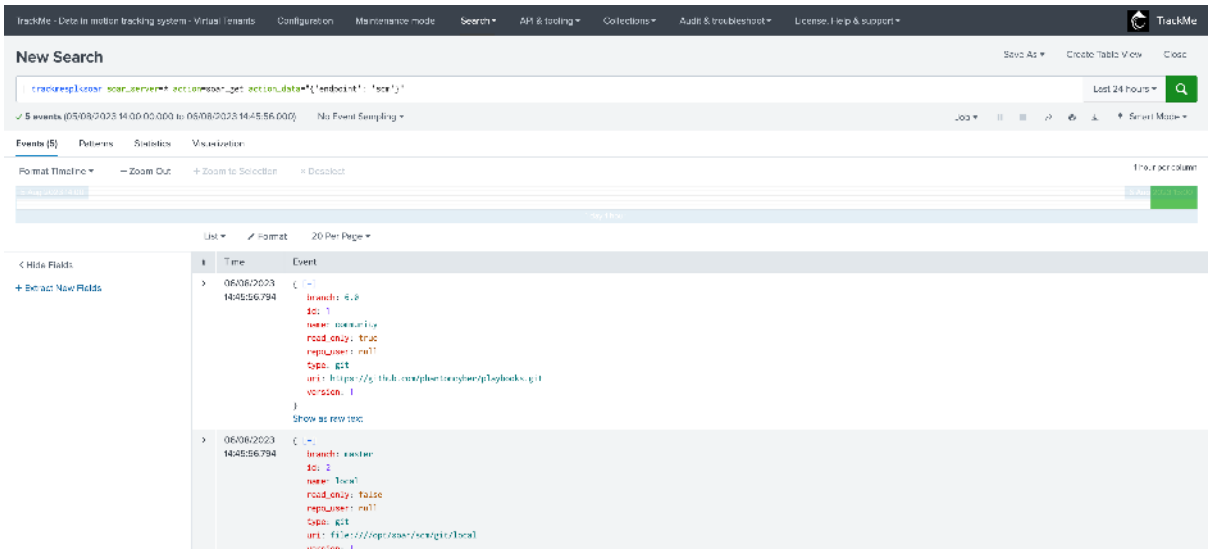
Another example, let's request the sync refresh of a Git Repository via the SCM endpoint:

See:

- https://docs.splunk.com/Documentation/SOAR/current/PlatformAPI/RESTScm

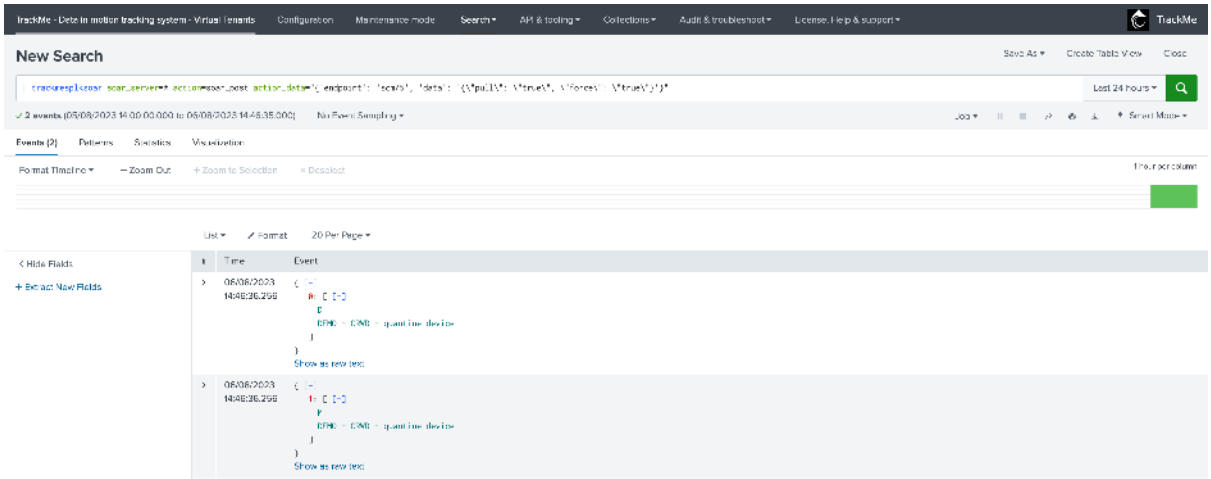
First, we run a GET call to get the ID of the SCM Git repository configuration:

```
| trackmesplksoar soar_server=* action=soar_get action_data="{'endpoint': 'scm'}"
```

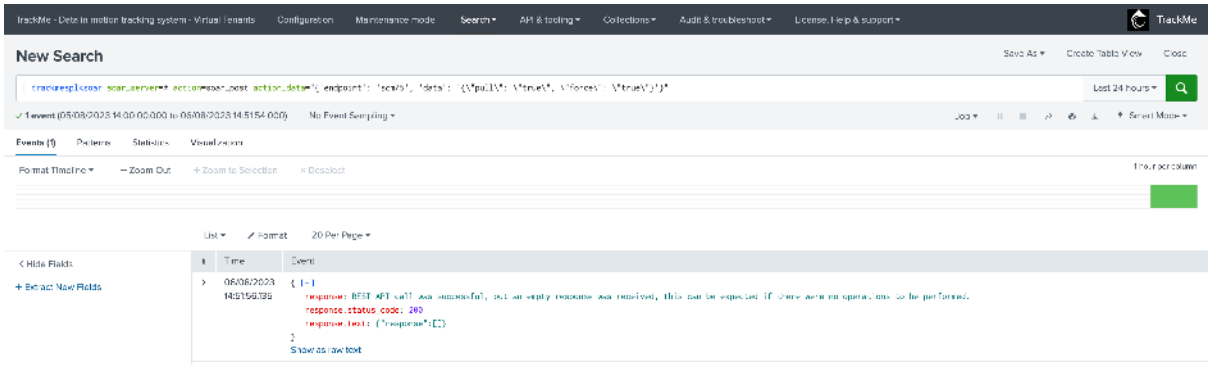


We run a POST call again to request the SCM update, we have a few changes that were actually merged to the branch:

```
| trackmesplksoar soar_server=* action=soar_post action_data="{'endpoint': 'scm/5',
↪ 'data': '{\"pull\": \"true\", \"force\": \"true\"}'}"
```



If we had no operations pending, we would have got the following message returned:



6.4 Performing live REST SOAR lookups in TrackMe using the command trackmesplksoarlookup

TrackMe provides a new command called `trackmesplksoarlookup` which allows to perform live REST lookups to the SOAR API, and to retrieve the metadata associated with the SOAR objects such as the playbooks, assets, applications and brokers associated with the detection.

Usage of the command:

```
[trackmesplksoarlookup-command]
syntax = | trackmesplksoarlookup soar_server=<soar_server>␣
↪endpoint_target=<endpoint_target> source_field=<source_field>␣
↪dest_field_name=<dest_field_name> dest_field_definition=<dest_field_definition>␣
↪definition_filter_fields=<A comma separated list of fields to retrieve from the␣
↪definition>
description = \
   This streaming command can be used to interact with the SOAR API in a lookup␣
↪way, so that from an id of an object, its definition can be retrieved easily in␣
↪native SPL \
   Syntax: \
   - soar_server: the name of the SOAR server as configured in the Splunk App for␣
↪SOAR, \
```

(continues on next page)

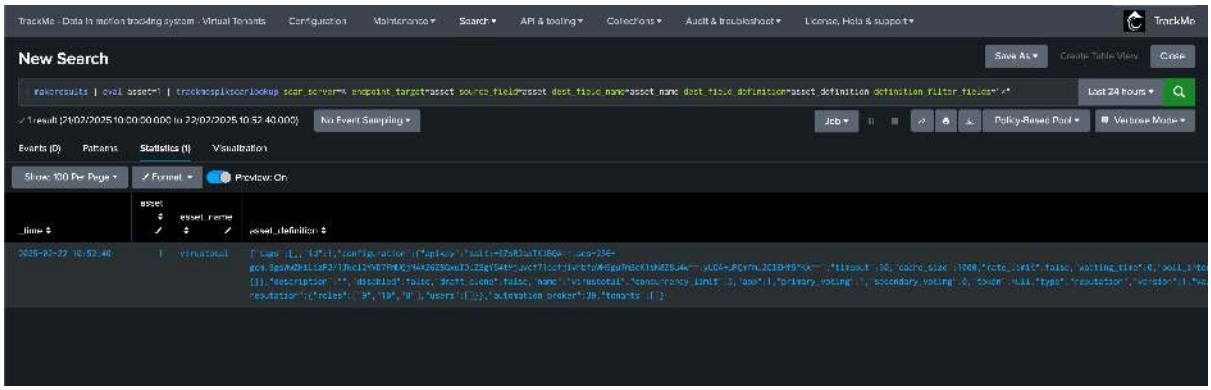
(continued from previous page)

```
   - endpoint_target: the endpoint target for the object to lookup\
   - source_field: the name of the field containing the object id, \
   - dest_field_name: the name of the field to store the logical name of the␣
↪corresponding object retrieved from this id (if any!), \
   - dest_field_definition: the name of the field to store the definition of the␣
↪corresponding object retrieved from this id (if any!) \
   - definition_filter_fields: a comma separated list of fields to retrieve from␣
↪the definition \
   | trackmesplksoarlookup soar_server=<soar_server>␣
↪endpoint_target=<endpoint_target> source_field=<source_field>␣
↪dest_field_name=<dest_field_name> dest_field_definition=<dest_field_definition>␣
↪definition_filter_fields=<A comma separated list of fields to retrieve from the␣
↪definition>
comment1 = \
   Lookup the definition of a SOAR object from its id in a streaming manner
example1 = \
   | makeresults | eval asset=1 | trackmesplksoarlookup soar_server=*␣
↪endpoint_target=asset source_field=asset dest_field_name=asset_name␣
↪dest_field_definition=asset_definition␣
↪definition_filter_fields="name,description"
shortdesc = Streaming command for TrackMe's Splunk SOAR integration
```

Example:

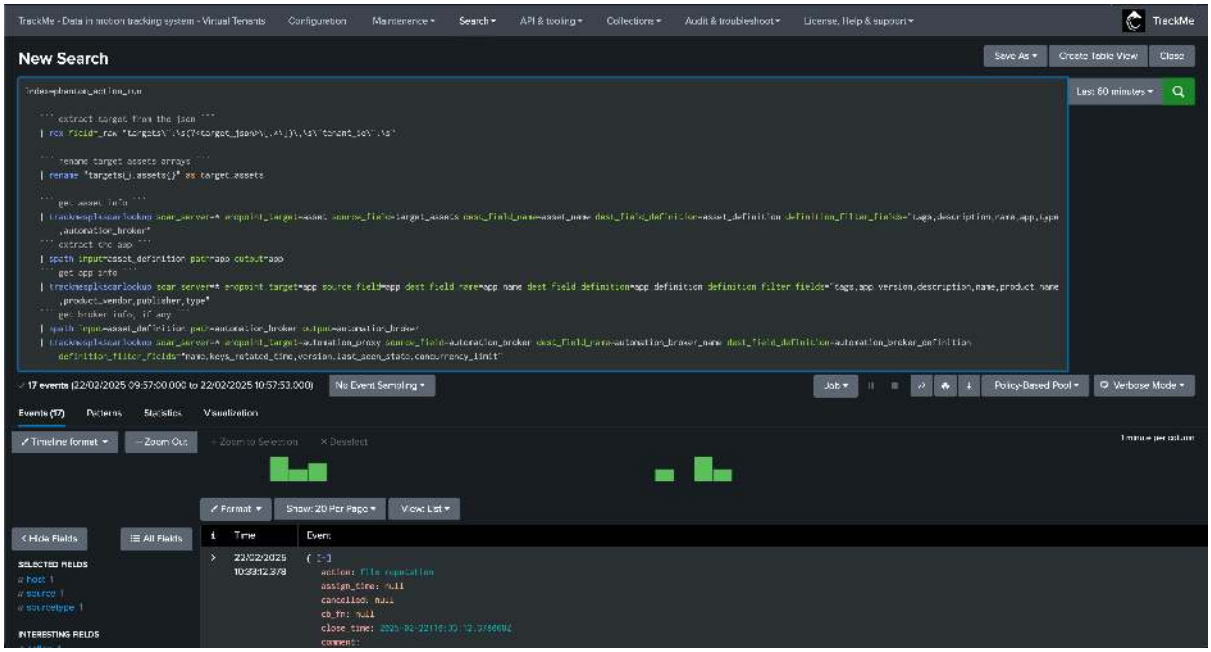
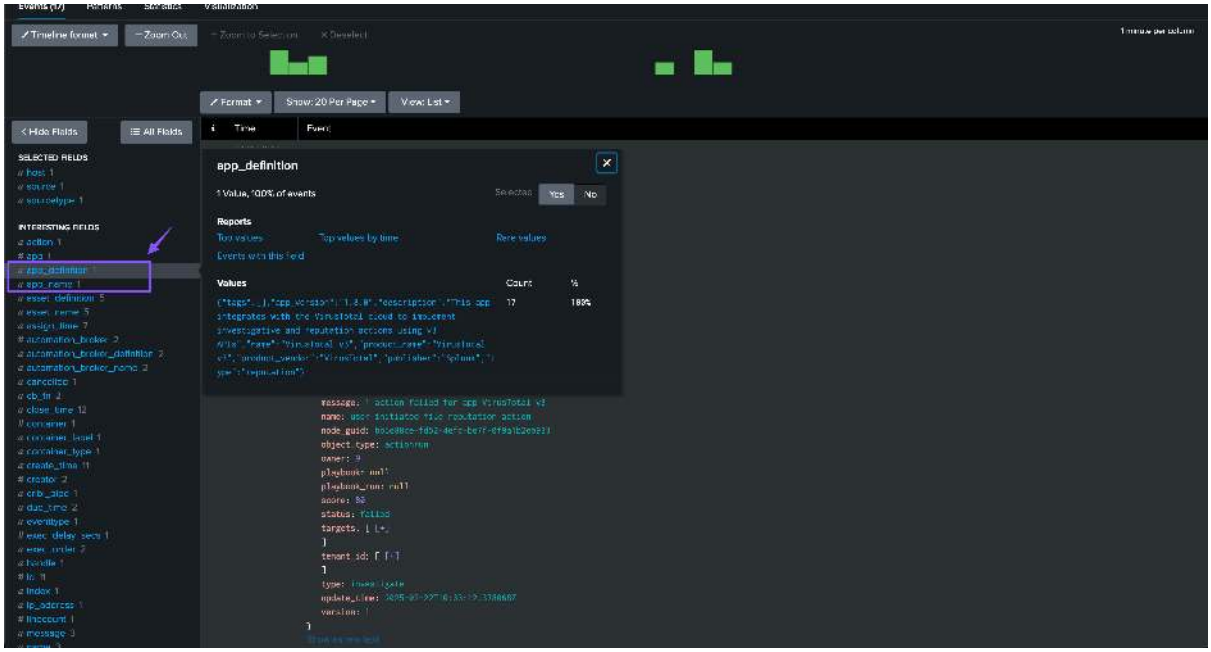
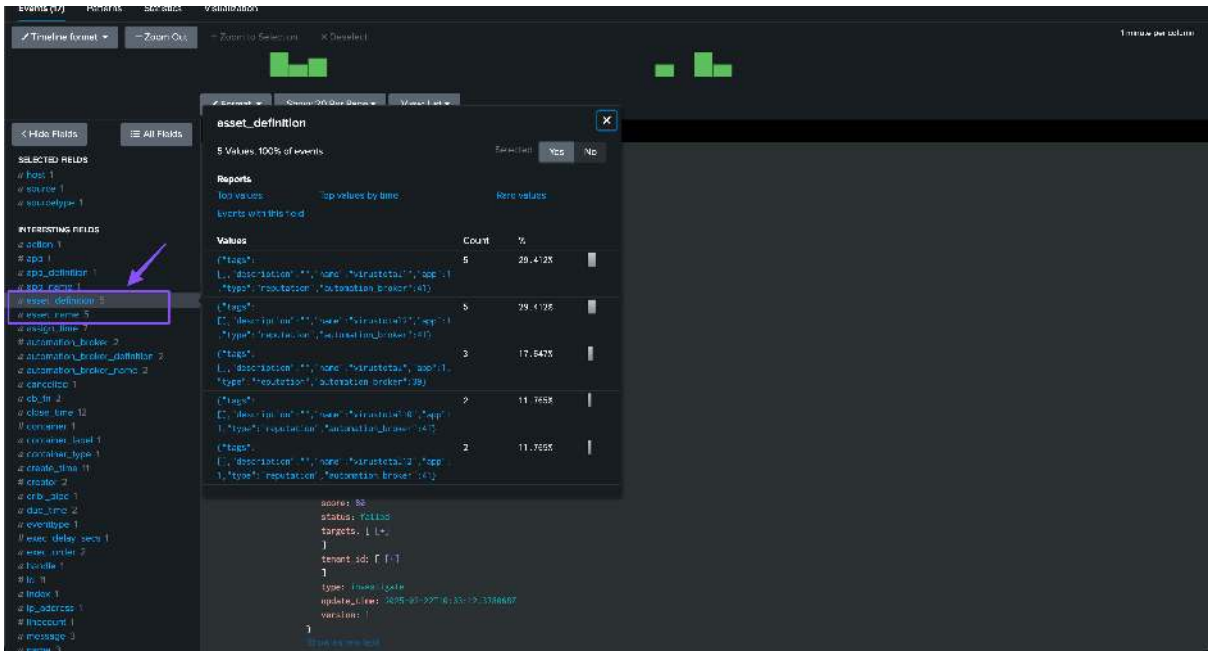
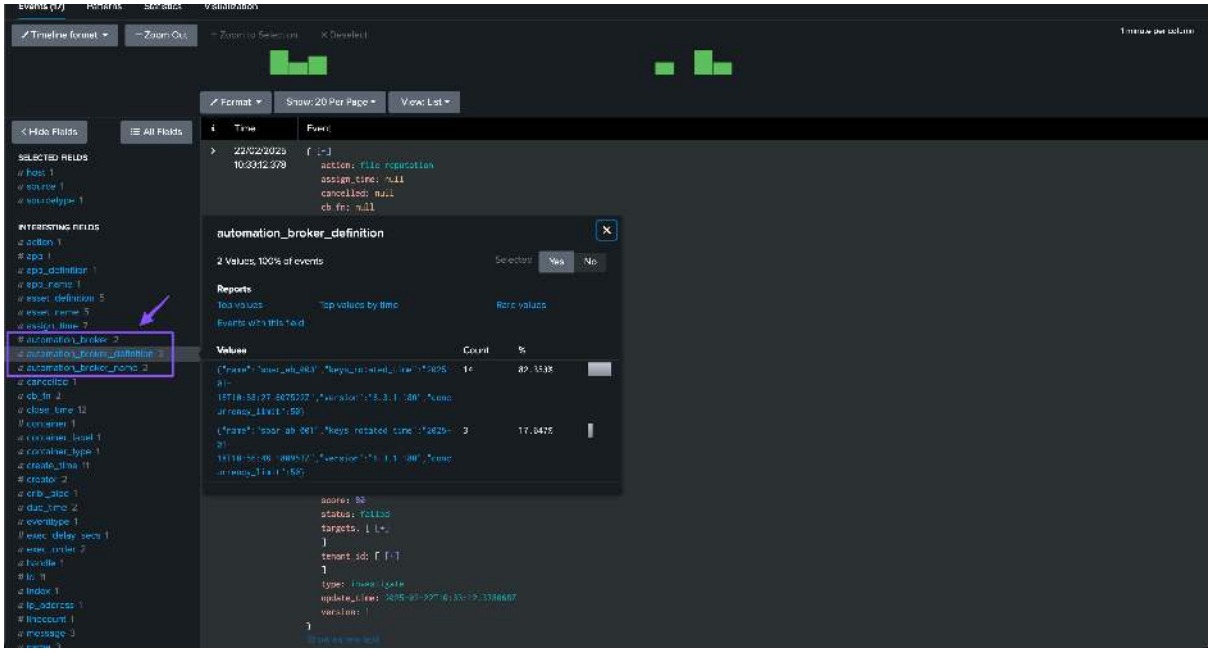
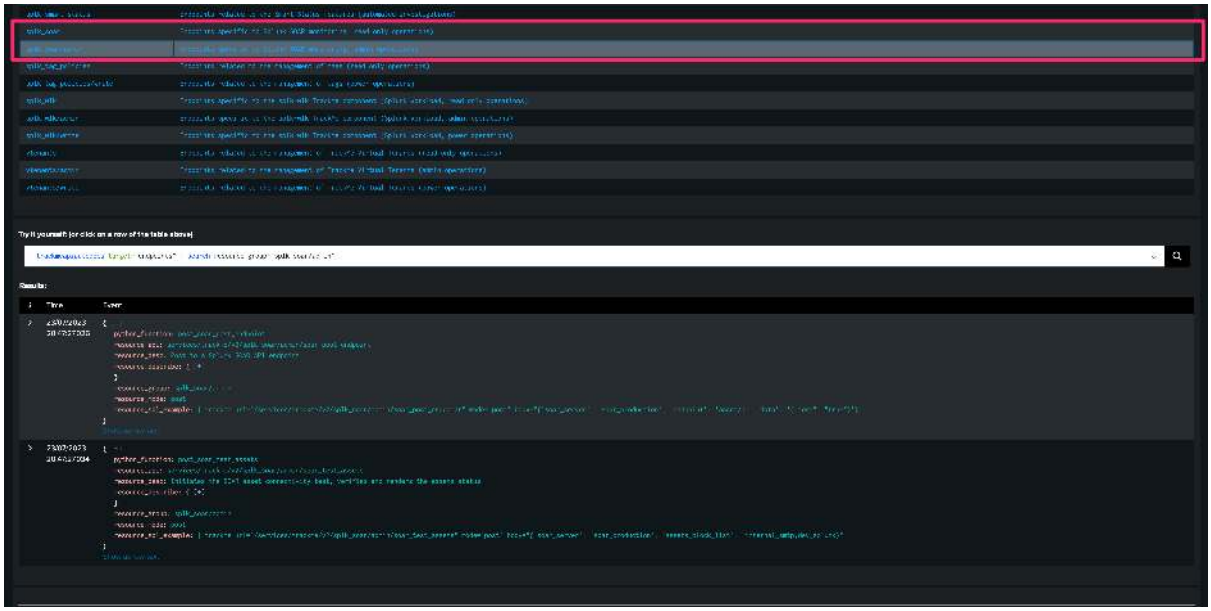
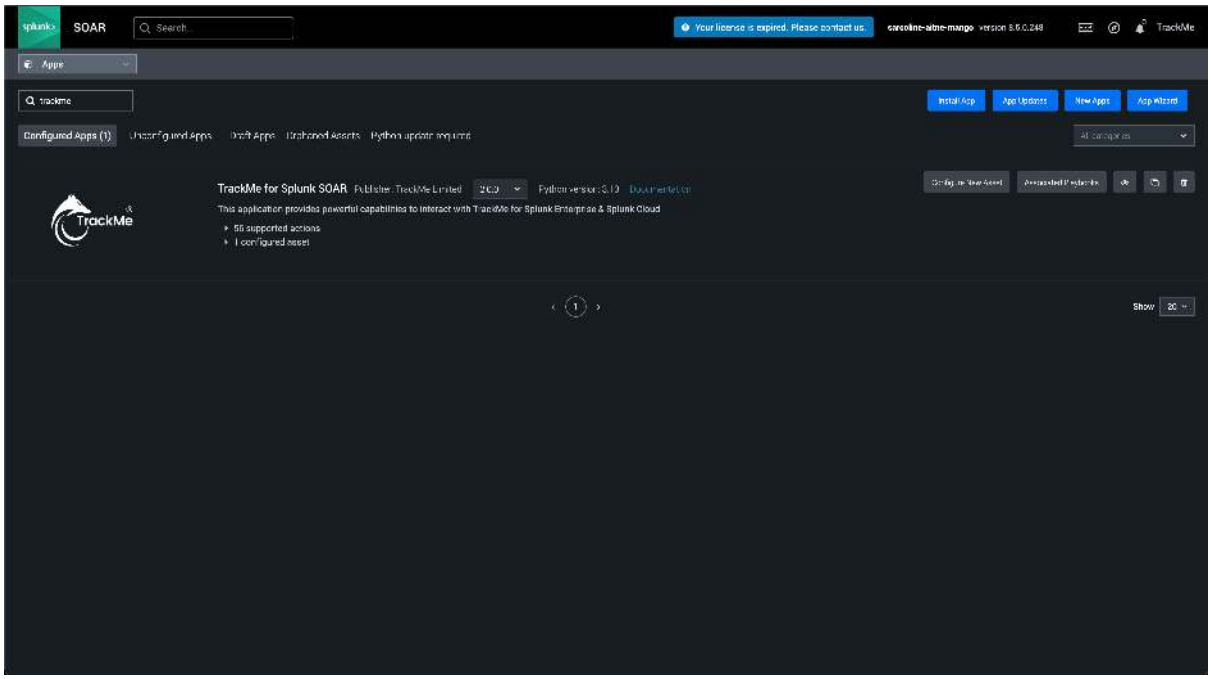
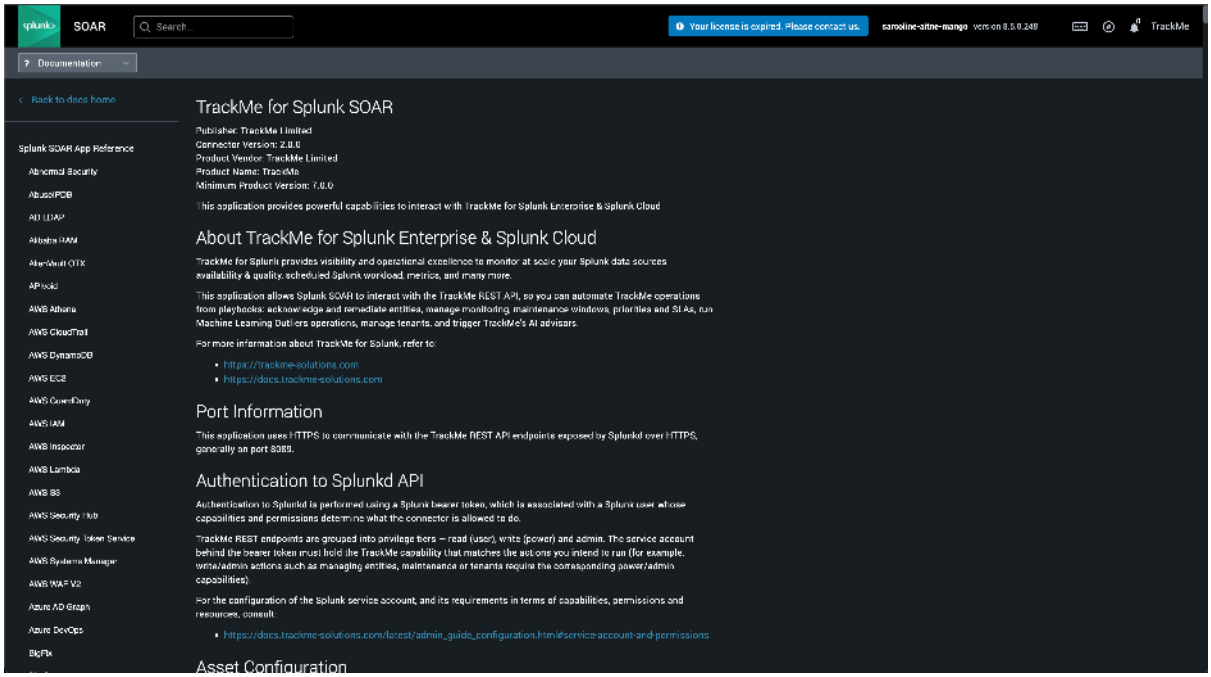
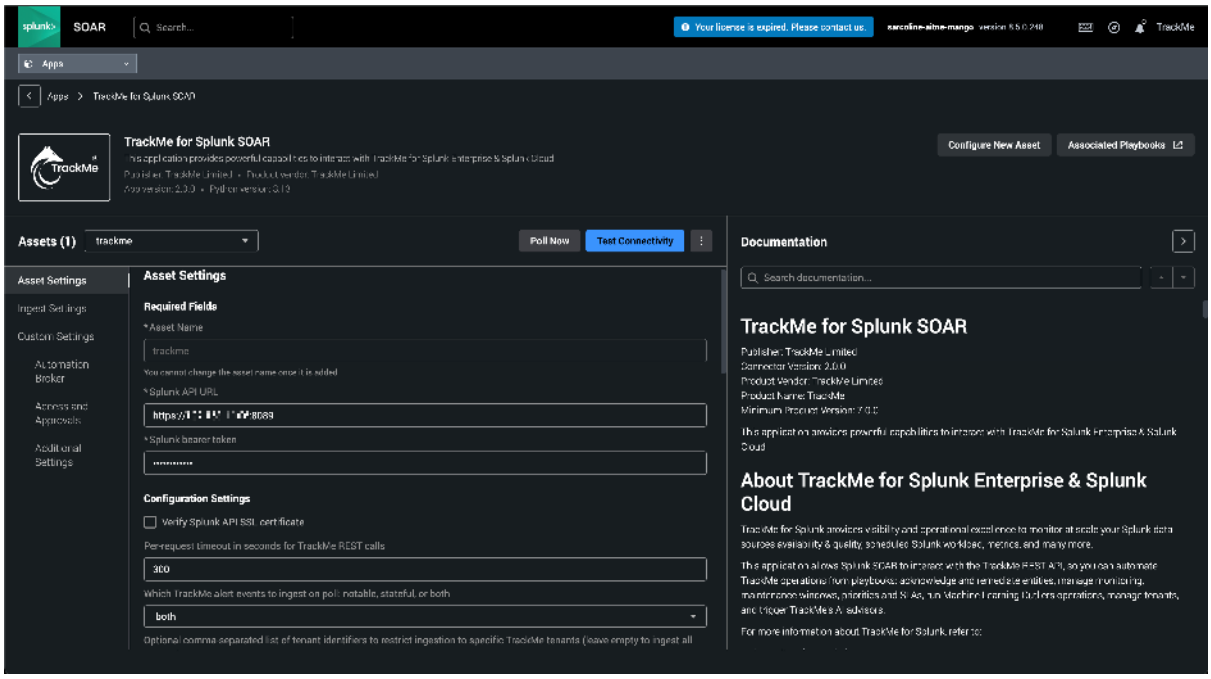
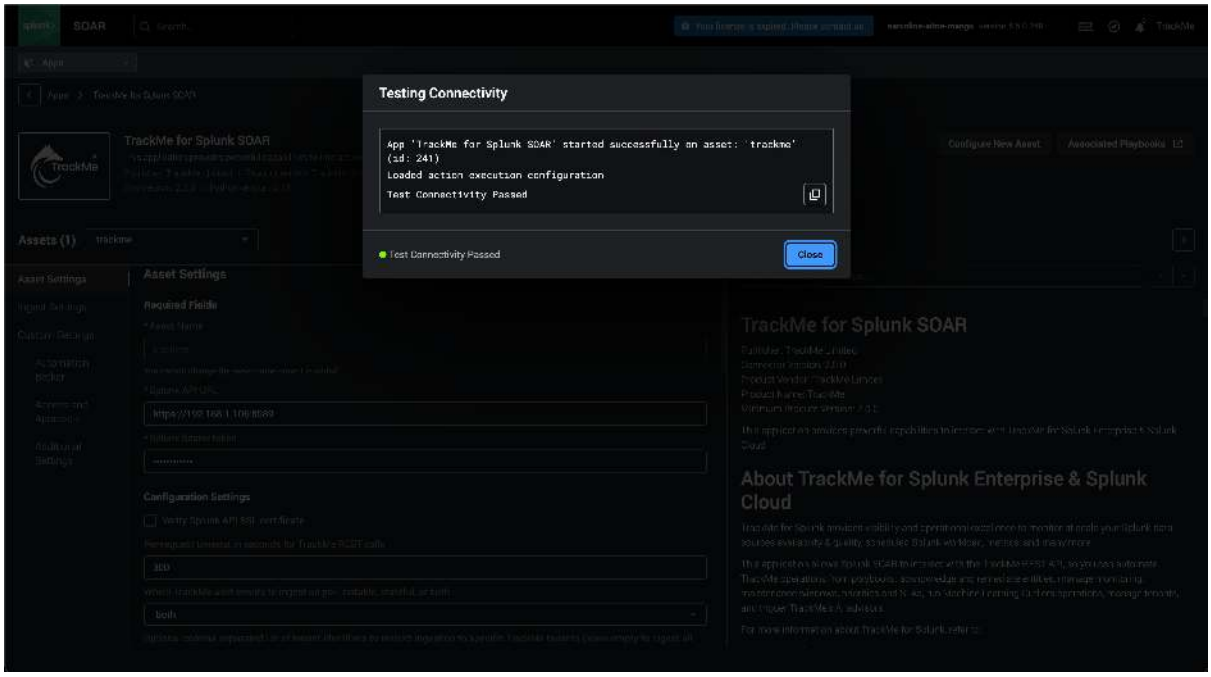
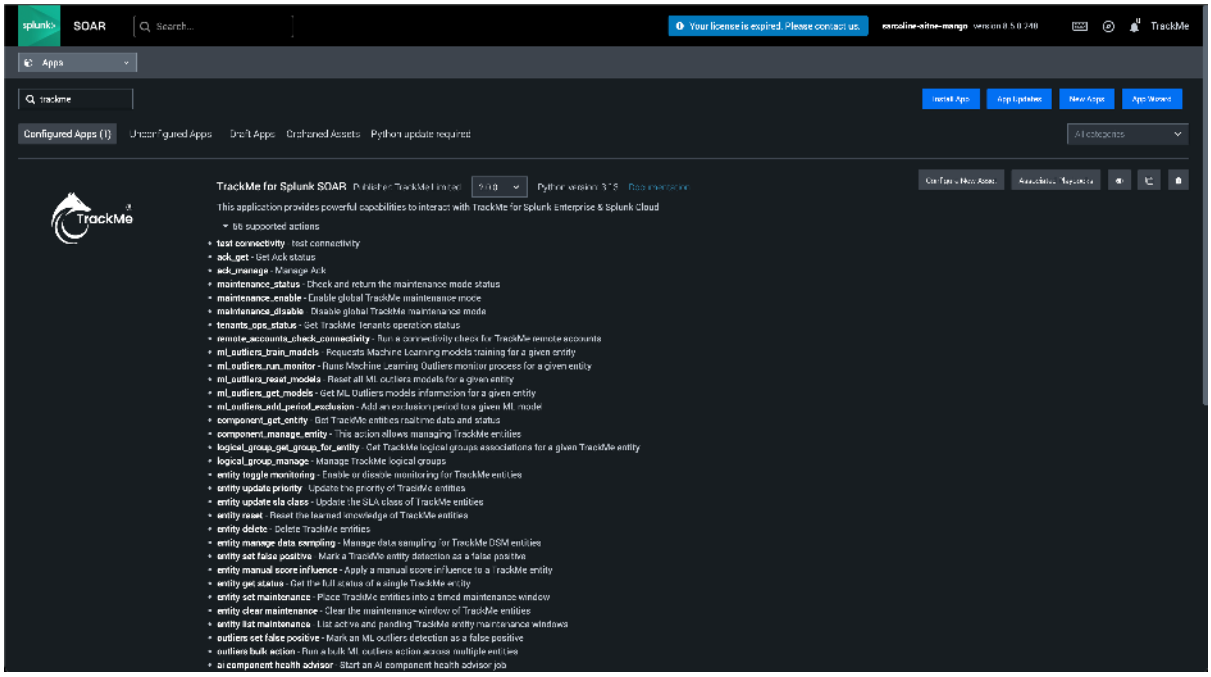
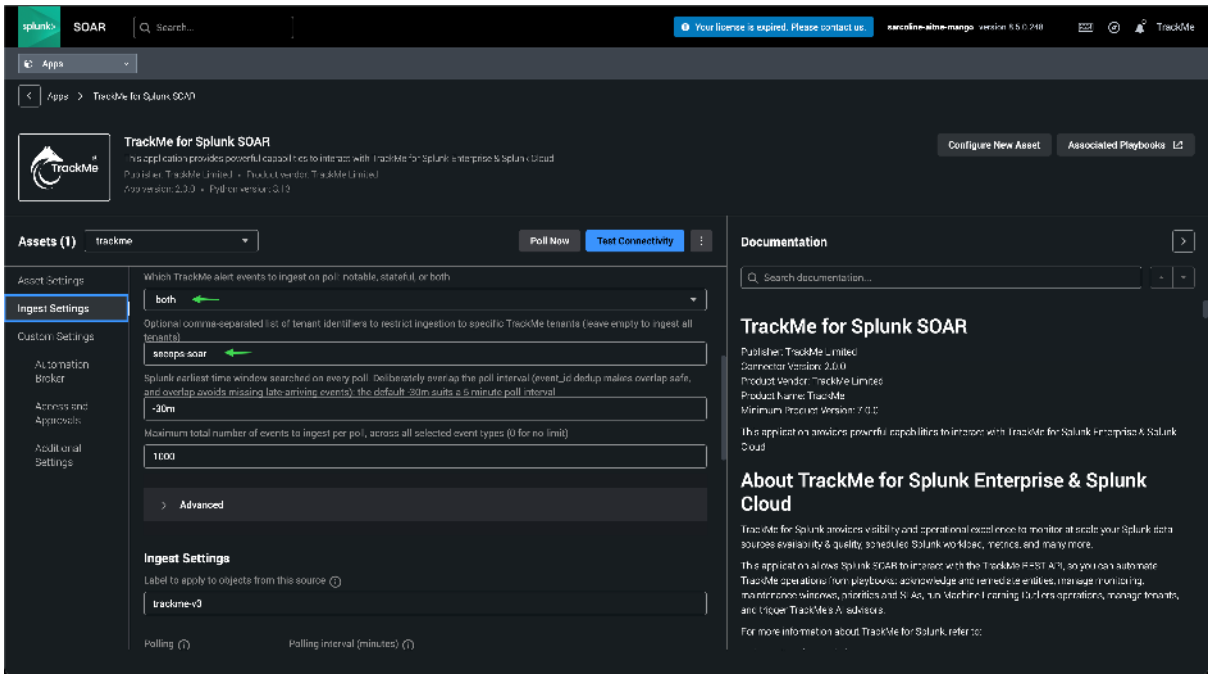
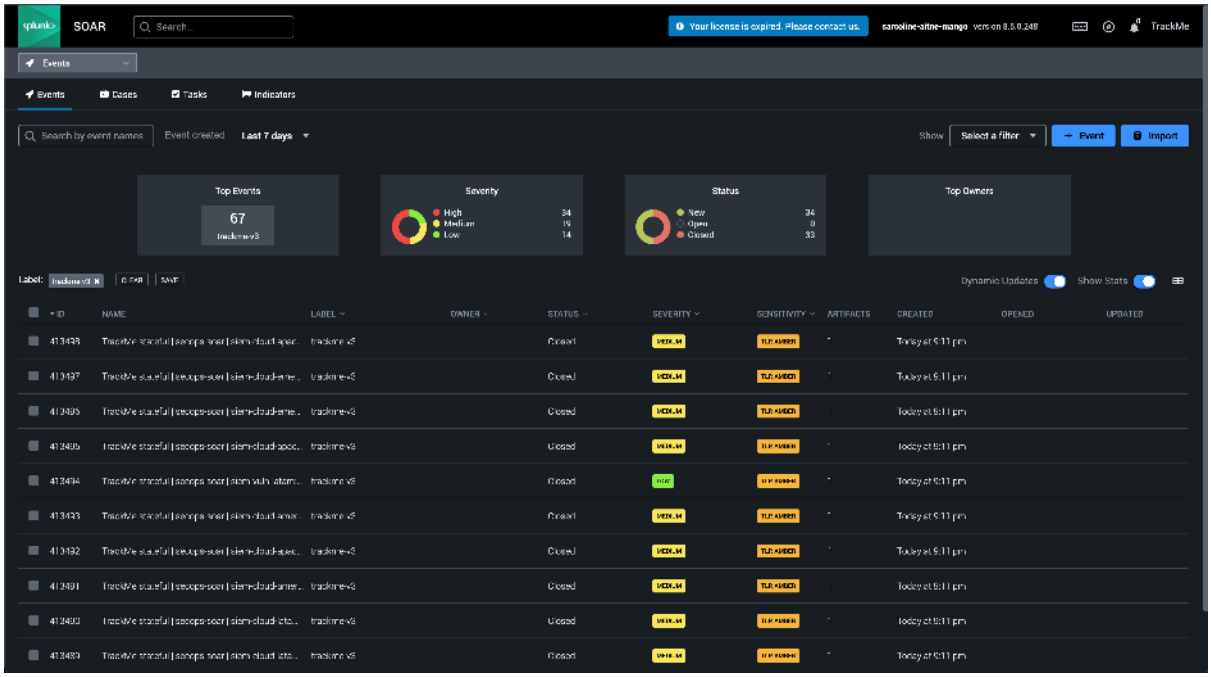
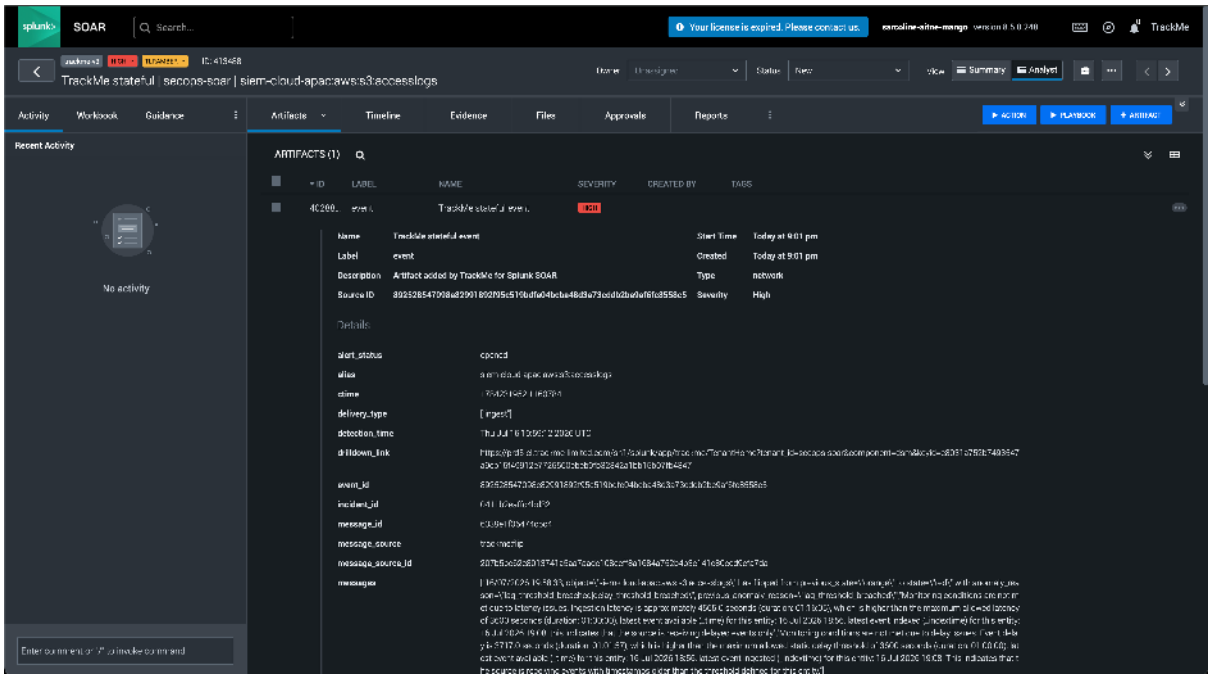
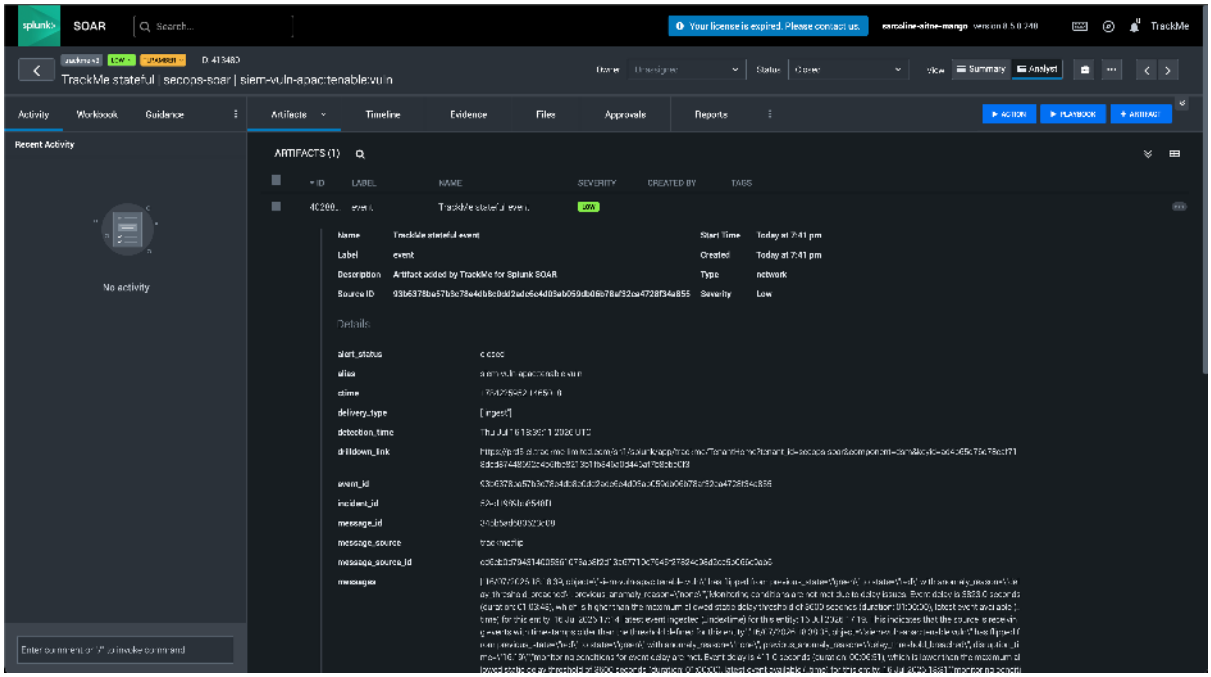
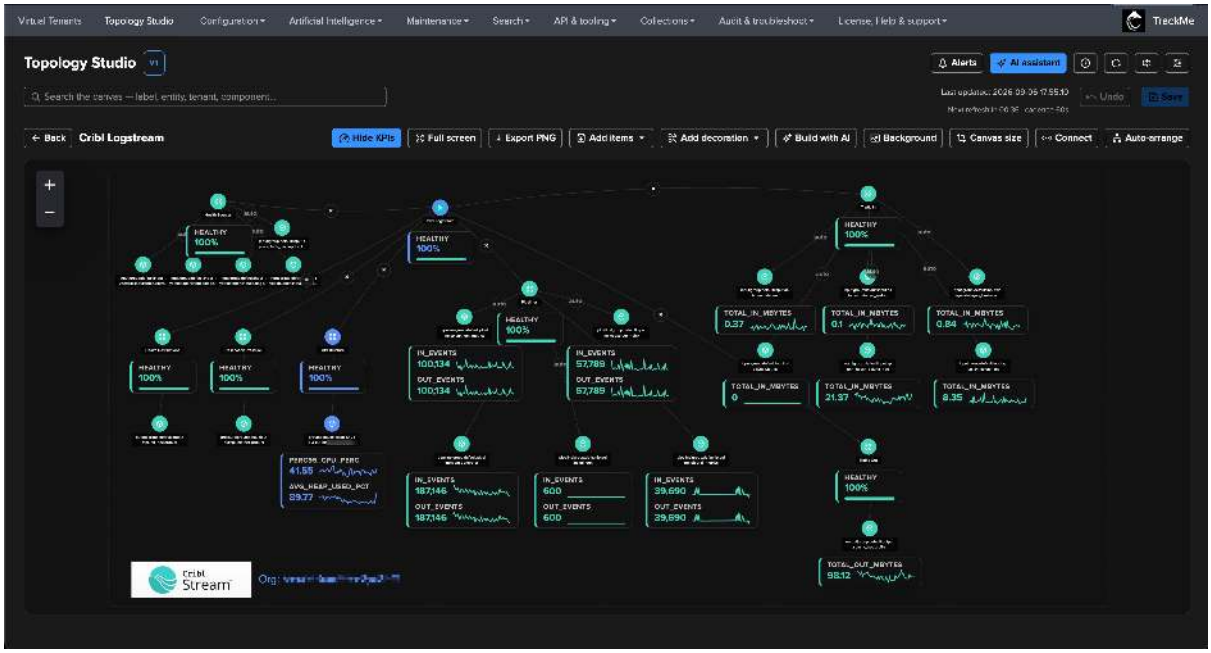
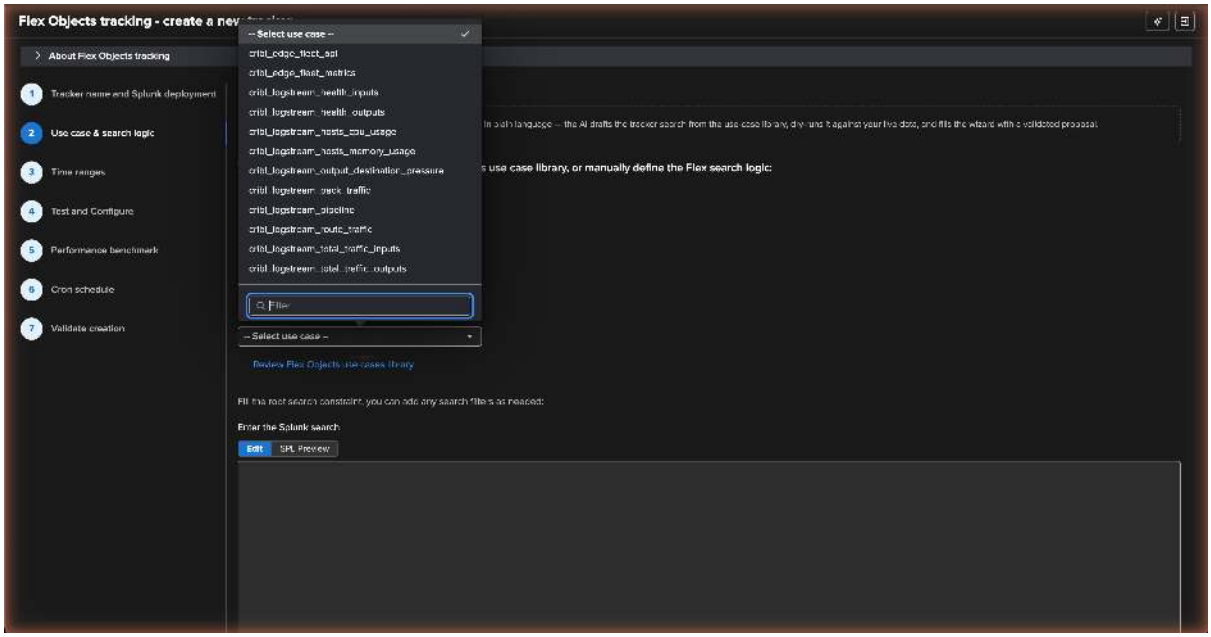
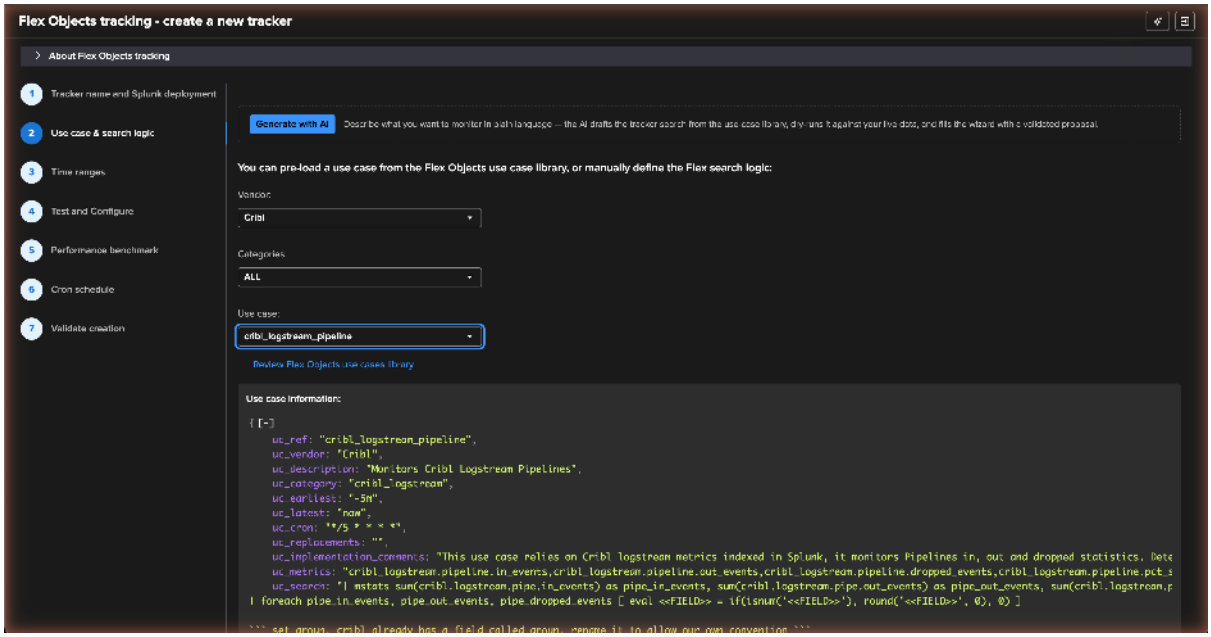
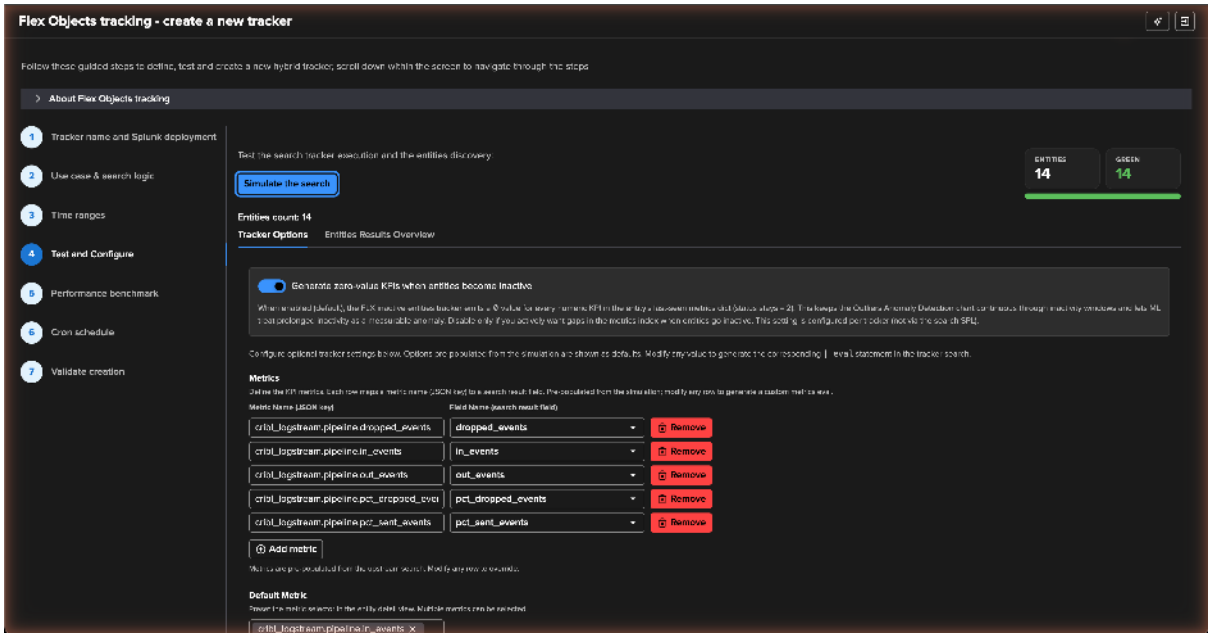
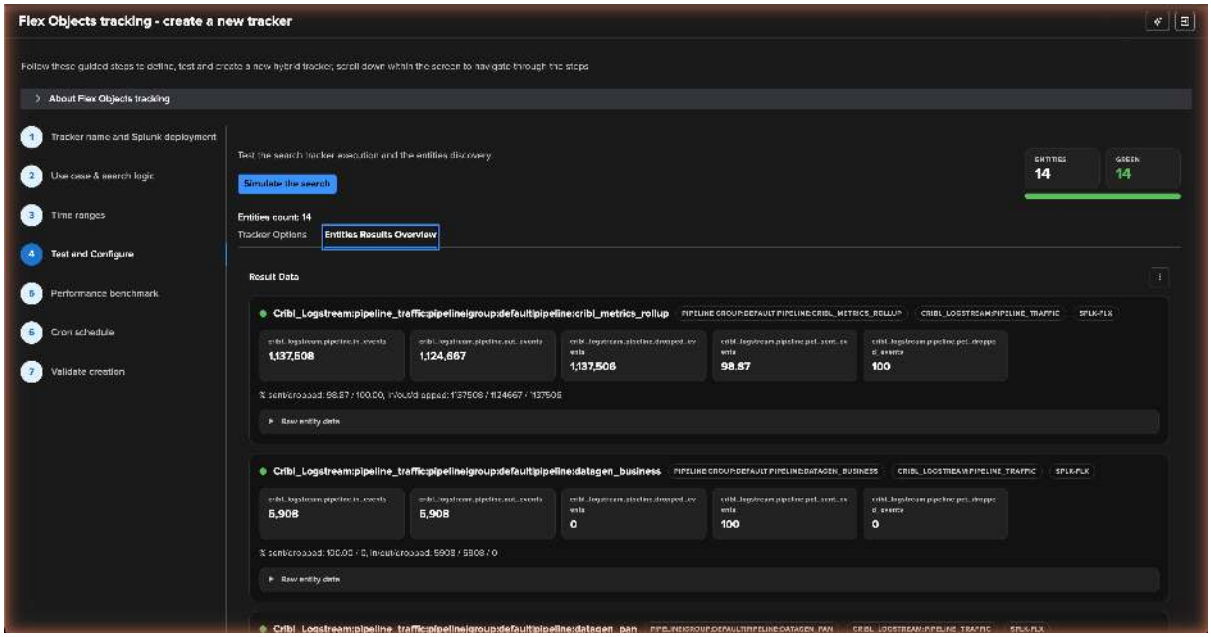
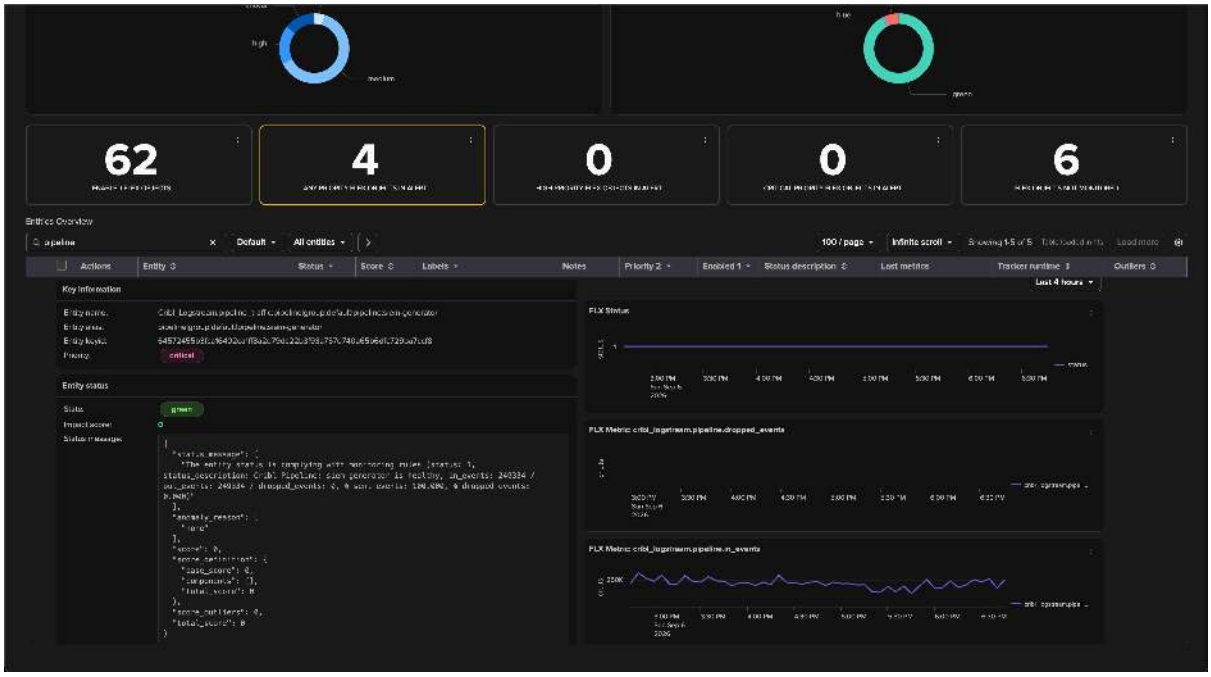
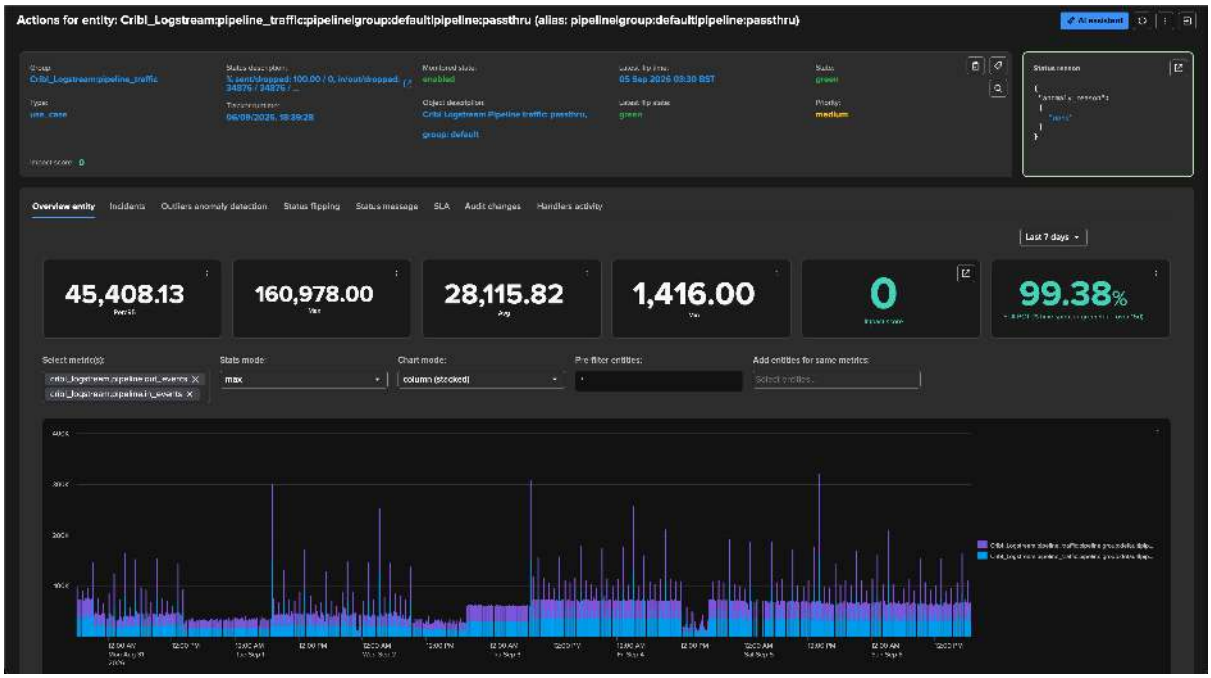
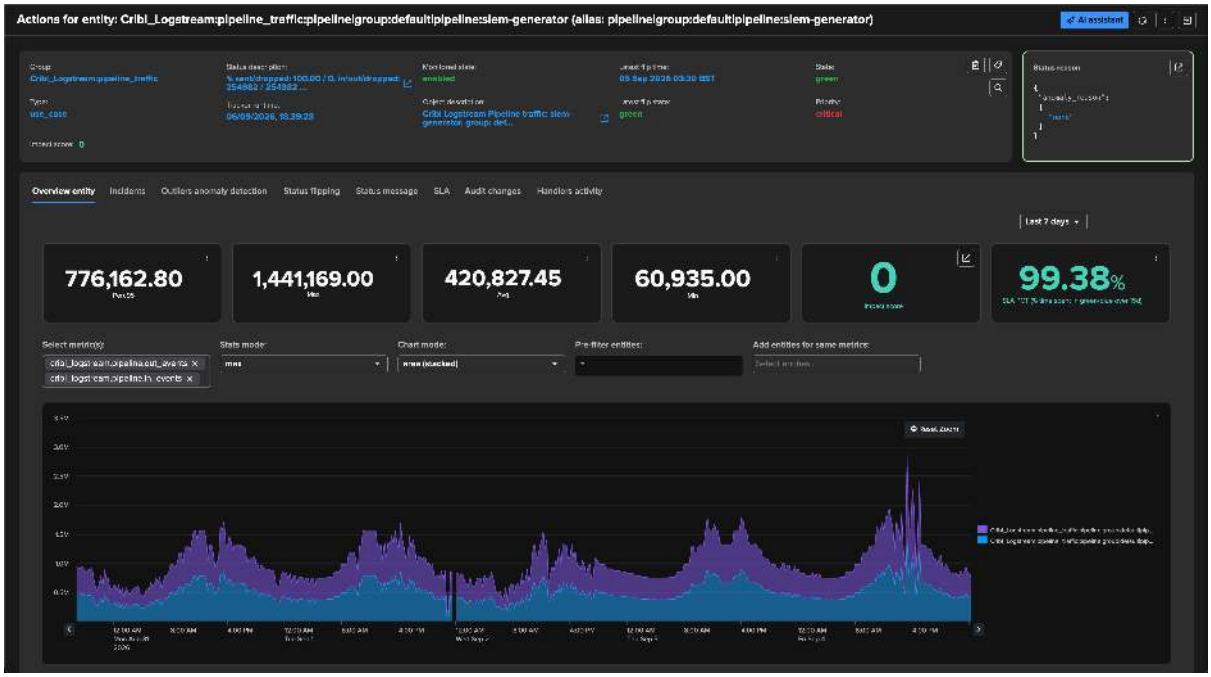
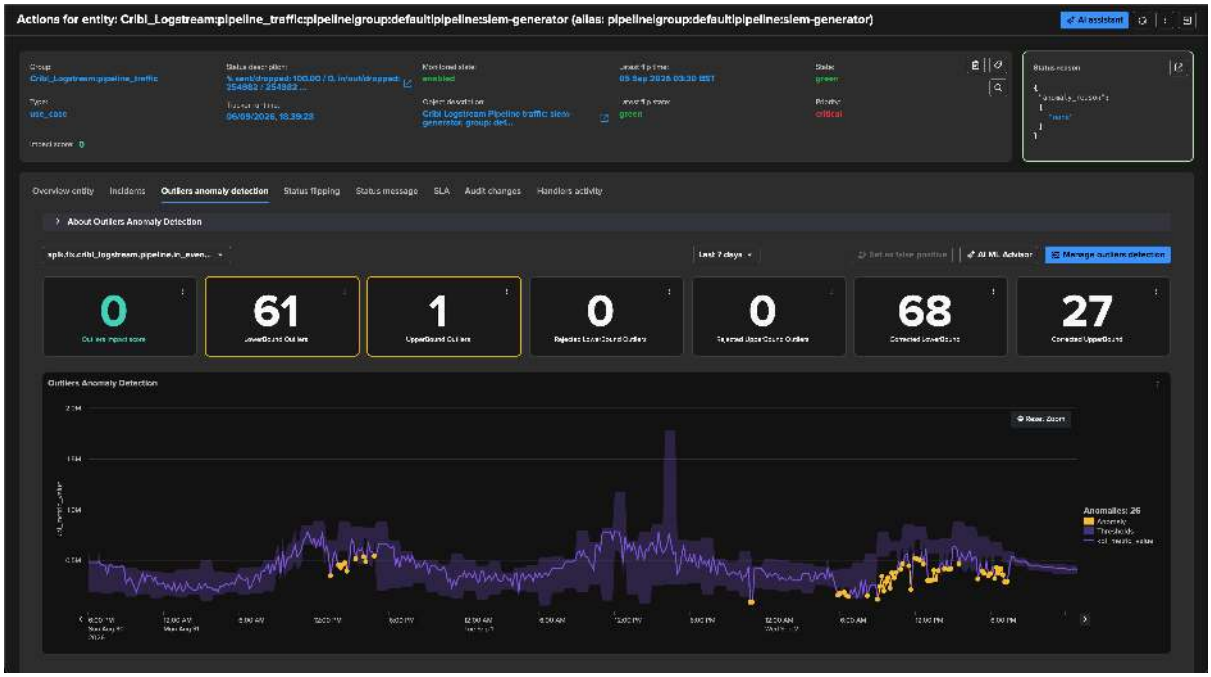
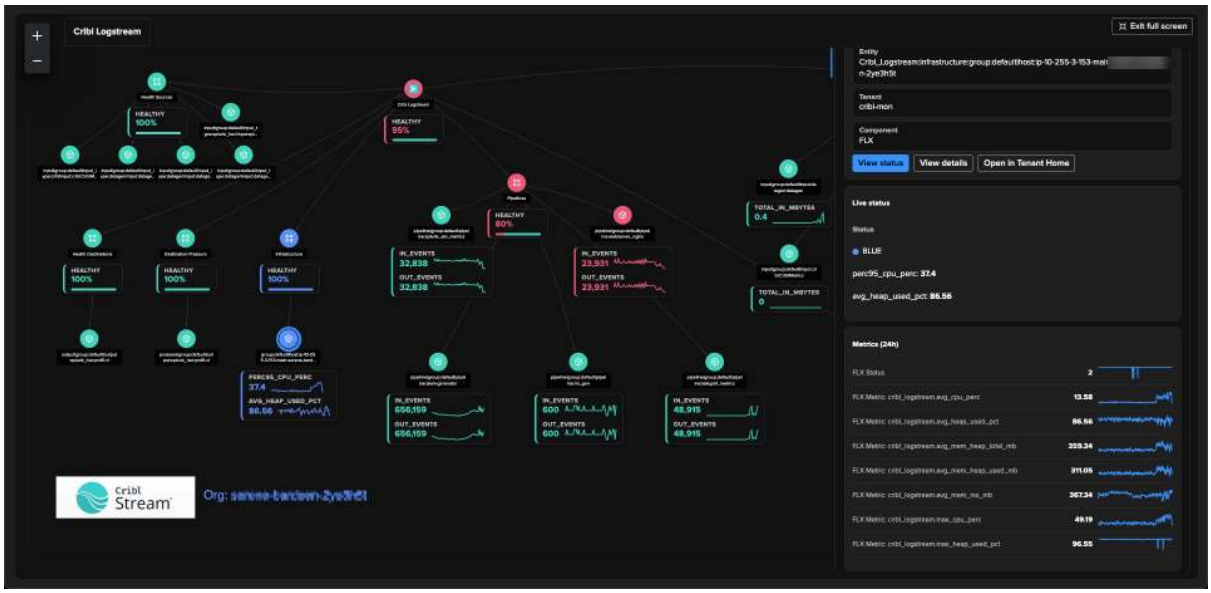
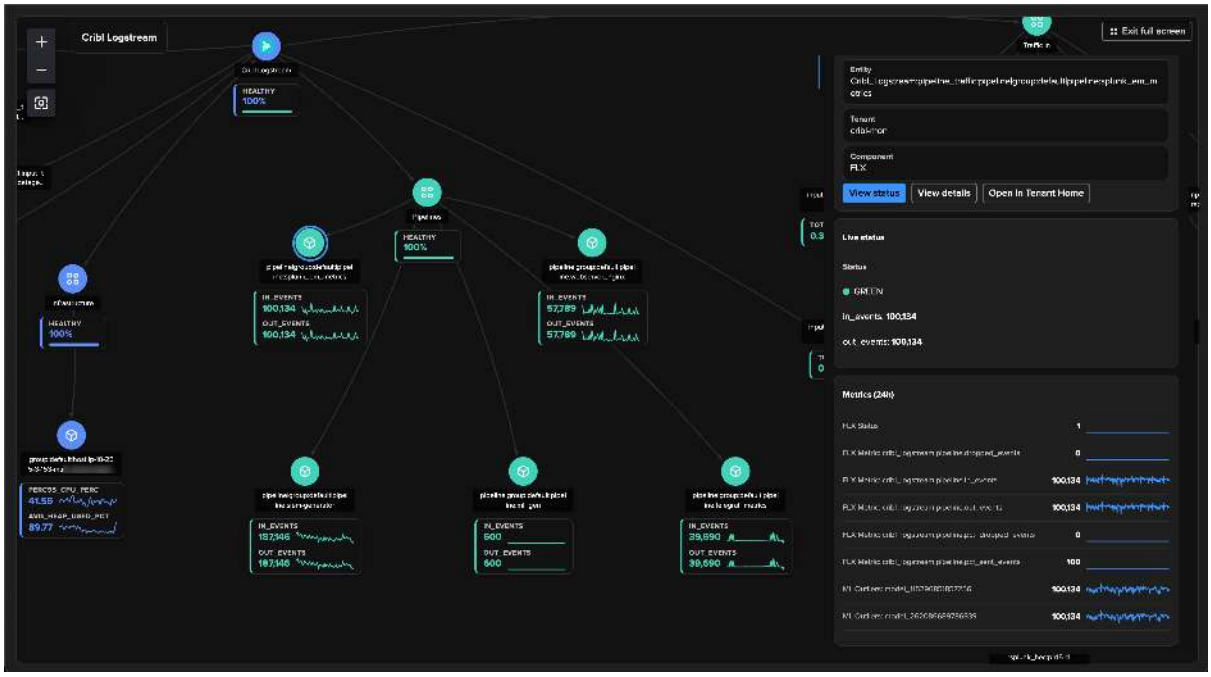
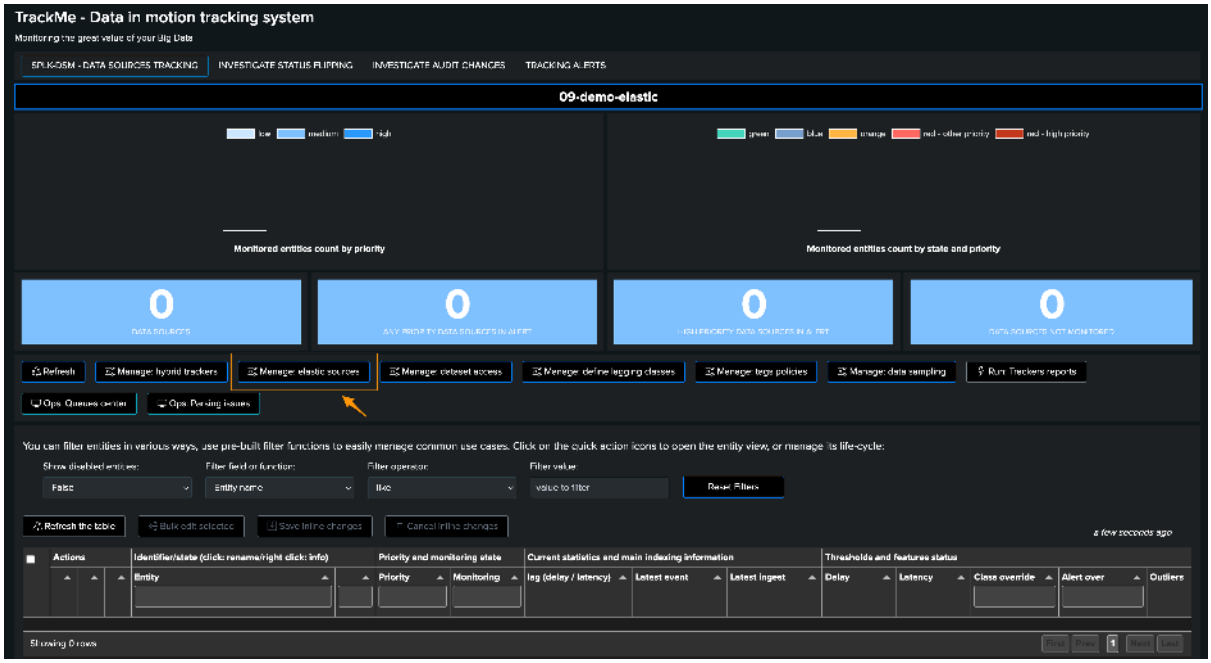
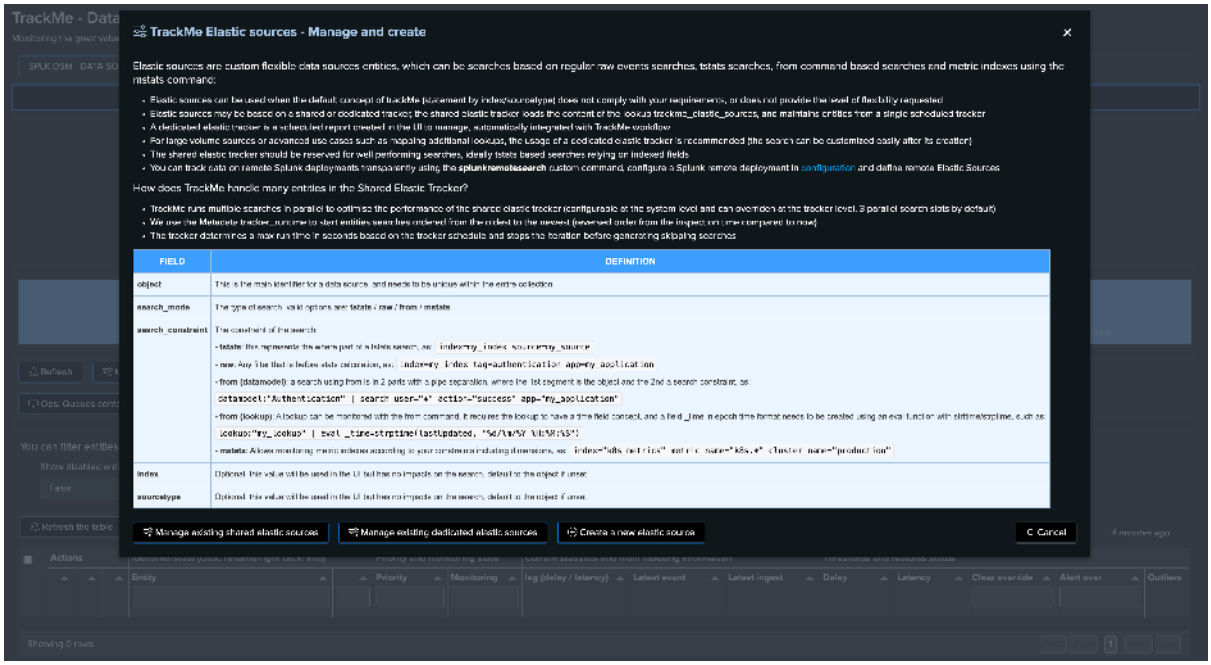
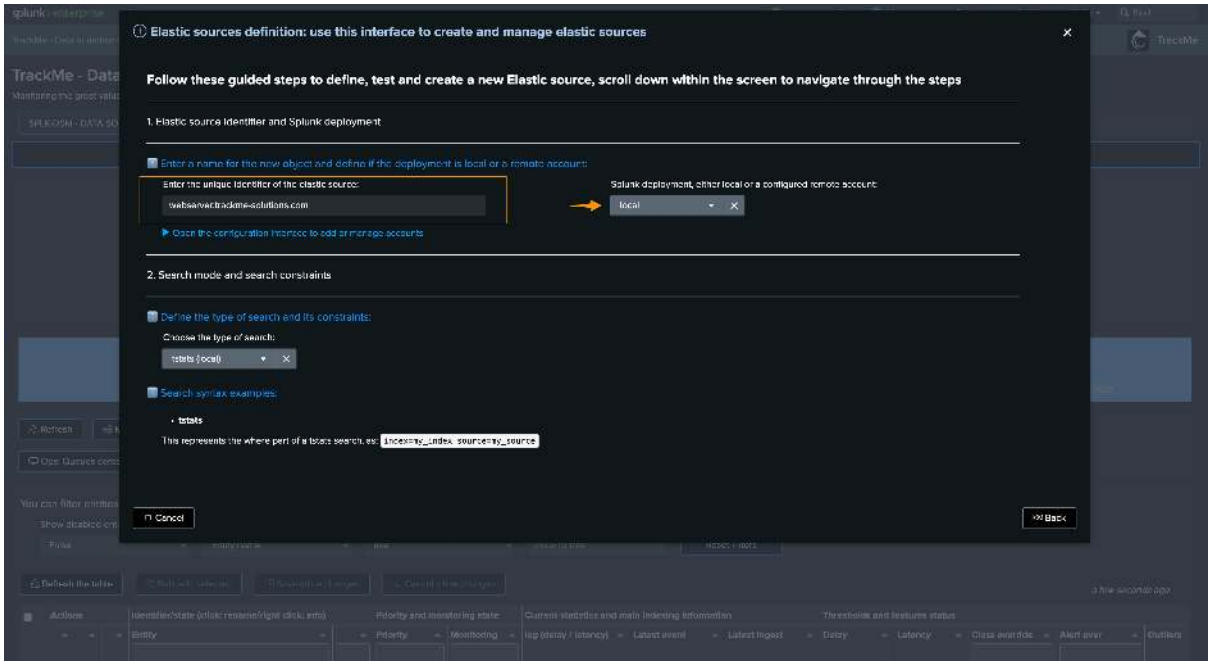
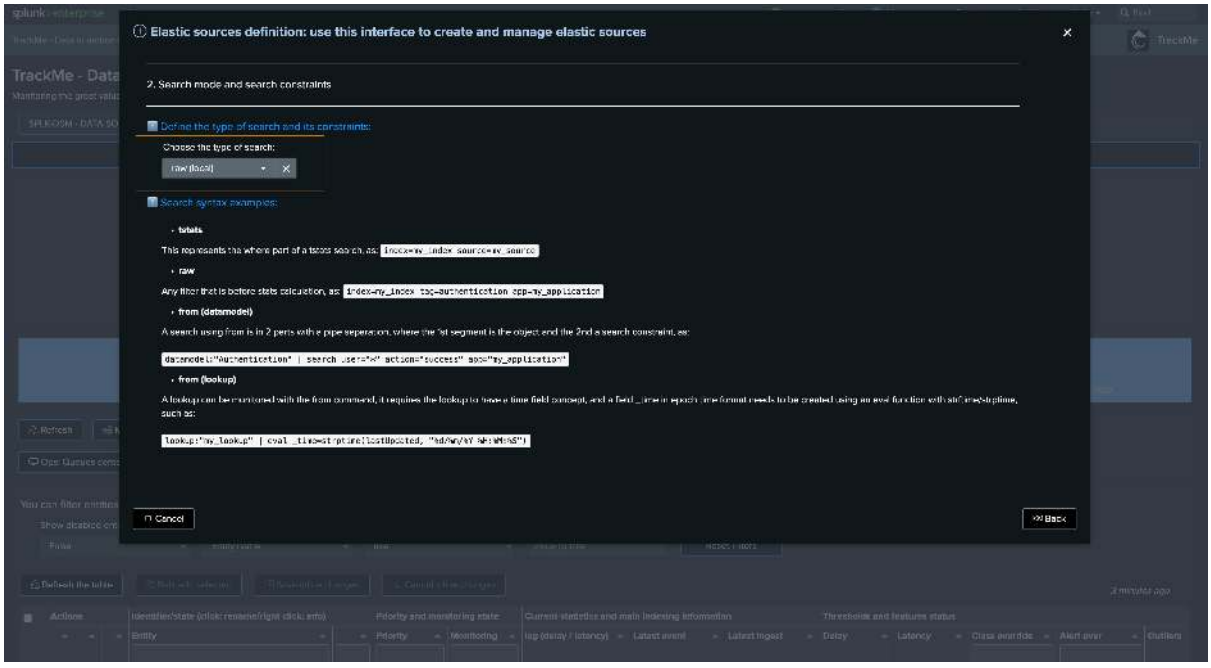
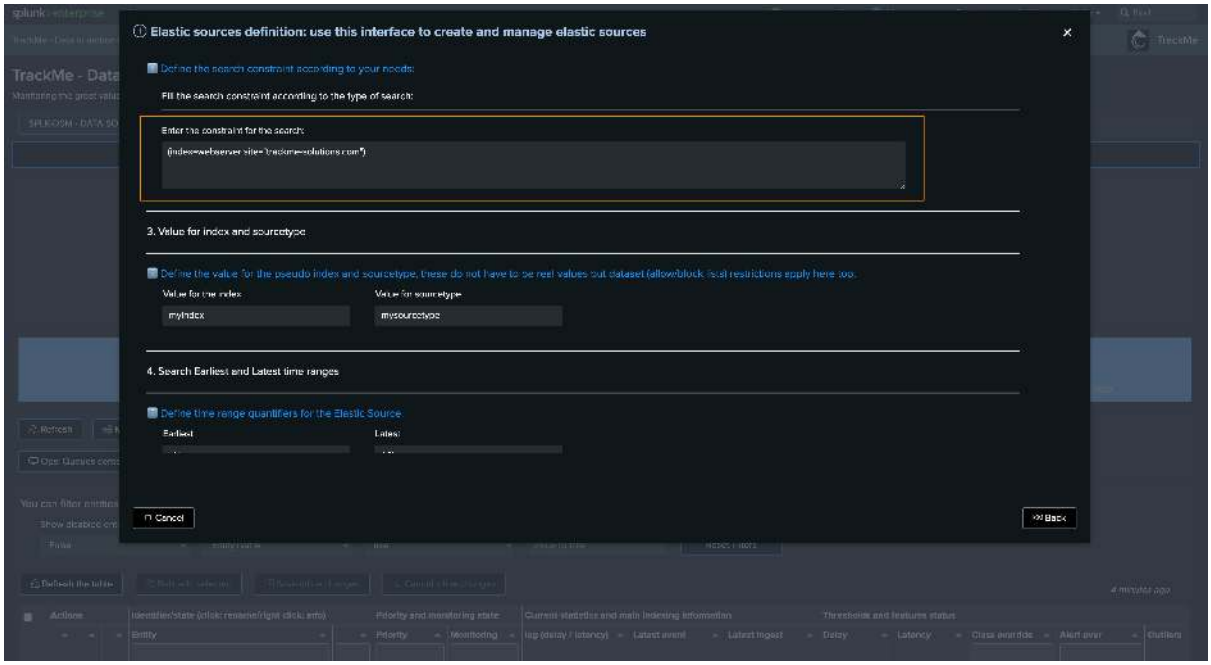
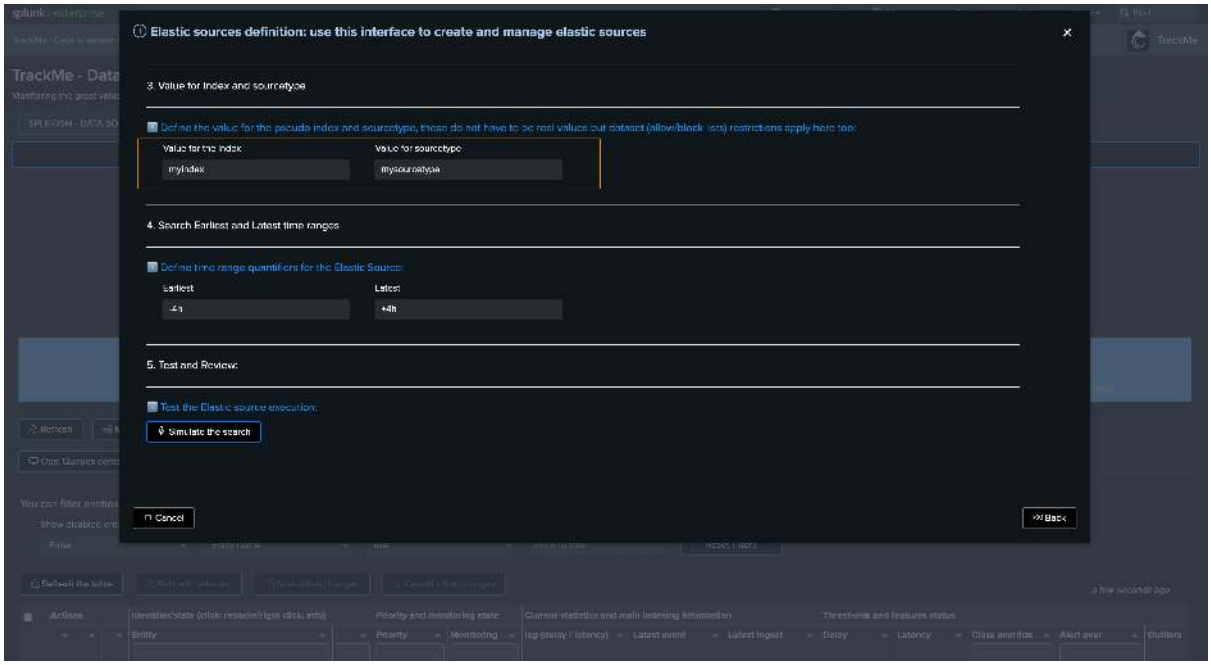
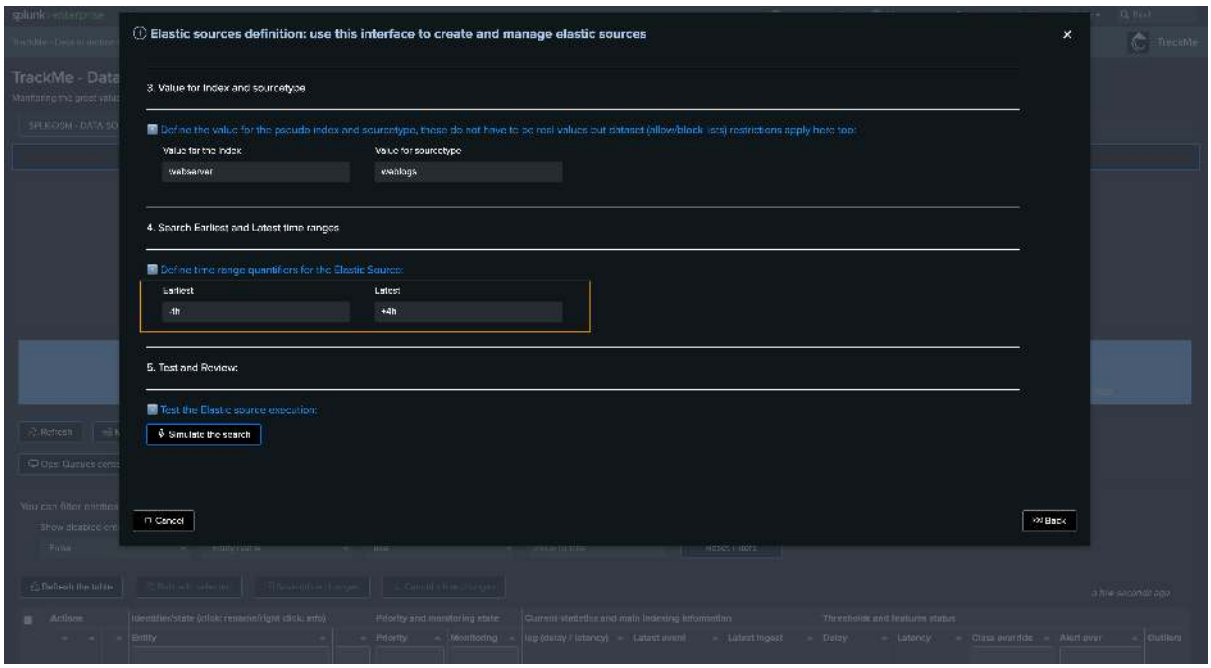
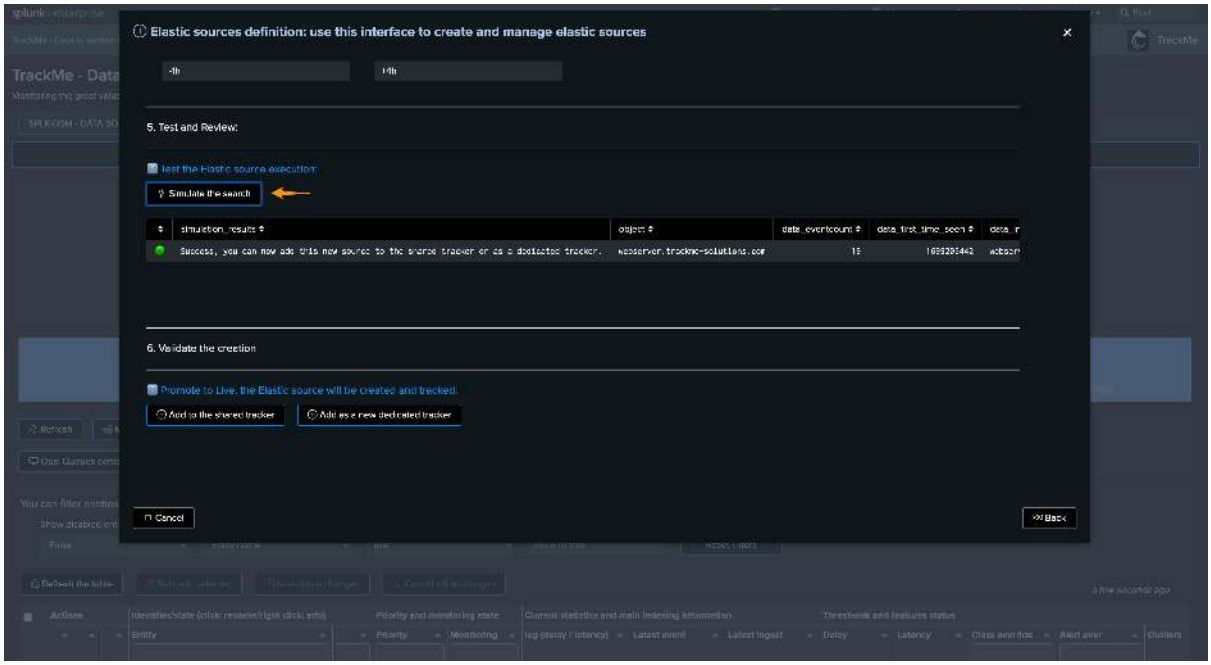
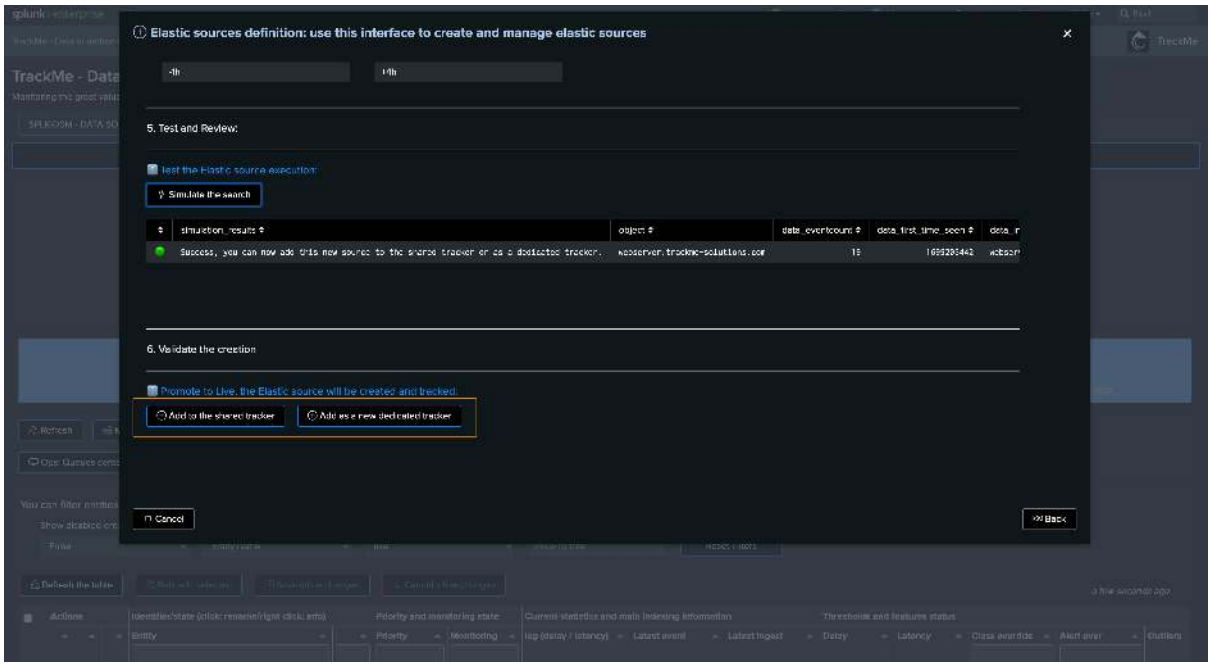
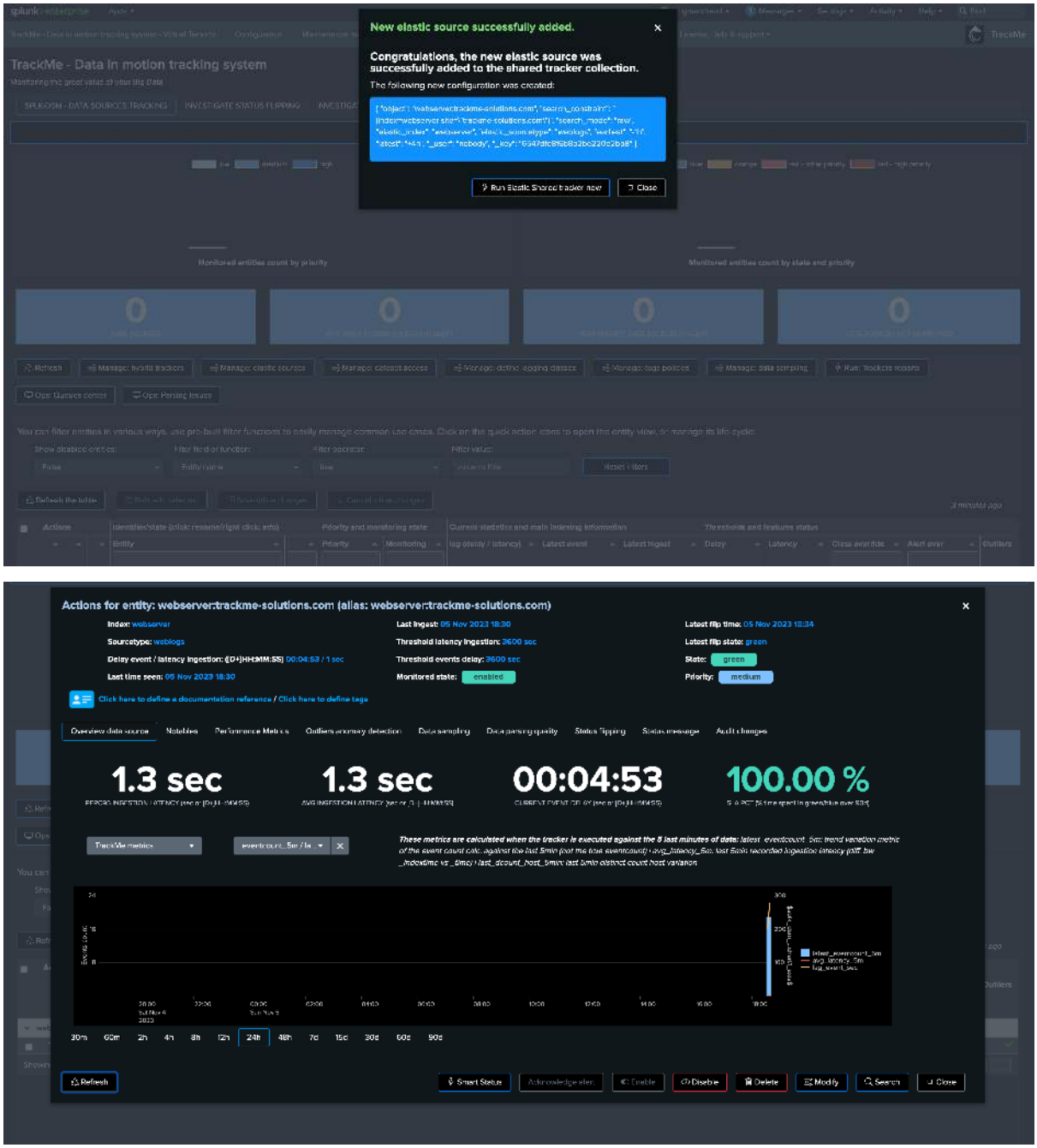
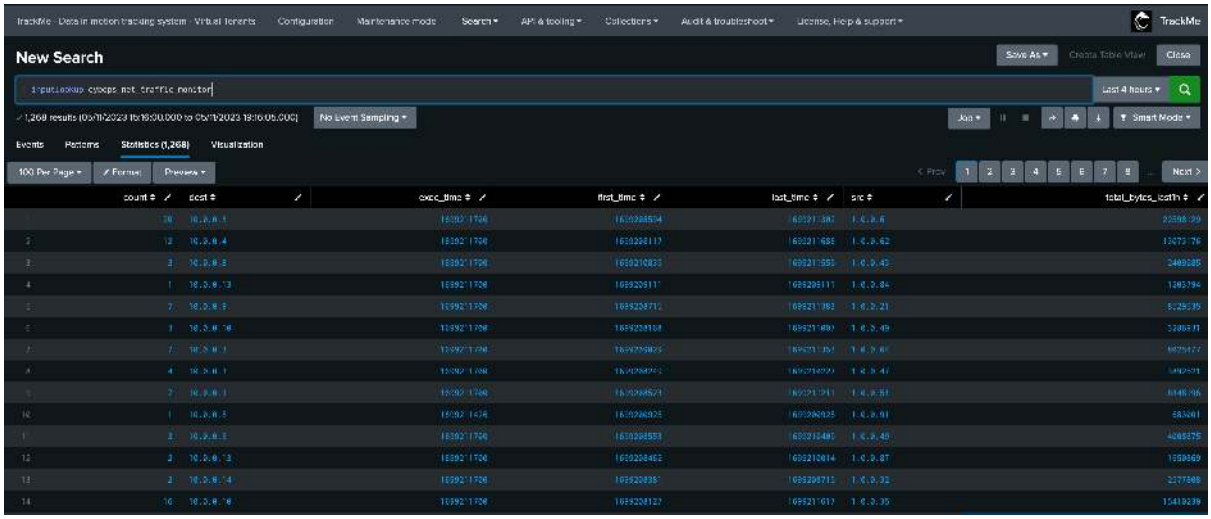
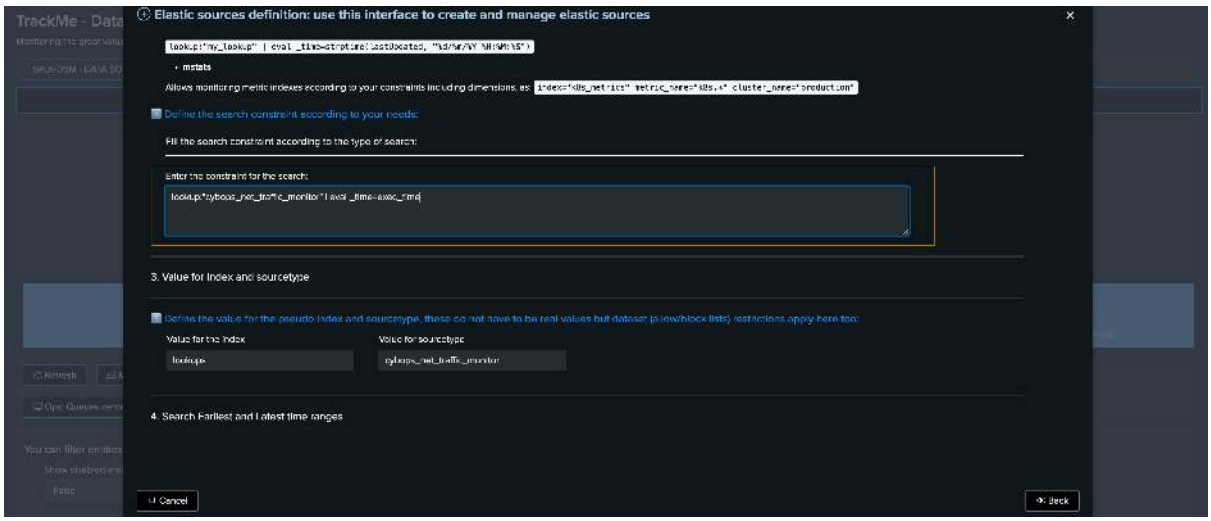
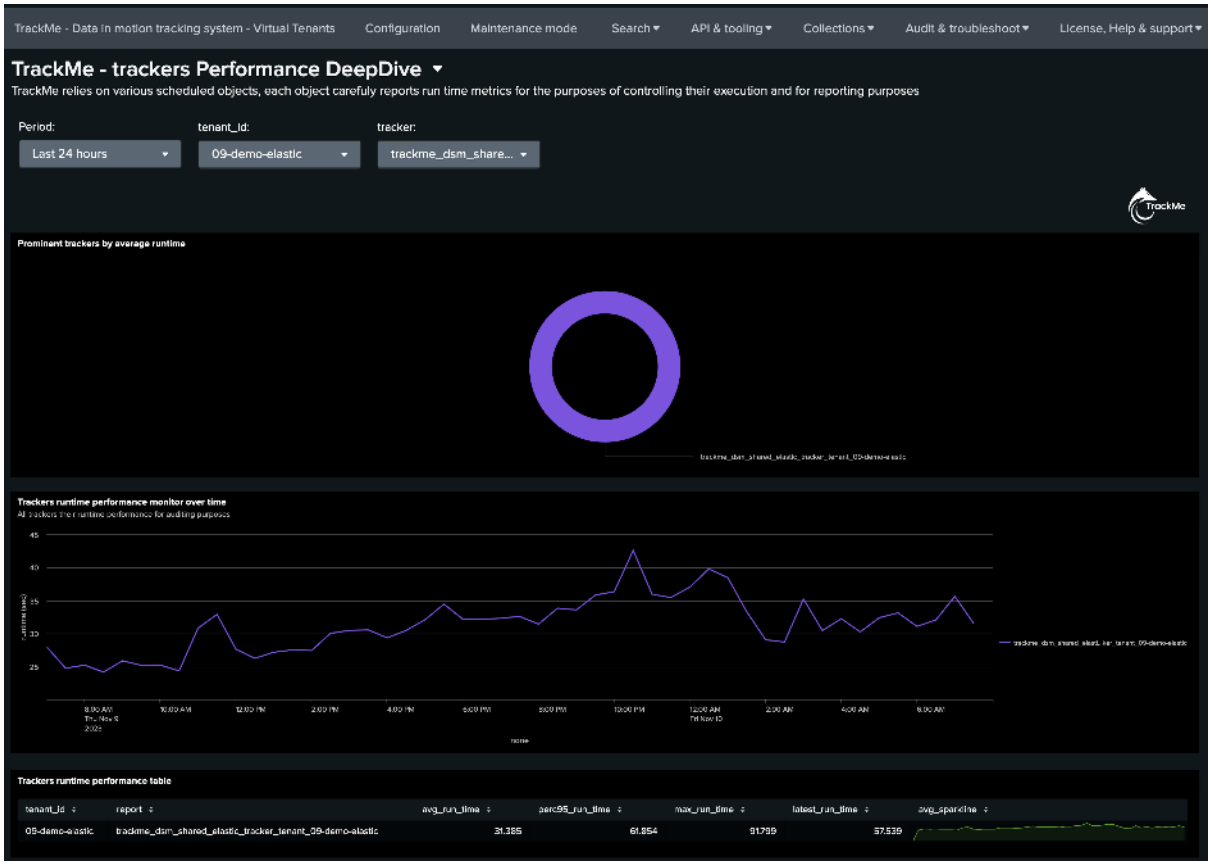
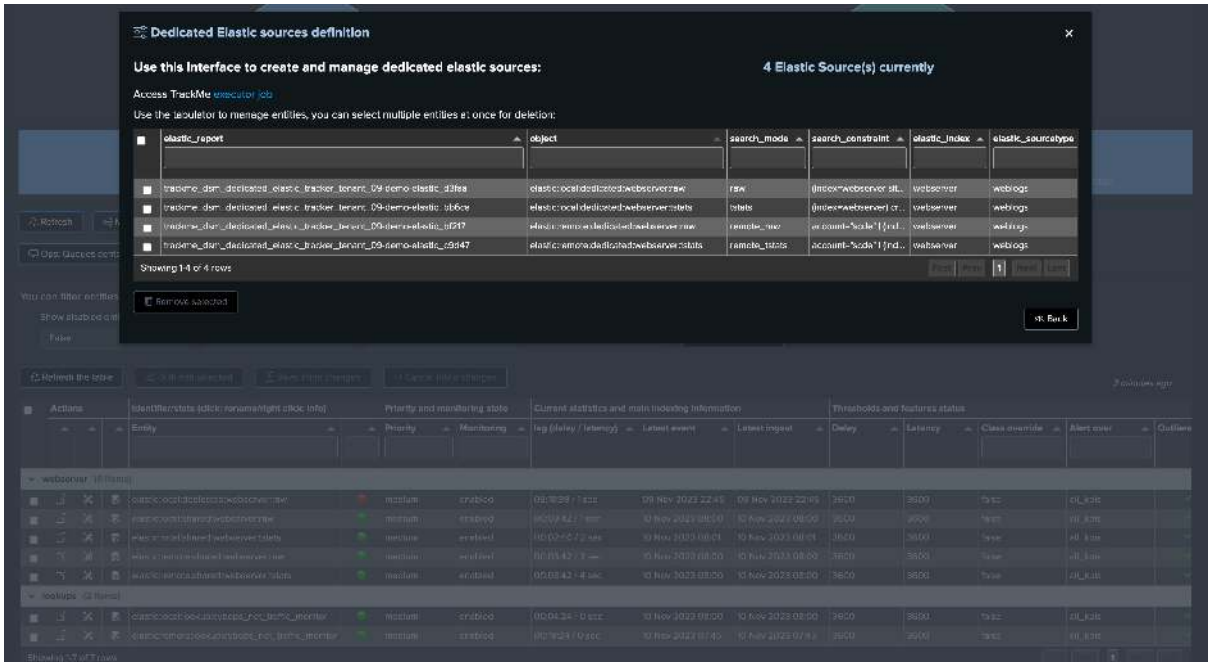
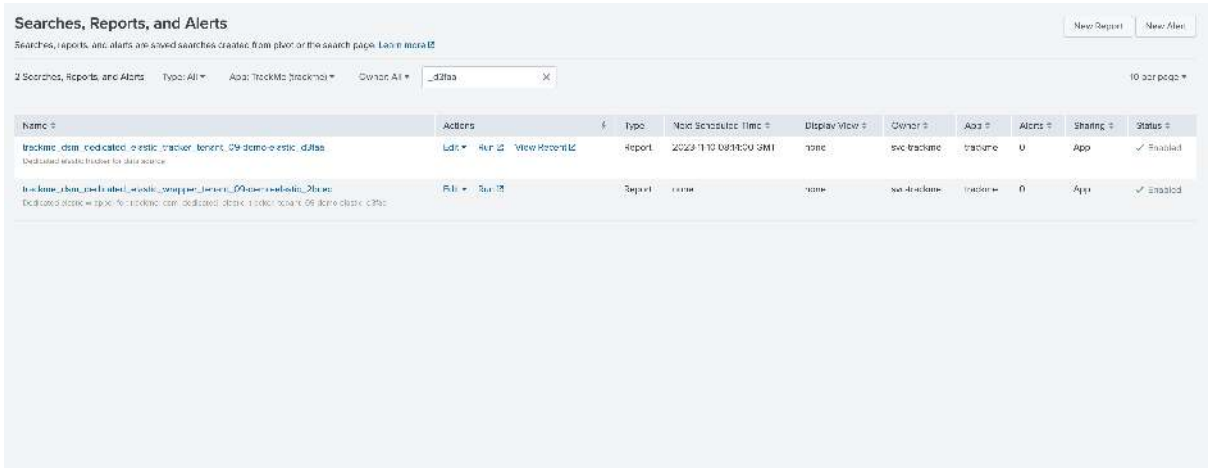
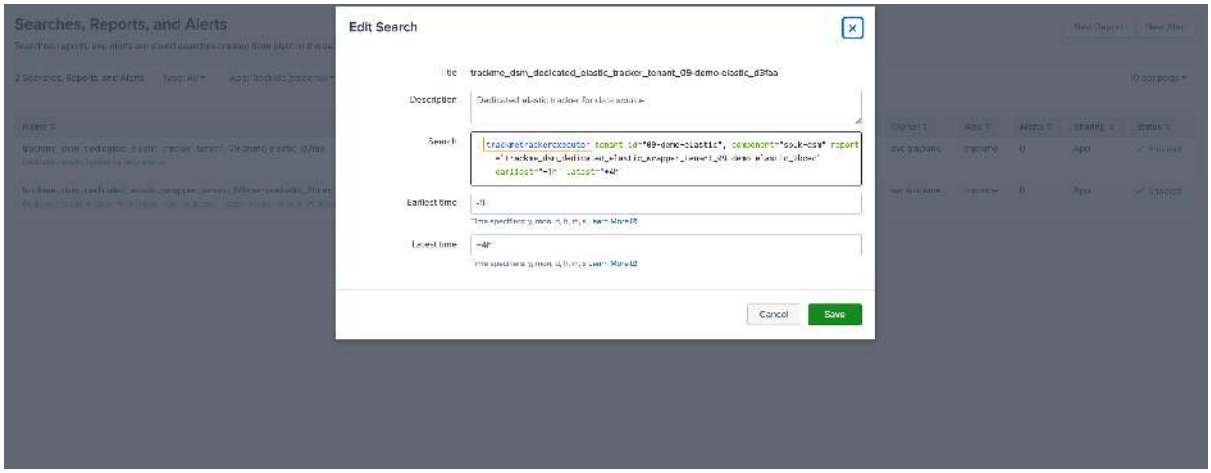
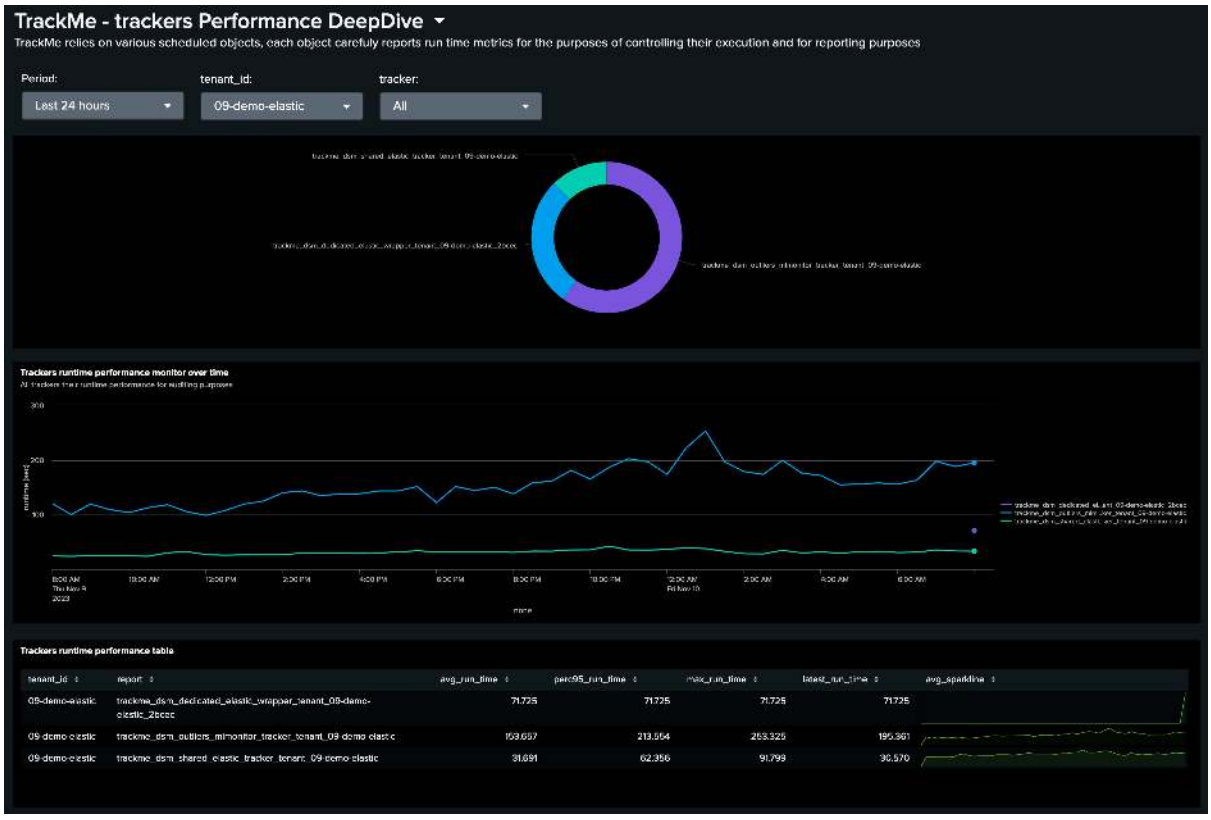
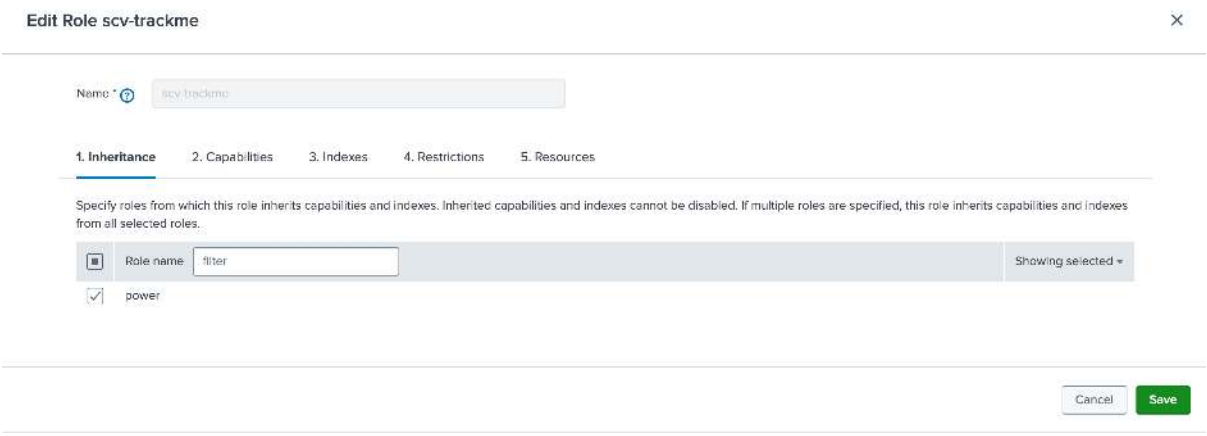
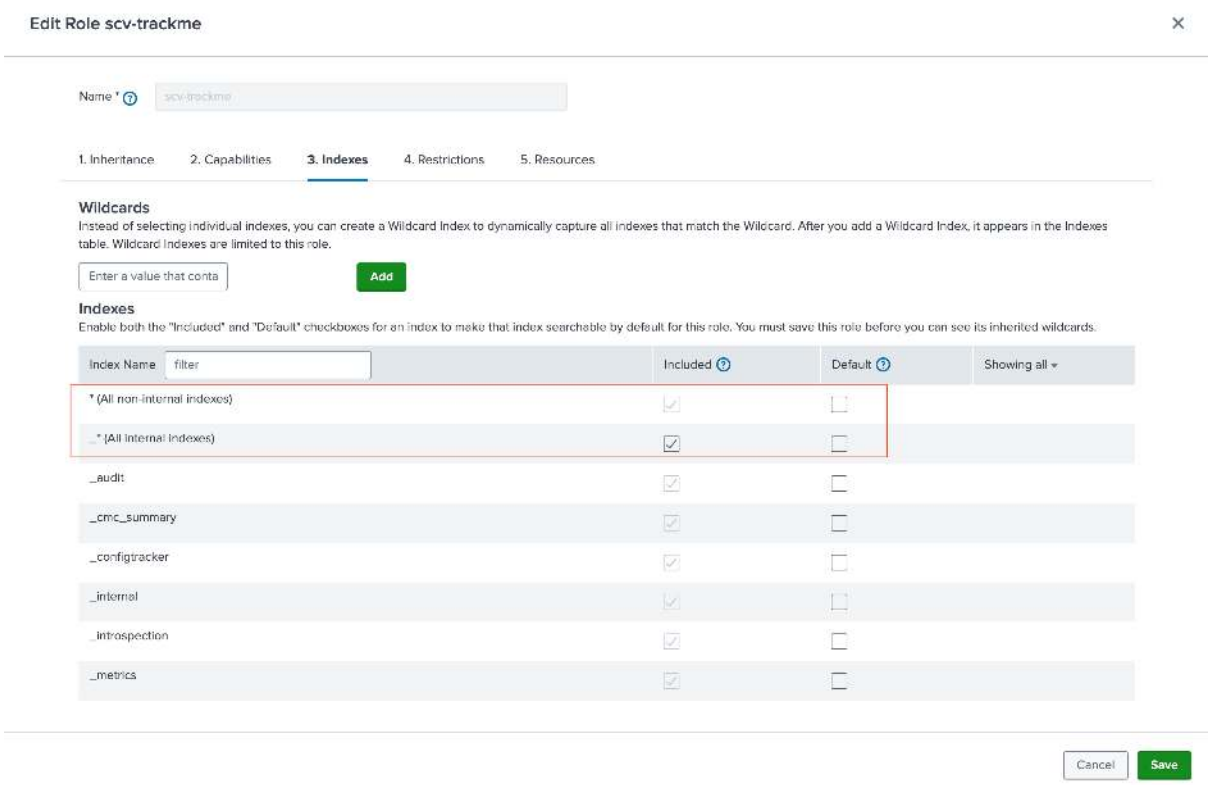
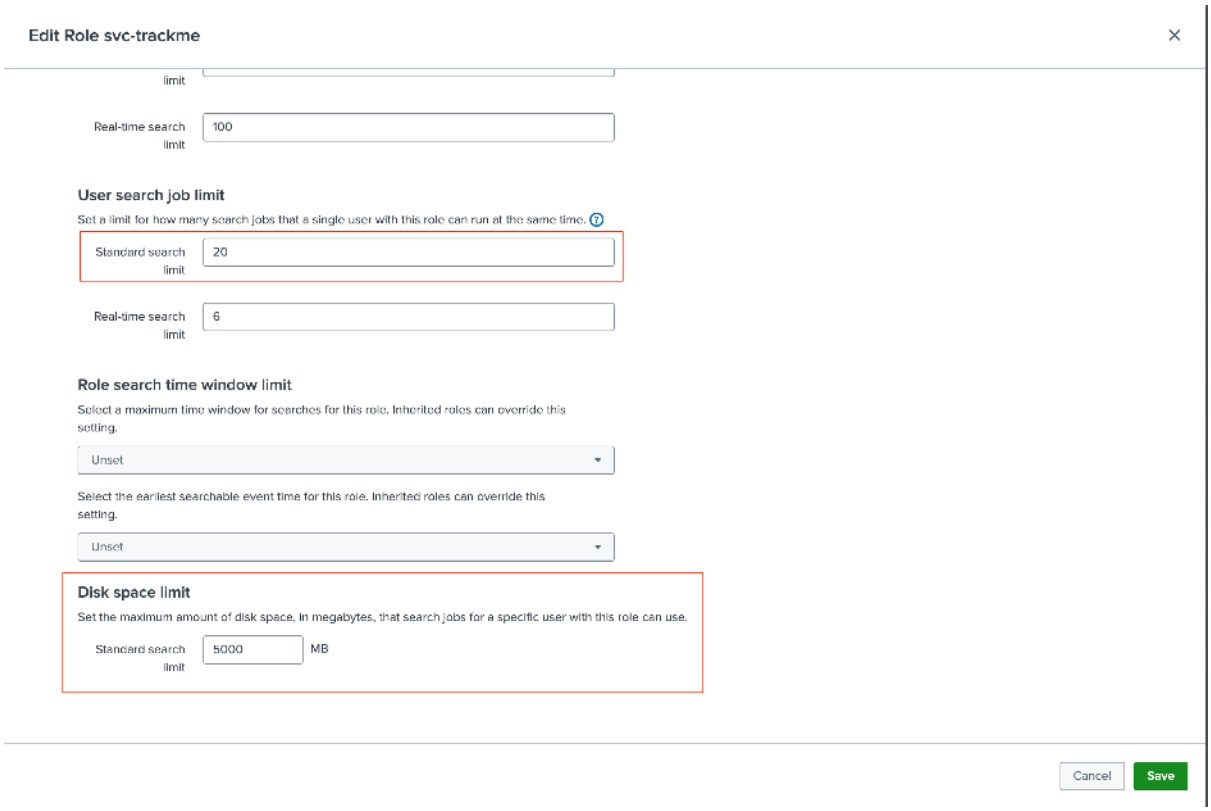
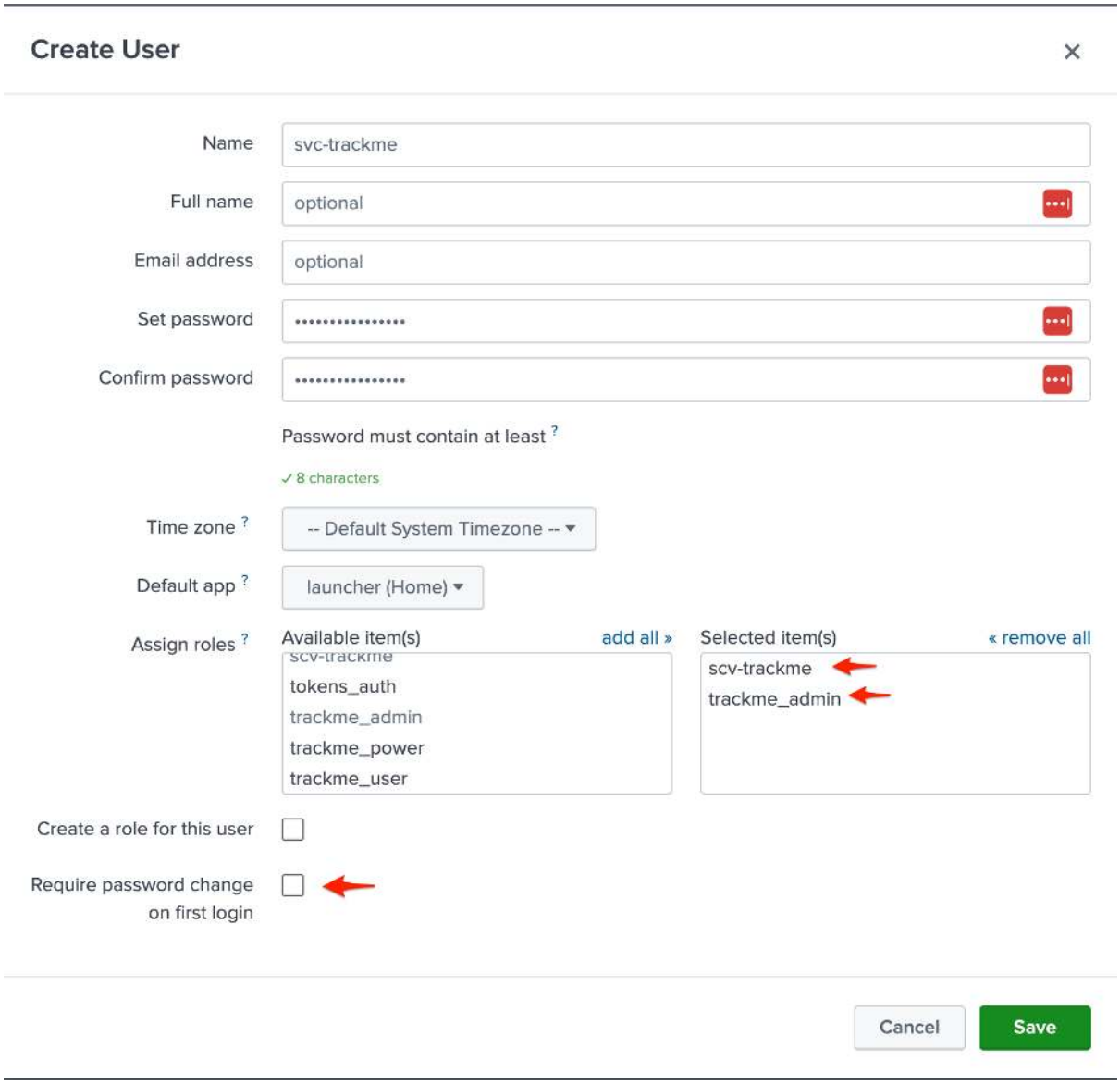
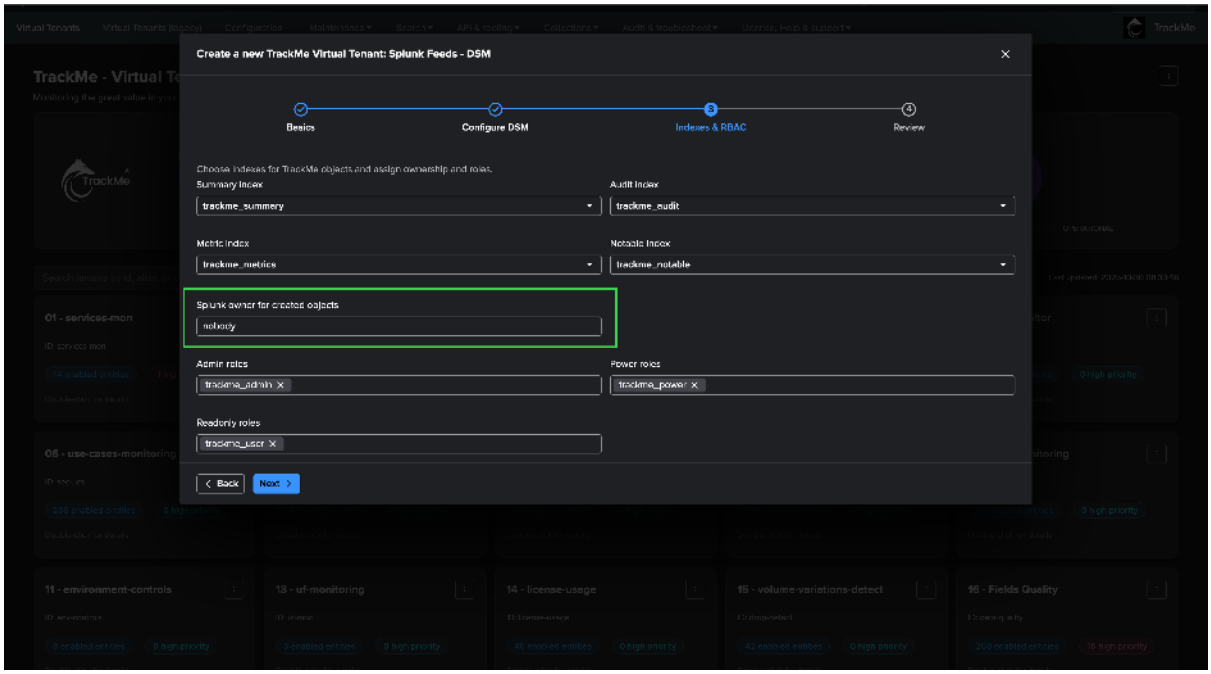
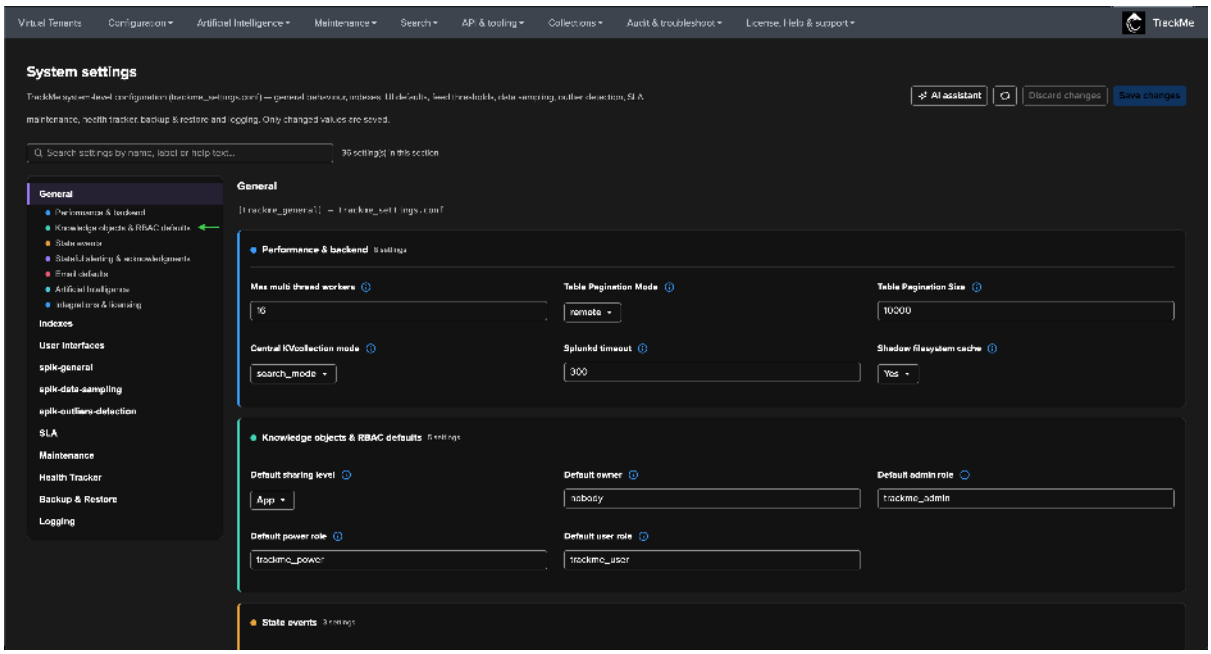
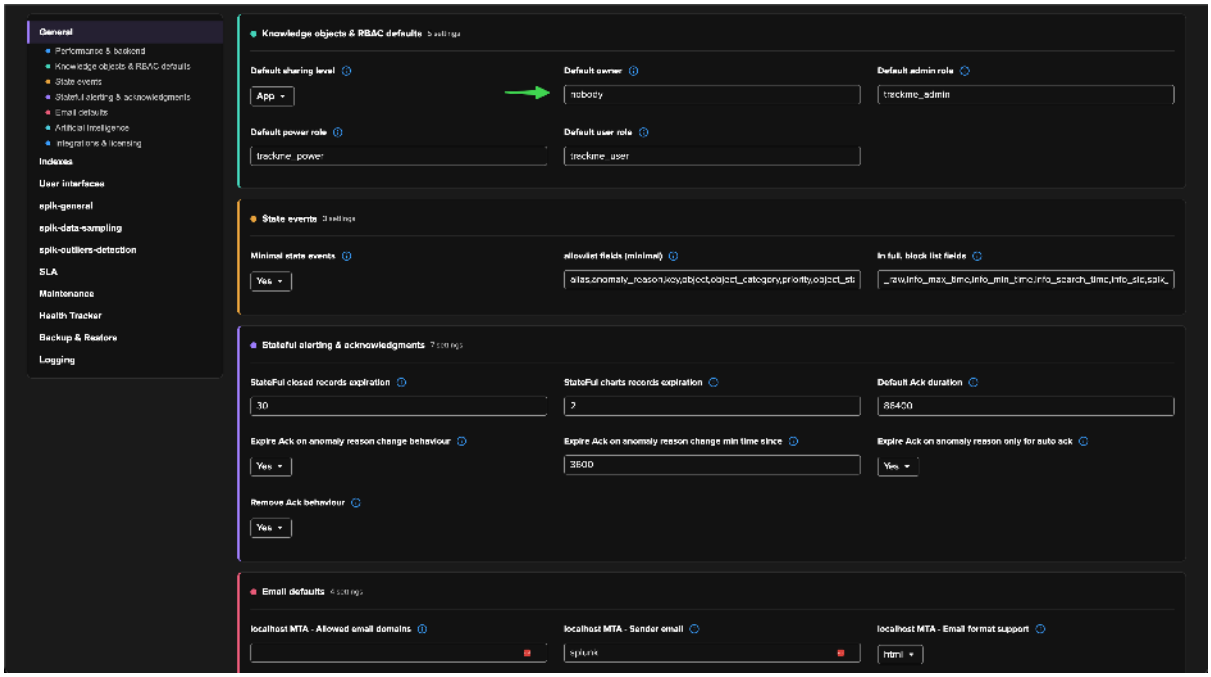
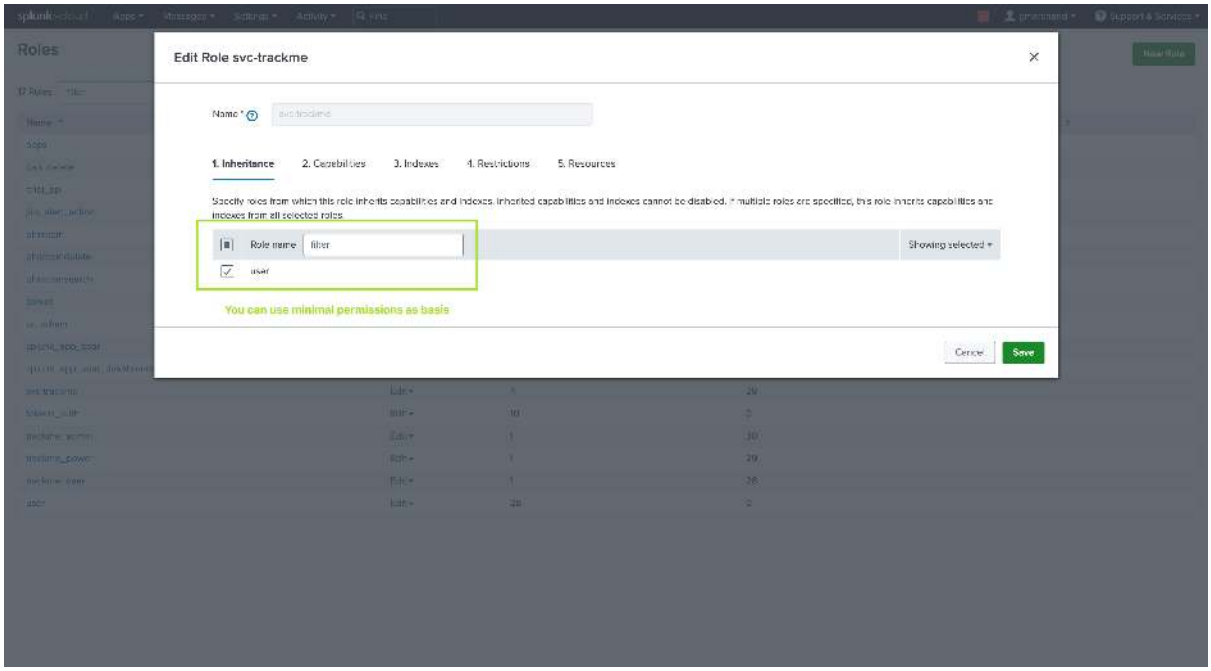
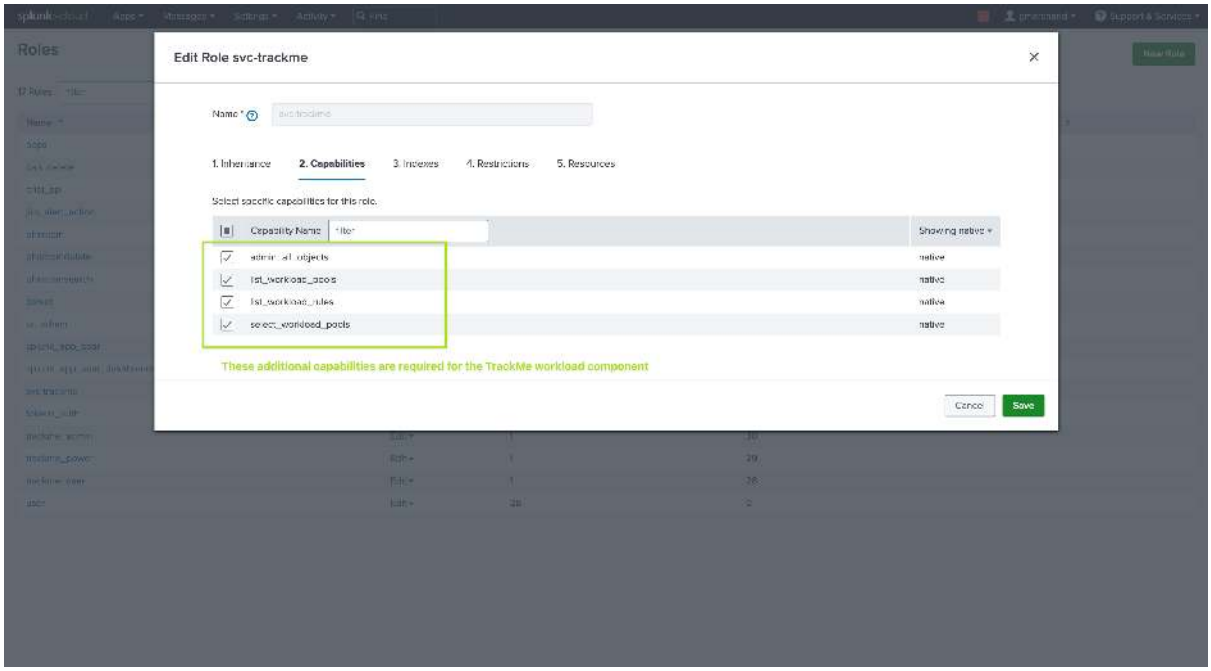
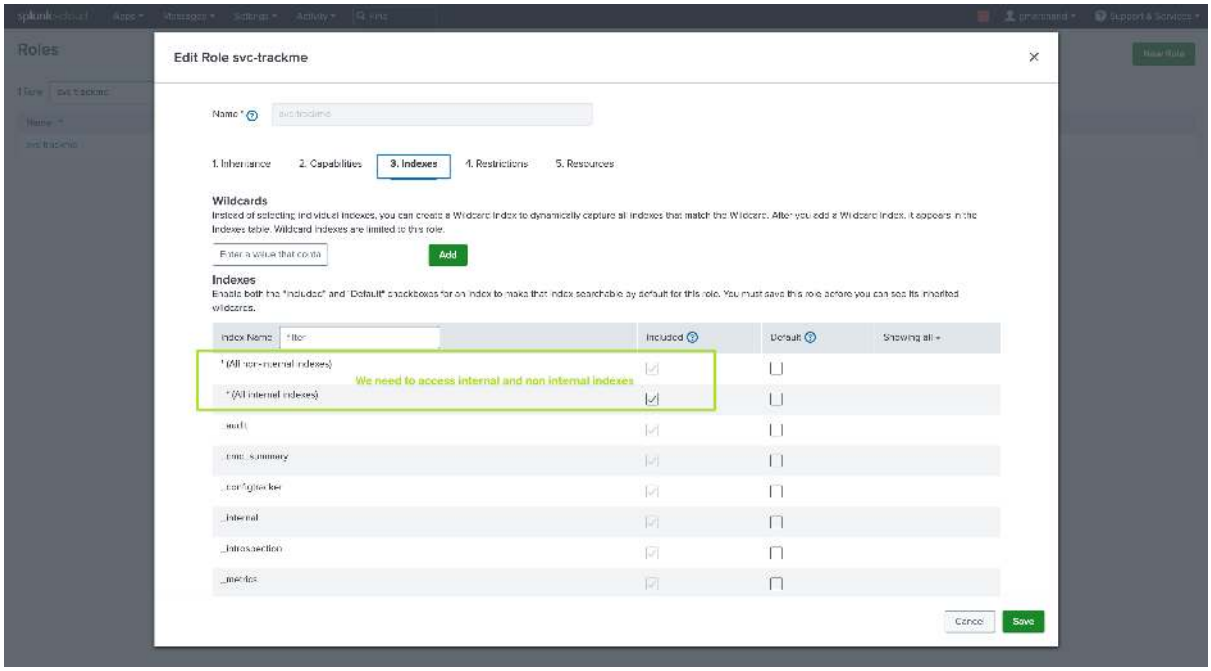
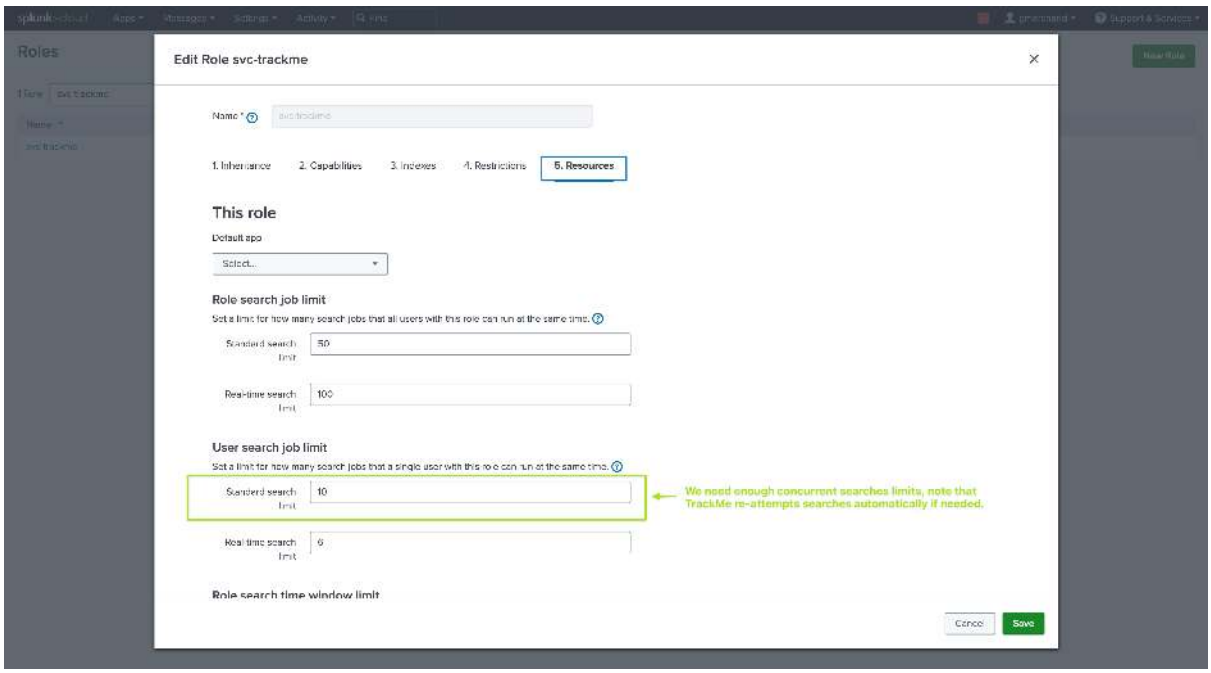
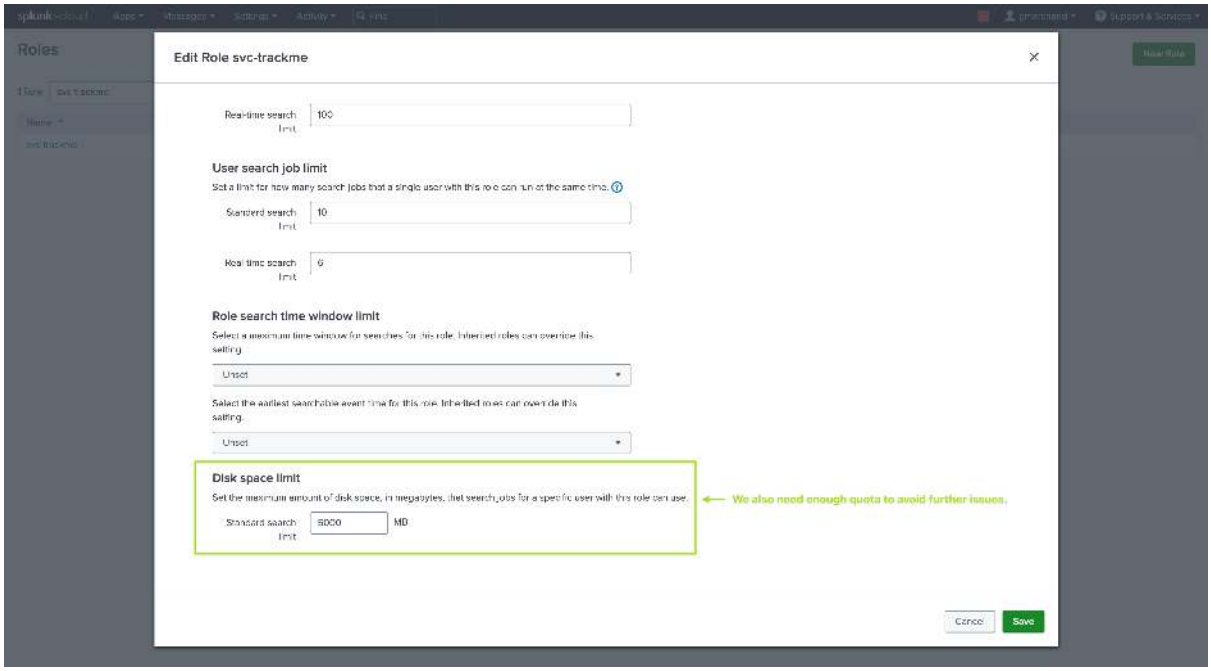
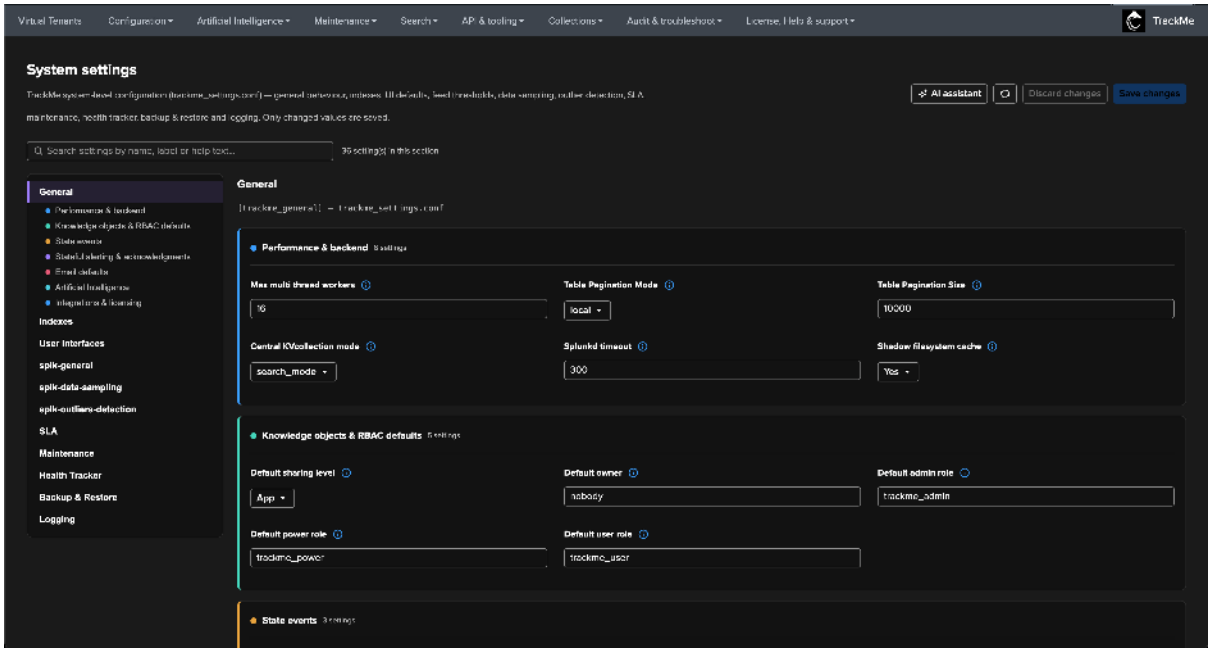
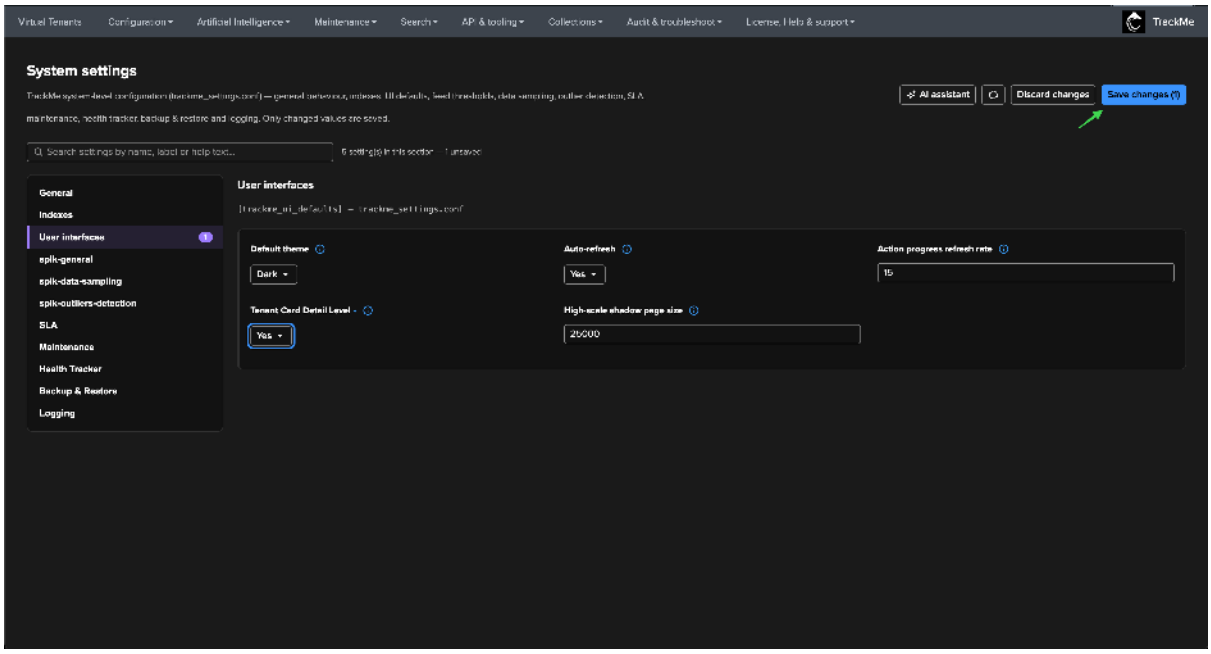
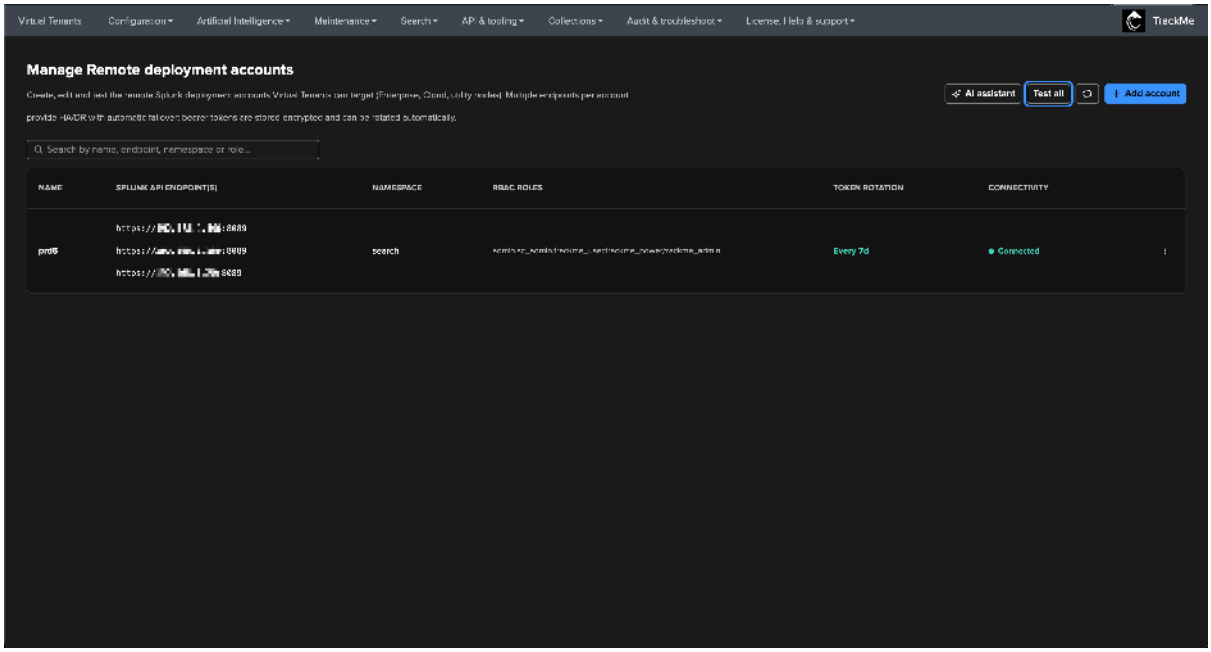
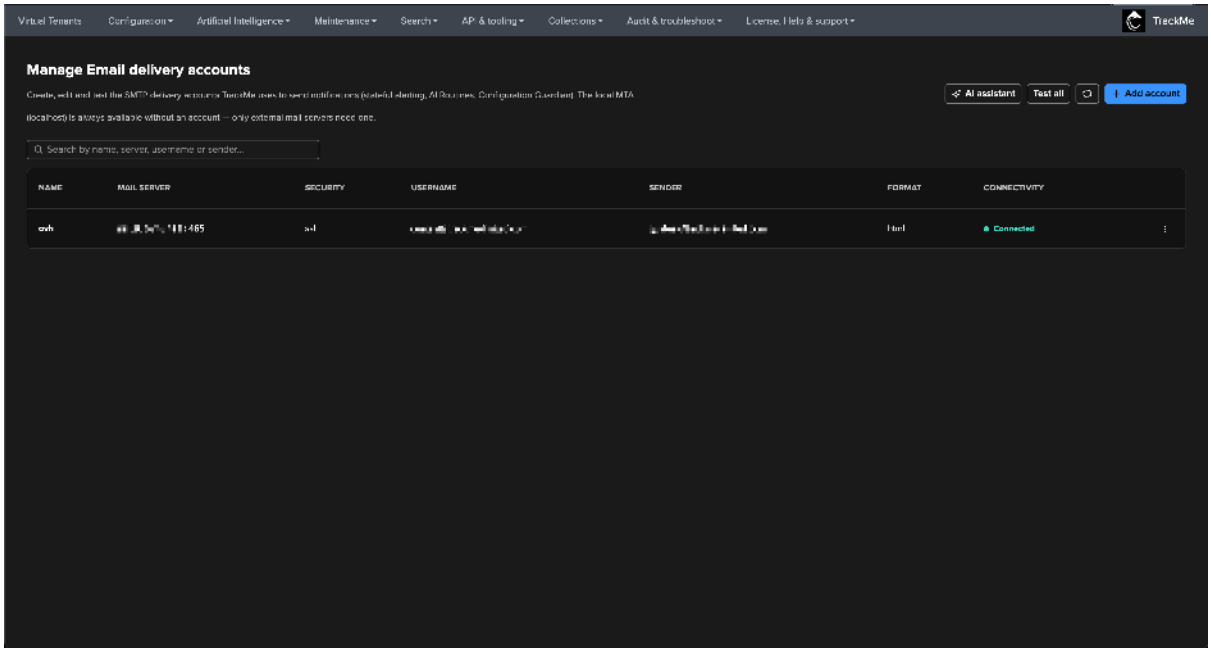
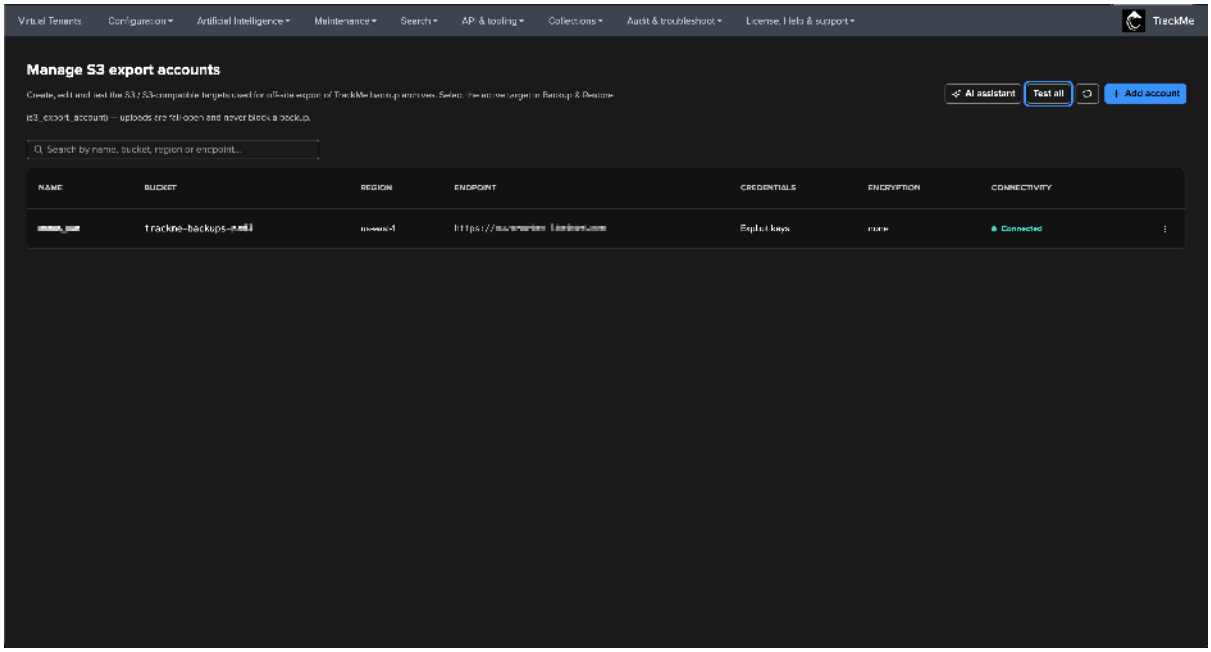
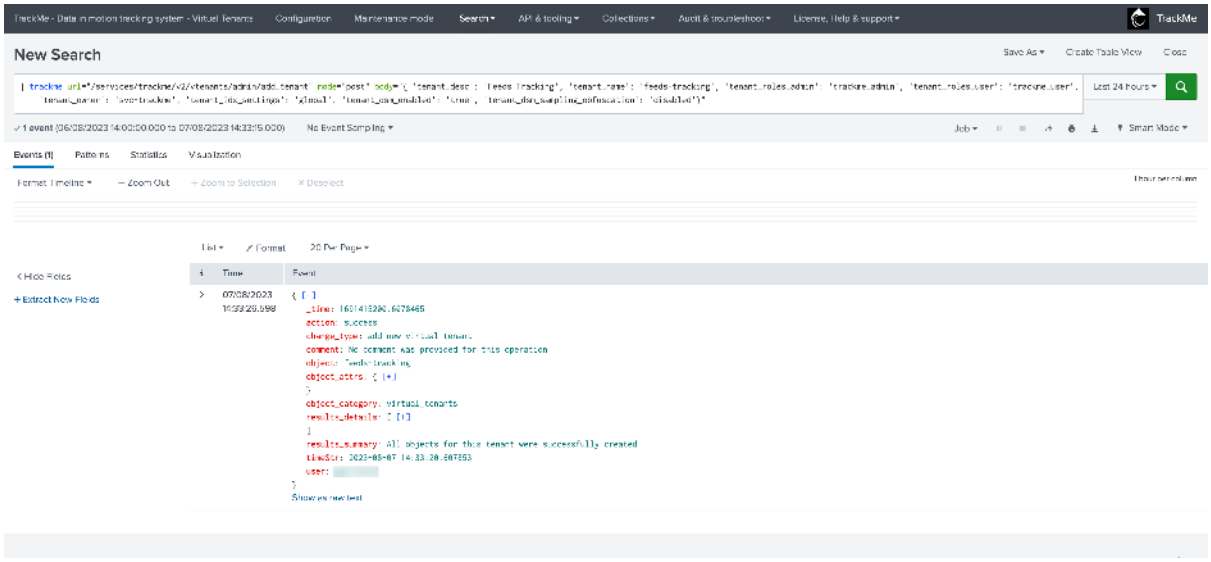
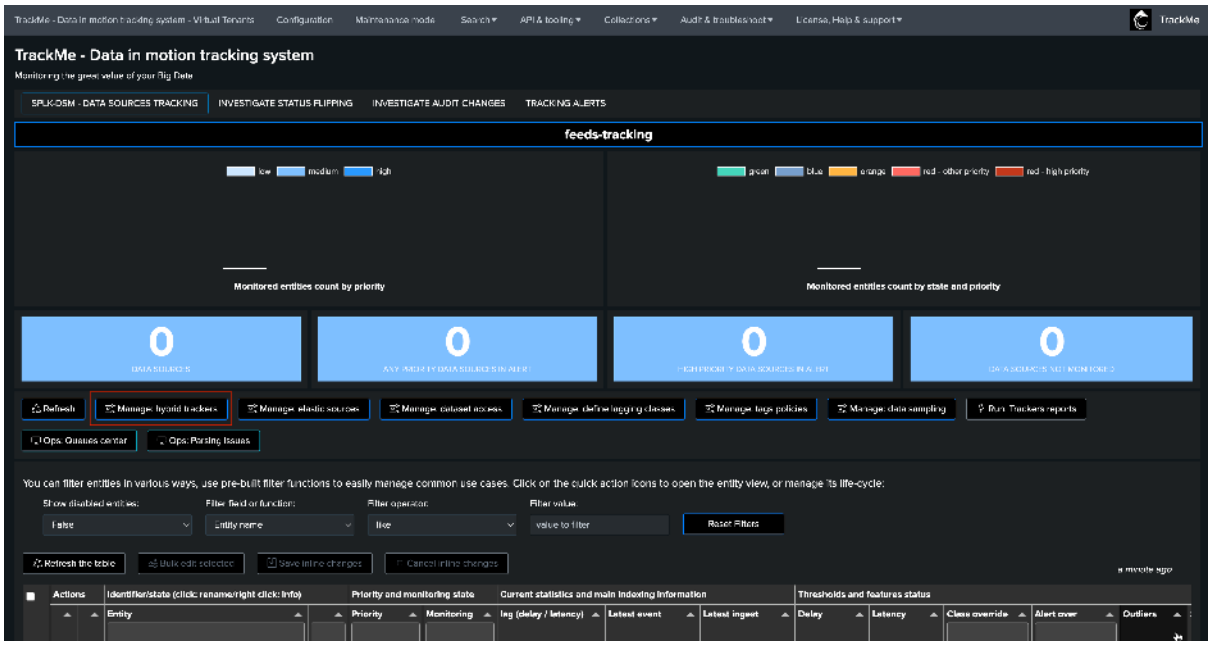
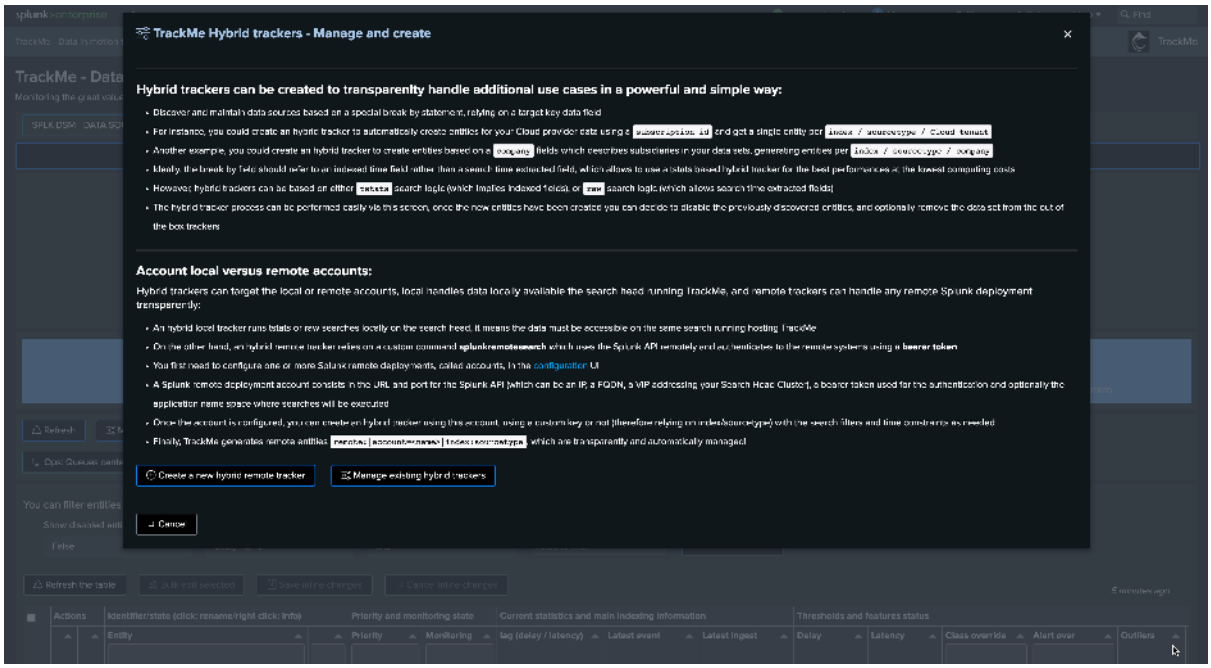
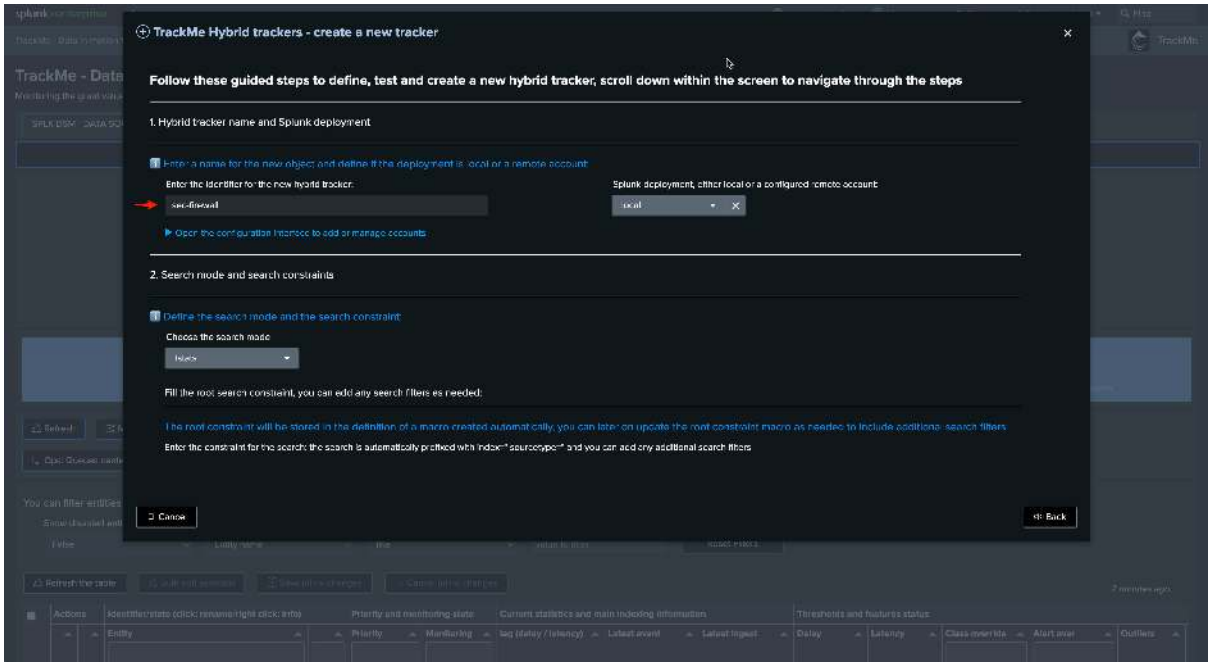
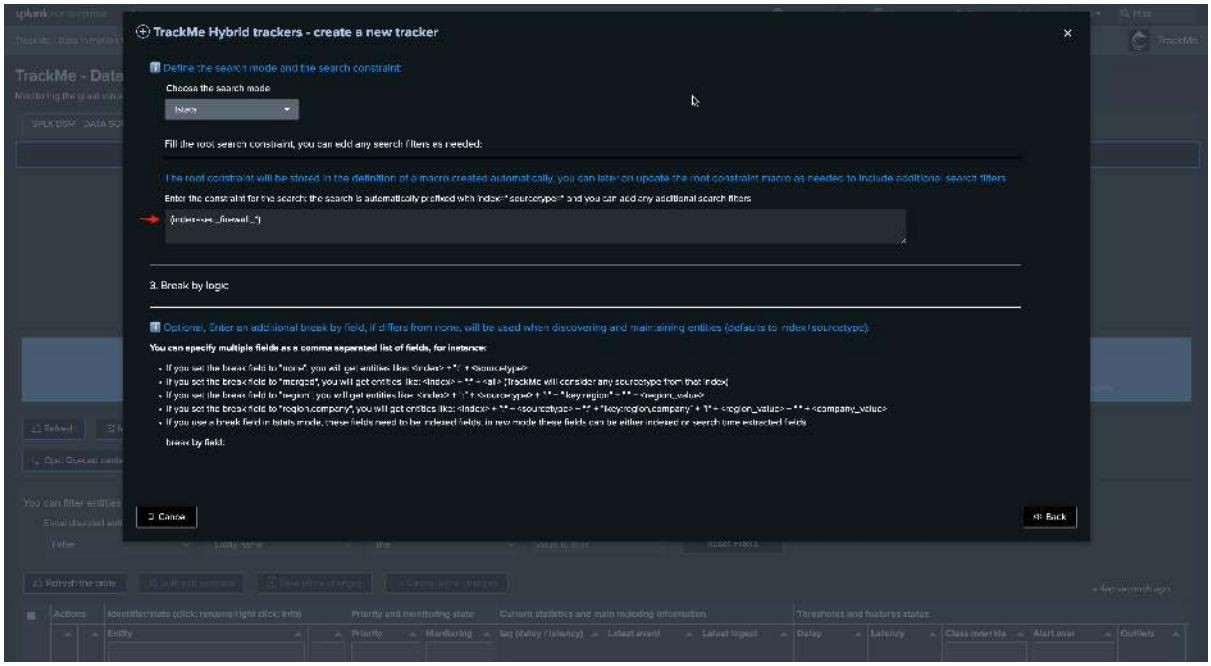
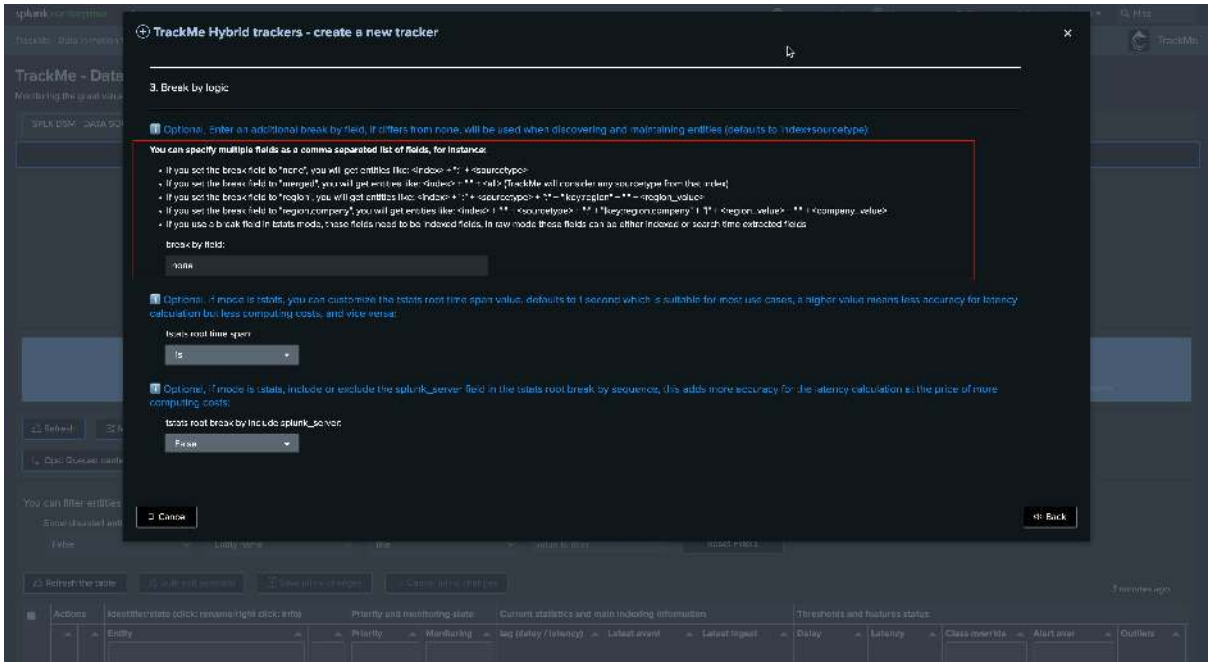
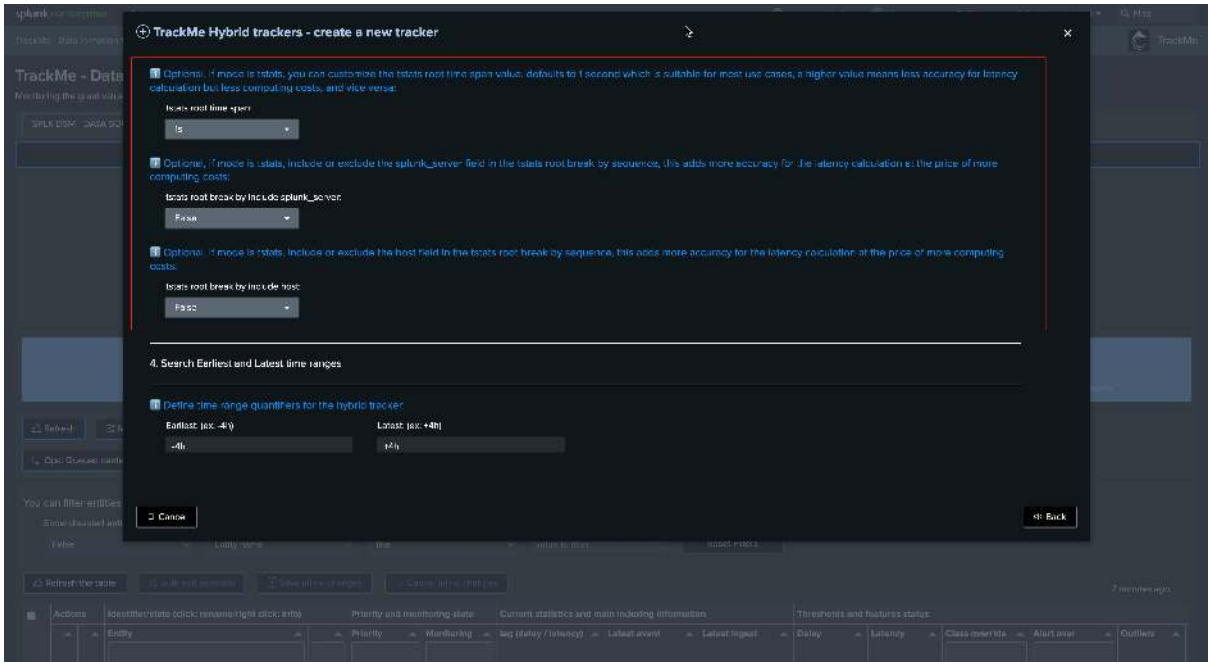
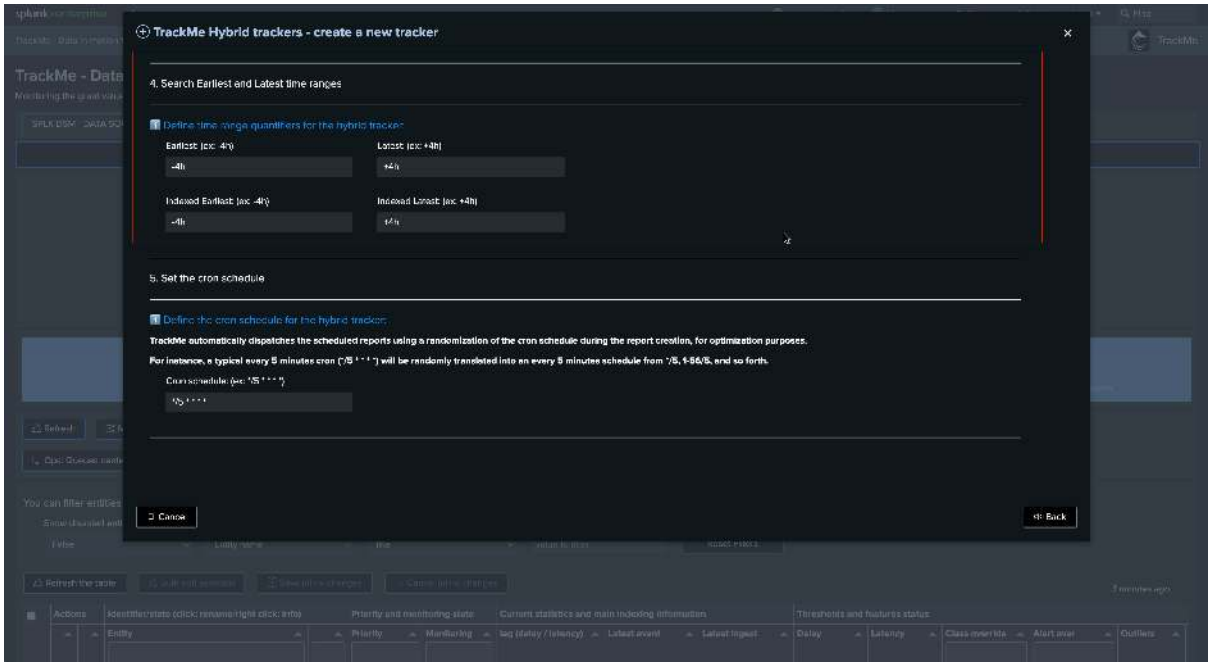
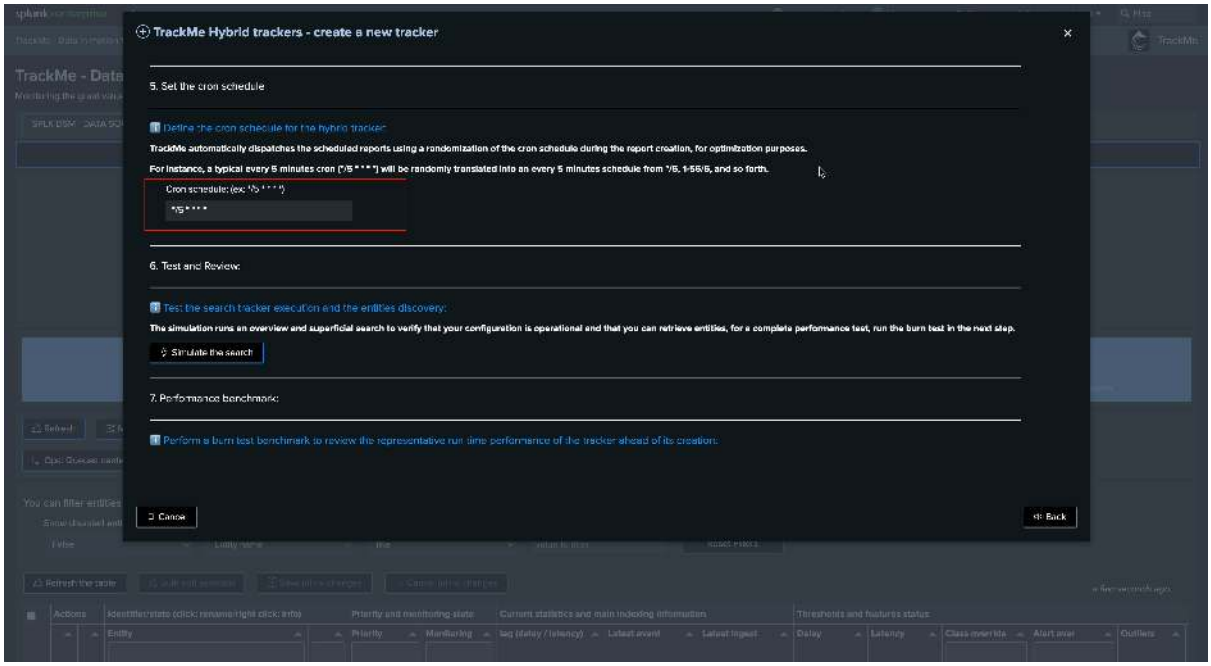
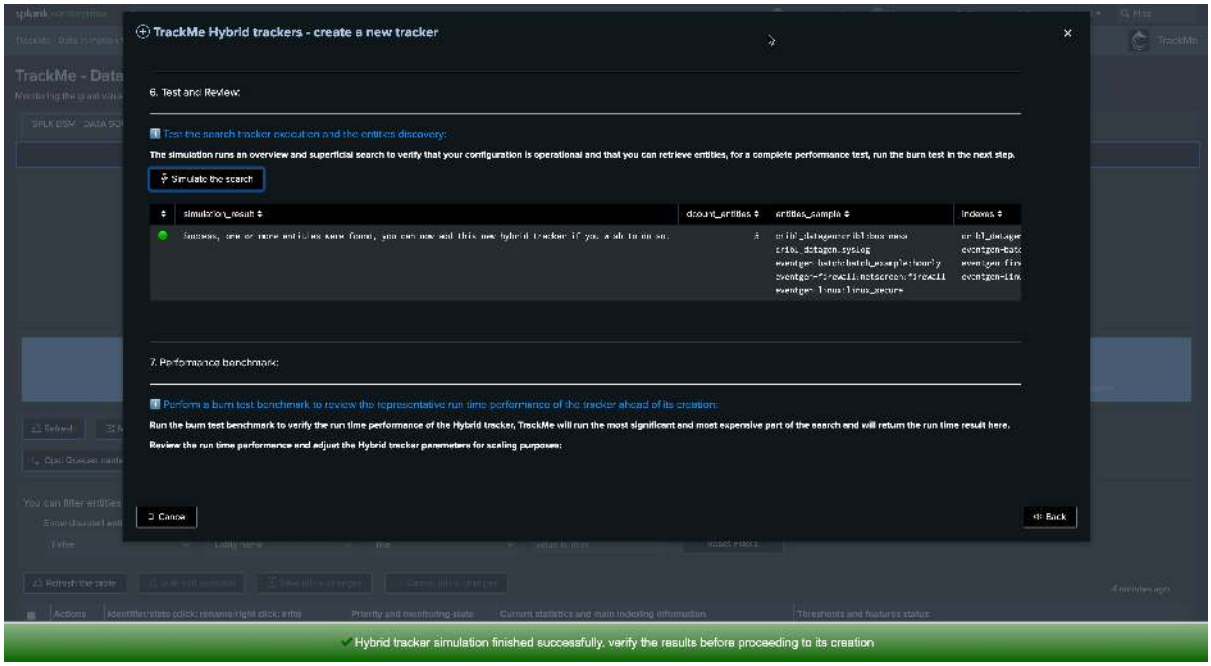
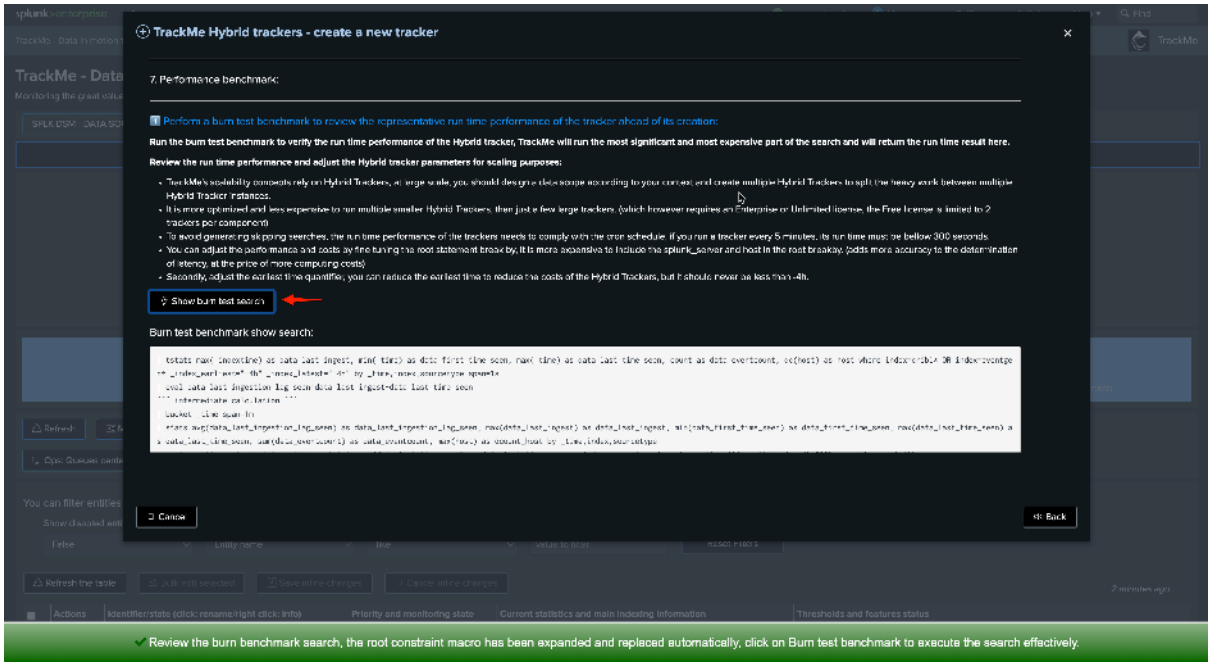
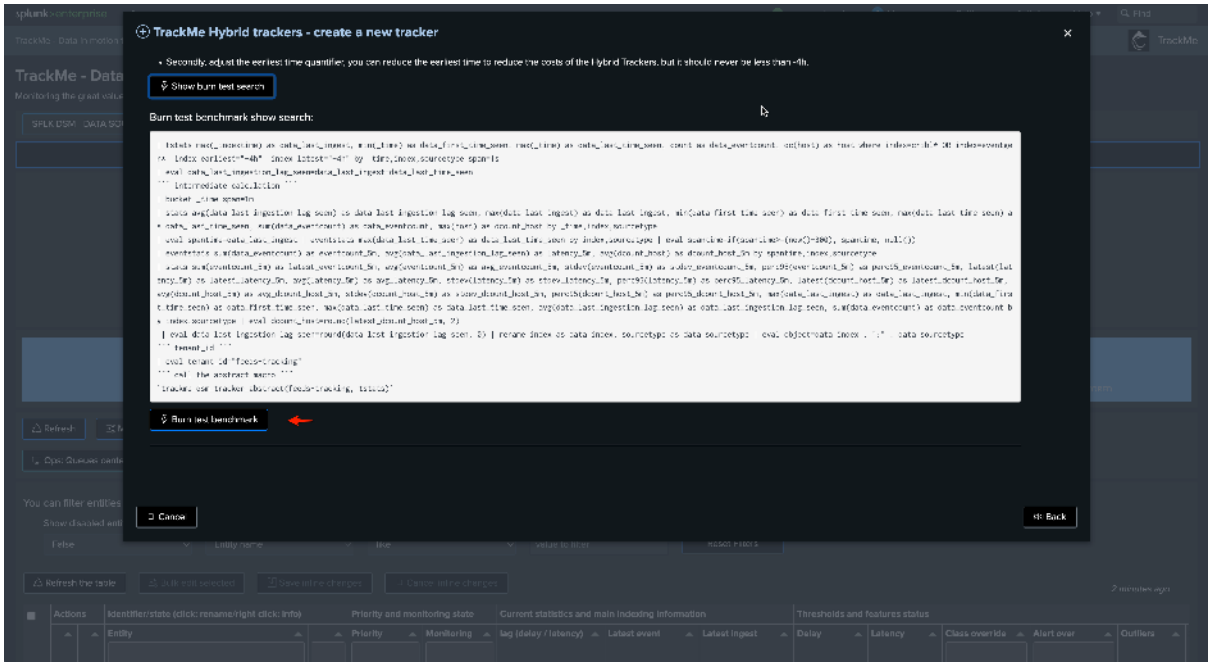
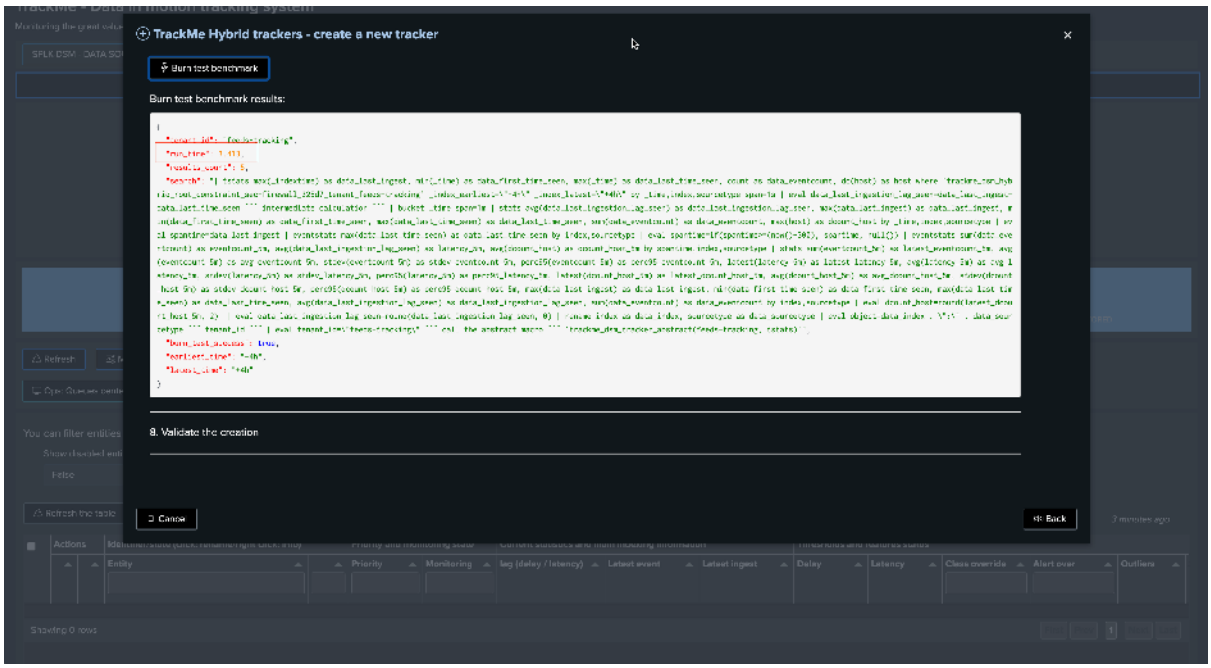
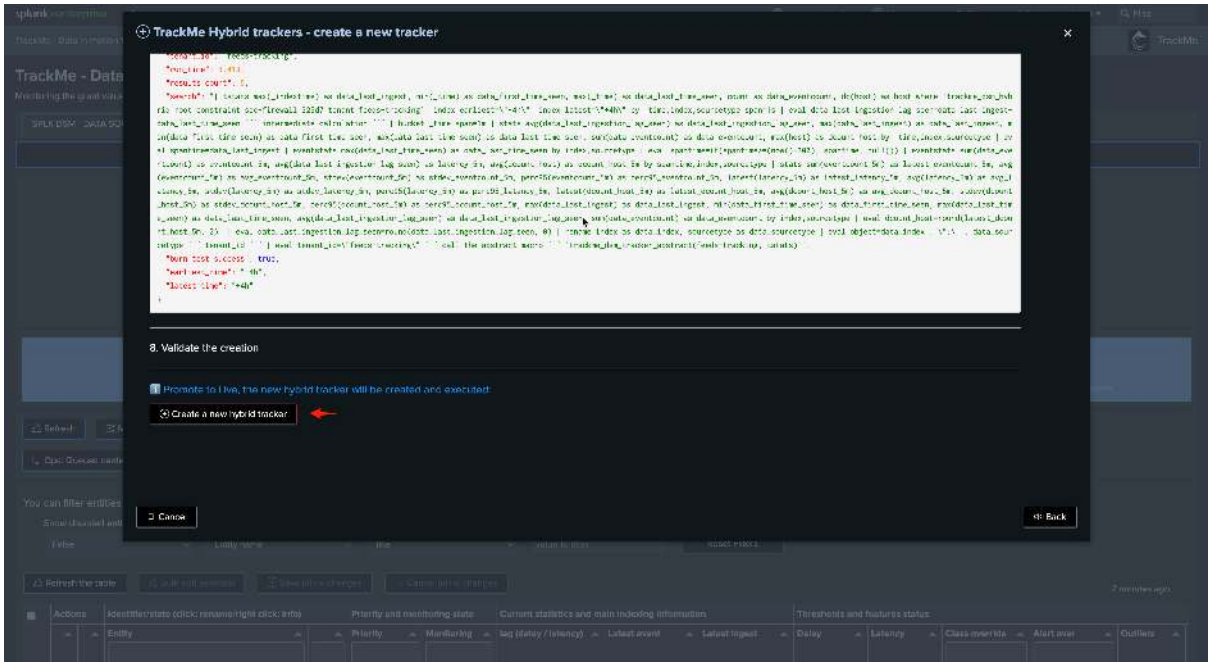
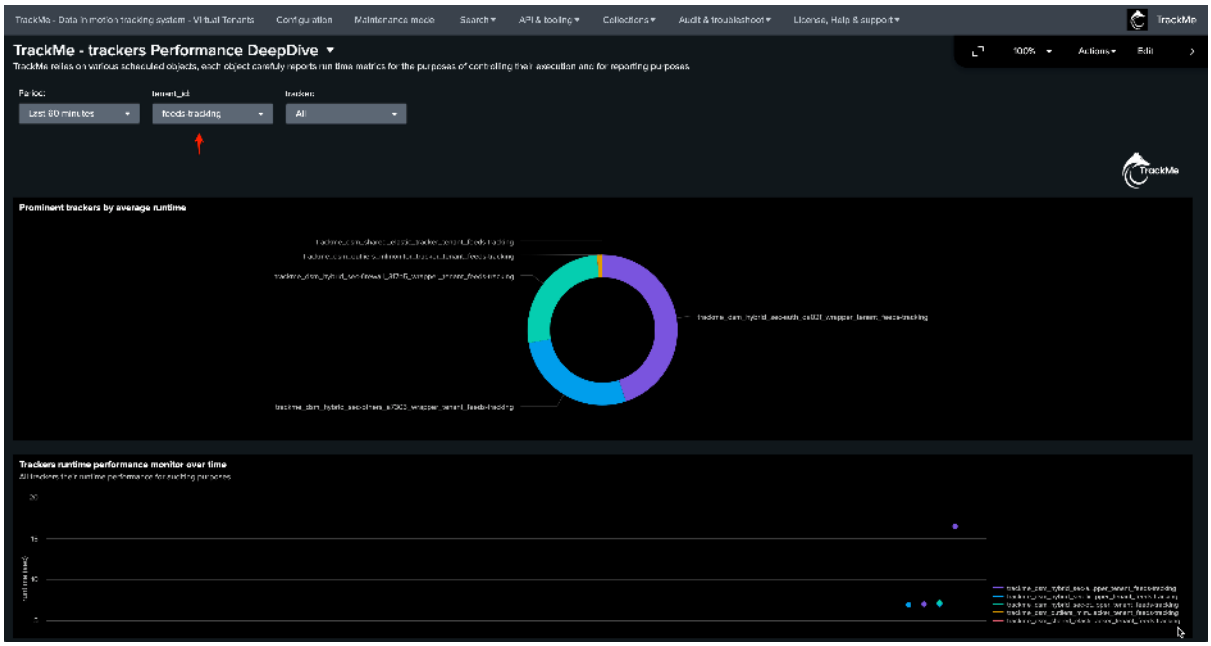
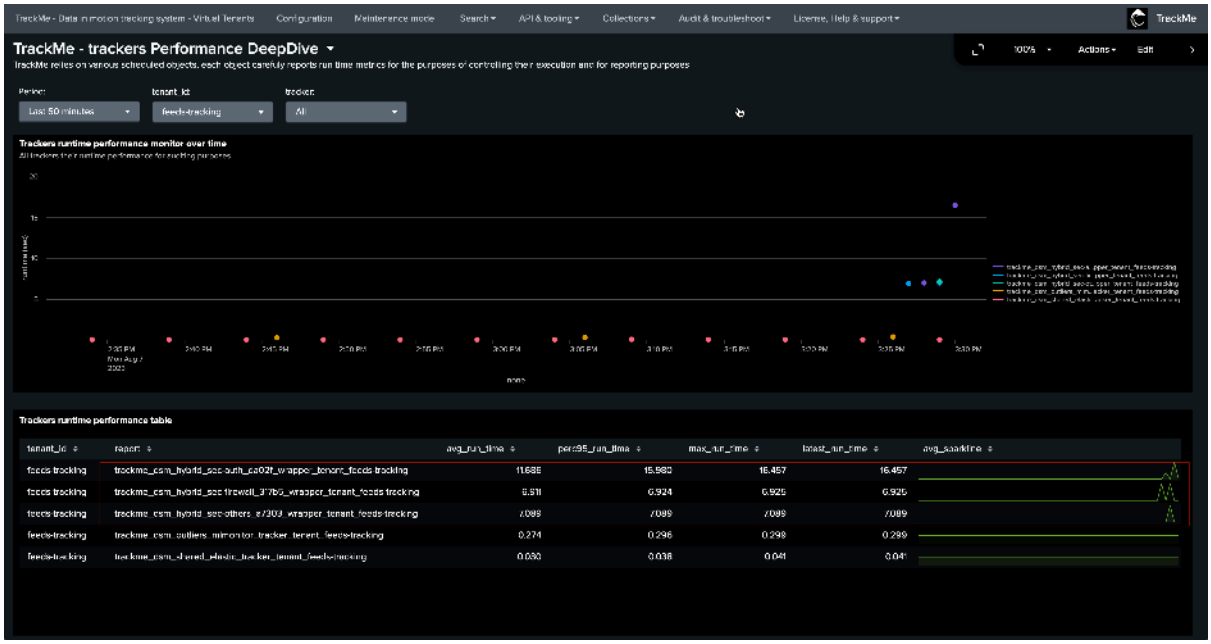
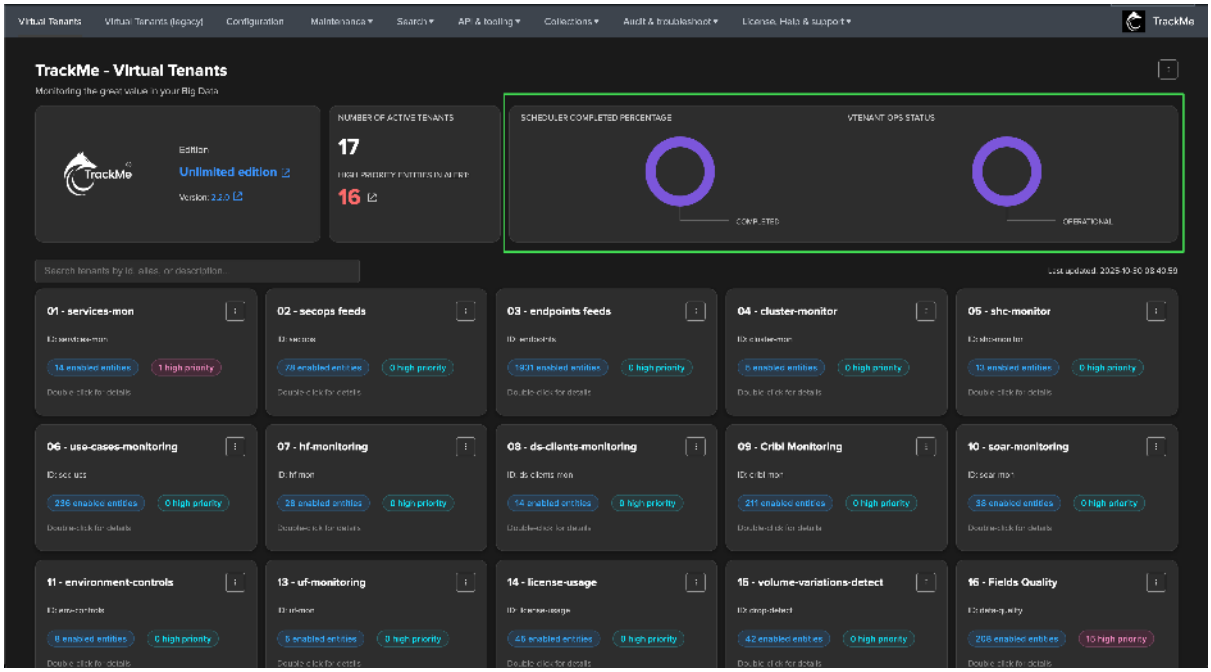
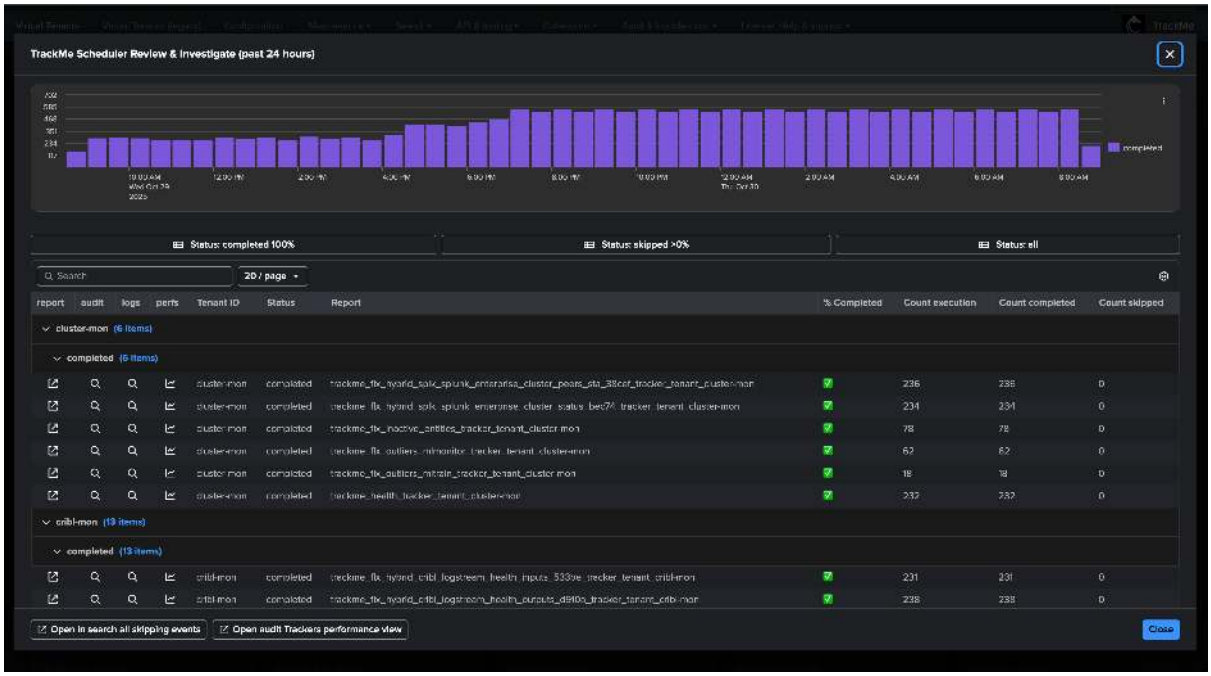
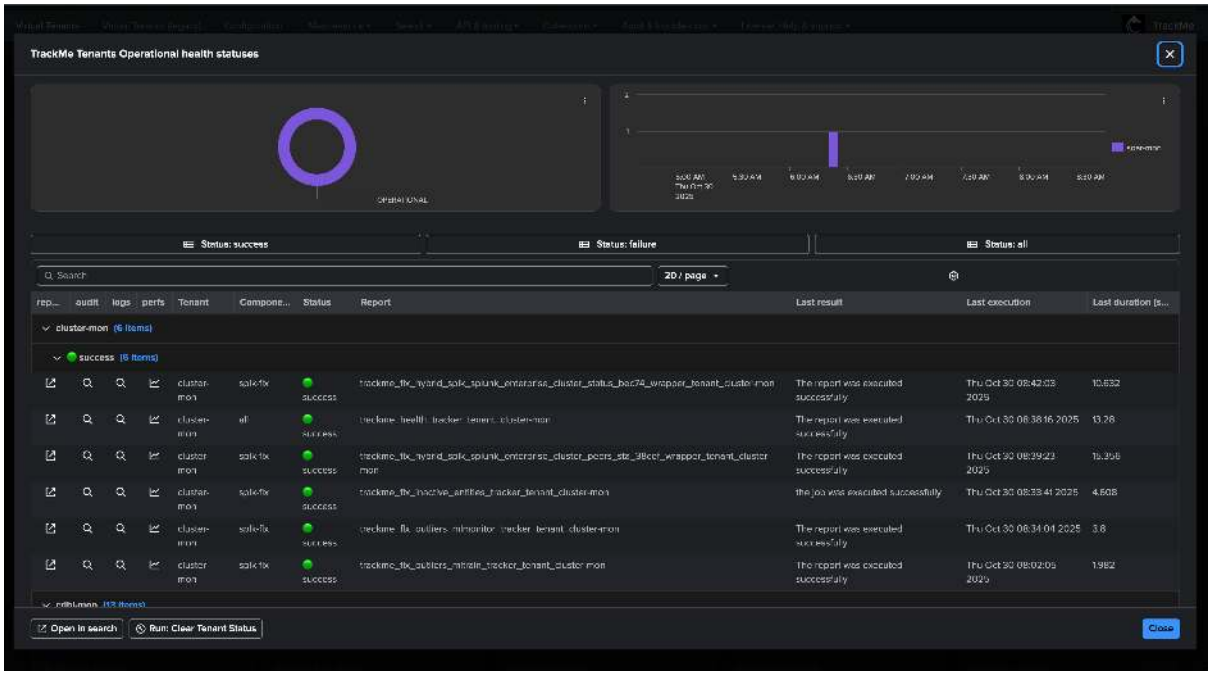
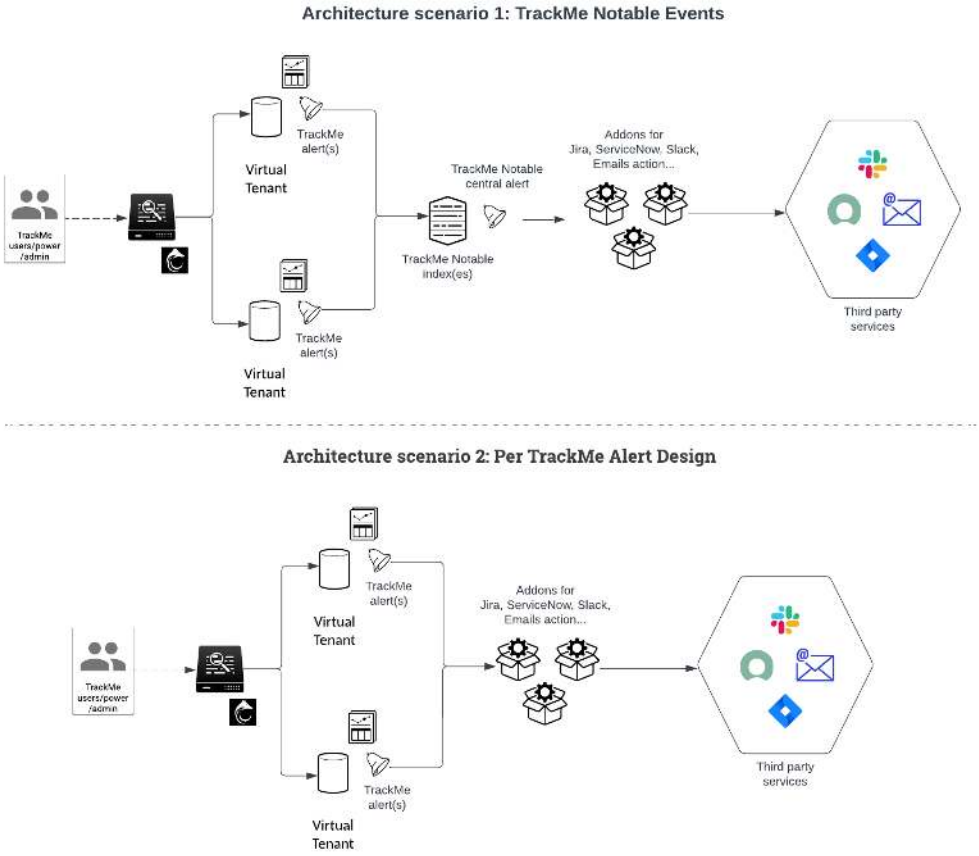
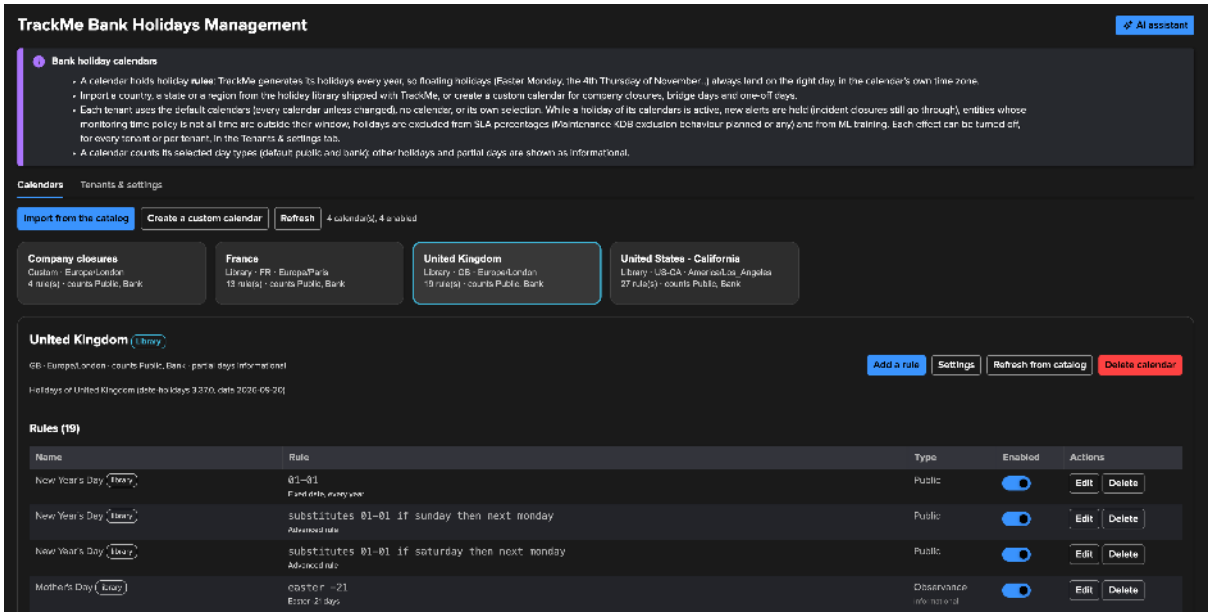
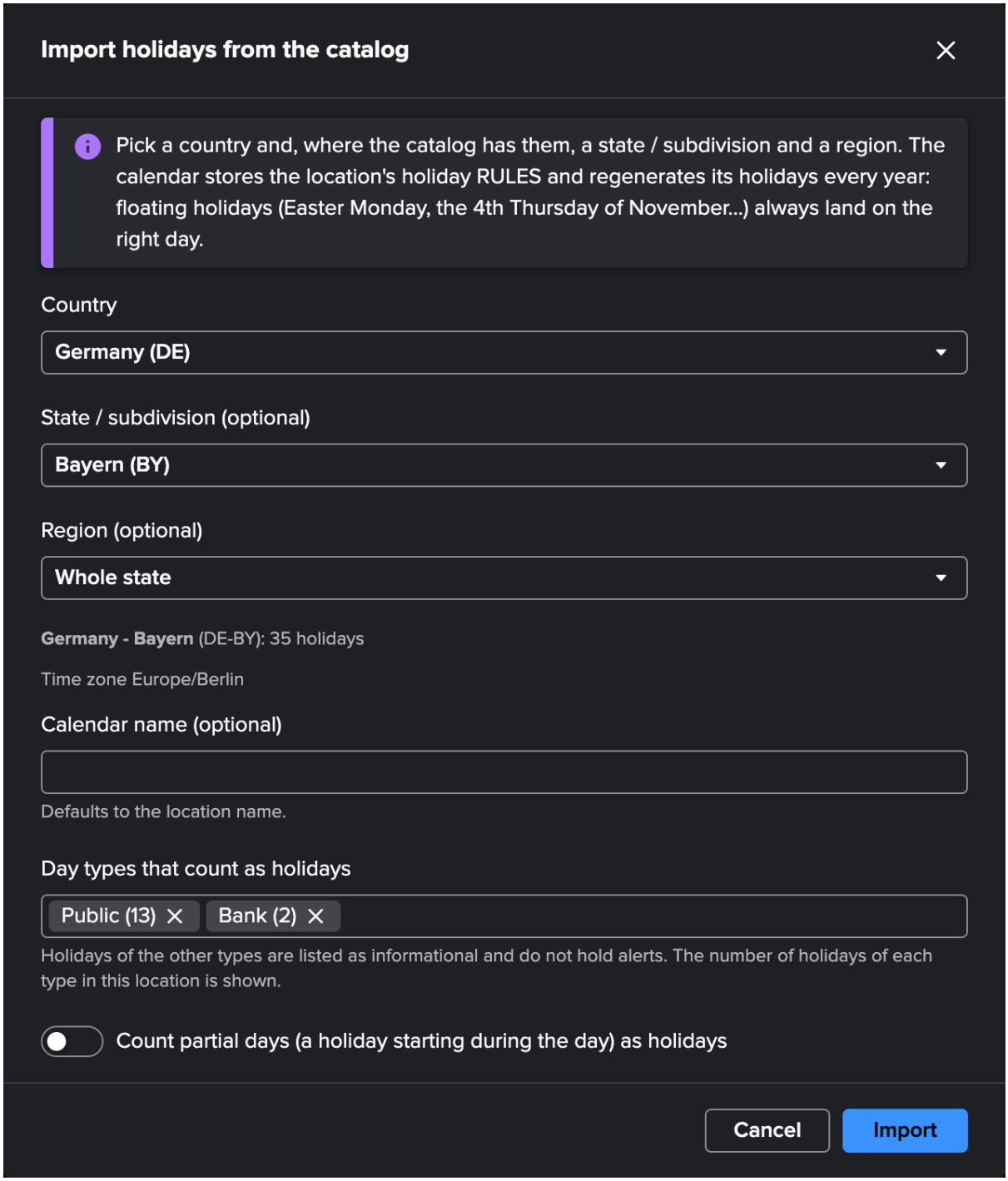
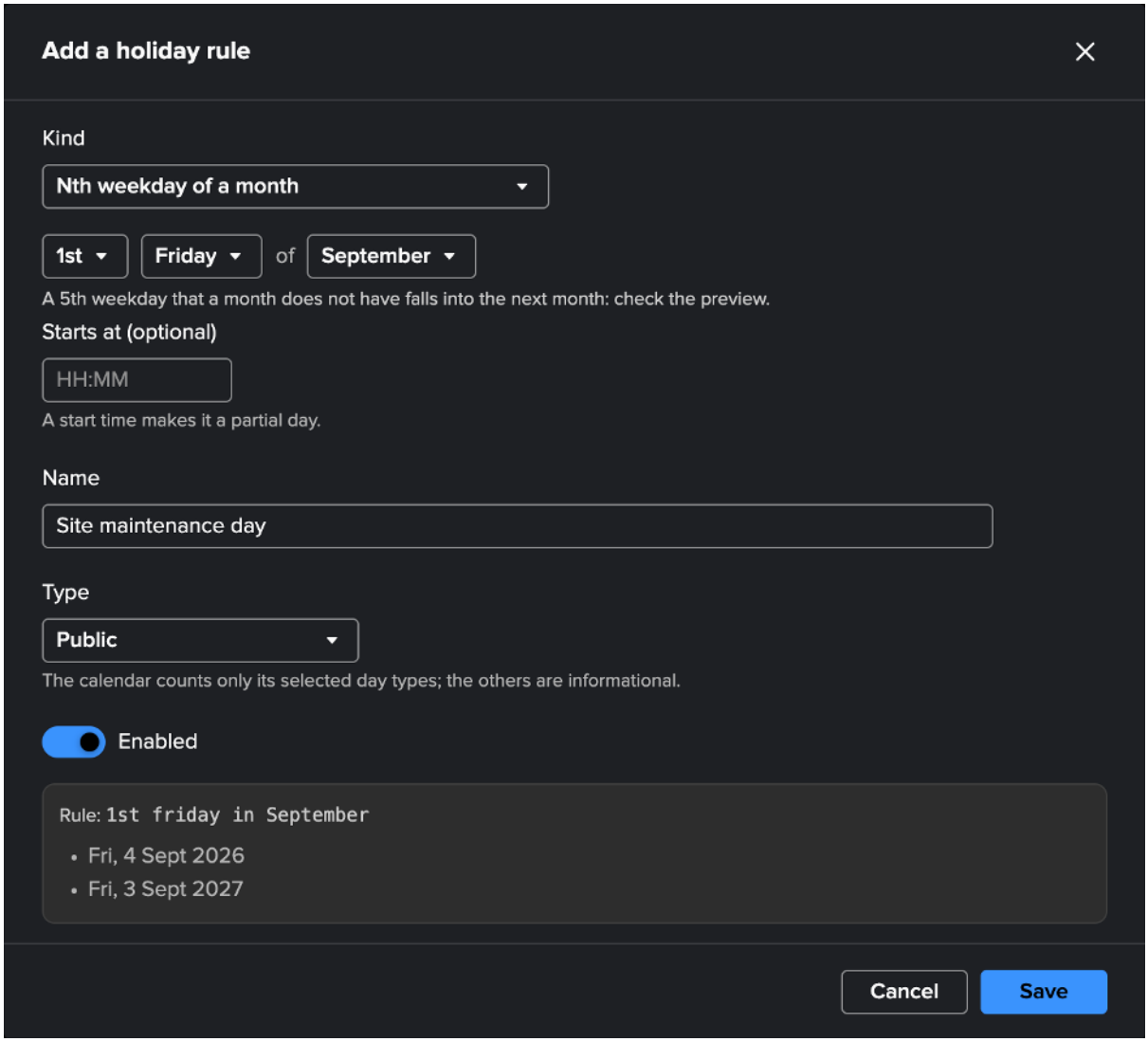
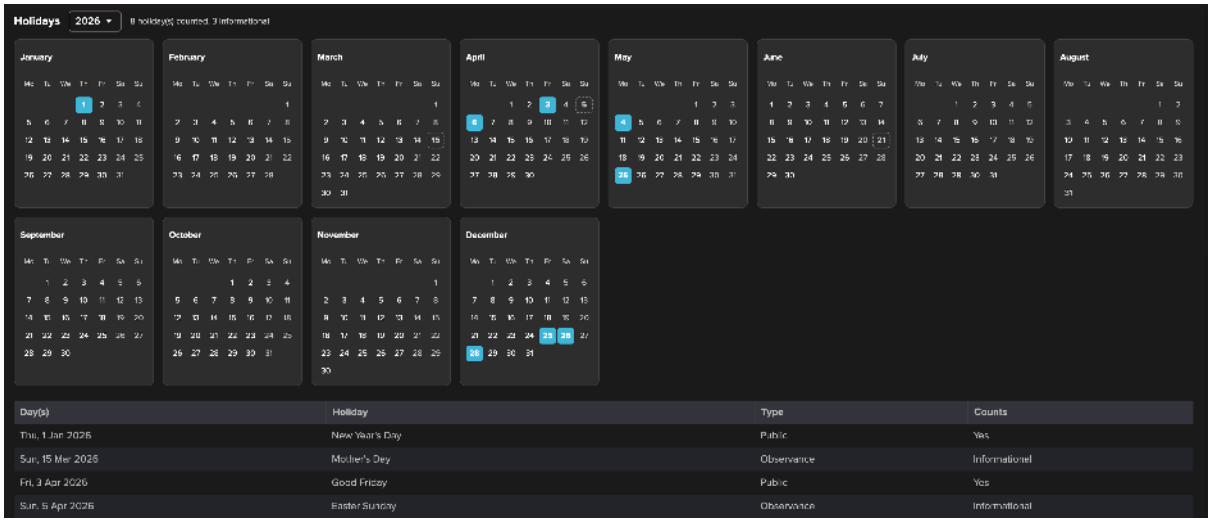
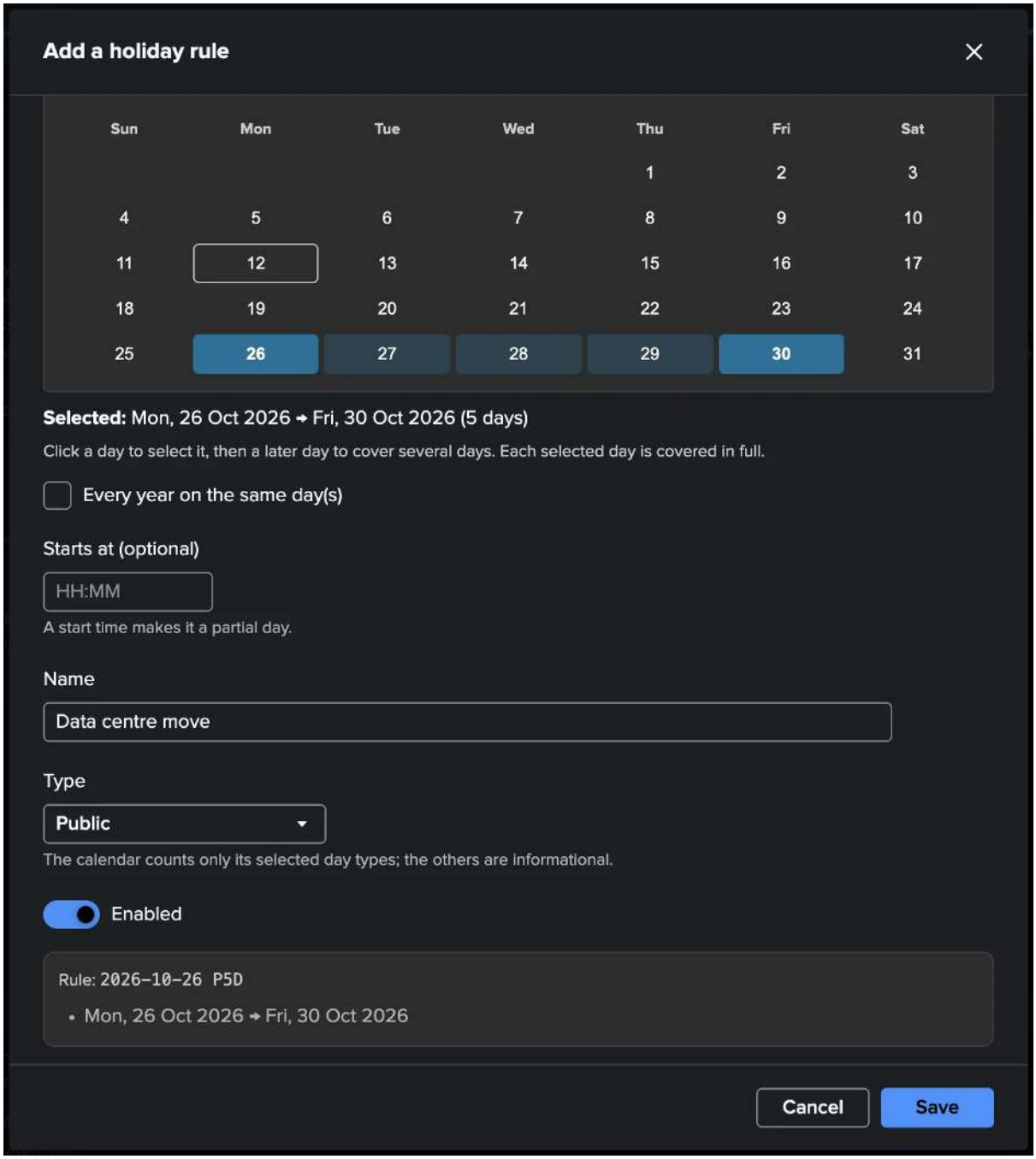
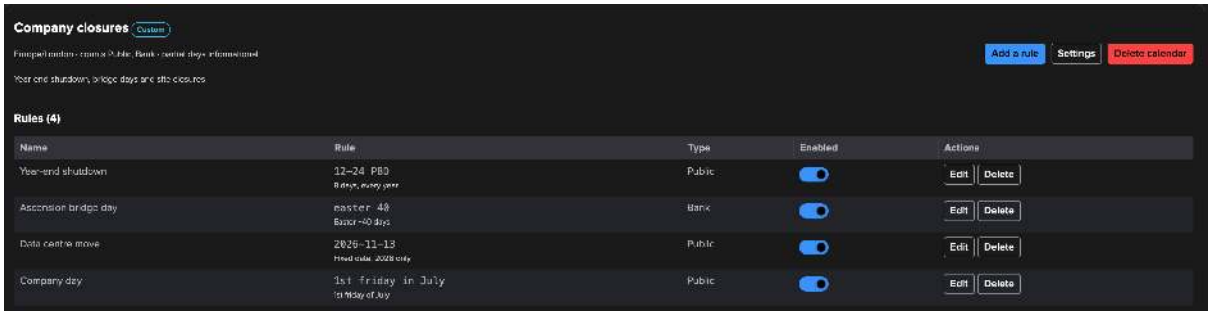
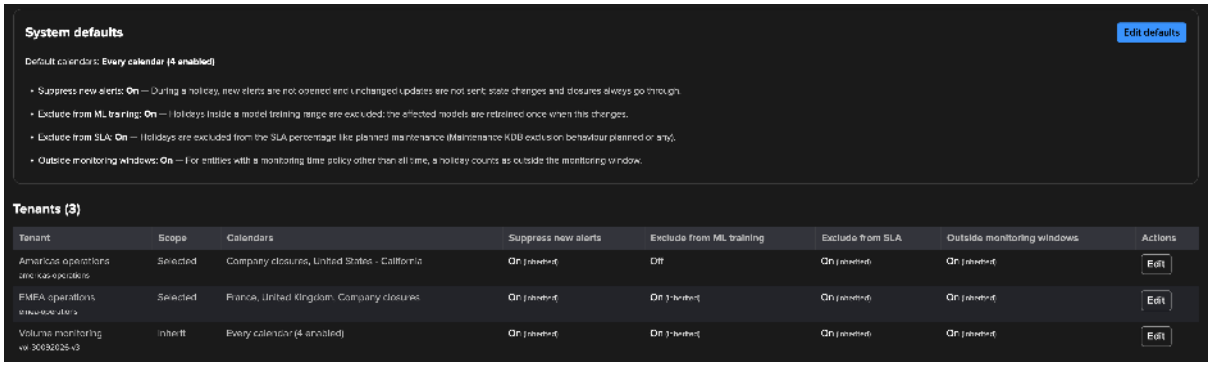
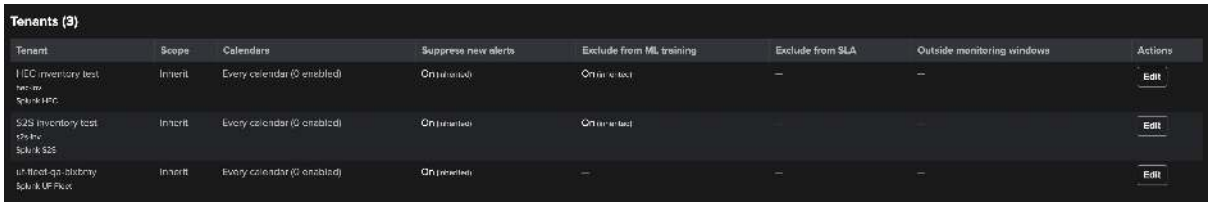
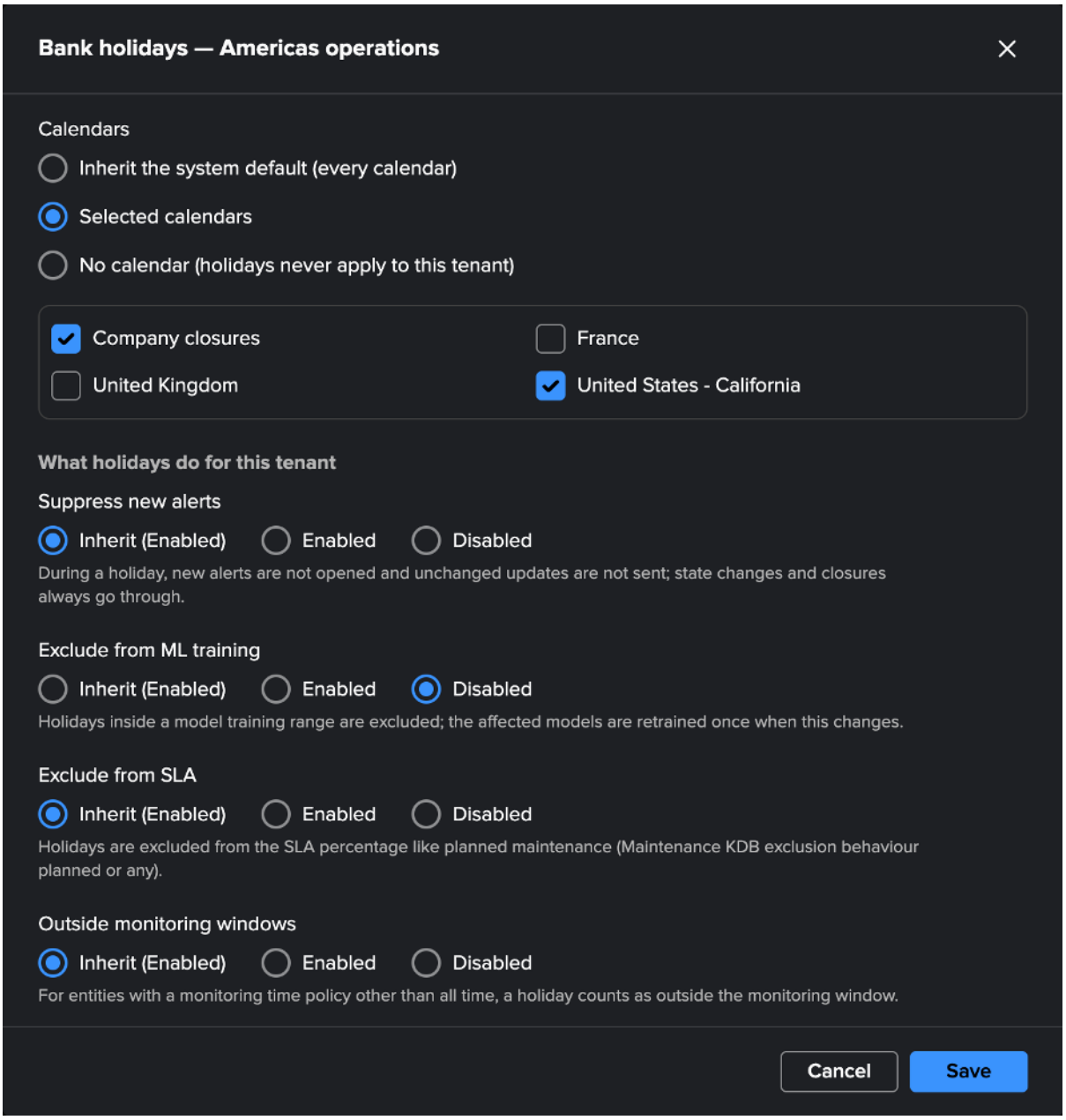
You can for instance use the lookup command to retrieve the definition any SOAR object in pure SPL:

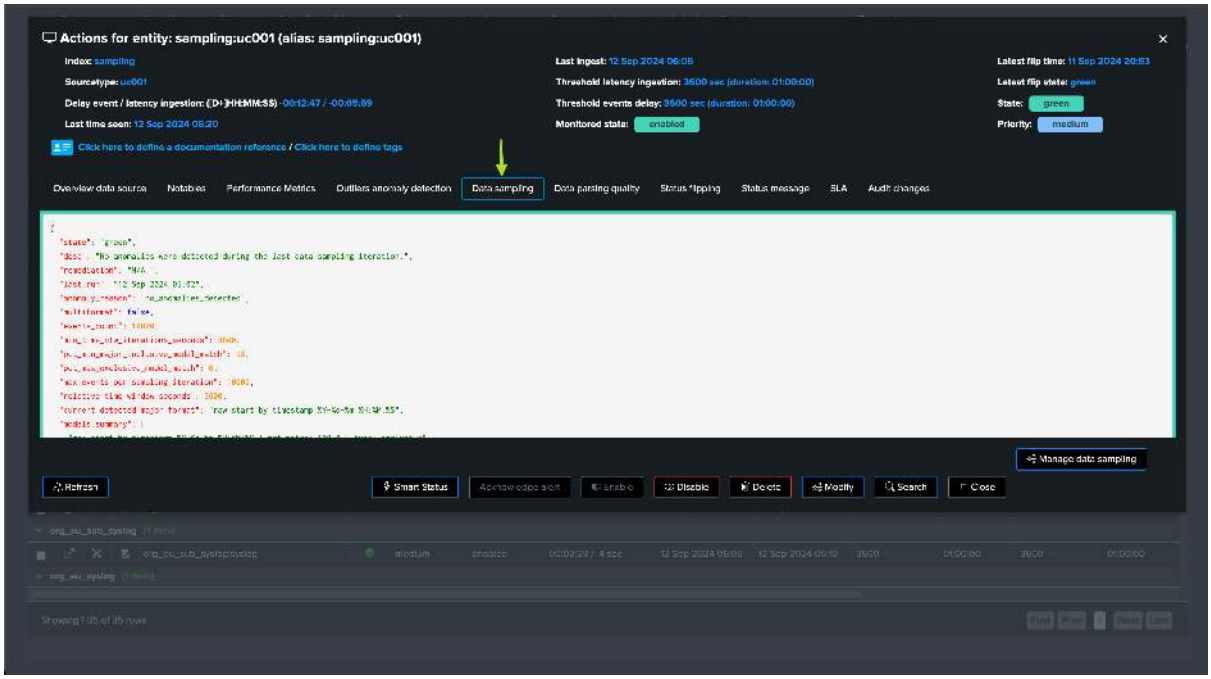
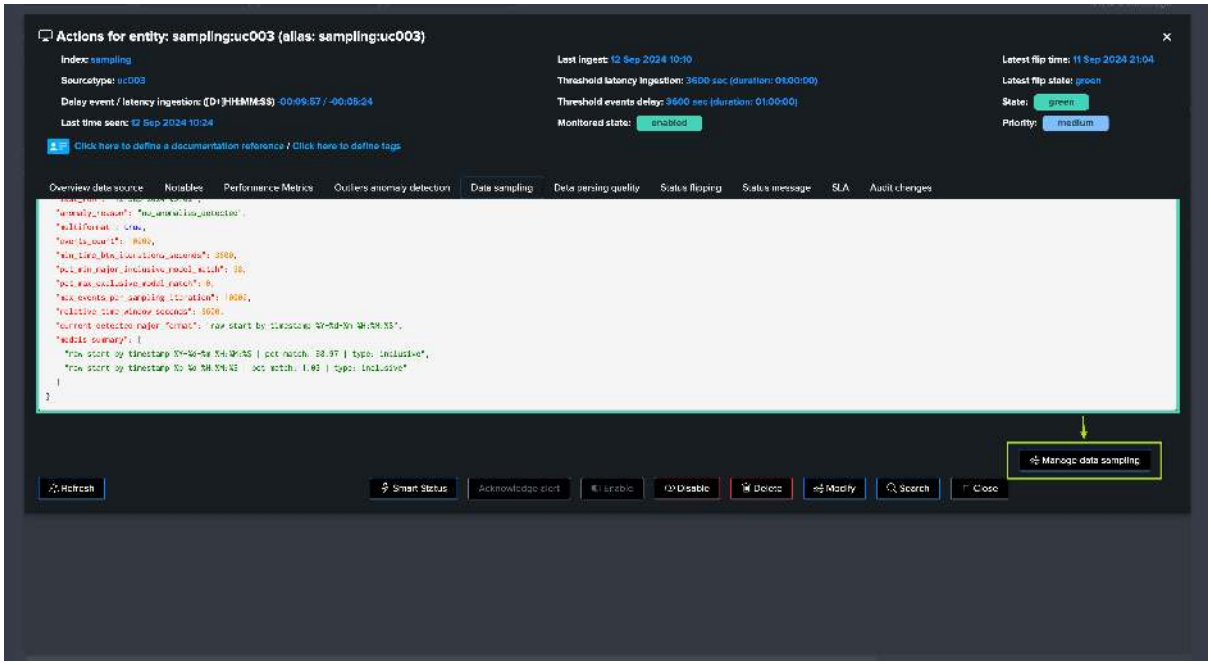
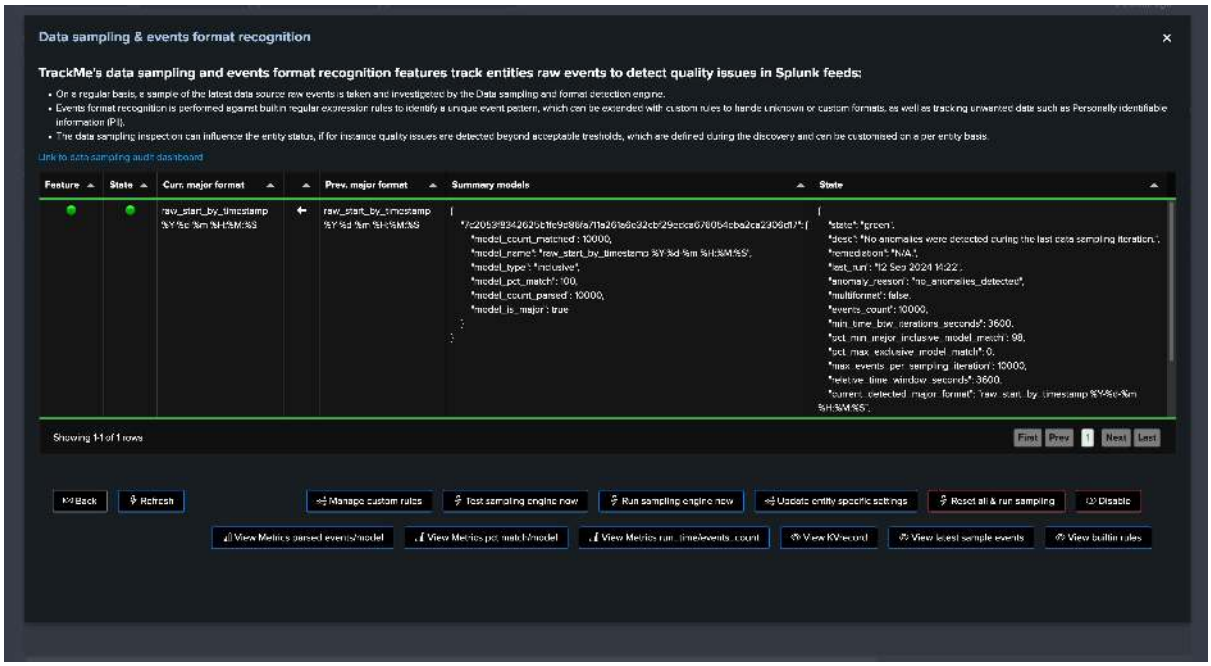
```
| makeresults | eval asset=1 | trackmesplksoarlookup soar_server=*␣
↪endpoint_target=asset source_field=asset dest_field_name=asset_name␣
↪dest_field_definition=asset_definition definition_filter_fields="*"
```



A full usage example can be to track SOAR actions and call the command to lookup associated objects metadata using their IDs:

```
index=phantom_action_run

``` extract target from the json ```
| rex field=_raw "targets\":\s(?<target_json>\[.*\])\,\s\"tenant_id\":\s"

``` rename target assets arrays ```
| rename "targets{}.assets{}" as target_assets

``` get playbook info ```
| trackmesplksoarlookup soar_server=* endpoint_target=playbook␣
↪source_field=playbook dest_field_name=playbook_name␣
↪dest_field_definition=playbook_definition␣
```

(continues on next page)
```

(continued from previous page)

```
↪definition_filter_fields="tags,active,category,comment,labels,name,metadata,playbook_type"
``` get asset info ```
| trackmesplksoarlookup soar_server=* endpoint_target=asset␣
↪source_field=target_assets dest_field_name=asset_name␣
↪dest_field_definition=asset_definition␣
↪definition_filter_fields="tags,description,name,app,type,automation_broker"
``` extract the app ```
| spath input=asset_definition path=app output=app
``` get app info ```
| trackmesplksoarlookup soar_server=* endpoint_target=app source_field=app␣
↪dest_field_name=app_name dest_field_definition=app_definition␣
↪definition_filter_fields="tags,app_version,description,name,product_name,product_vendor,publishe
``` get broker info, if any ```
| spath input=asset_definition path=automation_broker output=automation_broker
| trackmesplksoarlookup soar_server=* endpoint_target=automation_proxy␣
↪source_field=automation_broker dest_field_name=automation_broker_name␣
↪dest_field_definition=automation_broker_definition␣
↪definition_filter_fields="name,keys_rotated_time,version,last_seen_state,concurrency_limit"
```




app_definition
Reports
Values
Count
%


asset_definition
Reports
Values
Count
%



6.5 TrackMe's REST API splk-soar endpoints

Underneath, the `trackmesplksoar` command interacts with the TrackMe API endpoints for SOAR, you can find the endpoints references and their documentation in the API reference dashboard: (Menu API & Tooling)



8.9.13 TrackMe App on SOAR

The **TrackMe App on SOAR** is a dedicated Splunk SOAR app that lets SOAR playbooks and analysts call *into* TrackMe — reading entity statuses and driving TrackMe features — so TrackMe becomes an actor inside your SOAR automation. It is the reverse direction of *Splunk SOAR*.

What it does

Each app action maps to a TrackMe REST API endpoint. Typical uses include:

- retrieve the **real-time status** of TrackMe entities;
- update an entity's key **behaviour parameters**;

- update **acknowledgment** status (for example, auto-acknowledge an entity from a remediation playbook);
- interact with other TrackMe features, including the machine-learning capabilities and the AI advisors.

The app also **ingests TrackMe alert events into SOAR** natively: it polls TrackMe on a schedule and creates a container per notable or stateful alert event, de-duplicated on the event's own identifier. This is the recommended way to bring TrackMe alerts into SOAR — it needs no forwarding rule, no report and no field mapping.

You configure a **TrackMe asset** in SOAR with credentials to the TrackMe API, and the app's actions then become available to playbooks. With ingestion enabled, a single workflow can react to a TrackMe alert in SOAR and call back into TrackMe to act on it.

The app requires **Splunk SOAR 7.0.0 or later** and **Python 3.13**.

Note

The app ships with its own built-in documentation, and the exact action catalogue evolves with the app version — consult the in-SOAR documentation and the app's README for the authoritative, current list of actions.

See also

- *TrackMe App on SOAR — in depth* — compatibility, installation, the asset configuration, the action catalogue, and event ingestion.
- *Splunk SOAR* — monitoring the health of a Splunk SOAR deployment with TrackMe.
- *Operations & Administration* — the TrackMe REST API the app calls.
- *Roles & access control* — the access the SOAR asset authenticates with.

8.9.14 TrackMe App on SOAR — in depth

Tip

This is the **in-depth reference** for the TrackMe App on Splunk SOAR. For the readable overview, start with *TrackMe App on SOAR*. This page covers compatibility, installation, the asset configuration, the action catalogue, and **event ingestion** — the recommended way to bring TrackMe alerts into SOAR.

About TrackMe App on SOAR

- **TrackMe App on SOAR** is an application for **Splunk SOAR** that allows you to automate tasks and interact with TrackMe transparently from your SOAR environments.
- With the TrackMe App on SOAR, you can automate various actions in TrackMe, such as retrieving TrackMe entities' realtime statuses, updating their key behavior parameters, or interacting with TrackMe features such as the Machine Learning capabilities and the AI advisors.
- The app also **ingests TrackMe alert events into SOAR** natively, so a playbook can react to a TrackMe alert and call straight back into TrackMe to act on it.
- Working in collaboration with TrackMe, SOAR users can leverage the TrackMe App on SOAR to extend and enrich the workflow and fulfill any kind of sophisticated requirements.

Compatibility and requirements

Requirement	Details
Splunk SOAR	**7.0.0 or later** (the minimum product version declared by the app)
Python	**3.13** (the app also declares 3.14 compatibility)
TrackMe	A TrackMe deployment reachable over the Splunkd management API, normally on port **8089**
Authentication	A **Splunk bearer token** for a TrackMe service account — see *The service account and its RBAC*
Ingestion (optional)	The service account must additionally be allowed to **run searches** and to read the indexes holding TrackMe's alert events

Warning

Splunk SOAR releases from early 2026 require apps to be **Python 3.13** compatible; apps that have not been updated fail to execute. Version **2.1.0** of the TrackMe App on SOAR is the Python 3.13 release and therefore requires **SOAR 7.0.0+**. If you are running an older SOAR release, remain on the 1.x app until you can upgrade the platform.

Installation

Download the application package from:

- our release website: https://downloads.trackme-solutions.com/trackme-app-on-soar
- our GitHub repository: https://github.com/trackme-limited/trackme-app-on-soar

Splunkbase also lists a connector named **TrackMe for Splunk SOAR**, but it is currently a different build — see the warning below before installing from there.

Warning

The version on Splunkbase is not this application

Splunk maintains its own build of the TrackMe connector on Splunkbase, and that build currently carries a **2.x** version number. Despite the similar numbering it is **not** this SDK-based application: it is the earlier connector, which runs on the retired **Python 3.9** runtime and therefore **does not run on Splunk SOAR 7**.

Installing from Splunkbase today will not give you the application described on this page. Until the Splunk publication catches up, install the current release directly from our release website:

- https://downloads.trackme-solutions.com/trackme-app-on-soar/trackme_210/

or from the GitHub release:

- https://github.com/trackme-limited/trackme-app-on-soar/releases/tag/2.1.0

Both publish the release notes and a `release-sha256.txt` you can use to verify the download:

```
shasum -a 256 trackme_210.tgz
```

Install it from the SOAR UI with **Apps > Install App**, then confirm the app reports the expected version and Python runtime:



The application ships with its own built-in documentation, which is available directly in SOAR once installed and is the authoritative reference for the current action catalogue:



Note

The action catalogue evolves with the app version. The in-SOAR documentation and the README always describe the version you actually have installed.

The service account and its RBAC

The asset authenticates to TrackMe with a **Splunk bearer token**, which belongs to a Splunk user — the *service account*. Everything the app can do is bounded by what that account is allowed to do: **what the service account can do in TrackMe is what your playbooks can do.**

There are two ways to entitle it.

Option 1 — a Splunk admin account

A member of the Splunk `admin` role holds every capability, including all TrackMe capabilities, and has access to every application namespace and index. Every action works immediately, and there is nothing to configure.

Warning

This is **not recommended**. It grants the SOAR asset far more than it needs — well beyond TrackMe — and it conflicts with TrackMe's own least-privilege model. It is acceptable for a quick evaluation on a lab instance; use a scoped service account for anything that matters.

Option 2 — a dedicated service account (recommended)

Create a dedicated Splunk user for the SOAR asset and grant it a **custom role** carrying exactly what the app needs. Four things must line up — miss any one of them and actions fail with `403`, or ingestion silently returns nothing:

1. The TrackMe capability

Grant the capability matching the tier of the actions your playbooks run. The tiers are cumulative, and a higher role inherits the lower ones:

Capability	Built-in role	Needed for
`trackmeuseroperations`	`trackme_user`	Read-only actions: entity status and realtime data, get acknowledgements, list maintenance, tenant and scheduler status, get outliers models
`trackmepoweroperations`	`trackme_power`	Managing entities: acknowledge, toggle monitoring, priority and SLA, reset, delete, maintenance windows, notes and labels, outliers operations
`trackmeadminoperations`	`trackme_admin`	Administration: enable / disable tenants, run trackers, and other tenant-level operations

Most SOAR use cases are remediation-oriented, so `trackmepoweroperations` is the usual answer. Grant the admin tier only if your playbooks genuinely drive tenant-level operations.

2. The tenant's RBAC lists

The capability alone is not enough. TrackMe additionally filters by **tenant scope**: a request only sees the tenants whose configured role lists intersect the caller's Splunk roles. Your custom role must therefore be listed in the **admin / power / user role lists of each tenant** the SOAR asset is expected to reach.

Important

This is the step most often missed. If **Test Connectivity** succeeds but actions return nothing — or a tenant appears not to exist — the capability is present but the custom role is not listed in that tenant's RBAC configuration.

3. Access to the TrackMe application namespace

The app calls the TrackMe REST endpoints, which are served from the TrackMe application namespace. The service account's role must be able to reach the `trackme` app.

4. Access to the indexes (ingestion only)

Ingestion reads TrackMe's alert events from Splunk, so the service account additionally needs to be **allowed to run searches** and to **read the indexes** holding those events:

Event type	Default index	Sourcetype
`notable`	`trackme_notable`	`trackme:notable`
`stateful`	`trackme_summary`	`trackme:stateful_alerts`

Add these to the role's **allowed indexes**, and include them in the role's *default* indexes searched if you want the searches to work without an explicit index scope.

Note

The app's ingestion searches the **default** index names shown above. If a tenant writes its notable events to a custom index, those events are not ingested — align the tenant on the default notable index if you intend to ingest it. This applies to ingestion only; the action catalogue goes through the REST API and is unaffected.

Warning

The inheritance pitfall. If your custom role *inherits* a built-in TrackMe role rather than holding the capability natively, and that built-in role is still listed in a tenant's RBAC, the service account gains access to **every tenant left at default** — the opposite of what a scoped service account is for. Either make the TrackMe capability **native** to the custom role, or remove the built-in roles from each tenant's RBAC. See *Roles & access control*.

See also

- *Roles & access control* — the roles, capabilities and tenant-scope model.
- *Roles & access control — in depth* — the full capability matrix and a worked per-team example.
- Service account and permissions — the configuration details.

Configuring a TrackMe asset

Create the asset with **Configure New Asset**, then complete the **Asset Settings**:



Setting	Description
`Splunk API URL`	The Splunkd management URL of your TrackMe deployment, for example `https://mysplunk.mydomain.com:8089`
`Splunk bearer token`	The bearer token of the TrackMe service account (stored as a sensitive value)
`Verify Splunk API SSL certificate`	Whether to verify the Splunkd SSL certificate. Disabled by default; enable it when Splunkd presents a certificate trusted by SOAR
`Per-request timeout`	Timeout in seconds applied to each TrackMe REST call (default `300`)

Validating connectivity

Use **Test Connectivity** to validate the URL, the token and the network path before running any other action:



Hint

If **Test Connectivity** fails, the message distinguishes the two common causes: an authentication or capability problem (the token is invalid, or the service account lacks the TrackMe capability — see *The service account and its RBAC*) versus a transport problem (the URL, the port, or the certificate).

Actions overview

Each action maps to a TrackMe REST API endpoint. The app exposes actions across the following areas:

- **Acknowledgements** — get and manage acknowledgements.
- **Maintenance** — global maintenance mode, and per-entity maintenance windows (set, clear, list).
- **Entities** — realtime data and full status, toggle monitoring, update priority and SLA class, reset, delete, manage data sampling, set false positive, manual score influence, notes and labels.
- **Machine Learning Outliers** — get, train, monitor and reset models, exclusion periods, false positives, and bulk actions.
- **Logical groups** — get associations, manage groups, update members.
- **Tenants and operations** — enable and disable tenants, run trackers, components and scheduler status, alerts, blocklist, disruption tuning.
- **AI advisors and routines** — asynchronous AI component-health advisors and on-demand AI routines.



> **Note**
>
> Several action names use the original `snake_case` form (`ack_get`, `component_manage_entity`, and so on). These are the names the app has always shipped, and they are preserved deliberately so that existing playbooks keep working across upgrades — Splunk SOAR resolves `phantom.act()` on the action name.

Ingesting TrackMe alert events into SOAR

The recommended way to bring TrackMe alerts into SOAR is the app's built-in ingestion. The app polls TrackMe on a schedule and creates a SOAR container per alert event — no forwarding rule, no report, and no field mapping to maintain.

Why ingestion replaces the previous forwarding approach

Earlier versions of this documentation described pushing TrackMe Notable Events into SOAR with the *Splunk App for SOAR Export*: a globally shared report, an Event Forwarding Rule, a manually maintained field mapping, and a subsearch to avoid duplicates. **That approach is superseded.** Native ingestion is:

- **Built into the app** — one asset to configure, nothing to deploy or maintain on the Splunk side, and no dependency on the Splunk App for SOAR Export.
- **Complete** — every field of the TrackMe event reaches the artifact, so there is no field mapping to curate and no fields silently missing when TrackMe adds them.
- **Safe by design** — de-duplication is guaranteed on the event's own `event_id`, rather than relying on a subsearch against previously forwarded artifacts.
- **Broader** — both **notable** and **stateful** alert events are supported, where the forwarding recipe only covered notables.

If you already run the forwarding rule, see *Migrating from the SOAR Export forwarding rule* below.

What gets ingested

TrackMe produces two families of alert events, and the app can ingest either or both:

Event type	Description
`notable`	*TrackMe Notable Events* — the notable-style alert surface.
`stateful`	*TrackMe Stateful Alerts* — the `opened` / `ongoing` / `closed` alerting path.

Every event carries a unique `event_id`. The app maps it to the container's `source_data_identifier`, so **SOAR de-duplicates natively**: a repeated event never creates a duplicate container, which is what makes overlapping poll windows safe.

Ingest settings

Configure ingestion in the asset's **Ingest Settings**:



Setting	Description
`ingest_event_type`	Which events to ingest: `notable`, `stateful`, or `both` (default)
`ingest_tenant`	Optional **comma-separated list** of tenant identifiers, to restrict ingestion to specific TrackMe tenants. Leave empty to ingest all tenants
`ingest_lookback`	The Splunk earliest-time window searched on **every** poll (default `-30m`)
`ingest_max_events`	Maximum total number of events ingested per poll, across all selected event types (`0` for no limit, default `1000`)

In the standard SOAR **Ingest Settings**, set the **label** that ingested containers receive, and enable **Polling** with a **Polling interval** in minutes.

Recommended polling configuration

Important

Poll **every 5 minutes** and leave `ingest_lookback` at the default **-30m**.

Every poll searches the full `ingest_lookback` window rather than a narrow since-last-poll window. This is a deliberate trade of load for reliability: because `event_id` de-duplication is guaranteed, a repeated event never creates a duplicate container, so overlapping the window makes it far less likely that a late-arriving or briefly missed event is dropped. A 30-minute window polled every 5 minutes gives each event several opportunities to be picked up.

It is not free — each poll re-runs the whole lookback search and the connector re-processes rows it has already seen, so a wider window or a shorter interval costs more Splunk search and connector work. Size it for your environment:

- if your events arrive late, **widen** `ingest_lookback` rather than lengthening the poll interval;
- `ingest_max_events` caps how many events a single poll ingests, so a poll that reaches the cap will not reach every event in the window — raise it (or set it to `0`) in high-volume environments, and size it for the volume a wider window returns.

Results in SOAR

Ingested events appear as containers under the label you configured:



Each container carries one artifact holding the **complete TrackMe event**, so playbooks can read any field without a mapping step — including `tenant_id`, `object`, `object_state`, `priority`, `anomaly_reason`, `status_message`, `drilldown_link` and the TrackMe identifiers:



Stateful alerts carry an `alert_status`. When TrackMe closes the alert, the corresponding container is created with a **closed** status, so the SOAR view reflects the TrackMe lifecycle:



Container severity is derived from the event: the entity **state** drives it where present (`red` maps to *high*, `orange` to *medium*, `green` and `yellow` to *low*), falling back to the event **priority** otherwise.

Hint

Use **Poll Now** on the asset to validate ingestion immediately, without waiting for the scheduled poll. If no container appears, check that the asset's ingest label exists, and that the service account may run searches and read the alert indexes — see *The service account and its RBAC*.

Migrating from the SOAR Export forwarding rule

If you currently forward TrackMe Notable Events with the *Splunk App for SOAR Export*:

1. Configure **Ingest Settings** on the TrackMe asset and enable polling, as described above.
2. Use a **different label** from the forwarding rule, so the two paths remain distinguishable while you compare them.
3. Once you are satisfied that the ingested containers carry everything your playbooks need, **disable the Event Forwarding Rule** and repoint your playbooks at the new label.
4. The Splunk report and the field mapping that fed the rule can then be retired.

Note

The two paths de-duplicate independently: the forwarding rule and the app's ingestion do not share state, so running both at once creates one container per path for the same TrackMe event. Run them in parallel only for as long as you need to compare them.

Usage example

Beyond ingestion, the app's actions let a playbook act back on TrackMe. A common pattern is to react to an ingested TrackMe alert and update the entity's acknowledgement status, place it under a maintenance window, or trigger an AI component-health advisor — all from the container the ingestion created.

See also

- *TrackMe App on SOAR* — the readable overview.
- *Splunk SOAR* — monitoring the health of a Splunk SOAR deployment with TrackMe.
- *Notable events* — the TrackMe Notable Events ingested by the app.
- *Stateful alerting* — the TrackMe Stateful Alerts ingested by the app.
- *Operations & Administration* — the TrackMe REST API the app calls.
- *Roles & access control* — the access the SOAR asset authenticates with.

8.9.15 Cribl Stream monitoring with Flex Objects

Important

Monitoring a Cribl Stream deployment? Start with the native Cribl Stream tenant. TrackMe has a dedicated *Cribl Stream tenant type*: it reads the Cribl metrics already in your metric index and the Cribl Leader API, with no tracker to write, models the traffic of every Source, Destination, Pipeline and Route on its own, adds the inventory, the Worker machines and the deployment state, and raises findings you triage in a purpose-built workspace. See *Cribl Stream and Cribl Edge* and *Creating a Cribl Stream or Cribl Edge tenant*.

Cribl Edge fleets have their own dedicated tenant type too: see *Cribl Edge*.

This page describes the earlier approach, built on *Flex Objects* use cases. It remains supported, to build custom Cribl monitoring on your own SPL inside an entity tenant (stateful alerting, priorities, SLA).

Cribl Stream sits in the middle of the data path: every pipeline, route, pack and destination it runs is a place where data can slow down, get dropped or stop. TrackMe turns Cribl's own **internal metrics** into monitored **entities** — one per pipeline, route, input, destination and worker host — with a state, live metrics, 24-hour history and ML-learned baselines, then lets you arrange them into a **living map** of your Cribl deployment with *Topology Studio* and alert on it as a whole.



Nothing is installed on Cribl. Monitoring is built on the *Flex Objects* component and a ready-made library of **twelve Cribl use cases** — ten for Cribl Stream, two for Cribl Edge — that query the internal metrics Cribl already ships to Splunk.

Note

Cribl monitoring uses the Flex component (`splk-flx`), a restricted Enterprise / Unlimited capability (not available with the Foundation Edition trial).

How it works

Tip

This page monitors Cribl's own infrastructure. To monitor the **original data sources and hosts** observed in Cribl through DSM/DHM, follow the *Cribl activity signals recipe*.

Three ideas make the integration robust at scale:

- **One tracker covers every worker group.** Every search breaks by Cribl's `group` dimension, which becomes part of each entity's identity — add a worker group on the Cribl side and its pipelines, routes and hosts appear on the next run.
- **Each entity group decides its state the way that fits its data.** Health checks and backpressure use a **ratio** over the last 30 minutes so that a single blip never raises an alert; infrastructure uses **thresholds** you can edit per entity; traffic flows use **ML outliers** learned per pipeline, route, pack, input and output.
- **Everything is an entity.** Cribl objects get the same lifecycle as any TrackMe entity: stateful alerting, maintenance, priorities, SLA, notes, and a place on a topology.

The Cribl use-case library

Cribl Stream

Use case (uc_ref)	What it monitors	Entity group	State decided by
cribl_logstream_health_i	Health of every Logstream **input**, by input type	Cribl_Logstream:h	health-check ratio
cribl_logstream_health_o	Health of every Logstream **output** (destination)	Cribl_Logstream:h	health-check ratio
cribl_logstream_output_d	Destinations **blocked** or under **backpressure**	Cribl_Logstream:D	health-check ratio
cribl_logstream_hosts_cp	Worker hosts **CPU** (average, max, 95th percentile)	Cribl_Logstream:i	threshold (avg CPU < 90%)
cribl_logstream_hosts_me	Worker hosts **memory**: heap total / used, RSS, heap usage %	Cribl_Logstream:i	threshold (heap usage)
cribl_logstream_pipeline	**Pipelines**: events in, out, dropped, % sent, % dropped	Cribl_Logstream:p	ML outliers	c
cribl_logstream_route_tr	**Routes**: events and bytes in and out	Cribl_Logstream:r	ML outliers
cribl_logstream_pack_tra	**Packs**: events in and out	Cribl_Logstream:p	ML outliers
cribl_logstream_total_tr	**Total input traffic** per input: events, bytes, MB	Cribl_Logstream:t	ML outliers	l
cribl_logstream_total_tr	**Total output traffic** per destination: events, bytes, MB	Cribl_Logstream:t	ML outliers	al

The CPU and memory use cases share the same entity identity, so both trackers enrich **one** entity per worker host. Health checks classify the last 30 minutes of Cribl's own health values (0 green, 1 yellow, 2 red): an entity turns red when at least 75% of the checks are red, orange when they are mostly yellow — the ratio absorbs the transient flaps a raw health value would raise. Traffic use cases start green and let *ML outlier detection* learn what normal looks like for each pipeline or route, flagging a flow that drops, floods, or starts dropping events.

Each entity carries the metrics of its use case (`cribl_logstream.pipeline.in_events`, `cribl_logstream.route.route_out_mbytes`, `cribl_logstream.avg_cpu_perc`…), so they are available as KPIs on a topology, as sparklines in the inspector and in emails.

Cribl Edge

Two further use cases cover a **Cribl Edge** fleet: `cribl_edge_fleet_metrics` reads the Edge internal metrics indexed in Splunk and flags a node whose metrics stop arriving, and `cribl_edge_fleet_api` polls the fleet through the Cribl API with the `TA-trackme-cribl` add-on, reporting each node's heartbeat delay and input/output health. See *Cribl Stream & Edge API* for the add-on.

To list every Cribl use case with its full definition, run:

```
| trackmesplkflxgetuc | search uc_vendor=Cribl
```

Requirements

Cribl internal metrics in Splunk

The Stream use cases rely on **Cribl internal metrics** being indexed in a Splunk metrics index — the Cribl-side *Splunk* destination for internal metrics, typically an index such as `cribl_metrics`. Verify they arrive before creating trackers:

```
| mpreview index=cribl_metrics filter="metric_name=cribl.logstream.*"
```

The library searches read every metrics index by default (`where index=*`). For performance on large deployments, pin your Cribl metrics index when creating a tracker — the wizard shows the SPL and lets you edit it:

```
| mstats sum(cribl.logstream.route.in_bytes) as route_in_bytes, sum(cribl.
↪logstream.route.in_events) as route_in_events, sum(cribl.logstream.route.
↪out_bytes) as route_out_bytes, sum(cribl.logstream.route.out_events) as␣
↪route_out_events where index=cribl_metrics host=* by group, name
```

For the metrics Cribl emits, see https://docs.cribl.io/stream/internal-metrics.

Hint

Multiple worker groups

Nothing to do: a single tracker manages all worker groups individually, because every search breaks against the Cribl `group` dimension and that group is part of each entity's name (`pipeline|group:default|pipeline:webserver_nginx`).

A Flex-enabled tenant

You need a tenant with the **Flex Objects** component enabled — a dedicated tenant for Cribl (`cribl-mon` in the examples below) keeps the Cribl entities, their alerts and their topology together, but any Flex-enabled tenant works. If the metrics live on another search head, create the trackers against a *remote deployment*.

Setting it up

Trackers are created from the **hybrid tracker wizard** in your tenant (Tenant Home → Actions → *Flex Objects tracking*), one tracker per use case — see *Creating a Flex tracker* for the generic walkthrough. For Cribl the seven steps go like this:

1. **Tracker name and Splunk deployment** — a meaningful name such as `cribl_pipelines` (it prefixes the tracker's searches), local or a remote account.
2. **Use case & search logic** — load the search from the library: vendor **Cribl**, then the use case reference. The wizard shows the use case's description, its implementation notes and the metrics it produces, then the SPL itself, fully editable: pin your metrics index, exclude an internal destination, rename the group.
3. **Time ranges** — pre-filled from the use case (`-5m` for the traffic use cases, `-30m` for the health ratios).
4. **Test and configure — simulate** the search and review the entities it would create, with their parsed metrics and, for the traffic use cases, the outlier definitions.
5. **Performance benchmark** — the wizard times the search so you know what the schedule will cost.
6. **Cron schedule** — every five minutes for every Cribl use case.
7. **Validate creation.** The entities appear on the first run.





The wizard also offers **Generate with AI** at this step: describe what you want to monitor in plain language, and the AI drafts the tracker search from the use-case library, dry-runs it against your live data and fills the wizard with a validated proposal — handy when a Cribl use case needs a twist, such as one tracker per worker group or a filter on a naming convention.





Reading the results

Once the trackers run, Tenant Home lists the Cribl entities under their `Cribl_Logstream:*` groups, with the status description the use case wrote for each one — for a pipeline, the percentages sent and dropped and the raw in, out and dropped counts; for a destination, the share of green health checks and pressure checks. Filter the table on a group or a name to focus on one family, and expand a row for the entity's key information, its status message and its recent metrics without leaving the table:



Open an entity for the full picture. The **Overview** tab charts any metric of the use case over the period you choose — here the events in and out of a pipeline over seven days, as stacked columns or areas — alongside the entity's state, impact score and SLA percentage:





For the traffic use cases, the **Outliers anomaly detection** tab shows the model learned on each metric: the expected band, the values that breached it, and the counters of accepted, rejected and corrected outliers. This is where a pipeline that suddenly floods or goes quiet is caught, without a threshold anyone had to write — and where you tune the model, mark a false positive, or ask the AI ML Advisor for a second opinion:



From here the entity behaves like any other: adjust thresholds, tune outlier detection, set a priority, attach it to an SLA, or put it into maintenance.

A living map: Cribl Stream in Topology Studio

Entities tell you *which* pipeline is unhappy; a topology tells you **what that means for the flow**. The view above is eight aggregates and seven edges, built entirely from the Cribl entities of one tenant:

- a **root aggregate**, *Cribl Logstream*, scoped to the tenant's Flex component and filtered on `priority IN ("high", "critical")` — it folds every important Cribl entity into one node and shows the healthy percentage of the deployment;

- one **aggregate per entity group** — Health Sources, Health Destinations, Destination Pressure, Infrastructure, Pipelines, Traffic In, Traffic Out — same scope, same priority filter plus an object pattern that selects the group, for example `priority IN ("high", "critical") object="*pipeline_traffic*"`;
- **auto-generated members** on every group aggregate, so each pipeline, input, output and host is its own node, with **default member KPIs** in the **sparkline** style turning the entity metrics into on-canvas trends: `in_events` and `out_events` for pipelines, `total_in_mbytes` for inputs, `total_out_mbytes` for outputs, `perc95_cpu_perc` and `avg_heap_used_pct` for hosts;
- **edges** from the root to each group so the map reads top-down, and one edge from *Infrastructure* **into** the root with *impact propagation* enabled: a worker host in trouble repaints the whole deployment, because nothing flows without the hosts;
- a fixed **canvas size** (3000 × 1300) for a stable presentation, plus the Cribl Stream logo and the organisation name as *decorations*.

Because membership is dynamic, a pipeline added on the Cribl side appears on the map on the next refresh, with its sparklines, without anyone editing the view.

When something degrades, the map reads itself: below, one pipeline turned red on an outlier, the Pipelines aggregate dropped to 80% healthy, the root re-classified to 95% and a worker host went blue — all visible at a glance in full-screen mode, with the selected host's live metrics in the inspector.



Selecting a pipeline node shows the same entity the tracker maintains — its state, its live in and out events, and the 24-hour history of every pipeline metric plus the ML outlier models — with **View status**, **View details** and **Open in Tenant Home** one click away:



To build your own, follow the *end-to-end example* with your Cribl tenant — the filters above are all you need — or describe the map to *Build with AI*: *"one aggregate per Cribl_Logstream group of the cribl-mon tenant, high and critical priorities only, pipelines and inputs expanded with their traffic KPIs as sparklines"* — and refine the draft. Then add a *Topology Alert* on the view: one alert covers the whole Cribl deployment and its email embeds the rendered map.

See also

- *Cribl Stream and Cribl Edge* — the native Cribl Stream and Cribl Edge tenant types.
- *FLX — Flex Objects* — the Flex component and its template library.
- *Machine Learning* — tuning outlier detection on the traffic use cases.
- *Remote Splunk deployments* — monitoring Cribl metrics that live on a remote search head.
- *A self-managed Leader the stack cannot reach* — reaching a self-managed Leader from Splunk Cloud, for a Cribl connection, through a *TrackMe Broker*.
- *Cribl Stream & Cribl Edge* — the Cribl Stream & Edge API add-on.

8.9.16 Elastic sources

Elastic sources turn an arbitrary SPL search into a tracked DSM entity. Instead of discovering entities from straightforward `index` / `sourcetype` pairs, you write the search that *defines* what to monitor — `tstats`, `raw`, `mstats`, or `from` a data model or lookup — and TrackMe tracks its delay, latency and volume like any other feed. One search defines one entity.

They are the right tool when the thing you want to watch is not a plain feed: a syslog collector keyed by a search-time field, a data-model- or lookup-derived source, or any case that relies on fields only resolved at search time.

Two flavours

Kind	Behaviour
Shared	Many elastic entities are evaluated by a single, pooled, concurrency-capped tracker — efficient when you have many lightweight virtual entities.
Dedicated	Each entity is backed by its own independent scheduled report — use when an entity needs its own cadence or a heavier search.

Both can run against a local or a **remote** Splunk deployment (see *Remote Splunk deployments*).

Note

For high-cardinality discovery, *Hybrid trackers* have largely superseded elastic sources — and for monitoring **lookups** specifically, the dedicated *Lookups monitoring* add-on replaces the old elastic-source lookup example. Elastic sources remain the right tool for `from` / `raw` single-entity edge cases that depend on search-time fields.

See also

- *Elastic sources — in depth* — the Shared and Dedicated tracker internals, engine commands, KV Store collections, and worked examples.
- *DSM — Data Source Monitoring* — the DSM component elastic sources extend.
- *Lookup monitoring* — the modern way to monitor lookups.
- *Hybrid trackers* and *Replica trackers* — advanced tracker types.

8.9.17 Elastic sources — in depth

Tip

This is the **in-depth reference** for Elastic sources. For the readable overview, start with *Elastic sources*. This page covers the Shared and Dedicated Elastic tracker internals (engine commands, KV Store collections, concurrency, retry logic) and worked examples. For monitoring **lookups** specifically, the dedicated *Lookups monitoring* add-on now supersedes the old elastic-source approach.

Introduction to Elastic sources

Elastic sources feature versus Hybrid Trackers in TrackMe V2

- **In TrackMe V2 and in a large number of use cases, the concept of Elastic Sources is superseded by Hybrid Trackers**
- Indeed, an Elastic Source creates and maintains a single TrackMe entity, where Hybrid Trackers can manage from a few to tens of thousands of entities
- However, there are also use cases where Elastic Sources can be required and can be very valuable to meet your requirements or to address specific use cases such as tracking lookups and more

What does the Elastic sources feature do?

- The Elastic sources feature provides a built-in workflow to create virtual data sources based on any constraints and any Splunk language
- In TrackMe V2, the target can be the **local Search Head tier**, as well as any **remote Splunk deployment or instance** using TrackMe's built-in remote search capabilities
- Elastic Sources allows extending TrackMe feeds tracking capability in a rich and flexible way
- Elastic Sources can be based on `tstats`, `raw`, `from (datamodel and lookup)` and `mstats` searches

Some examples of use cases for Elastic Sources:

- Creating a Virtual entity which is the combination of different indexes, sourcetypes, sources and so forth with additional search filters, using indexed or search time extracted fields

- Creating a Virtual entity based on a Splunk search language not supported by Hybrid Trackers (`from` which allows calling lookups or datamodels)
- Create a Virtual entity based on a `raw` search, as opposed to `tstats` used typically on Hybrid Trackers for performance purposes, and where the use case requires using search time extracted only fields
- Tracking specific Splunk objects such as lookup content over time (CSV based lookup or KVstore), to detect if the lookup is being updated and also apply Machine Learning based outliers detection on the number of records

Shared Elastic versus Dedicated Elastic:

- An Elastic Source can be added to the **Shared tracker**, or created as an independent **Dedicated Tracker**
- Shared Elastic Sources are orchestrated by the Elastic Sources shared tracker
- Dedicated Elastic Sources are orchestrated by their own scheduled report
- The Shared Elastic Sources tracker processes searches in a multithreading parallel way (the max number of concurrent searches is configurable at the TrackMe system level and can be overridden on a per tenant basis)
- When the Shared Elastic Sources tracker reaches its maximal run time (which is automatically calculated depending on its cron schedule and time margin configuration), it stops its execution to avoid generating skipping searches
- On the next iteration, the Shared Elastic Sources tracker will resume its execution starting with entities which have not been inspected previously (but the oldest age of the inspection date)
- On the other hand, Dedicated Elastic trackers are fully independent scheduled reports, their life cycle remains orchestrated by TrackMe but these run independently from each other

Accessing the Elastic source creation UI

First, let's expose how to access the Elastic sources interface, from the data sources tab in the main UI, click on the `Elastic Sources` button:



The following screen appears:



Elastic source example 1: raw search with search time only extracted fields

Hint

tstats versus raw

- These steps are in fact the same whenever you use a raw search, or a tstats based search
- In the first case and in the example below, we use a raw search to benefit from search time extracted fields
- A tstats based search is how highly recommendable when possible, performances are infinitely better than raw searches, and computing costs much lower

In this example, requirements are:

- To monitor the Web data coming from a specific Website (trackme-solutions.com)
- Fields that can be used are only available at search time, creating indexed time fields for this use case would be challenging and possibly overkill for this edge use case (assuming we use tstats Hybrid Trackers globally)
- The volume of data is relatively low, allowing efficient searches to be performed

The following search constraints provides access to the data we need:

```
(index=webserver site="trackme-solutions.com")
```

We will create a new Shared Elastic source in a just a couple of easy steps:

Step 1: define name of the entity

- This defines the name of the entity, stored as the value for `object` and `alias` in TrackMe's collections (you probably want something meaningful)
- Also define the target, `local` means that we looking at data searchable from the local Search head tier, `remote` would target a remote account previously configured



Step 2: choose the language

- You can choose between different options, in our case we will use `raw` because we want to use search time extracted fields



Then we define the search constraint:

- The search constraint should target valid filters, allowing the search to be as efficient as possible and targeting the data we want to monitor



Hint

Defining the search constraint

- The search constraint is meant to be the list of filters providing access to the data you want to monitor
- This can be a simple constraint, or a complex one using OR and AND operators, different levels of parentheses, etc
- However, it is not meant to be the place where you would perform complex regular expression extractions for this instance, this should happen at the props and transforms level instead

In step 3, you should define proper values for index and sourcetypes:

- These are virtual values, they do not impact the search itself but are used to classify the entity in the UI
- You can choose anything you want, but it is recommended to use values that are meaningful to you



Finally in Step 4, define the earliest and latest time ranges:

- These are relevant for both Shared and Dedicated Sources
- The values influence how performing this search will be, especially with raw searches
- In this example, we reduce the earliest to -1h, latest should be in the future to be able to detect events abnormally defined in the future



Finally, after the simulation, we validate the creation as Shared or Dedicated:


Elastic sources definition: use this interface to create and manage elastic sources
5. Test and Review:
Test the Elastic source execution
Simulate the search
simulation_results
object
data_eventcount
data_first_time_seen
Success, you can now add this new source to the shared tracker or as a dedicated tracker.
webserver.trackme-solutions.com
6. Validate the creation
Promote to Live, the Elastic source will be created and tracked:
Add to the shared tracker
Add as a new dedicated tracker
Cancel
Back


Elastic sources definition: use this interface to create and manage elastic sources
5. Test and Review:
Test the Elastic source execution
Simulate the search
simulation_results
object
data_eventcount
data_first_time_seen
Success, you can now add this new source to the shared tracker or as a dedicated tracker.
webserver.trackme-solutions.com
6. Validate the creation
Promote to Live, the Elastic source will be created and tracked:
Add to the shared tracker
Add as a new dedicated tracker
Cancel
Back



Elastic source example 2: tracking lookups update and number of records

Note

Superseded by the lookups search mode of the Hybrid Trackers wizard — see *Lookup monitoring*.

The dedicated lookups search mode is significantly faster on large lookups, reads the **filesystem mtime** of CSV files directly (rather than a per-row event time that the lookup may or may not maintain), and probes an operator-supplied list of candidate timestamp fields for KVstore collections instead of assuming a single `_time` field. The example below is preserved for reference and for deployments that still rely on the Elastic Source approach.

In this example, we will demonstrate how we can monitor the content of a Splunk lookup, KVstore or CSV based lookup, and get alerted if the lookup is not updated as expected.

Lookups are very useful in Splunk, we use these all the time as intermediate storage, for enrichment purposes, etc.

However, and this is especially true for security use cases, being capable of detecting if a lookup used in other use cases has stopped being updated or maintained is a critical requirement.

In our example, we have a KVstore based lookup called `cybops_net_traffic_monitor` which is updated on a regular basis and contains some associations and pre-calculations for network devices:



We will use the following Elastic Source definition to monitor our lookup, and we will leverage the exec_time field to define the _time value:

```
| from lookup:cybops_net_traffic_monitor | eval _time=exec_time
```



Administering Shared Elastic Sources

Shared Elastic sources are orchestrated by the Shared Elastic tracker, this tracker is automatically created along with the Virtual Tenant creation. (if the component splk-dsm is enabled)

The tracker is called:

- trackme_dsm_shared_elastic_tracker_tenant_<tenant_id>

The tracker is a scheduler wrapper which calls a Python engine:

```
| trackmeelasticexecutor tenant_id="<tenant_id>" component="splk-dsm"
```

You can find its execution logs using the following search:

```
index=_internal sourcetype=trackme:custom_commands:trackmeelasticexecutor
```

The Shared Elastic tracker loads entities to be processed from a KVstore, the following search shows the content of the KVstore: (update the tenant_id)

```
| inputlookup trackme_dsm_elastic_shared_tenant_<tenant_id>
```

Its execution is influenced by the following configuration item(s), go in Configuration / splk-general:

Table 23: Shared Elastic tracker configuration

Option	Purpose	Default
Concurrent searches Elastic	System level number of parallel concurrent searches for Shared Elastic sources, this can be overridden on a per tenant basis using max_concurrent_searches on the Shared Elastic tracker	3

You can review the performance run time of the Shared Elastic Tracker using the deep dive dashboard:



When the job starts, the following message is shown:

```
2023-11-09 22:21:11,840 INFO trackmeelasticexecutor.py generate 426 tenant_id=
↪"09-demo-elastic", component="splk-dsm", report=
↪"trackme_dsm_shared_elastic_tracker_tenant_09-demo-elastic", Elastic Sources␣
↪shared job started, max_concurrent_searches=3, margin_sec=60
```

This message indicates the current number of max concurrent searches, as well as the time margin in seconds to calculate the max run time of the full job.

The max concurrent job is first defined at the global level as explained above, but it can also be overridden on a per tenant basis adding the argument in the search definition:

```
**Syntax:** **max_concurrent_searches=****
**Description:** The max number of searches to be executed in parallel, if set to␣
↪a different value than the system default, this value wins.
```

The margin in seconds defines the time in seconds to be deducted from the total cron translation in seconds, for instance if the job is scheduled to run every 5 minutes, the cron sequence is 300, the job will interrupt at 300-<margin_sec>, this value can be overridden on a per tenant basis by adding the argument:

```
**Syntax:** **margin_sec=****
**Description:** The time in seconds used as a margin when calculating the max␣
↪runtime depending on the cron schedule.
If the search is triggered every 5 minutes, the max runtime will be 5 minutes less␣
↪the margin_sec value.
```

When the job inspects an entity to be processed, it will show the following message: (example)

```
2023-11-09 22:32:28,903 INFO trackmeelasticexecutor.py process_elastic_object 176␣
↪tenant_id="09-demo-elastic", component="splk-dsm",␣
↪report="trackme_dsm_shared_elastic_tracker_tenant_09-demo-elastic", processing␣
↪elastic object, object="elastic:local:lookup:cybops_net_traffic_monitor", last␣
↪inspection with tracker_runtime="2023-11-09 22:26:57",␣
↪elastic_report_root_search="| from lookup:cybops_net_traffic_monitor | eval _
↪time=exec_time
| eventstats max(_time) as indextime | eval _indextime=if(isnum(_indextime), _
↪indextime, indextime) | fields - indextime
| eval host=if(isnull(host), "none", host)
| stats max(_indextime) as data_last_ingest, min(_time) as data_first_time_seen,␣
↪max(_time) as data_last_time_seen, count as data_eventcount, dc(host) as␣
↪dcount_host
| eval latest_eventcount_5m=data_eventcount
| eval object="elastic:local:lookup:cybops_net_traffic_monitor",␣
↪data_index="lookups", data_sourcetype="cybops_net_traffic_monitor" |␣
↪`trackme_elastic_dedicated_tracker("09-demo-elastic")`
| eval tenant_id="09-demo-elastic"
| stats count as report_entities_count by tenant_id
| `register_tenant_component_summary(09-demo-elastic, dsm)`"
```

Once the search has been executed, TrackMe shows is summary result and its run time information:

```
2023-11-09 22:32:57,742 INFO trackmeelasticexecutor.py process_elastic_object 245␣
↪tenant_id="09-demo-elastic", component="splk-dsm", object=
↪"elastic:local:lookup:cybops_net_traffic_monitor", report=
↪"trackme_dsm_shared_elastic_tracker_tenant_09-demo-elastic", Entity search␣
↪successfully executed, status="success", run_time="28.839"
```

If the system reaches the max number of concurrent searches (Splunk wise), TrackMe will re-attempt automatically for a certain number of times, which is visible with a warning message:

```
2023-11-09 22:46:54,302 WARNING trackmeelasticexecutor.py process_elastic_object␣
↪265 tenant_id="09-demo-elastic", component="splk-dsm", report=
↪"trackme_dsm_shared_elastic_tracker_tenant_09-demo-elastic", temporary search␣
↪failure, current_failures_count=4, max_failures_count=24, exception="HTTP 503␣
↪Service Unavailable -- b'{"messages":[{"type":"WARN","text":"Search not␣
↪executed: The maximum number of concurrent historical searches on this instance␣
↪has been reached., concurrency_category=\\"historical\\", concurrency_context=\\
↪"instance-wide\\", current_concurrency=30, concurrency_limit=30","help":""}]}'"
```

Once all entities to be processed have been processed effectively, the job will show the following final message which includes the total run time: (which runtime is shown in the deep dive dashboard)

```
2023-11-09 22:53:18,138 INFO trackmeelasticexecutor.py generate 535 tenant_id=
↪"09-demo-elastic", component="splk-dsm", report=
↪"trackme_dsm_shared_elastic_tracker_tenant_09-demo-elastic", Elastic Sources␣
↪shared job successfully executed, status="success", run_time="82.069",␣
↪entities_count="6"
```

Administrating Dedicated Elastic Sources

Dedicated Elastic trackers are independent scheduled reports, their life cycle is orchestrated by TrackMe but they run independently from each other.



Each tracker is composed by two reports, the tracker which is scheduled and the wrapper which is called by the tracker:



The tracker calls the trackmetrackerexecutor command:



Execution logs can be found with the following search:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerexecutor
```

Similarly, use the deepdive performance dashboard to review the tracker performance over time:



TrackMe stores dedicated Elastic trackers main Metadata in the following KVstore:

```
| inputlookup trackme_dsm_elastic_dedicated_tenant_<tenant_id>
```

8.10 Operations & Administration

The day-to-day mechanics of running TrackMe at scale: global configuration, scaling, maintenance, backups, the Configuration Guardian, config-as-code, the REST API, and the data TrackMe itself produces.

- **Configuration —** the service account, capabilities, and system-wide settings.
- **Large-scale environments —** designing and operating TrackMe at scale.

- **Maintenance mode** — suppress alerting and SLA accounting during planned work.
- **Bank holidays** — calendar-aware SLA exclusion.
- **Data sampling** — DSM event-format recognition for data quality.
- **Backup & restore** — protect and recover TrackMe's state.
- **Configuration Guardian** — automatic misconfiguration detection.
- **Configuration Manager** — config-as-code / CI-CD.
- **Support diagnostics** — generate a diagnostic archive for a support case.
- **REST API** — automate everything TrackMe does.
- **Consuming TrackMe data** — status counts and entity records in your own dashboards or external tools.
- **Events & sourcetypes** — the events TrackMe produces.
- **Metrics reference** — the metrics TrackMe writes, by component.

8.10.1 Configuration

The top-navigation **Configuration** menu provides native TrackMe interfaces for system-wide settings and integration accounts. **System settings** manages logging, default ownership and RBAC, and the defaults used across tenants. Dedicated pages manage **Remote deployment accounts**, **Email delivery accounts**, **S3 export accounts**, **Cribl connections** and **Webhook accounts**; **Configuration Guardian** checks configuration health. Native interfaces replaced the Splunk UCC configuration page in version **2.4.8**. AI Provider accounts are managed under the **Artificial Intelligence** menu (see *Configuring AI Provider accounts*).

The service account

A dedicated **service account** is optional but recommended. Without one, TrackMe creates its knowledge objects and runs its scheduled searches as the Splunk system account — which works, but a dedicated account gives you clean cost attribution and lets you apply Splunk Workload Management.

A service account needs, at minimum: the Splunk `power` role (for `search` and `schedule_search`), the three TrackMe capabilities (`trackmeuseroperations`, `trackmepoweroperations`, `trackmeadminoperations`), and the ability to search all non-internal indexes plus `_internal`. See *Roles & access control* for the role model and *The scheduled owner and service account* for component-specific scheduled-owner requirements beyond this baseline.

Important

System settings and account management require a **Splunk administrator-class user** with TrackMe admin capabilities, typically `trackme_admin` together with `admin` / `sc_admin`. If a configuration request is denied, the native interface displays an *Insufficient permissions* message. Tenant administration and scheduled monitoring use separate permissions. For next-generation components, named scheduled owners have additional requirements: use *The scheduled owner and service account* for the current owner checks and *Roles & access control — in depth* for each source/API account.

System-wide settings

The **System settings** page holds the settings that apply across all tenants, including:

- enabling the **AI Assistant** and the **Configuration Manager** receiver;
- **data-sampling** thresholds and **maintenance / SLA-exclusion** behaviour;
- **logging levels**, honoured by both the REST API and the custom commands;
- the **user-interface defaults** (theme, auto-refresh, volume unit labels and more) that apply until each user picks their own (see *Personal preferences*);

Remote Splunk accounts are managed separately under **Configuration → Manage Remote deployment accounts** (see *Remote Splunk deployments*). To change an existing tenant's RBAC or ownership, use its Virtual Tenant UI (see *Managing tenants*); system-wide defaults apply to tenant creation.

See also

- *Configuration — in depth* — every Configuration interface, the service-account & minimal-permission setup, and logging mechanics.
- *Roles & access control* — the capability and role model.
- *REST API reference* — the API behind the Configuration UI.
- *Large-scale environments* — using a service account at scale.

8.10.2 Configuration — in depth

Tip

This is the **in-depth reference** for TrackMe's configuration interfaces. For the readable overview, start with *Configuration*. This page walks every configuration interface, the service-account creation and minimal-permission setup, and the logging mechanics.

Hint

Looking for a Quickstart?

- If you are looking for a quickstart, you can refer to the *QUICK START - Starting with TrackMe: (feed tracking quickstart)*
- The quickstart guide provides a step-by-step guide to get started with TrackMe, focusing on feed tracking, the easy and simple way

Using a service account for TrackMe is not mandatory

- It is not mandatory to use a service account for TrackMe, as described in the next steps, but it can be considered a good practice which provides several advantages in terms of management and monitoring
- However, this is not required as such and not doing so will not prevent TrackMe from operating normally
- By default, TrackMe will create knowledge objects and run searches as the "nobody" user (Splunk system account), which does not require any setup

Hint

Distinguishing permissions requirements between service accounts, TrackMe administrators, Power and read-only users

- A **service account** is a Splunk user (internal or SAML) that is used by TrackMe to perform scheduled activities, such as the creation of knowledge objects, the execution of scheduled searches, the creation of Virtual Tenants, etc.
- TrackMe users can have different levels of permissions. TrackMe comes with 3 built-in concepts for users: **administrators**, **power** users, and **read-only** users

- TrackMe leverages sophisticated techniques to ensure that you can define minimal permissions, for both the service account and the TrackMe users
- This allows for TrackMe users (TrackMe admins, power and read-only users) to avoid having to provide potentially dangerous capabilities such as list_settings, list_storage_passwords, etc.
- TrackMe comes with 3 built-in roles, **trackme_admin**, **trackme_power** and **trackme_user**
- **trackme_user** inherits from Splunk built-in **user** role, **trackme_power** inherits from **trackme_user** and **trackme_admin** inherits from **trackme_power**
- Each TrackMe built-in role enables the associated TrackMe capability, for instance **trackme_admin** enables the **trackmeadminoperations** capability

Summary Requirements for the TrackMe Service Account

The following is the general service-account baseline. Next-generation components have additional, explicit named-owner checks and separate source/API readers. Before provisioning UAM, HEC, S2S, UF Fleet, Forwarder Collectors, Cribl Stream or Cribl Edge, follow *The scheduled owner and service account* and the component matrices in *Roles & access control — in depth*.

Indexes Access

The service account should be able to search all `non-internal indexes` and all `internal indexes` (or at least the indexes containing data to be monitored, as well as the _internal index)

Capabilities

The service account should have the following capabilities at the minimum:

Hint

Inherit from Power role

- At the minimum, the service account should inherit from the Power role, which provides the basic required Splunk capabilities.
- Then add additional capabilities as needed.

Capability	Note
schedule_search	Required for running scheduled searches (provided by the Power role)
search	Required for running searches (provided by the Power role)
trackmeadminoperations	Can be inherited from the trackme_admin role (not provided by the Power role)
trackmepoweroperations	Can be inherited from the trackme_power role (not provided by the Power role)
trackmeuseroperations	Can be inherited from the trackme_user role (not provided by the Power role)

Summary Requirements for TrackMe Administrators

Essentially, TrackMe administrators need to have the following capability:

Capability	Note
trackmeadminoperations	Can be inherited from the trackme_admin role

System settings and account management require a Splunk administrator-class user with TrackMe admin capabilities, typically `trackme_admin` together with `admin` / `sc_admin`. These native interfaces are available from the **Configuration** menu and show an *Insufficient permissions* message if a configuration request is denied.

This requirement is separate from the minimal service-account permissions above and from creating or managing tenants and trackers. For default ownership and RBAC, open **Configuration → System settings → General → Knowledge objects & RBAC defaults**. For an existing tenant, use its Virtual Tenant UI; see *Managing tenants*.

Service Account and Permissions

To operate, TrackMe allows and recommends defining a Splunk user that has the ownership of any knowledge objects created by TrackMe as part of the Virtual Tenant lifecycle:

- Knowledge Objects (such as reports, alerts…) will be assigned to the user tagged as the owner of the Virtual Tenant
- Scheduled activities will run on behalf of the service account owner

By default, TrackMe assigns the user "admin" as the default owner of the Virtual Tenant; it is best practice to create your own service account owner. The following minimal permissions and capabilities are required:

- The service account needs to be a member of the built-in role `trackme_admin` as this provides the `trackmeadminoperations` capability, or this capability needs to be granted explicitly
- The service account needs to be able to search all `non-internal indexes` and all `internal indexes`
- The service account needs to be able to run scheduled searches; typically you can use the Splunk built-in `power` role

TrackMe implements a strict least privileges approach; consult *Roles & access control*

Note

Local service account user or SAML service account

- You can set up the service account user as a local user or a SAML user on the TrackMe Search Heads tier
- For other Search Head tiers, TrackMe can interact with the Splunk API for various powerful use cases such as TrackMe Flex Object trackers or the Workload component
- This requires a service account on the target Search Heads tier and a bearer token to be created
- If you want to create a SAML service account for TrackMe's remote search capabilities, you need to have the SAML AQR setup, and an Identity Provider (IDP) supported by Splunk
- Reference: https://docs.splunk.com/Documentation/Splunk/latest/Security/Setupauthenticationwithtokens
- Reference: https://docs.splunk.com/Documentation/SplunkCloud/latest/Security/SAMLConfigJWT

Creating a Service Account for TrackMe with Minimal Permissions

Note

Version 2.0.48 and later required for minimal permissions

- TrackMe version 2.0.48 and later is required for the following procedure allowing a strict minimalist service account

- Before this version, the service account needs to have extended capabilities such as list_settings and list_storage_passwords capabilities; therefore, the recommendation was for the service account to be a member of admin/sc_admin
- Some advanced use cases such as Flex Object trackers dealing with the Splunk `| rest` command or SOAR-related use cases may need additional capabilities to be granted to the service account user

One option is to create a specific role for the TrackMe service account with:

- Inheritance roles: `power`
- Role membership: `trackme_admin`
- Indexes: `all non-internal` and `all internal` indexes
- Resources: While TrackMe is optimized to distribute scheduled searches, it should be capable of running sufficient concurrent searches and it requires a large file quota to avoid issues

Hint

trackme_admin membership for the service account

- Before version 2.0.61, the service account needs to be an explicit member of the `trackme_admin` role (or the admin role in the tenants); this is needed because TrackMe requires explicit role membership (opposed to inheritance) to grant access to the Virtual Tenants
- From version 2.0.61, all RBAC dimensions in TrackMe support inheritance transparently







You can can then create the service account itself, example:

- The user is a member of the `svc-trackme` role
- As mentioned above, it is also a member of `trackme_admin` to be granted access to the Virtual Tenants
- Uncheck the box "Require password change on first login"



When you create a Virtual Tenant, you will specify the service account as the owner of the Virtual Tenant:



Hint

preset RBAC for the tenant creation UI

- You can preset the owner and roles applied when creating a new Virtual Tenant from the UI
- Go to **Configuration → System settings → General**, in the **Knowledge objects & RBAC defaults** group



The **Default owner**, **Default admin role**, **Default power role** and **Default user role** fields pre-fill the corresponding dropdowns in the Virtual Tenant creation wizard:



Minimal capabilities and resources for Remote Accounts and the user associated with the bearer token

TrackMe remote capabilities rely on a Splunk bearer token authentication, this token is associated with a Splunk user on the remote side which itself is associated with specific roles, capabilities, permissions and resources restrictions:

- **Roles and capabilities:** The user can be created with minimal permissions using the Splunk `user` role out of the box role. (you can inherit from user or a role providing the same capabilities than power)
- **Indexes:** Make sure the user can access to both normal and `internal` indexes.
- **Restrictions:** The user for TrackMe should not have any time limits restrictions, there are use cases which require long term searches.
- **Resources:** It is recommended to give to this user `enough concurrent searches` (unlike very basic or minimal user) as well as a `sufficient quota`. (5GB or 10Gb for instance)
- **Additional capability required:** Finally, for the purposes of the Workload, this user also needs to have the following capabilities granted `admin_all_objects`, `select_workload_pool`, `list_workload_pools` and `list_workload_rules` which are required for TrackMe's backend to access to all objects of all applications in a remote manner. (for the Metadata in the Workload component)

Table 24: In addition with the basic capabilities provided by the user role, these capabilities must be granted

Capability	Comment
admin_all_objects	Required
select_workload_pool	Required
list_workload_pools	Required
list_workload_rules	Required


Roles
Edit Role svc-trackme
Name *
1. Inheritance
2. Capabilities
3. Indexes
4. Restrictions
5. Resources
Role name
filter
user
Showing selected
You can use minimal permissions as basis
Cancel
Save


Roles
Edit Role svc-trackme
Name *
1. Inheritance
2. Capabilities
3. Indexes
4. Restrictions
5. Resources
Select specific capabilities for this role.
Capability Name
filter
Showing native
admin_all_objects
list_workload_pools
list_workload_rules
search_workload_pools
native
These additional capabilities are required for the TrackMe workload component
Cancel
Save


Roles
Edit Role svc-trackme
Name *
1. Inheritance
2. Capabilities
3. Indexes
4. Restrictions
5. Resources
Wildcards
Instead of selecting individual indexes, you can create a Wildcard index to dynamically capture all indexes that match the Wildcard. After you add a Wildcard index, it appears in the Indexes table. Wildcard indexes are limited to this role.
Add
Indexes
Index Name
Included
Default
Showing all
We need to access internal and non internal indexes
Cancel
Save


Roles
Edit Role svc-trackme
Name *
1. Inheritance
2. Capabilities
3. Indexes
4. Restrictions
5. Resources
This role
Default app
Select...
Role search job limit
Set a limit for how many search jobs that all users with this role can run at the same time.
Standard search limit
50
Real-time search limit
100
User search job limit
Set a limit for how many search jobs that a single user with this role can run at the same time.
Standard search limit
10
We need enough concurrent searches limits, note that TrackMe re-attempts searches automatically if needed.
Real-time search limit
0
Role search time window limit
Cancel
Save



Users and roles

TrackMe is deeply RBAC capable, consult the following documentation to configure users accesses for TrackMe:

- *Roles & access control*

Web Browsers and system compatibility

TrackMe should work fine with most Web Browsers and systems; however, if you experience icon issues due to the lack of support of ASCII emojis, you can enable the Bootstrap compatibility mode:

- *Web Browser compatibility*

Accessing TrackMe Configuration

From version **2.4.8**, TrackMe's configuration is managed through **native React interfaces** grouped under the top-navigation **Configuration** menu (and, for the AI providers, under the **Artificial Intelligence** menu). These replace the retired Splunk UCC configuration page:

Table 25: The native configuration interfaces

Interface	Navigation	Replaces / covers
System settings	Configuration → System settings	Every system-level setting (`trackme_settings.conf`): general behaviour, indexes, UI defaults, feed thresholds, data sampling, outlier detection, SLA, maintenance, health tracker, backup & restore, and logging
Manage Remote deployment accounts	Configuration → Manage Remote deployment accounts	Remote Splunk deployment accounts — see *Remote Splunk deployments*
Manage Email delivery accounts	Configuration → Manage Email delivery accounts	External SMTP delivery accounts — see *Alerting & third-party integration — in depth*
Manage S3 export accounts	Configuration → Manage S3 export accounts	Off-site backup S3 / S3-compatible targets — see *Off-site export to S3 (2.4.3)*
Manage Cribl connections	Configuration → Manage Cribl connections	Cribl connection accounts
Manage Webhook accounts	Configuration → Manage Webhook accounts	Webhook delivery accounts
Configuration Guardian	Configuration → Configuration Guardian	Configuration health checks and findings
Manage AI Provider accounts	Artificial Intelligence → Manage AI Provider accounts	LLM provider accounts — see *Configuring AI Provider accounts*

Use these native interfaces for configuration management. System settings and account management require a Splunk administrator-class user with TrackMe admin capabilities; see *Summary Requirements for TrackMe Administrators* above. Existing tenant RBAC and ownership are managed from the Virtual Tenant UI, independently of system-wide defaults.

Default configurations are located in the following configuration file:

```
trackme/default/trackme_settings.conf
```

The configuration user interface writes a `local/trackme_settings.conf`, which is automatically replicated across the members when running in a Search Head Cluster. Alternatively, you can deploy a `local/trackme_settings.conf` directly (in a Search Head Cluster, under `shcluster/apps/trackme/local/`). The recommended method as a basis is to configure TrackMe through the intended configuration user interfaces described below.

System settings

Configuration → System settings is a single page covering every system-level setting stored in `trackme_settings.conf`. It provides a docs-style left-hand navigation, colour-coded topic cards, a cross-setting search box, and a per-setting documentation modal (the blue information icon next to each field) that explains what the setting does and when to change it.



Only **changed** values are saved: edit any field, and the **Save changes** button reports how many sections have pending changes; the sidebar and topic cards flag exactly where the unsaved edits are.



The left-hand navigation groups the settings by stanza — each entry corresponds to a stanza of `trackme_settings.conf`:

Table 26: System settings sections

Section	Purpose
General	General behaviour: performance & backend (multi-threading, pagination, KV-collection mode, splunkd timeout, shadow filesystem cache), knowledge-object & RBAC defaults (default owner, admin/power/user roles for the tenant-creation UI), state events, stateful alerting & acknowledgments, email defaults, the Artificial Intelligence feature toggles, and integrations & licensing
Indexes	The default TrackMe indexes for Virtual Tenants (summary, metric, audit, notable) and the base index search filter. If you create tenant-specific indexes, use a strict prefix convention such as `trackme_<context>_<index name>`
User interfaces	UI defaults (`[trackme_ui_defaults]`): default theme (dark/light), auto-refresh, the action-progress refresh rate, the tenant-card detail level, the high-scale shadow page size, the topology graph entity counts, the maximum visible labels per entity and the volume unit labels. Users override the theme, auto-refresh, card detail level and unit labels from their Settings panel — see *Application-wide defaults*
splk-general	Splunk-specific defaults: feed thresholds and delays, the delayed inspector and auto-disablement, elastic and workload options, and CMDB integration defaults
splk-data-sampling	Options for the Data Sampling feature (splk-dsm / splk-feeds)
splk-outliers-detection	Options for the Machine Learning outlier-detection features (training & scheduling, detection defaults, thresholds & boundaries, algorithms & storage)
SLA	SLA classes and the SLA calculation / breach-events behaviour
Maintenance	The Maintenance KDB SLA-exclusion behaviour
Health Tracker	The per-tenant health-tracker task frequencies and the maximum cycle duration
Backup & Restore	Backup retention, the archive directory, and the active S3 export account — see *Backup & restore*
Topology Studio	Deployment-wide Topology Studio controls: version retention, live refresh interval, the default views display mode, icon and background limits, alert run history, snapshot and email-image ceilings — see *System-wide settings*
Bank Holidays	Default calendars and the effects of holidays on alerting, ML training, SLA and monitoring windows — see *Bank holidays*
Logging	The logging level for TrackMe (see *TrackMe Logging* below)

Tip

AI Assistant on the configuration pages

When the AI Assistant is enabled, an **AI assistant** button appears on System settings and on each account-management page (and inside their modals). It has full, context-aware knowledge of the configuration on screen — ask what a setting does, how to configure an account, or how to interpret a failed connectivity test. Configuration values are shared with the assistant with sensitive items redacted; secrets are never exposed.

Remote Splunk deployment accounts

Configuration → Manage Remote deployment accounts is where you configure any remote Splunk environment you monitor with TrackMe (Enterprise, Cloud, utility nodes). The full walkthrough — adding an account, the pre-save connectivity test, HA/DR endpoints, RBAC roles and automatic token rotation — is documented here: *Remote Splunk deployments*.



Email delivery accounts

Configuration → Manage Email delivery accounts defines the external SMTP delivery accounts TrackMe uses to send notifications (stateful alerting, AI Routines, Configuration Guardian). The local MTA (`localhost`) is always available without an account — only external mail servers need one. The full walkthrough is documented in *Alerting & third-party integration — in depth*.



S3 export accounts

Configuration → Manage S3 export accounts defines the S3 / S3-compatible targets used for off-site export of TrackMe backup archives. S3 export is optional and fully opt-in; the active target is selected in the **Backup & Restore** settings. The full walkthrough is documented in *Off-site export to S3 (2.4.3)*.



AI Provider accounts

Artificial Intelligence → Manage AI Provider accounts defines the Artificial Intelligence (LLM) provider accounts used by TrackMe's AI features (AI Assistant, AI Advisors, stateful-alert AI status reports). This is documented in detail here: *Configuring AI Provider accounts*.

Virtual Tenants configuration

Virtual Tenant records are created and deleted automatically by TrackMe when managing Virtual Tenants through the Web or REST API. Tenant-level configuration is edited from the **Configure tenant** modal, reachable from the per-tenant 3-dot menu on the Virtual Tenants page — see *Managing tenants*.

TrackMe Logging

The logging level for TrackMe is set from Configuration → System settings → Logging. All custom commands, REST endpoints, and any other TrackMe components rely on this setting to define the level of logging.

It is not recommended in a Production context to set TrackMe in DEBUG mode in normal circumstances as TrackMe will be extremely chatty in debug.

A typical logging message will look like: (INFO mode in this example)

```
2023-01-10 17:22:04,520 INFO trackmesplkflxparse.py stream 366 tenant_id=
↪"flx-demo-dma", context="live", TrackMeSplkFlxParse has terminated successfully,␣
↪turn debug mode on for more details, results_count="2"
```

The logging level is extracted at search time, via props.conf settings, example:

```
# catch all sourcetype
[(?::){0}trackme:custom_commands:*]
EXTRACT-log_level = \d{4}-\d{2}-\d{2}\s\d{2}:\d{2}:\d{2}\,\d*\s(?<log_level>\w*)\s
```

Therefore, you can review errors for instance with the following SPL search which would review both REST API endpoints errors and the custom commands:

```
(index=_internal sourcetype=trackme:rest_api log_level=ERROR) OR (index=_internal␣
↪sourcetype=trackme:custom_commands:* log_level=ERROR)
```

We strongly believe that the truth stands in the logs; therefore, we take great care at making sure logging in TrackMe is giving you the greatest level of quality and reliability!

See the following documentation for more about logging & troubleshooting in TrackMe:

- *Troubleshooting TrackMe*

8.10.3 Large-scale environments

TrackMe is designed to scale to many tenants and large entity counts. This page collects the practices that keep it efficient as it grows.

Design principles

- **Tenants are the scaling unit.** Design them around perimeters, teams, and permissions. Document your index naming conventions and top-priority perimeters first, then map tenants onto them.
- **Prefer many narrow trackers over a few broad ones.** Several concurrent *hybrid trackers* with non-overlapping scope are more cost-efficient, and easier to reason about, than one sweeping tracker.
- **Onboard progressively.** Start a tenant with DSM only, then add hybrid trackers one at a time, benchmarking each, rather than enabling everything at creation.
- **Lead with DSM.** Data Source Monitoring delivers the most value first; add DHM (heavier, needs maturity) and MHM (IT-Ops-specific) as you grow.
- **Use a service account**, not `admin`, so you can apply Workload Management and attribute cost cleanly (see *Configuration*).

Important

Dedicate a search-head tier to TrackMe at scale. On-premises, give TrackMe its own SH tier; on Splunk Cloud, use the ad-hoc tier. TrackMe must **never** actively track on the same search-head tier as a Splunk premium app such as Enterprise Security or ITSI — that configuration is not supported. Use *remote deployments* to reach data on other tiers.

How the heavy backends run: batches under a runtime budget

When you review TrackMe in the Splunk scheduler, the per-entity backends — adaptive delay, ML training, ML detection, per-entity maintenance — all follow the same execution model, and misreading it is the single most common source of confusion in a workload review:

the dispatch window is a trigger, not a bound. A tracker scheduled every 5 minutes with a `-5m` window is *not* promising to finish in 5 minutes, and a multi-minute run is *not* a runaway search. On each occurrence, the saved search selects a **batch of entities**, does the real work in a per-entity processing loop, **stops when it nears its runtime budget**, and resumes with the next batch on the following cycle.

Two practical consequences:

- **Run time reflects batch size, not the window.** The more entities a backend has to process each cycle, the longer its runs — that is expected behaviour for a large tenant, not a fault.
- **The cost is self-limiting.** These backends only pick up entities that need work (adaptive delay only reviews entities actually in a delay breach; ML training only fits models past their retrain interval). The cleaner and more stable the estate, the smaller the batches — the cost scales with how much of the estate is *unstable*.

See *adaptive delay* and the *ML backend & scheduling* for what one run of each actually does.

Reading the Splunk scheduler: continued vs skipped

When reviewing TrackMe's searches in the scheduler (`index=_internal sourcetype=scheduler`), two statuses matter and they are **not** equivalent:

- `continued` — the scheduler could not dispatch the search at its slot because the **concurrent-search limit** was reached, so it **deferred and re-attempted** it on the next cycle. The run is postponed, not lost. It is a saturation symptom of the stack, not a TrackMe fault — and TrackMe is resilient to it by design, re-attempting its own work rather than failing the run.
- `skipped` — the scheduler **gave up** on that occurrence. This is the status that genuinely drops work and deserves investigation.

Tip

Give TrackMe a bounded slice of concurrency. Run each tenant under a **dedicated service account** as the tenant owner, and place **role-based concurrent-search limits** on that account — restrictive enough that TrackMe cannot crowd out business or premium-app searches, but with enough headroom to operate (on the order of **10 concurrent searches**). TrackMe then consumes a predictable slice of the stack instead of competing on the shared pool. This is good Splunk hygiene independent of any footprint concern.

To measure completion, skipping, run times, and TrackMe's share of compute objectively, see *Measuring TrackMe's scheduler footprint* — current TrackMe releases also bundle these scheduler reports in the **Support - generate diags** tooling.

Reducing the footprint: which levers, in which order

When a workload review asks for reductions, apply the levers in value order:

1. **Upgrade to the current release.** The structural gains live there (native ML engine, caching and cadence improvements) — most later levers work better after it.
2. **Disable optional convenience backends you don't rely on.** Adaptive delay is the classic example: it is valuable but entirely optional, and disabling it at the tenant level automatically un-schedules its backend search — thresholds simply hold their current or policy-defined values (see *Adaptive delay (ML)*).
3. **Scope ML outliers to where behavioural change is meaningful** — rather than a blanket on/off (see *Outlier detection*).
4. **Align cadences to the current defaults.** Long-lived tenants can carry legacy schedules: TrackMe deliberately **never rewrites an existing saved-search schedule on upgrade**, so re-aligning custom crons to today's defaults is a quick manual win.

Note

Do not reduce the ML-monitor frequency to save cost — that opens scoring gaps and delays detection. The correct ML cost lever is model *scope* (which entities carry models), not monitor cadence.

Note

Per-entity maintenance, data sampling, and Guardian checks all add scheduled load — stagger their cadences, and keep an eye on skipped searches (see *Managing tenants*).

See also

- *Large-scale environments — in depth* — the end-to-end hybrid-tracker build, index-distribution SPL, cron budget math, and benchmarking.
- *Virtual Tenants explained* — the tenant model.
- *Hybrid trackers* — hybrid and replica trackers.

- *Remote Splunk deployments* — monitoring remote deployments.

8.10.4 Large-scale environments — in depth

Tip

This is the **in-depth reference** for running TrackMe at scale. For the readable overview, start with *Scaling*. This page gives the end-to-end Hybrid Tracker build procedure, the index-distribution SPL, the cron/skipping-search budget math, and the benchmarking workflow.

Introduction to High Scale in TrackMe

This documentation aims to provide best practice configuration steps for TrackMe, particularly in large-scale environments.

TrackMe is highly configurable, flexible, and designed for performance and efficiency at scale.

It scales from daily ingests in TB to dozens and much more, with limited and optimized computing costs when respecting good design practices.

However, adhering to best practices and considering specific design aspects are crucial for optimal results.

Scaling Concepts with TrackMe

TrackMe scaling concepts depend on various aspects of the solution, with some key points highlighted below.

TrackMe Virtual Tenants

Virtual Tenants are a core concept in TrackMe V2. In summary:

- Virtual Tenants act as virtual instances of TrackMe within the application.
- By using Virtual Tenants, you can define the scope and design of your TrackMe implementation.
- A Virtual Tenant orchestrates the lifecycle of all TrackMe knowledge objects, from KVstore collections to TrackMe trackers.
- Virtual Tenants are created on-demand by TrackMe administrators and can be disabled or deleted later.
- This feature allows scoping of Splunk data and entities created by TrackMe, using technical or functional concepts tailored to your preferences and context.
- Virtual Tenants support Role-Based Access Control (RBAC) and cater to a wide range of functional and business requirements.

Virtual Tenants offer capabilities beyond the “MSP” (Managed Service Provider) requirements, making TrackMe a unique Splunk application.

TrackMe Trackers

TrackMe implements a concept of Trackers, these are scheduled backend jobs created:

- There are different types of Trackers, depending on the components
- For Feeds tracking, for instance, you would create multiple Hybrid trackers to address the required data scope, with no overlap between trackers

Dedicated TrackMe Search Head Tiers

At very large scale, it is obviously beneficial to consider a dedicated Search Head tier for TrackMe, either standalone or dedicated:

- Keep in mind that TrackMe can use its remote search capabilities; therefore, it technically can deal with any remote deployment, on-premise or Cloud
- On-Premise Splunk customers at large scale should consider dedicating a Search Head tier for TrackMe
- Cloud customers will leverage the ad-hoc Search Head tier; some customers can also have multiple ad-hoc Search Head tiers and therefore dedicate a specific Search Head tier to TrackMe

In any case, keep in mind that TrackMe should never be actively tracking in the same Search Head tier as a Splunk Premium application, and that we do **not** support this scenario (Enterprise Security, ITSI).

Requirements for High Scale

TrackMe Splunk Service Account

It is a best practice to use a service account; this allows, for instance, to easily identify related search workloads, investigate the costs related to TrackMe, and so forth.

Using a service account rather than the default `admin` user when assigning searches also facilitates the implementation of Splunk Workload Management (WLM).

TrackMe implements a least privileges approach, consult:

- *Configuration*
- *Roles & access control*

TrackMe Remote Accounts

With its unique capabilities, TrackMe allows transparent execution of searches against remote Splunk deployments or standalone instances, from Search Head Clusters to Heavy Forwarders, or utility nodes such as Deployment Servers, License Managers, etc.

Depending on the components and your context, you may need as a prerequisite to get service accounts created on remote systems, a bearer token to be created, and remote accounts to be configured in TrackMe.

Consult:

- *Remote Splunk deployments*

Feeds Tracking (splk-feeds components family)

Feeds tracking is the original core concept of TrackMe, this is covered by 3 components:

- `splk-dsm` which is the main and most valuable component; we recommend focusing on this component first
- `splk-dhm` which tracks data by the event host; this is also a valuable component which can however be more expensive from a compute costs perspective, and requires more maturity
- `splk-mhm` which tracks metrics from the metric endpoint perspective; this component is more specific to IT-Ops related use cases

Hint

Recommendations

- Focus first on the Data Source tracking (splk-dsm)
- Be specific and address in priority the most critical indexes in Splunk
- For large scale environments, we recommend some preparation work to be performed; the indexes naming convention should be known and documented, as well as top priority indexes and perimeters

- Create Tenants according to your needs taking into account perimeters, teams and permissions, etc.
- Create empty tenant and proceed to the Hybrid Trackers configuration manually
- Create concurrent Hybrid Trackers to address specific scope of your environment; it is **more cost efficient** to have multiple concurrent Hybrid Trackers addressing restricted scope of indexes than just a few dealing with a huge amount of data

Create an Empty Tenant

Open the tenant wizard creation, and create the first tenant:

- Only enable the Data Source tracking component (splk-dsm)
- Do not create Hybrid trackers at the phase of the Virtual Tenant creation; instead, create trackers progressively processing benchmarks in the same time

You can create a brand new virgin tenant in a single line of SPL:

Replace the name of the tenant, the service account and roles as needed

```
| trackme url="/services/trackme/v2/vtenants/admin/add_tenant" mode="post" body="{
↪'tenant_desc': 'Feeds Tracking', 'tenant_name': 'feeds-tracking',
↪'tenant_roles_admin': 'trackme_admin', 'tenant_roles_user': 'trackme_user',
↪'tenant_owner': 'svc-trackme', 'tenant_idx_settings': 'global',
↪'tenant_dsm_enabled': 'true', 'tenant_dsm_sampling_obfuscation': 'disabled'}"
```

Notes: this creates a brand new tenant only with the splk-dsm component; components can be later on added and removed from existing tenants without having to delete and re-create the tenant.



Enter the new tenant

Once the new Virtual Tenant is created, enter the tenant and click on Manage Hybrid Trackers:



Then click on Create new hybrid trackers:



This is where Hybrid Trackers will be defined, tested, benchmarked and finally created.

Open a Search tab in parallel

Have a search tab handy, and let's run the following simple Splunk search:

indexes convention

- In large scale environments, it is a best and right practice to have strong naming conventions
- If for instance, your goal is to track security related data, perhaps all these indexes start by `sec_` or `siem_`

- Then, make sure to include these concepts in the searches thereafter

```
| tstats count where index=* earliest=-7d latest=now by index
| eventstats sum(count) as total_count
| eval percent=round(count/total_count*100, 2)
| fields - total_count
| sort - limit=0 percent
```

The point of this basic search is to identify the volume of events in each Splunk index, then calculate the percentage against the total number of events ingested in the past 7 days.

This in addition with the knowledge of your context and requirements should drive the creation of Hybrid Trackers.

For instance, let's assume that majority of events is represented by

- Firewall related data (Palo Alto, etc)
- System logs related data (Windows eventlogs, etc)
- Cloud security logs (such as AWS, GCP, etc)
- Anything else

In a large scale environment, a likely scenario could be create 4 hybrid trackers, this could for instance results in basically 4 indexes constraints:

Table 27: Hybrid Tracker roles and indexes constraints

Role	Indexes constraint
Firewall	`(index=sec_firewall_*)`
System logs	`(index=sec_oswin_* OR index=sec_osnix_*)`
Cloud	`(index=sec_cloud_*)`
Other data	`index=sec_* NOT (index=sec_firewall_* OR index=sec_oswin_* OR index=sec_osnix_* OR index=sec_cloud_*)`

This might well be a first shot that you can review and update while you are running into the configuration, this however provides a clear and easy picture of what to do.

Define, test, benchmark and validate Trackers

Back in the Hybrid Tracker creation user wizard, let's define our first Hybrid Tracker, test and benchmark:

Provide a meaningful name, this will ease the management over time:



*Replace the entire search constraint, TrackMe will store this into a macro associated with the Hybrid Tracker which can be updated as needed later on:**



Review the break by statement, this all depends on your requirements, the default means index + ":" + sourcetype:



Review additional break by, earliest and latest time quantifiers, for high scale TrackMe licensed customers, these are generally the best cost efficient parameters, users using the Foundation Edition trial are limited to eight Hybrid Trackers in total and can restrict these furthers to reduce the run time of the trackers:





The cron by default is set to every 5 minutes, this is the recommended configuration also for high scale environments as long as the Hybrid Trackers are properly designed:



Notes:

- A crontab of 5 minutes means we have 300 seconds for the tracker to be executed before starting to generate skipping searches
- Skipping searches is the indicator that the configuration is not optimal and that the Hybrid Tracker has too much to be done for a single search instance

Finally, we can test the search, this test only returns a subset of entities found and gives us a first indication of the performance as well as the content that will be retrieved:



Now, let's return the benchmark search and run the benchmark test, our search needs to complete in well less than 300 seconds according to our cron time, ideally max approximately 200 seconds:





*At this stage, if the search run time super fast, this means that the index scope can safety be increased so you can reduce the total number of Hybrid Trackers, but remember that is much more cost efficient to run concurrent smaller Hybrid trackers than just a few very large:**



Notes:

- In some circumstances and in Splunk Cloud, the benchmark search can reach the Nginx timeout (which is picky picky)
- In that case, you can simply take the benchmark search out and run it in in a search with associated earliest and latest quantifiers
- The Benchmark search represents the most expensive part of the search logic that TrackMe will orchestrate

Once we are happy with the design and performance, we can validate the creation of the Hybrid Tracker:



Review Hybrid Trackers Performance

After all Hybrid Trackers were created, and after a few executions, consult the Performance deepdive dashboard: (menu Audit & Troubleshoot)

Types of trackers

- TrackMe has various types of trackers depending on the components and their mission
- Not all share the same type of schedule; some trackers like the ML related trackers are designed to run over long periods of time on a less frequent schedule, dealing with TrackMe own metrics and automatically stopping as needed
- You want to focus on the Hybrid Trackers





Other components and additional considerations

General Tracker Design

As a general rule, all TrackMe components and underneath associated trackers obey the same design:

- Trackers should complete in a runtime that is consistent with their cron schedule
- Trackers should not overlap; this means that for instance with feeds tracking, trackers should look at specific set of indexes but unless using a different break by, there are no reasons why these would share the same indexes
- Trackers systematically report their runtime performance and report failures to the component register
- Trackers are component specific; the activity of feeds tracking is not the same as Workload or Flex Trackers, but the philosophy remains similar

Review and Monitor Virtual Tenants Activity

TrackMe carefully reports any issue affecting the Virtual Tenants and trackers; for instance, concurrent execution being reached on the instance would lead to failures easily visible:


Virtual Tenants
Virtual Tenants (legacy)
Configuration
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
TrackMe - Virtual Tenants
Monitoring the great value in your Big Data
Edition
Unlimited edition
NUMBER OF ACTIVE TENANTS
17
16
SCHEDULER COMPLETED PERCENTAGE
TENANT OPS STATUS
01 - services-mon
02 - secops feeds
03 - endpoints feeds
04 - cluster-monitor
05 - shc-monitor
06 - use-cases-monitoring
07 - hf-monitoring
08 - ds-clients-monitoring
09 - Cribl Monitoring
10 - soar-monitoring
11 - environment-controls
13 - uf-monitoring
14 - license-usage
15 - volume-variations-detect
16 - Fields Quality


TrackMe Scheduler Review & Investigate (past 24 hours)
completed
Status: completed 100%
Status: skipped >0%
Status: all
report
audit
logs
perfs
Tenant ID
Status
Report
% Completed
Count execution
Count completed
Count skipped
cluster-mon
completed
cribl-mon
Open in search all skipping events
Open audit Trackers performance view
Close



Measuring TrackMe's scheduler footprint

What ultimately matters in a workload review is **whether searches complete, how long they take, and what share of the platform's compute they consume** — all three can be measured directly rather than inferred. Current TrackMe releases bundle these scheduler reports in the **Support - generate diags** tooling; the underlying searches also work standalone on any deployment.

Scheduler skipping report

Per TrackMe saved search, how many runs *completed* versus were *skipped* or *deferred* (`continued`), and the completion percentage — the objective way to separate a tracker being deferred and re-run (harmless) from a run genuinely dropped:

```
search (index=_internal sourcetype=scheduler app="trackme")
| eval status=case(status=="success" OR status=="completed", "completed",
                   status=="skipped", "skipped", status=="continued", "deferred")
| stats count(eval(status=="completed")) as count_completed,
        count(eval(status=="skipped")) as count_skipped, count by savedsearch_name
| eval pct_completed=round(((count_completed / count) * 100),2)
| sort 0 - count_skipped
```

Note that `continued` is reported as *deferred*, not skipped — this immediately confirms whether any TrackMe search is actually being dropped (see *continued vs skipped*).

Scheduler performance report

The average / p95 / max run time per TrackMe search — the performance baseline to work from:

```
search (index=_internal sourcetype=scheduler app="trackme" run_time=*)
| stats avg(run_time) as avg_run_time, max(run_time) as max_run_time,
        perc95(run_time) as perc95, min(run_time) as min_run_time by␣
↪savedsearch_name
| sort 0 - avg_run_time
```

Validating the compute share through the SVC lens (Splunk Cloud)

On Splunk Cloud, the meaningful currency is the **SVC** — CPU-hours of `tstats` / `mstats`-dominated searches do not translate linearly into SVC consumption. This search (from TrackMe's own *SVC usage audit* view) charts, hour by hour, the percentage of the stack's total consumed SVC attributable to TrackMe, and works standalone on any Splunk Cloud stack with the CMC summary data:

```
(index=_cmc_summary OR index=summary) source="splunk-svc-search-attribution"␣
↪svc_usage=*
| fields svc_usage svc_consumer svc_consumption_score search_type search_app␣
↪search_label
  search_user search_head_names unified_sid process_type
| fillnull value="" svc_consumer process_type search_provenances search_type␣
↪search_app
  search_label search_user unified_sid search_modes labels search_head_names␣
↪usage_source
| stats max(svc_usage) as utilized_svc by _time svc_consumer search_type search_app
  search_label search_user search_head_names unified_sid process_type
| bucket _time span=1h
| stats sum(utilized_svc) as svc by _time, search_app
| eval search_app=if(isnull(search_app) OR search_app=="", "NA", search_app)
| eventstats sum(svc) as total_svc by _time
| eval pct=svc/total_svc*100
| search search_app="trackme"
| fields _time search_app pct
| timechart span=1h first(pct) as pct by search_app
```

See also the white paper *Tracking Splunk Cloud SVC consumption in TrackMe*.

Alerting Architecture in TrackMe

We recommend defining an Alerting Architecture based on TrackMe Notable events which provides several advantages, especially at large scale:



However, both options are valid design options, which can be implemented in TrackMe in a straightforward and flexible way.

See *Stateful alerting*

8.10.5 Maintenance mode

Maintenance mode suppresses alerting during planned (or unplanned) work, and excludes that time from SLA accounting. TrackMe offers it at three scopes, plus a persistent record that drives the SLA exclusion.

The three mechanisms

Mechanism	What it does
Global / tenant maintenance mode	A runtime on/off switch that silences alerting for the whole instance or one tenant. It suppresses stateful-alert transitions and auto-ack actions — but the decision maker keeps running, so you still *see* red state in the UI. It is alert **silencing, not sleep**.
Per-entity maintenance	Forces one entity (or a bulk selection) into the **blue**, non-alerting state for an explicit window. It has **top precedence** over computed state and every other protection layer, and auto-returns to normal when the window expires.
Maintenance Knowledge Database (KDB)	A persistent log of windows — planned or unplanned, past or future — by scope. SLA accounting reads the KDB (it needs historical intervals), not the live switch.

Enabling then disabling maintenance mode automatically writes a **planned** KDB record, so your SLA stays honest with no extra steps.

How it interacts with SLA

SLA exclusion reads the **KDB**. Only `active` and `expired` windows are excluded — not future `scheduled` ones — and, by default, only **planned** windows (a global setting changes this). A background tracker flips windows `scheduled → active → expired` every few minutes and writes the audit trail.

Note

- **Always set a timeout.** Global maintenance left on without an end time stays on indefinitely.
- Re-enabling does **not** retroactively fire alerts missed during the window; an open stateful record may need manual cleanup.
- "SLA still counting downtime" almost always means the window's scope did not match the entity.
- Managing maintenance requires admin capability; reading the KDB is available to all TrackMe users.

Under the hood, maintenance status is driven by the `trackme_maintenance_mode_tracker` tracker, and the global setting `maintenance_kdb_exclusion_behaviour` controls whether only *planned* maintenance is excluded from SLA. Global maintenance can be toggled from the REST API — `/services/trackme/v2/maintenance/global_maintenance_enable` and `…/global_maintenance_disable` (both `mode=post`), accepting `maintenance_duration`, `update_comment`, and an optional `tenants_scope` to limit the window to specific tenants:

```
curl -u user␣
↪https://mysplunk:8089/services/trackme/v2/maintenance/global_maintenance_enable \
    -X POST -d "{\"maintenance_duration\": \"3600\", \"tenants_scope\": \"secops\",
                \"update_comment\": \"1-hour maintenance for the secops tenant\"}"
```

See also

- *SLA policies* — SLA accounting and exclusions.
- *The entity state machine* — the blue state per-entity maintenance forces.
- *Bank holidays* — the calendar-driven counterpart.

8.10.6 Bank holidays

Bank holidays tell TrackMe when a quiet feed is expected. You keep **calendars**: the holidays of a country, a state or a region, plus your own closures. Each **tenant** uses the calendars that apply to it. While one of its holidays is active, the tenant holds new alerts, and the time counts neither against its SLA nor in its ML models. The findings tenants (User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge) use the same calendars to hold the new findings of the policies you choose.

Bank holidays are **built on calendars** since TrackMe 2.4.19:

- **Every country of the holiday catalog**: 207 countries and 628 countries, states and regions ship with TrackMe.
- **Rules, not dates.** A calendar holds rules, and TrackMe computes the holidays every year, so floating holidays always land on the right day: Easter Monday, Thanksgiving, the last Monday of May, the substitute day when a holiday falls on a weekend.
- **Per tenant.** Each tenant chooses its calendars and which of its effects apply.
- **Findings tenants too.** Every findings tenant — User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge — takes the effects that apply to it, and each of its policies chooses whether it holds new findings on a holiday.
- **Closures always go through.** During a holiday new alerts are held, but an alert that recovers still closes.

The page is under **Maintenance & holidays → Manage Bank Holidays**. It has two tabs, **Calendars** and **Tenants & settings**, and its own AI Assistant. It is meant for TrackMe administrators: everyone can read the calendars, only an administrator can change them.

Calendars

A calendar holds holiday **rules**, not dates: `easter 1` is Easter Monday, `4th thursday in November` is Thanksgiving. TrackMe generates the holidays of every year from these rules, in the calendar's own time zone, from two years back to two years ahead. Nobody has to re-import anything when the year changes.



There are two kinds of calendar:

- **Imported — Import from the catalog** picks a country, then optionally a **state / subdivision** and a **region**, each level only where the catalog has one (**Whole country** and **Whole state** keep the level above). The dialog shows how many holidays the location has, offers a **Time zone** choice when the location spans several (the United States, for instance), and lists the **day types that count as holidays** with the number of holidays of each type here. You can also count partial days and give the calendar a name.
- **Custom — Create a custom calendar** for what no country publishes: company closures, bridge days, the closure of one site. A custom calendar has a name, a description, a time zone (UTC by default), the day types it counts, and an optional country code, used for display and grouping only.


Import holidays from the catalog
Pick a country and, where the catalog has them, a state / subdivision and a region. The calendar stores the location's holiday RULES and regenerates its holidays every year: floating holidays (Easter Monday, the 4th Thursday of November...) always land on the right day.
Country
Germany (DE)
State / subdivision (optional)
Bayern (BY)
Region (optional)
Whole state
Germany - Bayern (DE-BY): 35 holidays
Time zone Europe/Berlin
Calendar name (optional)
Defaults to the location name.
Day types that count as holidays
Public (13)
Bank (2)
Holidays of the other types are listed as informational and do not hold alerts. The number of holidays of each type in this location is shown.
Count partial days (a holiday starting during the day) as holidays
Cancel
Import



What counts as a holiday

Each calendar **counts** the day types you choose: **Public** and **Bank** holidays by default. The others (**Optional**, **School**, **Observance**) stay listed as **informational**: greyed out, they change nothing. A **partial day**, a holiday that starts during the day such as Christmas Eve from 12:00, is informational too, unless the calendar counts partial days.

A holiday runs from midnight to midnight **in the calendar's time zone**. Two sites in different time zones need two calendars.

The **year view** of a calendar shows which days count, and the list below it gives every holiday from two years back to two years ahead. A calendar card shows **Holiday today** when one of its holidays is active.



Rules

Add a rule builds a rule without writing it, and previews this year's and next year's dates before you save:

Kind	What it gives
A day or a period	Pick the days on a month calendar: click a day, then a later day to extend it to a period. Tick **Every year on the same day(s)** for a yearly holiday. For a single day, **On a weekend** chooses **No change**, **Moves to the next Monday**, or **Saturday to Friday, Sunday to Monday**, optionally keeping the weekend day as well.
Relative to Easter	A number of days from Easter Sunday (or Orthodox Easter).
Nth weekday of a month	The 1st to 5th, or the last, weekday of a month.
Weekday before / after a date	For instance the Monday after a given date.
Advanced (rule text)	A rule in the grammar of the `date-holidays` library.

An optional **Starts at** time makes the rule a partial day.



Rule text examples:

Rule	Holiday
`12-24 P8D`	An 8-day year-end closure from 24 December, every year
`2026-12-28`	One day, once
`easter 1`	Easter Monday
`4th thursday in November`	Thanksgiving
`01-01 and if sunday then next monday`	New Year's Day, moved to Monday when it falls on a Sunday



The holidays of an imported calendar come from the catalog: you cannot edit their rule, but you can rename them, change their type, or **disable** them with the toggle of the rules table. Disable a holiday rather than deleting it: the next **Refresh from catalog** would bring a deleted one back. Editing a rule changes past holidays too (within the two years kept), and the SLA history of those days follows.

Managing calendars

- **Settings → Enabled** turns a calendar off: it keeps its rules but generates no holiday and applies to no tenant.
- **Refresh from catalog** brings an imported calendar up to date with the catalog shipped with the installed TrackMe. The message names the holidays added and removed; the holidays you disabled stay disabled and your own rules stay. A calendar does not change by itself after an upgrade.
- **Delete calendar** removes it; its holidays stop applying at once.

Limit	Value
Calendar name	120 characters, unique
Rules per calendar	1000
Length of a period	366 days

Tenants

The **Tenants & settings** tab sets which calendars apply where:

- the **system defaults**: by default every calendar, including the ones you add later;
- each **tenant** either **inherits** the default calendars, uses **selected** calendars, or uses **none** (holidays never apply to it).

The table lists every tenant with its calendars and its effects, each marked as inherited or set for the tenant. A findings tenant is tagged with its type (**UAM**, **Cribl Stream**, **Cribl Edge**, **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet** or **Splunk Forwarder Collectors**) and shows "—" for the effects that do not apply to it (see *Findings tenants*). **Edit** changes one tenant. The same options are in the tenant's own configuration: **Virtual Tenants → tenant menu → Configure tenant → Bank Holidays**.



Note

One tenant, one set of calendars. There is no per-entity calendar, by design: if some entities follow another country's holidays, they belong in another tenant.

What a holiday does

While a holiday of one of its calendars is active, a monitoring tenant (the DSM, DHM, MHM, FLX, FQM, WLK and VOL components) gets four effects. Each is **on** by default; turn one off for every tenant in the system defaults, or for one tenant with its **Edit** button. The findings tenants take only the effects that apply to them: see *Findings tenants*.

Effect	What happens during the holiday
Suppress new alerts	TrackMe stateful alerts open no new incident and send no unchanged update. State changes and closures still go through, so an incident that recovers is closed as usual. The other alerts built in Tenant Home (component technical alerts, SLA breach alerts) are held during the holiday.
Exclude from SLA	The time is excluded from SLA percentages, like planned maintenance, when the Maintenance KDB exclusion behaviour is `planned` (the default) or `any`; see *SLA policies*.
Exclude from ML training	Holidays inside a model's training range are left out of ML outlier training (DSM, DHM, FLX, FQM, WLK and VOL models), so a quiet day does not become "normal". When a tenant's calendars or this switch change, the affected models retrain once, automatically, within about 30 minutes.
Outside monitoring windows	Entities whose monitoring time policy is not *all time* (business days, 08h-20h, your own rules) are out of monitoring, like at night or on a weekend. Their status message reads *This entity is not currently under monitoring (bank holiday: <name>)*. Entities monitored all the time are not affected.

Holidays never change a threshold or a delay.

Alerts benefit from holidays when they chain the maintenance macro, which every alert built with the Tenant Home alert builder does. A custom alert search must chain the `trackme_apply_maintenance_mode_v2` macro to be held. Topology Alerts are not held by holidays. The findings of the findings tenants follow their own rule, below.

Findings tenants



The dedicated findings tenants — **User Activity Monitoring** (UAM), **Cribl Stream**, **Cribl Edge**, **Splunk HEC**, **Splunk S2S**, **Splunk UF Fleet** and **Splunk Forwarder Collectors** — take their calendars like every tenant: the system default calendars, unless the tenant selects its own or none, in **Tenants & settings** or in **Configure tenant → Bank Holidays**. They take only the effects that have something to act on, each **on** by default and inherited from the system defaults:

Effect	Tenant types it applies to
Suppress new alerts (shown as **Hold new findings**)	Every findings tenant
Exclude from ML training	Cribl Stream (its traffic ML models), Splunk HEC (the traffic models of its pools and tokens), Splunk S2S (the traffic models of its pools) and Splunk Forwarder Collectors (its collected-volume models). UAM, Cribl Edge and Splunk UF Fleet have no ML model.
Exclude from SLA, **Outside monitoring windows**	None: a findings tenant has no entity for them to act on

The **Tenants & settings** tab and the tenant's **Configure tenant → Bank Holidays** offer only the effects that apply; the others show "—" and are refused, from the API too.

Hold new findings. While a holiday of the tenant's calendars is active, the **policies set to hold** open no new finding. Each policy chooses in its editor, under **During bank holidays**: **Hold new findings** or **Keep opening findings**.

- A finding already open is still evaluated: it is updated, escalated, and it recovers and closes as on any other day.
- A held match does not reopen a resolved or dismissed finding.
- After the holiday, a condition that is still there opens its finding as on any other day (on every findings tenant except UAM, once the policy's consecutive failing checks are counted again).
- If the calendars cannot be read, nothing is held: the policies open their findings as usual.
- A policy set to keep opening findings is not affected by holidays.

The defaults hold what a holiday itself causes, or what can wait for the next working day:

Tenant	Holds by default	Keeps opening findings by default
UAM	The six inventory policies: the five scheduled search policies and the CSV lookup footprint. Nobody reviews them on a day off, and they are still there the next working day.	The activity and resource policies (unusual activity on a day off is more telling, not less), and the custom policies unless their definition says otherwise.
Cribl Stream	**Traffic outliers**: traffic on a holiday is expected to differ from its model.	Every other policy, so an infrastructure failure on a holiday still opens its finding. The Leader policies have no setting.
Cribl Edge	**Fleet connectivity** and **Fleet coverage**: laptops and desktops switched off for the holiday.	**Node connectivity** (the nodes you selected as always connected) and every other policy. Leader availability has no setting.
Splunk HEC	**Pool traffic drop**, **Token silence**, **Pool traffic outlier** and **Token traffic outlier**: traffic on a holiday is expected to differ.	Every other policy, so an endpoint failure on a holiday still opens its finding.
Splunk S2S	**Pool traffic drop** and **Pool traffic outlier**: traffic on a holiday is expected to differ.	Every other policy, so a receiver or intermediate failure on a holiday still opens its finding.
Splunk UF Fleet	**Agent disconnected**, **Mass disconnect**, **Fleet coverage** and **Phonehome drop**: machines switched off for the holiday.	Every other policy, including the expected agents (selected as always connected).
Splunk Forwarder Collectors	**Collected volume drop**, **Collected volume outlier** and **Mass silence**: less to collect on a holiday.	Every other policy, so a collector or input failure on a holiday still opens its finding.

The full list and where each run reports what it held are on the component pages: *User Activity Monitoring*, *Cribl Stream*, *Cribl Edge*, *Splunk HEC*, *Splunk S2S*, *Splunk UF Fleet* and *Splunk Forwarder Collectors*.

Exclude from ML training on a Cribl Stream, Splunk HEC, Splunk S2S or Splunk Forwarder Collectors tenant: its traffic or collected-volume models do not learn the hours of its holidays. This applies from the next training run

on: a model that learned a holiday before keeps it until it fades or is relearned, and nothing is relearned automatically when the calendars change.

Volume Outliers can also skip holidays in its working-day figures: none, the tenant's calendars, or calendars you pick. That choice follows the tenant's calendars, not the four effects, and is set in the Global license usage view (see *Global license usage (VOL)*).



System defaults

The defaults live in `trackme_settings.conf`, section `[bank_holidays]`. Change them in **Tenants & settings → System defaults → Edit defaults**, which applies at once, or in **Configuration → System settings → Bank Holidays**, which reaches alert suppression within about 30 minutes.

Setting (UI label)	Default	Meaning
`default_calendars` (**Default calendars**)	`*`	The calendars a tenant inherits: `*` for every calendar (including later ones), a list of calendars, or empty for none.
`suppress_alerts` (**Suppress new alerts**)	`1`	Hold new alerts during holidays; on a findings tenant, hold the new findings of the policies set to hold.
`exclude_ml_training` (**Exclude from ML training**)	`1`	Leave holidays out of ML training, and out of the traffic and collected-volume models of the findings tenants.
`exclude_sla` (**Exclude from SLA**)	`1`	Exclude holidays from SLA percentages.
`monitoring_windows` (**Outside monitoring windows**)	`1`	Treat holidays as outside the monitoring windows.
`legacy_kdb_copies_retention_` (**Delete the legacy holiday copies after (days)**)	`0`	Upgraded deployments only, see below. `0` keeps the copies.

Upgrading from an earlier version

Before TrackMe 2.4.19, holidays were a flat list of periods, each copied into the Maintenance KDB, and any active period held every alert of every tenant, closures included. At upgrade:

1. **The periods become calendars automatically**, within one health-tracker cycle (about 5 minutes) or at the first visit of the Bank Holidays page. A period on a holiday the country's catalog calendar knows becomes that calendar; the other periods become custom rules in calendars named `<Country> - additional days`, `Custom holidays (<cc>)` or `Custom holidays`. Drifted copies of floating holidays are dropped, and unusable periods are removed and reported in the audit trail. There is nothing to do.
2. **Same coverage on day one**: every tenant inherits every calendar, and the four effects are on. Alerts and SLA keep the holidays they had; closures now go through.
3. **Two effects are new**, and on: ML training skips holidays (the affected models retrain once), and entities on a business-hours monitoring policy are out of monitoring on a holiday. Turn them off in the system defaults or per tenant if you prefer.
4. **Stored SLA breach alerts** are updated to carry their tenant, so the tenant's calendars apply to them.
5. **Maintenance KDB copies**: the copies of migrated periods were removed. Older copies are kept, so past SLA percentages do not change. The **Tenants & settings** tab shows how many remain and, with **Delete the legacy holiday copies after (days)**, deletes the copies whose holiday ended more than that many days ago, plus the copies of future days (the calendars cover those). It asks for confirmation: deleting a copy removes the holiday exclusion from the SLA history of that day.
6. **Older API calls**: `admin/update`, `admin/delete` and `admin/delete_all` return HTTP 410; `admin/create` still works and adds a full-day rule to the `Custom holidays` calendar; `list` and `check_active` keep their response shape.

Configuration Guardian

A few holiday systems are not computed by TrackMe: Islamic, Hebrew, Persian, Chinese, Korean and Vietnamese lunar, Bengali, and equinox-based holidays. Their dates come from the catalog, which covers ten years ahead (to 2036 in TrackMe 2.4.19). The *Configuration Guardian* check `bank_holiday_library_dates_expiring` raises one alert per enabled calendar holding such holidays: a **warning** when its dates end within two years (tunable from 1 to 9), **critical** once they end this year. To resolve it, upgrade TrackMe and **Refresh from catalog**, or add the days as custom one-off rules. The alert clears by itself.

AI Assistant

When the AI Assistant is enabled, the Bank Holidays page has its own chat. It knows the calendars, their rules, the active and upcoming holidays and the system defaults, answers questions on the feature, and proposes changes (import a region, add a closure, disable a holiday) through the *Concierge*, for an administrator to approve.

Permissions and REST API

Reading calendars and holidays requires the TrackMe user capability (`trackmeuseroperations`); changing calendars, rules, defaults and tenant options requires the TrackMe admin capability (`trackmeadminoperations`): `trackme_power` cannot change Bank Holidays. Calendars are global, so every TrackMe user sees every calendar. Every change is audited in `index=trackme_audit` (object category `bank_holidays`). Bank holidays are available in every edition.

Everything the page does is available under `/services/trackme/v2/bank_holidays/`. The most useful endpoints:

Endpoint	Purpose
`GET list_calendars`	The calendars
`POST list_occurrences`	The generated holidays, by calendar, year or day range
`GET check_active`	The holidays active now; with `tenant_id`, for that tenant
`POST admin/import_calendar`	Import a country (`country`, optional `state` and `region`)
`POST admin/add_rule`	Add a rule to a calendar
`POST admin/update_tenant_scope`	Set a tenant's calendars (`inherit`, `none` or `selected`) and its effects (`inherit`, `1` or `0`)
`POST admin/regenerate_occurrences`	Regenerate the holidays of one calendar or of all, after restoring a backup

Import the holidays of Bavaria, then add a company day to a custom calendar:

```
curl -u username␣
↪https://mysplunk:8089/services/trackme/v2/bank_holidays/admin/import_calendar \
  -X POST -d '{"country": "DE", "state": "BY"}'

curl -u username␣
↪https://mysplunk:8089/services/trackme/v2/bank_holidays/admin/add_rule \
  -X POST -d '{"calendar_id": "<calendar_id>", "rule": "1st friday in July",
↪"name": "Company day"}'
```

Limit a tenant to two calendars:

```
curl -u username␣
↪https://mysplunk:8089/services/trackme/v2/bank_holidays/admin/update_tenant_scope␣
↪\
  -X POST -d '{"tenant_id": "emea-operations", "bank_holidays_scope": "selected",
↪"bank_holidays_calendars": "<calendar_id>,<calendar_id>"}'
```

Check what a tenant sees right now, from a search:

```
| trackme mode=get url="/services/trackme/v2/bank_holidays/check_active" body="{
↪'tenant_id': 'example'}"
```

Each endpoint describes its parameters with `describe=true`; the *REST API* page explains how to call them.

Troubleshooting

Symptom	What to check
A holiday did not hold an alert	The tenant's calendars (**Tenants & settings**); that the calendar is enabled; that the holiday's day type counts and it is not a partial day (greyed out = informational); that the alert was built in Tenant Home or chains the maintenance macro. `check_active` with the tenant shows what TrackMe sees now.
A findings tenant opened a finding during a holiday	That the policy is set to **Hold new findings** (its editor, **During bank holidays**) and that the tenant's **Hold new findings** effect is on. A finding that was already open is evaluated as usual. The policy's last run says what it held, or that the holidays could not be read.
A change in Configuration → System settings is not applied yet	Allow about 30 minutes for alert suppression; the page's own **Edit defaults** dialog applies at once.
Holidays are missing after restoring a backup	Run `admin/regenerate_occurrences`, or wait for the daily maintenance task.
A tenant still lists a deleted calendar	A deleted calendar stays in the tenant's selection and matches nothing: edit and save the tenant's options to clean it.
Time zones of a business-hours entity	The weekday and hour part of a monitoring time policy uses the search head's local time; only the holiday test uses the calendar's time zone.

See also

- *Maintenance mode* — maintenance windows, the planned counterpart of holidays.
- *SLA policies* — SLA accounting and exclusions.
- *VOL — Volume Outliers* — working days in the Volume Outliers license view.

8.10.7 Data sampling

Data sampling is TrackMe's **event-format recognition** engine. It periodically samples a *DSM* feed's events and matches them against format models, so you catch unknown formats, mixed formats, and format drift — a data-*quality* signal on top of the data-*flow* signals delay and latency give you. It is DSM-only.

How it works

A per-tenant tracker samples selected feeds every ~20 minutes, takes a window of events, and matches each against the configured models. It records the match percentage per model and identifies the **major model**. If the major model's match percentage falls below the acceptable threshold, the feed is flagged with a data-sampling anomaly. It detects:

- **unknown formats** — events that match nothing;
- **multi-format feeds** — events matching more than one model;
- **format drift** — the dominant format changed from the previous run.

Models

- **Out-of-the-box models** ship for common formats — syslog (BSD and RFC 5424), JSON, Splunk HEC JSON, XML, CSV, key-value, and more.
- **Custom models** are per-tenant regular expressions. An **inclusive** model says "a match means this *is* format X"; an **exclusive** model says "a match means this is explicitly *not* X", which is the tool for suppressing false positives.

Acceptance thresholds are set system-wide in the Configuration page.

Note

- **Small samples are noisy** — low-volume feeds may return too few events; widen the window.
- Greedy out-of-the-box regex can match events you wanted flagged — add an exclusive model to suppress, and keep custom patterns from overlapping or you get a "multi-format storm".
- Each sample runs a search — stagger cadences across many feeds.

See also

- *Data sampling — in depth* — the engine SPL, the anomaly status schema, all settings, and custom sampling rules.
- *DSM — Data Source Monitoring* — the DSM component data sampling extends.
- *Impact scoring & thresholds* — how data-quality breaches feed the impact score.

8.10.8 Data sampling — in depth

Tip

This is the **in-depth reference** for TrackMe's data sampling engine. For the readable overview, start with *Data sampling*. This page covers the two-phase engine and its SPL, the anomaly status schema and value tables, the per-entity and system-wide settings, and creating custom sampling rules with `trackmesamplingexecutor`.

Introduction to TrackMe's Events Format Recognition Engine

This documentation applies from TrackMe 2.1.0:

- TrackMe 2.1.0 welcomed the sampling engine v2.0, which is a major rewrite and improvement of the previous version.
- Therefore, this documentation applies to TrackMe 2.1.0 and later versions.

About TrackMe's data sampling features:

- TrackMe's data sampling engine is a component of the splk-dsm module, which is responsible for the recognition of events and formats in the data being ingested by Splunk.
- The data sampling engine feature provides automated quality assessment of the data being ingested by Splunk, at scale.
- It works by regularly sampling events and applying built-in and custom rules to recognize the format of the events, which is called inclusive matching.
- The engine categorizes events by model match, and tracks the percentage of events matched per model, identifying the **major model** of the data.
- The inclusive major model percentage is then compared against the minimum acceptable threshold, and if the percentage is below the threshold, the entity is marked as in data sampling anomaly.

How Does the Data Sampling Stand in TrackMe's Workflow

You can find the data sampling status and access to its management in TrackMe's entity view, in the **Data Sampling** tab:



Depending on the status of the data sampling, as well as the global Virtual tenant configuration, the sampling status may or may not influence the entity status.

You access to the full detailed by clicking on "Manage Data sampling" button:





High-Level Overview of the Data Sampling Process

In a very high-level overview, the data sampling process works as follows:

`Step 1: Data Sampling Tracker Execution and Entities to be Sampled Identification`

1. The Virtual Tenant's Data sampling tracker, which is a scheduled job called **trackme_dsm_data_sampling_tracker_tenant_<tenant_id>**, is executed every 20 minutes.

2. The scheduled backend starts by extracting the list of TrackMe entities to be sampled, using the following Splunk search:

Example:

```
| inputlookup trackme_dsm_tenant_mytenant where monitored_state="enabled"
| eval key=_key
| `trackme_exclude_badentities`
```

(continues on next page)

(continued from previous page)

```
| where data_last_time_seen>relative_time(now(), "-24h")
| lookup trackme_dsm_data_sampling_tenant_mytenant object OUTPUT␣
↪data_sample_feature, relative_time_window_seconds,␣
↪data_sample_last_entity_epoch_processed, min_time_btw_iterations_seconds,␣
↪data_sample_mtime
``` only consider entities where the last processed epoch␣
↪(data_sample_last_entity_epoch_processed) is older than data_last_time_seen, or␣
↪null (entities has not been processed yet) ```
| where (isnull(data_sample_last_entity_epoch_processed) OR␣
↪data_sample_last_entity_epoch_processed<data_last_time_seen)
| eval data_sample_feature=if(isnull(data_sample_feature), "enabled",␣
↪data_sample_feature) | where (data_sample_feature!="disabled")
``` only consider entities where the min_time_btw_iterations_seconds is older than␣
↪the current time (bigger or equal to the time spent since last run, or null for␣
↪new entities) ```
| eval time_spent_since_last_run=now()-data_sample_mtime
| where (isnull(min_time_btw_iterations_seconds) OR␣
↪time_spent_since_last_run>=min_time_btw_iterations_seconds)
``` define a priority rank, entities that have been set as disabled_auto should be␣
↪processed last compared to entities in disabled_audo ```
| eval priority_rank=if(data_sample_feature=="enabled", 1, 2)
``` order ```
| sort limit=0 priority_rank, data_sample_mtime
| fields object, key, data_last_time_seen, *
| eval earliest_target=if(isnum(relative_time_window_seconds),␣
↪data_last_time_seen-relative_time_window_seconds, data_last_time_seen-3600)
```

Notes:

- Some of these settings are defined at the system wide level, the search without variable values is the following, the example above is a simplified version of the search for a tenant named "mytenant"
- To see what is the actual search executed by TrackMe, look at:

```
index=_internal sourcetype=trackme:custom_commands:trackmesamplingexecutor
↪"Executing upstream definition search"
```

Step 2: Run the Data Sampling Process for Each Entity

1. For each entity to be sampled, the data sampling process is executed.
2. This process means that TrackMe identifies the search to be executed, which depends on the context of the entity as well as various settings such as the relative time window (against the last known event for that entity).
3. The search is executed, each event result from the search goes through the rules, customer rules if any, and the built-in rules.
4. The engine extracts a sample of events per model matched and per entity, which will be stored in the data sampling KVstore collection for investigation purposes.
5. Additionally, the engine performs various calculations, notably the percentage of events matched per model, and the major model of the entity.
6. Finally, the engine generates various metrics (KPIs) in the tenant metrics index, and updates the KVstore record defining the data sampling status, messages, and other metadata.

Notes:

- You can review the logs associated with the data sampling operation of a given TrackMe entity as easy as:

```
index=_internal sourcetype=trackme:custom_commands:trackmesamplingexecutor object="
↪<object value>"
```

Inspecting the anomaly status summary message

In this screen, TrackMe provides a summary of the data sampling status in JSON format, which includes various high level information so you can understand easily the current status, example:

```
{
    "state": "green",
    "desc": "No anomalies were detected during the last data sampling iteration.",
    "remediation": "N/A.",
    "last_run": "12 Sep 2024 05:02",
    "anomaly_reason": "no_anomalies_detected",
    "multiformat": false,
    "events_count": 10000,
    "min_time_btw_iterations_seconds": 3600,
    "pct_min_major_inclusive_model_match": 98,
    "pct_max_exclusive_model_match": 0,
    "max_events_per_sampling_iteration": 10000,
    "relative_time_window_seconds": 3600,
    "current_detected_major_format": "raw_start_by_timestamp %Y-%d-%m %H:%M:%S",
    "models_summary": [
        "raw_start_by_timestamp %Y-%d-%m %H:%M:%S | pct_match: 100.0 | type:␣
↪inclusive"
    ]
}
```

Data sampling statuses

field: state

The data sampling state can be reported as:

Value	Description
green	No anomalies were detected during the last data sampling iteration.
red	Anomalies were detected during the last data sampling iteration, if the Virtual Tenant configuration allows it, it will turn the entity red with anomaly_reason containing **data_sampling**.
orange	Since the discovery of the entity, TrackMe has detected quality issues and anomalies, TrackMe will continue performing sampling in case conditions change, but it will not affect the overall entity status.

The state is also reflected as the border colour of the data sampling tab in the entity view.

field: anomaly_reason

The data sampling anomaly reason can be reported as:

Value	Description
no_anomalies_detected	No anomalies were detected during the last data sampling iteration.
exclusive_rule_match	Anomalies detected, one or more exclusive rules have been matched.
inclusive_rule_match	Anomalies detected, quality issues were detected, the min percentage of the major model matched does not meet requirements which indicates that a too large number of events do not share the same format as the majority of events.
format_change	The major event format (the format previously detected for the majority of events) has changed from <model_name> to <model_name>, this might indicate a non expected quality issue or condition change in the ingestion of this feed in Splunk.
anomalies_at_discovery	Anomalies were detected since the entity discovery, multiple formats were detected and the major model is under the acceptable threshold of percentage of events matched by the major model. The data sampling feature was automatically disabled (disabled_auto) to avoid generating false positives for this entity (the feature will not be allowed to influence the entity status), however TrackMe will continue attempting to process in case conditions for this feed change.
anomalies_since_discovery	Anomalies were detected since the entity discovery, multiple formats were detected and the major model is under the acceptable threshold of percentage of events matched by the major model. The data sampling feature was automatically disabled (disabled_auto) to avoid generating false positives for this entity (the feature will not be allowed to influence the entity status), however TrackMe will continue attempting to process in case conditions for this feed change.

field: models_summary

This field is a list of the models detected during the last data sampling iteration, with the following information:

- The model name (model_name)
- The percentage of events matched by this model (pct_match)
- The type of match (inclusive or exclusive)

Example:

```
"models_summary": [
    "raw_start_by_timestamp %Y-%d-%m %H:%M:%S | pct_match: 100.0 | type: inclusive"
]
```

When multiple models are matched, the list will contain multiple entries, example:

```
"models_summary": [
    "raw_start_by_timestamp %Y-%d-%m %H:%M:%S | pct_match: 98.97 | type: inclusive
↪",
    "raw_start_by_timestamp %b %d %H:%M:%S | pct_match: 1.03 | type: inclusive"
]
```

fields: settings for this entity (min_time_btw_iterations_seconds, etc)

The following fields are settings for this entity, defined when the entity was discovered using system wide corresponding settings, and that can be updated on a per entity basis:

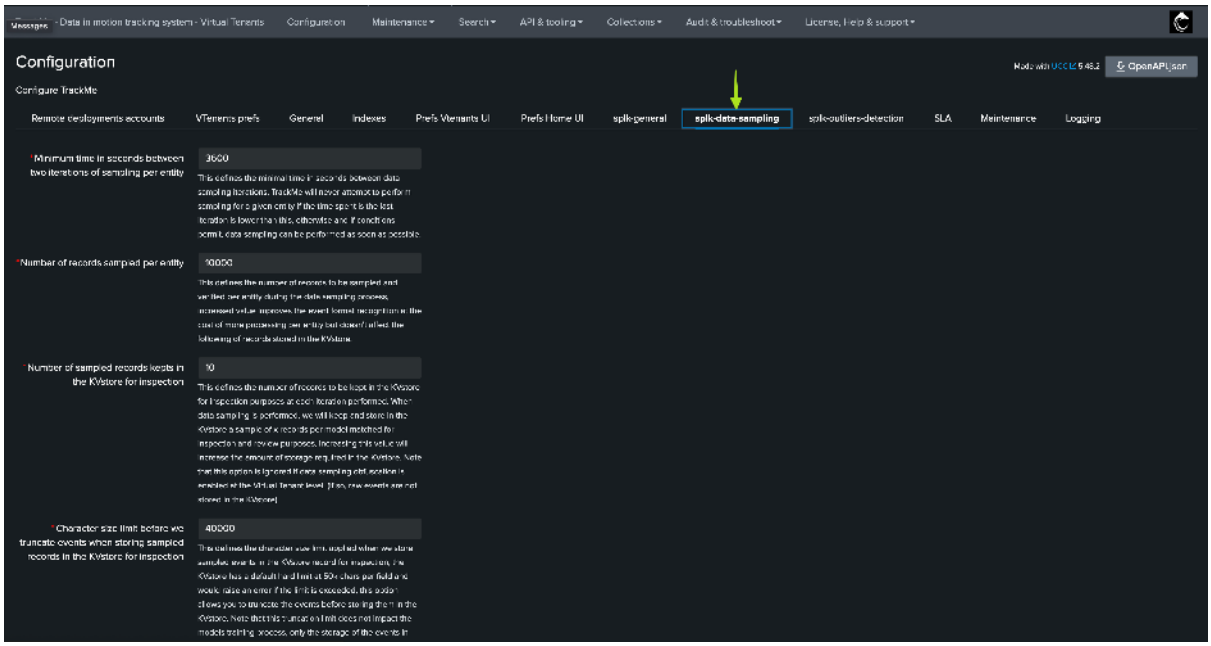
Value	Description	Default Value
min_time_btw_itera	Minimum time in seconds between two iterations of sampling per entity. TrackMe will never attempt to perform sampling for a given entity if the time since the last iteration is lower than this value.	3600
relative_time_window_s	The size in seconds of the time window for the sampling operation, relative to the latest event time known for the entity. This is used to calculate the earliest time for the sampling search.	3600
max_events_per_sa	Defines the number of records to be sampled and verified per entity during the data sampling process. Increasing this value improves event format recognition at the cost of more processing per entity.	10000
pct_min_major_incl	The minimal percentage of events to match the major inclusive model. If the main model matched has less than this percentage of events matching, the entity state will be impacted.	98
pct_max_exclusive_	Defines the maximum percentage of events matching an exclusive model that can be accepted per iteration. By default, no events matching an exclusive model are accepted.	95

Hint

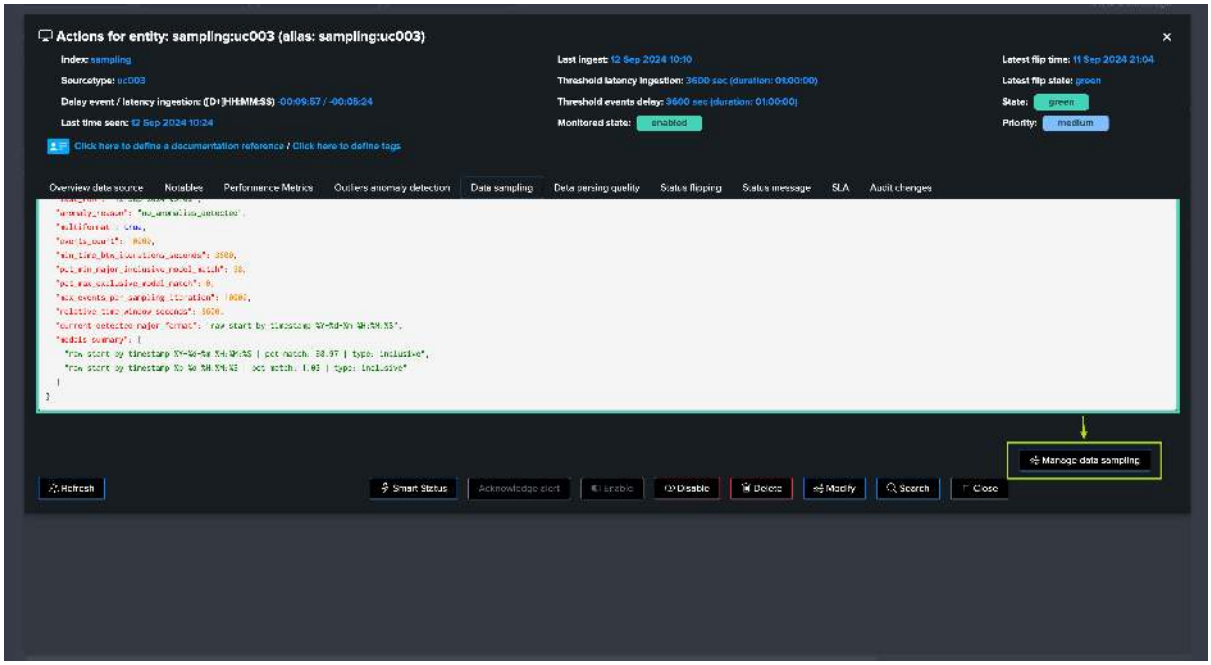
System level settings:

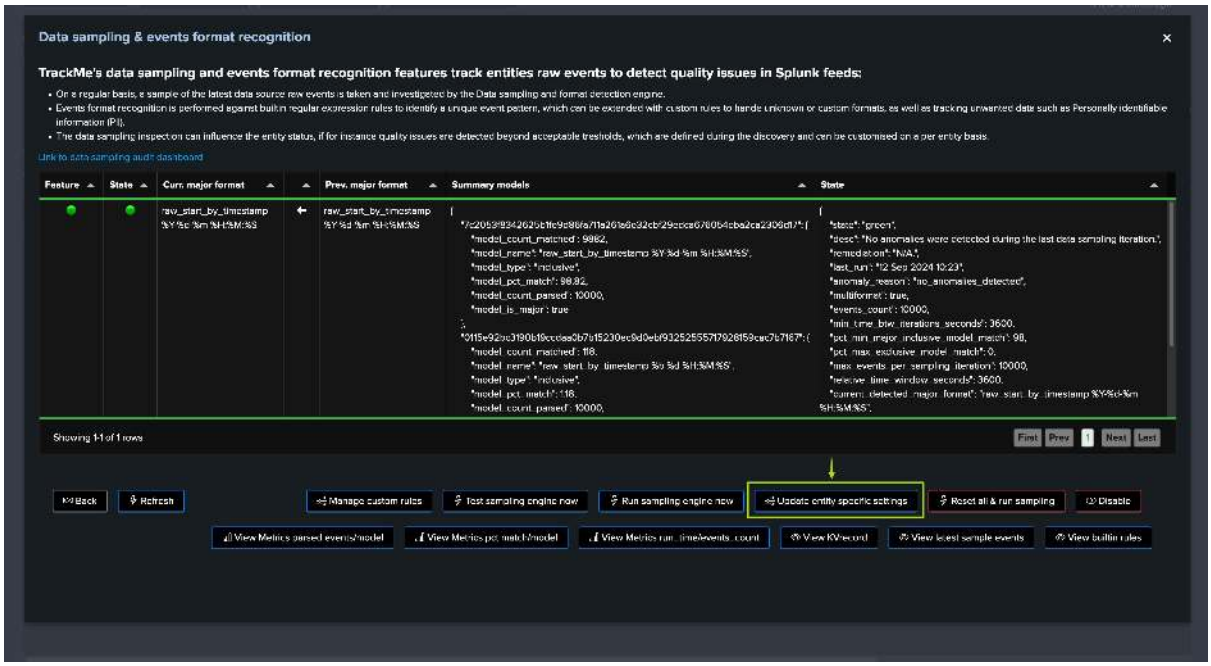
- These settings are applied at the level of TrackMe entities where the data sampling discovers the entity.
- Each of these settings can be updated on a per entity basis, and the entity settings will override the system level settings.

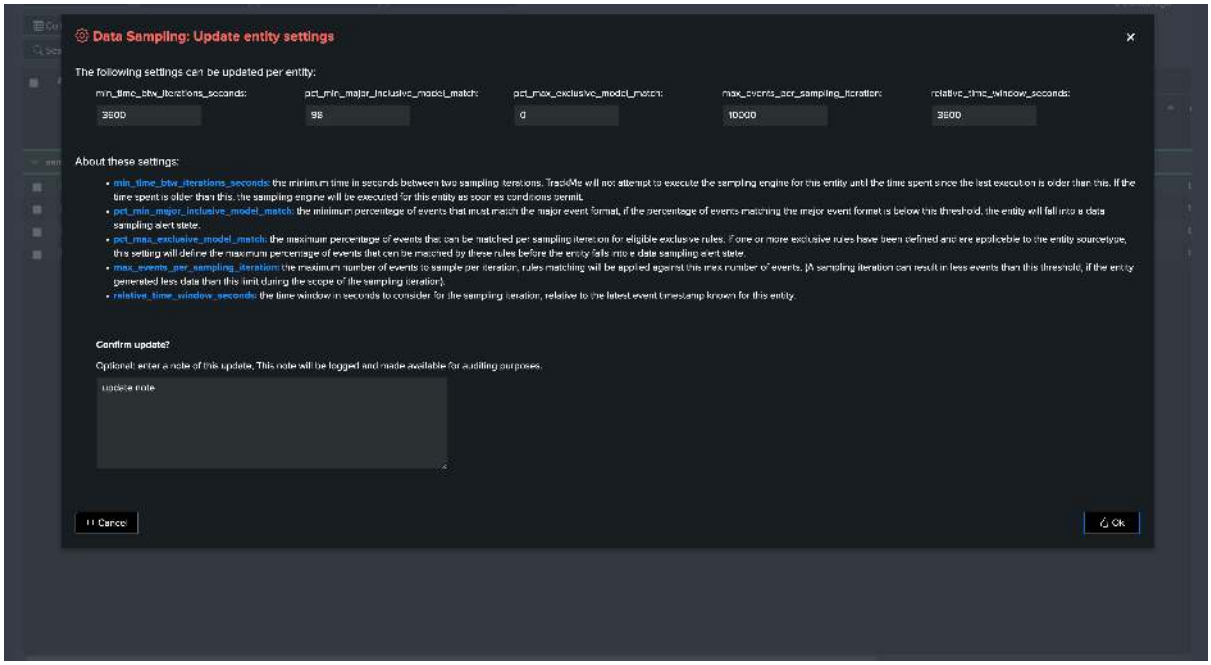
System level settings screen:



Entity level settings management screen:


Actions for entity: sampling:uc003 (alias: sampling:uc003)
Overview data source
Notables
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Manage data sampling
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close


Data sampling & events format recognition
TrackMe's data sampling and events format recognition features track entities raw events to detect quality issues in Splunk feeds:
Feature
State
Curr. major format
Prev. major format
Summary models
State
Showing 1-1 of 1 rows
Back
Refresh
Manage custom rules
Test sampling engine now
Run sampling engine now
Update entity specific settings
Reset all & run sampling
Disable



Inclusive and Exclusive Models Matching, Sourcetype Scope, Major Model and Thresholds

Hint

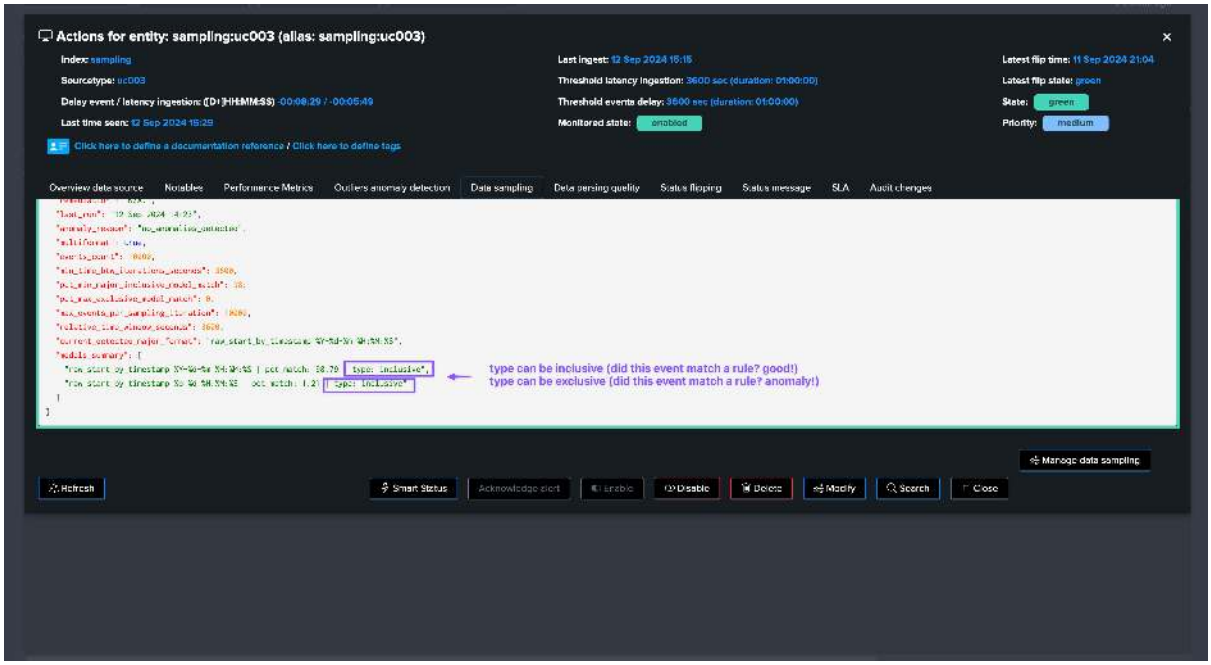
About these concepts:

- Built-in rules in TrackMe are always inclusive rules, with sourcetype_scope=*.
- For inclusive rules, when a given event matches a certain rule, the engine stops processing other rules for this event, therefore, an event can only match one rule.
- For exclusive rules, a given event can match multiple rules (as we basically look for “bad things” such as PII data), and the engine will track the percentage of events matching each exclusive rule.
- You can create custom rules which can be inclusive OR exclusive, but not both.
- Each rule is associated with a **sourcetype_scope**, which can be an explicit sourcetype, a list of sourcetypes, and as well any combination of wildcards for each.

Inclusive and exclusive models matching

Data sampling rules in TrackMe can be defined with two types of matching:

- `inclusive`: The rule is applied to the events for format recognition purposes, this means “does the rule regular expression match the event?”.
- `exclusive`: The rule is applied to the events, however, if the rule is matched this means that a bad condition is detected “this rule should never match events”.



Sourcetype scope

When TrackMe's data sampling engine processes an event, it will only apply the rules that are associated with the sourcetype of the event, relying on the sourcetype_scope:

- All builtin rules are inclusive type rules, with a sourcetype_scope of "*".
- Custom rules can be defined with a specific sourcetype_scope, which can be a single sourcetype, a list of sourcetypes, and wildcards can be used in each value submitted.
- When creating custom rules, the sourcetype_scope predicates which events will be processed by the rule, and which will not.

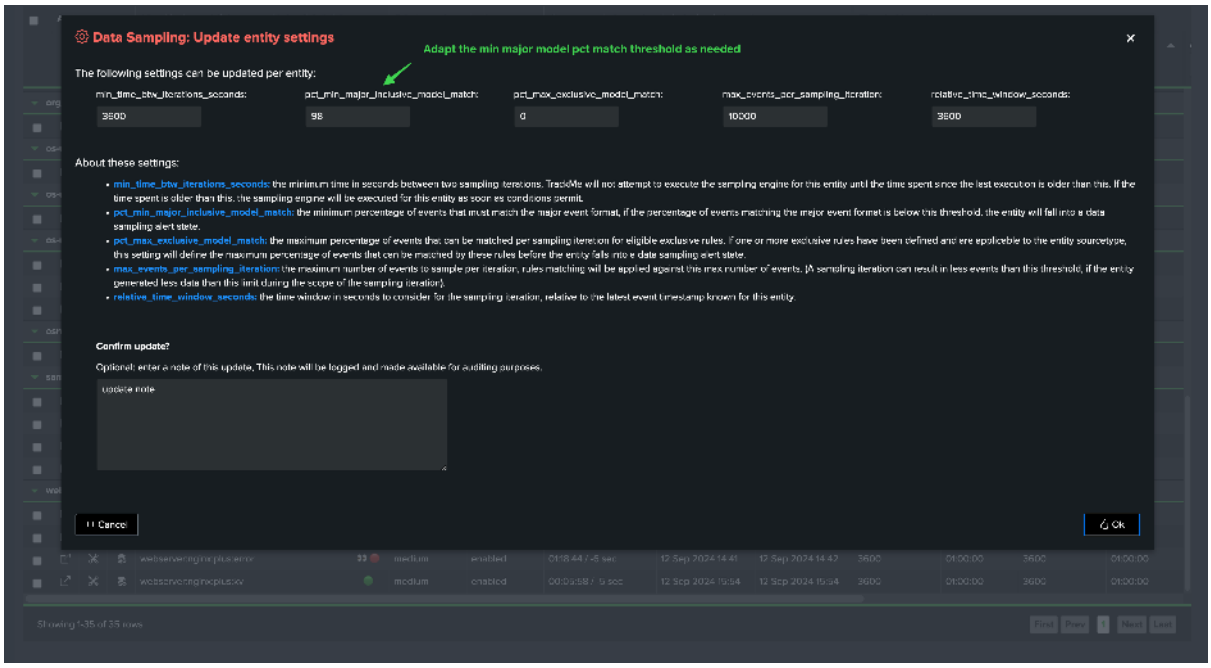
Major model

When the data sampling engine performs an iteration for a given entity, it calculates the percentage of events that match each model, and identifies the major model:

- The major model matters for the entity status, as it is the model that the majority of events match.
- This concept applies only to inclusive rules matching.
- During the sampling process, if the percentage of events matching the major model is below the threshold `pct_min_major_inclusive_model_match`, a sampling anomaly status will be raised as `inclusive_rule_match`.
- However, if this happens at the first iteration of the entity, the status will be raised as `anomalies_at_discovery`, and will not be allowed to influence the entity status.
- If from the second iteration, the issue persists, the status will be raised as `anomalies_since_discovery`, TrackMe will continue to attempt to process sampling for this entity in case conditions change, but it will not allow the entity status to be impacted.
- If later on in the life cycle of the Splunk feed, the percentage of events matched for a given model meets the minimal threshold, the status will be updated to `no_anomalies_detected` and it becomes eligible for data sampling.
- Then, if later on a quality issue affects that same entity, the sampling engine would raise an alert for this entity.

In short!

- The concept of major model is used to make the distinction between the models that represent the vast majority of events, so we can exclude minor pollution while focussing on the true events formats.
- We can then detect abnormal variations, such as a format change or a quality issue that is introduced in the feed, while at the same time avoiding false positives.
- Finally, the threshold can be customised per entity, so the engine can adapt to the specificities of all feeds with flexibility.



Max Events count per iteration, relative time period, and minimum interval between iterations

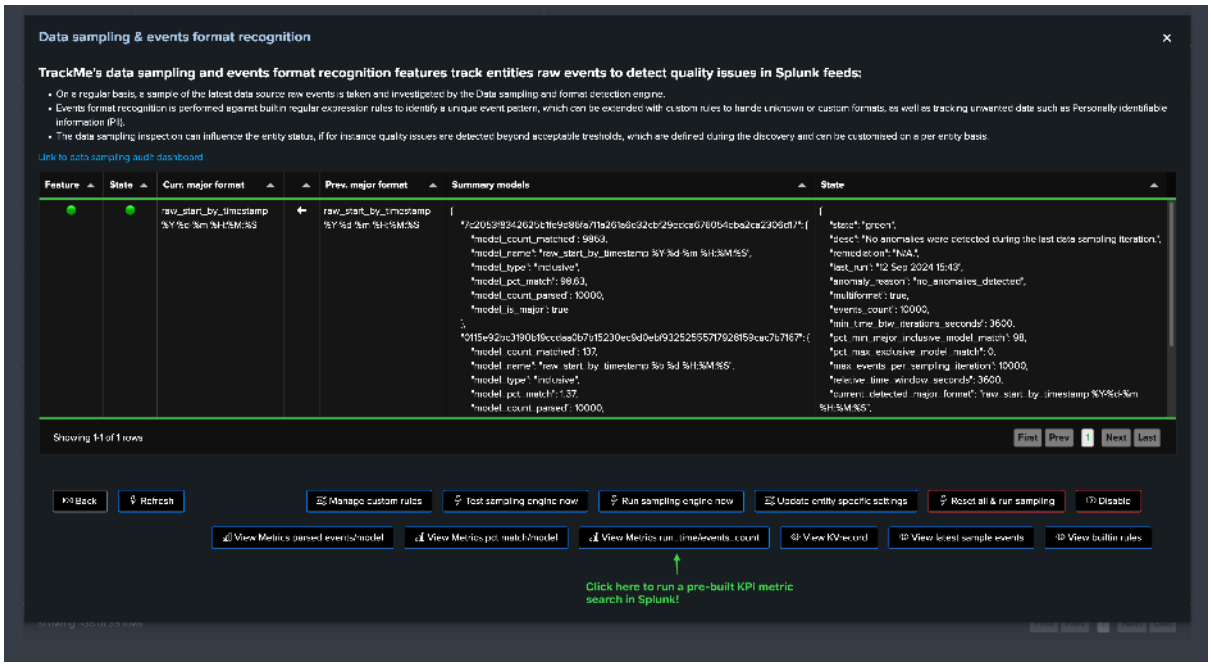
When the engine performs a data sampling iteration, it also takes into account the following concepts and settings:

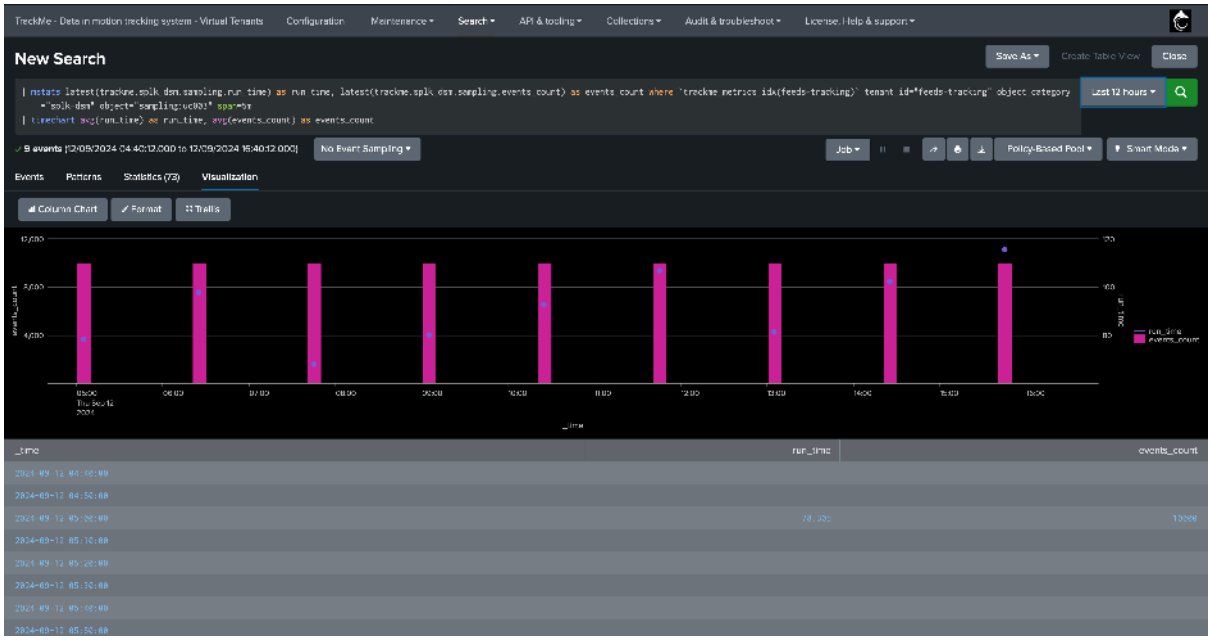
- **max_events_per_sampling_iteration**: This setting defines the maximum number of events that will be sampled per iteration for a given entity. Increasing this value will improve the quality of the data sampling, but will also increase the processing time.
- **relative_time_window_seconds**: This setting defines the time window for the sampling operation, relative to the latest event time known for the entity. This is used to calculate the earliest time for the sampling search. (In short, if the value is 3600 seconds, and the last event time known for the entity is 12:00, the search's earliest time will start at 11:00).
- **min_time_btw_iterations_seconds**: This setting defines the minimum time in seconds between two iterations of sampling per entity. TrackMe will never attempt to perform sampling for a given entity if the time since the last iteration is lower than this value.

About processing time, truncation and costs:

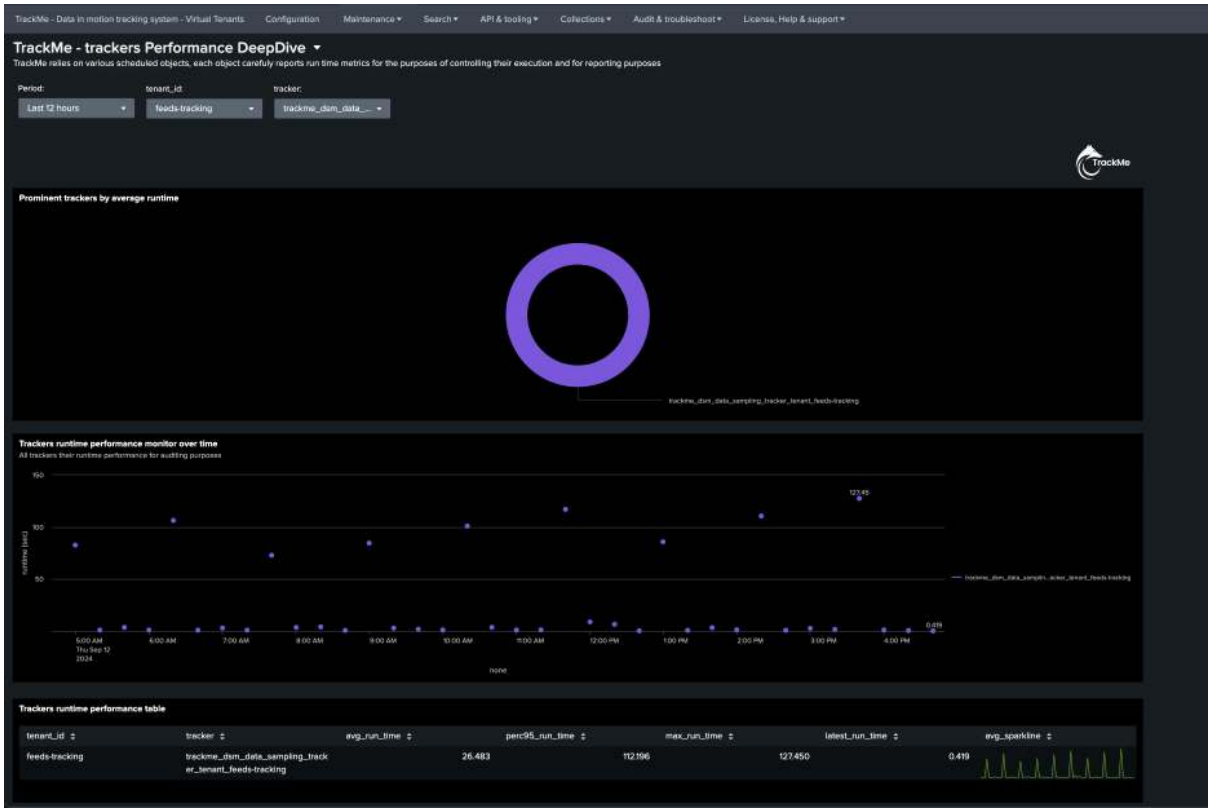
- **About processing time:** These settings will influence the processing time of each sampling iteration, the more events to be processed and the larger the time window, the more time it will take to process the entity.
- **About truncation:** The sampling engine is not affected by truncation, the regular expression is applied to the full events, truncation is only applied against a subset of the sampled events which we store in the KVstore for investigation purposes.

- **About costs:** The sampling engine generates run_time and processed events count KPIs, so you can easily track and understand the run time costs of the sampling operations.





The total run time costs of the data sampling tracker itself can be tracked using TrackMe's deepdive UI:



About the run time of the data sampling tracker!

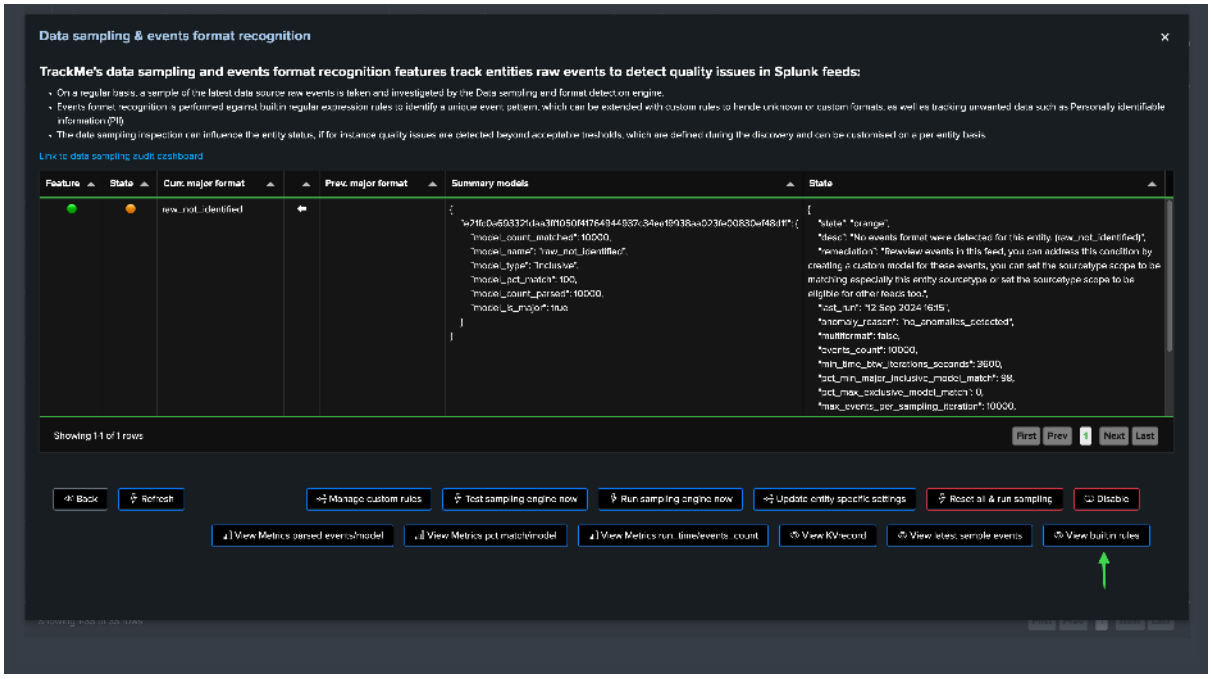
- The data sampling tracker is a scheduled job that runs every 20 minutes, and it will process all entities that are eligible for sampling.
- In fact, it will attempt to process as many entities as possible in its eligible time frame, and will stop automatically before generating skipping searches.
- At the next iteration, it will process the next chunk of entities, which are ordered based on their latest iteration, as well as a priority rank based on if a format was detected or not.

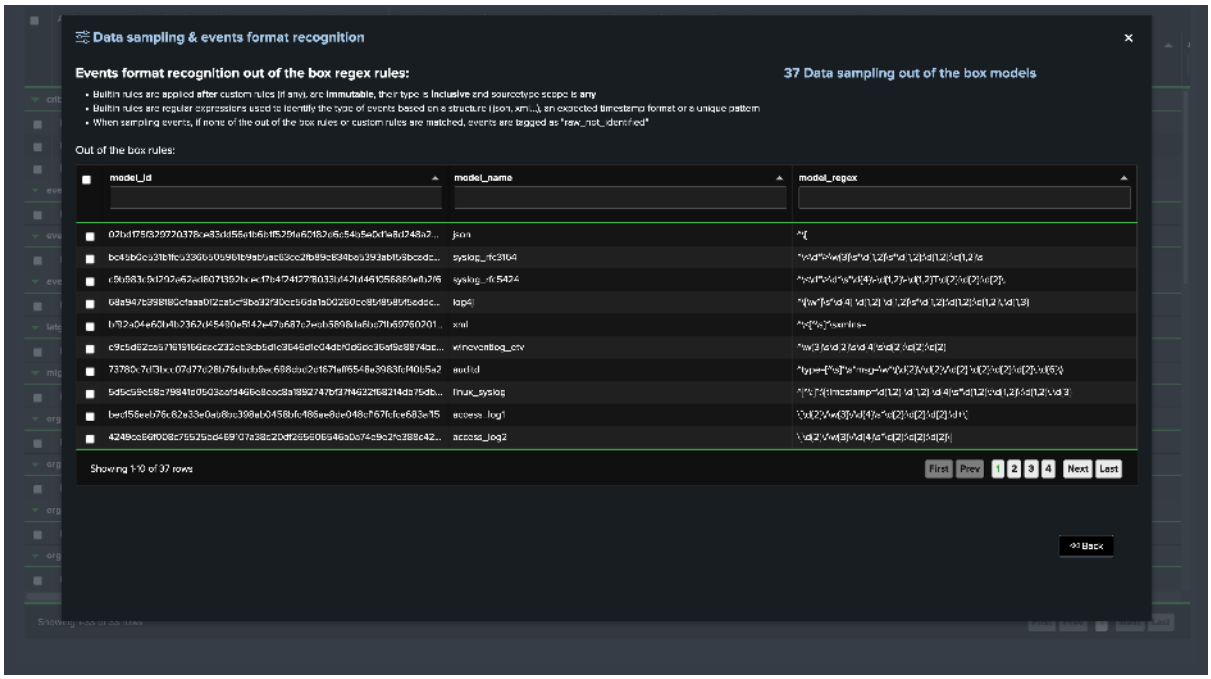
Out-of-the-Box Sampling Rules

TrackMe provides a set of built-in rules that are applied to the events during the data sampling process, you can access these through the sampling management screen:

About out-of-the-box rules:

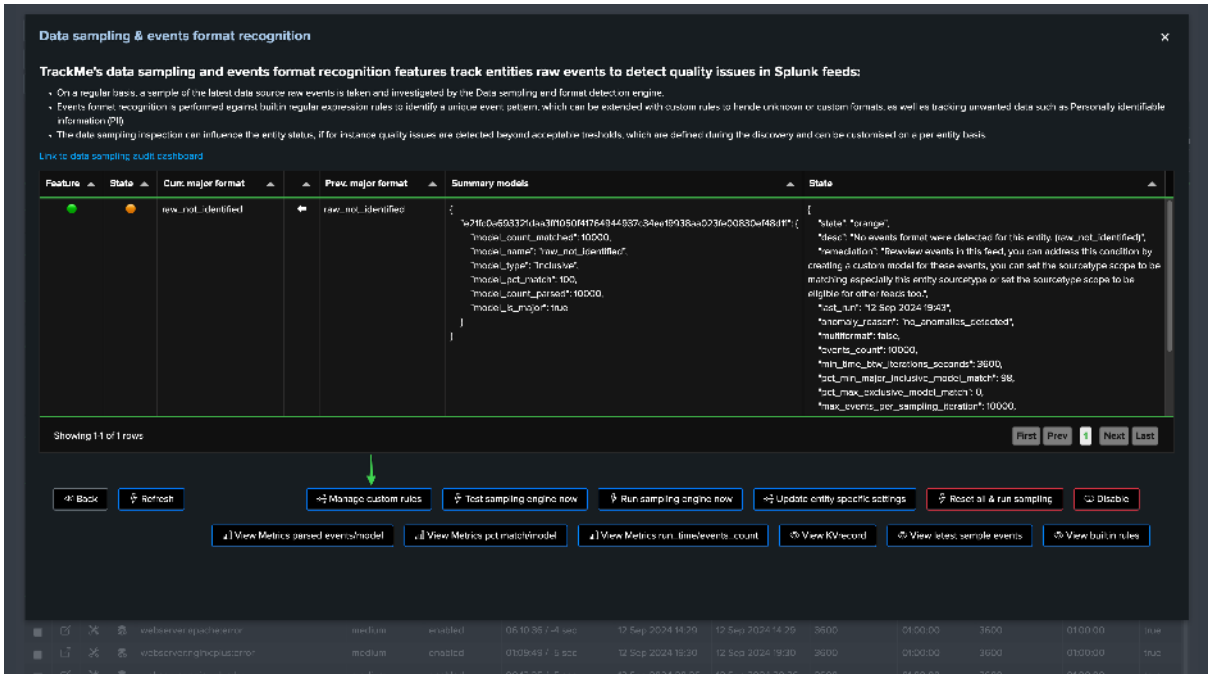
- Out-of-the-box rules are always inclusive rules, with sourcetype_scope=*.
- Out-of-the-box rules are always applied after custom rules, if any.
- For inclusive rules, the engine breaks for a given event at the first rule that matches the event, so an event can only match one rule. (with custom rules applied first)

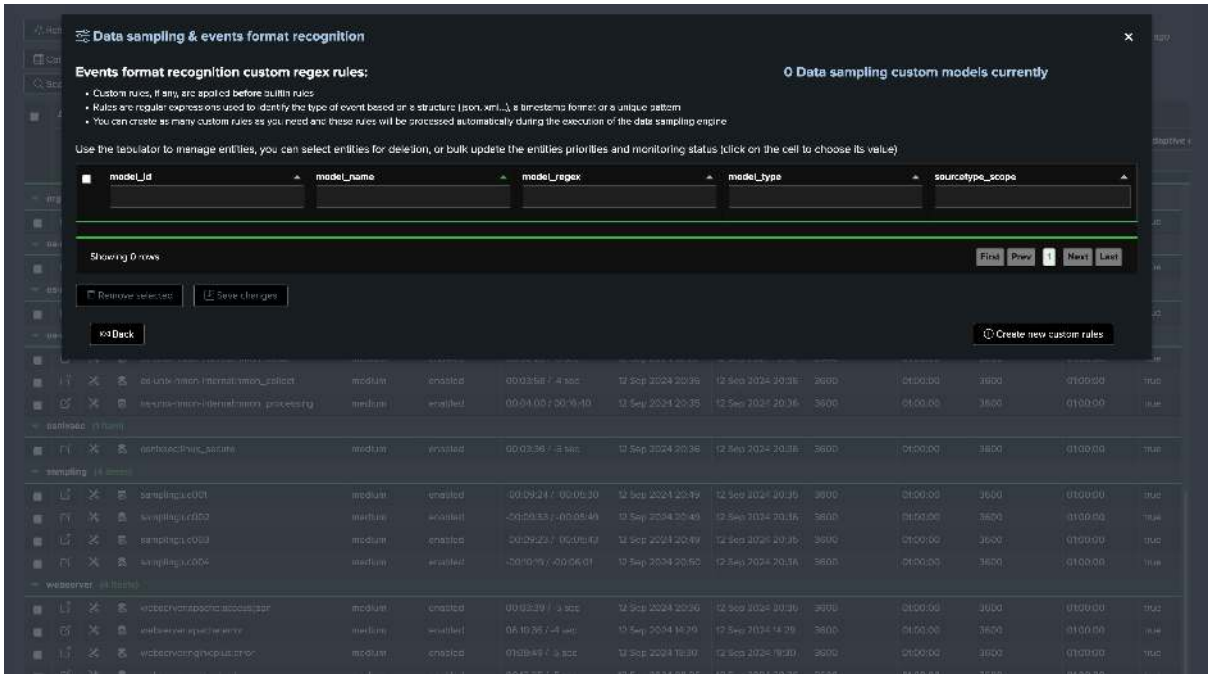




Custom Rules: Creating and Managing Custom Rules

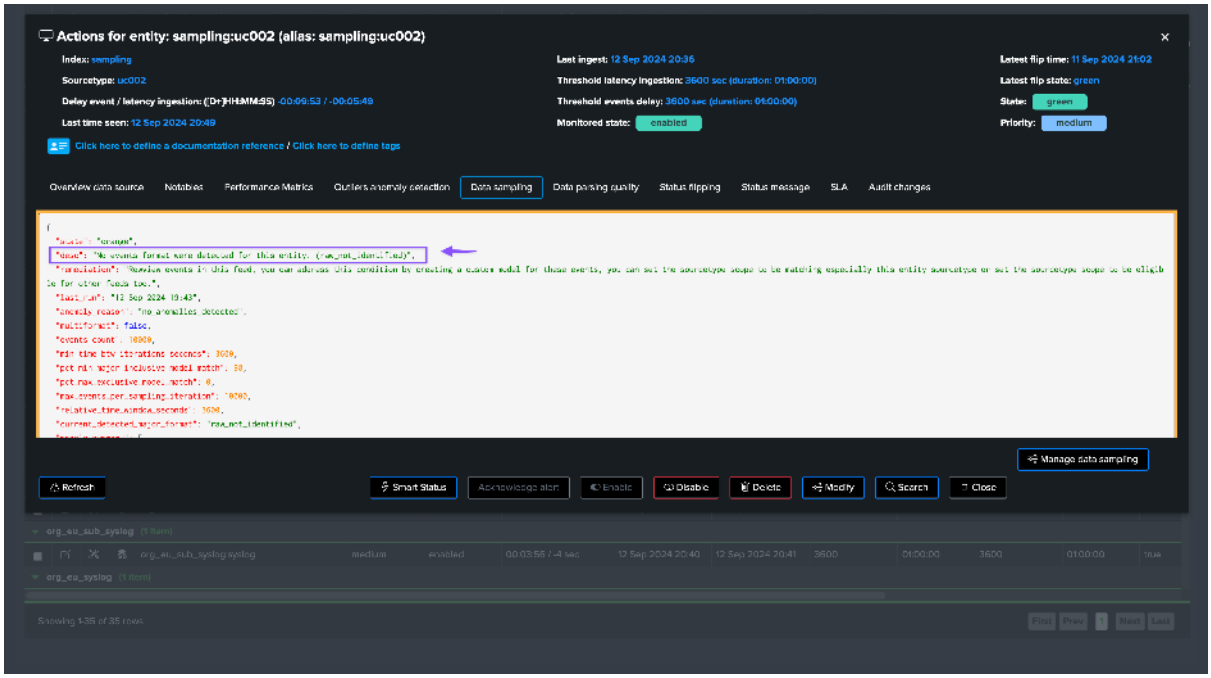
Custom rules can be created to address custom formats, or event formats which are not detected by TrackMe, you can access custom rules management from the entity screen:



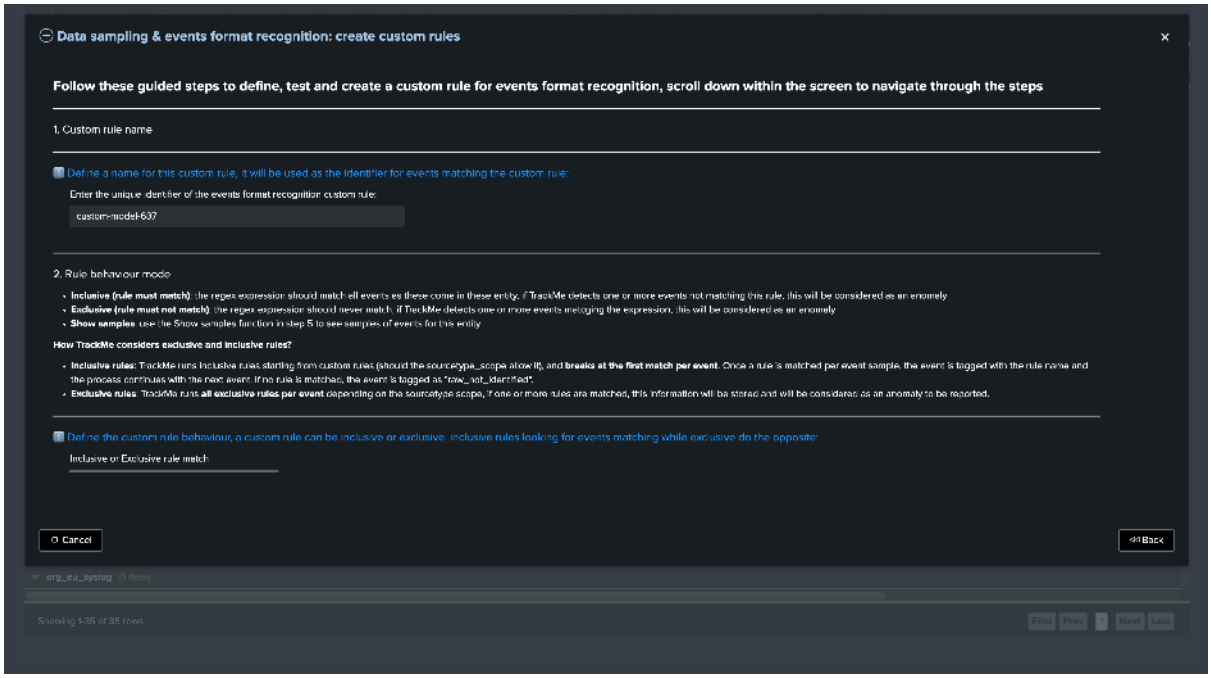


Example of Creating a Custom Rule

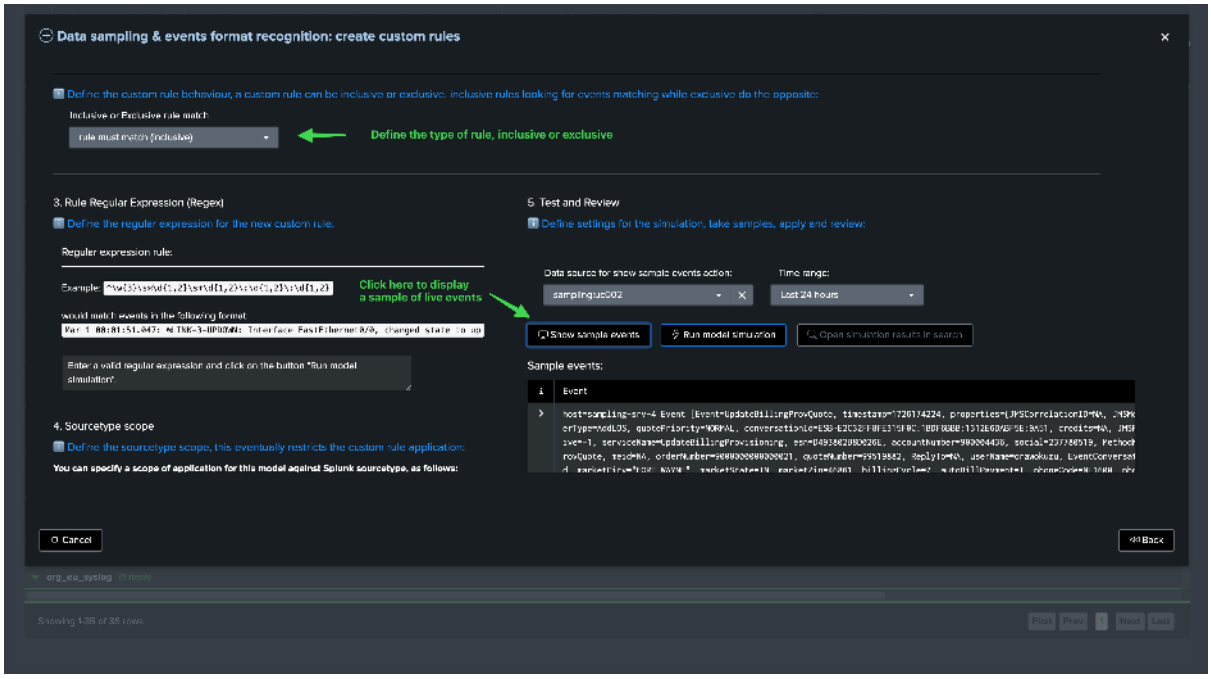
In the following example, we can see that the engine does not recognize the format of the events, and we can create a custom rule to address it, so that the engine can start tracking the format of the events:



The following screen shows the welcome start of the custom rule creation screen:



Scroll down, you can from the UI define the sourcetype scope of the custom rule, as well as displaying a sample of events, defining the regular expression of the rule and testing in real conditions the rule:

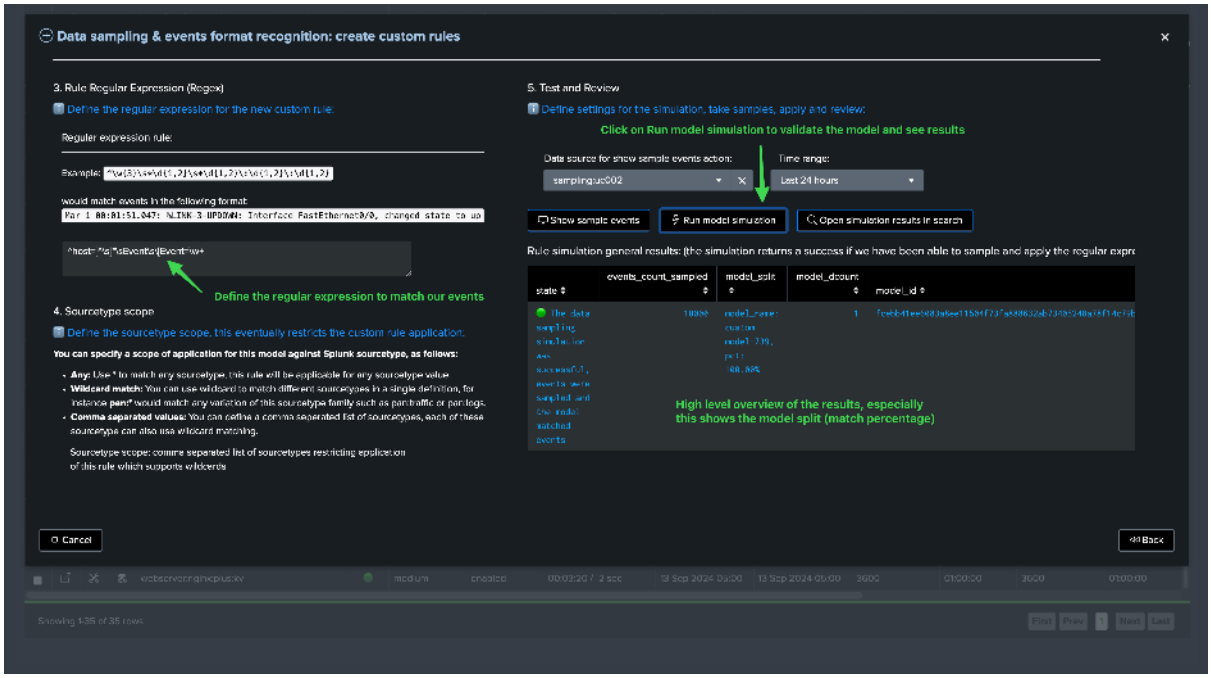


The “Show sample events” button executes underneath the following TrackMe command:

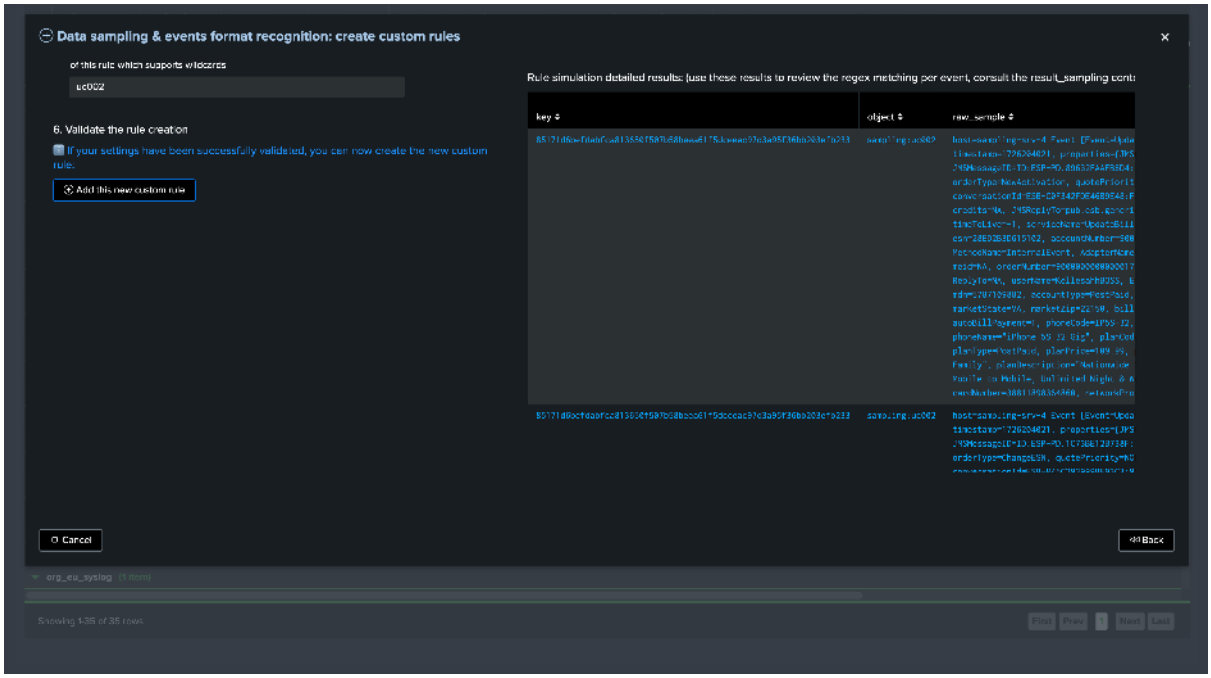
Notes: replace mytenant and myobject with the actual tenant and object values

```
| trackmesamplingexecutor tenant_id="mytenant" object="myobject" mode=
↪"get_live_samples" get_samples_max_count="10000" earliest="-24h" latest="now"
```

Define the regular expression for the model and click on “Run model simulation”, the top table shows the high-level overview of the model simulation and notably its matching percentage:



Scroll down to see the detailed result of the sampling exercise per event:



The overview table is powered by the following TrackMe command:

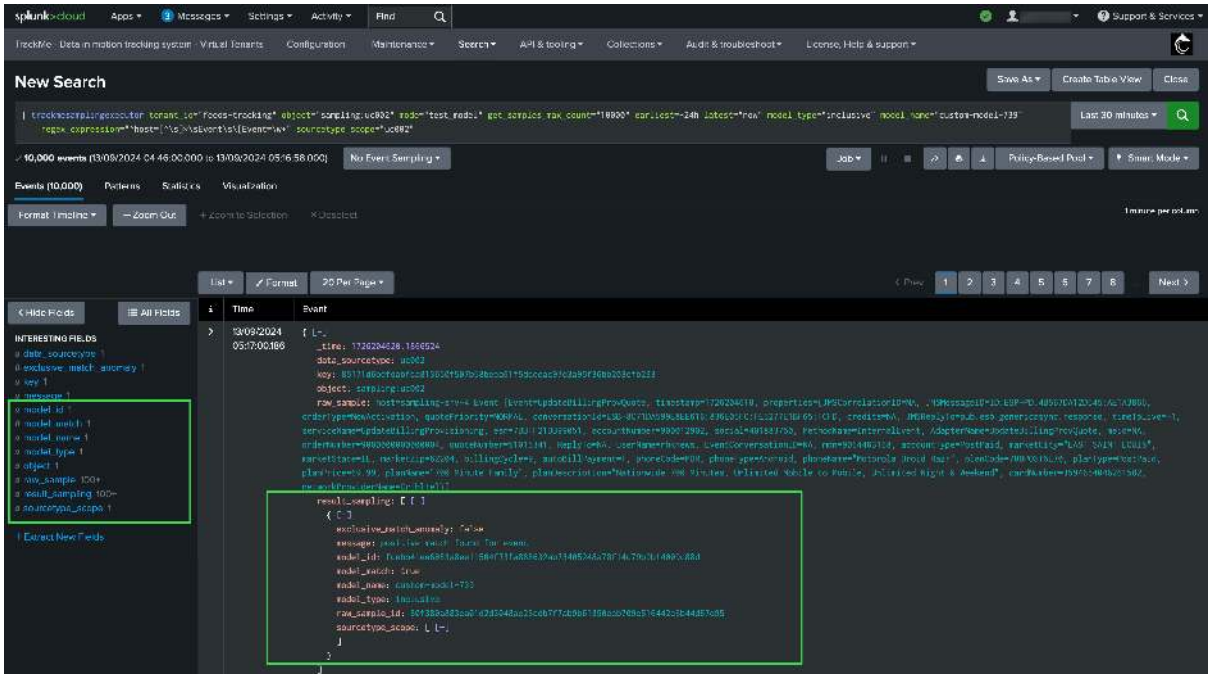
Notes: this command below shows the context for this simulation, tenant_id, object, sourcetype_scope, model_name and model_regex especially are dynamically defined based on your inputs

```
| trackmesamplingexecutor tenant_id="feeds-tracking" object="sampling:uc002"␣
↪mode="test_model" get_samples_max_count="10000" earliest="-24h" latest="now"␣
↪model_type="inclusive" model_name="custom-model-739"␣
↪regex_expression="^host=[^\s]*\sEvent\s\[Event=\w+" sourcetype_scope="uc002" |␣
↪`trackme_data_sampling_format_simulation_results`
```

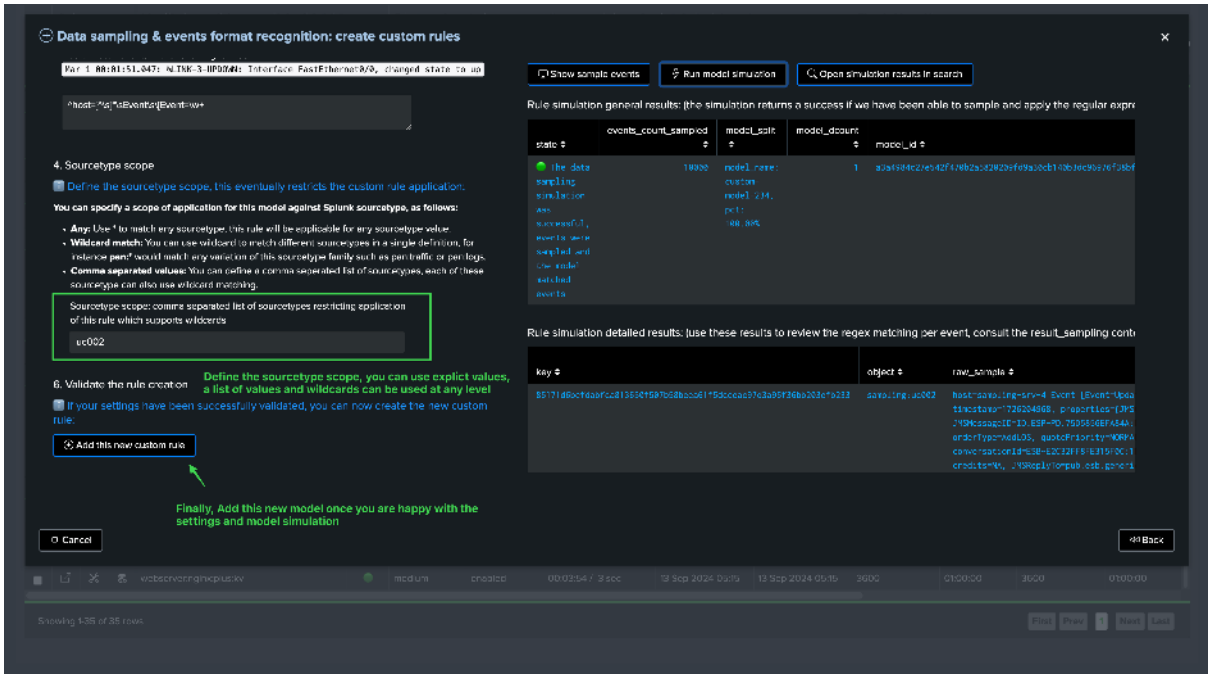
The detailed table is powered by the following TrackMe command:

```
| trackmesamplingexecutor tenant_id="feeds-tracking" object="sampling:uc002" mode=
↪"get_live_samples" get_samples_max_count="10000" earliest="-24h" latest="now"
```

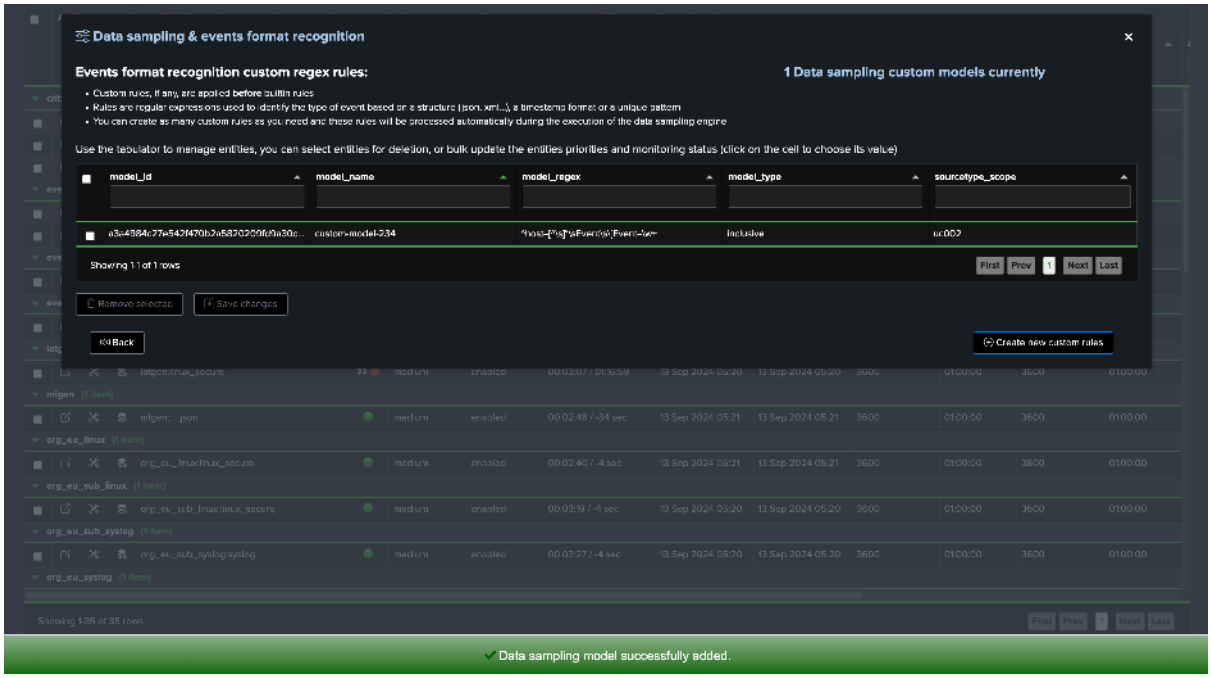
You can click on “Open simulation results in search” to review results in Splunk search UI:

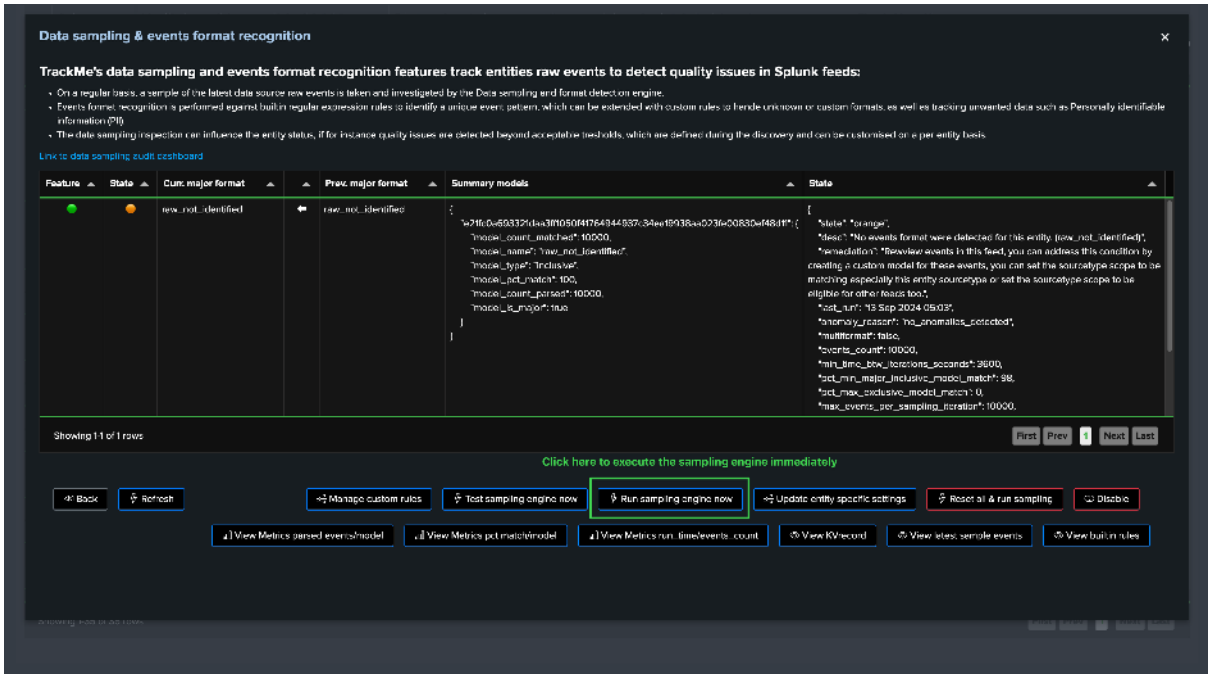


Finally, define the sourcetype scope and add the model:

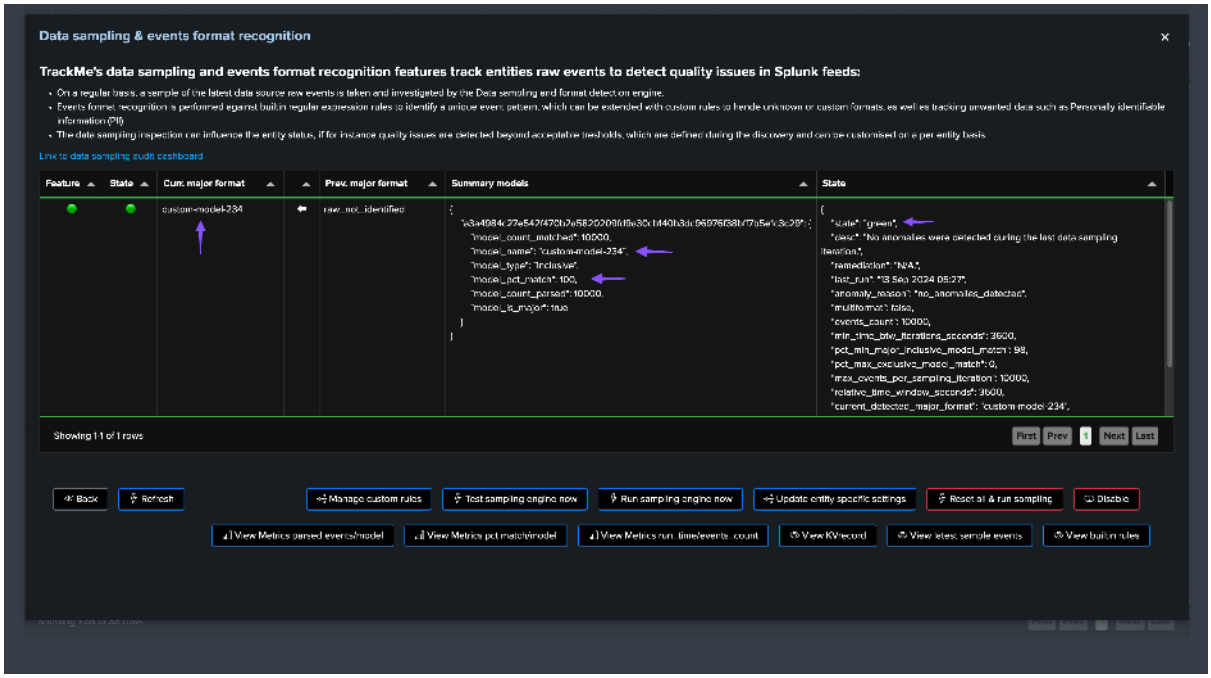


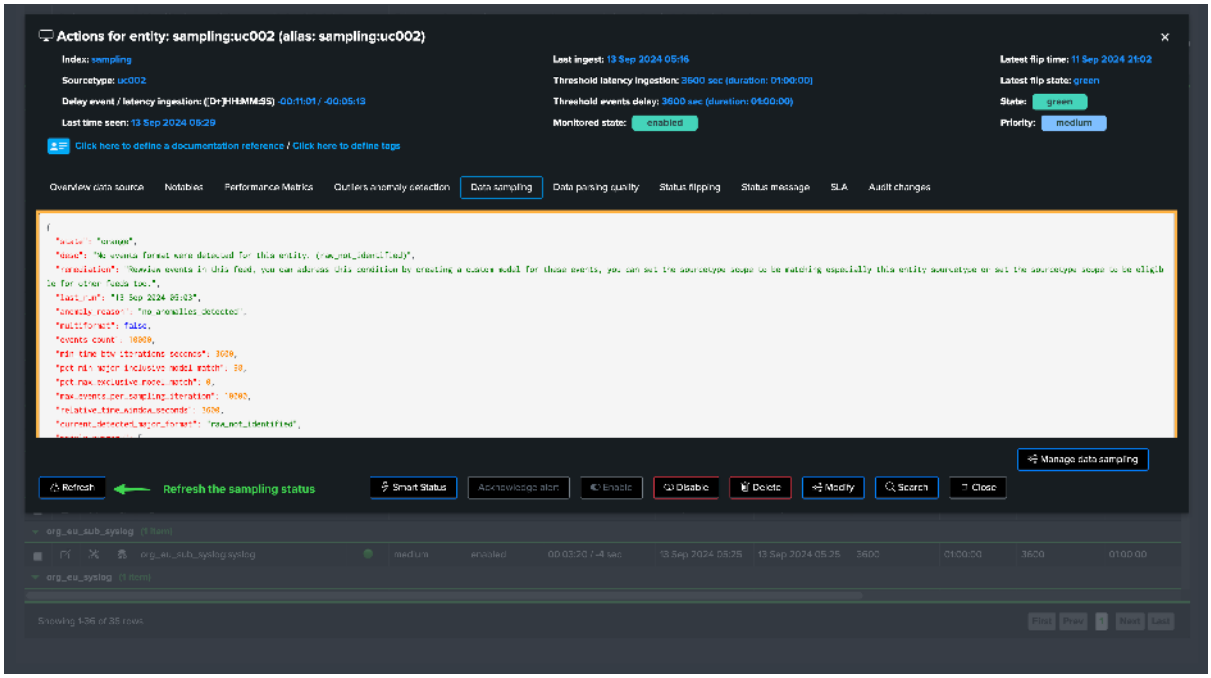
The model has been added, we can request a manual execution so the model is applied immediately:

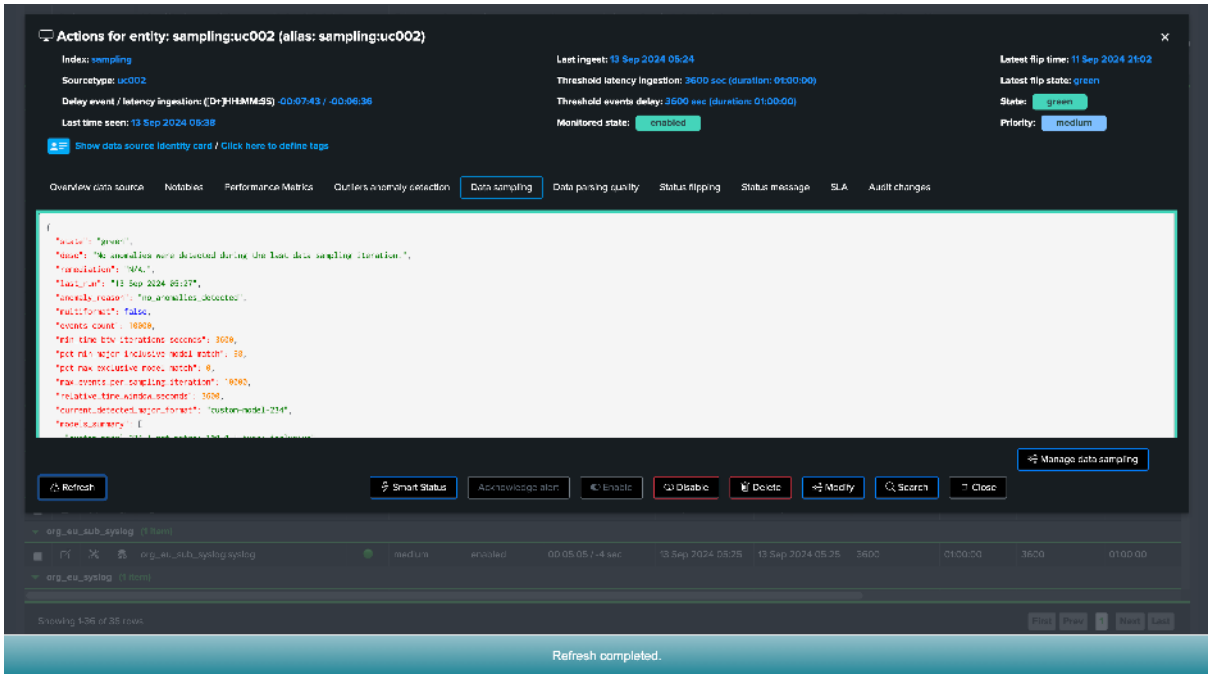




Once executed, the engine took our new model into account, the sampling results updates the status and from this stage, sampling is operational and tracking events for this entity:

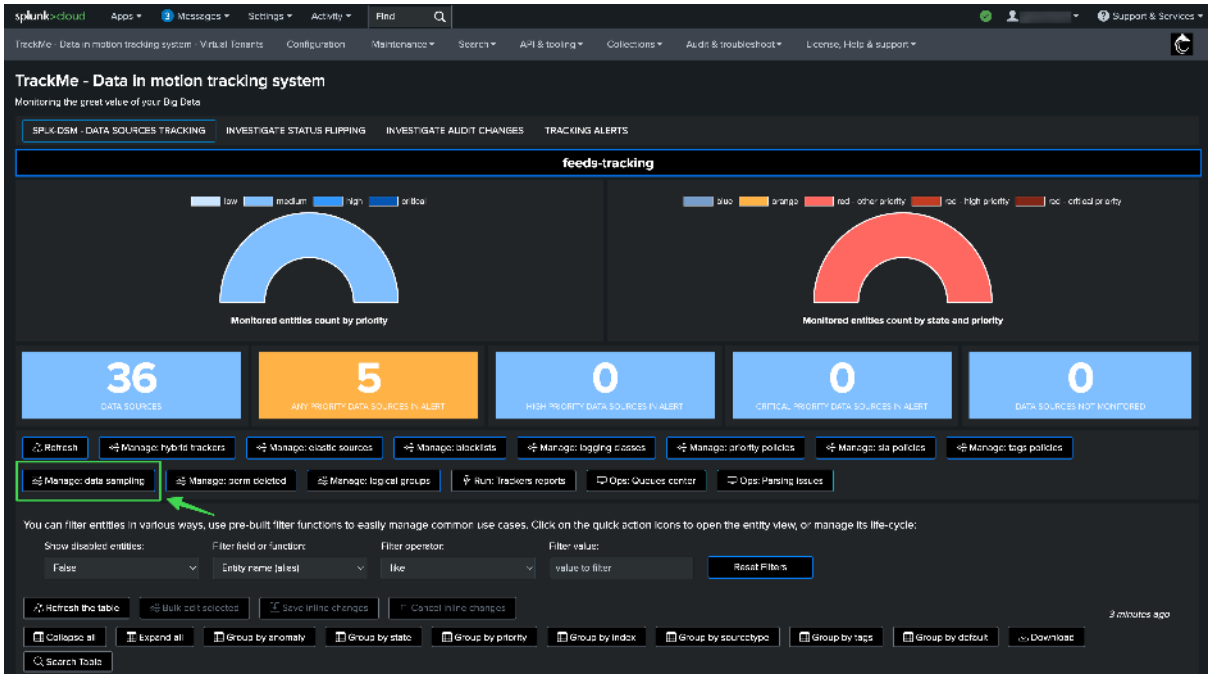

Data sampling & events format recognition
TrackMe's data sampling and events format recognition features track entities raw events to detect quality issues in Splunk feeds:
Feature
State
Curr. major format
Prev. major format
Summary models
State
custom-model-234
raw_not_identified
Showing 1 to 1 of 1 rows
First
Prev
Next
Last
Back
Refresh
Manage custom rules
Test sampling engine now
Run sampling engine now
Update entity specific settings
Reset all & run sampling
Disable
View Metrics parsed events/model
View Metrics pct match/model
View Metrics run_time/events_count
View KV record
View latest sample events
View built-in rules

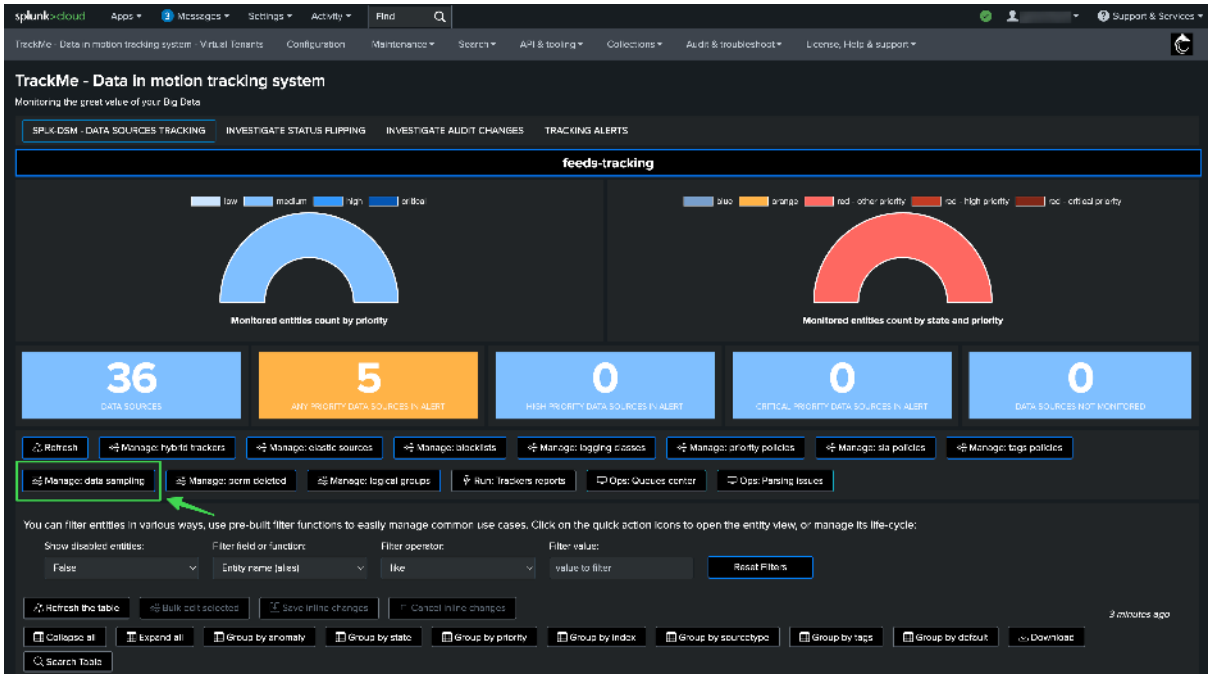

Actions for entity: sampling:uc002 (alias: sampling:uc002)
Index: sampling
Sourcetype: uc002
Delay event / latency ingestion: (D+)HH:MM:SS) -00:11:01 / -00:05:13
Last time seen: 13 Sep 2024 05:29
Last ingest: 13 Sep 2024 05:16
Threshold latency ingestion: 3600 sec (duration: 01:00:00)
Threshold events delay: 3600 sec (duration: 01:00:00)
Monitored state: enabled
Latest flip time: 11 Sep 2024 21:02
Latest flip state: green
State: green
Priority: medium
Click here to define a documentation reference / Click here to define tags
Overview data source
Notables
Performance Metrics
Outliers anomaly detection
Data sampling
Data parsing quality
Status flipping
Status message
SLA
Audit changes
Manage data sampling
Refresh
Refresh the sampling status
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search
Close



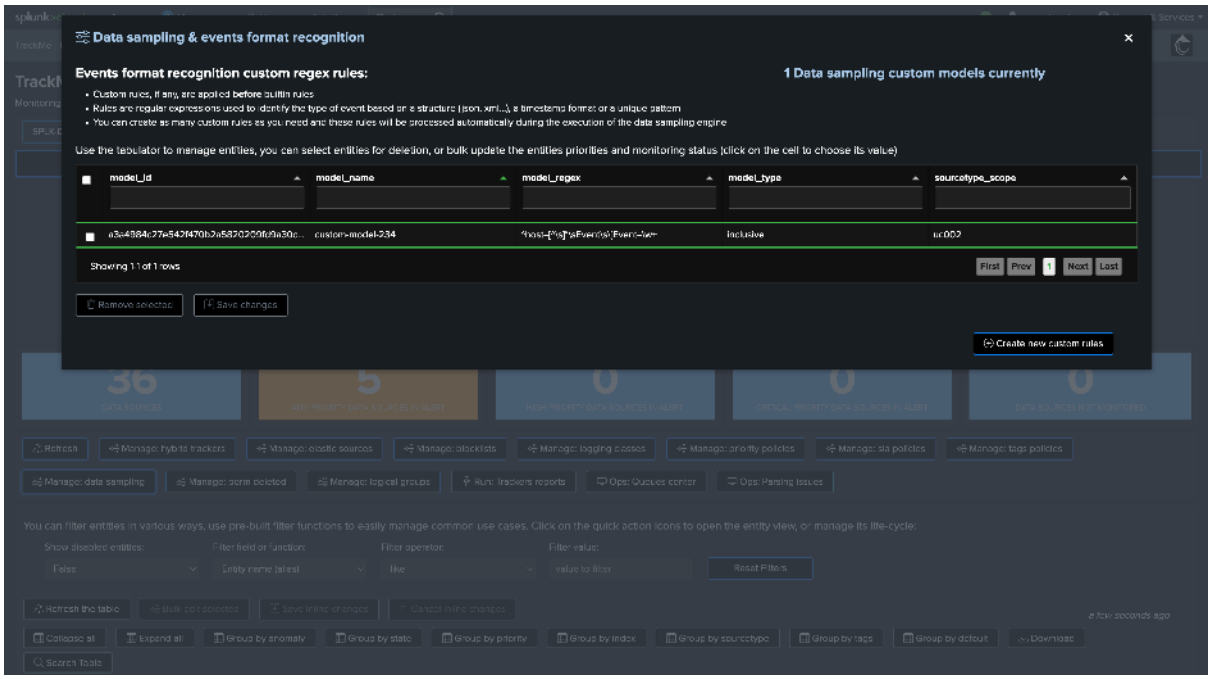
Managing Models

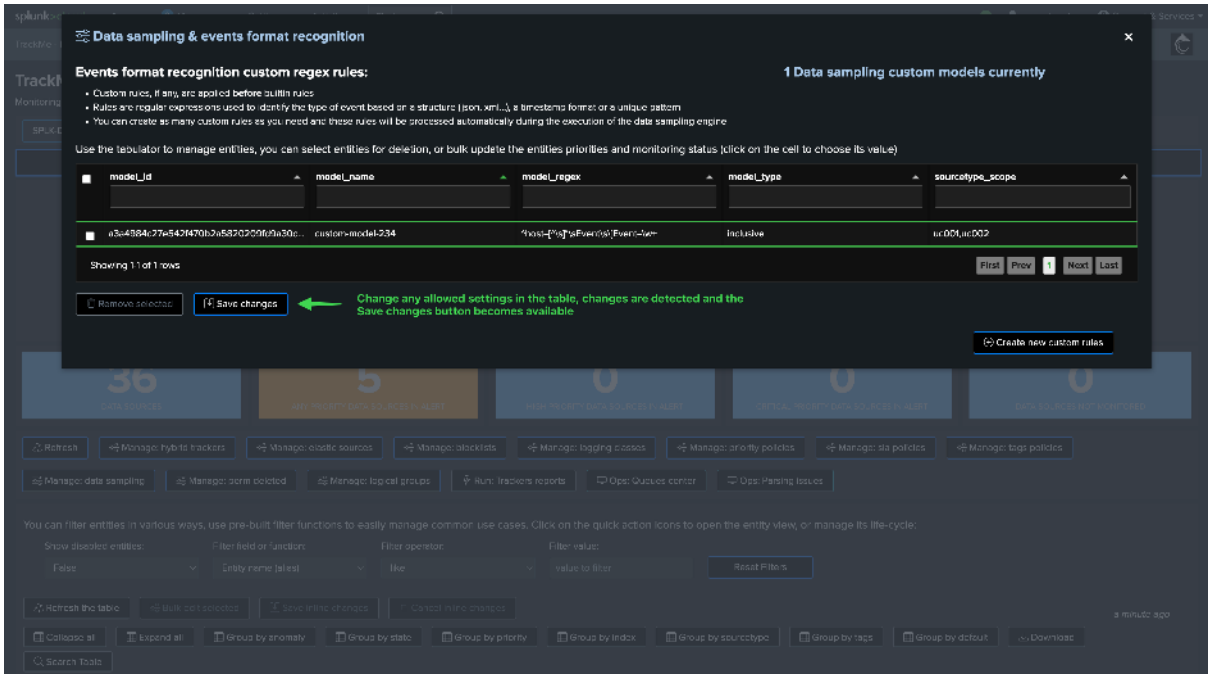
You can manage models at any time, either from the entity screen, or from the data sampling management screen:





From this screen, you can modify the model settings (sourcetype scope, regular expression), or you can delete the model:

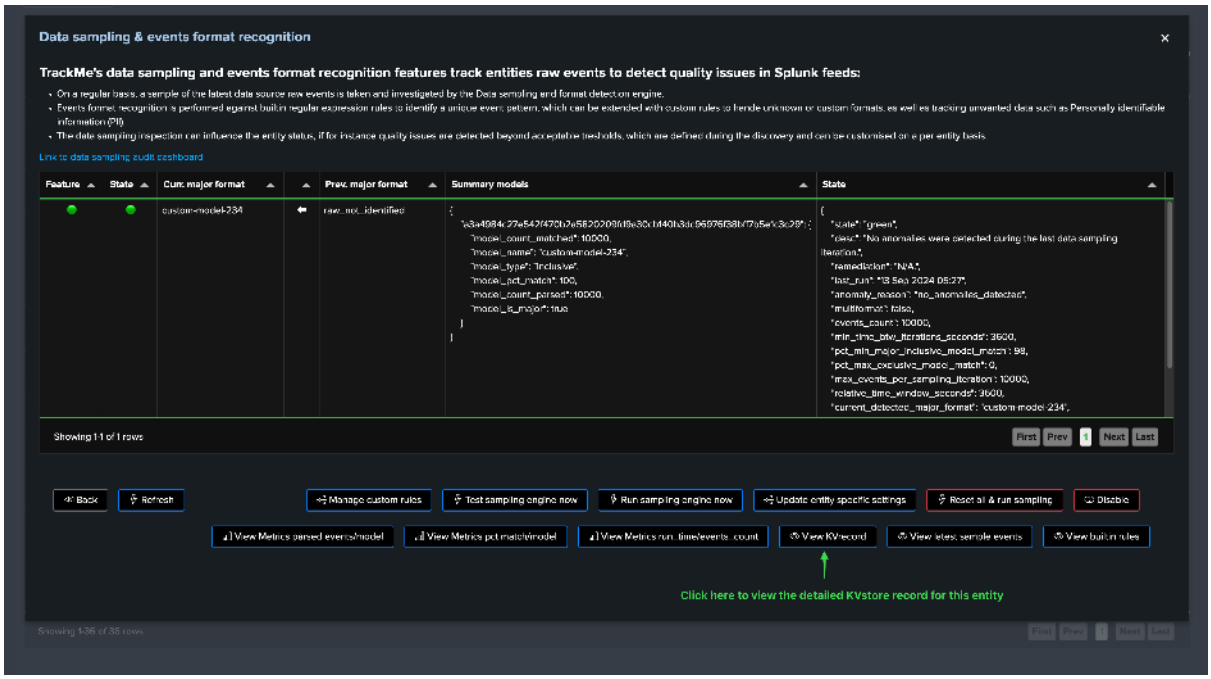


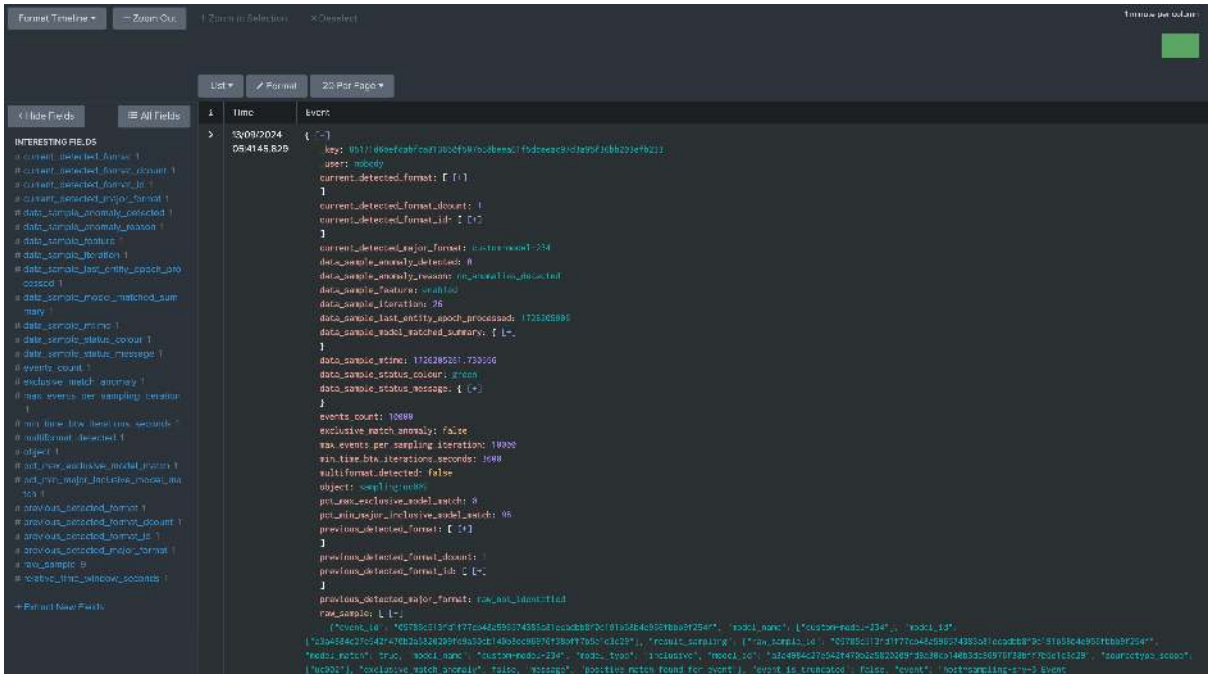


Entity KVstore Record and Data Sampling KVstore Collection

Entity KVrecord and Metadata Fast Access

The sampling engine stores and maintains entity-level Metadata as well as a subset of the sampled events for investigation purposes, you can easily access the KVrecord via the UI:



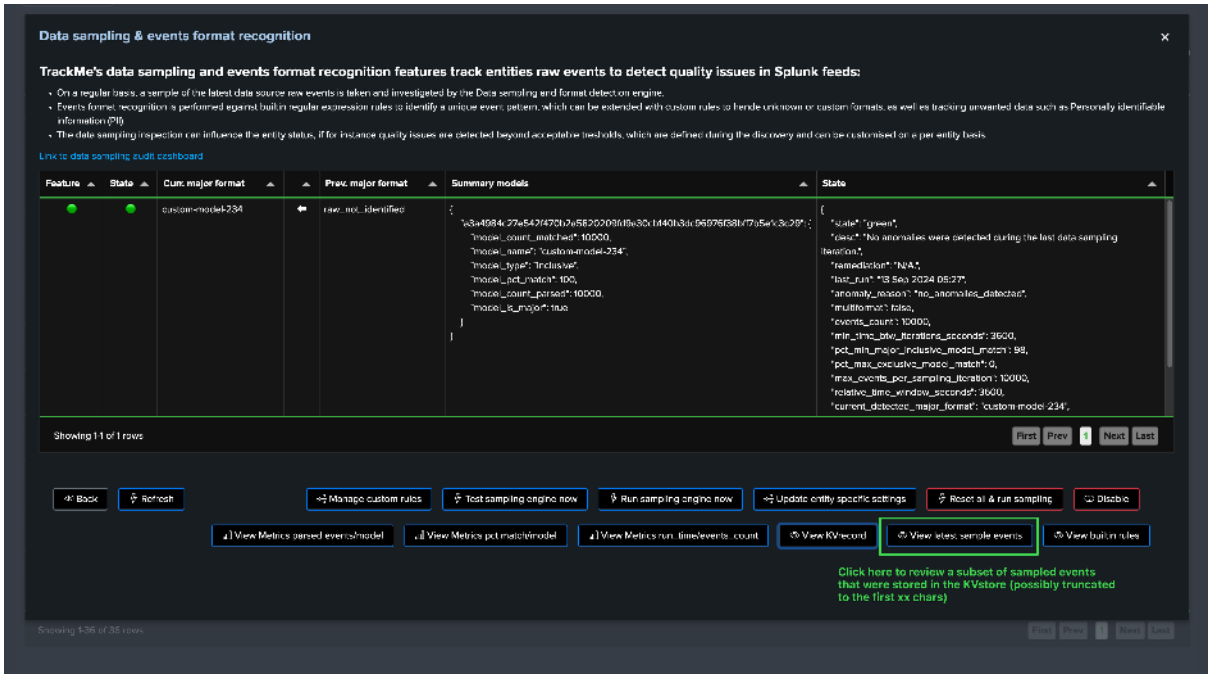


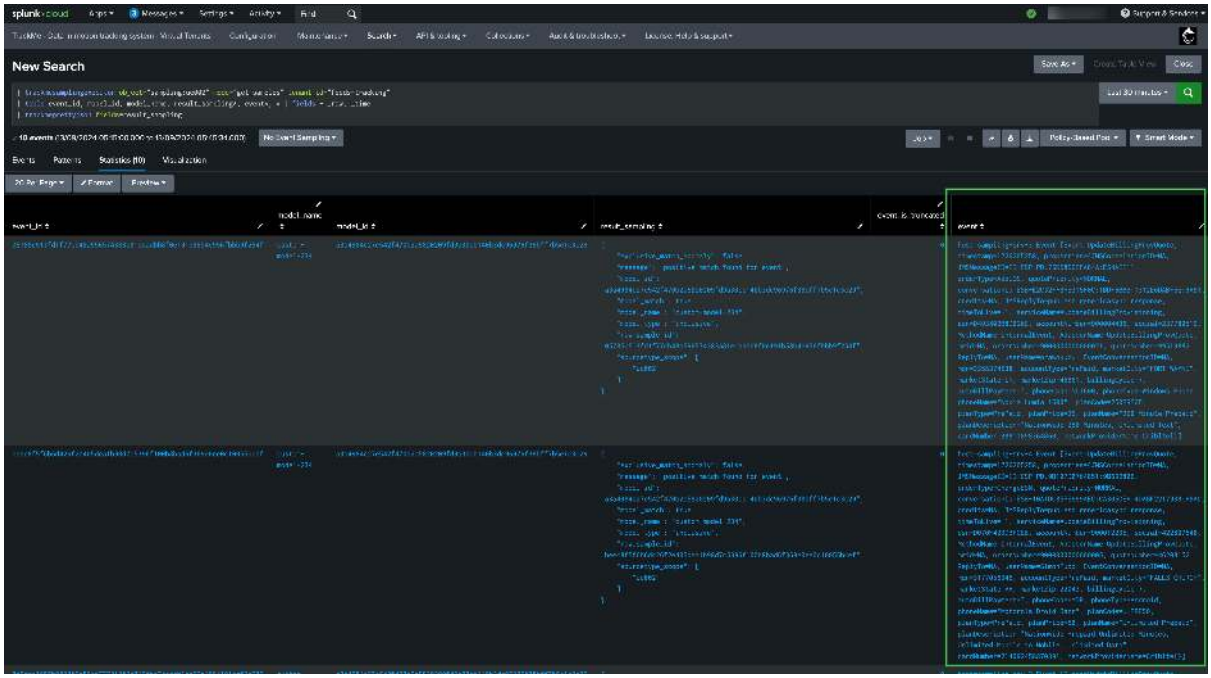
We can see that the engine stores and maintains a subset of sampled events, in fact by default it will take a subset of 10 events per model matched, and it will truncate these events to the first 40k characters, you can also access the stored samples via the button “Show samples”

Hint

About number of events stored in the KVstore collection and truncation:

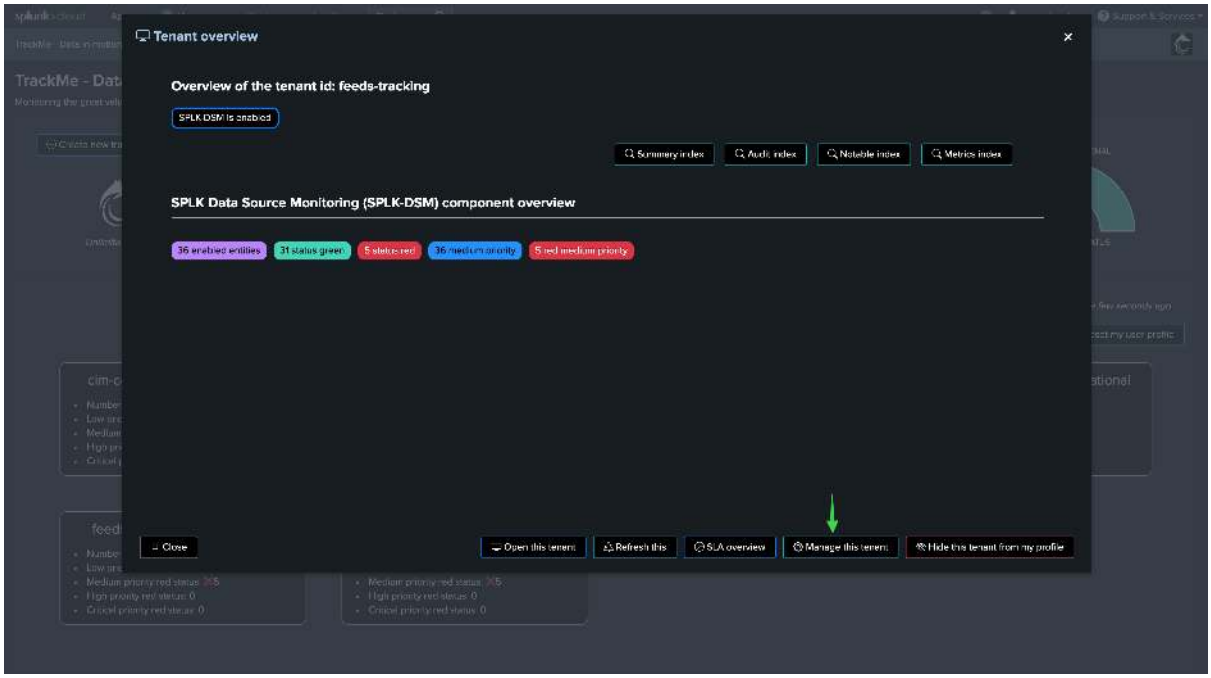
- The number of events stored in the KVstore collection per entity and per model matched is defined by the system-wide setting “splk_data_sampling_no_records_saved_kvrecord”.
- The truncation is only applied when the engine stores events in the KVstore for investigation purposes, not when the regular expression is applied to the event.
- The truncation limit is defined by the system-wide setting “splk_data_sampling_records_kvrecord_truncate_size”.

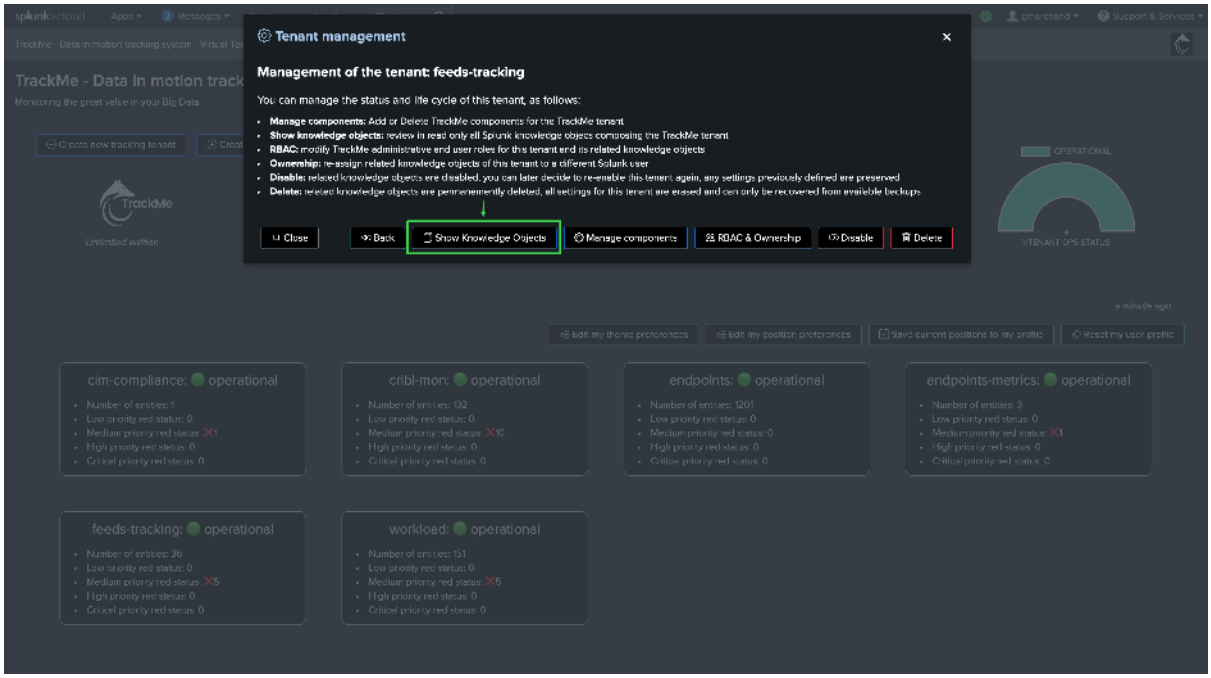


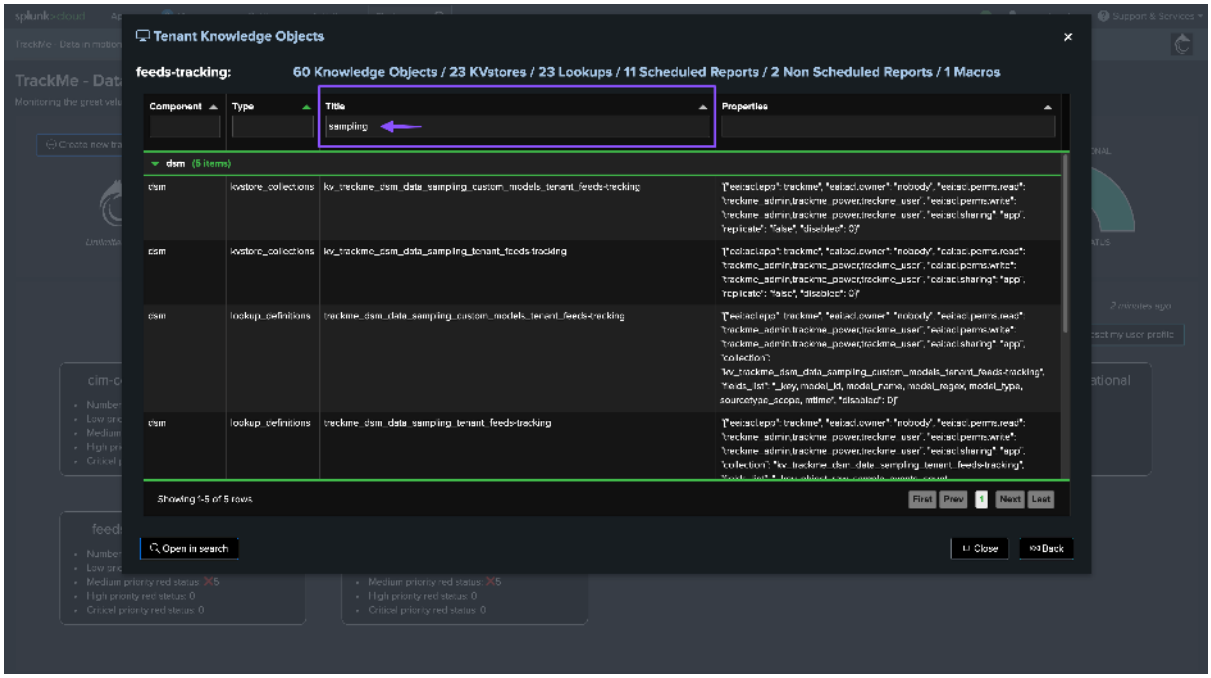


Data Sampling KVstore Collection

The data sampling KVstore collection is created when the Virtual Tenant is created, you can find out the KVstore collection and the transforms name easily from the Virtual Tenant UI:







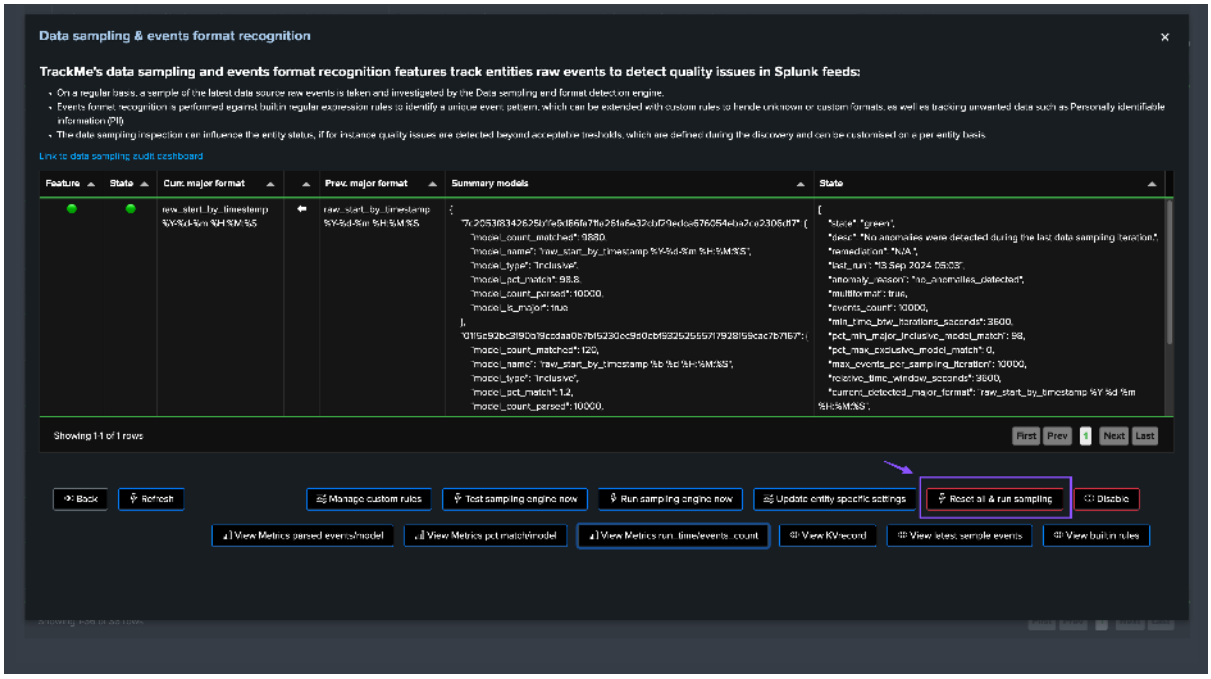
The KVstore collection transforms definition is honoring the following convention:

```
| inputlookup trackme_dsm_data_sampling_tenant_<tenant_id> | eval key=_key
```

Resetting Data Sampling Status and Settings for a Given Entity

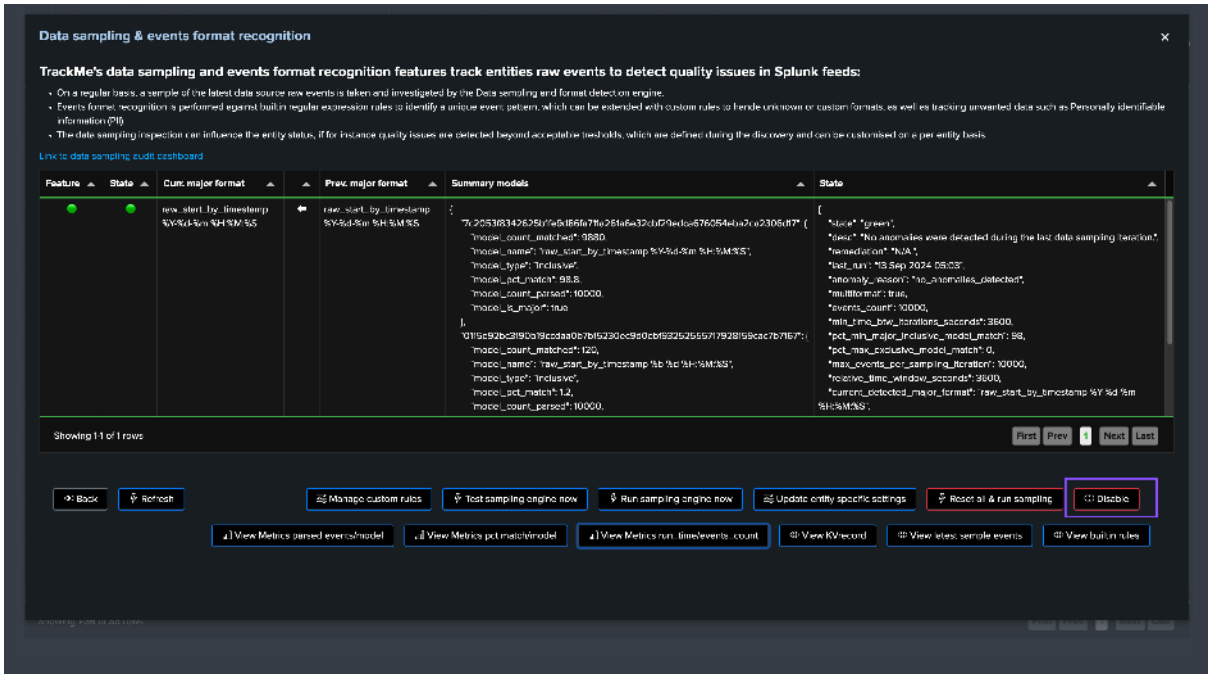
Reset Status & Settings for a Given Entity

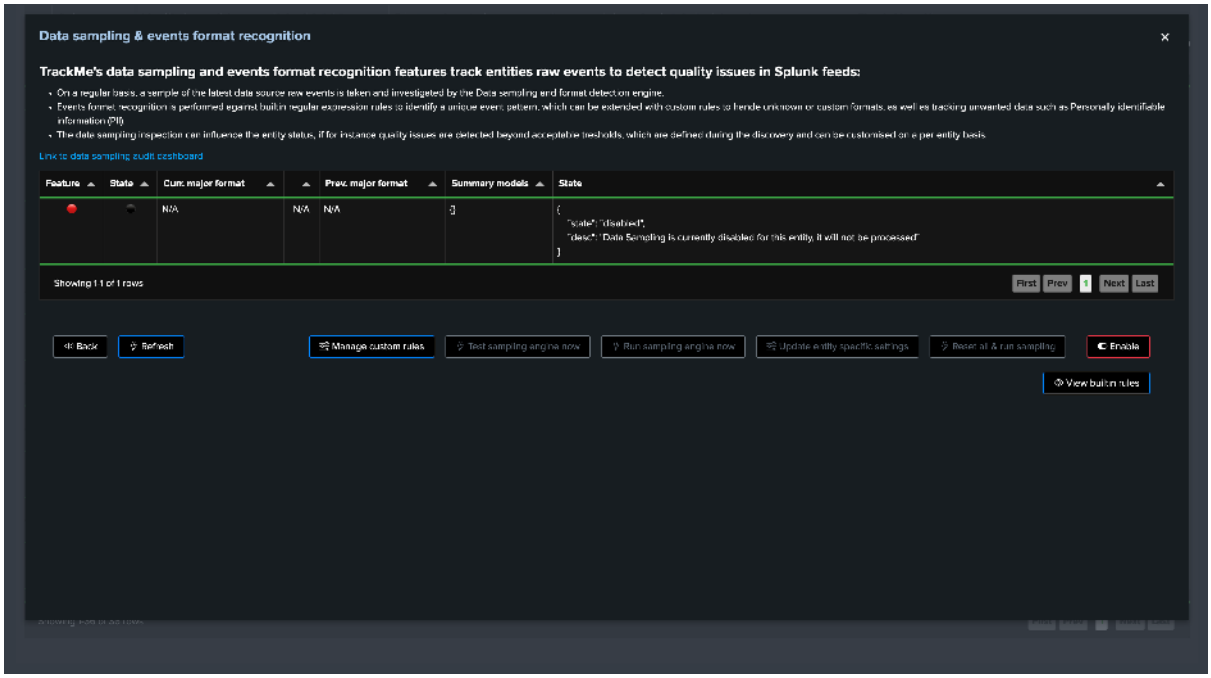
You can reset the data sampling status and settings for a given entity, this will reset the data sampling status, the models, and the settings to the default values, you can access this feature from the entity screen:

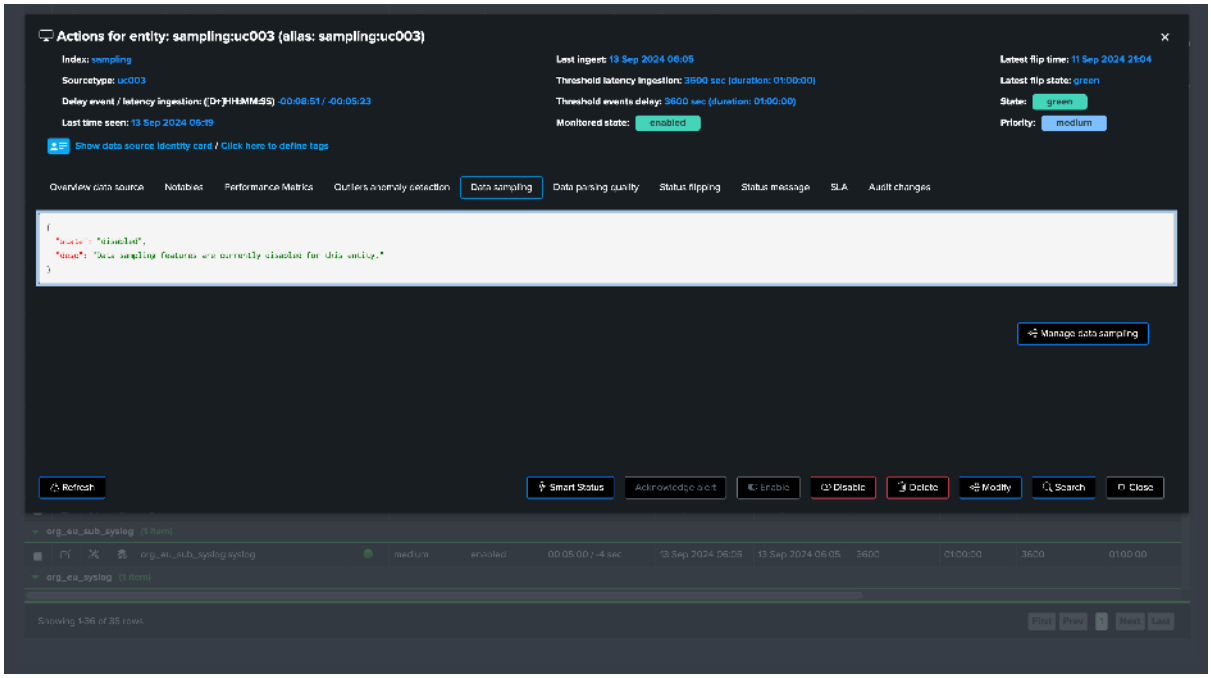


Disable Sampling for a Given Entity

You can disable the data sampling feature for a given entity, the sampling feature will be entirely disabled and the engine will never attempt to process sampling for it:

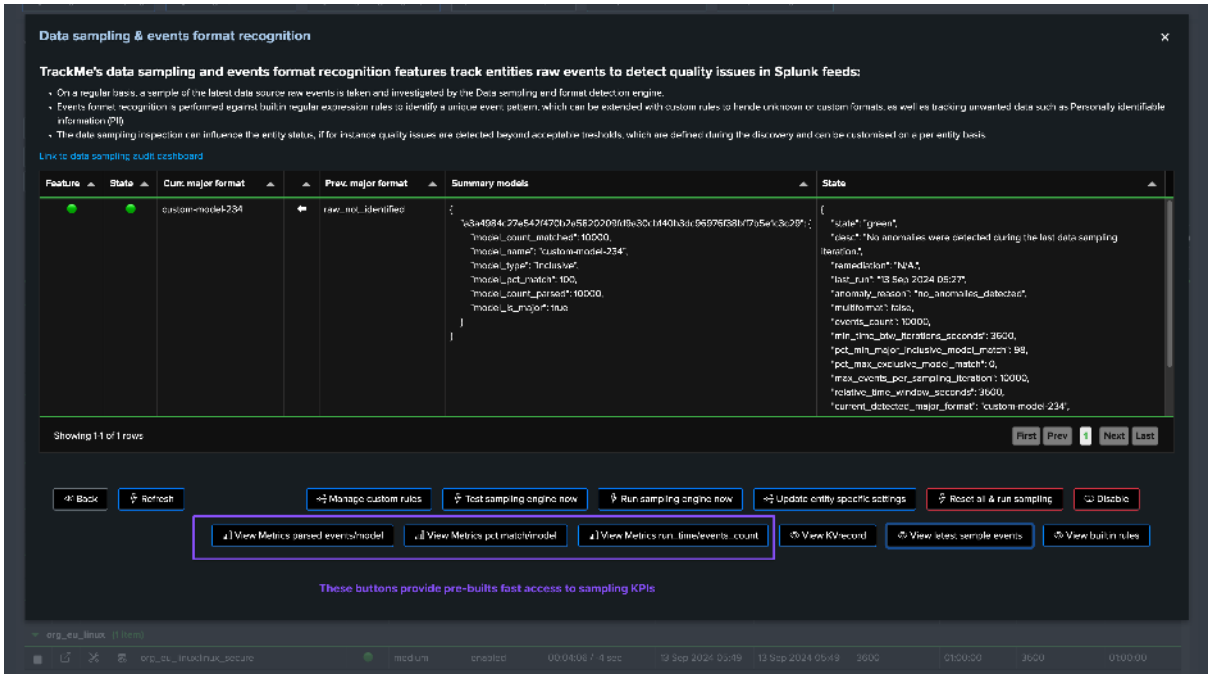




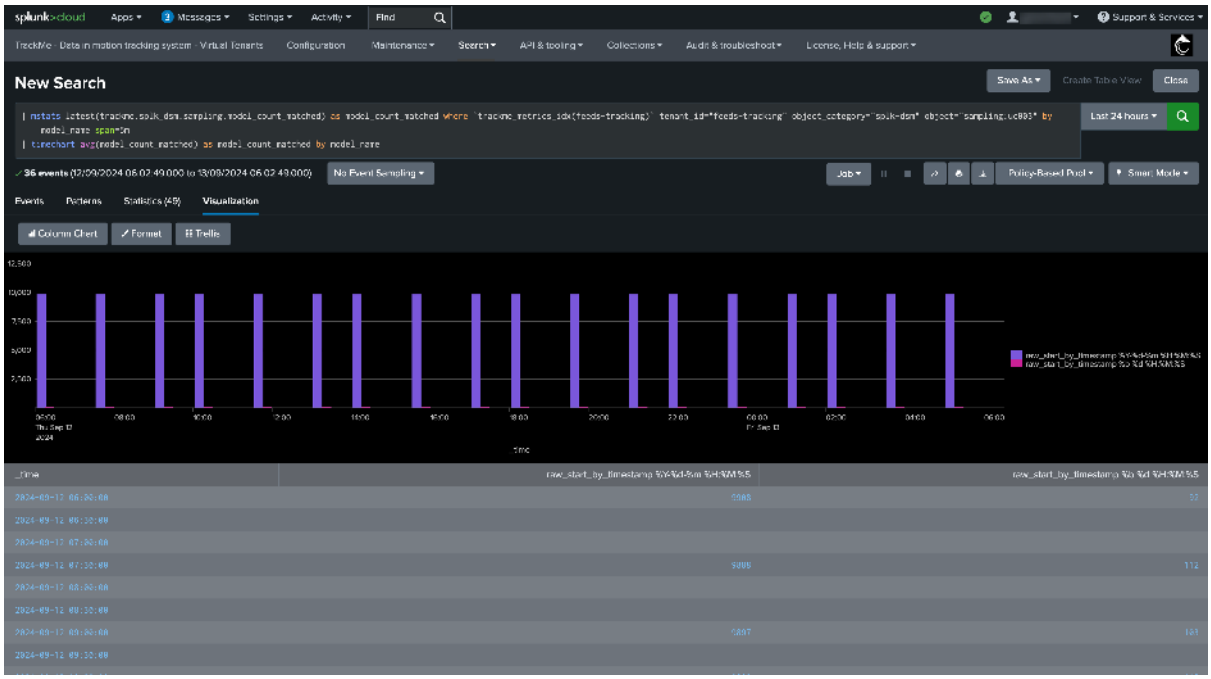


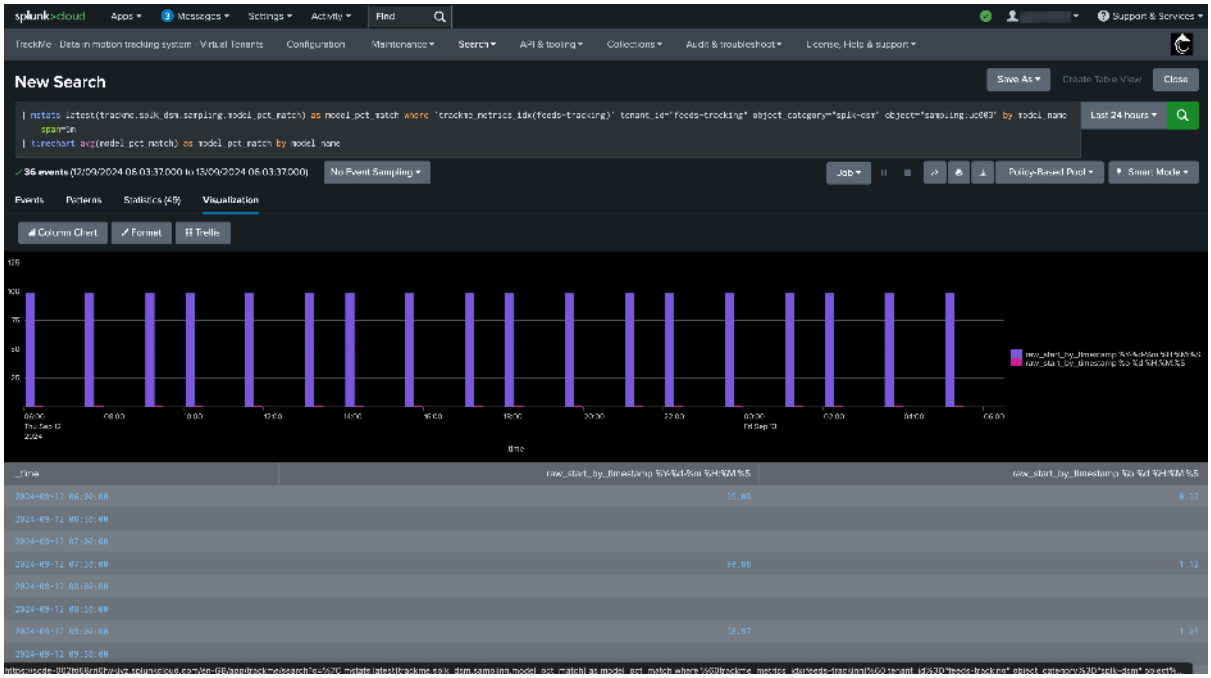
KPIs and Metrics

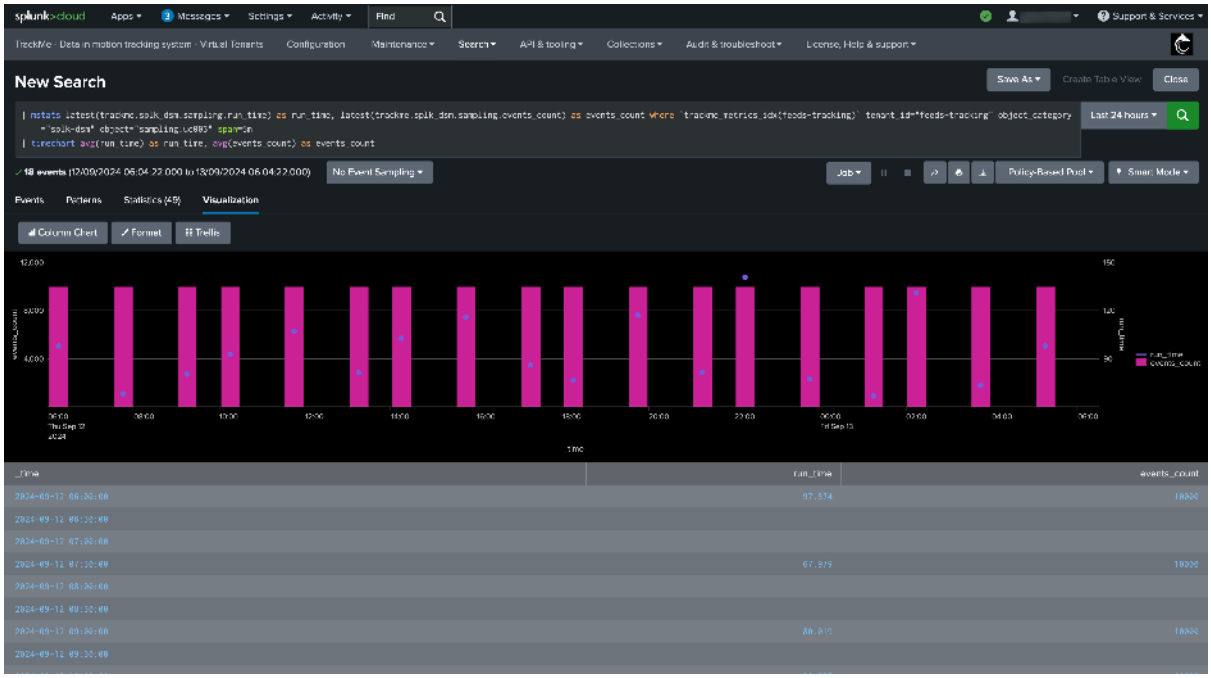
The data sampling engine generates various KPIs and metrics, which are stored in the tenant metrics index, you can access these metrics via the UI:


Data sampling & events format recognition
TrackMe's data sampling and events format recognition features track entities raw events to detect quality issues in Splunk feeds:
Feature
State
Curr. major format
Prev. major format
Summary models
State
Showing 1 to 1 of 1 rows
First
Prev
Next
Last
Back
Refresh
Manage custom rules
Test sampling engine now
Run sampling engine now
Update entity specific settings
Reset all & run sampling
Disable
View Metrics parsed events/model
View Metrics pct match/model
View Metrics run_time/events_count
View KV record
View latest sample events
View built-in rules
These buttons provide pre-builts fast access to sampling KPIs

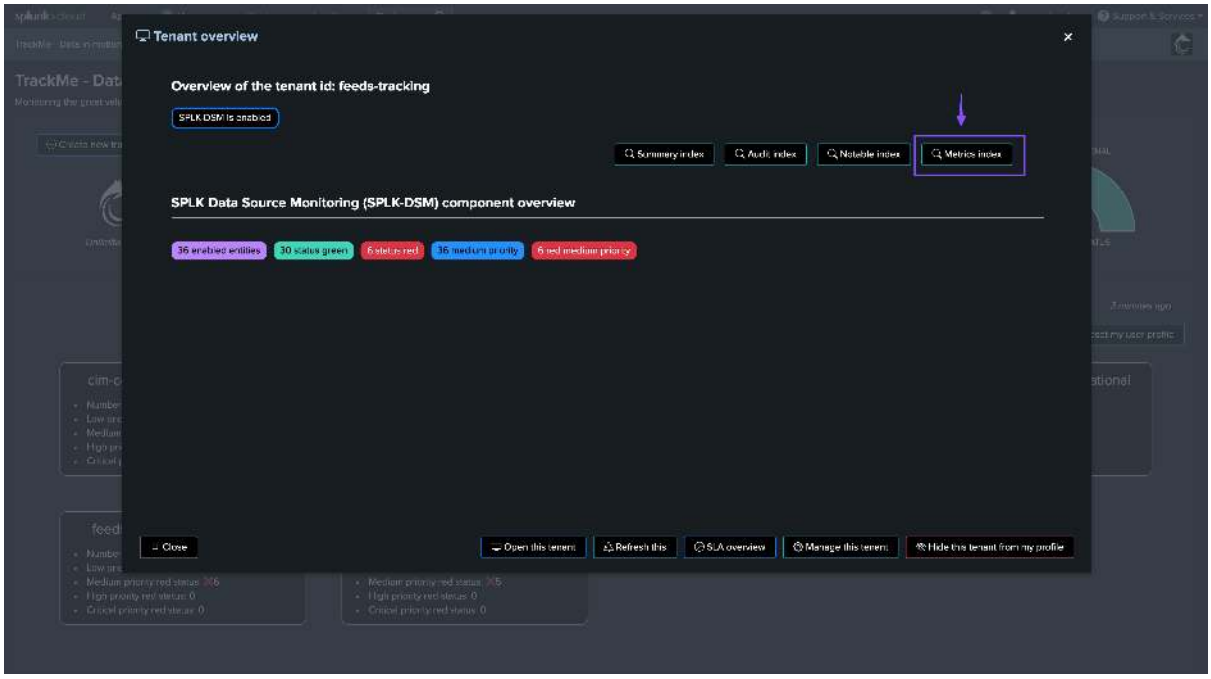
Examples:


splunk>cloud
New Search
Column Chart
Format
Trellis



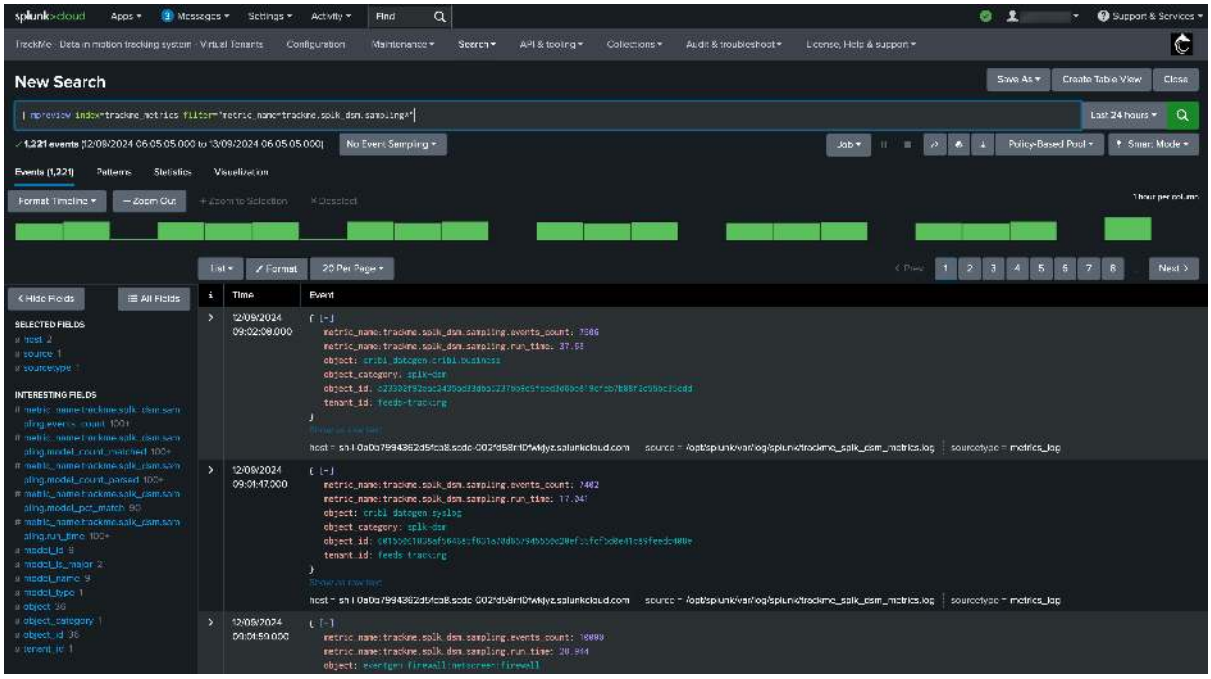


The Virtual Tenants UI provides a quick mpreview access to the tenant-related metrics:



You can also use the following generic mpreview command to access the metrics: (adapt the index name if needed)

```
| mpreview index=trackme_metrics filter="metric_name=trackme.splk_dsm.sampling*"
```



Troubleshooting

Sampling Logging

Data Sampling tracker logs can be inspected through the following search:

```
index=_internal sourcetype=trackme:custom_commands:trackmesamplingexecutor
```

Any errors, unexpected exceptions can be found easily using the log_level:

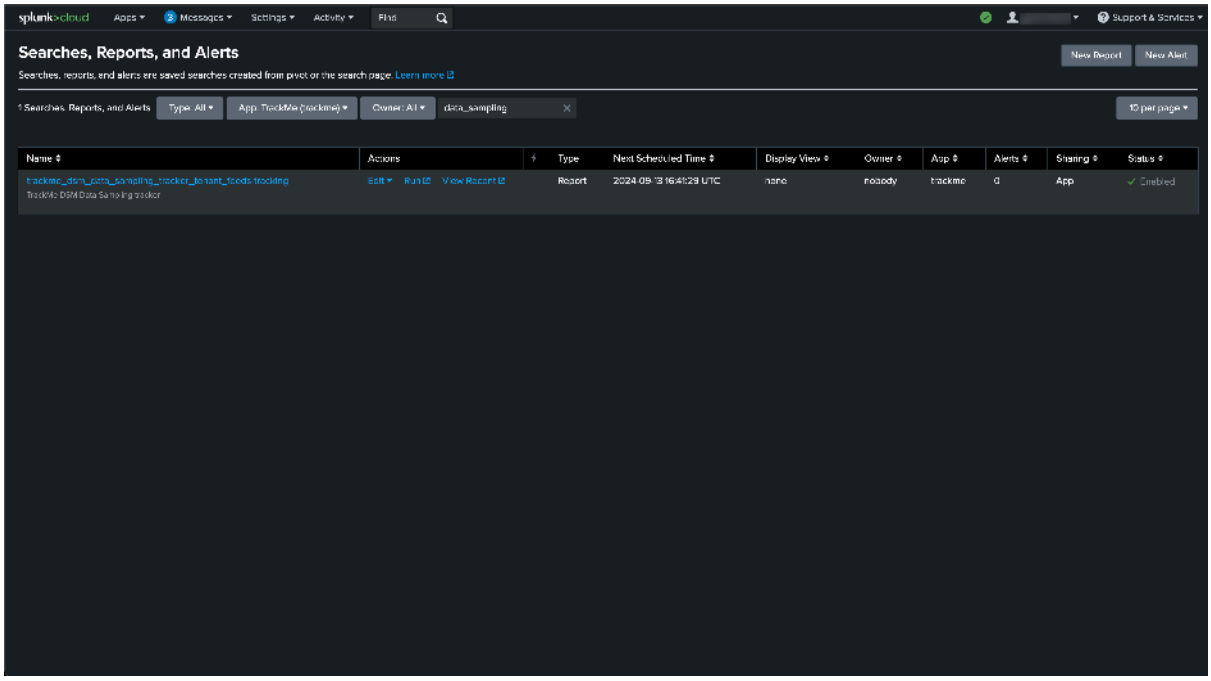
```
index=_internal sourcetype=trackme:custom_commands:trackmesamplingexecutor␣
↪log_level=error
```

Activity related to a specific Splunk feed, a TrackMe entity, can easily be inspected by filtering on the object value:

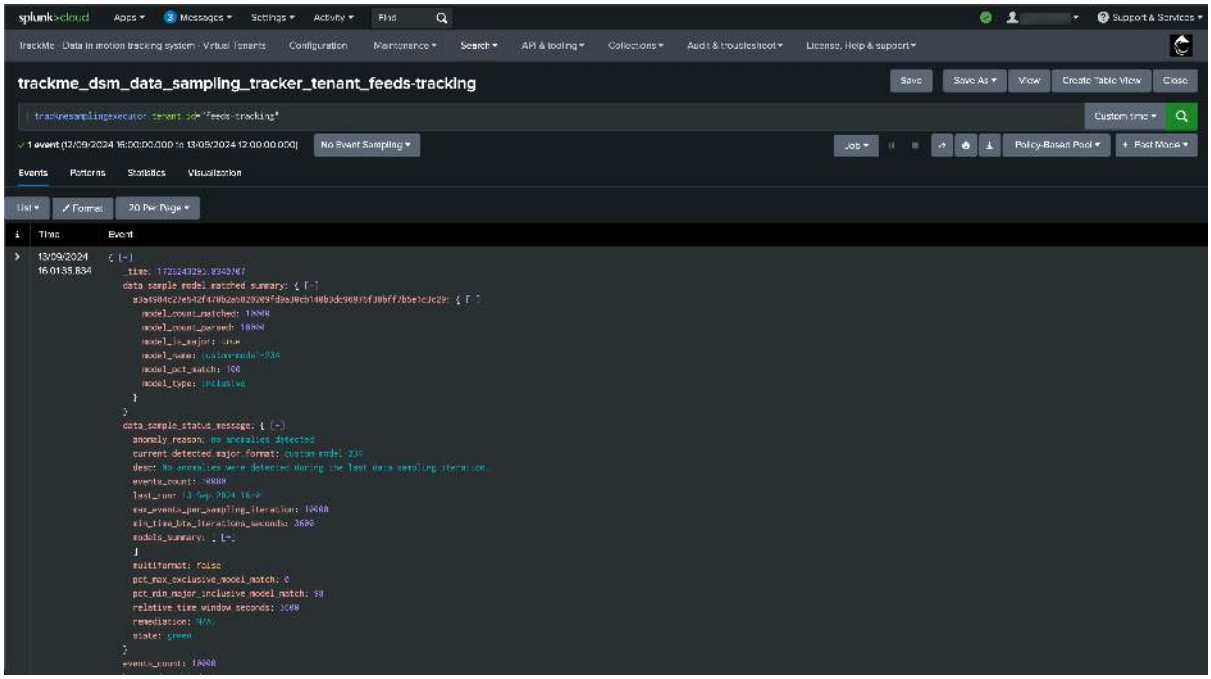
```
index=_internal sourcetype=trackme:custom_commands:trackmesamplingexecutor object="
↪<object value>"
```

Inspecting the Data Sampling Tracker Execution

In Splunk, you can easily review the data sampling scheduled tracker, you can access the previous artifacts, or manually execute the job in case of issues:



Each entity processed results in one JSON event with all the details of the data sampling status:



Annexes

Annexe 1: System-wide Settings

Value	Description	Default Value
splk_data_sampling	This defines the minimal time in seconds between data sampling iterations. TrackMe will never attempt to perform sampling for a given entity if the time since the last iteration is lower than this value.	3600
splk_data_sampling	Defines the number of records to be sampled and verified per entity during the data sampling process. Increasing this value improves event format recognition but requires more processing per entity.	10000
splk_data_sampling	Defines the number of records kept in the KVstore for inspection purposes at each iteration. This sample is stored for inspection and review, and increasing this value requires more KVstore storage.	10
splk_data_sampling	Defines the character size limit before truncating events when storing sampled records in the KVstore for inspection. This truncation only affects storage and does not impact the model training process.	40000
splk_data_sampling	Defines the minimum percentage of events that must match the major inclusive model. If the main model has less than this percentage of matching events, the entity's state will be impacted.	98
splk_data_sampling	Defines the maximum percentage of events matching an exclusive model that can be accepted. By default, no events matching an exclusive model are accepted, but this can be increased per entity.	95
splk_data_sampling	Defines the size of the time window for sampling operations in seconds, relative to the latest event time known for the entity. This time window is used to calculate the earliest time for sampling searches.	3600

8.10.9 Backup & restore

TrackMe backs up its KV-store state and knowledge objects to compressed archives, and restores them per tenant, per whole run, or selectively. This page is an overview — the *Backup & Restore white paper* has the full step-by-step procedure.

How backups are structured

A backup **run** produces **one archive per tenant plus one global archive**, rather than a single monolithic file (which became unrestorable at multi-gigabyte scale). Each run has a `RUN_ID` and a manifest listing every archive and its checksum:

- a **per-tenant** archive holds that tenant's collections, knowledge objects, and account record;
- the **global** archive holds everything else — the tenant registry, maintenance, bank holidays, license, ML models, and Guardian state.

Crucially, the archives are **independent**: a single corrupt archive is marked failed and does not abort the rest of the run, a restore, or a delete.

Restoring

There are three restore modes — a **single archive**, a **whole run**, or a **legacy** flat archive from before the multi-archive format — and you can **selectively** restore just specific collections or knowledge objects.

Important

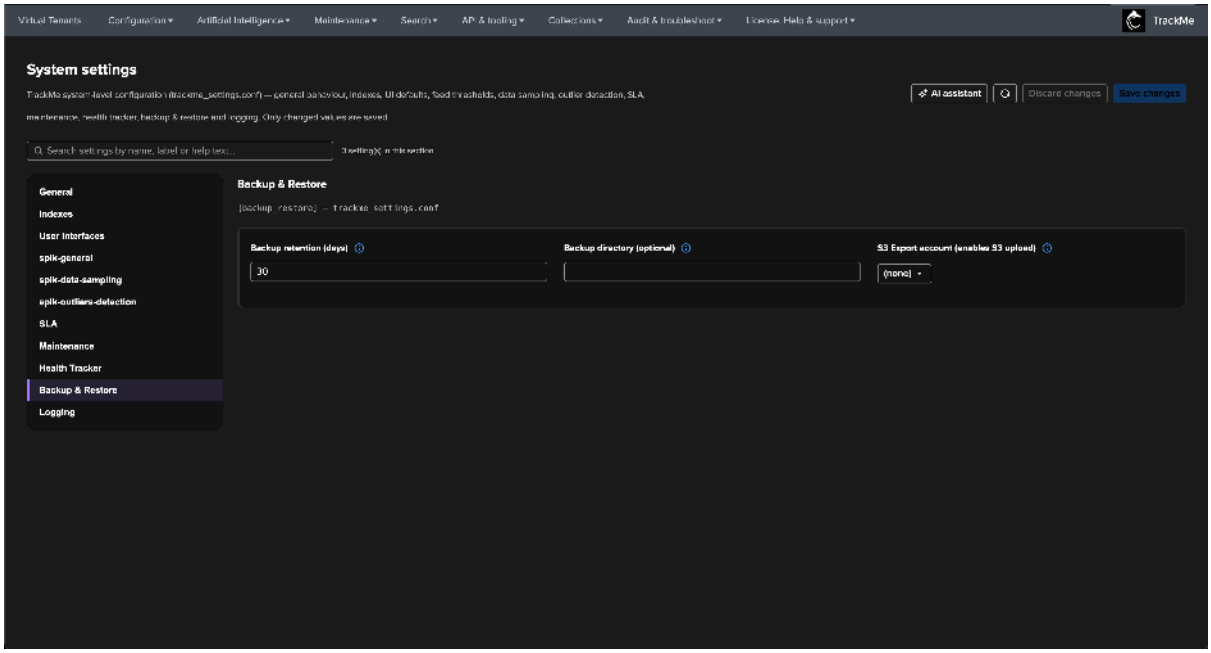
Always run a restore as a dry run first, review what it will change, then run it for real. A single-tenant restore overwrites that tenant's live state by design — coordinate with operators, ideally inside a maintenance window.

Scheduling, storage, and integrity

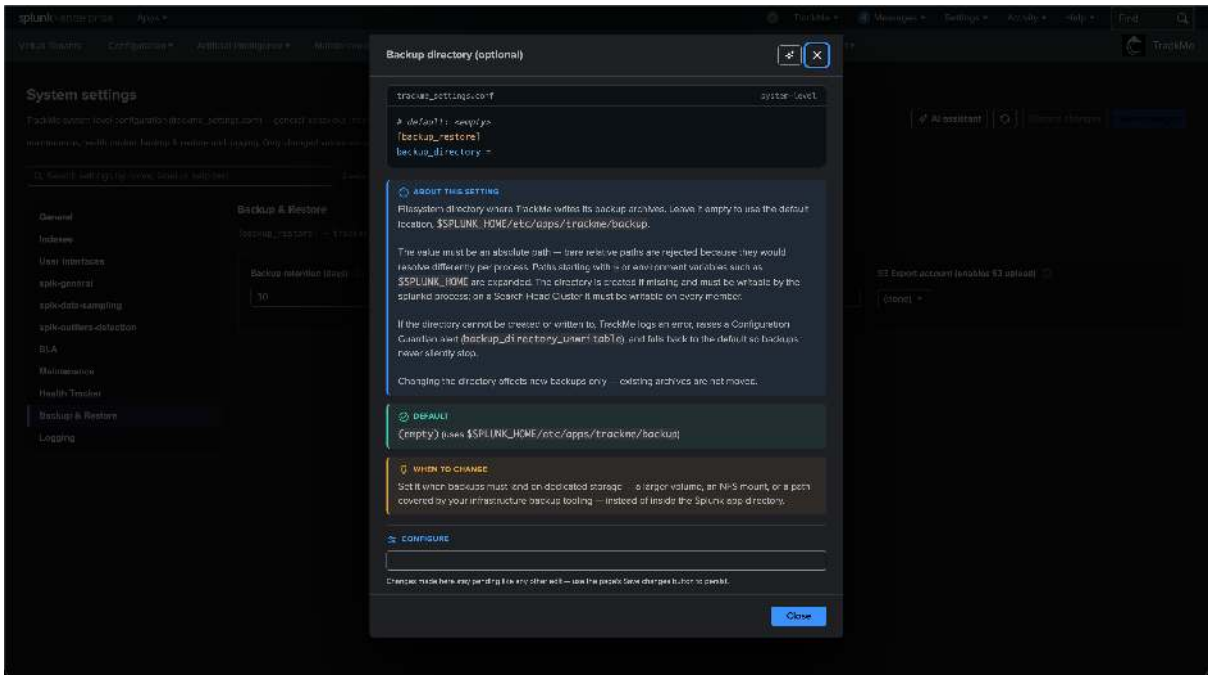
- A **scheduled backup** runs daily and is enabled automatically once you create your first tenant, with a retention sweep.
- Every archive is integrity-checked when produced, and you can validate one before a critical restore.
- The **backup directory** and **retention** (in days) are configurable under **Configuration → System settings → Backup & Restore** (see below), with a fail-safe fallback to the default location (see *Backup storage — retention and location (2.4.3)*).
- Two *Configuration Guardian* checks watch this — one for an archive that has grown too old, and one for a run that is missing tenants.

Where the backup settings live

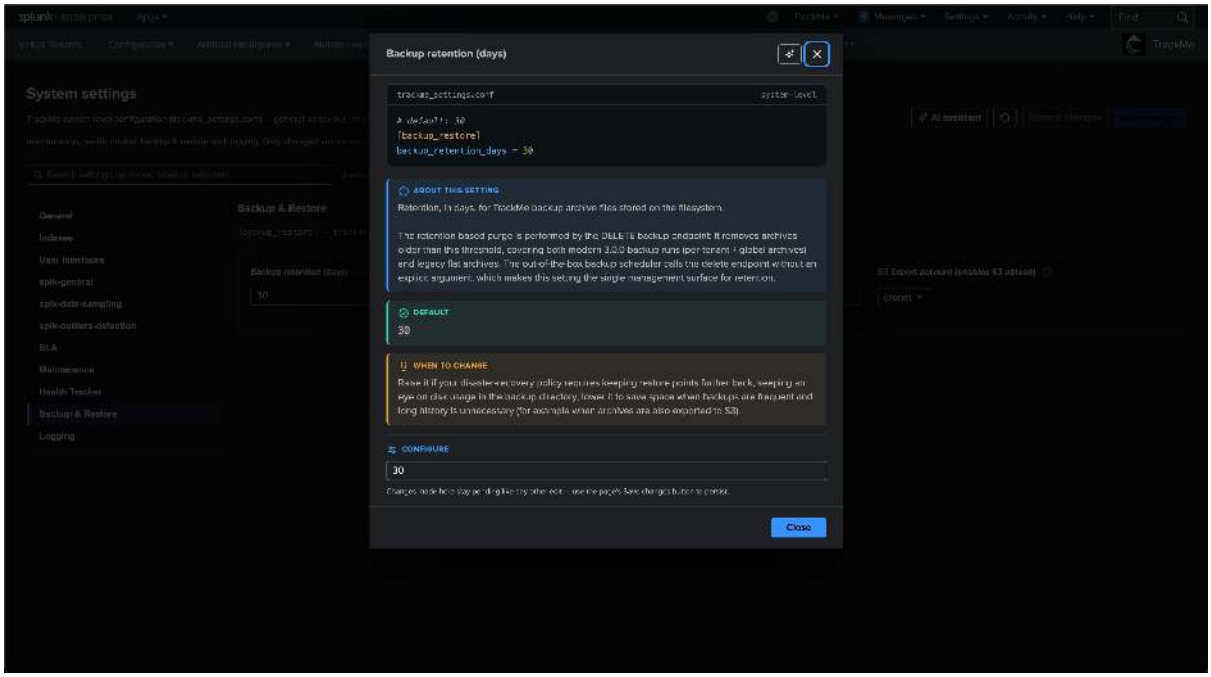
The location, retention and S3 export target are all set from **Configuration → System settings → Backup & Restore**. The section holds three settings — **Backup retention (days)**, **Backup directory (optional)** and **S3 Export account (enables S3 upload)** — and, like every System setting, each carries an info (?) button explaining what it does, its default, and when to change it:



Backup location. By default, archives are written to `$SPLUNK_HOME/etc/apps/trackme/backup`. Set **Backup directory (optional)** to an absolute path (env vars and ~ are expanded) to write them elsewhere — a larger volume, an NFS mount, or a path covered by your infrastructure backup tooling. If the directory cannot be created or written to, TrackMe logs an error, raises the `backup_directory_unwritable` Configuration Guardian alert, and **falls back to the default** so backups never silently stop. On a Search Head Cluster the path must be writable on every member, and changing it affects new backups only.



Backup retention. **Backup retention (days)** (default `30`) controls the age-based purge of local archives, covering both modern per-tenant/global runs and legacy flat archives. Raise it if your disaster-recovery policy requires keeping restore points further back; lower it to save disk space (for example when archives are also exported to S3).

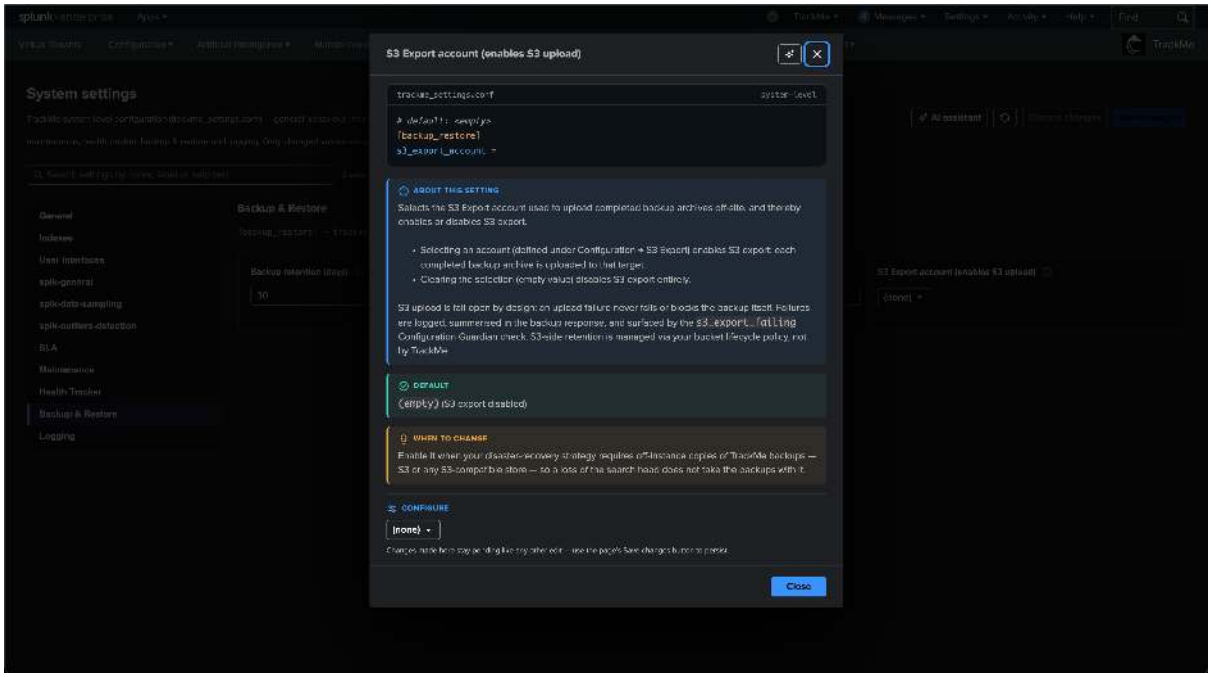


Off-site export to S3

From 2.4.3, completed archives can be **exported off-site to S3** — Amazon S3 or any S3-compatible store (MinIO, Ceph, …) — for a disaster-recovery copy outside the Splunk deployment, with no external scripting. It is:

- **Opt-in** — nothing is uploaded until you create an *S3 Export* account (under **Configuration → Manage S3 export accounts**) and select it in the Backup & Restore settings. Connectivity is verified on save, so a broken target cannot be saved silently.
- **Fail-open** — if S3 is unreachable the backup still completes and the local archives are always intact; the upload failure is recorded per archive, never raised.
- **Additive** — S3 export is an extra copy of the same archives, uploaded after each run.

Once you have created an S3 Export account, enable off-site upload by selecting it in the **S3 Export account (enables S3 upload)** dropdown of **Configuration → System settings → Backup & Restore**. Leaving it on `(none)` keeps S3 export disabled; selecting an account uploads every completed backup archive to that target from the next run onwards:



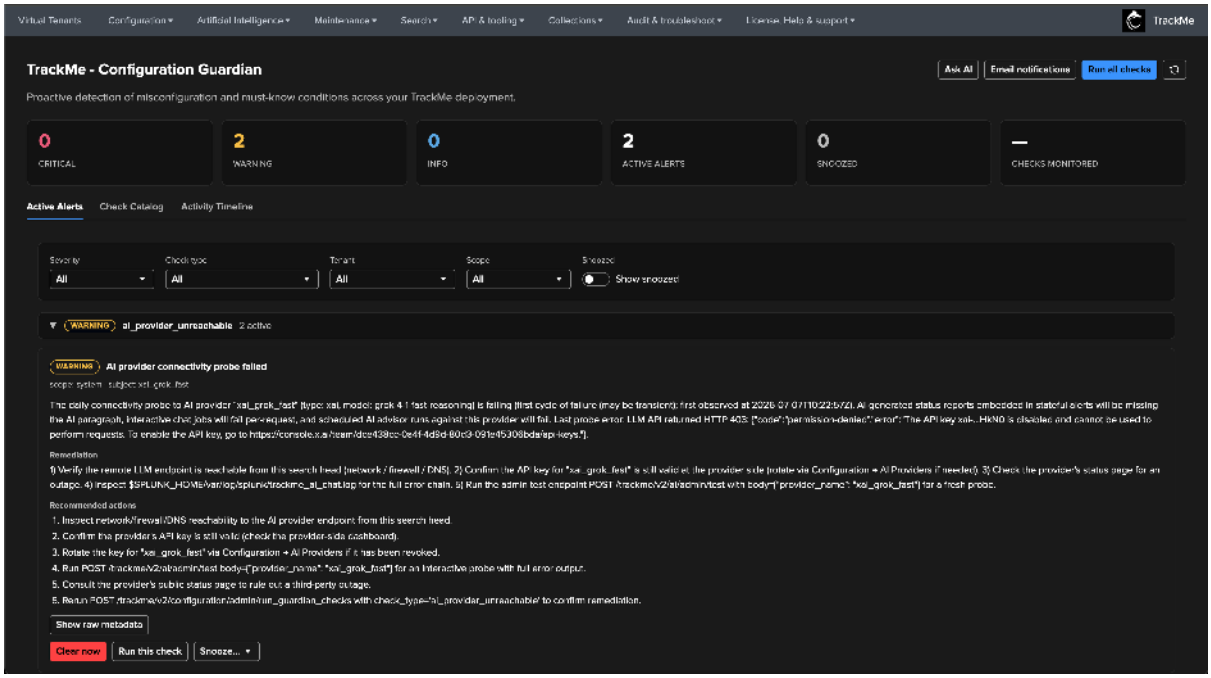
TrackMe does **not** delete objects from the bucket — S3-side retention is left to a bucket lifecycle policy. A third Configuration Guardian check, `s3_export_failing`, surfaces an unreachable target or a run that failed to upload. The full setup is in the *white paper*.

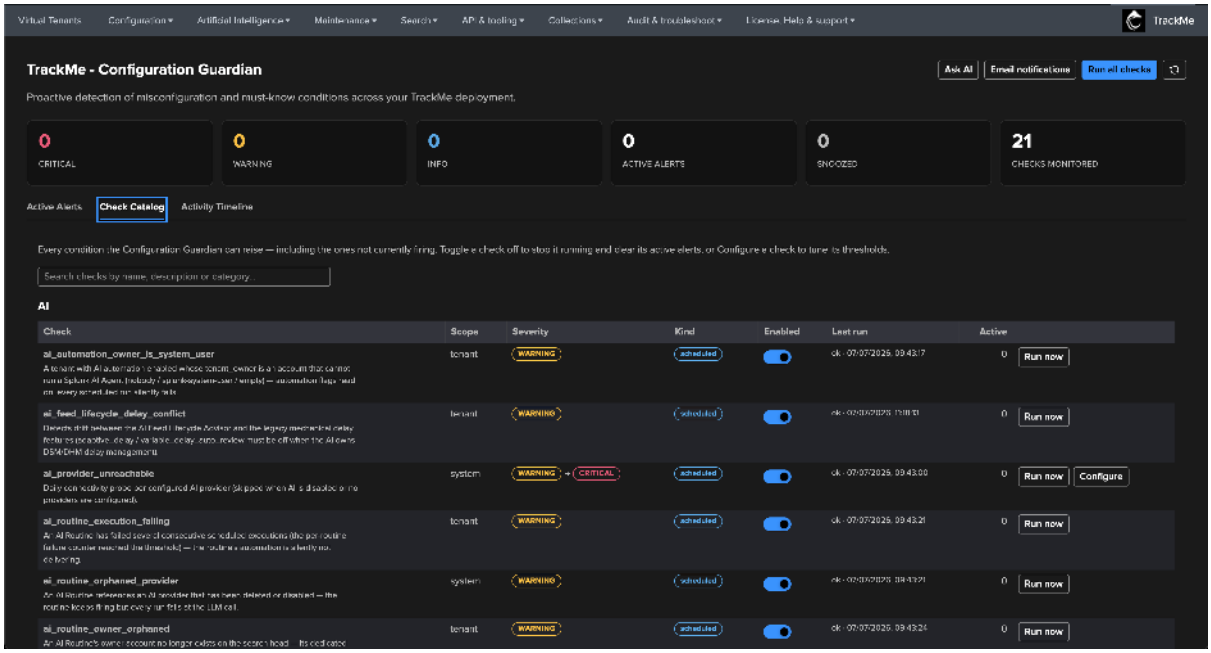
See also

- *Backup & Restore white paper* — the full procedure.
- *Off-site S3 export* — configuring an S3 target.
- *Configuration Guardian* — the checks that watch your backups.
- *Configuration Manager (CI/CD)* — config-as-code, the complement to state backups.

8.10.10 Configuration Guardian

The Configuration Guardian is a built-in framework that continuously checks TrackMe's *own* setup and surfaces problems before they cost you data or missed alerts. Each finding is severity-rated and comes with concrete, ordered remediation steps — which both an admin and the AI Assistant can act on directly.





In short:

- A modular framework of **checks** runs continuously in the background (through TrackMe's scheduled health components), each validating one specific condition — service-account permissions, remote-account connectivity and token expiration, backup freshness, AI-provider reachability, degraded tenants, stale maintenance windows, and more.
- When a condition is detected, the Guardian raises an **alert** carrying a human-readable title, message, remediation guidance and a structured list of **recommended actions** — actionable both for administrators and for the TrackMe AI Assistant.
- Alerts are **self-healing**: as soon as the underlying condition is resolved, the next detection cycle clears the alert automatically. No manual housekeeping required.

Hint

Where to find it

- The Configuration Guardian page is available under **Configuration → Configuration Guardian**.
- Active Guardian alerts also surface as **toast notifications** on the Virtual Tenants page, each with a *View in Configuration Guardian* link that deep-links to the alert.

About the checks themselves

The catalog of checks grows over release cycles and each check is fully described in the product itself: the **Check Catalog** tab documents every check (including the ones that are currently quiet), its severity behaviour, its scheduling host and its tunable parameters. For this reason, this page focuses on the framework and its management features rather than maintaining a per-check list.

Severities, scopes and self-healing

Every Guardian check declares:

- **A scope**: `tenant` (one alert per affected Virtual Tenant) or `system` (deployment-wide conditions, such as a remote account or a backup issue).
- **A severity**:
  - `info` — a sticky informational notice (for example, the record of an automatic self-healing action). Informational notices do not clear automatically — dismiss them once read.
  - `warning` — attention needed, but the platform keeps working (a fallback or degraded state).
  - `critical` — active degradation: data loss, dropped alerts, or a meta-failure such as the health tracker itself not executing.
- **An escalation ladder** (for probe-style checks): transient conditions start at `warning` on first detection and **escalate automatically to critical** once the condition persists beyond a configurable window (24 hours by default for most checks, tunable per check in the Check Catalog). The alert's metadata tracks since when the condition has been failing.

Findings surface in three places: **toasts** in the Virtual Tenants UI (coloured by the highest severity present), the **REST API** (a structured list of active alerts with their recommended actions), and the **audit index** (every raised, updated and cleared alert). And they **self-heal** — an alert auto-clears when you fix the underlying condition. Dismissing one in the UI is only temporary: if the condition persists, the next cycle raises it again.

Examples of what it checks

A few illustrative checks — the full, current list lives in the in-product **Check Catalog**:

Check	Severity	Catches
Tenant owner missing a capability	warning	A service account that cannot do its job.
Assigned index does not exist	warning	A tenant index slot pointing at a missing index (or a non-metric metric index).
Remote-account token expiring soon	warning → critical	A bearer token about to expire (see *Remote Splunk deployments*).
Remote-account connectivity degraded	warning → critical	A remote deployment that has stopped responding.
TrackMe Broker offline	warning → critical	No forwarder of a broker has pulled for a while, so calls through it fail. It covers a forwarder accepted that never pulled, and a broker left with no forwarder only if accounts depend on it (see *Pools, leases and failure*).
TrackMe Broker degraded	warning	One of a broker's forwarders stopped while others carry on, or their target lists differ.
Backup archive too old / run incomplete	warning → critical	Backups that have gone stale or missed tenants (see *Backup & restore*).
S3 export failing	warning → critical	An off-site S3 export target that is unreachable, or a run that failed to upload (see *Backup & restore*).
Health tracker not executing	critical	The meta-check — a tenant's own health tracker has stopped running.
Topology view integrity (2.4.15+)	warning / critical	Stale tenant/component references, missing or unresolvable referenced views, and inconsistent Topology Alerts; see *View integrity and consistency checks* for detection, severity and worked remediation examples.

Detection hosts and cadence

Checks are executed by TrackMe's existing scheduled health components (there is no additional scheduled search to manage):

- **Per-tenant checks** run within each tenant's health tracker cycles.
- **System-wide checks and per-tenant meta-checks** (checks that must not depend on the tenant's own tracker — such as detecting that a health tracker is not executing, or that a tenant is operationally degraded) run within TrackMe's general health manager (daily cadence).
- Every check can also be executed **on demand** at any time (see *Running checks on demand*).

Some checks also refresh in response to configuration changes. For example, *Topology Studio integrity* refreshes on view and alert writes and triggers a background scan after tenant lifecycle changes, with the daily general health manager run acting as a safety net.

The two *TrackMe Broker* checks also run every 15 minutes, through the report `trackme_broker_watch`. TrackMe schedules it only while a broker is declared. See *their alerts* for what each says and recommends.

A special case: **inline safeguards** (such as the sourcetype-explosion safeguard) detect their condition in real time within the processing hot path, rather than through periodic scans. They are surfaced **read-only** in the Guardian view — marked with an *INLINE SAFEGUARD* badge — so administrators get a single must-know list; they can be cleared but not snoozed, disabled or tuned.

Permissions

Read-only access to the Configuration Guardian page:

- TrackMe user (capability `trackmeuseroperations`).
- Tenant-scoped alerts are RBAC-filtered: users only see alerts for the Virtual Tenants they are allowed to access. System-scoped alerts are generally visible to everyone accessing the page, with administrator-only exceptions

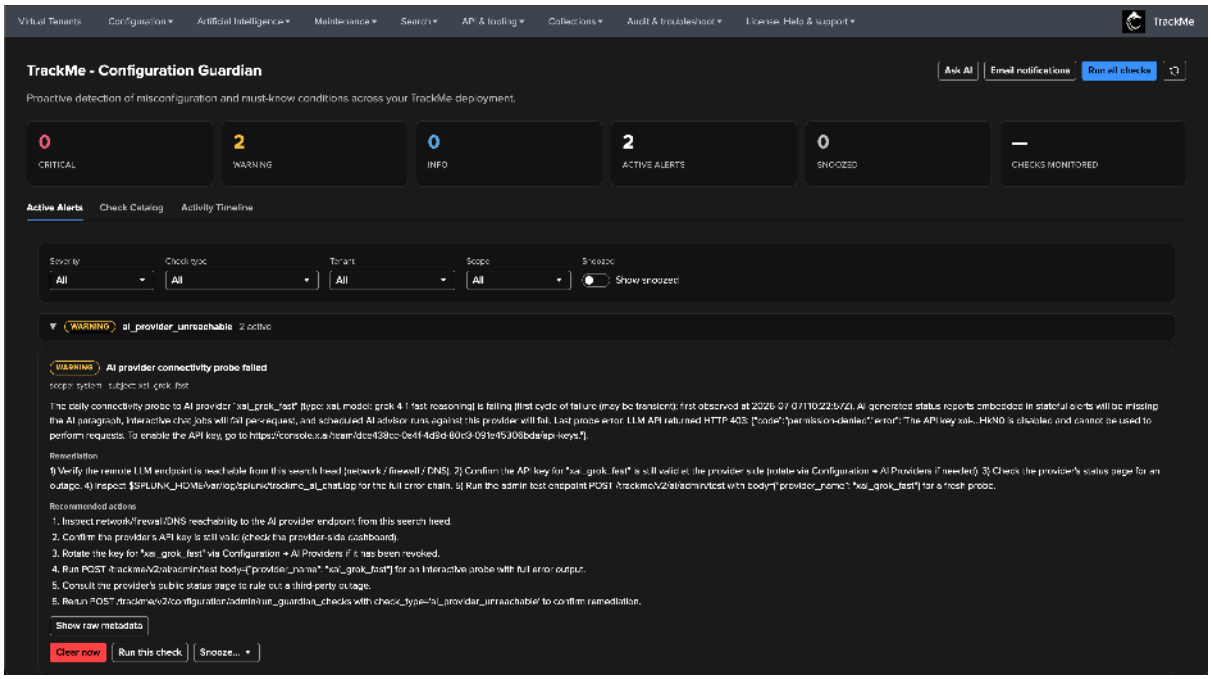
for checks whose details require additional access controls. The **Topology Studio view integrity** check is one such exception: its shared Guardian findings and Virtual Tenants toasts are administrator-only; other users inspect findings in Topology Studio under the view's read permissions and tenant visibility rules.

Management actions (snooze, clear, enable/disable checks, tuning, on-demand runs, email notifications): TrackMe admin (capability `trackmeadminoperations`).

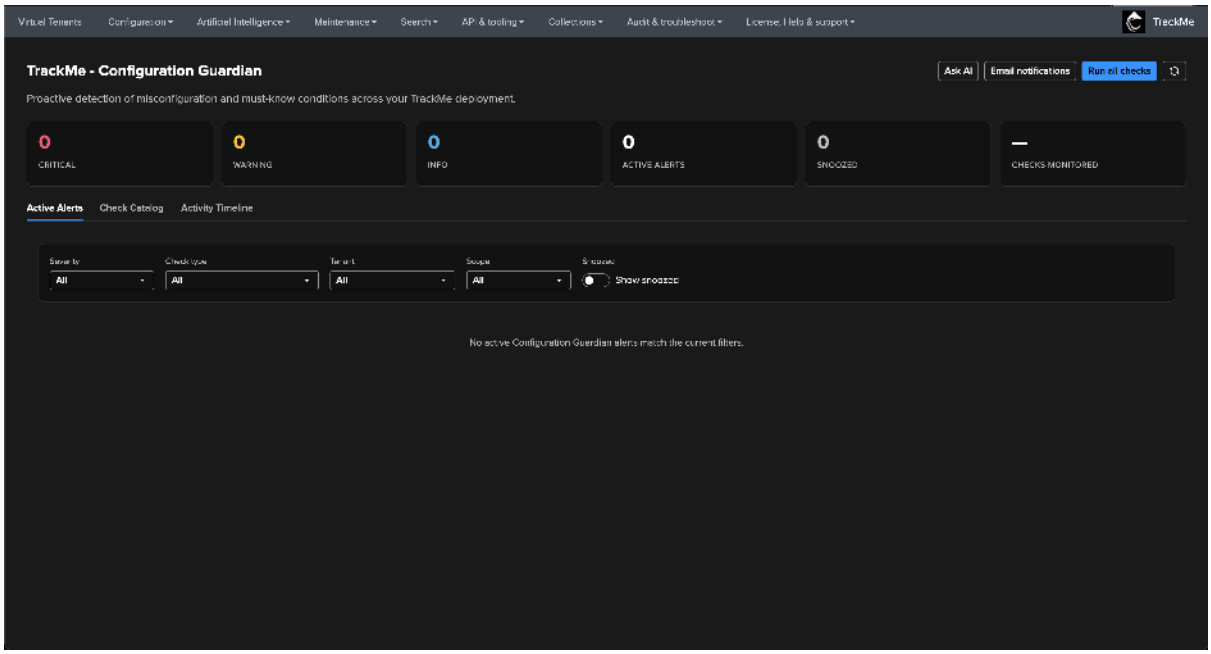
The Configuration Guardian page

The page is organized in three tabs, with a KPI strip on top summarizing critical / warning / info counts, active alerts, snoozed alerts and the number of monitored checks.

Active Alerts



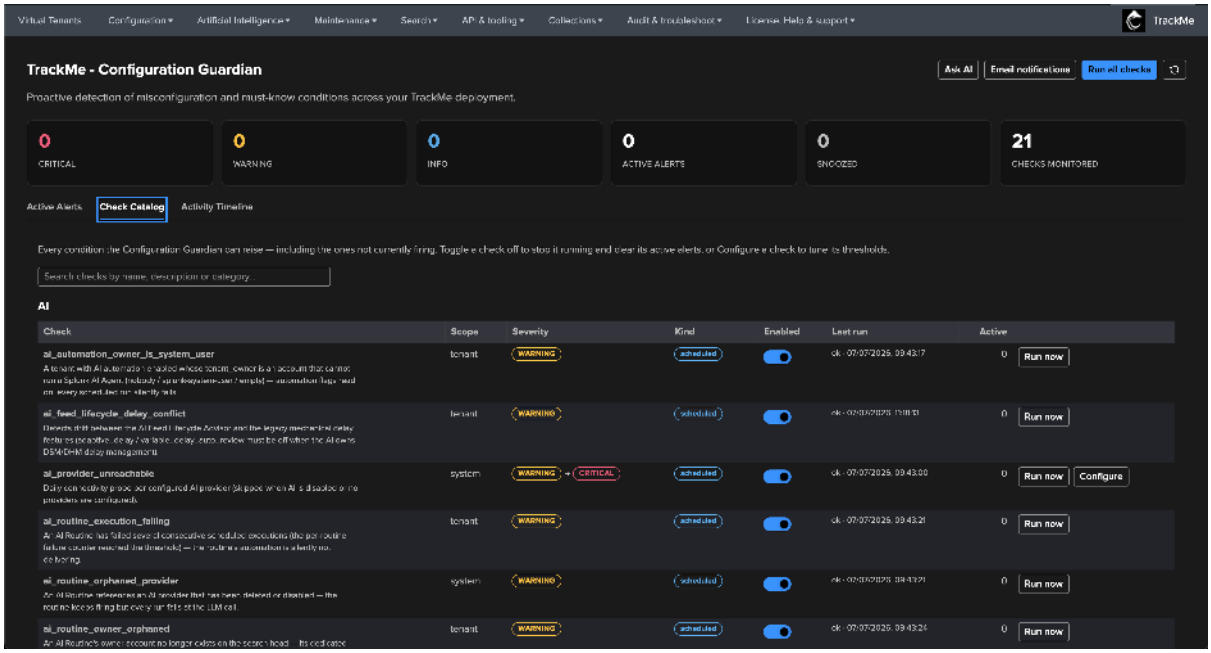
The alert inbox: every active alert, grouped by check type and ordered by severity, with filters on severity, check type, tenant, scope, and a *Show snoozed* toggle. On a healthy deployment, the tab simply reports that no alerts match:



Each alert renders its full context: severity badge, scope and tenant, the detection message, the remediation guidance, the structured recommended actions, and the raw metadata (*Show raw metadata*). Per-alert actions:

- **Clear now** — immediately deletes the alert. This is a *temporary* action by design: if the underlying condition still holds, the alert is re-created on the next detection cycle. Use it to force a re-evaluation, or to acknowledge an informational notice.
- **Snooze…** — silences the alert for a chosen duration (1, 7 or 30 days) with an optional reason (recorded in the audit trail). See *Snooze semantics* below.
- **Run this check** — re-executes the corresponding check on demand.
- **Ask AI** — opens the TrackMe AI Assistant pre-contextualized on the Guardian, so you can ask for explanations and remediation guidance about the alert (requires the AI Assistant to be configured).

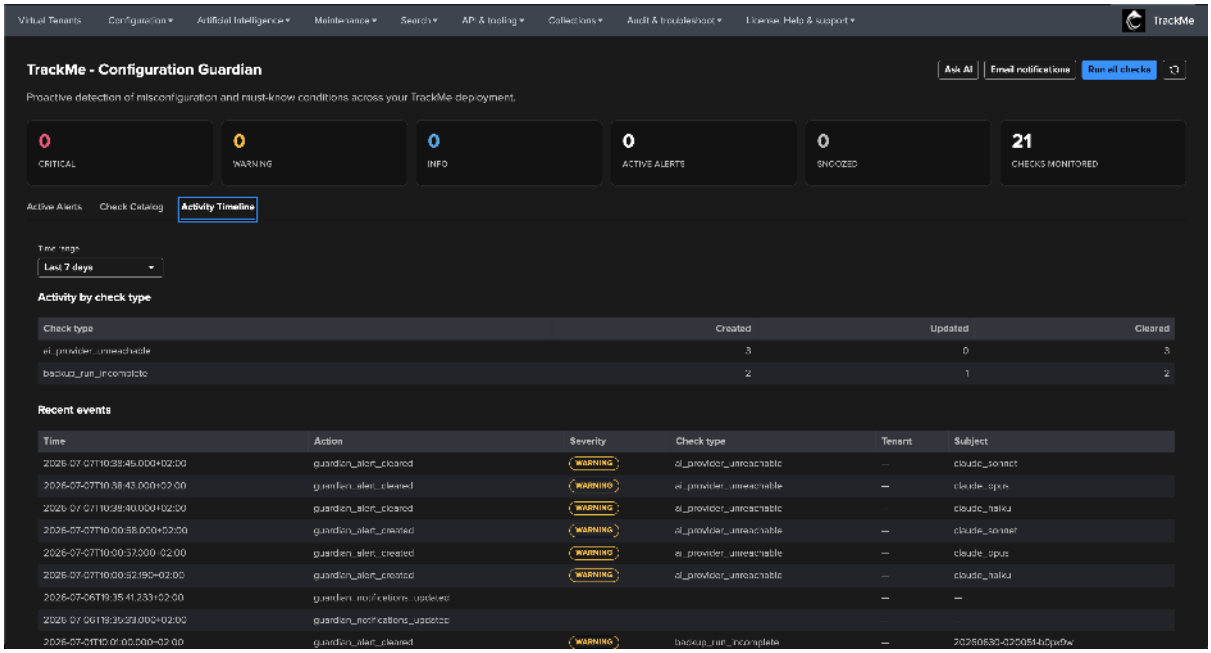
Check Catalog



The catalog lists **every check the Guardian can fire — including the ones that are currently quiet** — so administrators can discover what is monitored without waiting for something to go wrong. For each check:

- Description, scope, category, severity behaviour (including the warning → critical escalation ladder when applicable) and scheduling host.
- **Enable / disable toggle** — disabling a check stops its scheduled executions and clears its currently active alerts; re-enabling resumes scanning automatically on the next cycle.
- **Configure** — per-check threshold tuning through a generated configuration modal (for example: token-expiry warning/critical windows, backup-freshness ceilings, escalation windows). Leaving a value at its default keeps the built-in behaviour — only explicit differences are persisted, so future TrackMe default adjustments keep applying to untouched parameters.
- **Run now** and the check's **last run** status (when it last executed, by which host, and its outcome).
- A search bar to quickly locate a check.

Activity Timeline



The timeline is driven by the Guardian's audit trail (see *Audit trail and generated events*): per-check counts of created / updated / cleared transitions over a selectable time range, plus a feed of the most recent events. Use it to answer questions like "which check is the noisiest?" or "when did this condition first appear?".

Snooze semantics

Snoozing is the Guardian's way of saying *"I know about this condition, stop showing it to me for a while — but don't stop watching it"*:

- **Detection never stops.** A snooze is a read-time suppression: the check keeps running and the alert record keeps being updated, but the alert is hidden from the toasts and from the default Guardian views until the snooze expires. The *Show snoozed* toggle reveals snoozed alerts, flagged with a *SNOOZED* badge, and the KPI strip counts them separately.
- **Critical breaks through.** A snooze remembers the severity it was taken at. If a condition snoozed at `warning` later escalates to `critical`, the alert surfaces anyway — flagged *BROKE THROUGH SNOOZE* — because you silenced a warning, not an active degradation.
- **Expiry is automatic.** When the snooze expires, the alert simply surfaces again if the condition still holds; if it self-healed in the meantime, nothing comes back.
- Snoozes are keyed per check and per subject (for example, per tenant), so snoozing one tenant's alert does not silence the same check on other tenants. An optional free-text reason is recorded with the snooze and lands in the audit trail.

Hint

Clear now vs Snooze vs Disable

Action	Detection	When does it come back?
Clear now	keeps running	immediately on the next cycle if the condition persists
Snooze	keeps running	after the snooze expires (or on critical escalation)
Disable check (Catalog)	stops entirely, alerts cleared	only when the check is re-enabled

Running checks on demand

Administrators can force a scan at any time:

- **Run all checks** (page header) — executes every enabled check across every enabled Virtual Tenant.
- **Run this check** — from an alert or from the Check Catalog, optionally narrowed to the alert's tenant.

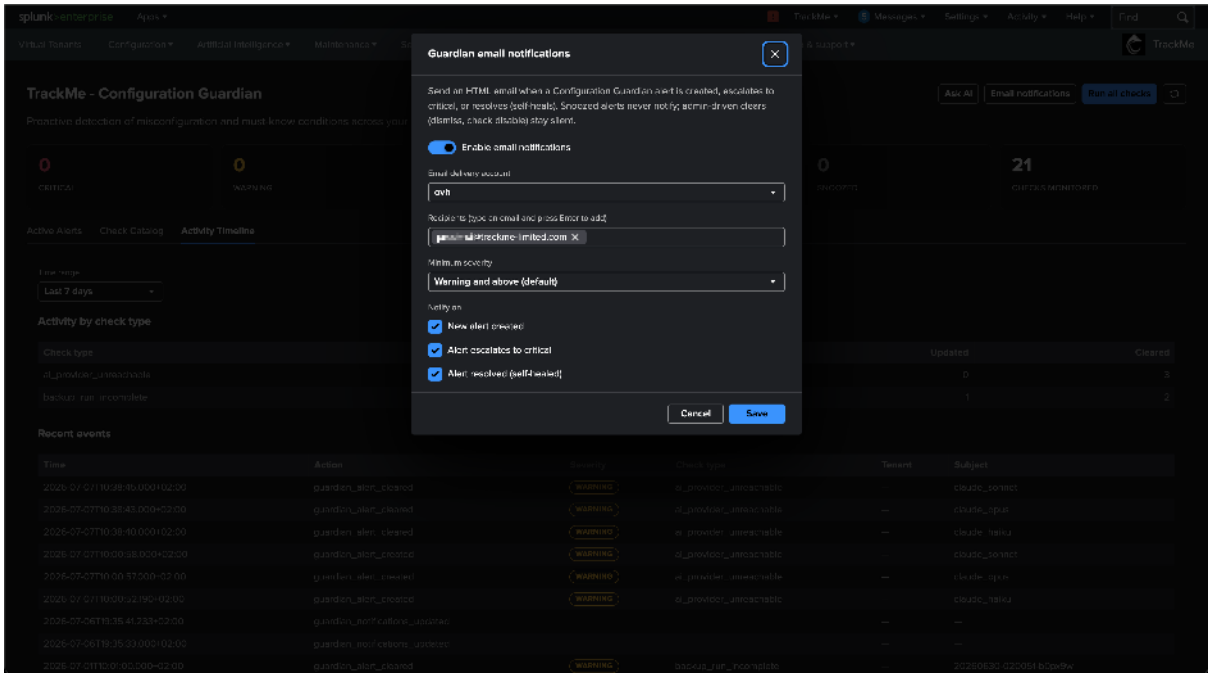
On-demand runs from the UI are dispatched as a **Splunk search job** running the `trackmeguardianrunchecks` command. This matters on larger deployments: a fleet-wide run can legitimately take several minutes (some checks probe remote accounts and AI providers with generous per-subject timeouts), and running it as a search job guarantees the action cannot be interrupted by web-proxy or load-balancer timeouts.

The same command is available directly from the Splunk search bar:

```
| trackmeguardianrunchecks

| trackmeguardianrunchecks tenant_id="mytenant" check_type=
↪"tenant_operations_degraded"
```

The command returns one summary row: `status`, `created`, `cleared`, `unchanged`, `skipped` and `run_time` (a failed run yields `status="failure"` plus a `message` field). Both arguments are optional; with no filters, every registered check runs against every enabled Virtual Tenant. The command requires the `trackmeadminoperations` capability.

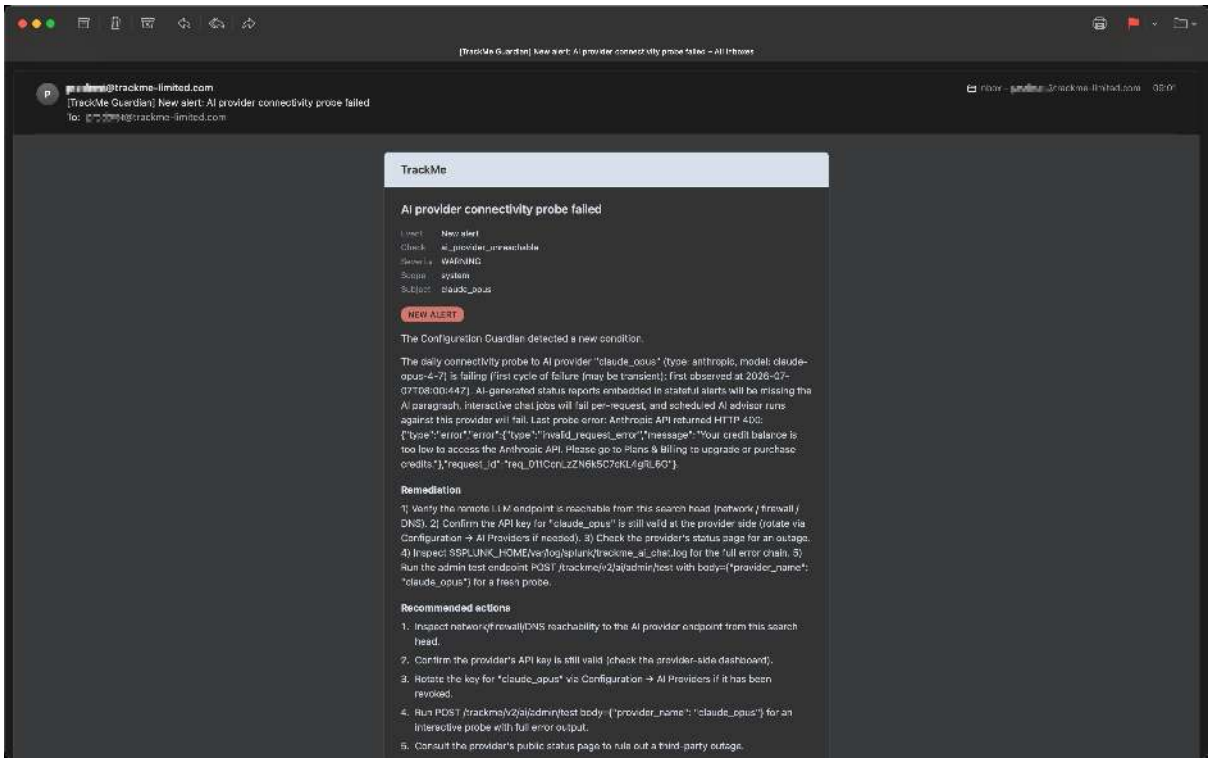
Email notifications

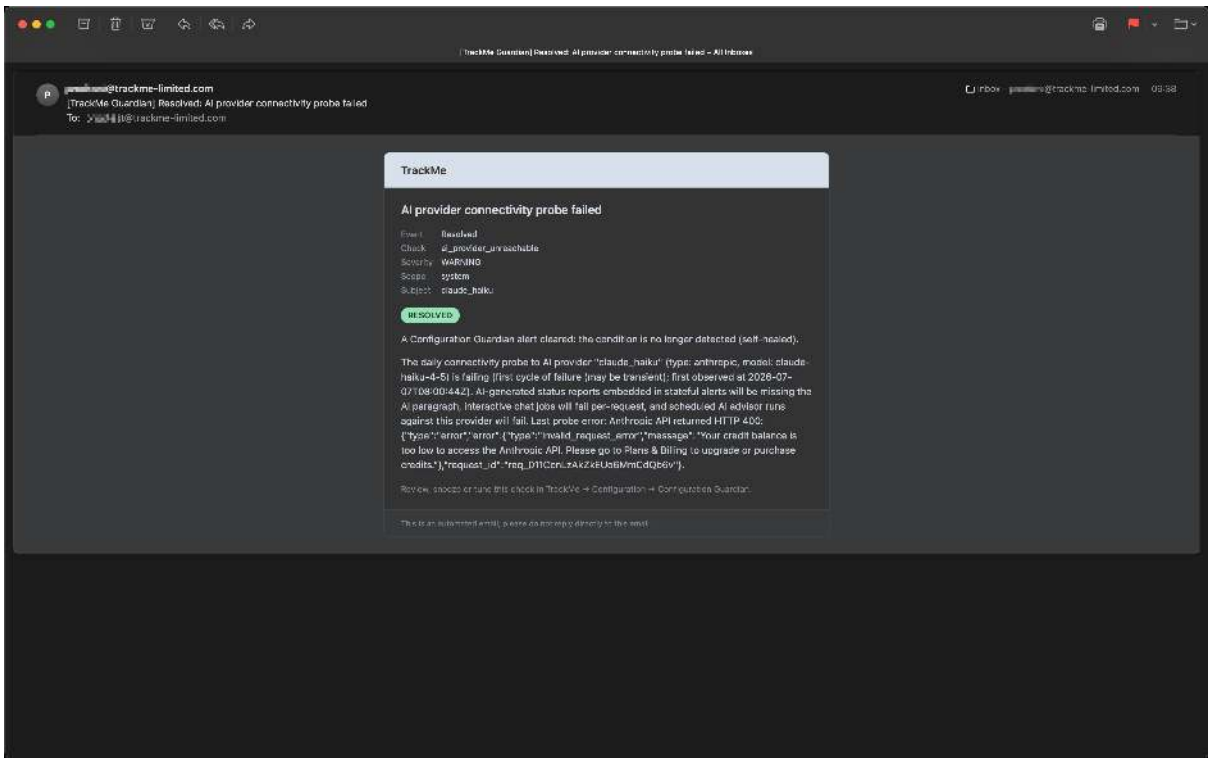


The Guardian can notify administrators by email on alert lifecycle transitions. Click **Email notifications** in the page header to configure:

- **Email delivery account** — any account configured under **Configuration → Emails delivery** (or the default localhost MTA).
- **Recipients** — one or more email addresses (the delivery account's allowed email-domains policy applies).
- **Minimum severity** — `warning` by default, so informational notices never generate emails; set to `critical` to only be notified about active degradations.
- **Events** — three independent toggles: **New alert created**, **Alert escalates to critical**, and **Alert resolved** (the condition self-healed).

Notifications are rendered as TrackMe-branded HTML emails (with a plain-text fallback) carrying the alert's message, remediation guidance and recommended actions; the resolved variant is a compact all-clear.





Notification semantics — designed to stay trustworthy

- Emails fire on **genuine transitions only**: routine re-detections of an unchanged condition never notify.
- **Snoozed alerts never notify** — you muted them. A critical alert breaking through a lower-severity snooze does notify, consistently with the UI behaviour.

- A **resolved** email is only sent when the condition actually went away (self-healed). Administrative actions — clearing an alert manually, disabling a check, deleting a tenant — stay silent.
- Delivery is **fail-open**: an SMTP problem can never affect detection; failures are logged and the next transition tries again.

Alert actions integration (advanced)

For teams routing notifications through their existing alerting pipelines (email templates, Splunk alert actions, third-party integrations), TrackMe additionally ships an **opt-in** saved search: `trackme_guardian_critical_notifier`.

- Disabled by default; enable it and attach any Splunk alert action(s) of your choice.
- It fires when an alert **becomes critical** — either created as critical, or escalated from warning to critical — with per-alert throttling so a persistent condition notifies once, not every cycle.

Audit trail and generated events

Every Guardian state transition is recorded in the TrackMe audit index (`trackme_audit` by default, honouring your `trackme_audit_idx` configuration) under the sourcetype `trackme:audit:guardian`:

Action	Emitted when
`guardian_alert_created`	a new alert is raised for a (check, subject) pair
`guardian_alert_updated`	an existing alert changes severity or meaningful metadata
`guardian_alert_cleared`	an alert is cleared (self-healed, manually, or via tenant deletion — the `reason` field says which)
`guardian_alert_snoozed`	an administrator snoozes an alert
`guardian_alert_unsnoozed`	an administrator removes a snooze before expiry
`guardian_check_enabled`	a check is re-enabled from the Check Catalog
`guardian_check_disabled`	a check is disabled from the Check Catalog
`guardian_check_config_upd`	a check's tunable parameters are updated
`guardian_notifications_up`	the email notification configuration is updated

Each event carries the alert key, check type, scope, tenant, subject, severity (and prior severity on transitions) and title. Routine re-detections of an unchanged condition are deliberately **not** re-emitted, so the audit trail reflects genuine state changes. Example searches:

```
`trackme_audit_idx` sourcetype="trackme:audit:guardian"␣
↪action="guardian_alert_created" severity="critical"

`trackme_audit_idx` sourcetype="trackme:audit:guardian" check_type="<check_type>"␣
↪| sort - _time
```

State storage

The Guardian persists its state in two global KV Store collections, also exposed as lookups:

- `trackme_configuration_guardian_alerts` (collection `kv_trackme_configuration_guardian_alerts`): one record per active alert.
- `trackme_configuration_guardian_settings` (collection `kv_trackme_configuration_guardian_settings`): the management state — per-check enable/disable and tuning overrides, active snoozes, per-check last-run bookkeeping, and the email notification configuration.

```
| inputlookup trackme_configuration_guardian_alerts
```

On-demand run logs

Executions of the `trackmeguardianrunchecks` command log to the sourcetype `trackme:custom_commands:trackmeguardianrunchecks` (index `_internal`).

REST API

The Guardian exposes a complete REST surface under `/services/trackme/v2/configuration` — listing alerts, running checks, clearing, snoozing, enabling/disabling and tuning checks, and managing the email notification configuration. Refer to the built-in **REST API Reference** page (API & tooling menu) for the full, self-documenting endpoint catalog; the on-demand scan endpoint is also the recommended integration point for programmatic callers:

```
| trackme mode=post url="/
↪services/trackme/v2/configuration/admin/run_guardian_checks" body="{}"
```

AI Assistant integration

When the TrackMe AI Assistant is configured:

- The **Ask AI** button (page header and per alert) opens a chat pre-loaded with the Guardian context: the assistant sees the active alerts, their recommended actions and the general knowledge of every check, and can guide you through remediation and verification.
- The Virtual Tenants and Tenant Home assistant contexts also include the Guardian state, so questions like *"why aren't my entities updating?"* can be answered by cross-checking the relevant Guardian alerts.

See also

- *Remote Splunk deployments* — remote accounts the Guardian watches.
- *Backup & restore* — the backup checks.
- *Events & sourcetypes* — the audit sourcetype Guardian writes to.
- *Artificial Intelligence* — the AI Assistant that reads live Guardian state.

8.10.11 Configuration Manager (CI/CD)

The TrackMe Configuration Manager (TCM) brings **config-as-code** to TrackMe. It captures your admin operations as replayable transactions, so configuration flows from a non-production environment, through source control, into production — instead of being hand-applied on the live system.

How it works

TCM runs in two modes:

- **Receiver mode** (non-production) — every admin-level operation you perform (creating a tenant, editing a policy, and so on) is recorded as a transaction in a journal.
- **Producer / replay mode** (production) — the production instance reads the journal and replays the **exact same REST calls**, applying each transaction **exactly once** (it is idempotent, keyed on the transaction).

The workflow is: configure in non-production → your CI/CD pipeline picks up the journal and version-controls it with the TCM release → release to production → TCM replays it.

Note

- TCM is a **separate add-on** (`TA-trackme-conf-manager`), downloaded from the TrackMe downloads site rather than Splunk Base, so Cloud customers can repackage and self-publish it.

- In a non-production **search-head cluster**, the journal is a local file and is not replicated across members — run TCM on a standalone non-production instance, or manage journal synchronisation yourself.

See also

- *CI/CD & config manager — in depth* — the producer/receiver config keys, the receiver endpoint and journal, `trackmeconfmanager`, and Cloud vs Enterprise packaging.
- *REST API reference* — the REST calls TCM records and replays.
- *Backup & restore* — state backups, the complement to config-as-code.

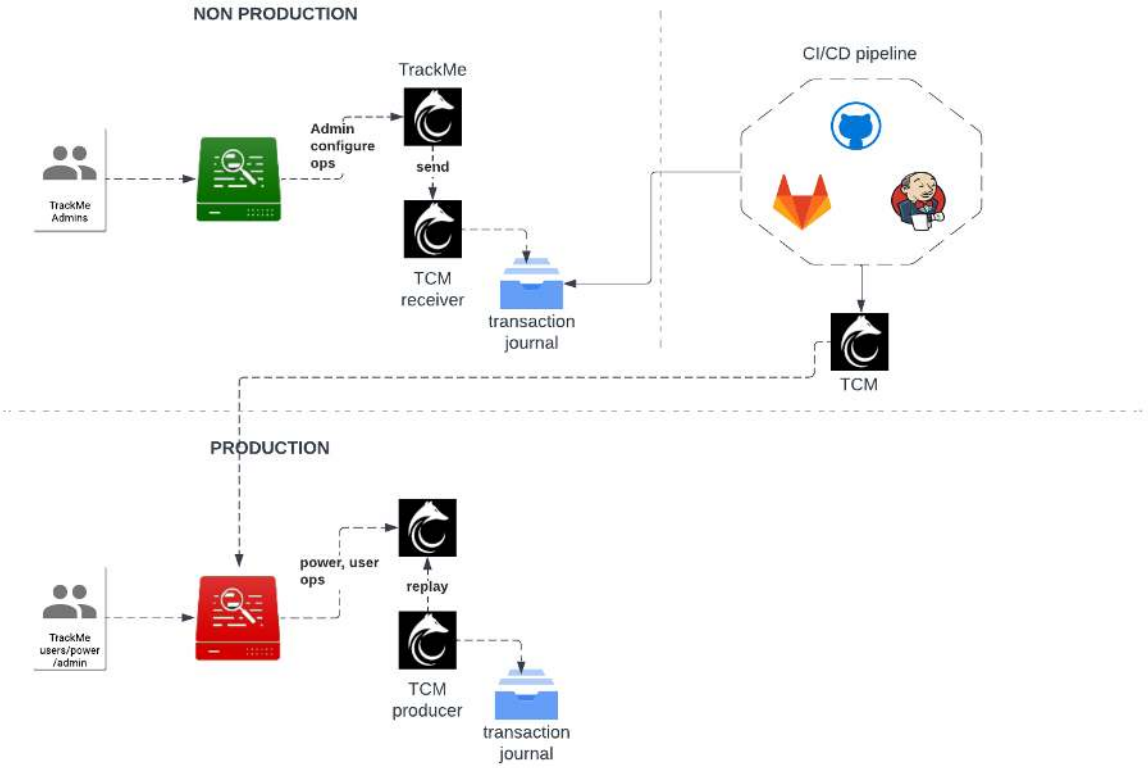
8.10.12 CI/CD & config manager — in depth

Tip

This is the **in-depth reference** for TrackMe CI/CD management (TCM) / the configuration manager. For the readable overview, start with *Configuration manager*. This page covers the producer/receiver config keys and modes, the receiver REST endpoint and transaction journal, the `trackmeconfmanager` command, and Splunk Cloud vs Enterprise packaging.

Minimal version required: 2.0.36

- The following documentation and capabilities require version 2.0.36 and later
- It also requires an additional package called TCM, which stands for the TrackMe Configuration Manager
- This package is provided to all TrackMe users at the following URL: https://downloads.trackme-solutions.com/TA-trackme-conf-manager



Introduction to the TrackMe Configuration Manager

In some contexts, it is not allowed to perform any kind of configuration in a Production environment context, and all configuration need to be performed first in a non Production environment, published to a versioning source control and finally released in a strict manner to the target environment(s).

TrackMe is a highly flexible and dynamic solution which implies on the fly creation of knowledge objects and associated content, generally stored in KVstores.

Therefore, typical approaches that work for more basic applications cannot be transposed to complex applications such as TrackMe, to achieve these goals we provide a configuration manager tool called TrackMe Configuration Manager. (TCM)

Downloading TCM

The TCM package is not available on Splunk Base, it can be freely downloaded at the following URL on our download Website:

- https://downloads.trackme-solutions.com/TA-trackme-conf-manager

We do not publish this package in Splunk Base on purpose, notably to allow Splunk Cloud customers to repackage TCM (as described in the next sections) and publish it to Splunk Cloud easily.

Principles

The TCM workflow relies on the following logic:

In non-Production:

- In the non-Production environment, the TCM is deployed and running in **receiver** mode
- TrackMe is configured via a built-in option to enable sending **transactions** to the TCM
- When an admin-level operation is made in non-Production, such as creating a new Virtual Tenant, the relevant transaction is written to the TCM JSON journal file

- The TCM journal is then taken in charge by your CI/CD pipeline, packaged with your TCM release, and published to your versioning control

In Production:

- The TCM is deployed and configured in **producer** mode
- The built-in TCM scheduled backend is enabled
- When the Python backend is executed, **transactions** from the TCM journal are read and verified
- If the transaction has not been processed yet, TCM runs the associated operation by performing the exact same REST API calls to TrackMe that were made in non-Production
- Once the transaction has been performed, a record is written to the TCM resilient KVstore; transactions are uniquely associated with a transaction MD5
- At the next iteration, TCM knows that the transaction is done already and does not need to be processed again

Configuring TrackMe and TCM in non-Production

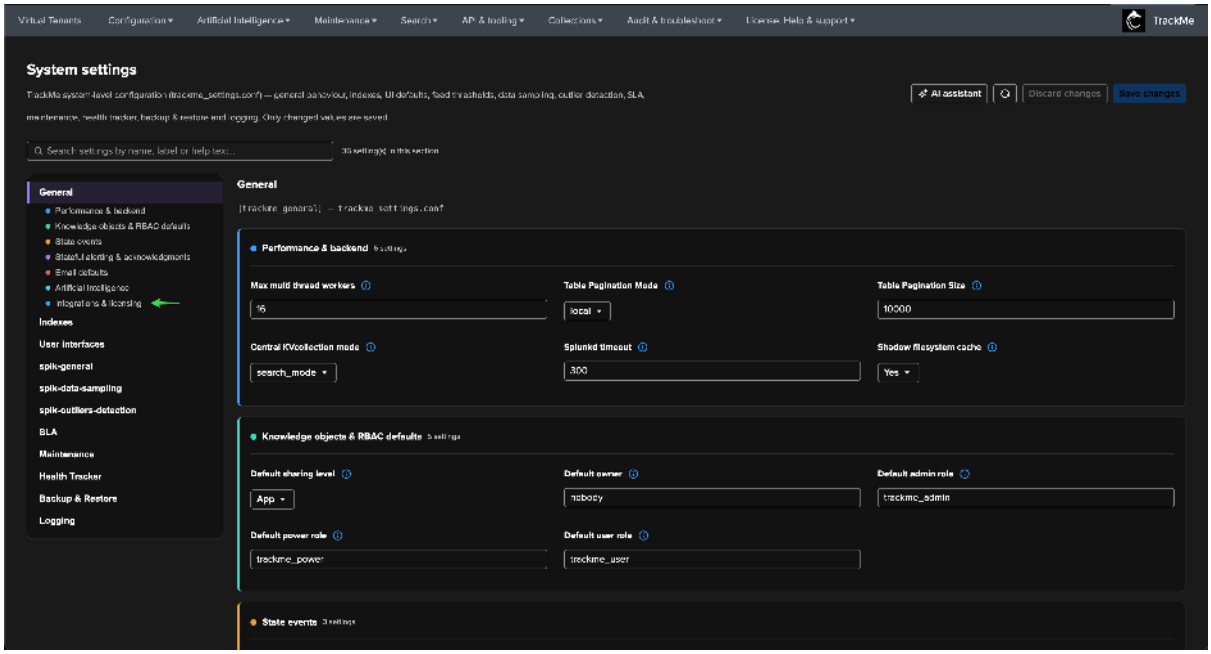
Note about SHC in non-Production

- Transactions are written to a local file in the TCM application, which would not be replicated amongst the members of the SHC
- Therefore, the TCM can be used in non-Production in an SHC cluster; however, you will need to manage the synchronization of the journal file
- Ideally, you should run TCM in a standalone non-Production environment to avoid having to deal with this question

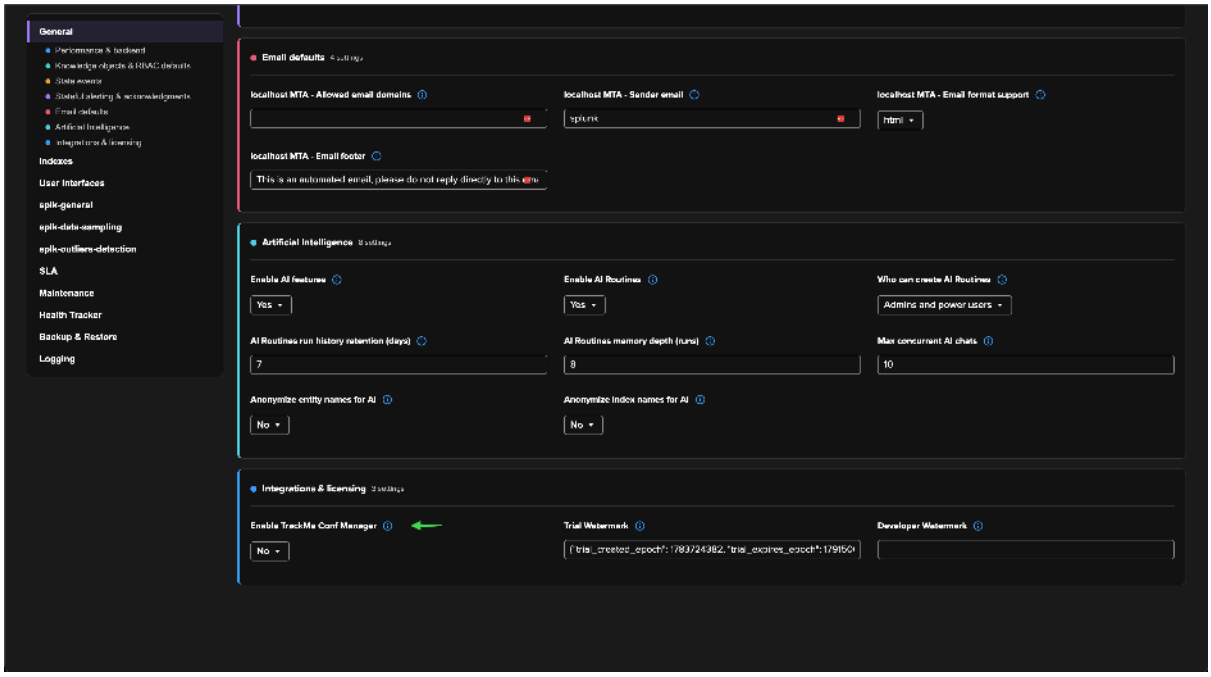
Non-Production workflow

In non-Production, which means where the configuration of TrackMe is made, the TCM is deployed in its default configuration, which acts in **receiver** mode by default.

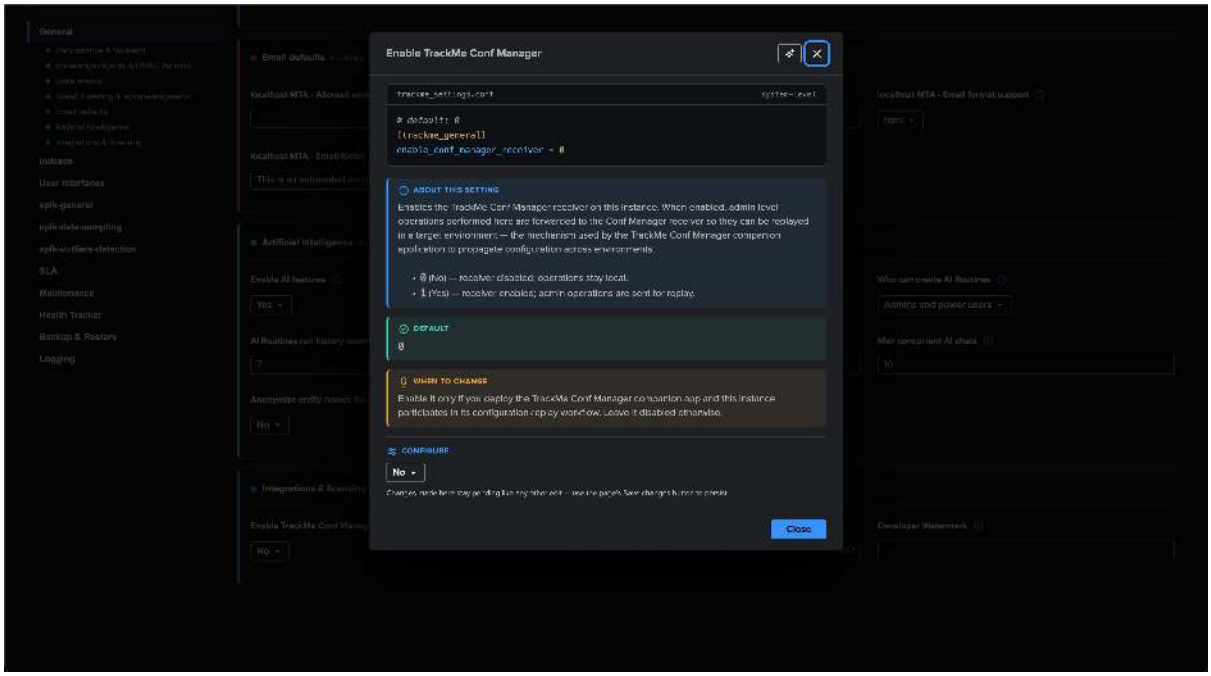
TrackMe needs to be instructed to send the transactions to TCM. The setting lives under **Configuration → System settings → General**, in the **Integrations & licensing** group:



Set Enable TrackMe Conf Manager to Yes to turn the receiver on:



The setting's info (?) button documents exactly what it toggles — when enabled, admin-level operations performed on this instance are forwarded to the Conf Manager receiver so they can be replayed in a target environment:



Which corresponds to the following option in TrackMe configuration file:

```
[trackme_general]
# Enable the TrackMe Config Manager (TCM) and sets the mode, the TCM is used for␣
↪CI/CD purposes, the receiver mode sends the transaction to TCM, the replay mode␣
↪replays transactions from TCM
enable_conf_manager_receiver = 0
```

By enabling this option, when TrackMe performs an admin level operation which creates and manages contents such as Virtual Tenants, TrackMe performs a REST call to the TCM API:

```
https://<local splunkd uri>/services/trackme_conf_manager/v1/conf_manager_receiver
```

The TCM API receives the following information:

- `transaction_request`: The request payload for the TrackMe API call, these are all the instructions underneath a REST call made to TrackMe's API endpoints
- `transaction_http_mode`: The requested HTTP mode for the transaction (GET, POST, DELETE)
- `transaction_http_service`: The TrackMe API service that was requested for this transaction, this is the TrackMe API endpoint that was contacted for this operation

TrackMe REST API logs the TCM forwarding actions:

success:

```
trackme_send_to_tcm was successfully executed
```

Example:

```
index=_internal sourcetype=trackme:rest_api trackme_send_to_tcm
```

```
2023-06-05 21:55:38,258 INFO trackme_rest_handler_splk_replica_trackers_admin.py␣
↪post_replica_tracker_create 224 trackme_send_to_tcm was successfully executed
```

In case of failures:

```
trackme_send_to_tcm has failed with exception=<exception>
```

On the file-system, the TCM writes transactions to a JSON journal file in local:

```
/opt/splunk/etc/apps/TA-trackme-conf-manager/local/
↪trackme_conf_manager_transactions_journal.json
```

Each transaction is a JSON object which represents the REST call to be performed depending on the actions requested in TrackMe, through the UI and the REST API:

Example:

```
"9c2f7f8264a86913f34a97e161b128e8": {
    "transaction_request": {
        "tenant_desc": "SIEM",
        "tenant_name": "feeds-tracking",
        "tenant_roles_admin": [
            "trackme_admin"
        ],
        "tenant_roles_power": [
            "trackme_power"
        ],
        "tenant_roles_user": [
            "trackme_user"
        ],
        "tenant_owner": "svc-trackme",
        "tenant_idx_settings": "{\"trackme_summary_idx\": \"trackme_summary\", \
↪"trackme_audit_idx\": \"trackme_audit\", \"trackme_notable_idx\": \
↪"trackme_notable\", \"trackme_metric_idx\": \"trackme_metrics\"}",
        "tenant_dsm_enabled": true,
        "tenant_dsm_sampling_obfuscation": "disabled",
        "tenant_dhm_enabled": true,
        "tenant_dhm_alerting_policy": "track_per_host",
        "tenant_mhm_enabled": true
    },
    "transaction_http_mode": "post",
```

(continues on next page)

(continued from previous page)

```
    "transaction_http_service": "/services/trackme/v2/vtenants/admin/add_tenant",
    "ctime": 1686000734.0977793
}
```

A transaction is identified by a unique MD5 hash, this same hash will be used by TCM on the producer side (therefore in Production) to identify transactions which have been processed already, and transactions which are pending.

Configuring TrackMe and TCM in Production

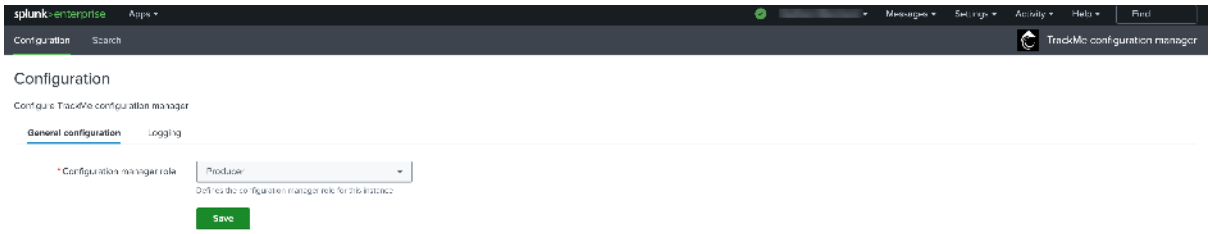
Note about SHC in Production

- SHC is fully supported for all TCM replay transactions

TCM packaging and deployment

In Production, therefore the target environment where transactions will be replayed, the TCM must be deployed in Producer mode:

Through the UI:



Which corresponds to the following configuration:

ta_trackme_conf_manager_settings.conf

```
[trackme_general]
conf_manager_role = producer
```

TCM packaging:

Depending on your context:

- If you are a **Splunk Cloud** customer, you need to include the journal file in the default directory, package the application release and deploy it through Splunk ACS (through the API or UI), once Appinspect is passed, TCM is ready
- If you are a **Splunk Enterprise** customer, you can equally publish the journal in default or local, but local has precedence over the default directory

TrackMe configuration

TrackMe does not require any configuration for TCM.

TCM acts independently once it is deployed and configured, and will interact with TrackMe as needed to replay the transactions and apply your settings accordingly.

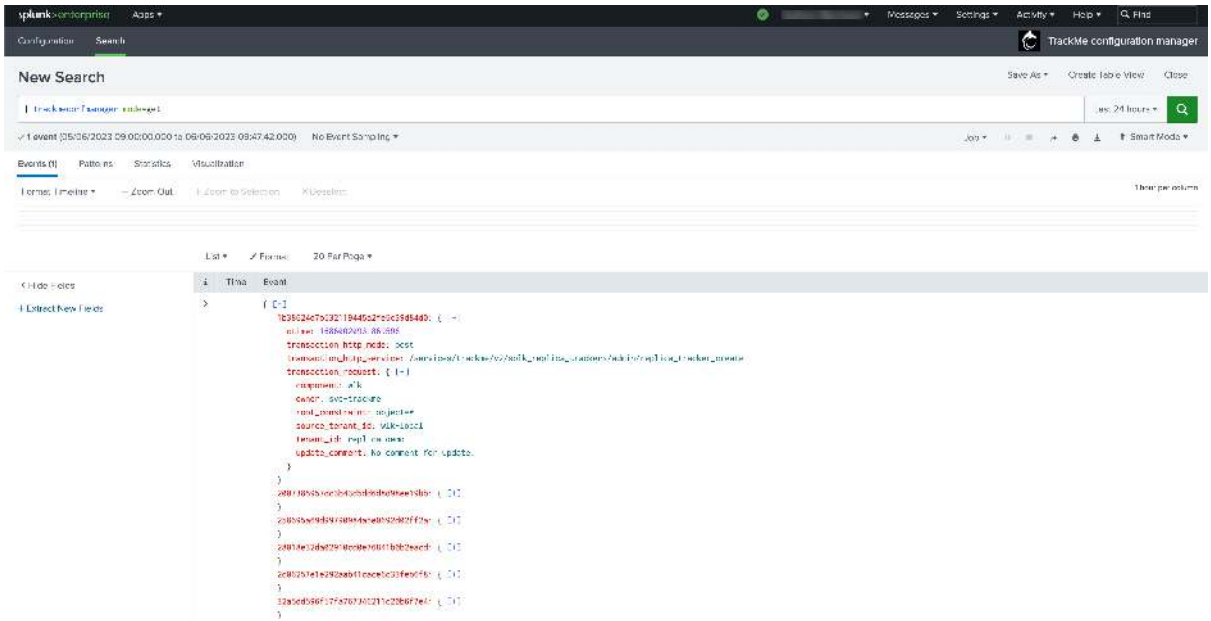
Note that forwarding to TCM must be **disabled** in the Production context, which is the default: `enable_conf_manager_receiver = 0` under **Configuration → System settings → General**, in the **Integrations & licensing** group. The Production instance is the *target* where TCM replays transactions, so it must not itself forward — otherwise a replayed change would be forwarded straight back.

Viewing the transaction journal

You can access the transaction journal via the following command:

```
| trackmeconfmanager mode=get
```

Example:



Checking transactions statuses

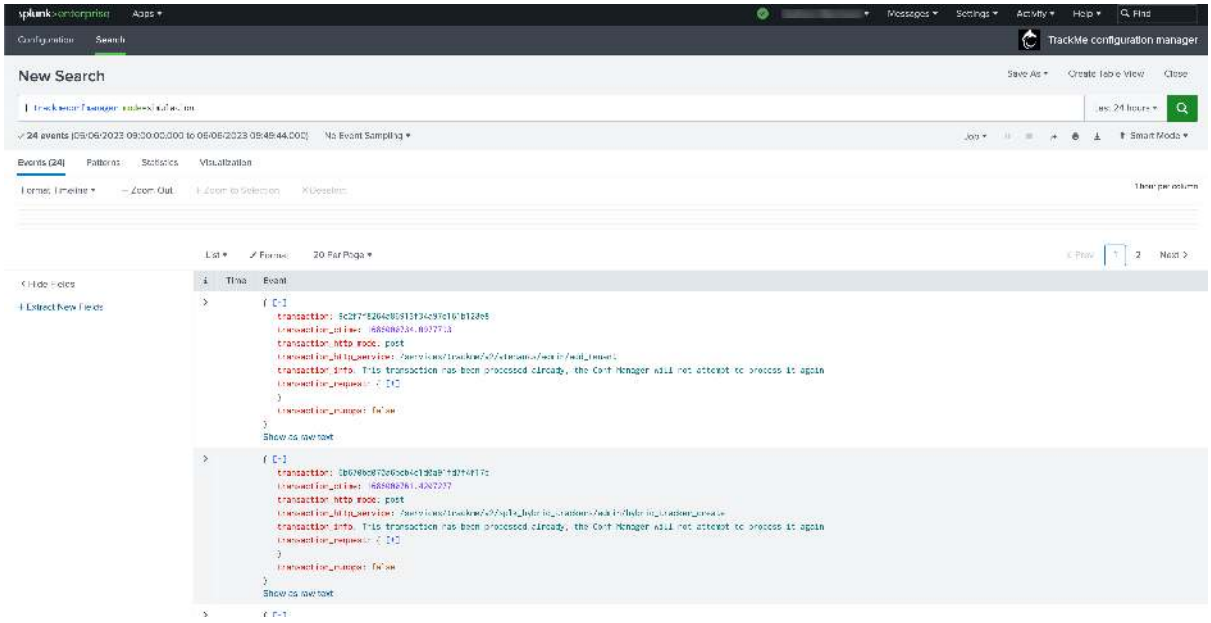
You can check the status of transactions via the following command:

```
| trackmeconfmanager mode=simulation
```

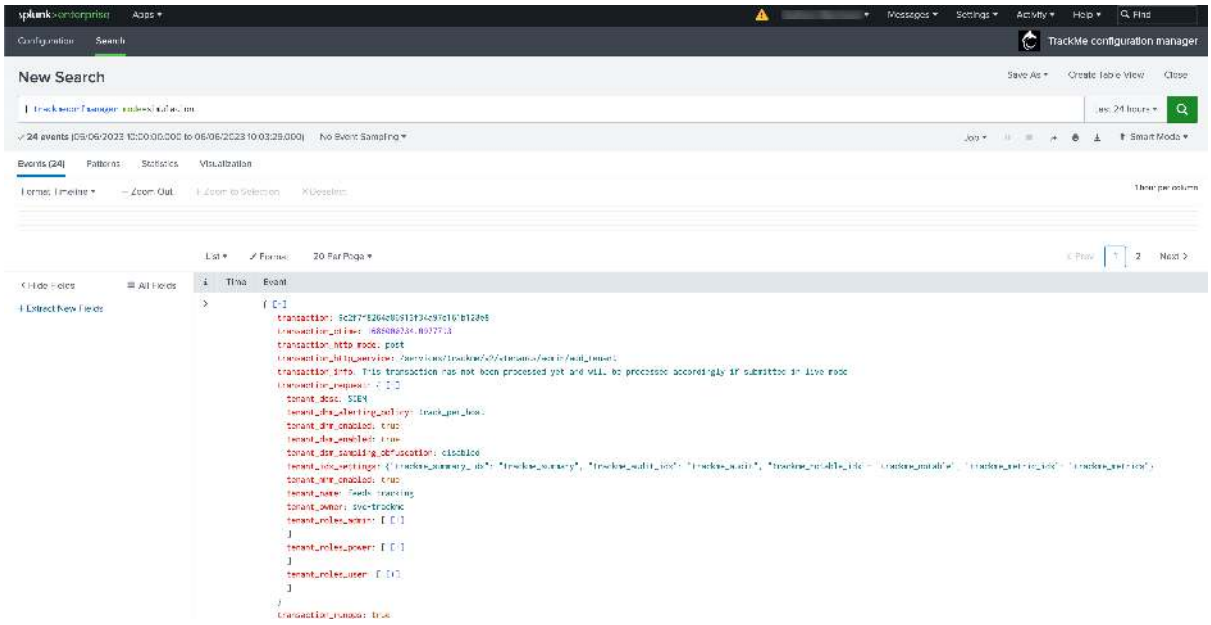
A transaction which has not been completed yet a `transaction_info` message explaining the action which will be taken.

A transaction which has been completed already shows a `transaction_info` message: "This transaction has been processed already, the Conf Manager will not attempt to process it again".

In the following example, all transactions have been proceeded:



In the following example, transactions are pending and would be proceeded:



Enabling the scheduled report trackme_conf_manager_producer

The TCM packages provides a built-in scheduled report which is disabled by default; the report runs the following command:

```
| trackmeconfmanager mode=live
```

This is the process which replays transactions and configures TrackMe according to the transaction journal provided by your CI/CD pipeline.

The process works as follows:

- For each transaction, the Python backend verifies if the transaction can be found in the resilient KVstore `kv_trackme_conf_manager`:

```
| inputlookup trackme_conf_manager | eval key=_key
```

- If the transaction has not been processed, TCM performs the associated call to the TrackMe REST API

- If the transaction has been processed, then there are no actions to be taken

The TCM transaction replay backend logs its activity into the following logs:

```
index=_internal sourcetype=trackme_conf_manager:trackmeconfmanager
```

Troubleshooting TCM

TCM receiver REST API

When a transaction is sent from the non-Production environment to TCM, activity is logged here:

```
index=_internal sourcetype=trackme_conf_manager:rest_api
```

example:

```
2023-06-05 21:54:12,145 INFO trackme_conf_manager_rest_handler.py␣
↪post_conf_manager_receiver 216 conf manager received configuration, resp_dict="{
↪'transaction_request': {'tenant_id': 'replica-demo', 'source_tenant_id':
↪'cim-demo', 'component': 'cim', 'root_constraint': 'object=*', 'update_comment':
↪'No comment for update.', 'owner': 'svc-trackme'}, 'transaction_http_mode': 'post
↪', 'transaction_http_service': '/
↪services/trackme/v2/splk_replica_trackers/admin/replica_tracker_create'}"
2023-06-05 21:54:12,158 INFO trackme_conf_manager_rest_handler.py␣
↪post_conf_manager_receiver 284 {'response': 'transaction received with␣
↪md5=5df36dfae06b237408a0cecd30e87015 and successfully written to the journal',
↪'journal': '/
↪opt/splunk/etc/apps/TA-trackme-conf-manager/local/trackme_conf_manager_transactions_journal.json
↪'}
```

TCM transaction replay (command trackmeconfmanager)

Activity of the trackmeconfmanager command is logged here:

```
index=_internal sourcetype=trackme_conf_manager:trackmeconfmanager
```

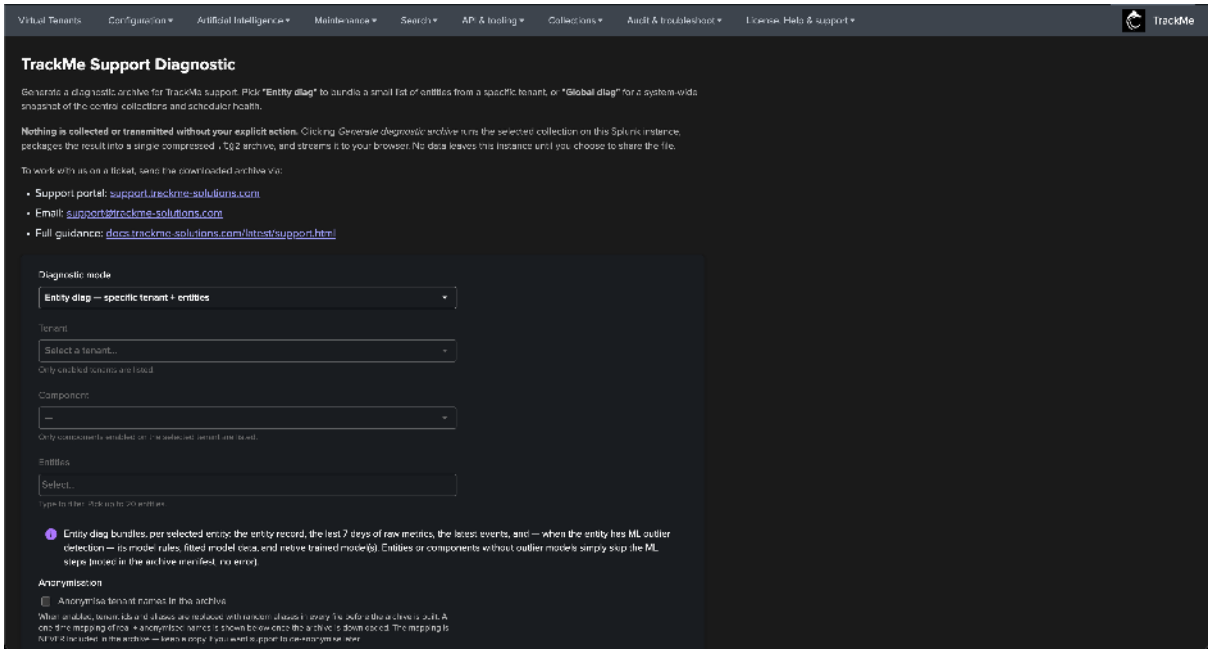
example:

```
2023-06-06 09:16:29,916 INFO trackmeconfmanager.py generate 156 verifying␣
↪transaction status, transaction_md5="ca315b5f58e2766e60bf231e86a5f551"
2023-06-06 09:16:29,916 INFO trackmeconfmanager.py generate 162 transaction was␣
↪not found in the KVstore and will be processed in live mode
2023-06-06 09:16:41,030 INFO trackmeconfmanager.py generate 219 transaction was␣
↪registered successfully to the KVstore with key="ca315b5f58e2766e60bf231e86a5f551
↪"
```

8.10.13 Support diagnostics

When you raise a support case, TrackMe can generate a **diagnostic archive** — a `.tgz` bundle of the configuration and state TrackMe Support needs to investigate — instead of you hand-collecting KV-store records, metrics, and logs. It is available from the TrackMe UI and the REST API, and requires the power capability (`trackmepoweroperations`).

The UI is reached from **API & tooling → Support - generate diags**. You pick the diagnostic mode, scope it (tenant / component / entities for an entity diag), optionally enable anonymisation, then **Generate diagnostic archive** — nothing is collected or transmitted until you click it, and the archive is streamed to your browser so no data leaves the instance until you choose to share the file:



What goes in the bundle

You generate the archive at one of two scopes:

- **Entity-scoped** — a specific tenant and component, plus up to **20 entities**. The bundle collects those entities' KV records, their recent metrics (last 7 days), and their latest summary and notable events — and, since **2.4.3**, their **ML outlier models** (the outlier rules, the fitted data, and the native ML models).
- **Global** — the central (cross-tenant) KV collections, plus the last 24 hours of scheduler and runtime metrics.

Optionally, **anonymise tenants**: real tenant identifiers and aliases are replaced with anonymised tokens in both filenames and content, and the mapping is returned to you so you can still correlate. Use this when the environment is sensitive.

How it runs

Generation is **asynchronous** — you start a job, poll its status, and download the archive once it completes. Through the REST API:

```
# start a job — returns a job_id
| trackme mode=post url="/services/trackme/v2/support_diag/generate" body="{ ... }"

# poll the job (queued / running / complete / error / cancelled)
| trackme mode=get url="/services/trackme/v2/support_diag/status" body="{'job_id':
↪'<id>'}"

# cancel a running job
| trackme mode=delete url="/services/trackme/v2/support_diag/cancel" body="{'job_id
↪': '<id>'}"
```

When the job is **complete**, download the archive with the one-time token returned by the status call. Jobs run a few at a time, time out after about an hour, and their records are cleaned up shortly after completion. The exact request parameters (scope, entity list, anonymisation) are documented in the *REST API Reference* UI.

Tip

Attach the generated `.tgz` to your support case. If your data is sensitive, enable tenant anonymisation when you generate it.

See also

- *REST API reference* — the REST API the diagnostics endpoints belong to.
- *Roles & access control* — the power capability the action requires.

8.10.14 REST API reference

Everything you can do in the TrackMe UI, you can do through its REST API — the UI itself is built entirely on it. The API is versioned under `/trackme/v2/*` and is fully self-documenting.

How the API is organised

Endpoints are grouped into **resource groups** by feature area — tenancy (`vtenants`, `virtual_groups`, `configuration`), the seven components (`splk_dsm` … `splk_wlk`, `splk_vol`), alerting (`alerting`, `ack`), policies, data and ML, operations (`maintenance`, `backup_and_restore`, `audit` …), the dedicated findings tenants (`uam`, `cribl` for Cribl Stream, Cribl Edge and the Splunk HEC, S2S, UF Fleet and Forwarder Collectors tenants, `findings_actions`, `ai_findings_agent`), and integrations (`ai`, `splk_soar`, `licensing`).

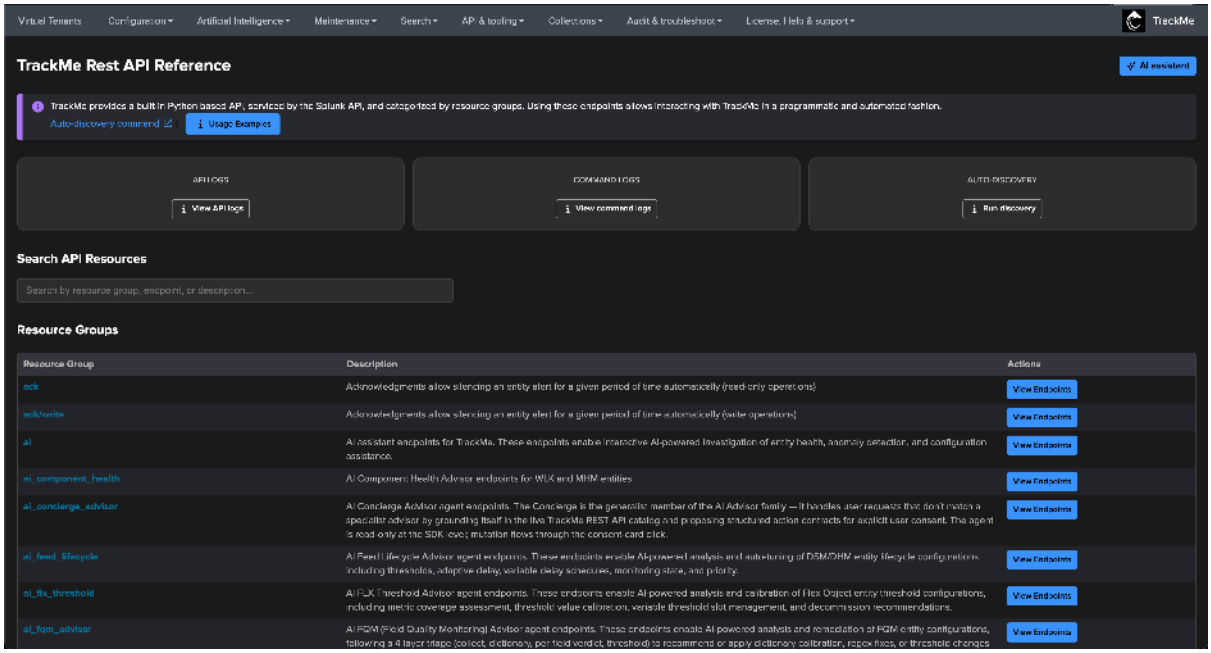
Access follows the same three-tier model as the rest of TrackMe, enforced by Splunk capability (see *Roles & access control*):

Scope	Capability	Reaches
user	`trackmeuseroperations`	Read endpoints (list / get / search).
power	`trackmepoweroperations`	`*/write` — manage entities (ack, set priority…).
admin	`trackmeadminoperations`	`*/admin` — create tenants, policies, trackers; migrations.

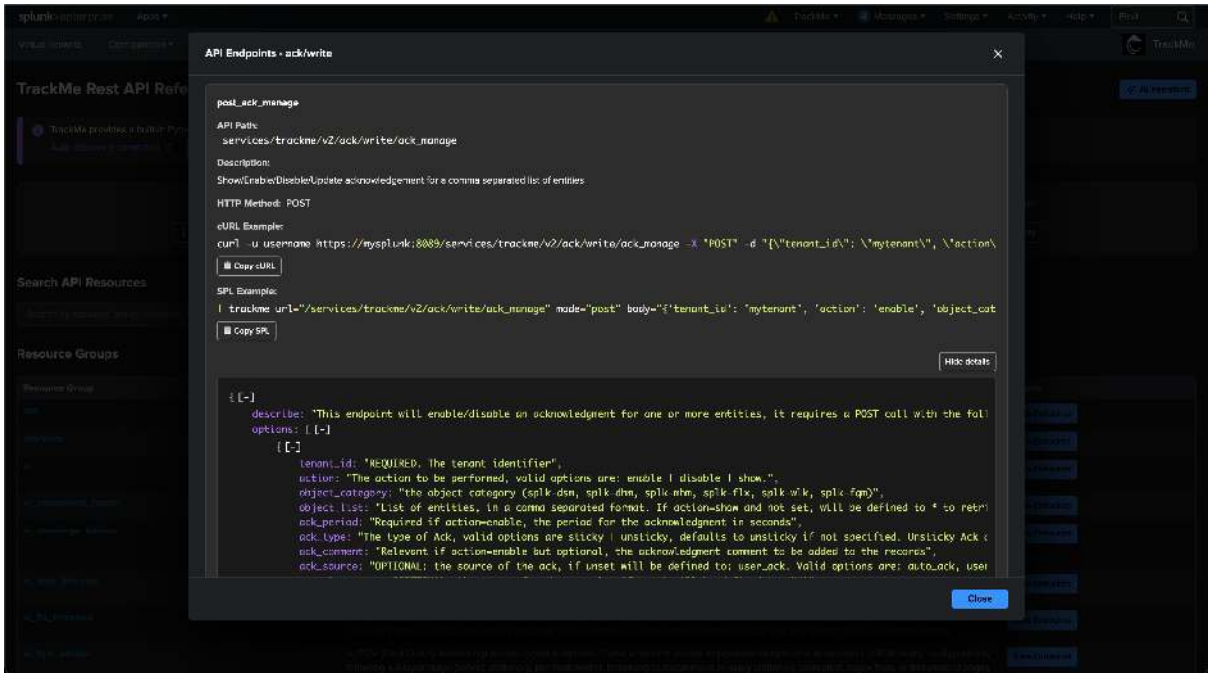
Every endpoint requires authentication; there are no anonymous endpoints. All activity is logged, and any change-making call writes audit events (see *Events & sourcetypes*).

Finding the docs

The API documents itself. In the UI, open **API & Tooling → TrackMe REST API Reference**. The page lists every resource group with its description, a live search box, and quick links to the API and command logs and the auto-discovery command:



Click **View Endpoints** on any resource group to see each endpoint's API path, description, HTTP method, and ready-to-copy **cURL** and **SPL** examples, plus the full `describe` schema of its options (required and optional parameters):



From SPL, `| trackmeapiautodocs target="groups"` lists the resource groups, and `| trackmeapi-autodocs target="endpoints"` lists every endpoint with its URI, HTTP mode, options, and an example. You can call the API from inside Splunk with the `| trackme url=... mode=...` command, or externally with `curl` or any HTTP client.

Authenticating from outside Splunk

External callers authenticate to splunkd like any Splunk REST client. The preferred method is a **bearer token** (`-H "Authorization: Bearer <token>"`); **basic auth** (`-u user:password`) also works, as does exchanging credentials at `/services/auth/login` for a session key used as `-H "Authorization: Splunk <token>"`.

See Splunk's documentation:

- https://docs.splunk.com/Documentation/Splunk/latest/Security/UseAuthTokens

- https://docs.splunk.com/Documentation/Splunk/latest/RESTUM/RESTusing#Authentication_and_authorization

Important

The `| trackme` SPL command accepts a JSON body with **single or double quotes**, but a REST call from **outside** Splunk must send valid JSON with **double quotes** — a common cause of "bad request" errors when porting an SPL example to `curl`.

See also

- *Roles & access control* — the capability model the API enforces.
- *Consuming TrackMe data from your own dashboards* — reading status counts and entity records from your own dashboards or external tools.
- *Configuration Manager (CI/CD)* — replaying API calls as config-as-code.
- *Events & sourcetypes* — the API and audit logs.

8.10.15 Consuming TrackMe data from your own dashboards

A frequent question from teams that run TrackMe: *"I want to show TrackMe's entity status counts — total, red, orange, green — on my own department dashboard, or pull the entity table into another system. How do I get at that data?"*

Everything the TrackMe UI displays comes from its REST API, and TrackMe ships SPL wrappers around that API. This page walks through the options, from the quickest way to get the headline counts to bulk access to every entity record, and the sharing-level prerequisite you need in place first when your dashboard lives outside the TrackMe app.

Prerequisite: sharing level and permissions

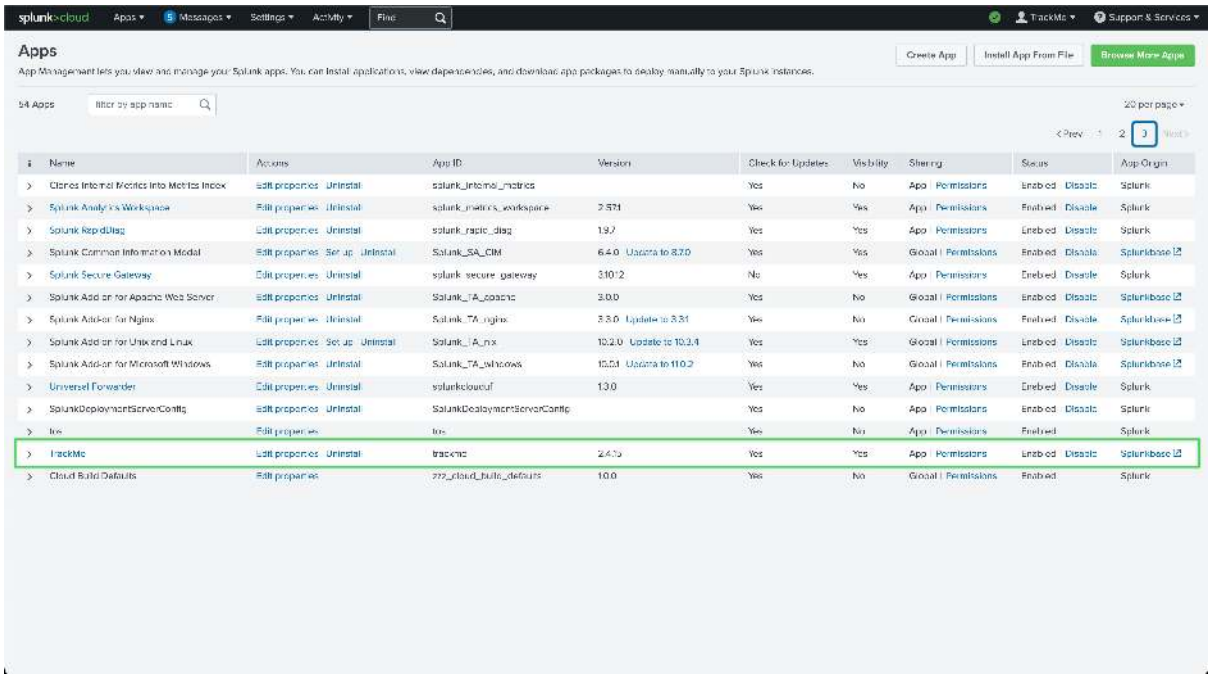
TrackMe's custom commands (`trackmegetcoll`, `trackme` …) and the per-tenant KV-store collections and lookup definitions it creates are Splunk knowledge objects. By default, TrackMe shares them at the **application** level only: a search that runs in the context of another app (a dashboard in your own app, for instance) cannot see them, and the command reports as unknown even though the user is otherwise entitled to use it.

Important

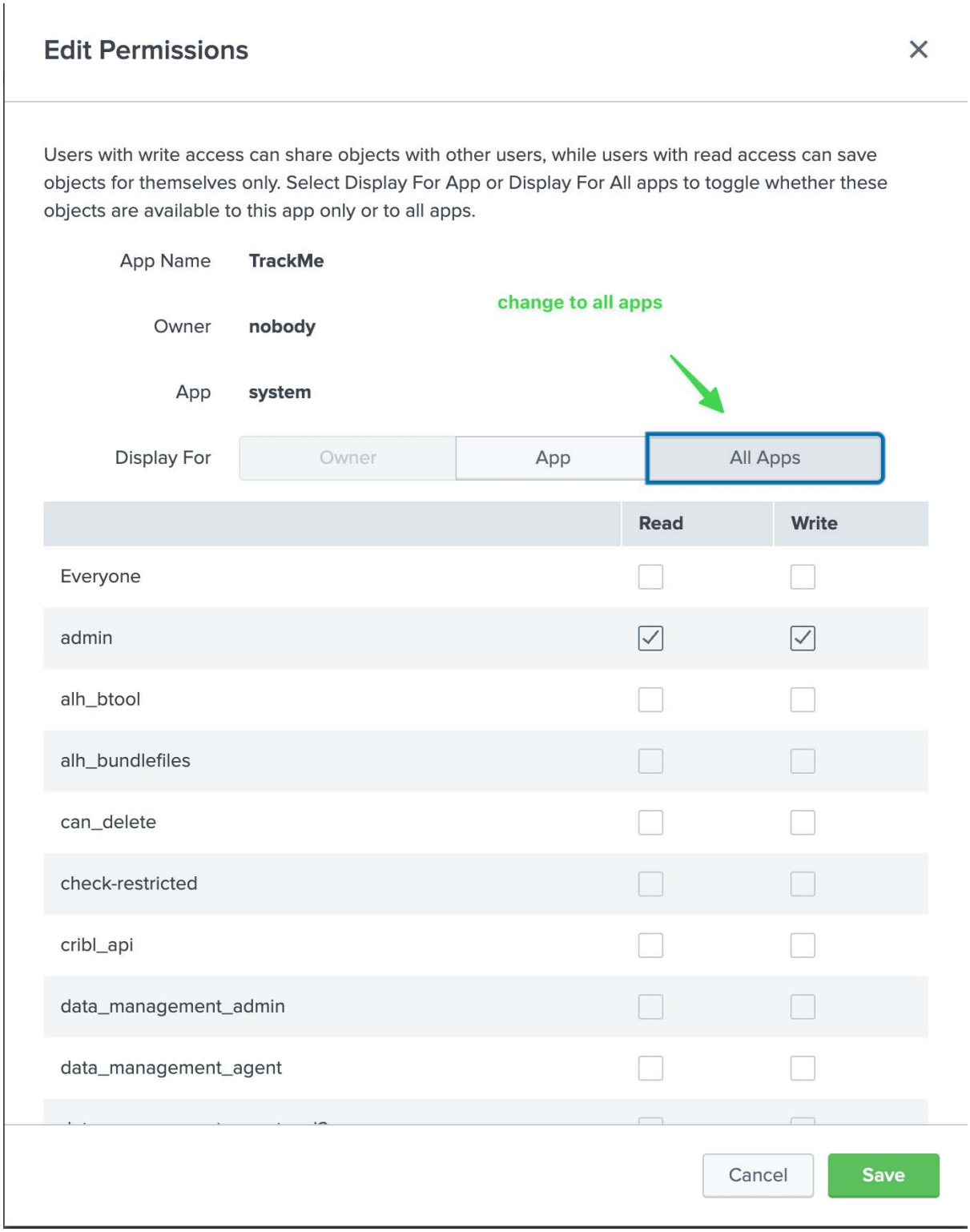
Granting permissions on the individual command is not enough. The visibility of the objects is governed by the **sharing level** of the TrackMe application, and by the sharing level TrackMe applies to the knowledge objects it creates for each tenant.

Step 1 — share the TrackMe application globally

In Splunk Web, open **Apps → Manage Apps**, locate **TrackMe** and click **Permissions**:

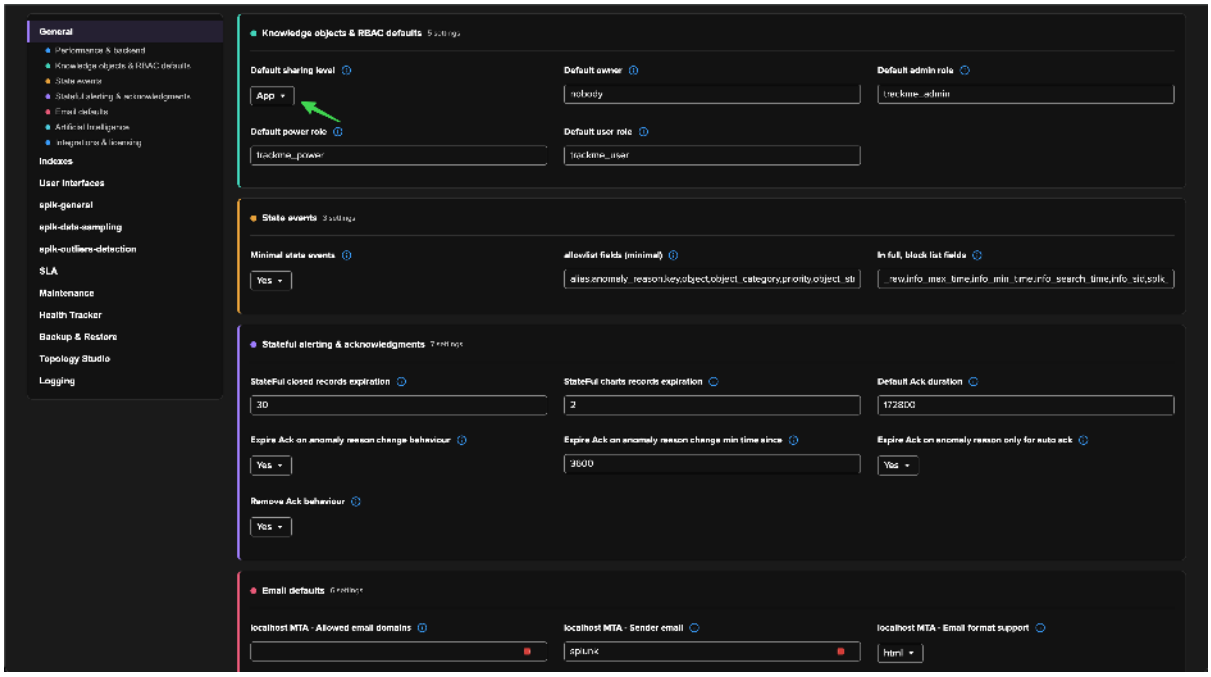


Set **Display For** to **All Apps** and save:



Step 2 — set TrackMe's default sharing level to global

Sharing the app exports the objects that exist today. TrackMe also creates knowledge objects on an ongoing basis (a new tenant creates its own collections, lookup definitions and scheduled searches), so tell it to create future objects at the global level too. Open **Configuration → System settings → General**, then in the **Knowledge objects & RBAC defaults** group set **Default sharing level** to **Global**:



Step 3 — check the user's access

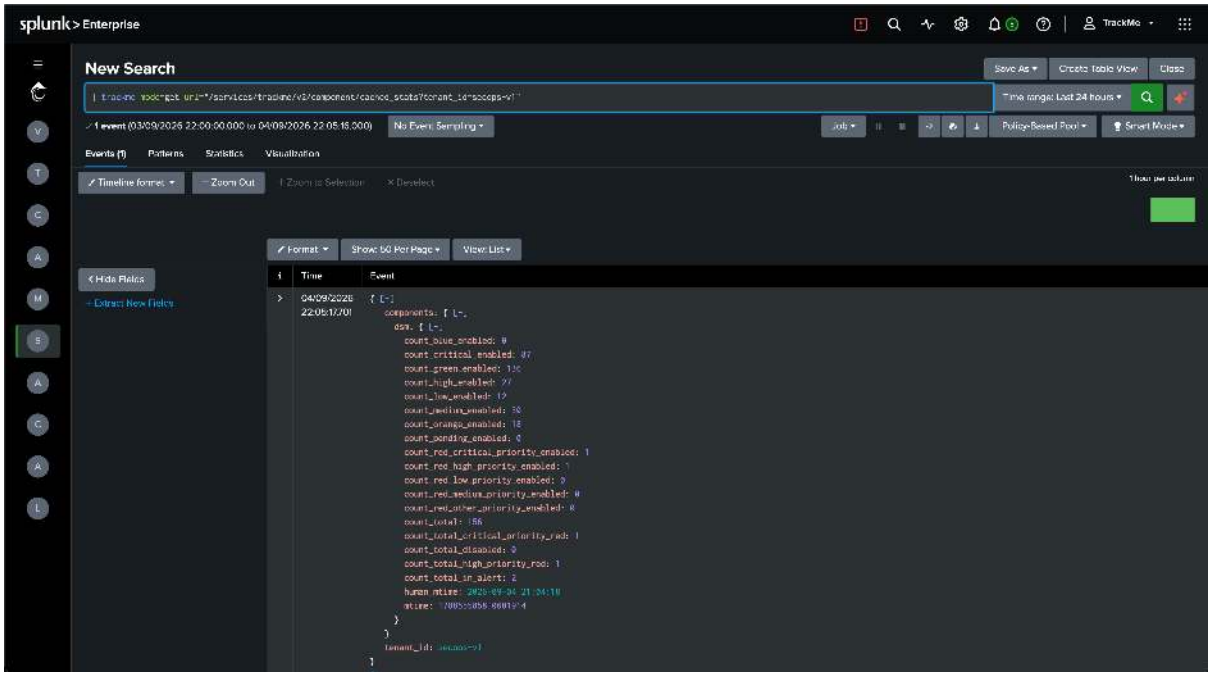
Whichever access path you use, the user running the search needs the TrackMe user capability (`trackmeuseroperations`, typically through the `trackme_user` role) **and** must be a member of one of the roles granted on the Virtual Tenant. Tenant RBAC is enforced by the API on every read, so a user who cannot open the tenant in the UI gets no data from SPL or REST either. See *Roles & access control*.

Option 1 — the headline counts, without any SPL processing

If all you need are the summary figures shown at the top of a tenant — total entities, and how many are red, orange, green, or in alert by priority — do not recompute them yourself. The decision maker precomputes these counts for every component of every tenant, and the `cached_stats` endpoint returns them in a single, very cheap read:

```
| trackme mode=get␣
↪url="/services/trackme/v2/component/cached_stats?tenant_id=<tenant_id>"
```

Add `&component=dsm` (or `dhm`, `mhm`, `flx`, `fqm`, `wlk`) to restrict the response to one component. The result is a JSON document with one block per component:



The fields you will most often use:

Field	Meaning
`count_total`	All entities of the component, including disabled ones.
`count_total_disabled`	Entities whose monitoring is disabled.
`count_green_enabled` / `count_orange_enabled` / `count_blue_enabled`	Enabled entities currently in that state.
`count_total_in_alert`	Enabled entities currently **red**, regardless of priority.
`count_red_critical_priority_enabl` / `count_red_high_priority_enabled` / `count_red_medium_priority_enable` / `count_red_low_priority_enabled`	Red entities broken down by priority.
`count_critical_enabled` / `count_high_enabled` / `count_medium_enabled` / `count_low_enabled`	Enabled entities broken down by priority, whatever their state.
`mtime` / `human_mtime`	When the counts were last refreshed.

To reproduce the tiles from the tenant home in your own dashboard, flatten the JSON with `spath`:

```
| trackme mode=get␣
↪url="/services/trackme/v2/component/cached_stats?tenant_id=<tenant_id>"
| spath path=components.dsm output=dsm
| spath input=dsm
| eval entities = count_total - count_total_disabled,
       red = count_total_in_alert,
       orange = count_orange_enabled,
       green = count_green_enabled
| table entities, red, orange, green
```

The same counts are also available from the `trackmegetcoll` command with `mode=cachedstats`, which is what TrackMe's own `tenant_get_info` macro uses:

```
| trackmegetcoll tenant_id=<tenant_id> component=dsm mode=cachedstats
```

Note

Both forms return the cached figures. If the cache is older than about 15 minutes for a component, `trackmegetcoll mode=cachedstats` transparently recomputes the counts from the live entity records instead; `mode=stats` always recomputes.

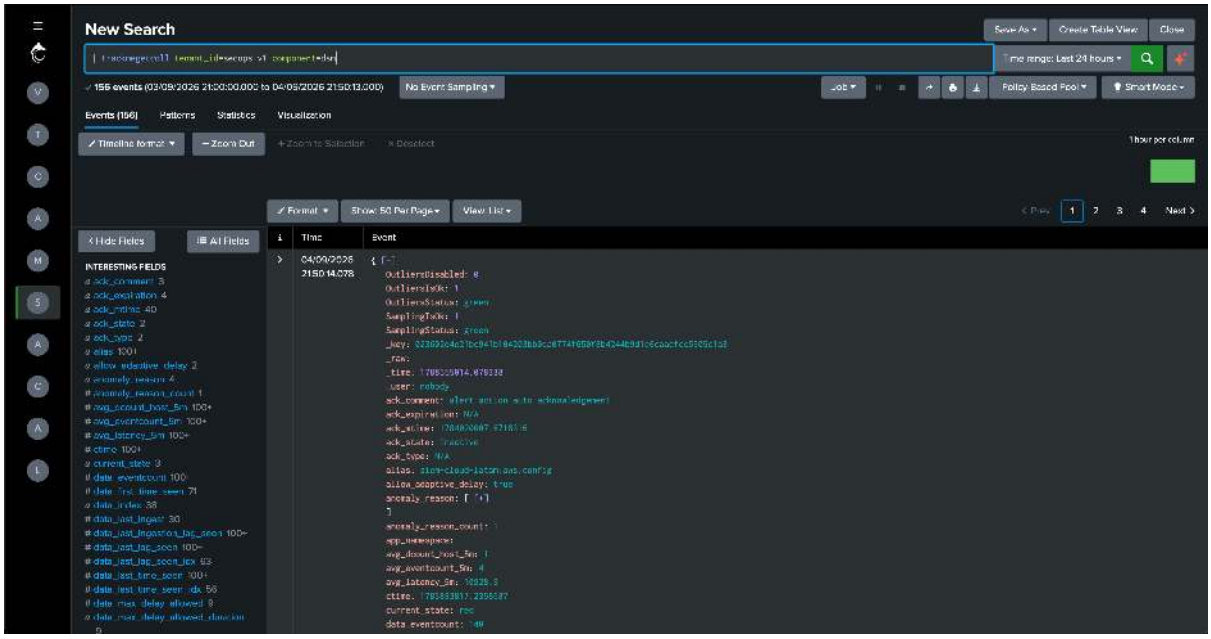
From outside Splunk, the same endpoint answers a plain HTTP GET:

```
curl -k -H "Authorization: Bearer <token>" \

↪"https://<splunk>:8089/services/trackme/v2/component/cached_stats?tenant_id=<tenant_id>"
```

Option 2 — the entity records in SPL with `trackmegetcoll`

To work with the entities themselves — build your own table, filter on a field, count by anything — use the `trackmegetcoll` generating command. It is the exact search TrackMe runs behind the **Search table** button of a component tab:

```
| trackmegetcoll tenant_id=<tenant_id> component=<component>
```



Each result is one entity with every field the UI shows: `object`, `alias`, `object_state`, `priority`, `anomaly_reason` (as a multivalue field), `tags`, `monitored_state`, the latency and delay metrics, the status message, and so on. The records are **processed by the decision maker in real time** — states, refreshed dates and enrichments are computed on the fly — so what you get is exactly what the UI would show at that moment.

Options of the command:

Option	Description
`tenant_id`	Required. The tenant identifier.
`component`	Required. One of `dsm`, `dhm`, `mhm`, `flx`, `fqm`, `wlk`.
`mode`	`records` (default) returns the entities; `stats` returns the summary counts, recomputed from the records; `cachedstats` returns the precomputed counts.
`filter_key` / `filter_object`	Return a single entity, by its key or by its `object` value. Mutually exclusive.
`mode_view`	`minimal`, `compact` or `full` field projection, when applicable.

Then use any SPL you like. For instance, the same headline counts as Option 1, computed from the live records:

```
| trackmegetcoll tenant_id=<tenant_id> component=dsm
| where monitored_state="enabled"
| stats count as entities,
        count(eval(object_state="red")) as red,
        count(eval(object_state="orange")) as orange,
        count(eval(object_state="green")) as green
```

Or a list of what is currently in alert, with the reason:

```
| trackmegetcoll tenant_id=<tenant_id> component=dsm
| where object_state="red" AND monitored_state="enabled"
| table object, priority, anomaly_reason, status_message
```

Tip

Because the command is a wrapper around the `load_component_data` API endpoint, the results honour tenant RBAC and reflect the live state, at the cost of one decision-maker run per search. On large tenants the command automatically serves the records from the shadow cache described below, so this stays fast.

Option 3 — reading the KV-store collections directly

Important

Recommendation: read the shadow collection

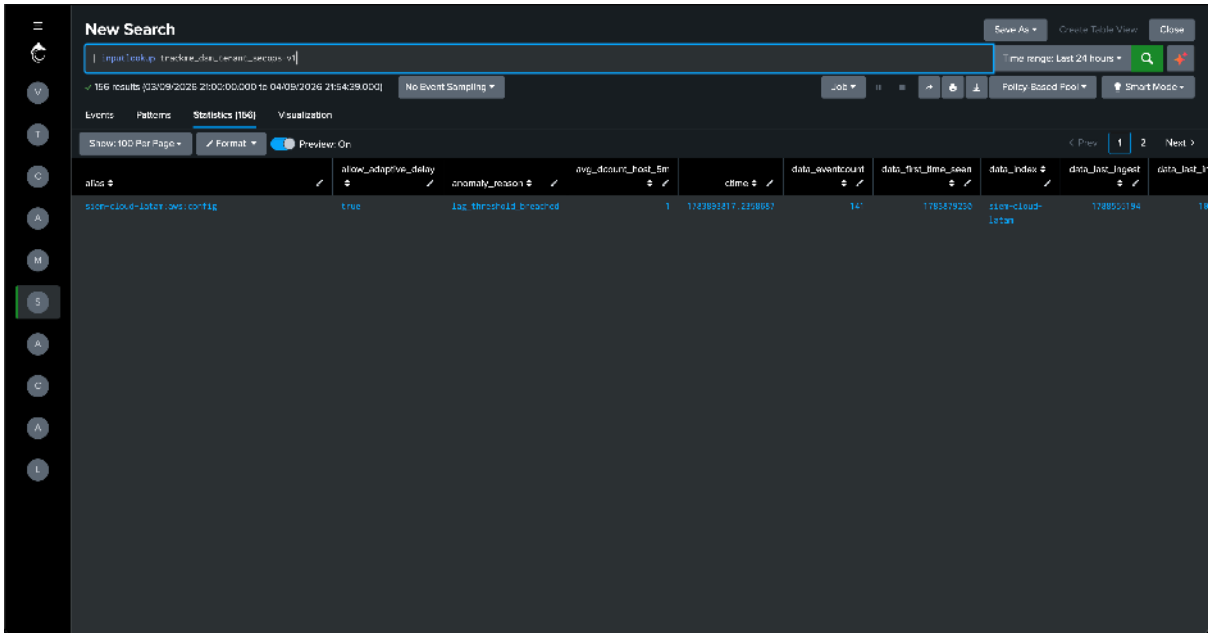
If you want to consume TrackMe data through the KV-store in Splunk, use the **shadow** collection, `trackme_<component>_shadow_tenant_<tenant_id>`, described just below. It holds the entities *after* processing by the decision maker, is near real time, and is what the TrackMe UI itself reads. The raw per-tenant collection is documented first only so you know what it is and why it is not the right source for entity status.

TrackMe stores the core data of each component in a per-tenant KV-store collection, exposed through a lookup definition named:

```
trackme_<component>_tenant_<tenant_id>
```

You can read it with `inputlookup` like any other collection:

```
| inputlookup trackme_<component>_tenant_<tenant_id>
```



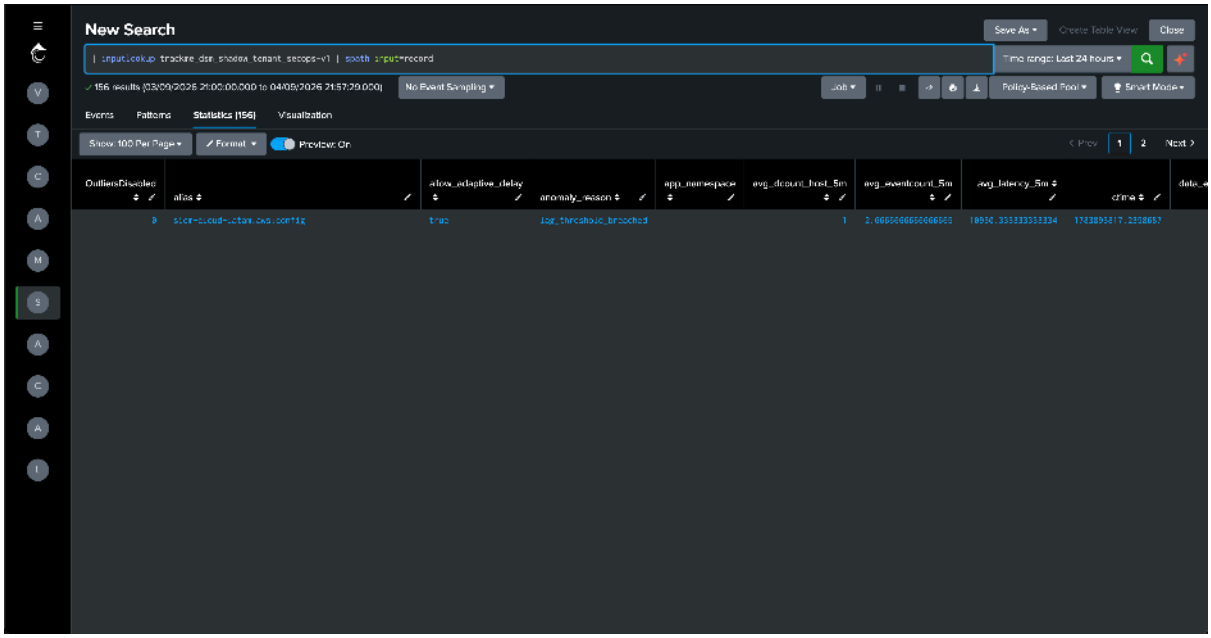
Warning

The raw collection is the **input** of the decision maker, not its output. It is not refreshed in real time and does not carry everything the UI shows or that TrackMe acts upon (the current state after all rules, the status message, alerting decisions …). Use it for bulk exports of the stored attributes; do not use it as the source of truth for entity status.

The shadow records: the recommended KV-store source

This is the collection to use for KV-store access to the *processed* data: the **shadow** collection. Since TrackMe 2.4.3, the decision maker writes a post-processing snapshot of every entity to a second collection, which the UI and TrackMe's backends read themselves. Each row holds the enriched entity as a JSON document in the `record` field, so expand it with `spath`:

```
| inputlookup trackme_<component>_shadow_tenant_<tenant_id> | spath input=record
```



The shadow record is the same document that `trackmegetcoll` returns for the entity, but reading it costs a KV-store scan rather than a decision-maker run. It is refreshed on every tracker cycle and patched immediately when an entity is edited (priority, tags, acknowledgement …), which makes it near real time. Shadow records are enabled by default for every tenant; a tenant can opt out through its `shadow_enabled` setting, in which case the collection is empty. You can confirm a shadow copy is available with:

```
| trackme mode=get␣
↪url="/services/trackme/v2/component/check_shadow?tenant_id=<tenant_id>&component=dsm"
```

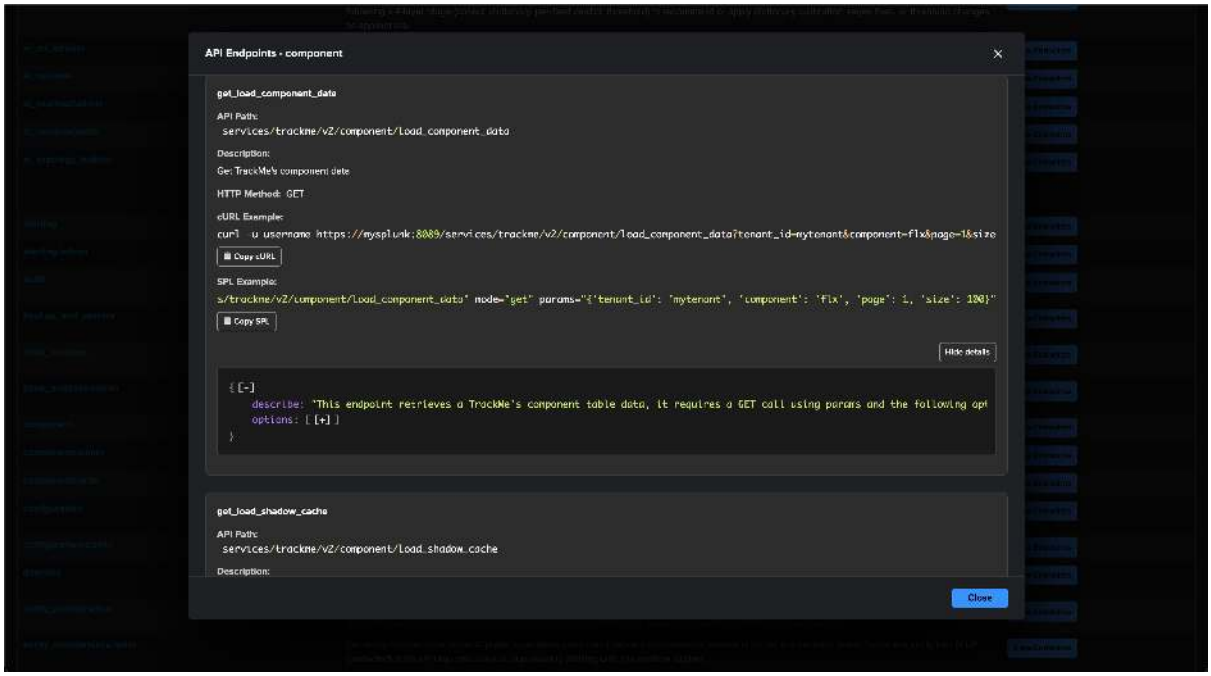
Tip

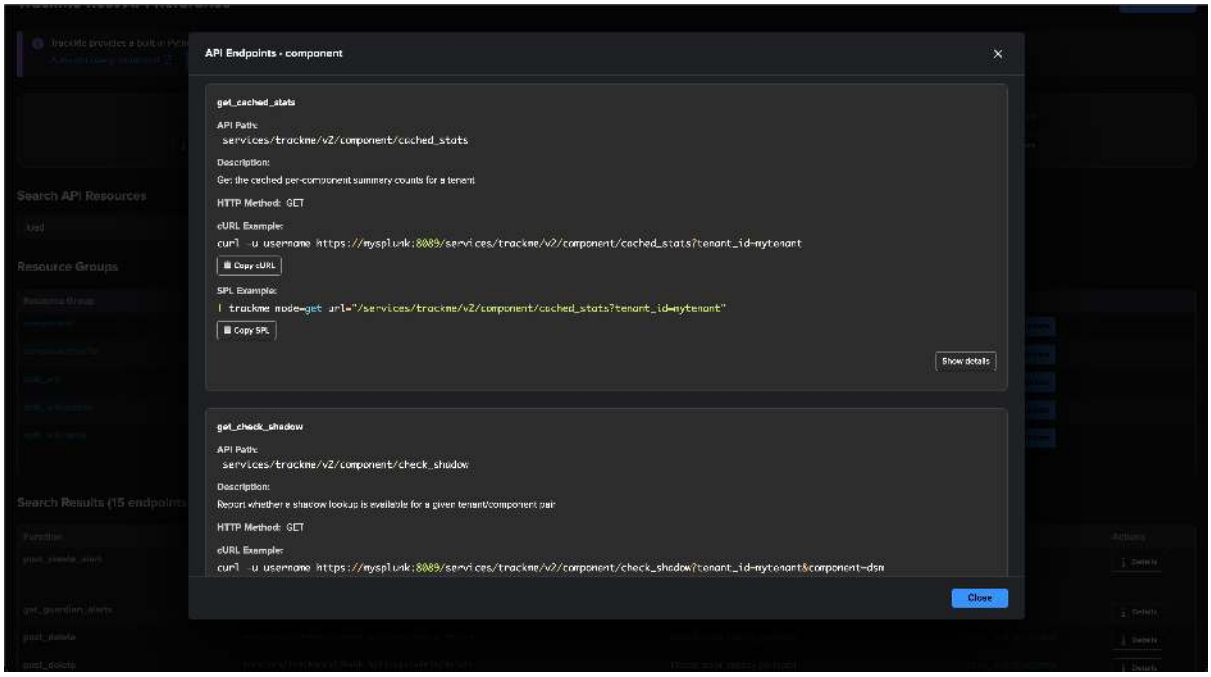
In the support case that prompted this page, the customer's own dashboard ended up reading the shadow collection: pre-processed, cheap to query, and current enough for a status board. This is our recommendation whenever a Splunk dashboard or search consumes TrackMe entities through the KV-store.

Option 4 — from outside Splunk, through the REST API

When the consumer is not a Splunk search at all — an external portal, a script, another monitoring platform — call the REST endpoints directly. All of them are documented in the UI under **API & Tooling → TrackMe REST API Reference**, with ready-to-copy cURL and SPL examples (see *REST API reference*). The endpoints of the `component` resource group are the ones to use:

Endpoint (GET)	Purpose
`/services/trackme/v2/component/cached_stats`	The precomputed per-component counts of a tenant (Option 1).
`/services/trackme/v2/component/load_component_data`	The entity records, processed live by the decision maker, paginated with `page` and `size` (`size=0` returns everything). This is what `trackmegetcoll` calls.
`/services/trackme/v2/component/load_shadow_cache`	The entity records from the shadow cache, without a decision-maker run.
`/services/trackme/v2/component/check_shadow`	Whether a shadow copy is available for a tenant and component.





For example, to offload every DSM entity of a tenant:

```
curl -k -H "Authorization: Bearer <token>" \

↪"https://<splunk>:8089/services/trackme/v2/component/load_component_data?tenant_id=<tenant_id>&
```

Authentication is the standard splunkd authentication (bearer token, basic auth or session key), and the same TrackMe capabilities and tenant RBAC apply as in the UI.

Which one should I use?

You want...	From SPL	From outside Splunk
The status counts of a tenant	`| trackme mode=get url="../component/cached_stats?tenant_id=..."`	GET `/component/cached_stats`
The live entity table, exactly as the UI	`| trackmegetcoll tenant_id=... component=...`	GET `/component/load_component_data`
The processed entity table through the KV-store (**recommended** for KV-store access)	`| inputlookup trackme_<component>_shadow_ | spath input=record`	GET `/component/load_shadow_cache`
The stored attributes only, bulk export (not for entity status)	`| inputlookup trackme_<component>_tenant_`	Splunk's KV-store REST API

See also

- *REST API reference* — the REST API reference and how to authenticate from outside Splunk.
- *Roles & access control* — the capability and tenant-role model that gates every read.
- *Configuration — in depth* — the system settings, including the shadow filesystem cache.

8.10.16 Events & sourcetypes

TrackMe produces its own data — events, metrics, and logs — as it runs. Knowing what it writes, and where, is the key to troubleshooting and to building your own reports on top of TrackMe.

Indexes

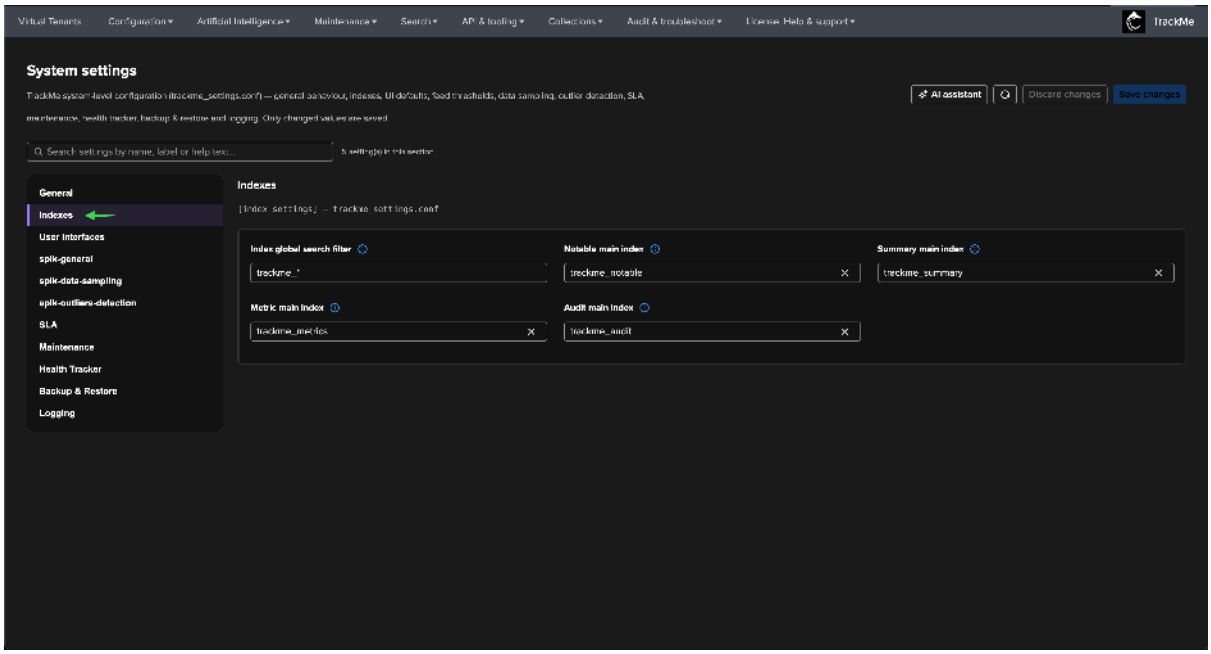
Each tenant writes to four configurable index roles (the defaults are shown):

Role	Default	Holds
Summary	`trackme_summary`	Per-entity summary events.
Audit	`trackme_audit`	Audit events (including Configuration Guardian).
Metrics	`trackme_metrics`	The metric store (**must** be a `metric` index).
Notable	`trackme_notable`	Notable events.

Because indexes are configurable per tenant, do not assume the default names when troubleshooting — check the tenant's index settings.

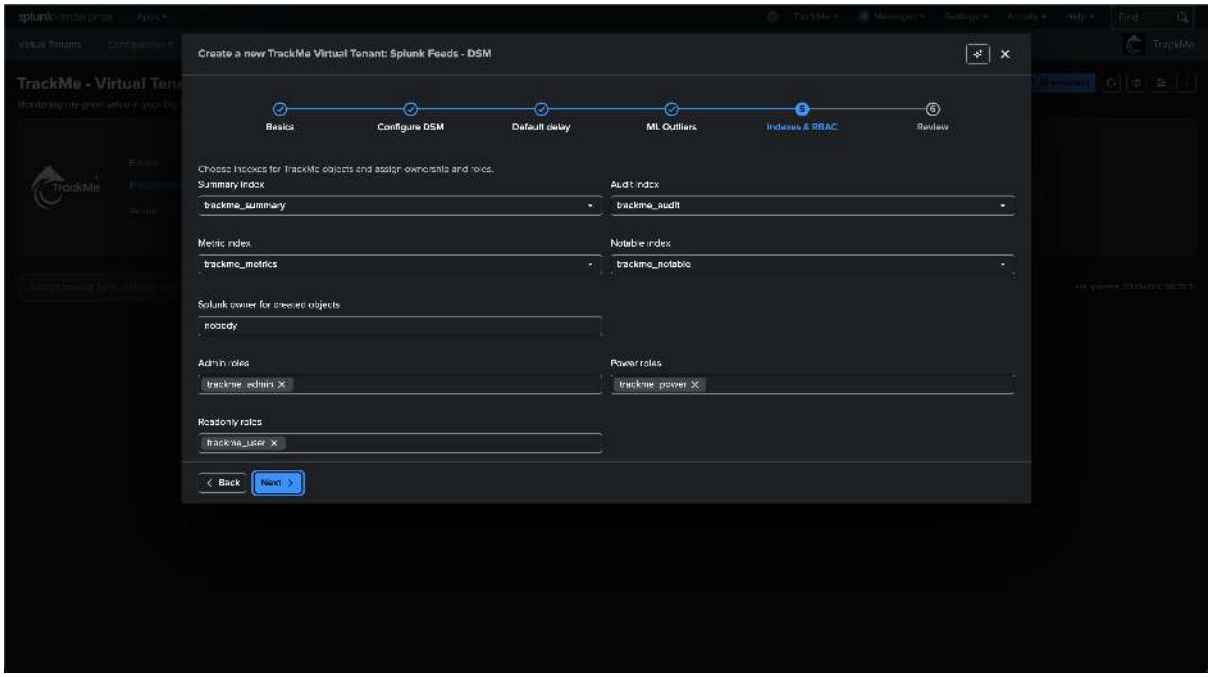
Setting the default indexes

The deployment-wide defaults — the four index roles plus the **index global search filter** (`trackme_*` by default, used to scope every TrackMe search) — are set under **Configuration → System settings → Indexes**. New tenants inherit these values unless you override them at creation:



Custom indexes per tenant

You can point a tenant at its own indexes when you create it — the **Indexes & RBAC** step of the tenant-creation wizard pre-fills the system defaults, and you can change any of the four roles (for example, to isolate a sensitive tenant into a dedicated summary/audit index). The metric role **must** be a `metric` index:



Tip

If you create tenant-specific indexes, use a strict prefix convention such as `trackme_<context>_<index name>` so the `trackme_*` global search filter keeps matching them.

Event sourcetypes

Events are JSON. The main sourcetypes:

Sourcetype	What it records
`trackme:audit`	Changes made to entities and configuration.
`trackme:flip`	Entity state transitions (see *Status flapping*).
`trackme:state`	Entity state snapshots.
`trackme:health`	Component and tracker health.
`trackme:sla_breaches`	SLA breach events.
`trackme:stateful_alerts`	Stateful incidents opened / updated / closed.
`trackme:notable`	Notable events from the notable alert action.
`trackme:smart_status`	SmartStatus automated-investigation results.
`trackme:handler`	TrackMe Handler events — traces backend logic execution such as hybrid trackers and the delayed entities inspector.

To enumerate the TrackMe sourcetypes present in your environment:

```
| tstats count where index=trackme_* by sourcetype
```

Metrics and logs

- **Metrics** follow the `trackme.*` convention in the metrics index — the SLA state (`trackme.sla.object_state`), the feed KPIs (`trackme.splk.feeds.*`), workload counters (`trackme.splk.wlk.*`), operational health (`trackme.components_register.*`), and your own Flex / FQM metrics. The full catalogue, by component, is in the *Metrics reference*.
- **Logs** land in `_internal`: the REST API as `trackme:rest_api` and custom commands as `trackme:custom_commands:*`. Their verbosity is the logging level set in the *Configuration* page.

See also

- *Configuration* — index and logging settings.
- *SLA policies* — the SLA metrics and breach events.
- *Alerting* — the alerting events.

8.10.17 Metrics reference

TrackMe writes numeric KPIs to its **metrics index** (default `trackme_metrics`), all under the `trackme.*` namespace. There are two kinds:

- **Metrics TrackMe controls** — a fixed catalogue it emits for every entity and tracker: the SLA state, the feed KPIs (delay, latency, volume…), workload counters, and operational health. These are documented below.
- **Metrics you define** — for *Flex* and *FQM*, the metric names come from *your* tracker's SPL, so there is no fixed list (see *below*).

Tip

The authoritative list for *your* deployment is always live in Splunk. List every metric name with:

```
| mcatalog values(metric_name) WHERE index=trackme_metrics
```

The tables below name the metrics as of current TrackMe; treat `mcatalog` as the source of truth if a name differs in your version.

Common to all components

One metric is emitted for **every** entity, regardless of component — it is the backbone of SLA reporting:

Metric	Meaning	Unit
`trackme.sla.object_state`	The entity's health state expressed as a number (the green / orange / red / blue state, including ack), one sample per entity per cycle. This is what SLA dashboards integrate over time.	state code

Operational / system-level

These describe how TrackMe *itself* is running — the health and runtime of its trackers, and the scoring pipeline. They are per tracker / per entity rather than per data feed.

Metric	Meaning	Unit
trackme.components_register.runtime	Execution run time of a tracker / report — how long a hybrid tracker (or any component tracker) took on its last run.	seconds
trackme.components_register.status	Last execution status of a tracker — 1 success, 2 failure, 3 other.	status code
trackme.scoring.score	The computed score sample for an entity.	score
trackme.impact_score.*	Durable record of impact-score adjustments (false-positive and manual increase / decrease). Leaves are the numeric keys of the adjustment record.	score

The components_register metrics are the ones to watch for **tracker performance** — for example, average hybrid-tracker runtime per tenant — and feed the Performance Deepdive dashboards.

Feed KPIs — DSM, DHM, MHM

The data-flow components share a single family, trackme.splk.feeds.*, emitted by their hybrid trackers. These are the delay, latency, and volume KPIs behind the entities table and the entity overview charts. The object_category dimension (splk-dsm / splk-dhm / splk-mhm) tells you which component a sample belongs to.

Metric	Meaning	Unit
trackme.splk.feeds.lag_event_sec	**Delay** — seconds since the most recent event (now − last event time).	seconds
trackme.splk.feeds.lag_ingestion_sec	**Latency** — seconds between an event's time and its index time.	seconds
trackme.splk.feeds.eventcount_4h	Event count over the tracker window.	count
trackme.splk.feeds.hostcount_4h	Distinct host count over the tracker window.	count
trackme.splk.feeds.avg_eventcount_5m	Average events per 5-minute bucket.	count
trackme.splk.feeds.latest_eventcount_5m	Latest 5-minute event count.	count
trackme.splk.feeds.perc95_eventcount_5m	95th-percentile 5-minute event count.	count
trackme.splk.feeds.stdev_eventcount_5m	Standard deviation of the 5-minute event count.	count
trackme.splk.feeds.avg_latency_5m	Average ingestion latency per 5 minutes.	seconds
trackme.splk.feeds.latest_latency_5m	Latest 5-minute latency.	seconds
trackme.splk.feeds.perc95_latency_5m	95th-percentile 5-minute latency.	seconds
trackme.splk.feeds.stdev_latency_5m	Standard deviation of the 5-minute latency.	seconds
trackme.splk.feeds.avg_dcount_host_5m	Average distinct host count per 5 minutes.	count
trackme.splk.feeds.latest_dcount_host_5m	Latest 5-minute distinct host count.	count
trackme.splk.feeds.perc95_dcount_host_5m	95th-percentile 5-minute distinct host count.	count
trackme.splk.feeds.stdev_dcount_host_5m	Standard deviation of the 5-minute distinct host count.	count
trackme.splk.feeds.global_dcount_host	Global distinct host count for the entity.	count

Note

Not every component emits every leaf. **DSM** emits the full set above. **DHM** emits a subset — the event-count, latency, `eventcount_4h`, and the two lag metrics — but **not** the distinct-host-count metrics. **MHM** rides the same family for the lag / availability metrics (notably `lag_event_sec`).

Component-specific metrics

On top of the shared feed KPIs, each data component emits a small set under its own root (note the **underscore** — `trackme.splk_dsm` etc., distinct from the dotted `trackme.splk.feeds`):

Metric	Meaning	Unit
`trackme.splk_dsm.sampling.model_pct_match`	DSM data sampling — % of sampled events matching a format model (per `model_name`).	percent
`trackme.splk_dsm.sampling.model_count_matched`	DSM data sampling — events matched by the model.	count
`trackme.splk_dsm.sampling.model_count_parsed`	DSM data sampling — events parsed.	count
`trackme.splk_dsm.sampling.run_time`	DSM data-sampling run time.	seconds
`trackme.splk_dhm.last_eventcount`	DHM — last event count for the host, per index / sourcetype.	count
`trackme.splk_dhm.last_ingest_lag`	DHM — last ingestion lag.	seconds
`trackme.splk_dhm.last_event_lag`	DHM — last event lag (delay).	seconds
`trackme.splk_mhm.last_metric_lag`	MHM — last lag of the monitored metric series, per `metric_category`.	seconds

Workload (WLK)

Workload monitoring emits a fixed set of scheduled-search execution KPIs under `trackme.splk.wlk.*` (`object_category=splk-wlk`):

Metric	Meaning	Unit
`trackme.splk.wlk.count_execution`	Scheduled-search executions.	count
`trackme.splk.wlk.count_completed`	Executions that completed.	count
`trackme.splk.wlk.count_skipped`	Skipped executions.	count
`trackme.splk.wlk.count_errors`	Errored executions.	count
`trackme.splk.wlk.elapsed`	Search run time.	seconds
`trackme.splk.wlk.scan_count`	Events scanned.	count
`trackme.splk.wlk.pct_cpu`	CPU usage.	percent
`trackme.splk.wlk.pct_memory`	Memory usage.	percent
`trackme.splk.wlk.svc_usage`	Splunk Cloud SVC consumption.	SVC
`trackme.splk.wlk.count_ess_notable`	ES notable events produced (where applicable).	count
`trackme.splk.wlk.status`	Last status of the scheduled search.	status code

Volume Outliers (VOL)

Volume Outliers emits the licensed volume of every index under `trackme.splk.vol.*` (`object_category=splk-vol`; `object` is the index name). The points carry explicit bucket timestamps written by the collector, so the series is regular whatever the tracker's own run time:

Metric	Meaning	Unit
`trackme.splk.vol.volume_5m_mb`	Licensed volume of the 5-minute bucket — the base series (the hourly profile is a `sum` over `span=1h`).	MB
`trackme.splk.vol.last_60m_volume_mb`	Rolling volume over the last 60 minutes, at each bucket end.	MB
`trackme.splk.vol.last_4h_volume_mb`	Rolling volume over the last 4 hours.	MB
`trackme.splk.vol.last_12h_volume_mb`	Rolling volume over the last 12 hours.	MB
`trackme.splk.vol.last_24h_volume_mb`	Rolling volume over the last 24 hours — the default outliers input.	MB
`trackme.splk.vol.last_24h_pool_pct_used`	Share of the license pool consumed over the last 24 hours (Splunk Enterprise only — never emitted on Splunk Cloud, where a pool has no limit).	percent
`trackme.splk.vol.daily_volume_mb`	Licensed volume of a closed **license day** (the day rolls at the license manager's local midnight), stamped in the middle of that day.	MB
`trackme.splk.vol.status`	Decision-maker status of the index.	status code

Note

The rolling values are computed by the collector from the buckets it holds and *then* written as metrics — they never depend on the metrics index's own indexing latency. A backfill at tenant creation writes the same series over the requested history with historical timestamps, so `mstats` over the past works from day one. See *Volume Outliers — in depth*.

Flex and FQM — your own metrics

Flex and *FQM* are different: **you** decide the metrics. Whatever numeric keys your tracker's SPL puts in its `metrics` JSON become metrics, one per key:

- **Flex** → `trackme.splk.flx.<your_metric_key>`
- **FQM** → `trackme.splk.fqm.<your_field_metric>` (field-quality results land under the `trackme.splk.fqm.fields_quality.*` sub-family, e.g. `percent_success`).

So there is no fixed catalogue for these — list what a given tracker produces with `mcatalog` filtered by `object_id`. A few leaves are **reserved** (TrackMe's own, not your data): `trackme.splk.flx.status`, and for converging trackers `trackme.splk.flx.count_entities_up` / `count_entities_down`; likewise `trackme.splk.fqm.status`.

Note

There is **no** `trackme.splk.cim.*` namespace — CIM compliance tracking is built on Flex, so its KPIs are emitted as `trackme.splk.flx.*` from the CIM tracker's SPL.

Dimensions and how to query

Every metric carries identifying **dimensions** — most commonly `tenant_id`, `object`, `object_id`, and `object_category` (the component), plus `alias` and `priority` on the SLA metric, `component` / `tracker` on the operational metrics, `model_name` on DSM sampling, `idx` / `st` on DHM, and `metric_category` on MHM. The value itself is in `_value`.

```
# List every metric name in the index
| mcatalog values(metric_name) WHERE index=trackme_metrics

# See the dimensions carried by a metric
| mcatalog values(_dims) WHERE index=trackme_metrics␣
↪metric_name="trackme.splk.feeds.lag_event_sec"

# Preview raw data points
| mpreview index=trackme_metrics filter="metric_name=trackme.sla.object_state"␣
↪earliest=-1h

# Aggregate — average DSM latency per entity over 5-minute spans
| mstats avg(trackme.splk.feeds.avg_latency_5m) AS avg_latency
    WHERE index=trackme_metrics object_category="splk-dsm" BY object span=5m
```

See also

- *Events & sourcetypes* — the event sourcetypes and indexes.
- *Impact scoring & thresholds* — how delay, latency, and thresholds drive the score.
- *SLA policies* — SLA reporting built on `trackme.sla.object_state`.

CHAPTER

NINE

WHITE PAPERS:

9.1 TrackMe's White Papers

About TrackMe's white papers

- TrackMe is a powerful, flexible and large solution that can be used to monitor Splunk environments, from feeds tracking to the search Workload, Performance and infrastructure tracking, functional controls and more!
- With these white papers, tutorials and documentations, we aim at providing and sharing deep knowledge around our most valuable practices and use cases.
- This collection of articles will only get better over time!

Browse the white papers by theme in the menu:

- *Splunk infrastructure monitoring* — SVC consumption, workload, indexer clusters, and search head clusters.
- *Detection & alerting use cases* — volume outliers on the license usage with VOL, who is doing what on your Splunk with UAM, event-drop detection, log-level anomalies with ML, and two-tier SLA alerting.
- *Operations & administration* — backup & restore, entity auto-deletion, and mass operations.

For guided starting points and worked demonstrations, see the *Quickstart* and the *use-case demos*.

9.2 Running a TrackMe Proof of Concept

Running a TrackMe Proof of Concept (POC) is surely the best way to demonstrate the key values of the product, and how it can help you tackle your monitoring challenges. This document aims to guide you through the main steps to run a successful POC.

- We stand by to help you at any time, so please do not hesitate to reach out to us if you need any assistance: contact@trackme-solutions.com
- TrackMe has different licensing modes. Without a license, TrackMe runs in Foundation Edition trial mode, which is intended for evaluation and eventually becomes read-only after expiry. For a POC, we recommend using a full temporary license to avoid limitations.
- Instead, contact us and we will be delighted to provide you a full temporary license, which can cover at least 60 days of usage, which could eventually be renewed for a further 30 days if the buying process is still ongoing.
- TrackMe is a very rich and powerful solution, you may not test every single feature or capability during the POC, but we recommend you to focus on the key features that are most relevant to your monitoring challenges.

9.2.1 Identifying the Key Use Cases

TrackMe can handle many different use cases, below is a non exhaustive list of the most common use cases that TrackMe can help you tackle:

- `Feeds tracking`: Monitor any kind of feeds making to Splunk, from the data source perspective (index / sourcetype / additional Metadata if needed), from the event host perspective, and more.
- `Scheduled search and Splunk Enterprise Security correlation searches`: Track the scheduled activity of your Splunk environment, detect scheduled anomalies and investigate capacity planning, TrackMe as your single pane of glass.
- `TrackMe Flex Objects, monitor your Splunk deployment, configuration, perform SIEM controls, anything!`: Use TrackMe Flex objects, locally or remotely, and monitor any layer of Splunk, from Search Head Cluster and Indexer Clusters health and activity, Knowledge bundle size, licence usages and literally anything you can think of.

9.2.2 Identify where to deploy TrackMe

Review TrackMe requirements and identify where to deploy TrackMe:

- Review the main installation documentation: *Installation of TrackMe*
- For the purpose of a POC, you may want to prioritise a deployment on a standalone server for simplicity, good candidates are usually a standalone Search Head that is already used for monitoring purposes, or the utility node running the Splunk Monitoring Console. (eg. DMC)
- If you are a Splunk Cloud customer, then there are not any questions and TrackMe will run on the ad hoc Search Head tier.
- You can always at a later time decide to migrate TrackMe to a definitive Search Head layer target.

9.2.3 Service account and permissions

- By default, TrackMe runs as the Splunk nobody user (eg. splunk-system-user), it is not strictly required to create a dedicated service account for TrackMe.
- However, it is recommended to create a service account for TrackMe as a good configuration practice, review: *Configuration*
- In the context of a POC, you may totally skip this step and save more time to focus on the use cases instead.
- If this is however a strong requirement and especially in some specific contexts, this can be done since day 1 easily by following the documentation above.

9.2.4 Roles Based Access Control (RBAC)

- TrackMe supports Roles Based Access Control, relying on Splunk native capabilities with our implementation.
- TrackMe comes with 3 builtin Splunk Roles: `trackme_admin`, `trackme_power`, `trackme_user`.
- TrackMe also requires its users to have TrackMe capabilities. (which are provided by the roles above)
- Review: *Roles & access control*

9.2.5 Install TrackMe

- Installing TrackMe is straightforward, Review the main installation documentation: *Installation of TrackMe*
- Ensure you are meeting application dependencies requirements
- Ensure you have defined and published TrackMe indexes in your indexing layer, notably for Splunk Enterprise customers. (in Splunk Cloud, this is automated once the application is deployed)
- Remember that TrackMe does not do anything at all once installed and until you start creating TrackMe Virtual Tenants and trackers, so it is fully safe to have TrackMe installed ahead of time.

- Also, pay attention to TrackMe new releases, we aim at publishing new releases at least once per month, fixing, enhancing and adding new features to enrich the product.
- We also recommend to install and enable the `TrackMe Configuration Manager` app (TCM), this allows registering each administrative action in TrackMe (such as the creation of tenants or trackers) and replay these eventually later on, this is also a good mean for you to understand how TrackMe works underneath!
- Review TrackMe TCM: *Configuration Manager (CI/CD)*

9.2.6 Register a TrackMe licence for the POC

- In the context of a POC, we advise you to contact us to get a temporary licence for an extended period of time.
- You can also generate a TrackMe trial licence for 90 days directly in TrackMe. (requires your Search Head to have external traffic connectivity with our public licence API services, https/443 outgoing)
- Below are the different TrackMe offerings and their features and restrictions; the pricing page (https://trackme-solutions.com/pricing) is the reference for the current limits:

TrackMe Foundation Edition (trial):

- 3 Virtual Tenants limited
- 1 remote Splunk deployment limited
- Splunk feeds tracking (DSM, DHM, MHM)
- Volume Outliers (VOL): the licensed volume of every index
- Up to 8 Hybrid Trackers (total)
- Unlimited Elastic trackers
- Machine Learning Outliers detection (Splunk feeds)
- Standard support

TrackMe Enterprise Edition:

- Up to 6 Virtual Tenants limited
- Up to 8 remote deployments
- Splunk feeds tracking
- Up to 16 Hybrid Trackers
- Unlimited Elastic trackers
- Machine Learning Outliers detection for all components
- Splunk Workload
- Fields Quality Monitoring and Common Information Model compliance tracking (32 trackers)
- Flex Object Tracking (32 trackers)
- Volume Outliers (VOL)
- The dedicated tenant types: User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge
- Premium support (24 hours SLA, 8.am to 8.pm UK time)

TrackMe Unlimited Edition:

- Unlimited number of Virtual Tenants
- Unlimited remote deployments
- Splunk feeds tracking
- Unlimited number of Hybrid Trackers

- Unlimited Elastic trackers
- Machine Learning Outliers detection for all components
- Splunk Workload
- Fields Quality Monitoring and Common Information Model compliance tracking
- Flex Object Tracking
- Volume Outliers (VOL)
- The dedicated tenant types: User Activity Monitoring, Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge
- Premium support (24 hours SLA, 8h AM to 8h PM UK time)

TrackMe Unlimited Deployments: the Unlimited Edition for every Splunk deployment of the organisation, under one licence.

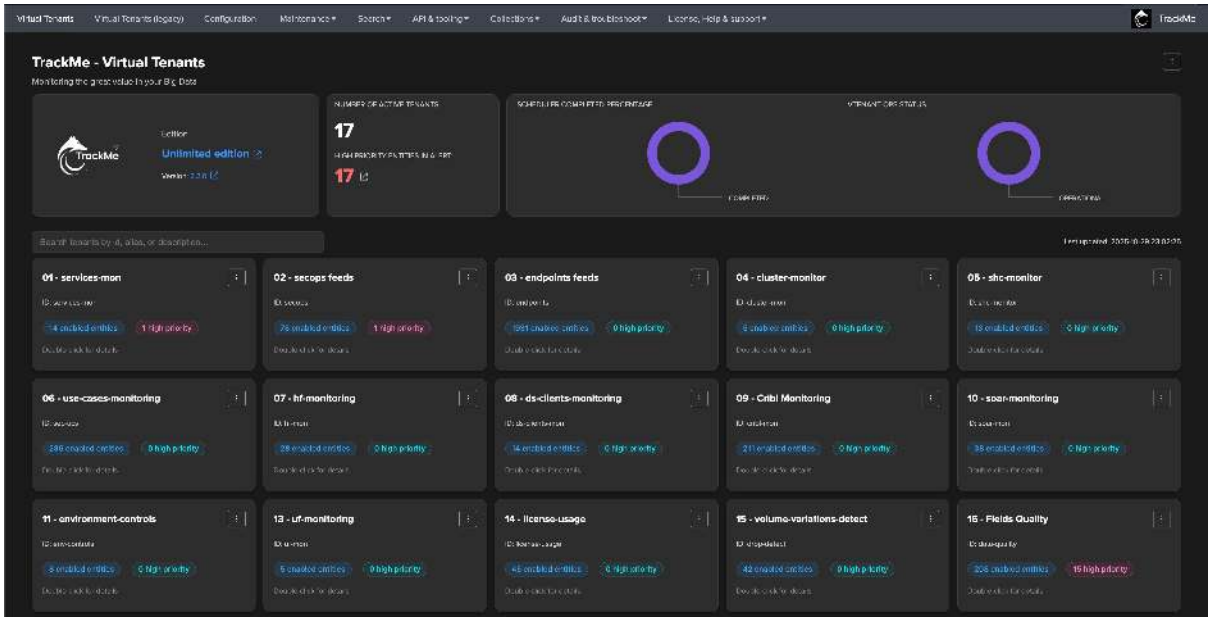
9.2.7 Remote deployment and multiple Search Head tiers

- TrackMe provides native capabilities to interact with any remote Splunk deployment or instance, from Search Head tiers to utility nodes such as the Cluster Manager, or Heavy Forwarders.
- When it comes to the TrackMe Workload component (monitoring of Splunk scheduling), we also use this feature to interact with the Splunk API for purposes such as the search versioning.
- Review: *Remote Splunk deployments*
- For this, you should identify the targets, and get a bearer token created per target as well as satisfying basic networking requirements.

9.2.8 Design TrackMe tenants

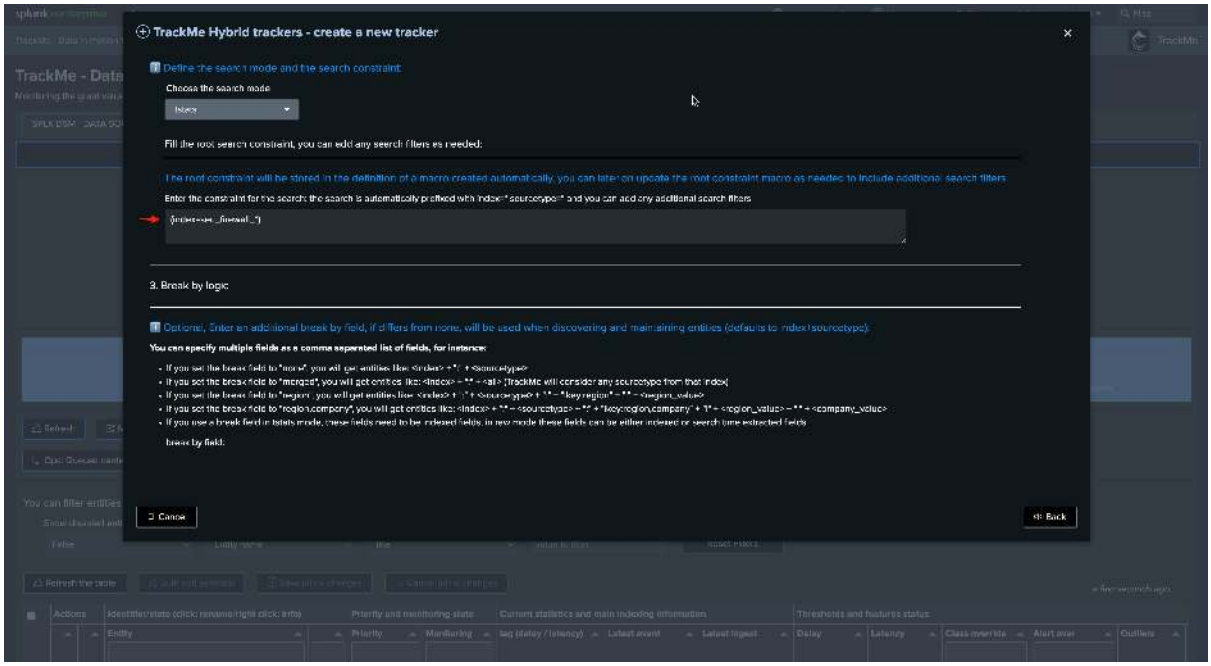
- Ahead of the POC or during the first phases, you will want to consider how tenants and the purpose of each tenant for your TrackMe deployments.
- These decisions are usually influenced by your own technical / functional contexts, as well as the features you want to use in TrackMe.
- A tenant can run multiple TrackMe component (example: splk-dsm for Data source tracking and splk-flx)
- You can dedicate a TrackMe Virtual Tenant for a specific perimeter, or a specific team, or a specific use case, or a mix of all of these depending on your needs.
- Remember that you can experiment, a Virtual Tenant can be created, destroyed, disabled, etc!

A realistic example:

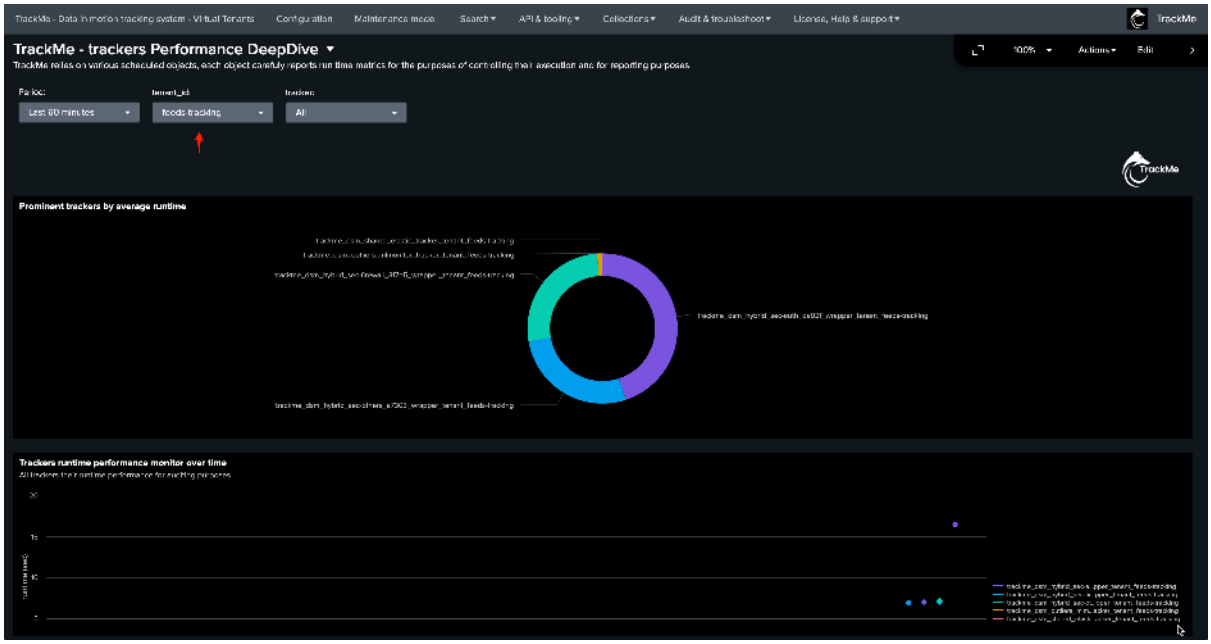


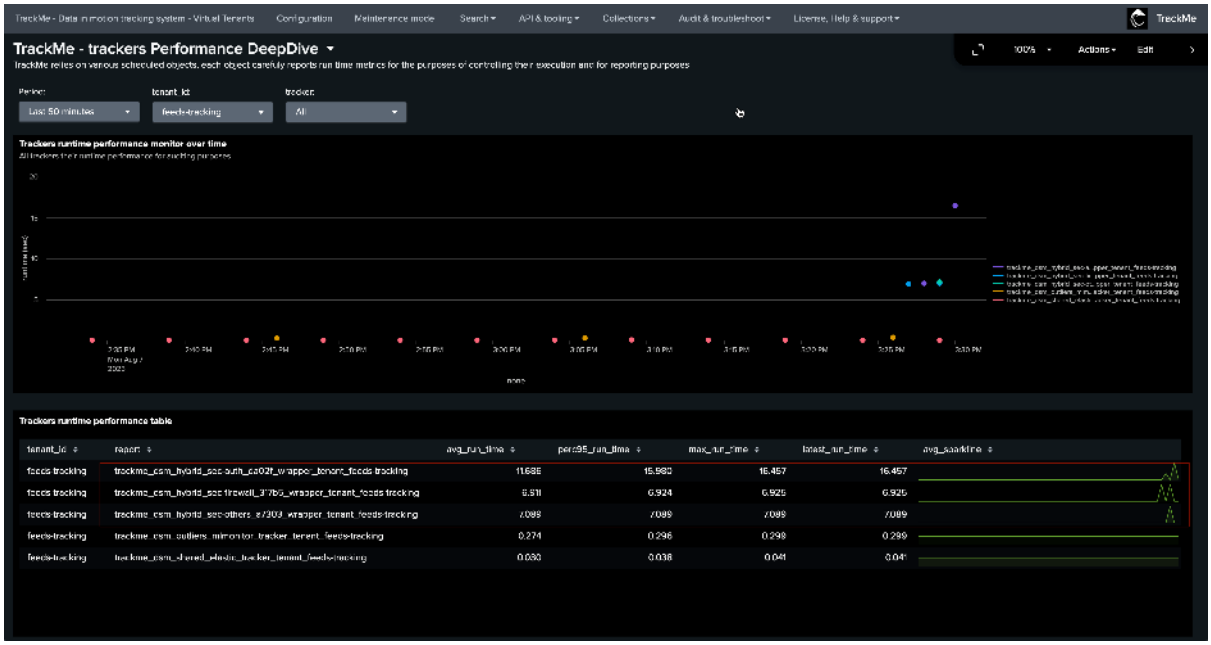
9.2.9 Design TrackMe at scale

- A best practice when creating Tenants and for Feeds tracking is to create Hybrid Trackers after the creation of the Virtual Tenants rather than when creating the Tenants.
- This allows more control and more flexibility, allowing to adopt scaling best practices and design performing and efficient Hybrid Trackers easily.
- Remember that you can leverage various Splunk techniques easily in TrackMe, such as using Splunk indexed fields to filter out or influence TrackMe entities definition and reflect your data pipelines easily.
- Review: *Large-scale environments*



- Use out of the box TrackMe tooling to monitor the performing and costs of your Hybrid Trackers easily:

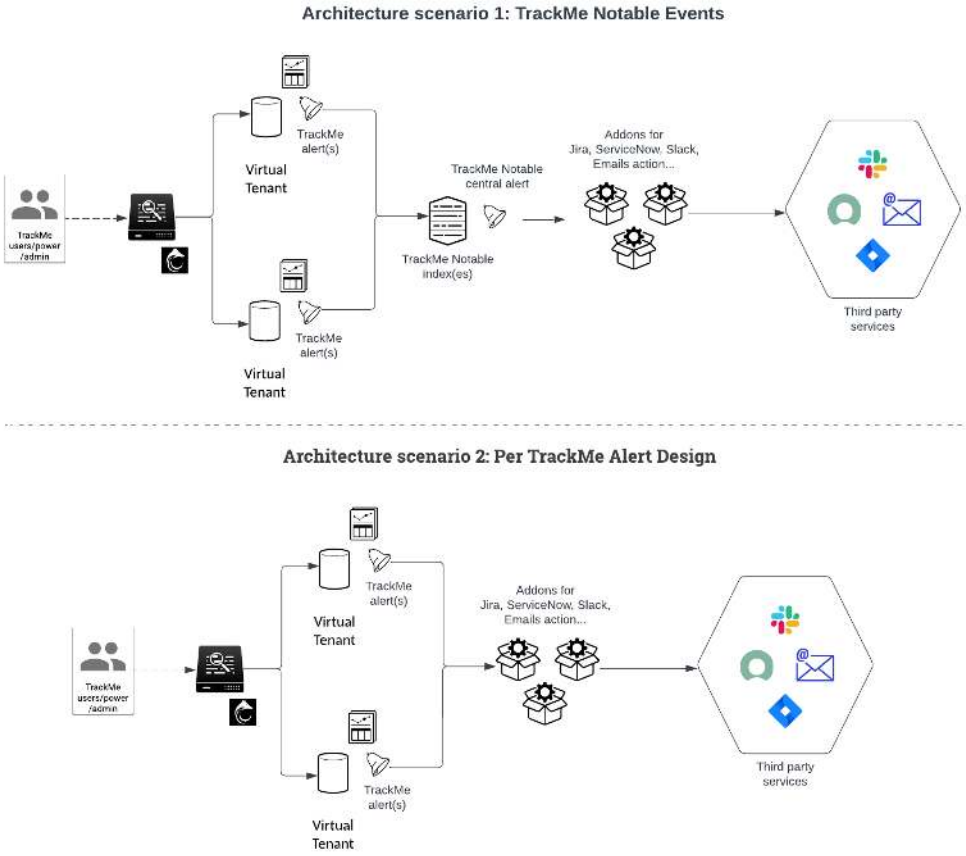




9.2.10 Design your alerting strategy

TrackMe provides a flexible out of the box workflow when it comes to alerting:

- On a per tenant basis, you can create in a very simple step a TrackMe alert
- TrackMe alerts are designed to run automated TrackMe alert actions, such as generating `TrackMe Notable` events, performing `automated Acknowledgements` and run `SmartStatus` investigations.
- A TrackMe architecture best practice for alerting is to rely on TrackMe notable events rather than implementing third party notifications from TrackMe alerts, allowing to perform further correlation, enrichment or filtering on the notable events.
- Review: *Stateful alerting*



9.3 Splunk infrastructure monitoring

White papers on using TrackMe to monitor Splunk's own infrastructure — SVC consumption, the search workload, indexer clusters, and search head clusters.

9.3.1 Tracking Splunk Cloud SVC consumption in TrackMe

Monitoring the Splunk Cloud SVC consumption using TrackMe

- This white paper aims at providing a detailed guide on how to monitor the Splunk Cloud SVC consumption using TrackMe.
- The Splunk Cloud SVC consumption is a key metric to monitor in order to ensure the Splunk Cloud service is used efficiently and effectively.

Introduction about Splunk Cloud SVC

Whenever you are a Cloud customer paying by the volume, or by the capacity allocated to your environment (stack), SVC consumption is the key indicator in Splunk Cloud which allows you to understand the costs associated with anything that consumes computes in Splunk:

Essentially:

- Indexing related activities
- Users' ad-hoc searches

- Users and applications scheduled searches
- DataModel acceleration (DMA)

In TrackMe, you can leverage this key performance indicator to monitor the consumption of your Splunk Cloud environment, and detect abnormal trends.

To do so, we can leverage two components in TrackMe:

- TrackMe Flex Objects trackers, which can monitor SVC consumptions based on different patterns, and notably leverage Machine Learning to detect abnormal trends.
- TrackMe Workload, which can monitor the consumption of SVC at the levels of scheduled searches.

Splunk generates SVC consumption metrics in the summary index:

Global SVC consumption:

```
index=summary source=splunk-svc
```

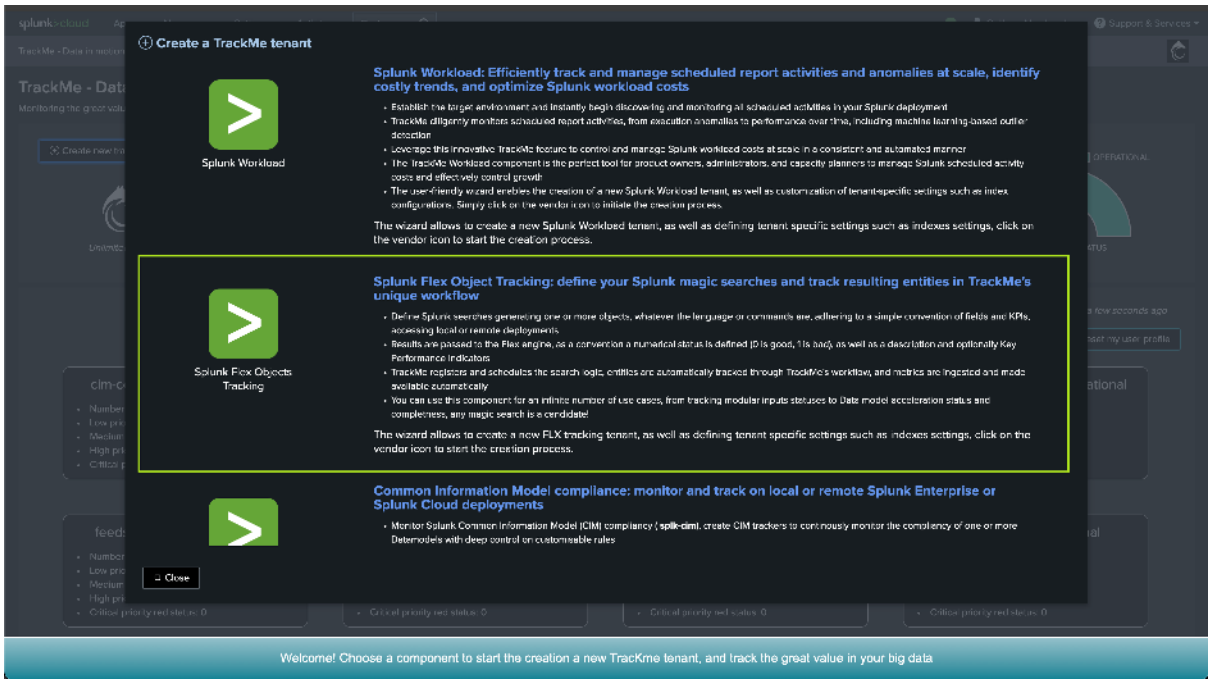
Detailed SVC consumption per consumer:

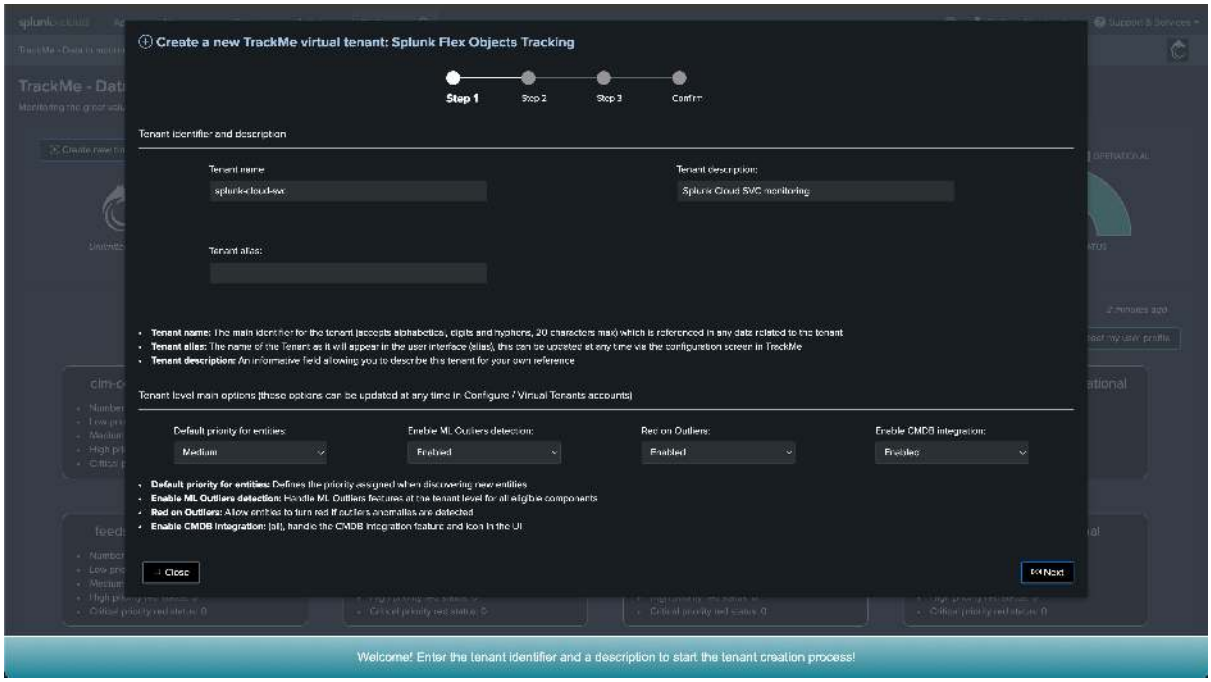
```
index=summary source=splunk-svc-consumer
```

TrackMe leverages these indicators and stores these as metrics into the TrackMe metric indexes, and can leverage basic logics or more advanced logics using Machine Learning Outliers detection.

TrackMe Flex trackers for Splunk Cloud SVC consumption

We will start by creating a dedicated Virtual Tenant for the purposes of tracking Splunk Cloud SVC consumption, we can use the UI and create a new Flex Object enabled tenant:





Or we can use the trackme custom command:

```
| trackme url="/services/trackme/v2/vtenants/admin/add_tenant" mode="post" body="{
↪'tenant_alias':'splunk-cloud-svc','tenant_desc':'Splunk Cloud SVC monitoring',
↪'tenant_name':'splunk-cloud-svc','tenant_roles_admin':['trackme_admin'],
↪'tenant_roles_power':['trackme_power'],'tenant_roles_user':['trackme_user'],
↪'tenant_owner':'nobody','tenant_idx_settings':'{\'trackme_summary_idx\': \
↪'trackme_summary\', \'trackme_audit_idx\': \'trackme_audit\', \
↪'trackme_notable_idx\': \'trackme_notable\', \'trackme_metric_idx\': \
↪'trackme_metrics\'}','tenant_outliers_set_state':'1','tenant_mloutliers':'1',
↪'tenant_cmdb_lookup':'1','tenant_default_priority':'medium','tenant_flx_enabled
↪':1}"
```

Once the Virtual Tenant is created, we can leverage the TrackMe Flex object library and rely on the two out of the box Flex Object use cases to monitoring SVC:

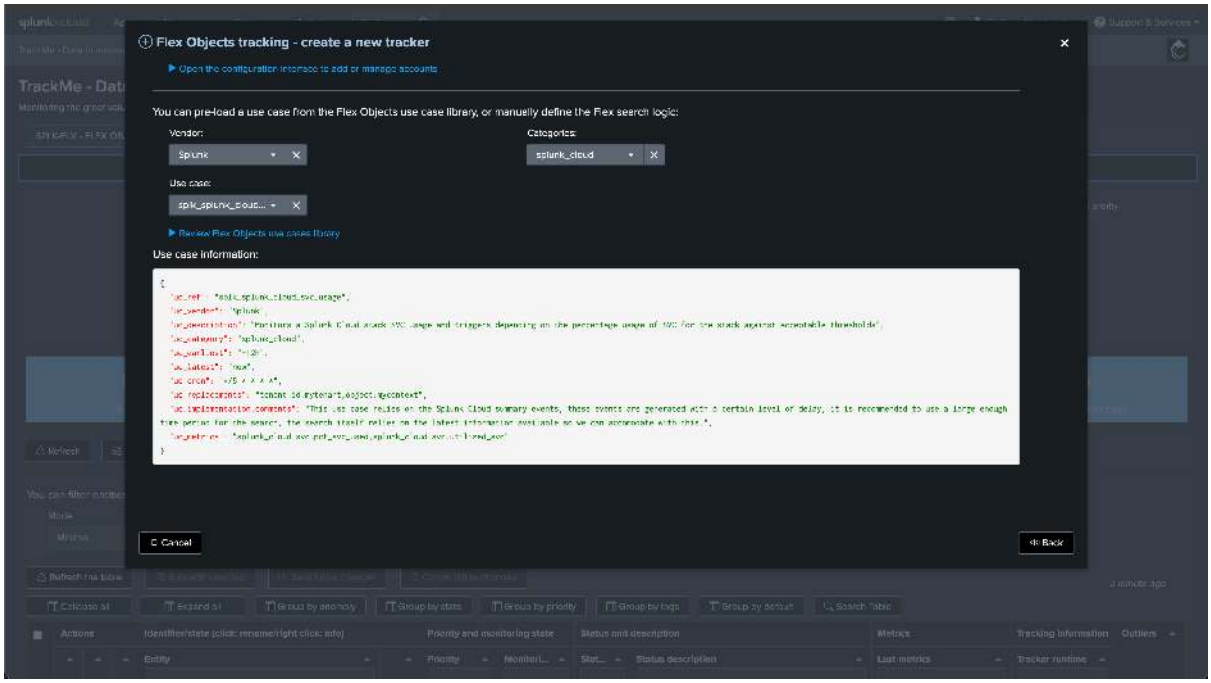
Hint

TrackMe version 2.0.99

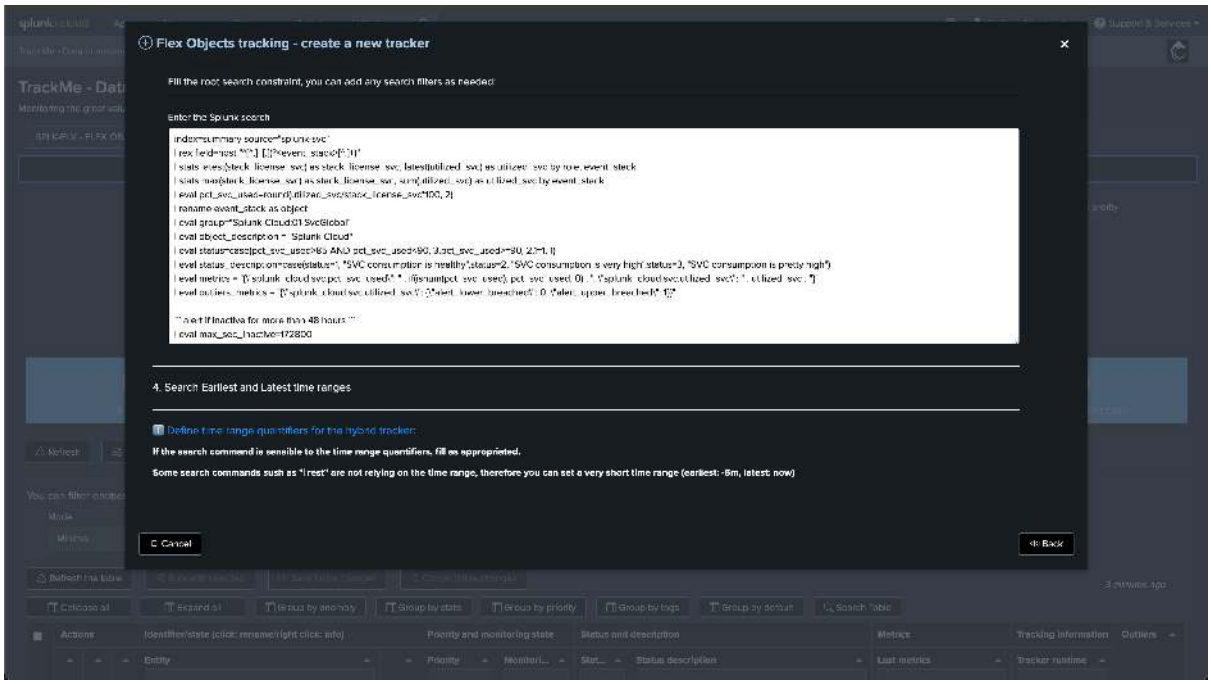
- These two use cases were last updated in TrackMe 2.0.99, make sure you are using this version or later.

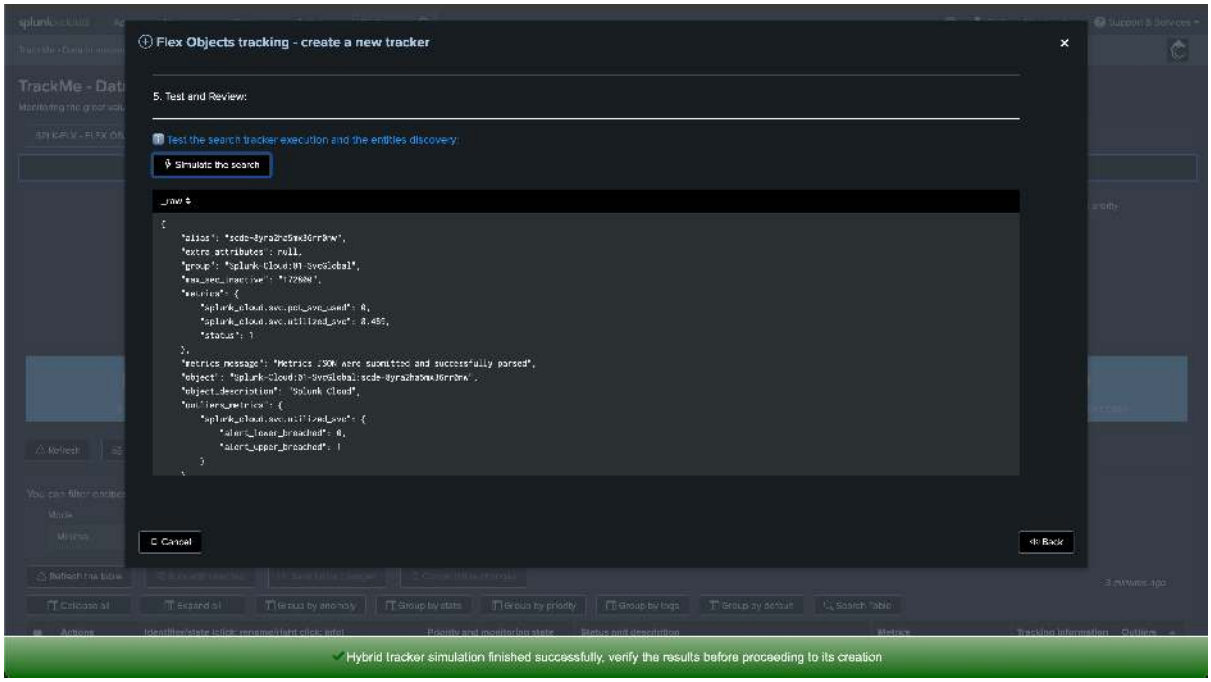
TrackMe stack global SVC consumption

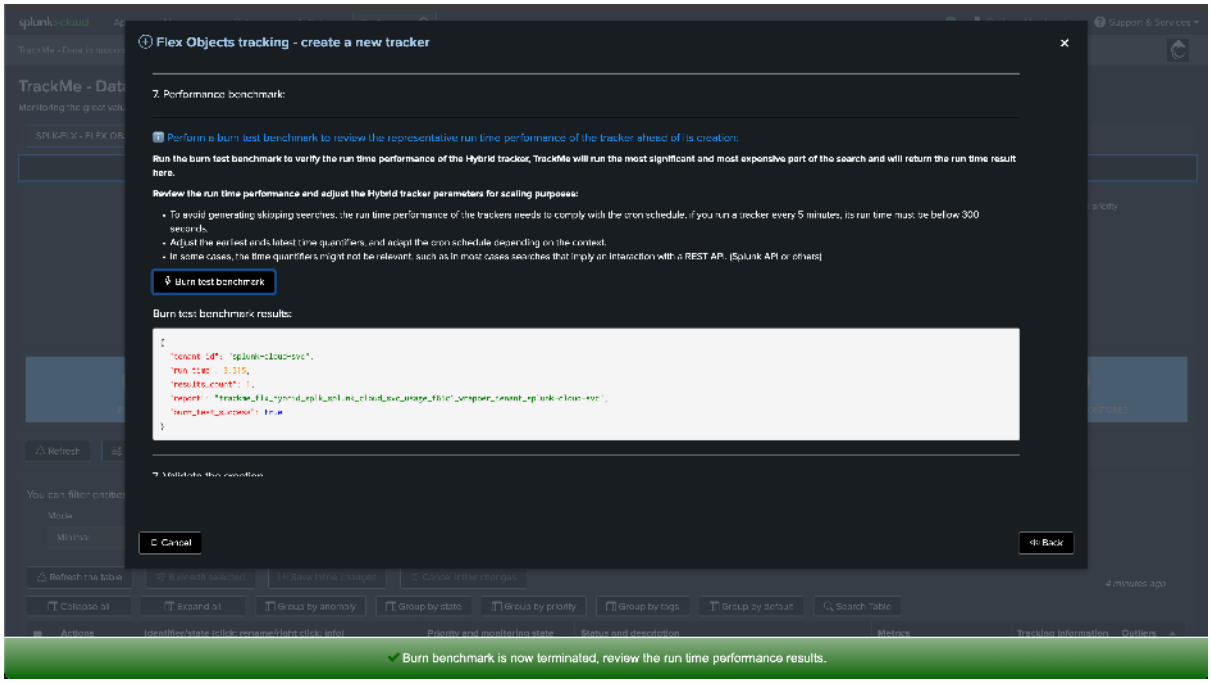
Once in the TrackMe UI, create a new Flex Object tracker and select the following use case template:

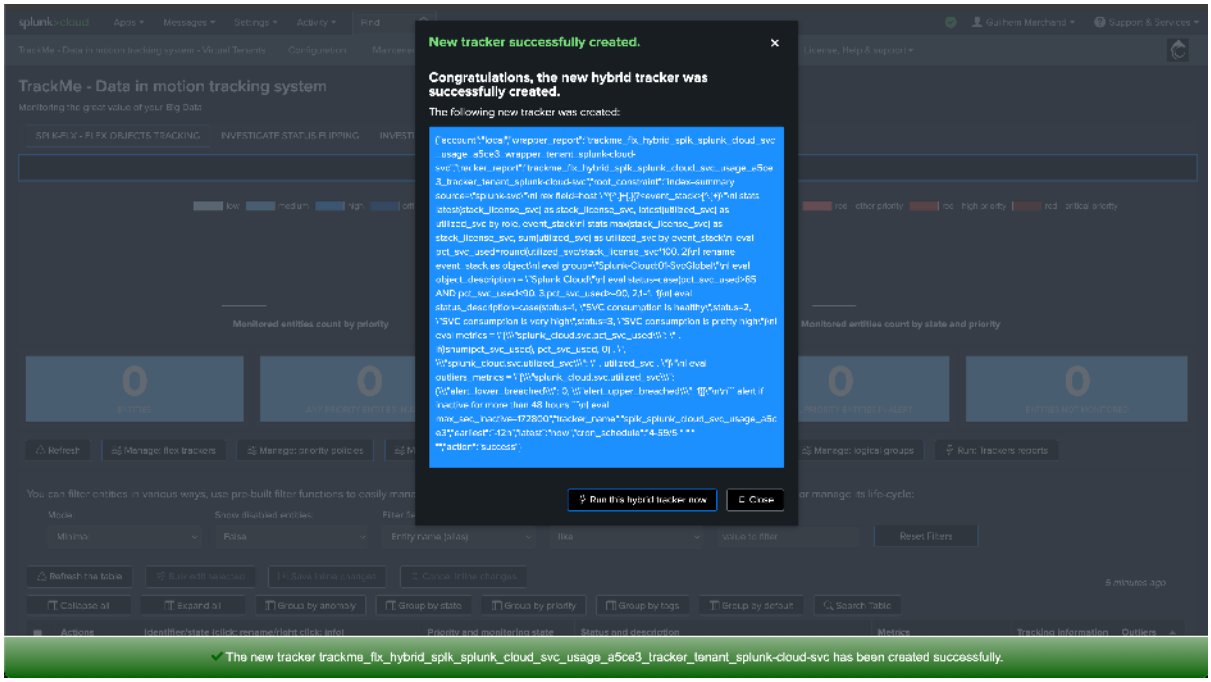


Scroll down and execute the tracker. You can also customize the tracker if you wish to do so, although this is not mandatory and the use case will work out of the box:

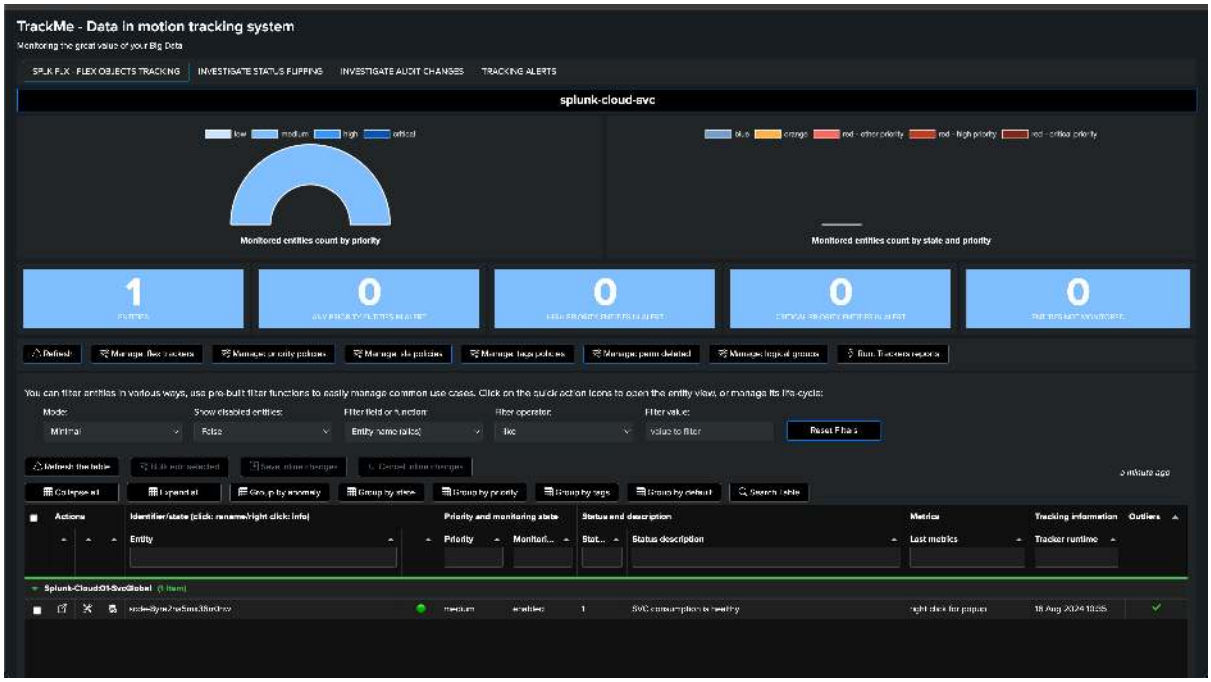



Flex Objects tracking - create a new tracker
5. Test and Review:
Test the search tracker execution and the entities discovery:
Simulate the search
Cancel
Back
Hybrid tracker simulation finished successfully, verify the results before proceeding to its creation


Flex Objects tracking - create a new tracker
7. Performance benchmark:
Perform a burn test benchmark to review the representative run time performance of the tracker ahead of its creation:
Run the burn test benchmark to verify the run time performance of the Hybrid tracker, TrackMe will run the most significant and most expensive part of the search and will return the run time result here.
Review the run time performance and adjust the Hybrid tracker parameters for scaling purposes:
Burn test benchmark
Burn test benchmark results:
Cancel
Back
Burn benchmark is now terminated, review the run time performance results.



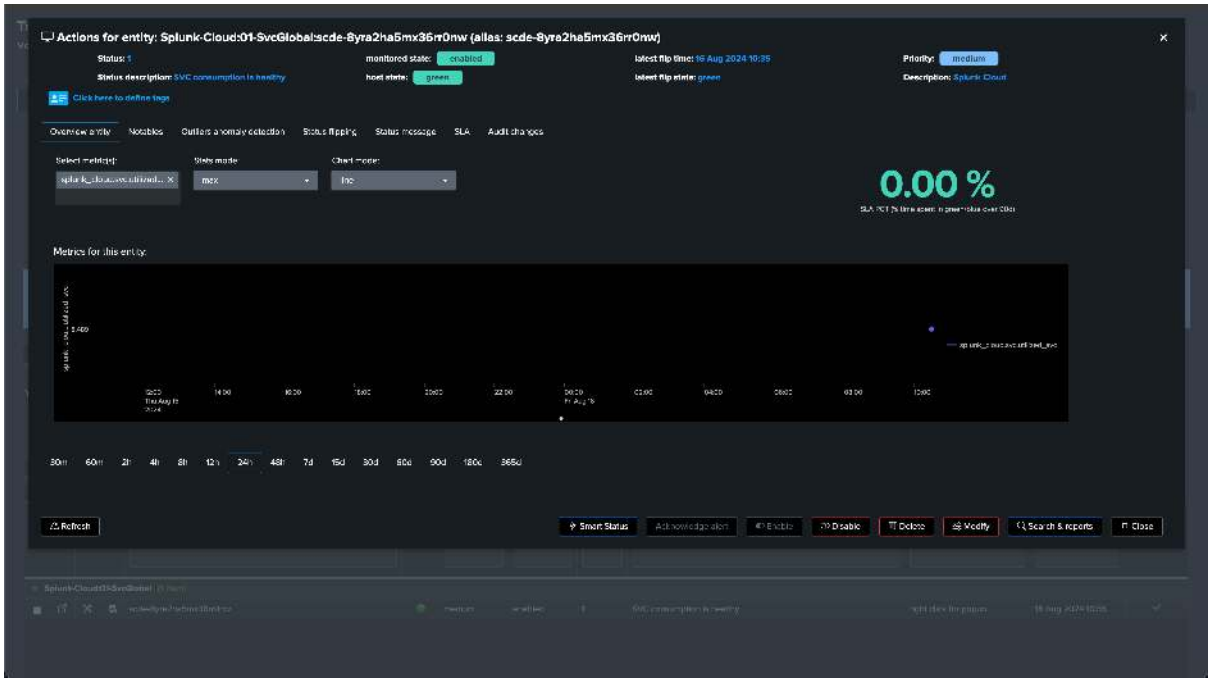
Once executed, you will see one TrackMe entity, which corresponds to the Splunk Cloud stack name:



If you open the entity, you will see two SVC kpis:

- `splunk_cloud.svc.pct_svc_used`: the percentage of SVC used in the stack
- `splunk_cloud.svc.utilized_svc`: the amount of SVC used in the stack

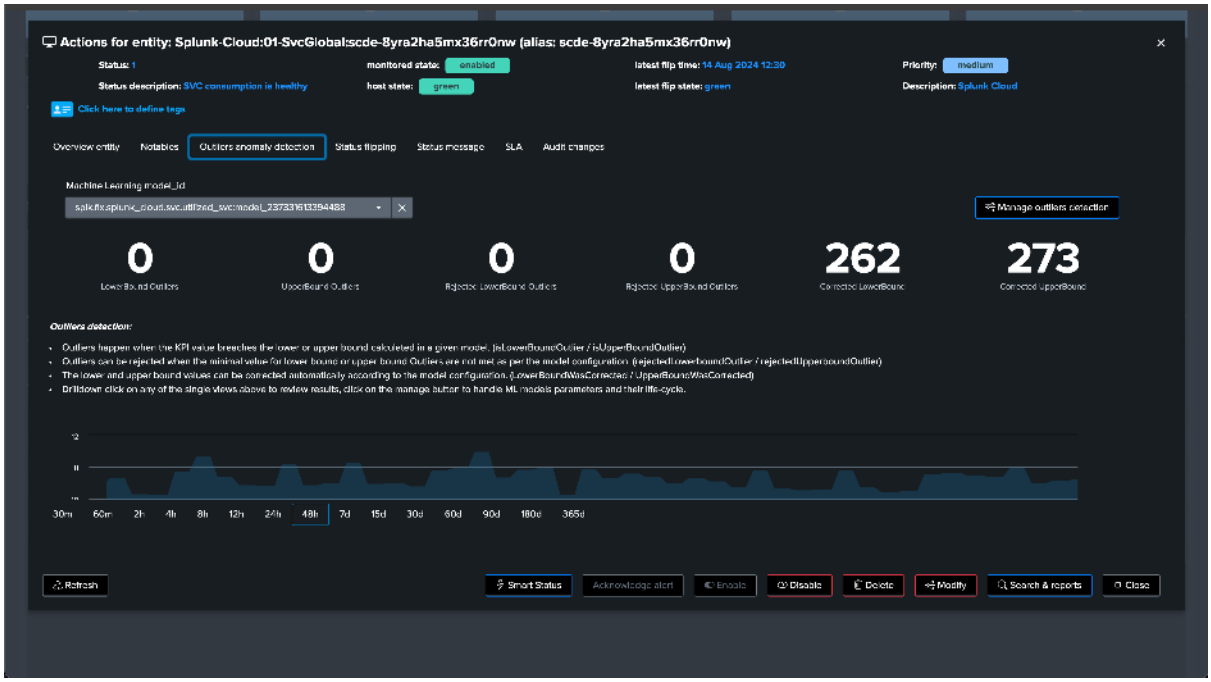
Example:

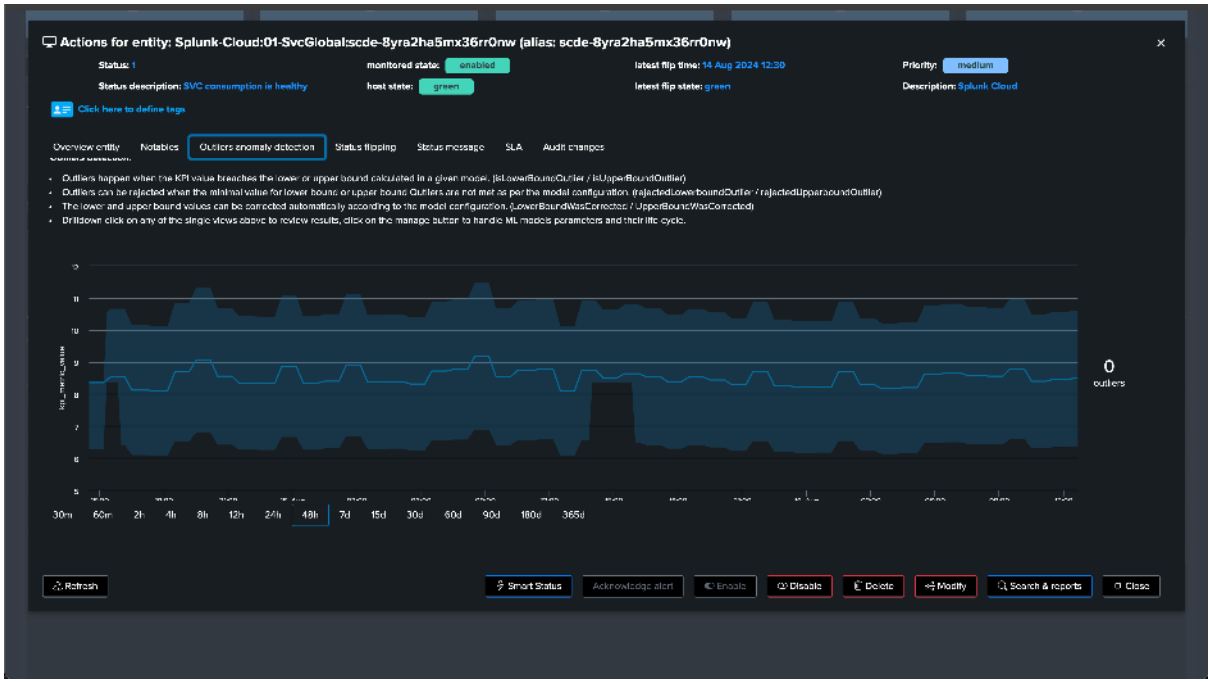


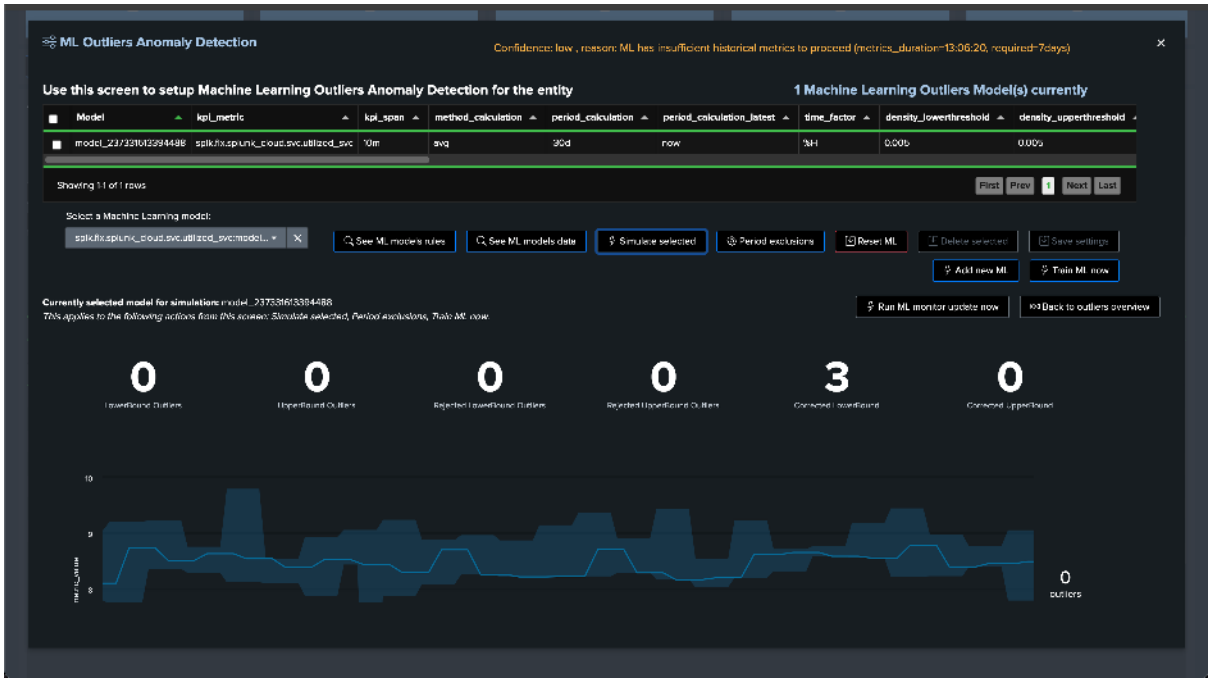
After some time, this will be look like:



Looking at Outliers anomaly detection, the use case automatically created a model for SVC consumption based on the SVC units, it would eventually trigger an outlier alert if an increasing abnormal trend is detected:

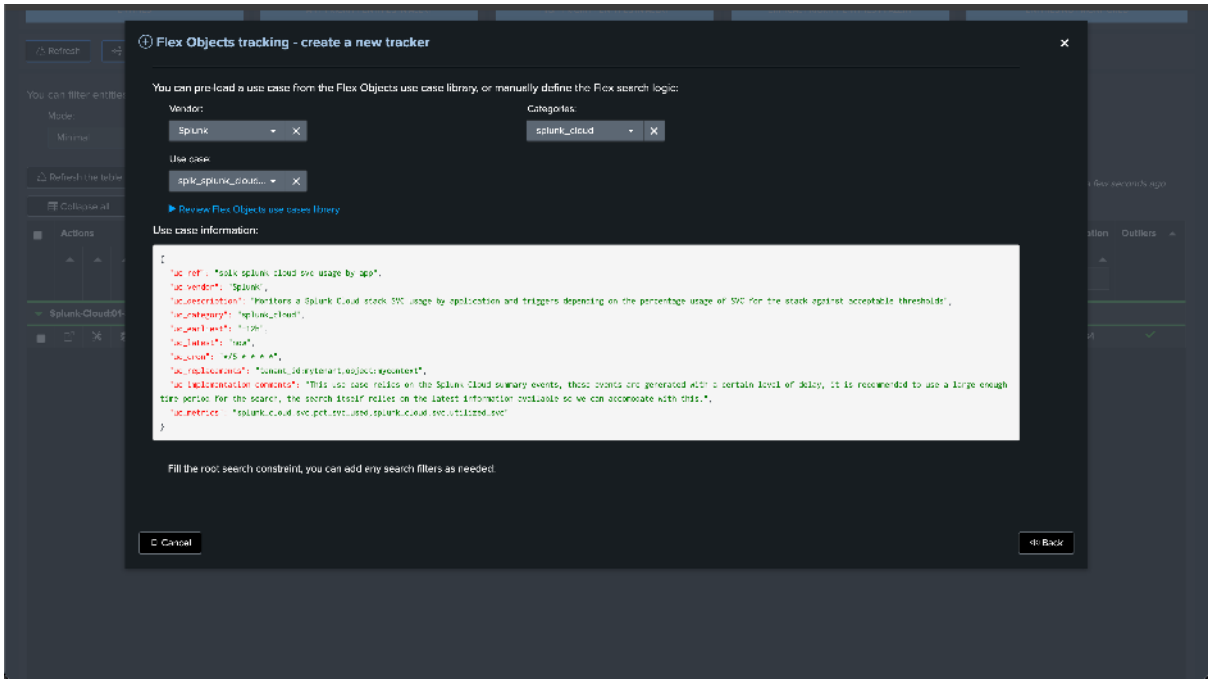

Actions for entity: Splunk-Cloud:01-SvcGlobal:scde-8yra2ha5mx36rr0nw (alias: scde-8yra2ha5mx36rr0nw)
Status: 1
monitored state: enabled
latest flip time: 14 Aug 2024 12:30
Priority: medium
Status description: SVC consumption is healthy
host state: green
latest flip state: green
Description: Splunk Cloud
Click here to define tags
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Machine Learning model_id
Manage outliers detection
0
0
0
0
262
273
Outliers detection:
Outliers happen when the KPI value breaches the lower or upper bound calculated in a given model. (isLowerBoundOutlier / isUpperBoundOutlier)
Outliers can be rejected when the minimal value for lower bound or upper bound Outliers are not met as per the model configuration. (rejectedLowerboundOutlier / rejectedUpperboundOutlier)
The lower and upper bound values can be corrected automatically according to the model configuration. (LowerBoundWasCorrected / UpperBoundWasCorrected)
Drilldown click on any of the single views above to review results, click on the manage button to handle ML models parameters and their life-cycle.
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search & reports
Close


Actions for entity: Splunk-Cloud:01-SvcGlobal:scde-8yra2ha5mx36rr0nw (alias: scde-8yra2ha5mx36rr0nw)
Status: 1
monitored state: enabled
latest flip time: 14 Aug 2024 12:30
Priority: medium
Status description: SVC consumption is healthy
host state: green
latest flip state: green
Description: Splunk Cloud
Click here to define tags
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
SLA
Audit changes
Outliers happen when the KPI value breaches the lower or upper bound calculated in a given model. (isLowerBoundOutlier / isUpperBoundOutlier)
Outliers can be rejected when the minimal value for lower bound or upper bound Outliers are not met as per the model configuration. (rejectedLowerboundOutlier / rejectedUpperboundOutlier)
The lower and upper bound values can be corrected automatically according to the model configuration. (LowerBoundWasCorrected / UpperBoundWasCorrected)
Drilldown click on any of the single views above to review results, click on the manage button to handle ML models parameters and their life-cycle.
0
outliers
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
180d
365d
Refresh
Smart Status
Acknowledge alert
Enable
Disable
Delete
Modify
Search & reports
Close

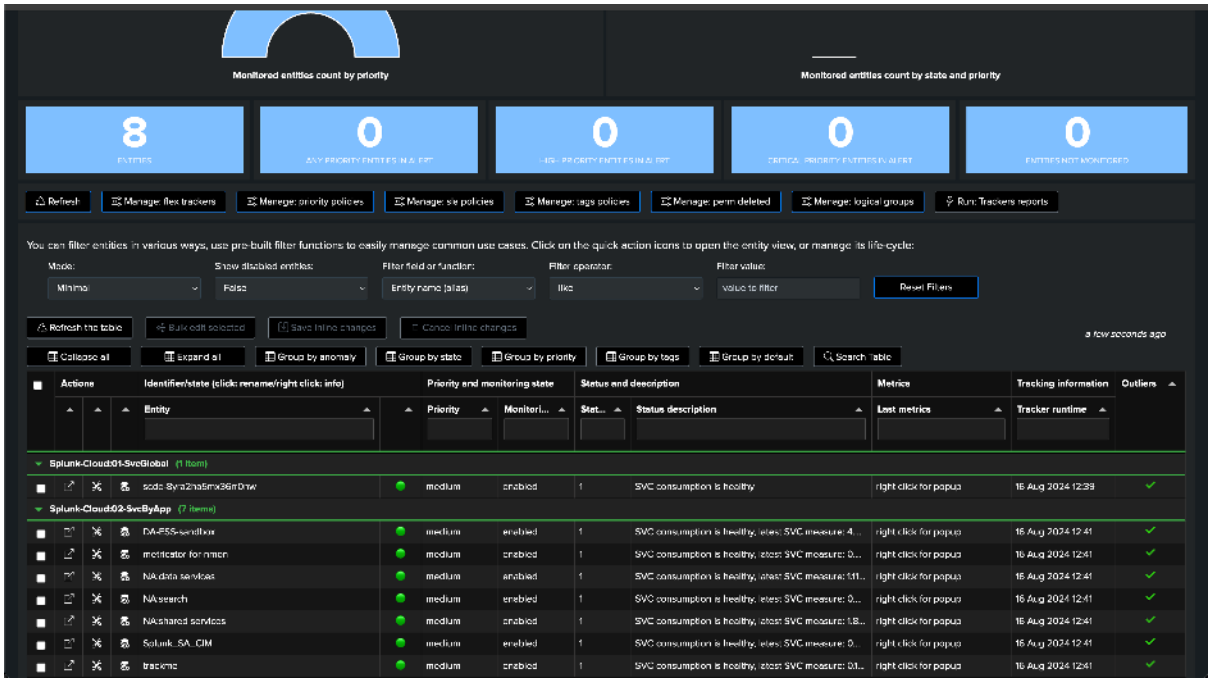


TrackMe stack app consumer SVC consumption

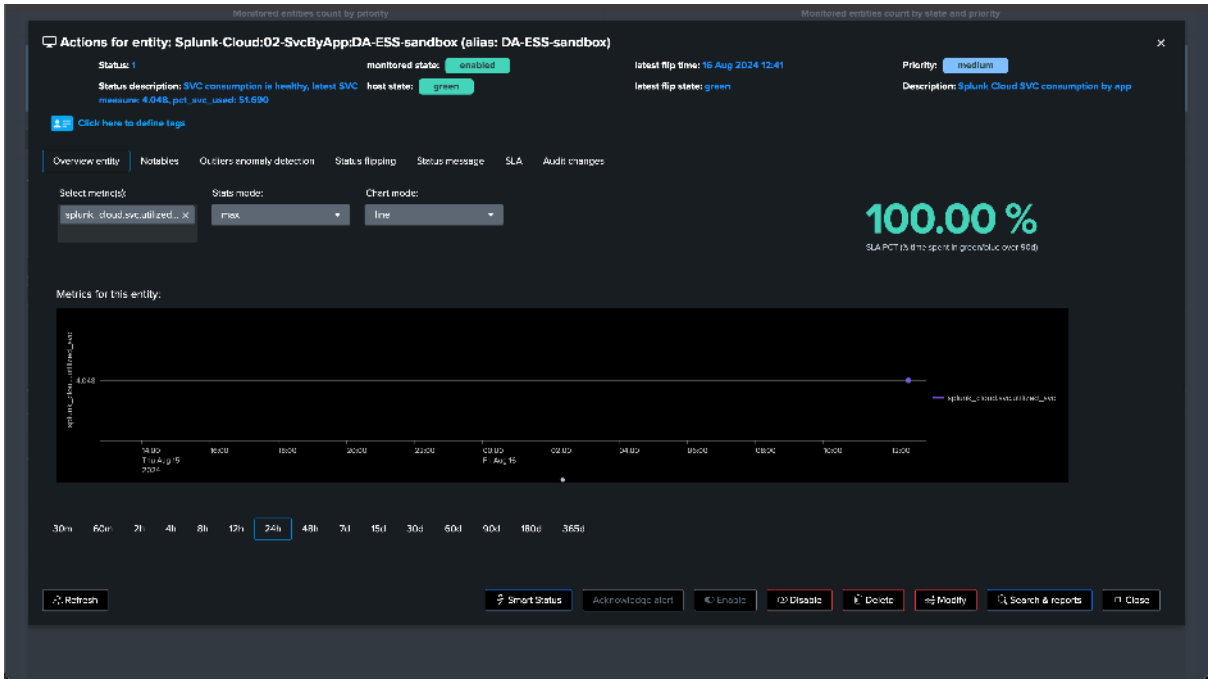
Similarly, we can now leverage the second out of the box Flex Object use case to monitor the SVC consumption at the app level:



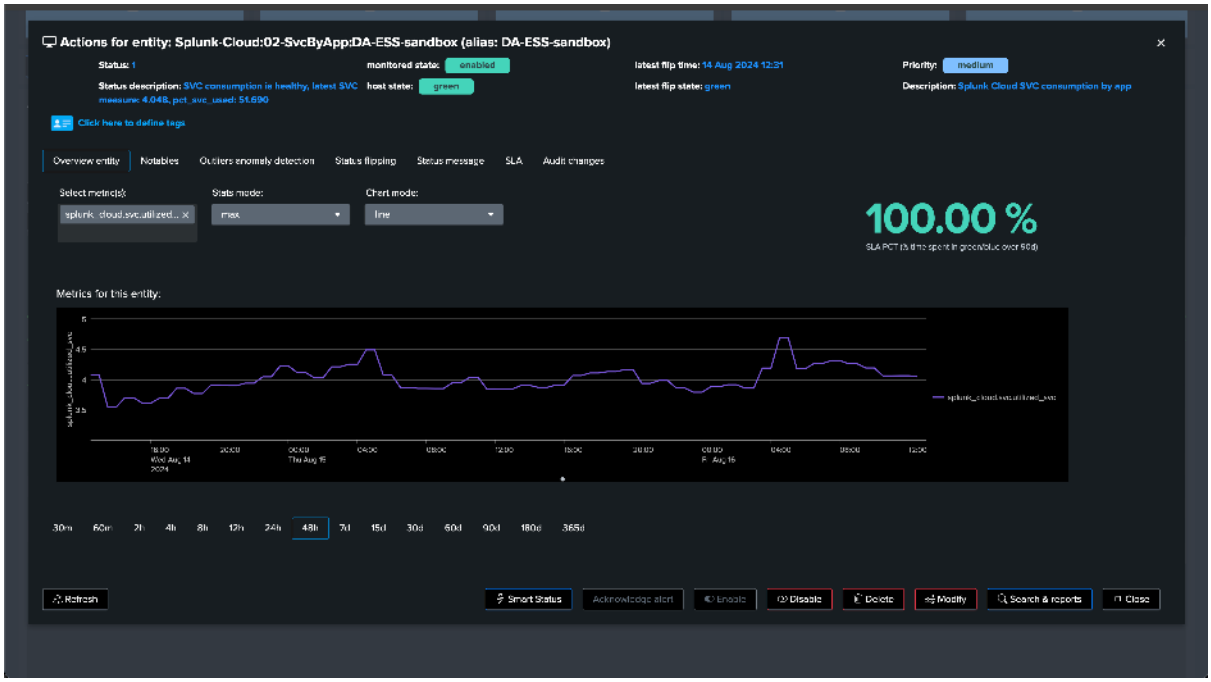
Once created and executed, this use case tracks the SVC consumption by the app consumer as registered by the SVC consumption KPIS:



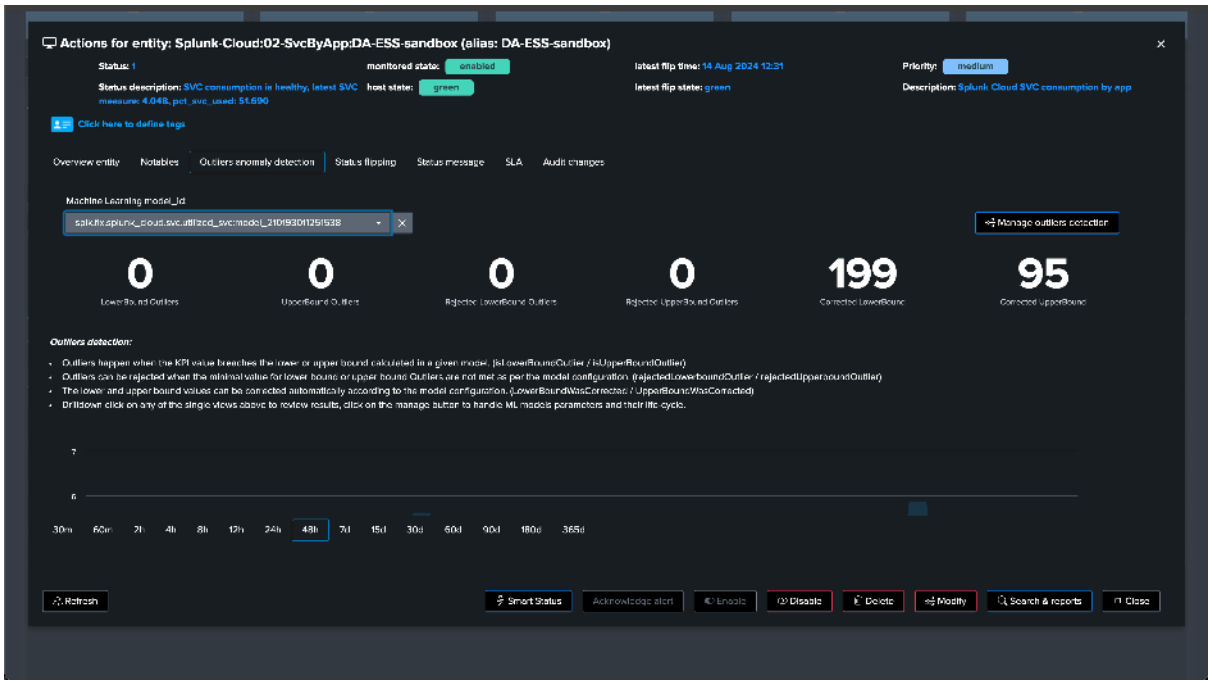
The same SVC KPI is leveraged per consumer:

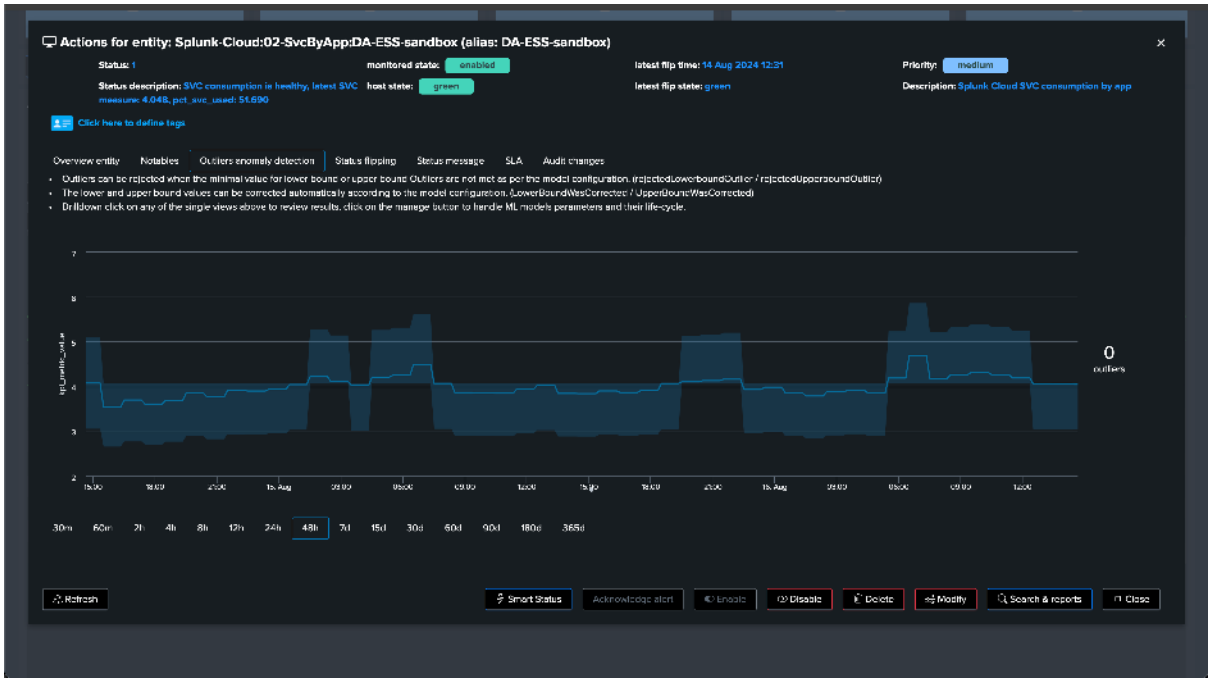


After some time, once we have started to collect enough historical knowledge:



Anomaly Outliers detection:

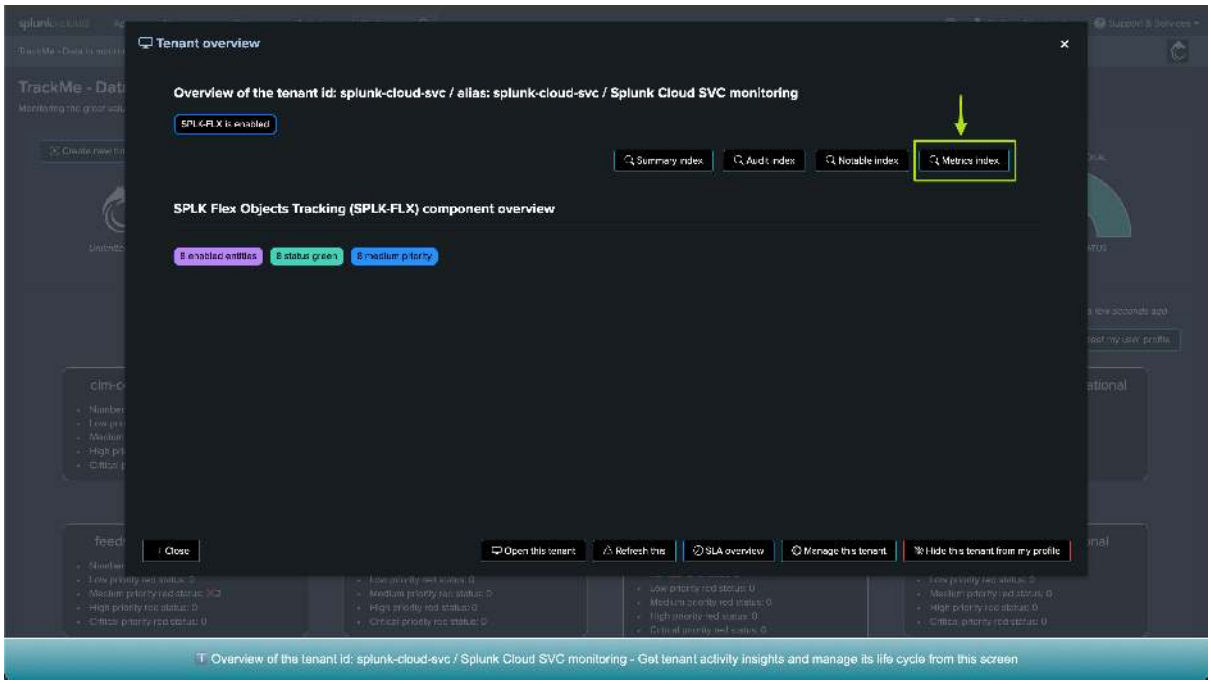


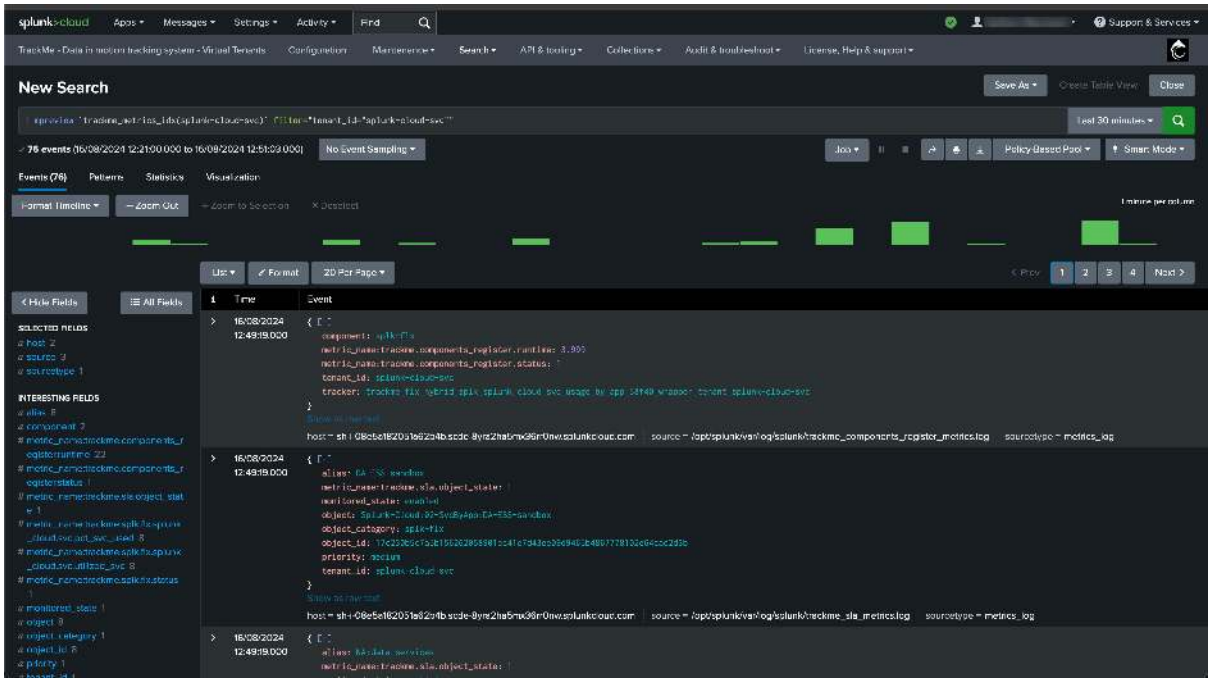


From this stage, if a consumer starts to abnormally consume SVC, TrackMe's Outlier detection will eventually trigger an alert.

Accessing TrackMe's metrics and building your own dashboards and reports

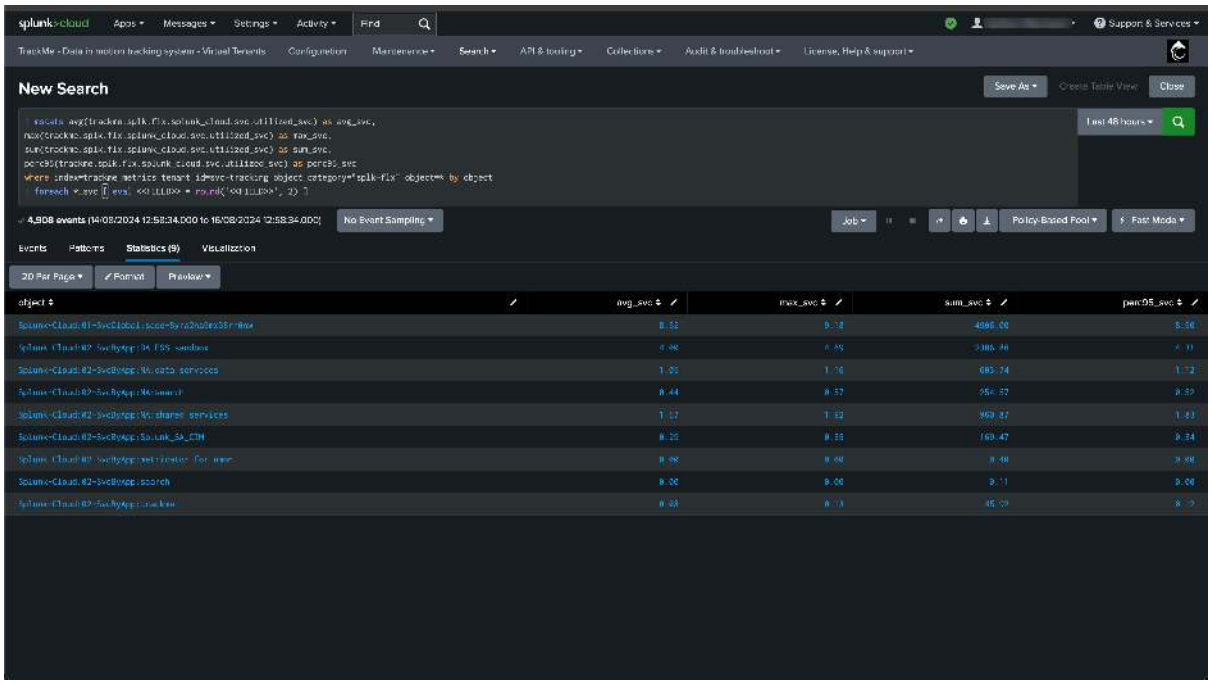
You can easily access to the metrics collected for a given TrackMe Virtual Tenant from the Virtual Tenant home UI:





Query example:

```
| mstats avg(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as avg_svc,
max(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as max_svc,
sum(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as sum_svc,
perc95(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as perc95_svc
where index=trackme_metrics tenant_id=svc-tracking object_category="splk-flx"␣
↪object=* by object
| foreach *_svc [ eval <<FIELD>> = round('<<FIELD>>', 2) ]
```



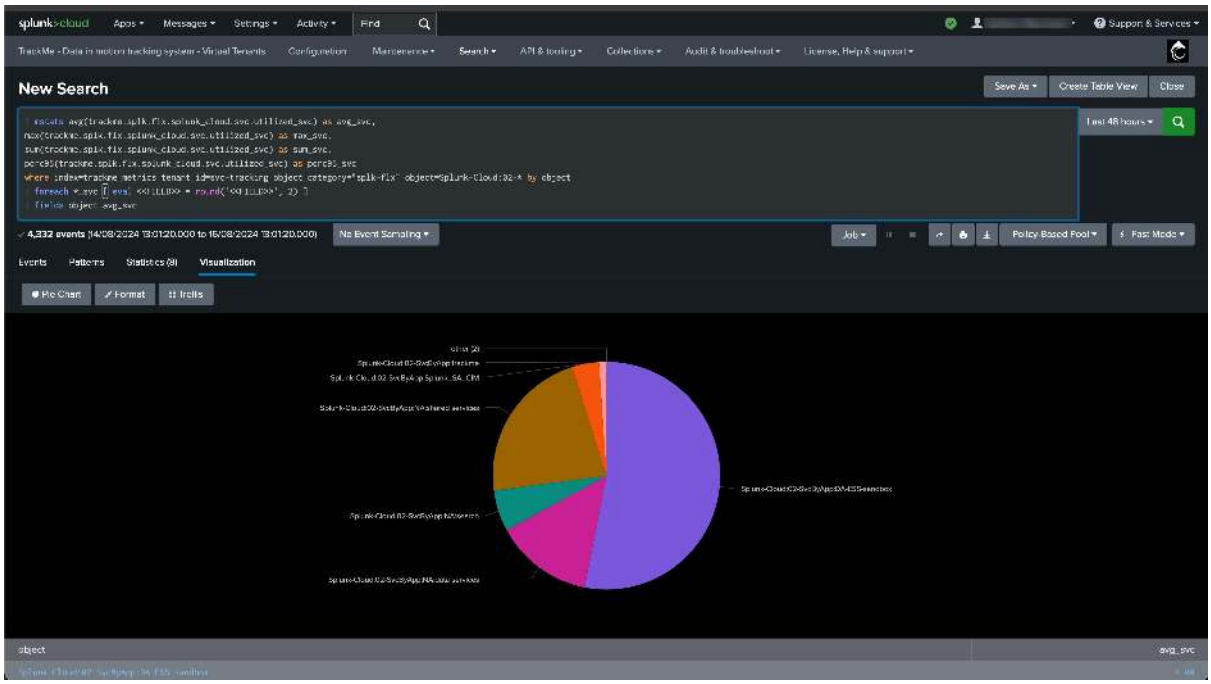
Building a dashboard based on the above, the following would easily show the bigger SVC consumers:

```
| mstats avg(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as avg_svc,
max(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as max_svc,
```

(continues on next page)

(continued from previous page)

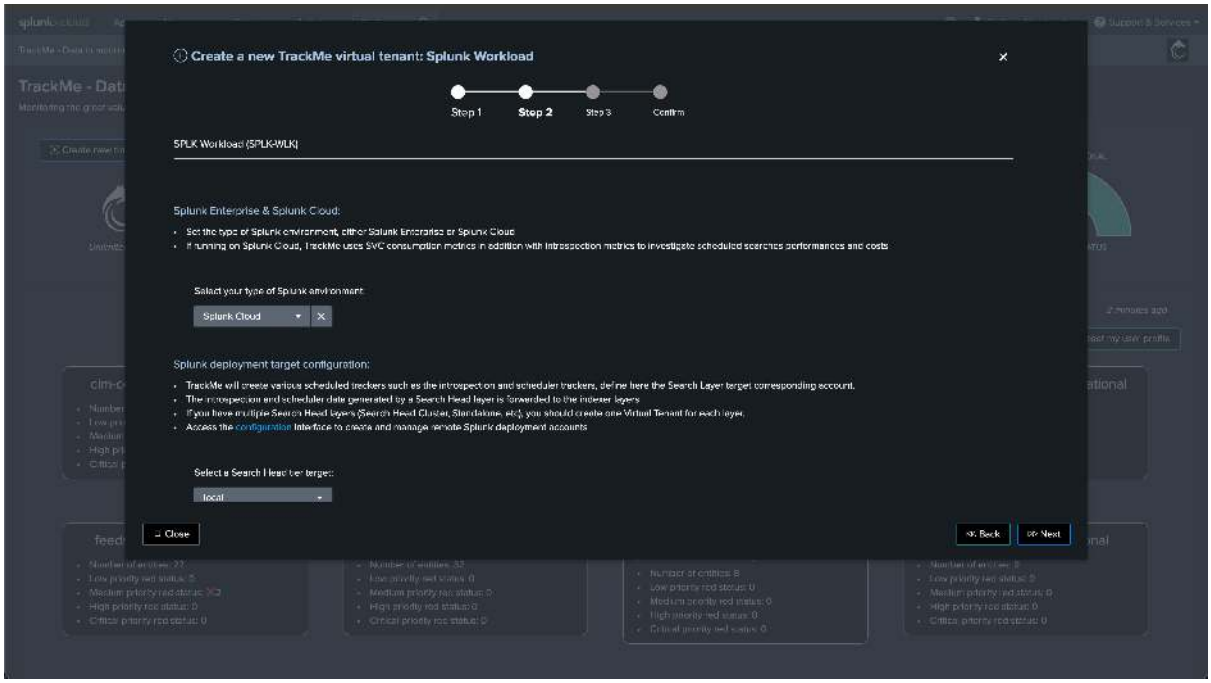
```
sum(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as sum_svc,
perc95(trackme.splk.flx.splunk_cloud.svc.utilized_svc) as perc95_svc
where index=trackme_metrics tenant_id=svc-tracking object_category="splk-flx"␣
↪object=Splunk-Cloud:02-* by object
| foreach *_svc [ eval <<FIELD>> = round('<<FIELD>>', 2) ]
| fields object avg_svc
```



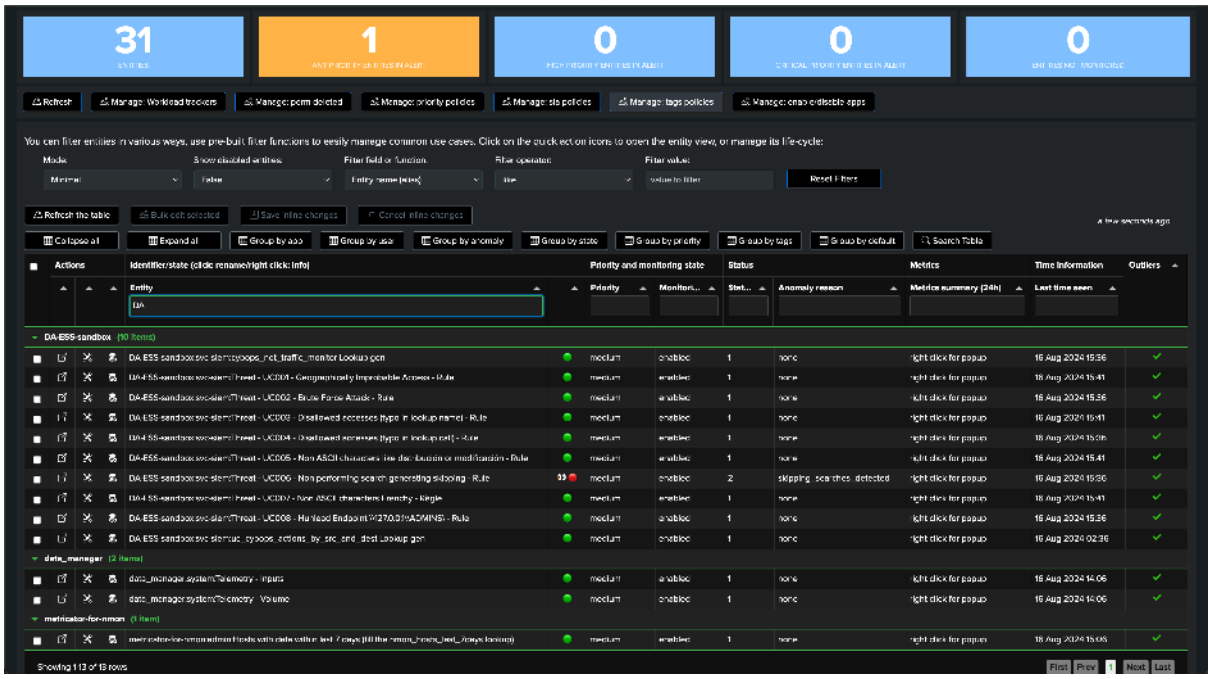
TrackMe Workload component and Splunk Cloud SVC consumption

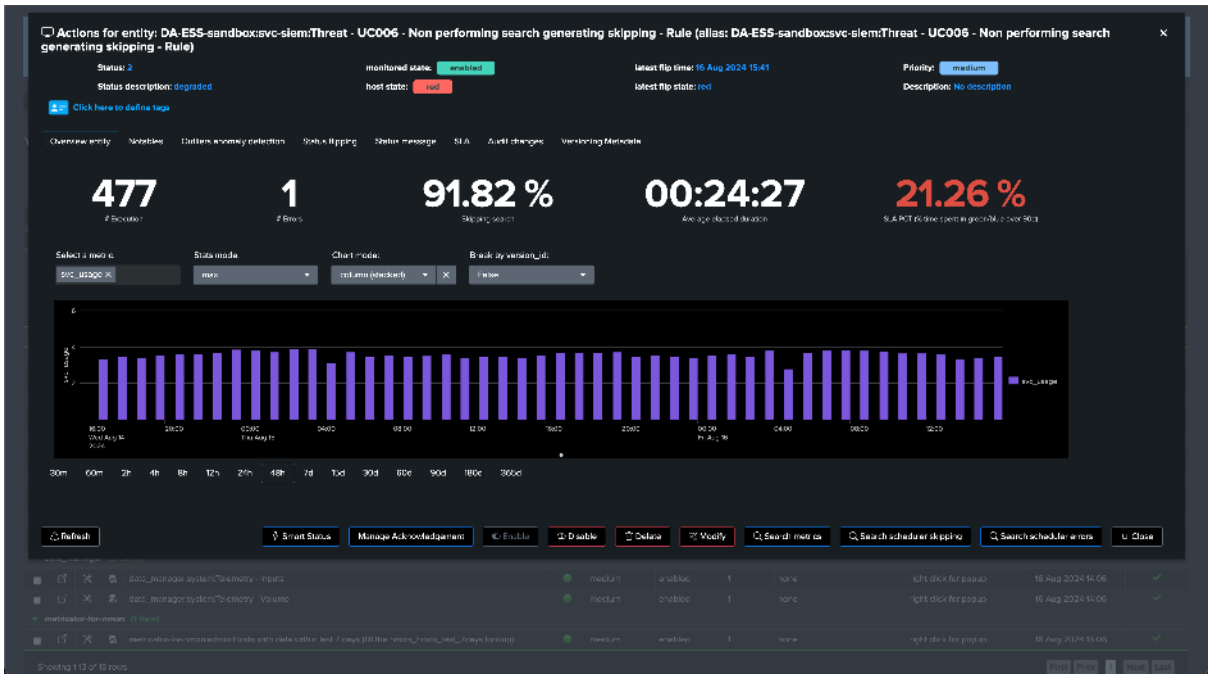
TrackMe's licensed customers can leverage the Workload component, which can also track SVC consumption on a per Splunk scheduled basis.

When you create the Workload Virtual Tenant, ensure to select "Splunk Cloud" as the type of Splunk environment:

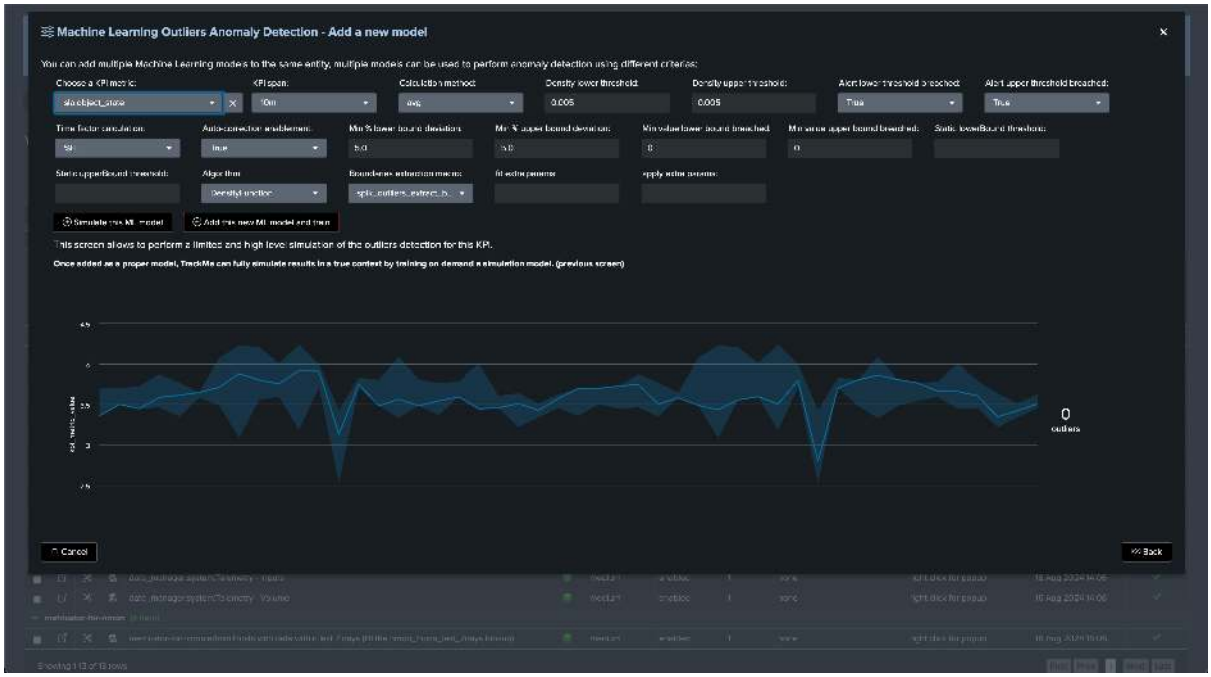


Once configured, the Workload component tracks various aspects of the health of Splunk scheduled, between other KPIs, the Workload component also tracks SVC usage:

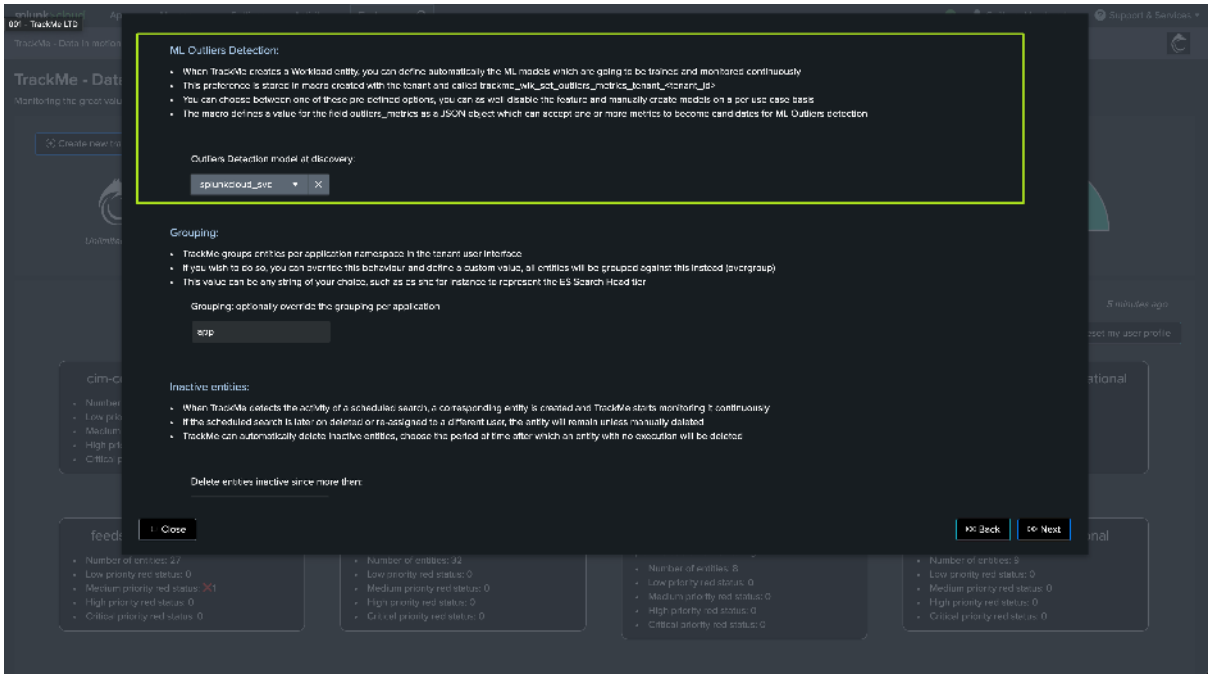




We can for instance add an Outlier models against the SVC consumption:



Note that by default, the Workload component would enable ML against the elapsed run time KPI, you can for instance while creating the tenant, or at a later stage, use SVC consumption instead.



Accessing metrics from the Workload component

The following search query can be used to access the SVC consumption metrics collected by the Workload component:

You need to replace the tenant_id name with the one you have created.

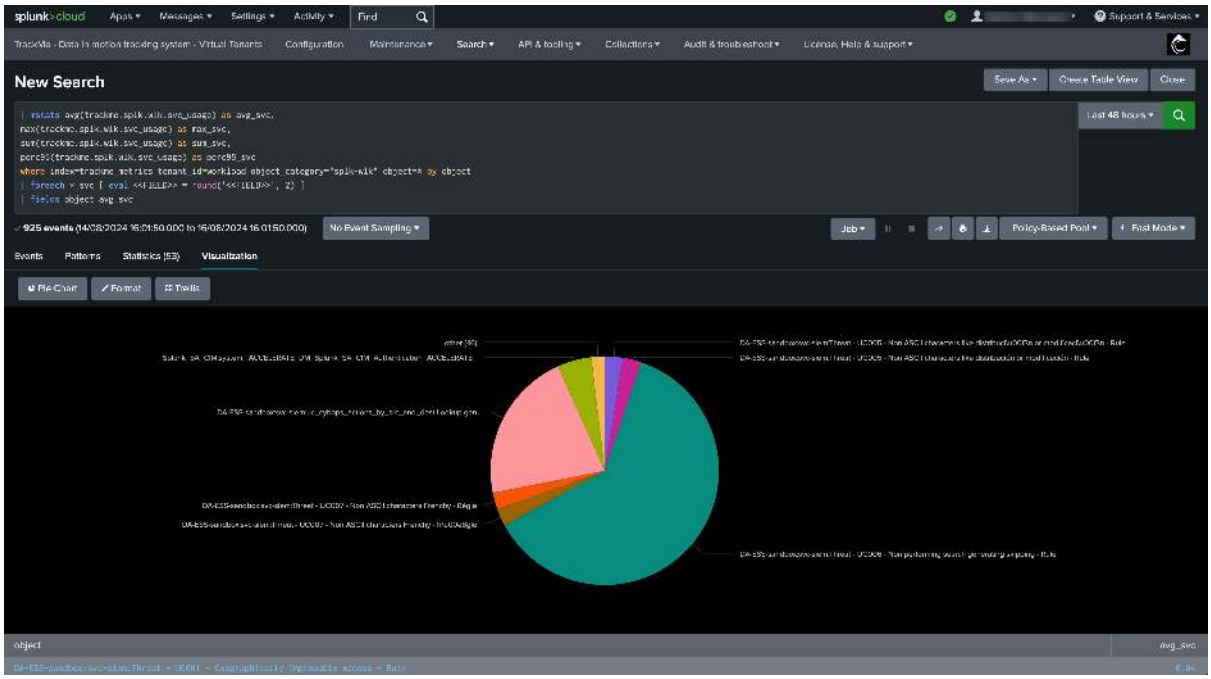
```
| mstats avg(trackme.splk.wlk.svc_usage) as avg_svc,
max(trackme.splk.wlk.svc_usage) as max_svc,
sum(trackme.splk.wlk.svc_usage) as sum_svc,
perc95(trackme.splk.wlk.svc_usage) as perc95_svc
where index=trackme_metrics tenant_id=workload object_category="splk-wlk" object=*␣
```

(continues on next page)

(continued from previous page)

```
↪by object
| foreach *_svc [ eval <<FIELD>> = round('<<FIELD>>', 2) ]
```

Based on the previous example, you can easily build very interesting dashboards on top of this:

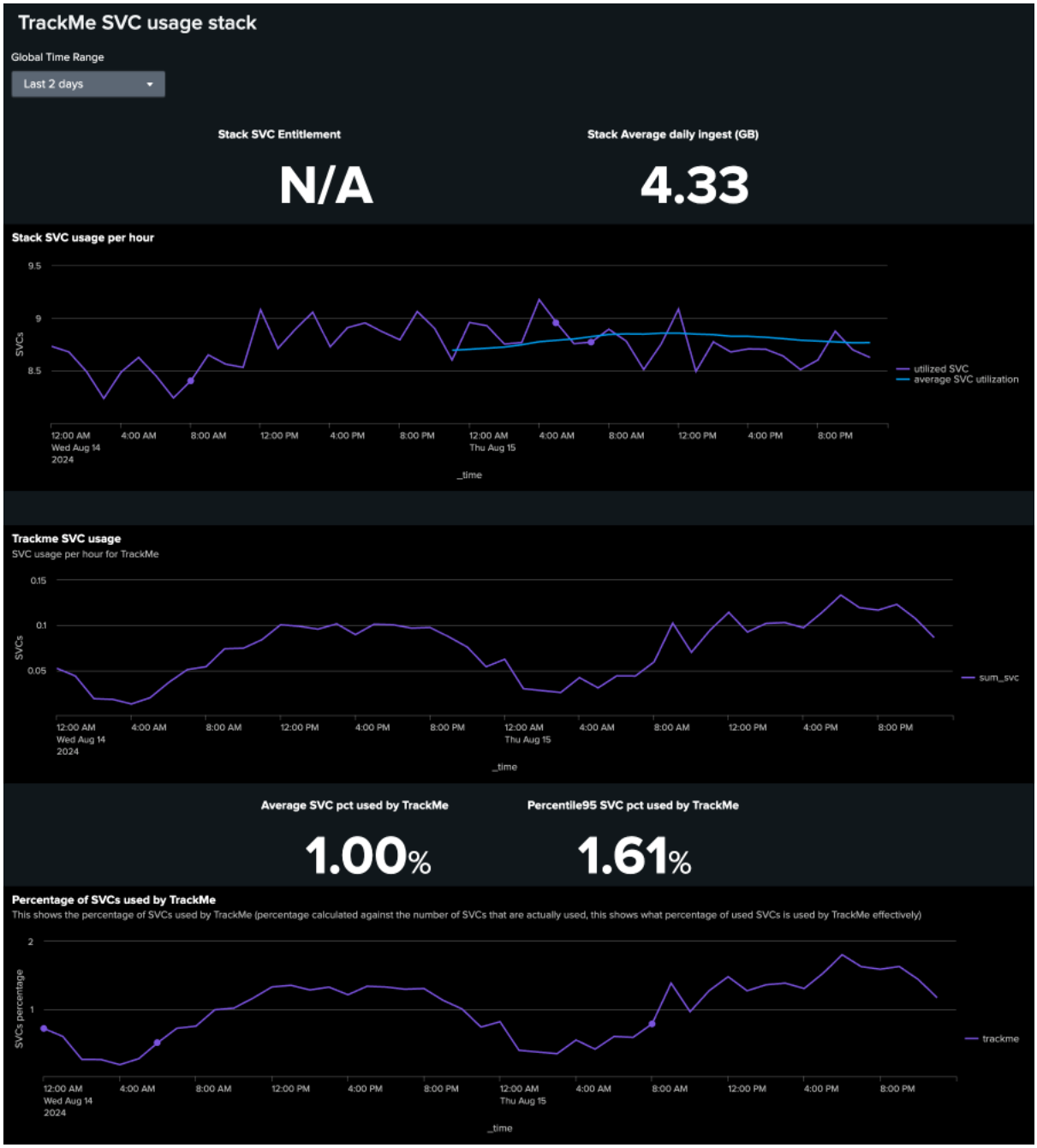


TrackMe out of the box SVC consumption dashboard

While not directly related to the SVC consumption TrackMe tracks through the components explained above, it is worth mentioning that TrackMe also comes with an out of the box dashboard which is designed to audit the SVC usage that is related to TrackMe itself.

You can find this dashboard in the menu "Audit & Troubleshoot" / "TrackMe SVC usage stack":

Note: the stack SVC entitlement comes as N/A because it is a dev Cloud stack which doesn't come with a normal limit, unlike a proper Cloud stack.



9.3.2 Monitor Splunk Workload with TrackMe's Workload component

Monitoring Splunk Search Head Clusters with TrackMe Flex Objects

- This tutorial demonstrates the monitoring of **Splunk Workload** with the TrackMe Workload component.
- The Workload component is a component restricted to licensed customers. Please contact Splunk Sales for more information.
- Using these steps will enable TrackMe to continuously monitor Splunk scheduled activity, detect execution anomalies and changes in behavior, as well as performing versioning of searches definition.
- Monitoring the Splunk scheduling workload is a critical part for Splunk. For instance, detecting when your SIEM correlation searches are affected by any issue is as crucial as detecting when feeds have stopped feeding your use cases.

- With TrackMe's remote search capabilities, you can monitor as many Search Head tiers as you need from a single pane of glass.

Hint

Evolutions in TrackMe 2.1.28

- Since TrackMe 2.1.28, you can now control the list of parameters that are taken into account to calculate the version_id hash.
- To do so, go in Configure / splk-general and update the `splk_general_workload_version_id_keys` parameter.
- The default is to take into account the following parameters: search,dispatch.earliest,dispatch.latest,description,cron_schedule

Requirements

In most cases, you will want to target a different Search Head tier than the one where the TrackMe instance is running.

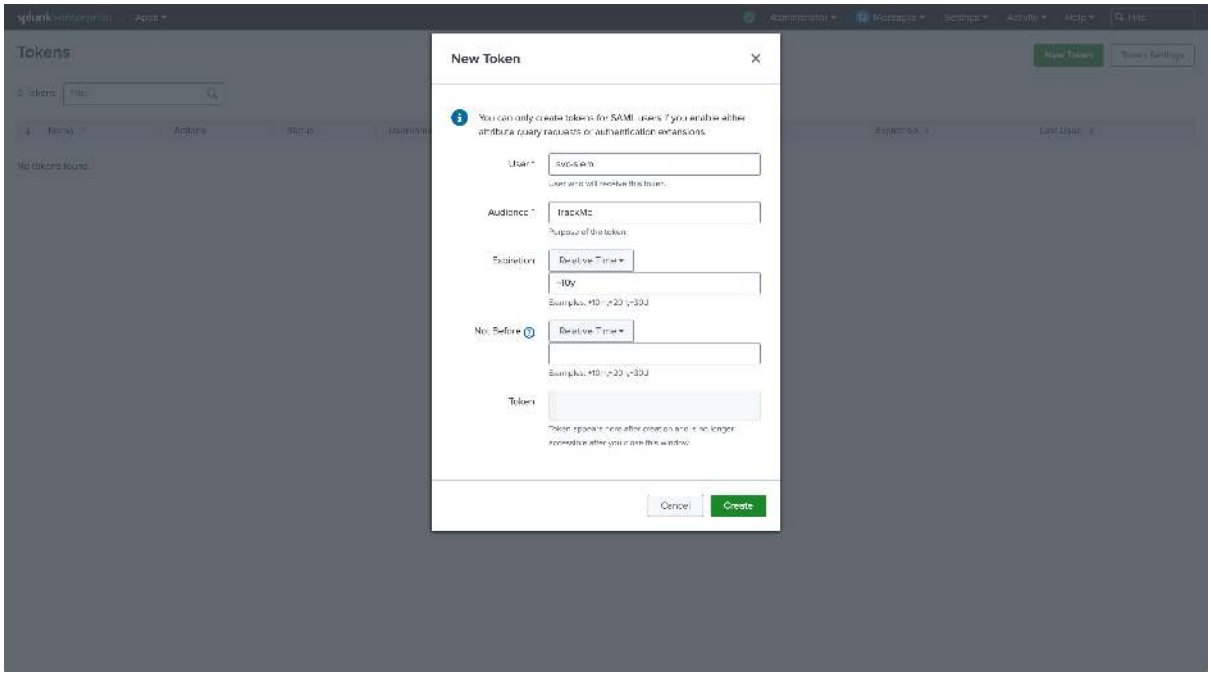
For example, if you are running **Splunk Enterprise Security**, this is normally in a different Search Tier. We will therefore start by creating a Splunk Remote Account in TrackMe to access the Search Head tier, programmatically speaking.

Step 1: Create a Splunk Remote Deployment Account for the SHC

The first step is to create a Splunk Remote Deployment Account for the Cluster Manager. For more information about TrackMe Remote Search capabilities and configuration:

Remote Splunk deployments

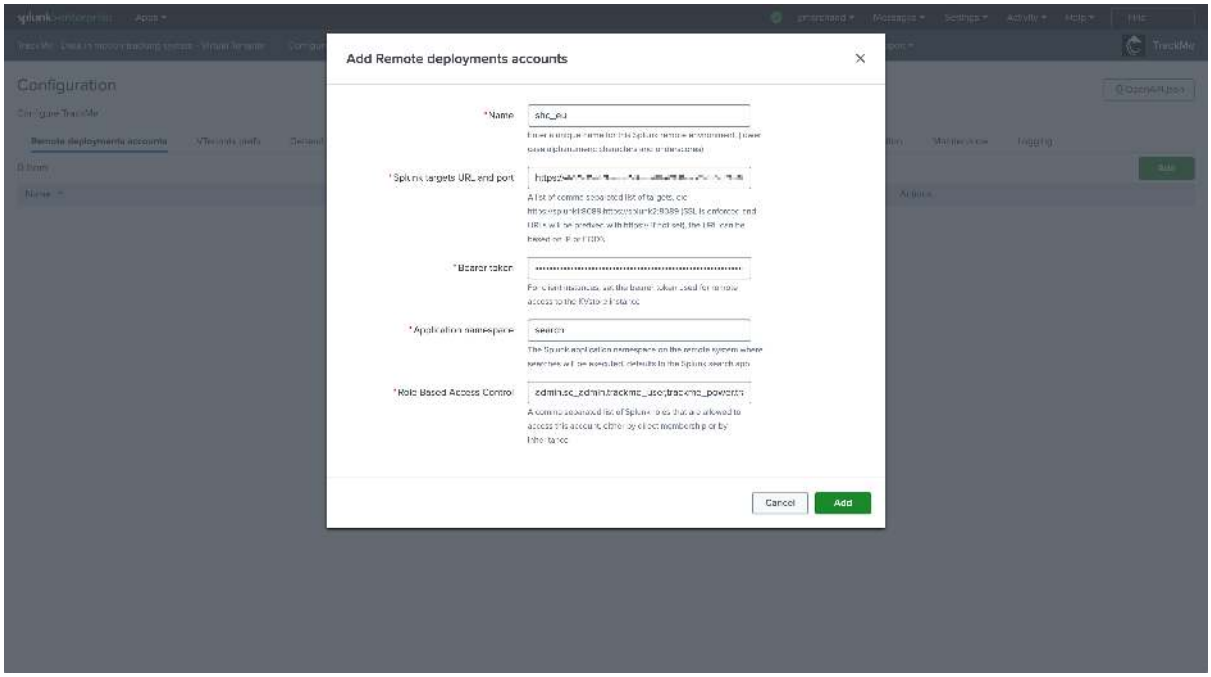
On the Search Head Cluster, create a new Splunk bearer token for the TrackMe Remote Deployment Account:

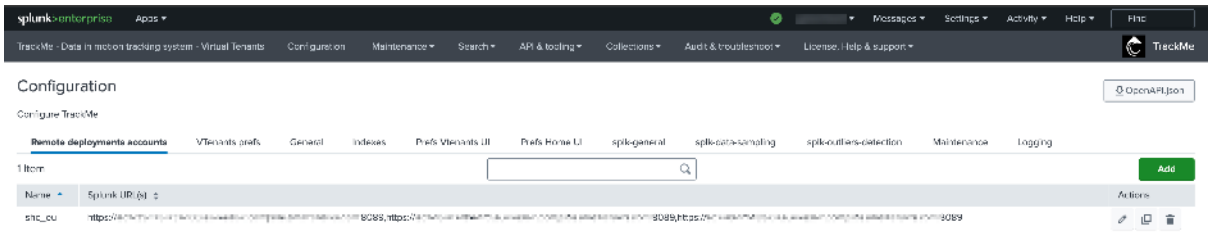


In TrackMe, click on Configure / Remote Deployment Accounts and add a new account:

- If running a Search Head Cluster, you can specify each of the SHC members in a comma-separated list or use a load balancer URL.

- If running a standalone Search Head, you can specify the Splunk API URL of the Search Head.
- If multiple endpoints are specified, TrackMe automatically dispatches searches randomly amongst available and responding members (it validates connectivity and authentication)





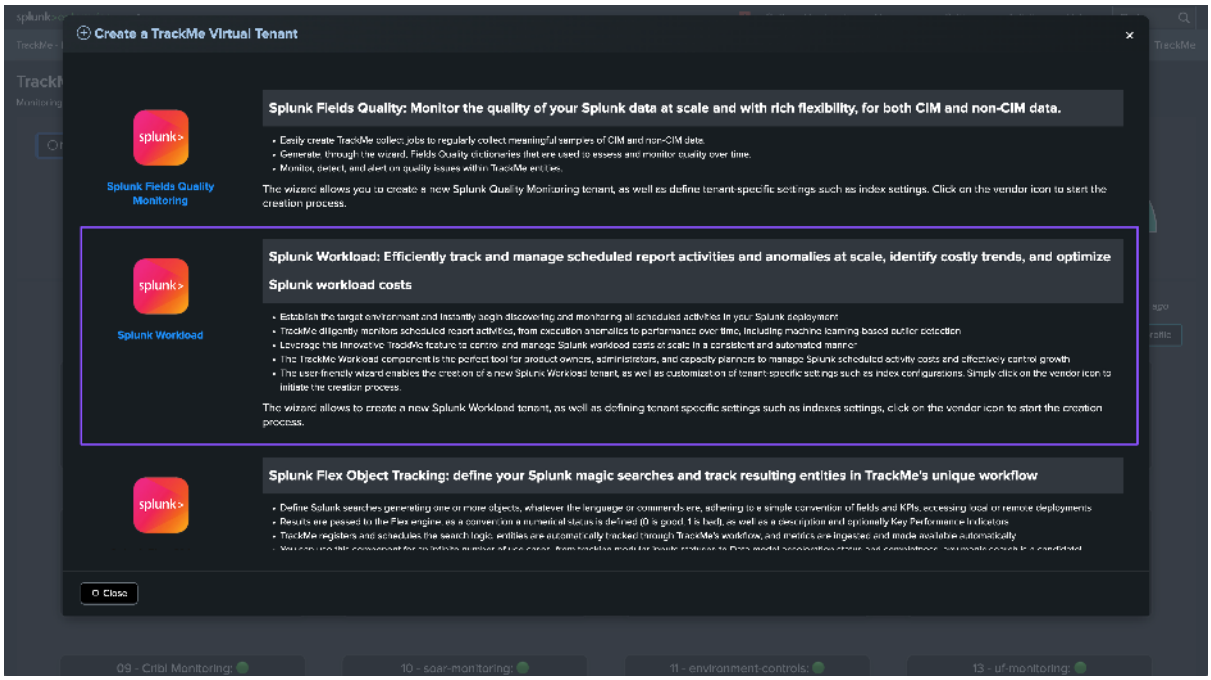
Managing multiple Search Head tiers

- If you have multiple Search Head tiers to be monitored, you can create a Remote Deployment Account for each tier.
- You will then be able to manage and monitor as many Search Head tiers as you need from a single pane of glass in TrackMe.

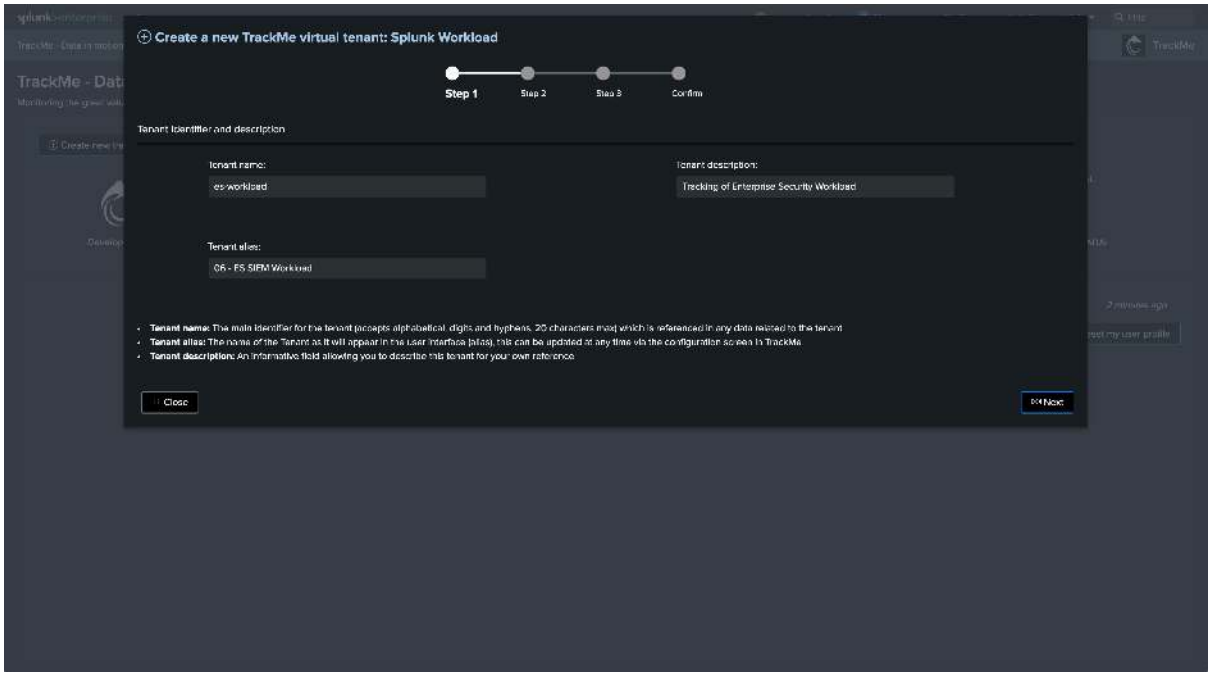
Step 2: Create a Workload tenant and use the wizard to create trackers automatically

Access the tenant creation wizard and select the Workload component

Click on "New tracking tenant":



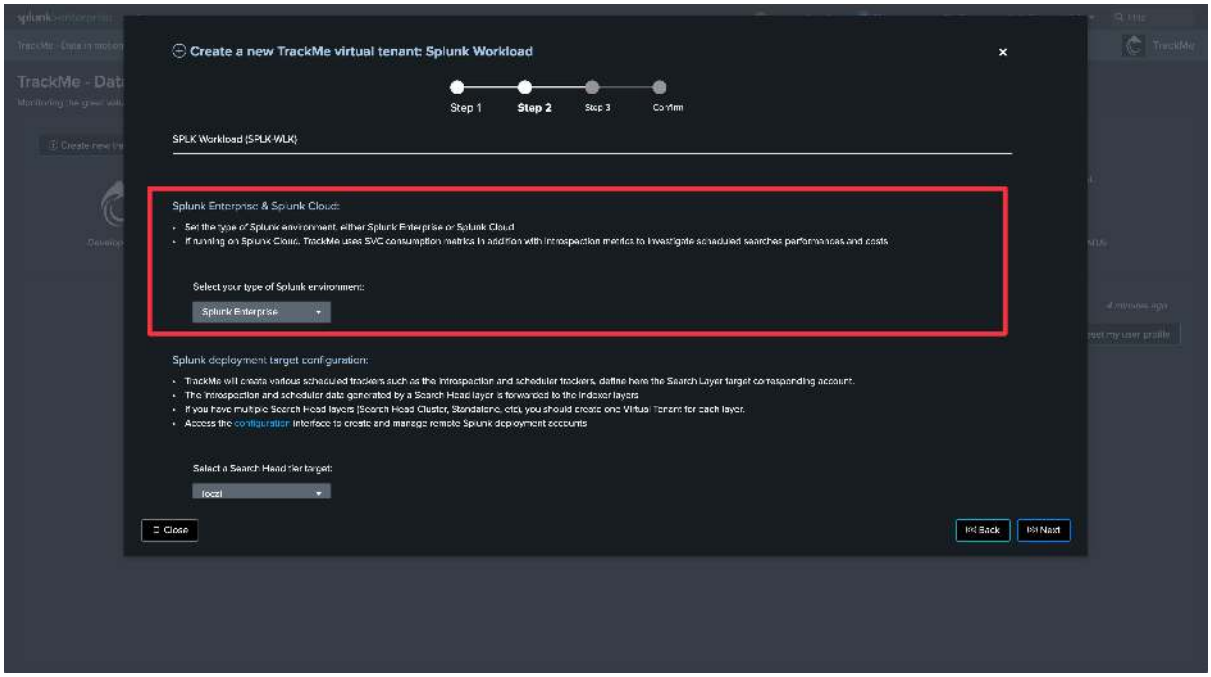
Choose the tenant_id and other main information



Define the type of Splunk deployment (Enterprise/Cloud)

Splunk Enterprise vs Splunk Cloud

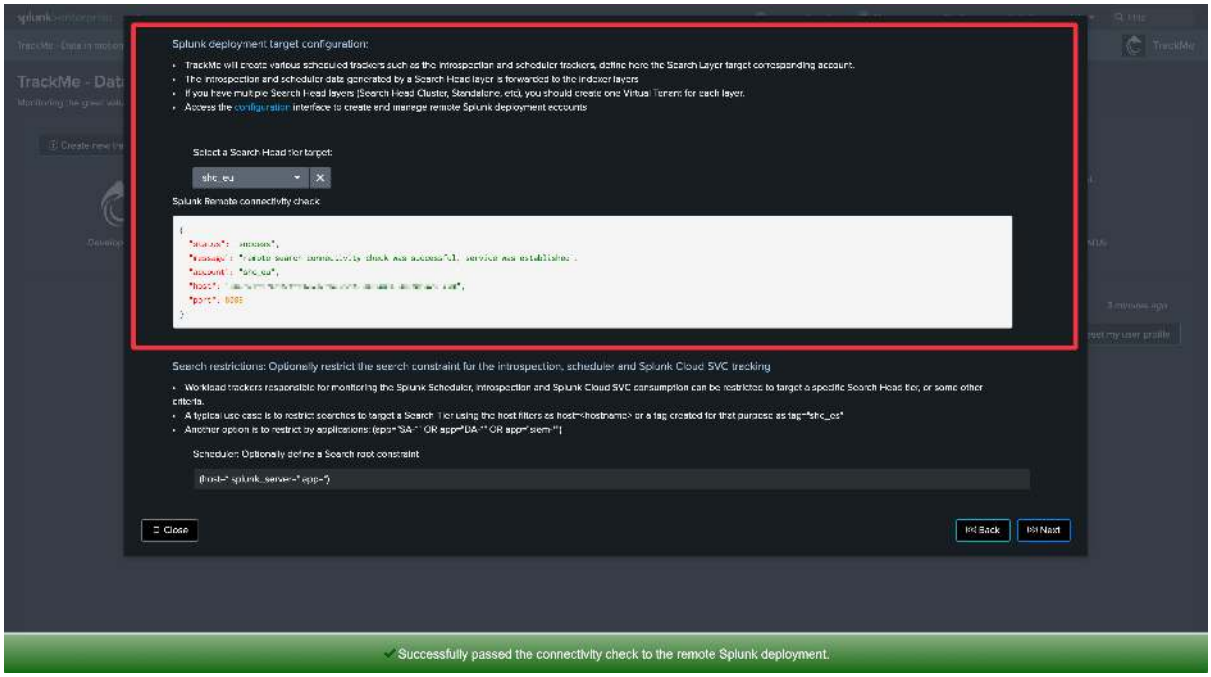
- The main difference here is that if Splunk Cloud is selected, TrackMe will create necessary trackers to monitor the SVC consumption of monitored scheduled searches.
- Select Splunk Cloud if the environment being monitored is a Splunk Cloud environment.



Select the Search Head tier target

Search Head target

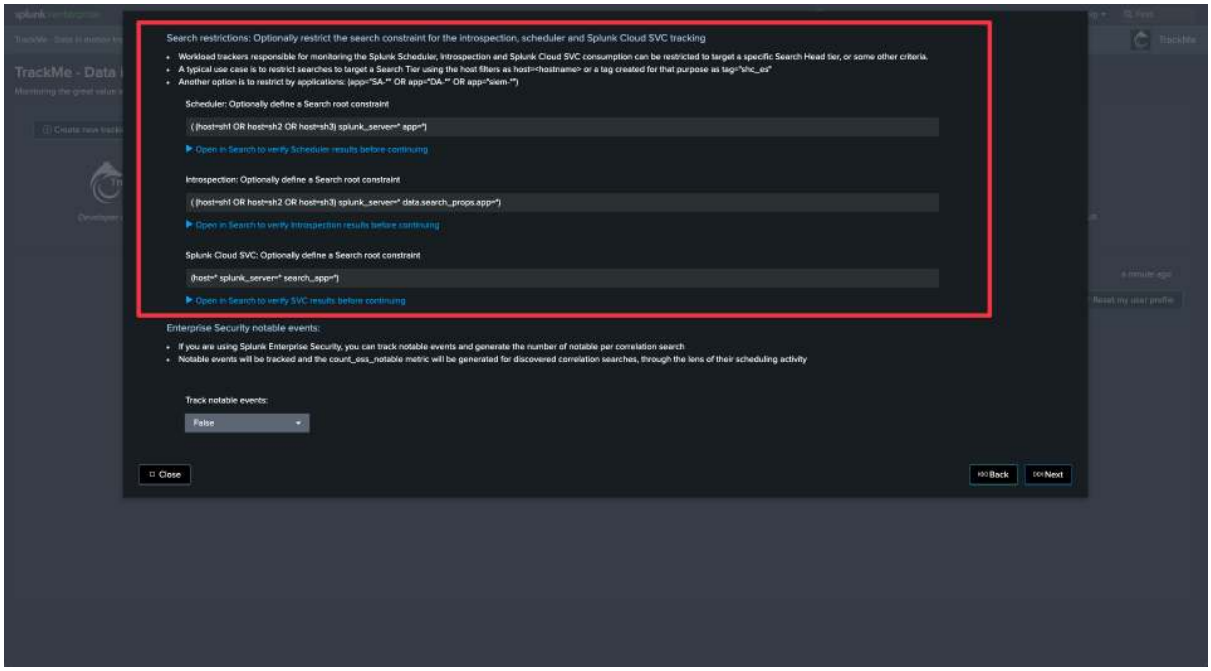
- Although some of the search logics TrackMe implements could indeed be run on the TrackMe instance itself if both can search the same indexers, there are much deeper API-related logics which come into account for the Workload components.
- Therefore, it is important to select the right Splunk Remote Deployment Account targeting the desired Search Head tier.



Define Search Restrictions

Search Restrictions

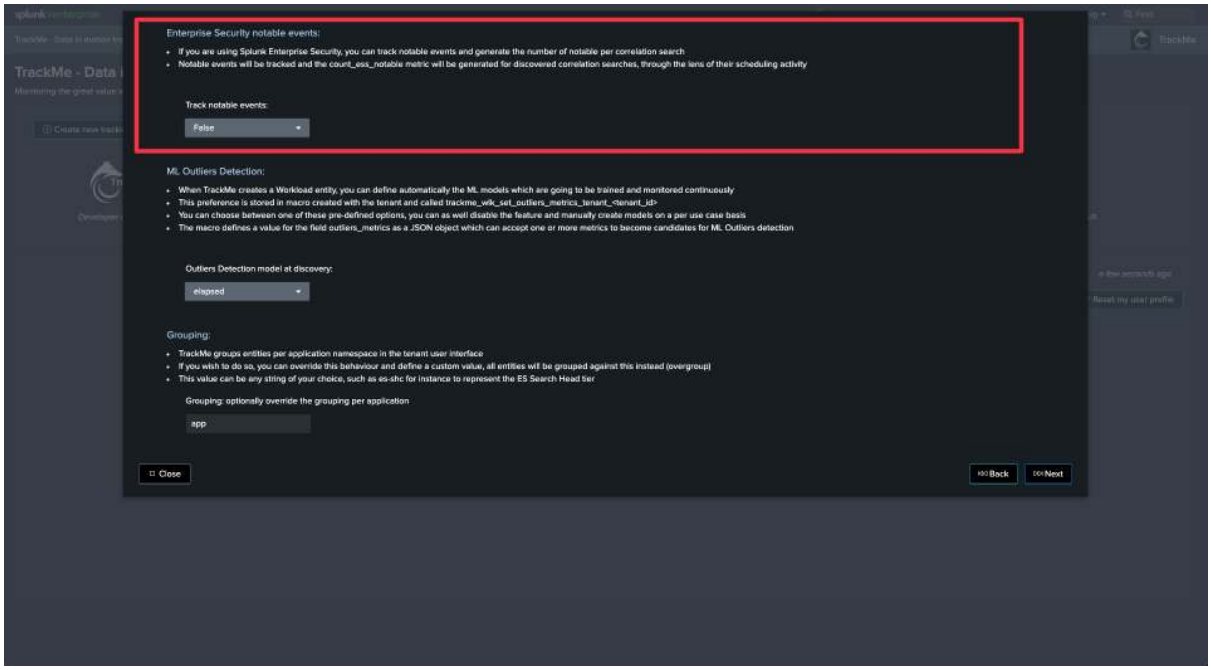
- For each of the main categories of activity, you need to define and/or restrict the Search Heads to be considered for monitoring.
- This is important to avoid monitoring searches that are out of the scope of the Search Head you target, but whose underlying activity can be accessed from the TrackMe instance.
- In the example below, we restrict to a given Search Tier using the filter `(host=sh1 OR host=sh2 OR host=sh3)`



Tracking Enterprise Security Notables

Tracking Notable events

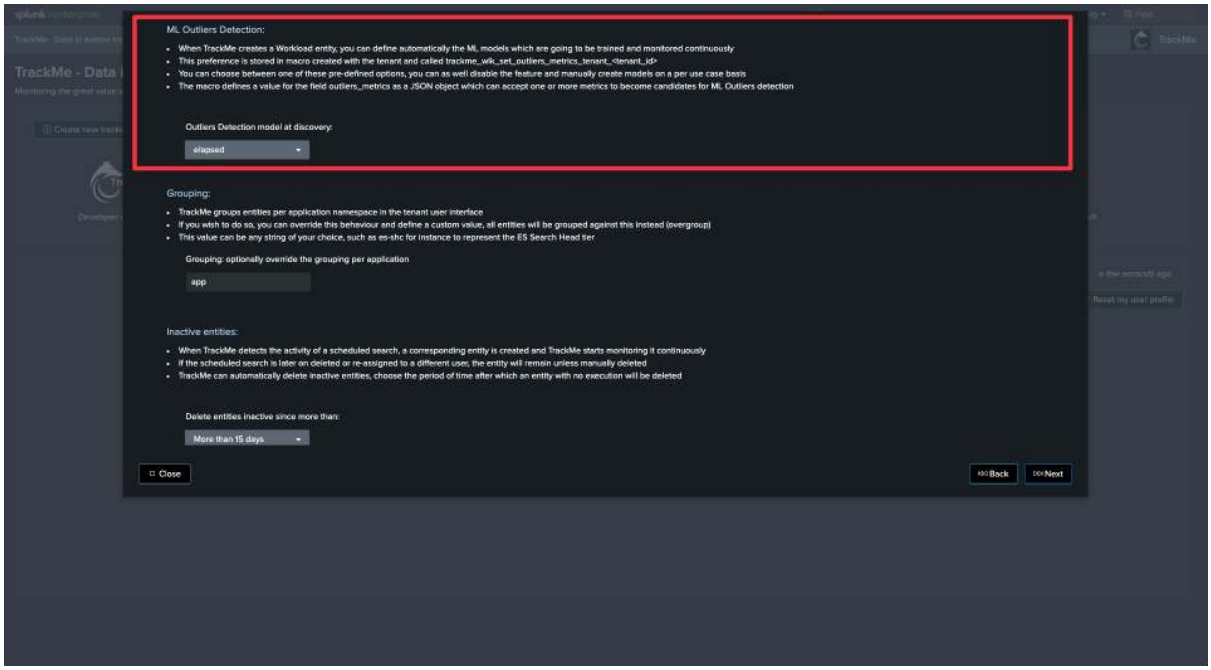
- The Workload component can track notable events generated by correlation search, and turns this into a TrackMe metric.
- You can then use TrackMe to review the number of notable events created by your correlation searches.



ML Outliers Detection

Outliers detection and Workload

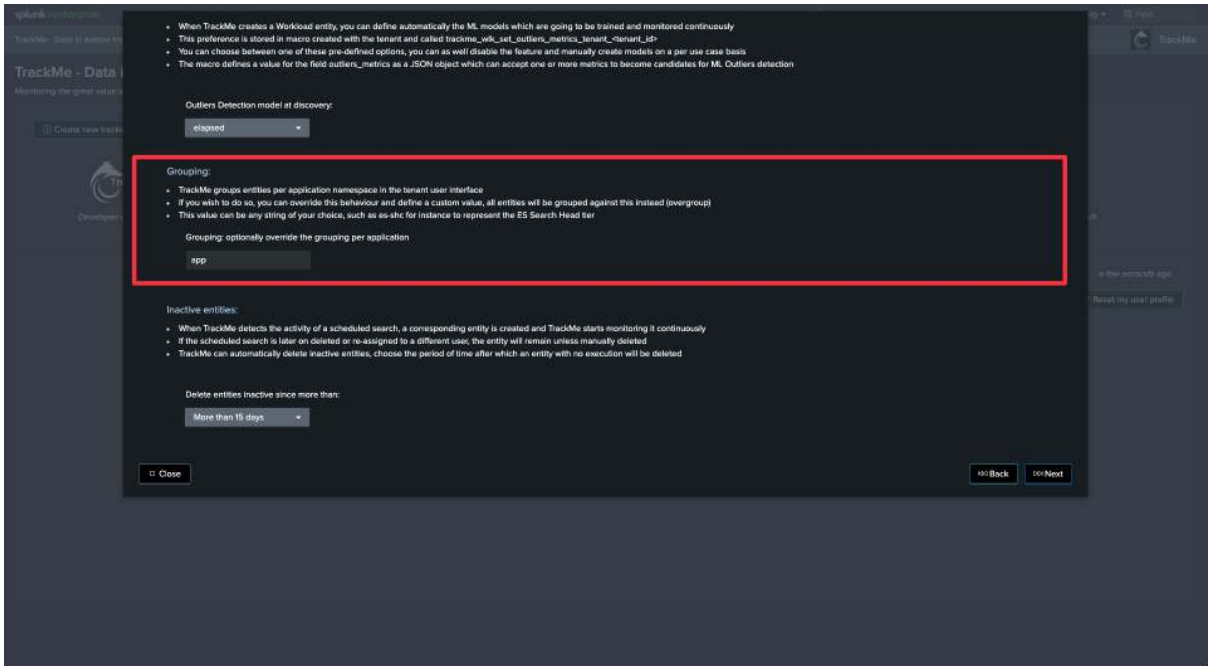
- TrackMe can use ML Outliers detection to automatically detect abnormal trends for a given metric.
- In the context of the Workload, TrackMe by default generates and trains models against the run time in seconds of the searches, so it can detect an abnormal trend in the time taken by searches to be executed.
- You can customize and/or disable this behavior.



Grouping

Workload grouping

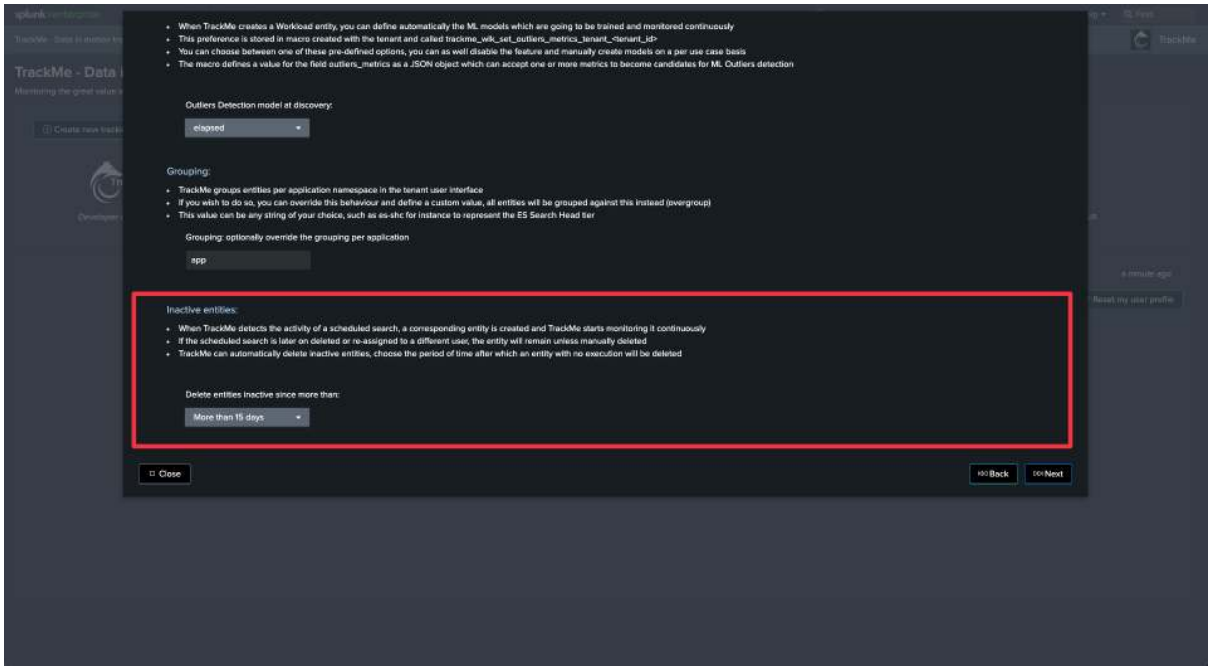
- In the Workload component, the default behavior is to group TrackMe entities per Splunk application namespace.
- This means you should not monitor multiple Search Head tiers within the same tenant (entities would be mixed all together).
- If you wish to target multiple Search Head tiers in the same tenant, you need to use the grouping to group against a custom pattern.
- You can also simply use this setting to force TrackMe not to group per application namespace.



Inactive entities

Managing entities without any activity

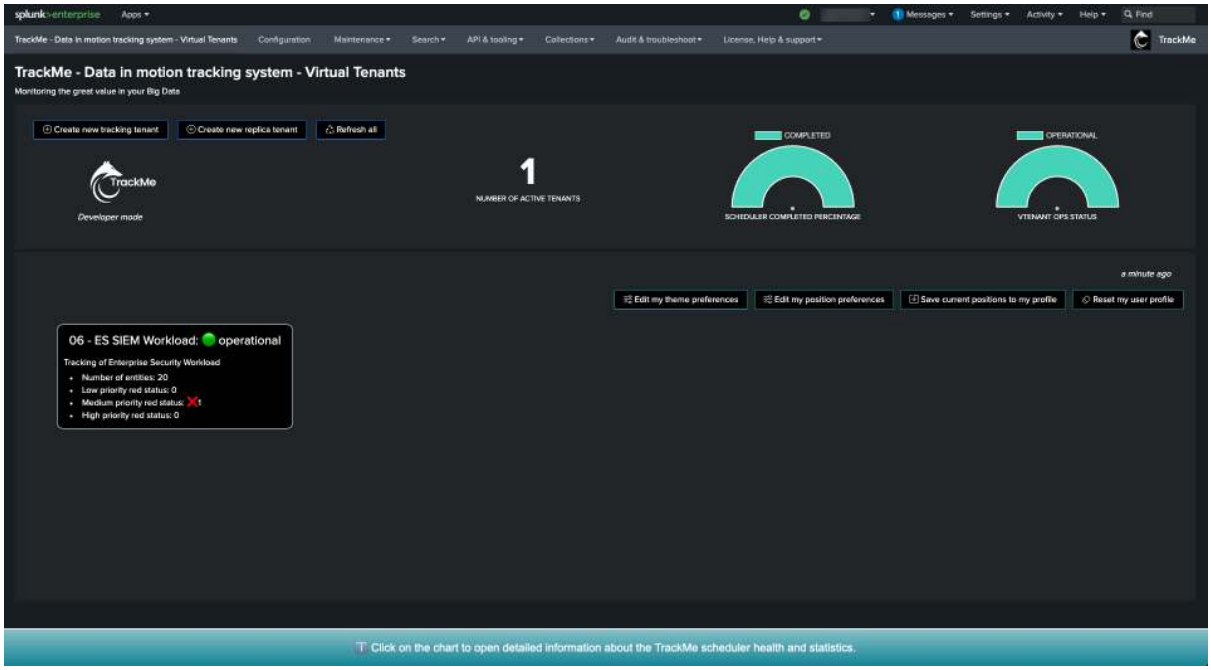
- When a search schedule is disabled, it will become inactive after some time.
- This setting instructs what TrackMe should do with these entities. The default behavior is to remove entities which have not been active for more than 15 days.

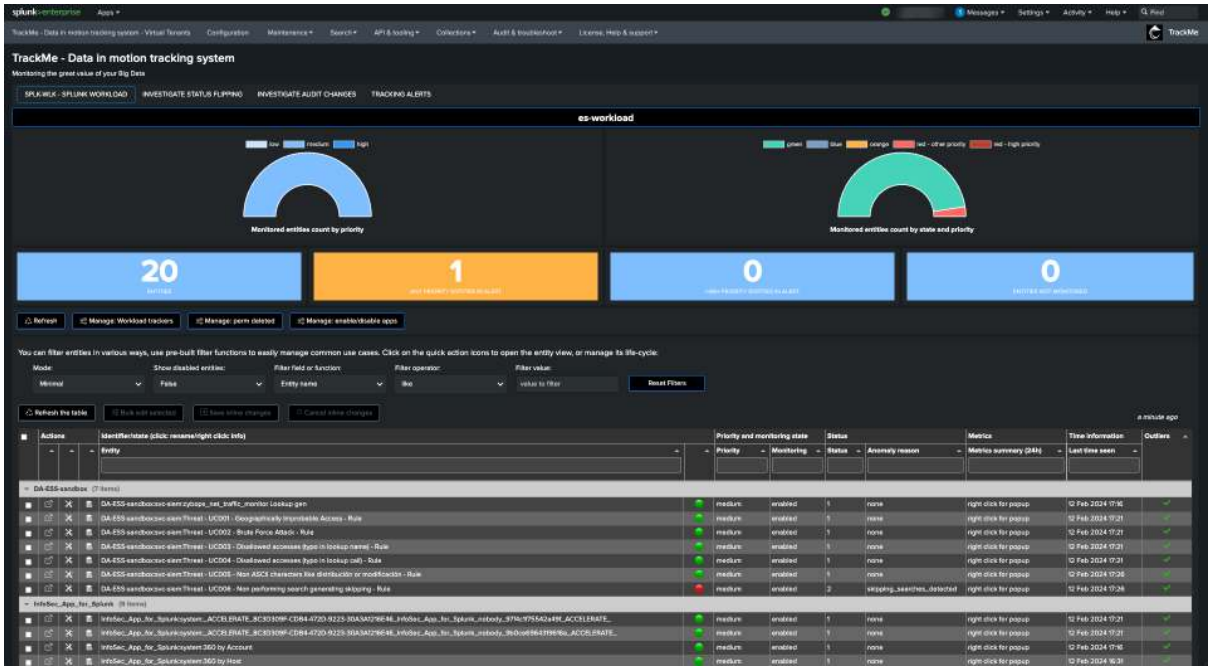


Validate the tenant creation and start

After the tenant is created, wait up to 5 minutes before seeing the first entities in TrackMe

- Immediately after the tenant is created, TrackMe starts to orchestrate the Workload detection logic.
- This can take up to 5 minutes before you can start seeing entities created in the Workload component.

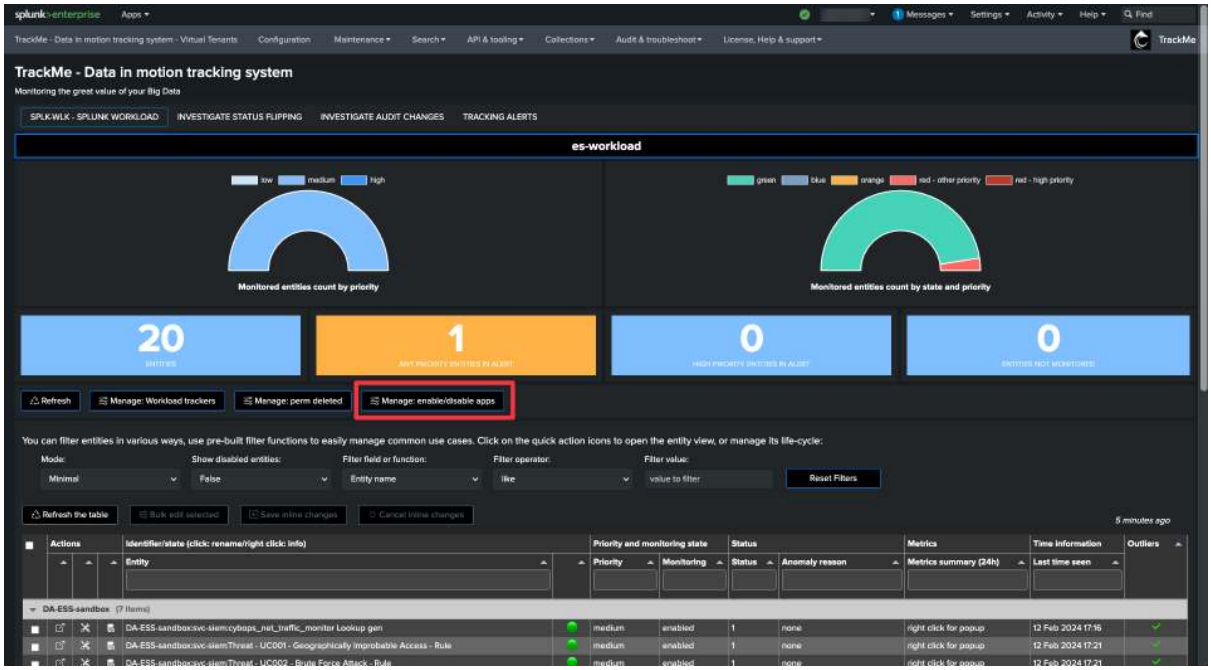


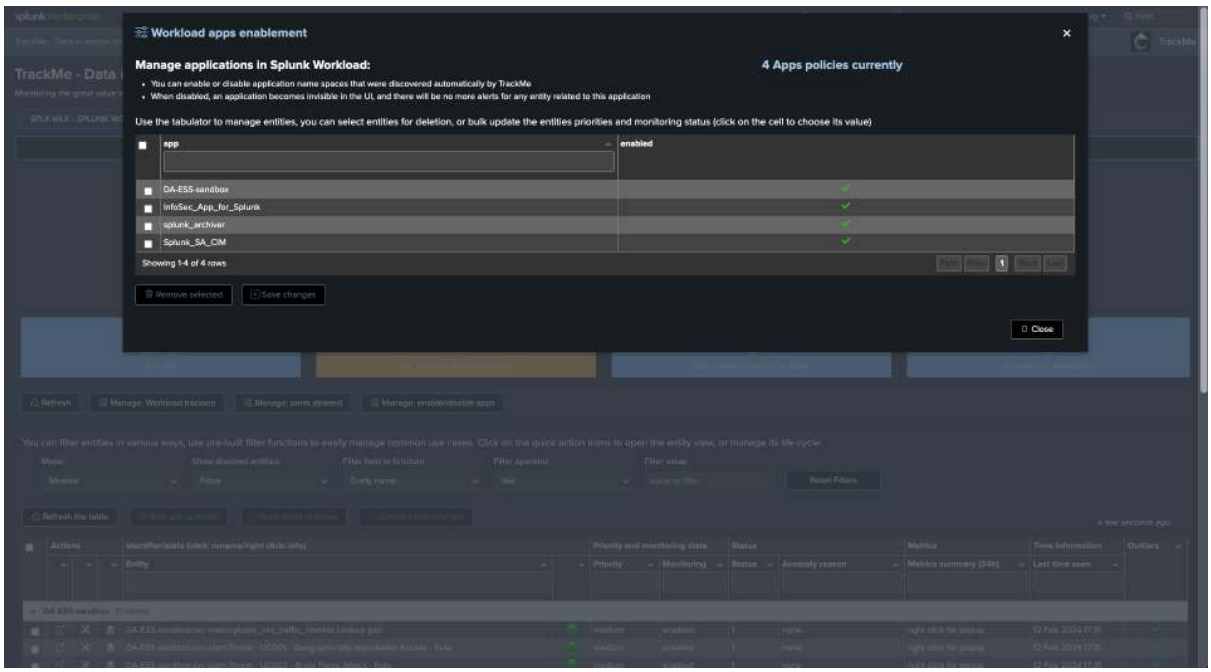


Step 3: Start using the Workload component, detect and investigate scheduling issues

Ignoring specific Splunk application namespaces

- You can ignore specific Splunk application namespaces from the Workload component.
- This can be useful to remove Splunk out-of-the-box applications which are not relevant for your monitoring.
- Click on “Manage: enable/disable apps” to access the configuration screen





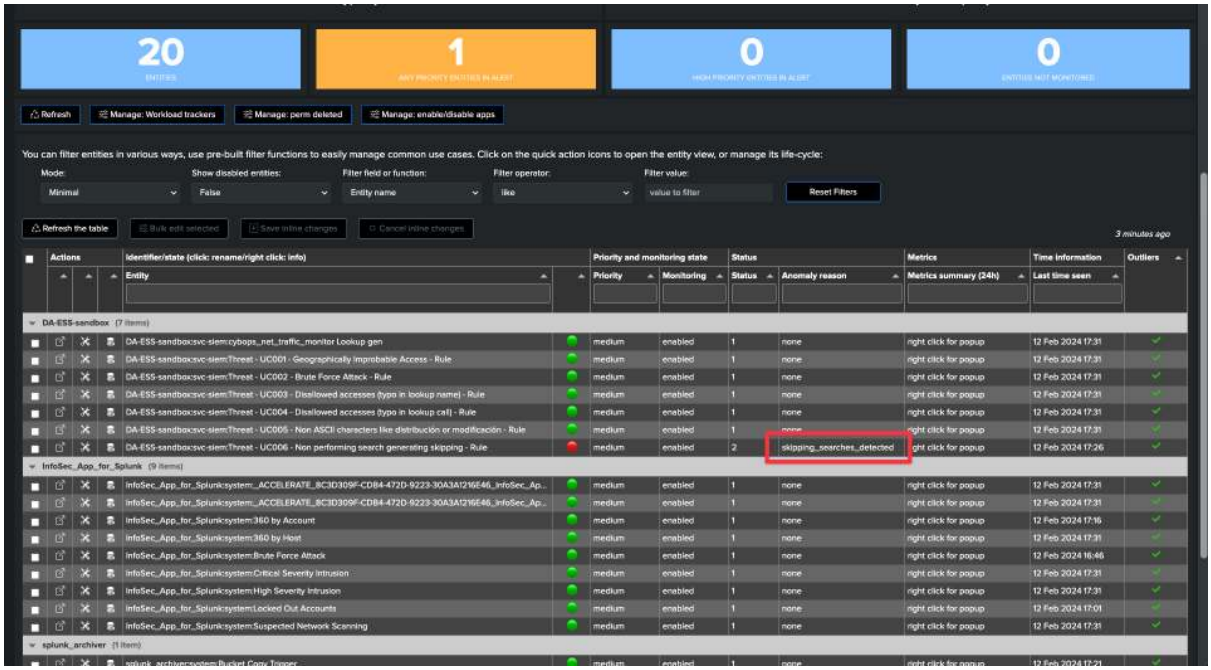
Use case: detecting skipping searches

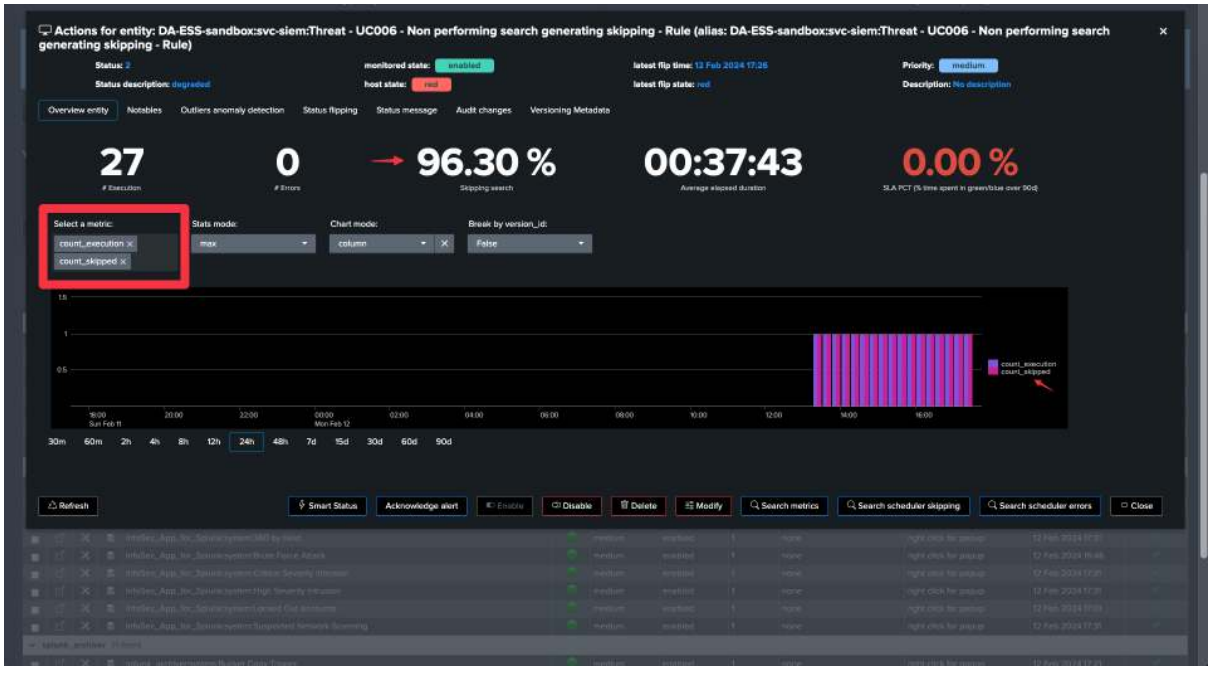
What are skipping searches and how these affect Splunk workload

- Skipping searches happen when a given search takes too long to be executed and is then skipped by the scheduler.

- The current cron schedule defines how often the search is executed and indirectly defines how long in seconds the search can run before a new search should normally be executed.
- If Splunk encounters that a search is already running at this stage, its execution will be skipped.

Depending on the percentage of skipping searches, TrackMe sets the entities status to orange or red:



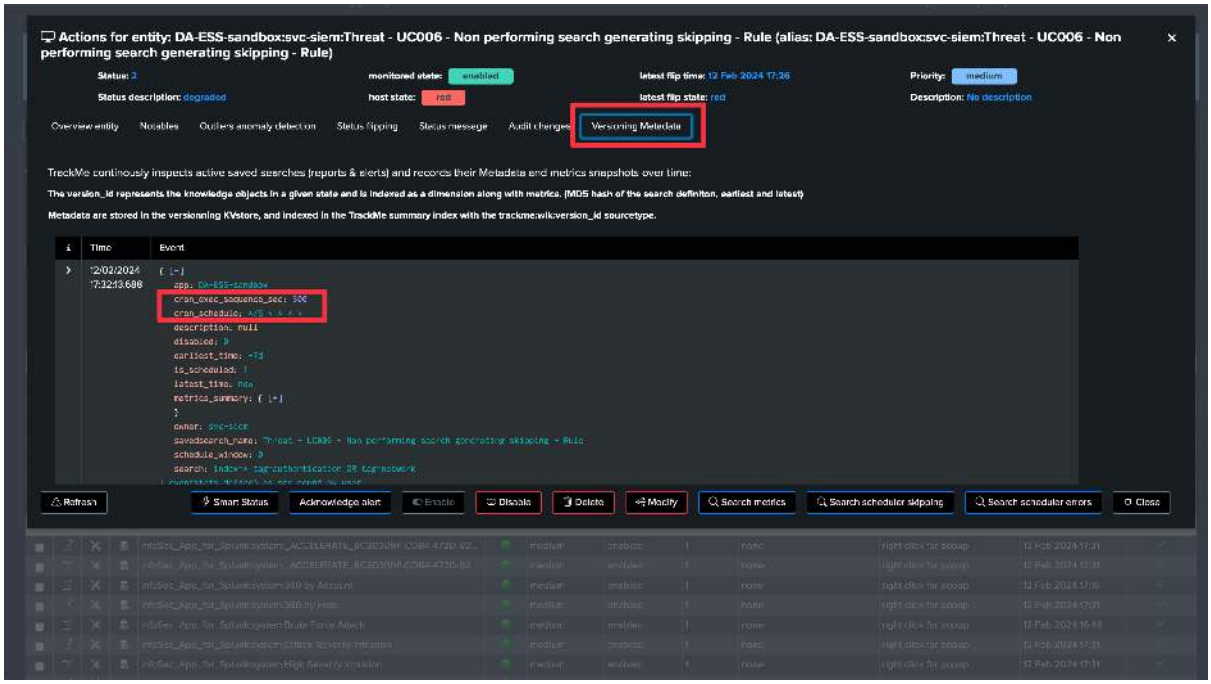


TrackMe's versioning, cron schedule and cron sequence

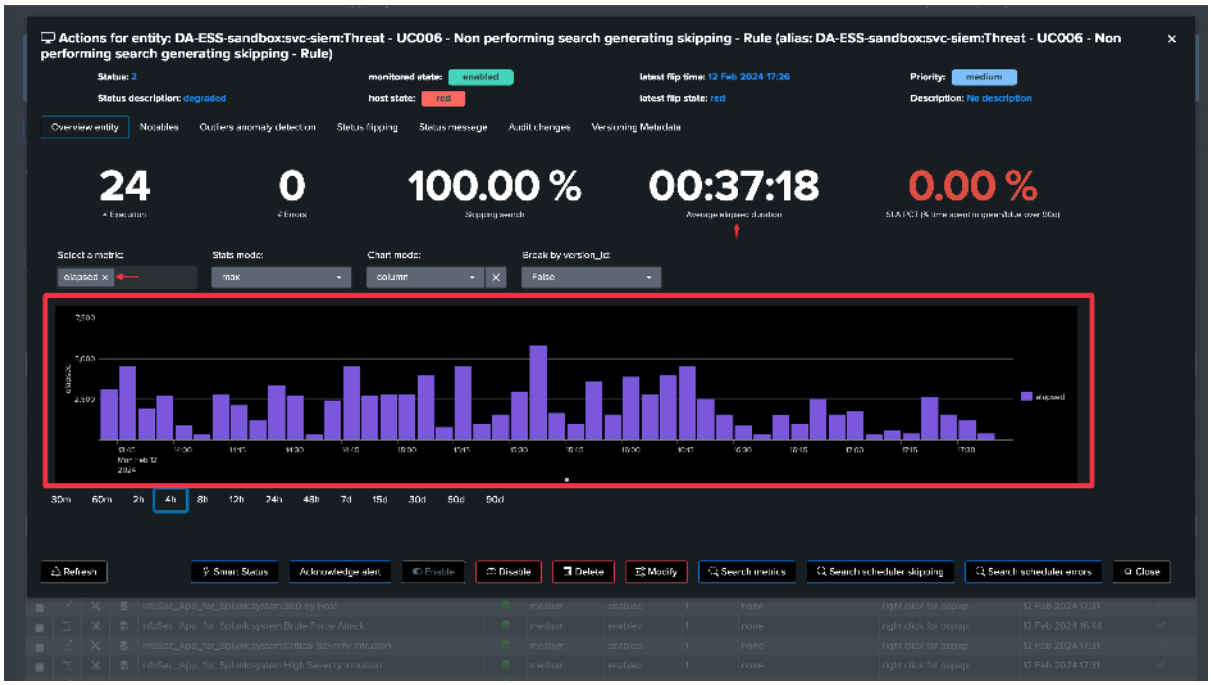
- TrackMe performs versioning of all monitored searches and detects when a search is updated.
- It also extracts the cron schedule and uses the `croniter` Python library to calculate the maximum run time sequence in seconds of the search.
- You can use this information to easily understand, without quitting TrackMe, the reasons for skipping

searches.

In this example, we can see that the cron schedule is set to run every 5 minutes, and that the maximum run time sequence is 300 seconds:



However, The run time execution is well over acceptable values, and the search is skipped:



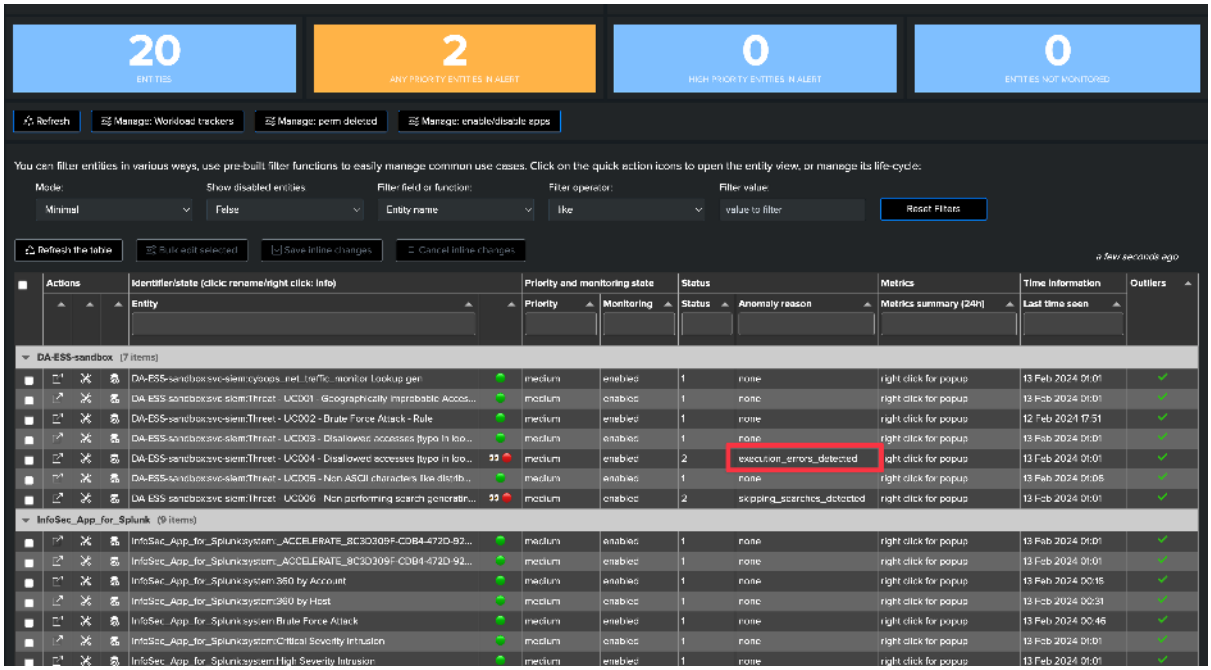
This search is clearly encountering quality issues, and needs to be reviewed and redesigned to perform properly or have a more adapted cron schedule.

Use case: detecting searches suffering from execution anomalies

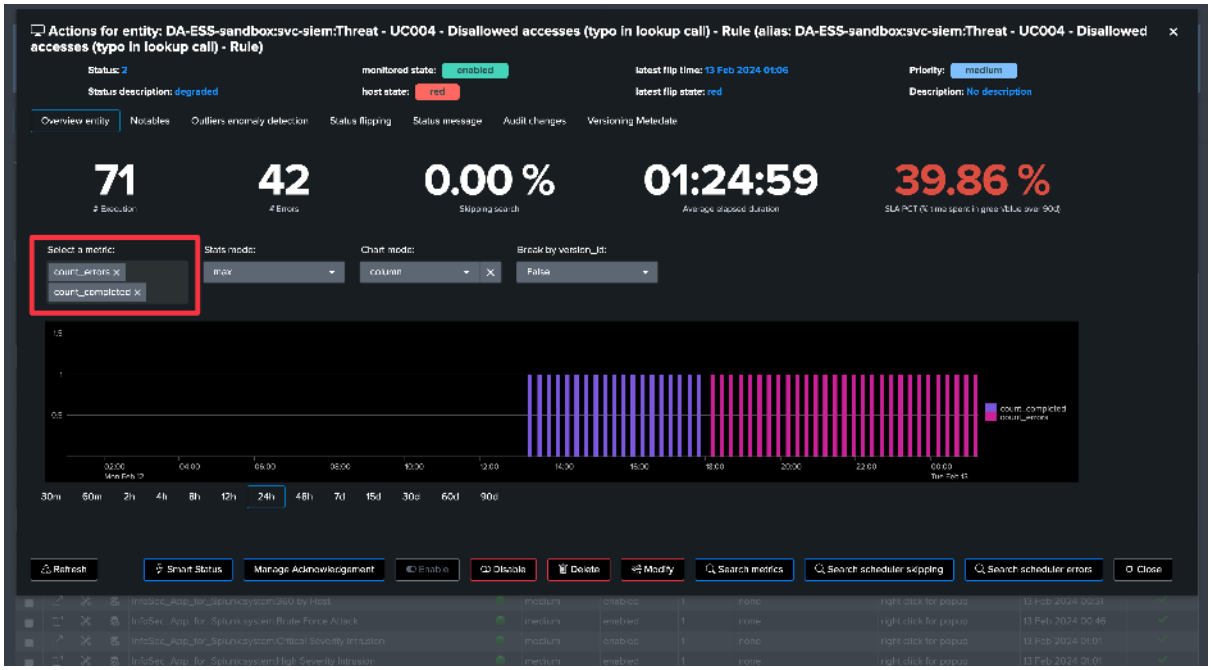
Execution errors typically indicate non-working searches and may slightly affect your Splunk use cases

- There are plenty of reasons why a Splunk search could start failing at some point in its lifecycle.
- This may happen due to temporary infrastructure issues, incorrect syntax, breaking change introduction, third-party application installation or updates… plenty!
- This gets also more complex and challenging with the continuous growth of the environment and use cases. Very often, use cases fail silently and are not detected until a major issue is encountered.
- With TrackMe's Workload component, anomaly executions are detected at scale, made easily available and automatically investigated for use cases and platform owners to review as soon as possible.

Searches suffering from execution anomalies are tagged with the anonaly_reason=execution_errors_detected



Open the entity and review when execution errors were detected

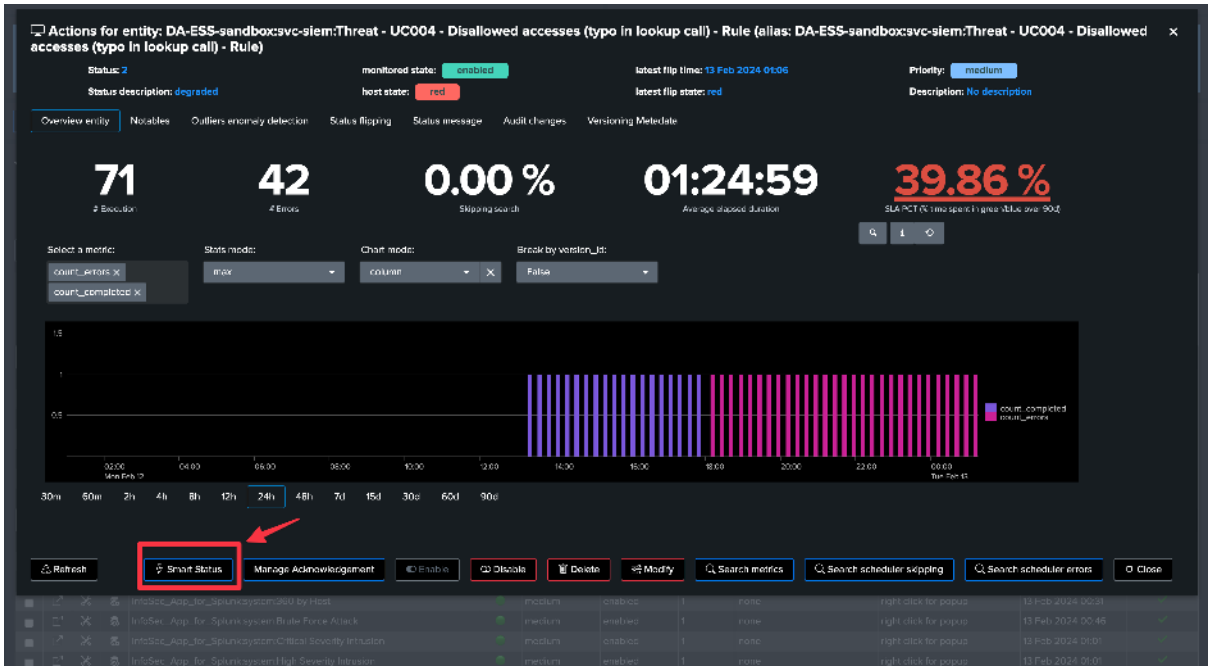


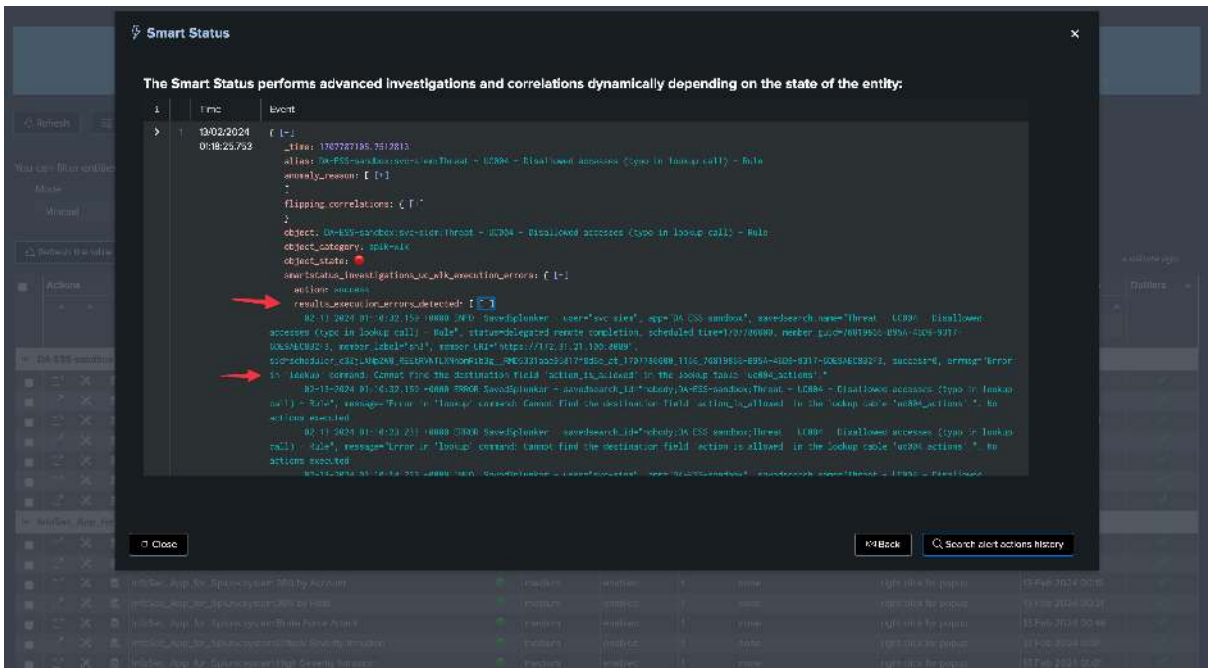
In a single click, you can use the SmartStatus to retrieve a meaningful sample of the Splunk scheduler execution errors:

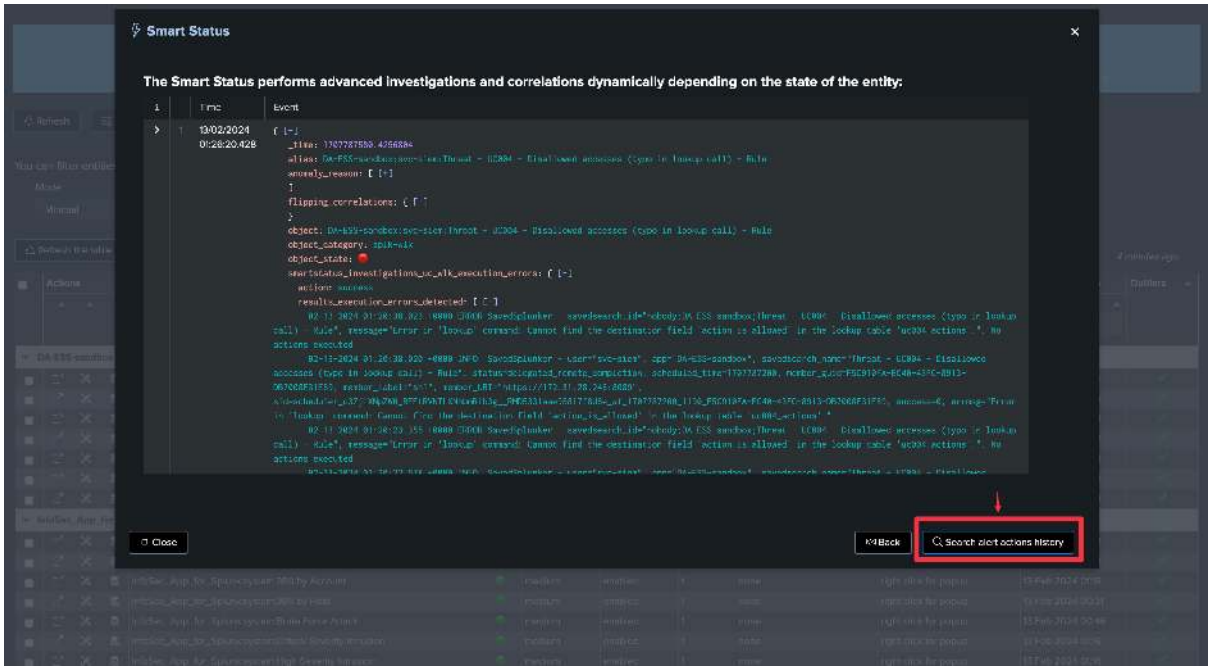
Hint

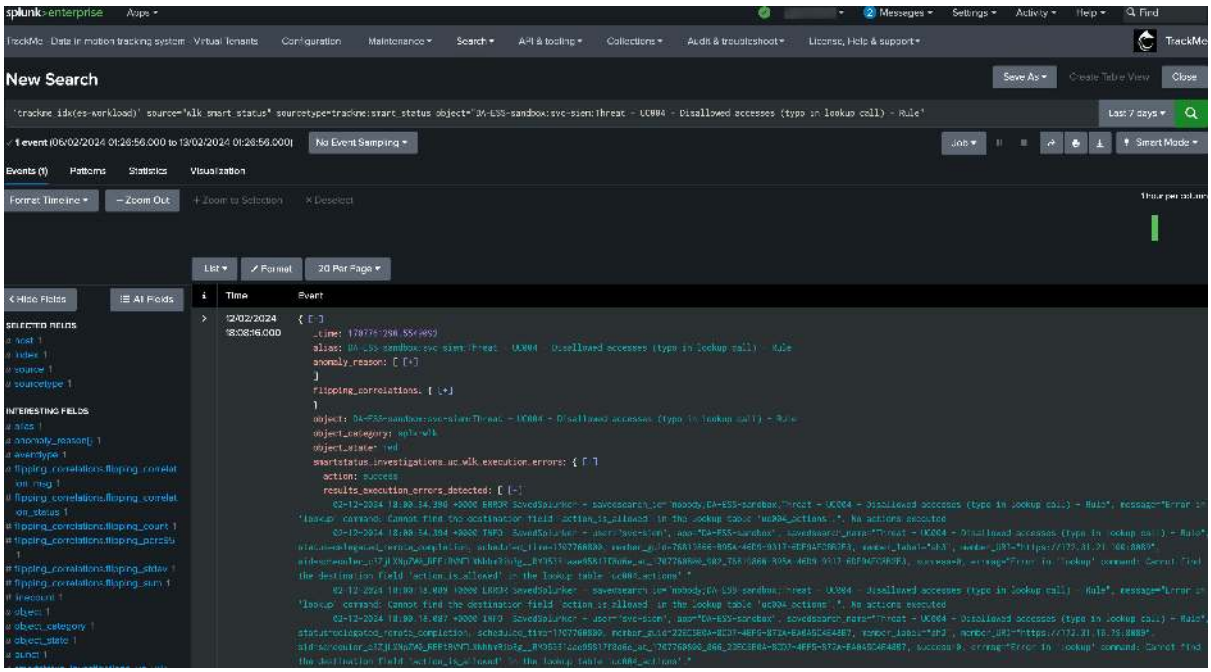
SmartStatus provides access to the _internal execution errors to use cases owners with no access to the _internal index

- With the SmartStatus and TrackMe's RBAC management, use case owners can transparently access to the execution errors from the _internal, even if they do not have access to the _internal index.
- This is possible via controlled elevation of privileges, users that are granted access to the TrackMe API endpoints through the RBAC configuration, will be able to execute the SmartStatus action which itself is executed with system level privileges.
- Finally, TrackMe also executes the SmartStatus automatically via the builtin alert action, and indexes the results of it into the associated TrackMe summary index.


Actions for entity: DA-ESS-sandbox:svc-siem:Threat - UC004 - Disallowed accesses (typo in lookup call) - Rule (alias: DA-ESS-sandbox:svc-siem:Threat - UC004 - Disallowed accesses (typo in lookup call) - Rule)
Status: 2
monitored state: enabled
latest flip time: 13 Feb 2024 01:06
Priority: medium
Status description: degraded
host state: red
latest flip state: red
Description: No description
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
Versioning Metadata
71
42
0.00 %
01:24:59
39.86 %
Select a metric:
Stats mode:
Chart mode:
Break by version_id:
count_errors
count_completed
max
column
False
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
Refresh
Smart Status
Manage Acknowledgement
Enable
Disable
Delete
Modify
Search metrics
Search scheduler skipping
Search scheduler errors
Close


Smart Status
The Smart Status performs advanced investigations and correlations dynamically depending on the state of the entity:
Time
Event
13/02/2024
01:18:25.793
Close
Back
Search alert actions history





Use case: Use TrackMe's version to establish relations between search updates and changes in behavior

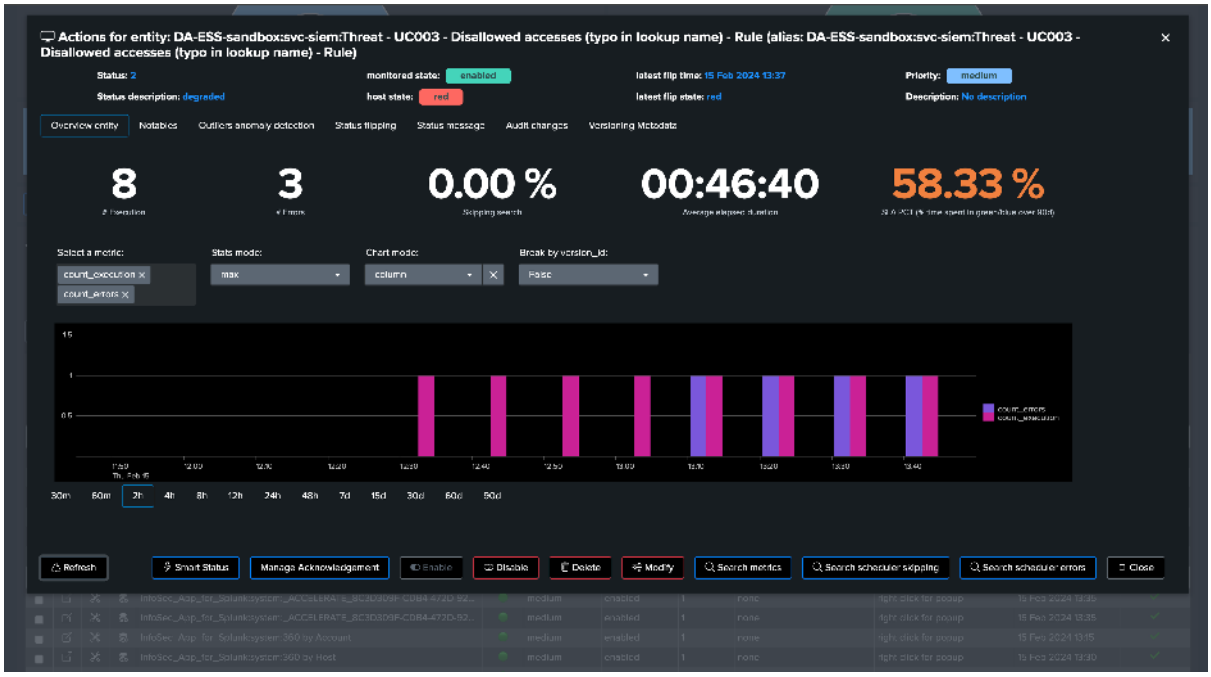
Using TrackMe's Workload search versioning, you can easily detect changes and make the link between these changes and how these impact the search behavior

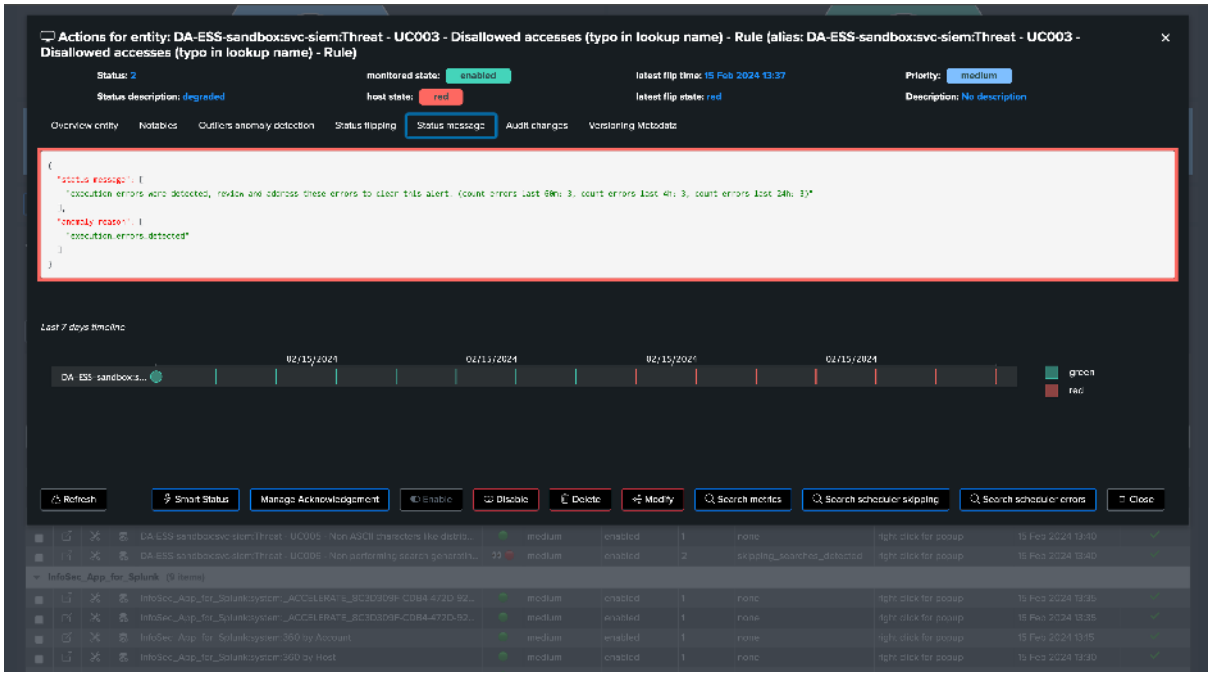
- Once TrackMe discovers a search entity, it starts to continuously track the code version of the search.
- TrackMe creates a dimension called `version_id` which is the unique SHA256 hash based on the Workload keys in configuration. (by default: search,dispatch.earliest,dispatch.latest,description,cron_schedule)
- This dimension is indexed with the Workload metrics and represents the unique instance of a search at some point in time.
- When a change to the search is detected, a new value of the `version_id` is generated, the search ver-

sion KVstore record is updated and TrackMe generates a version event which is indexed in the TrackMe summary index of the tenant.

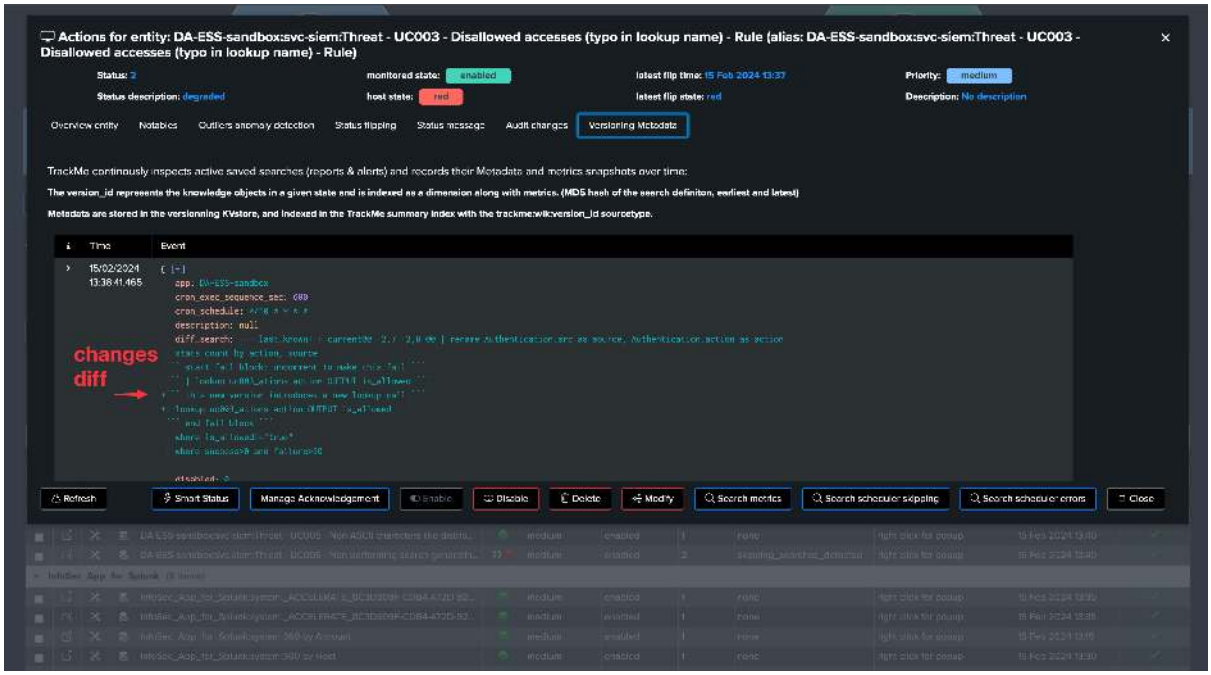
- TrackMe also issues a diff of the search definition, time quantifiers and attempts to identify when and who updated the search, which information is added to the versioning record and event.
- Finally, you can review changes easily in TrackMe UI, whether this search is local to the TrackMe host or hosted on any remote deployment, without quitting TrackMe for a second.

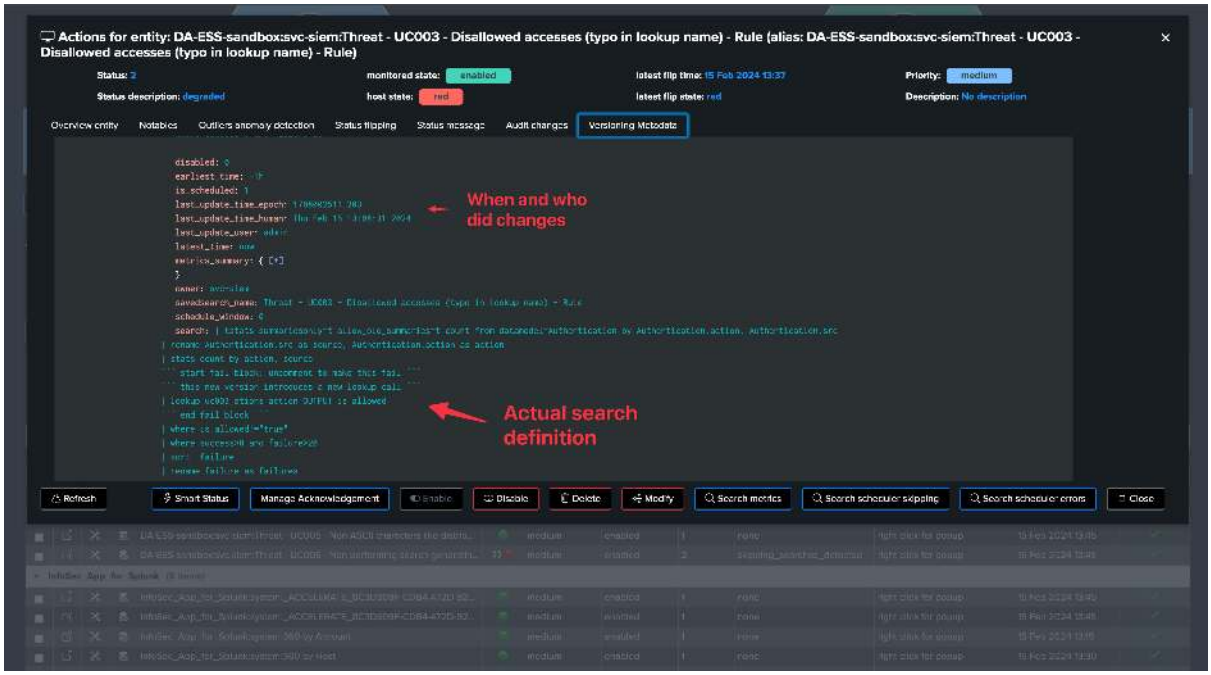
Say you have a search which suddenly started to generate anomalies execution, TrackMe captures this fact and impacted the entity:



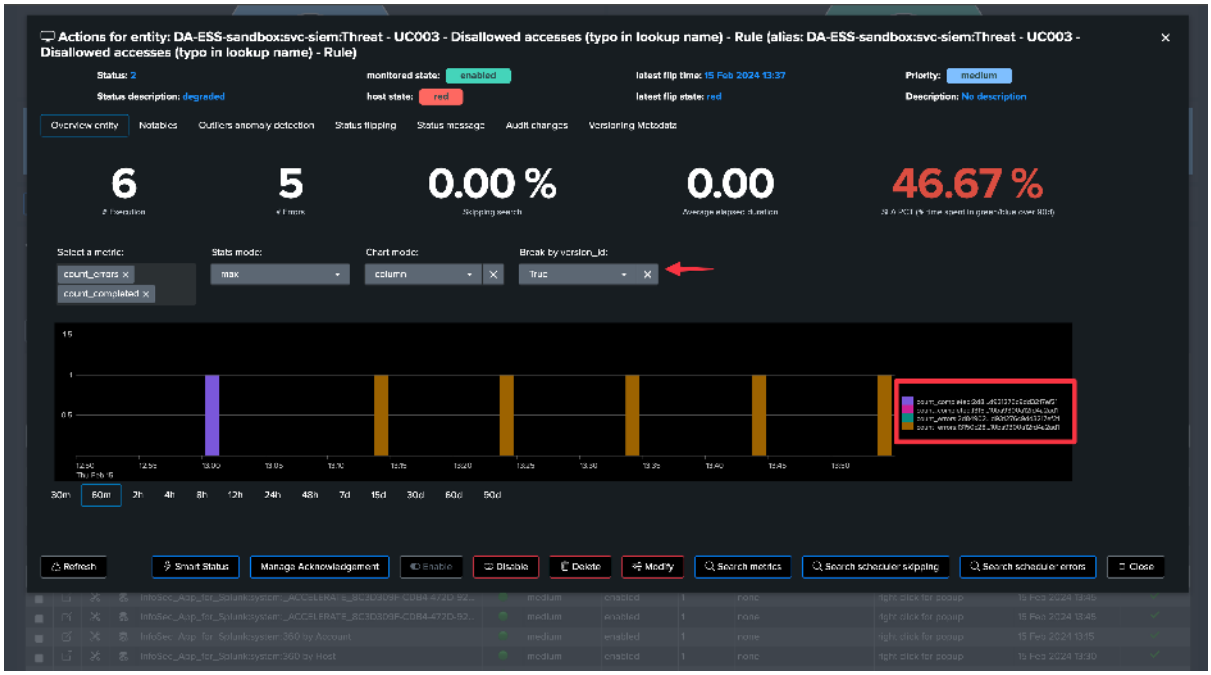


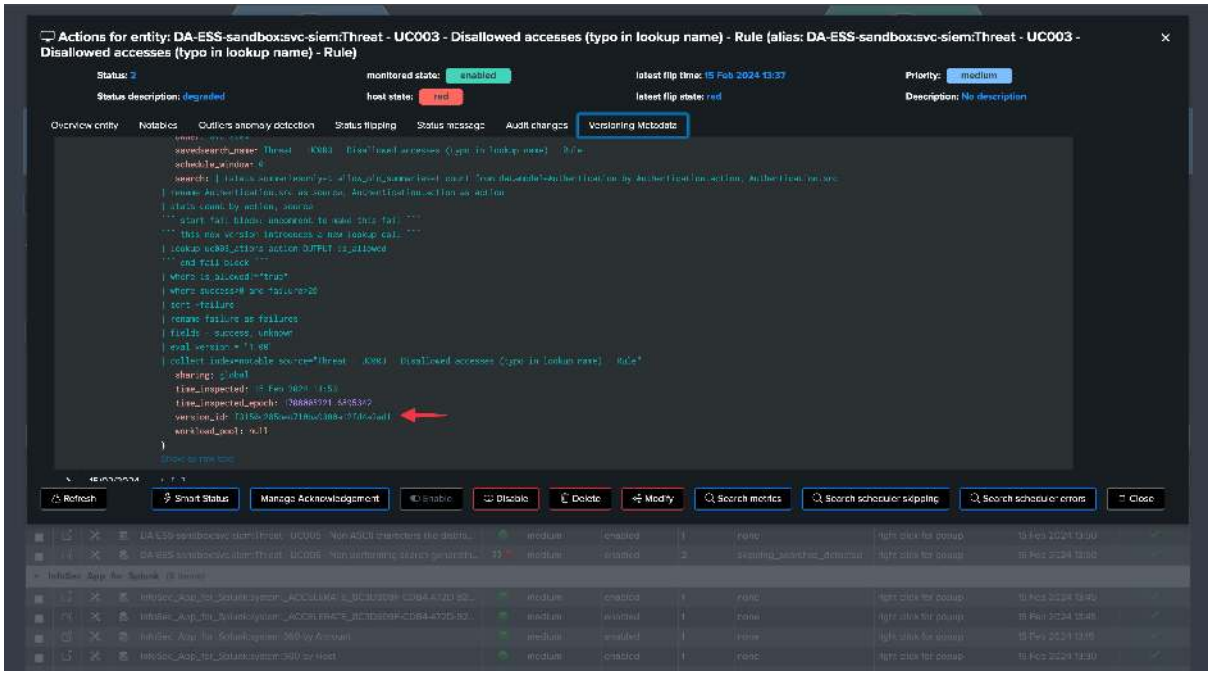
Looking at the version tab, you can see the changes made to the search, and when these were made:





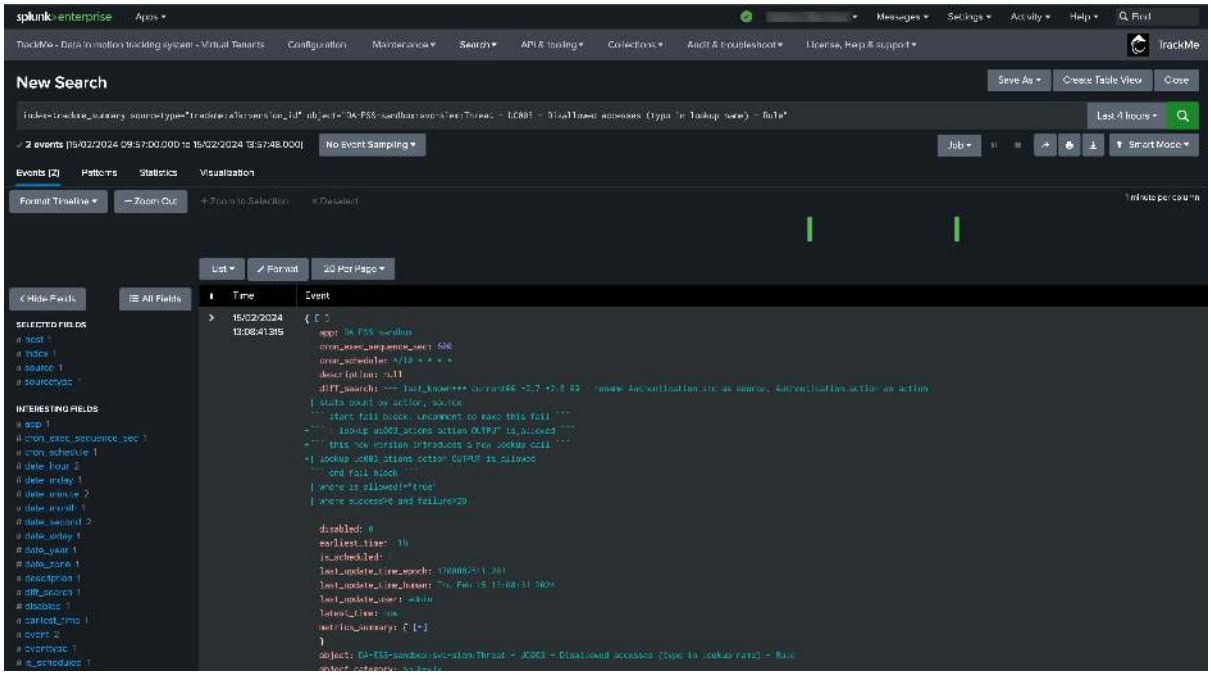
From the metrics perspective, we break by the version_id and identify when the change was made and how this impacted the search behavior:





You can also look at the corresponding versioning events:

```
index=trackme_summary sourcetype="trackme:wlk:version_id" object=*
```

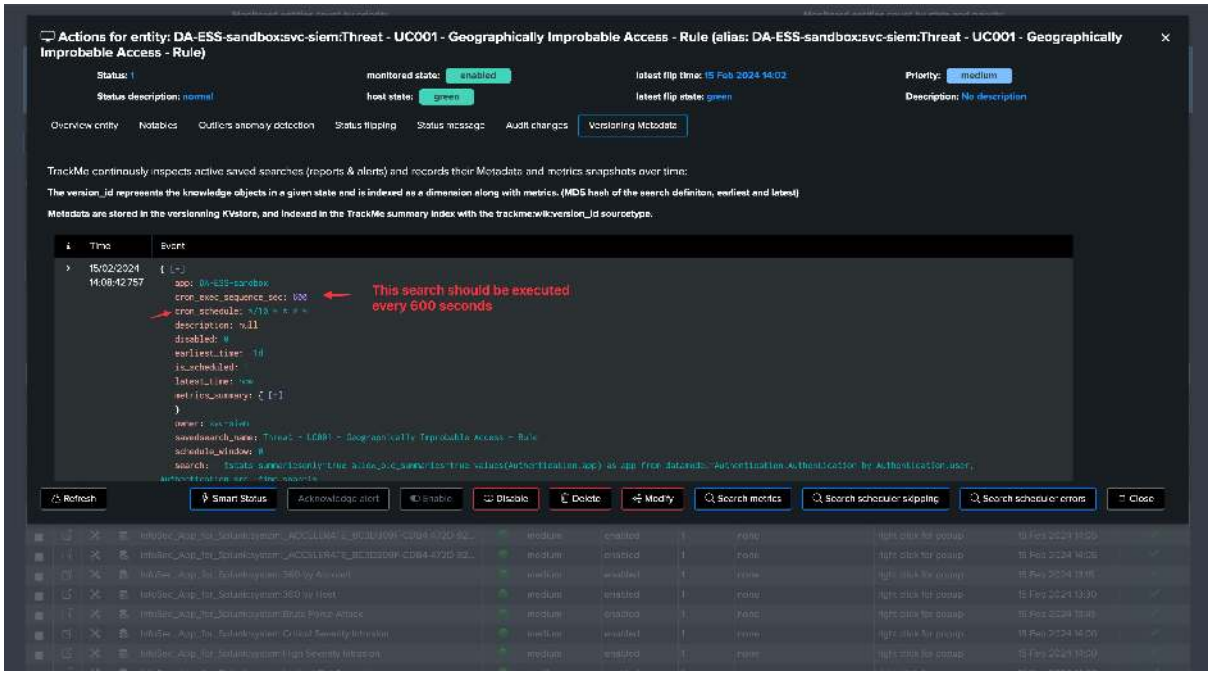


Use case: Detect delayed searches

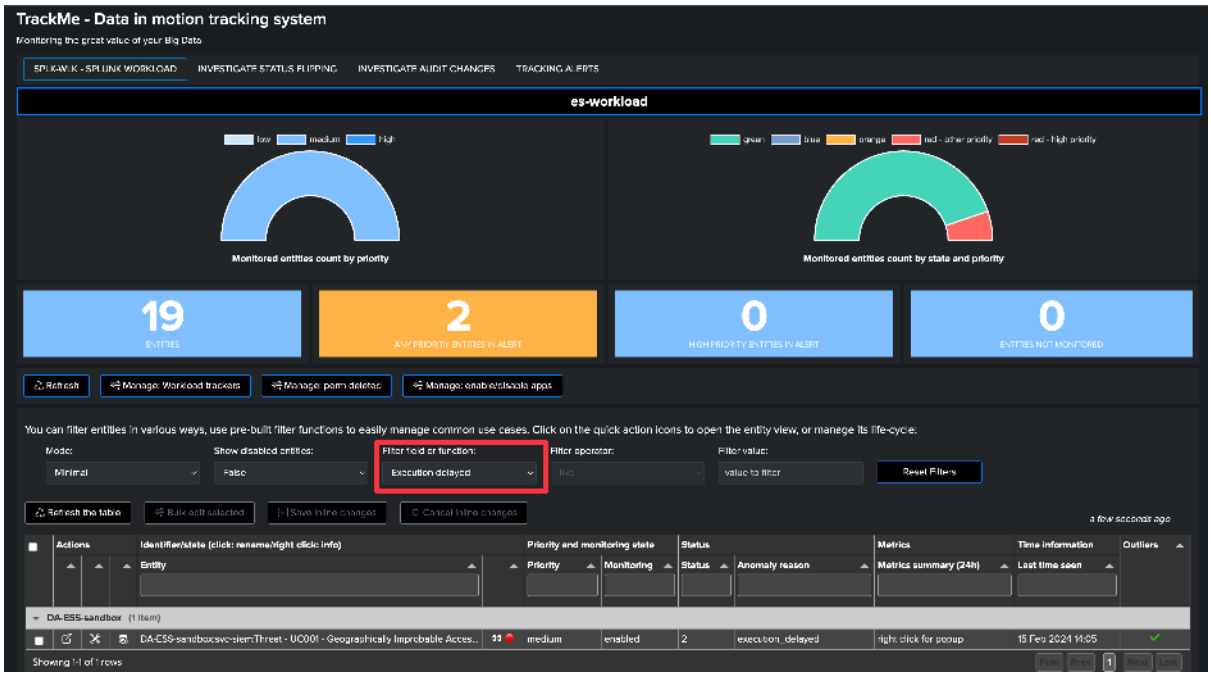
By monitoring scheduled searches, TrackMe also detects if a search has stopped being executed

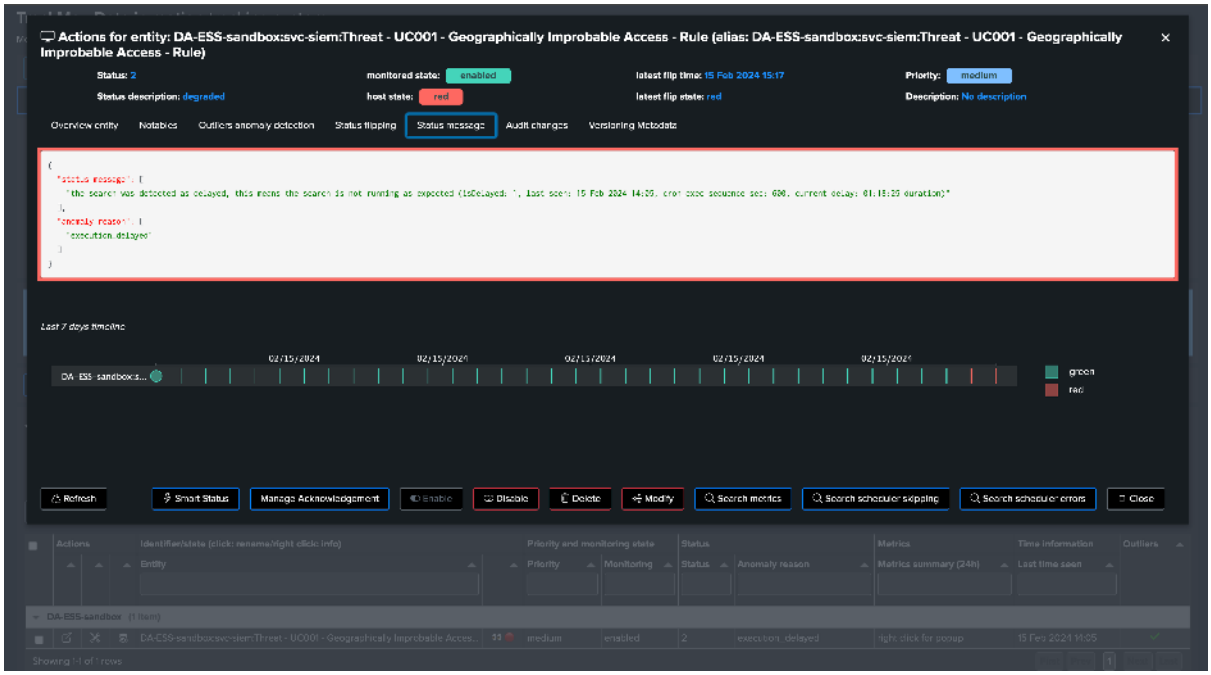
- Using its versioning capabilities, TrackMe interprets the cron schedule and defines the `cron_exec_sequence_sec` (using the Python library croniter).
- This value in seconds represents the time expected between two executions of the search.
- If a search that was discovered suddenly stops being executed, TrackMe will detect it and tag the entity as delayed.
- TrackMe applies a 60-minute delay grace time before impacting the entity.
- This is a vital verification, especially for SIEM environments. Being capable of detecting use cases that are not executed anymore is very important and should lead to the investigation of the issue.

In the versioning tab, you can see the cron_schedule as well as the cron_exec_sequence_sec:

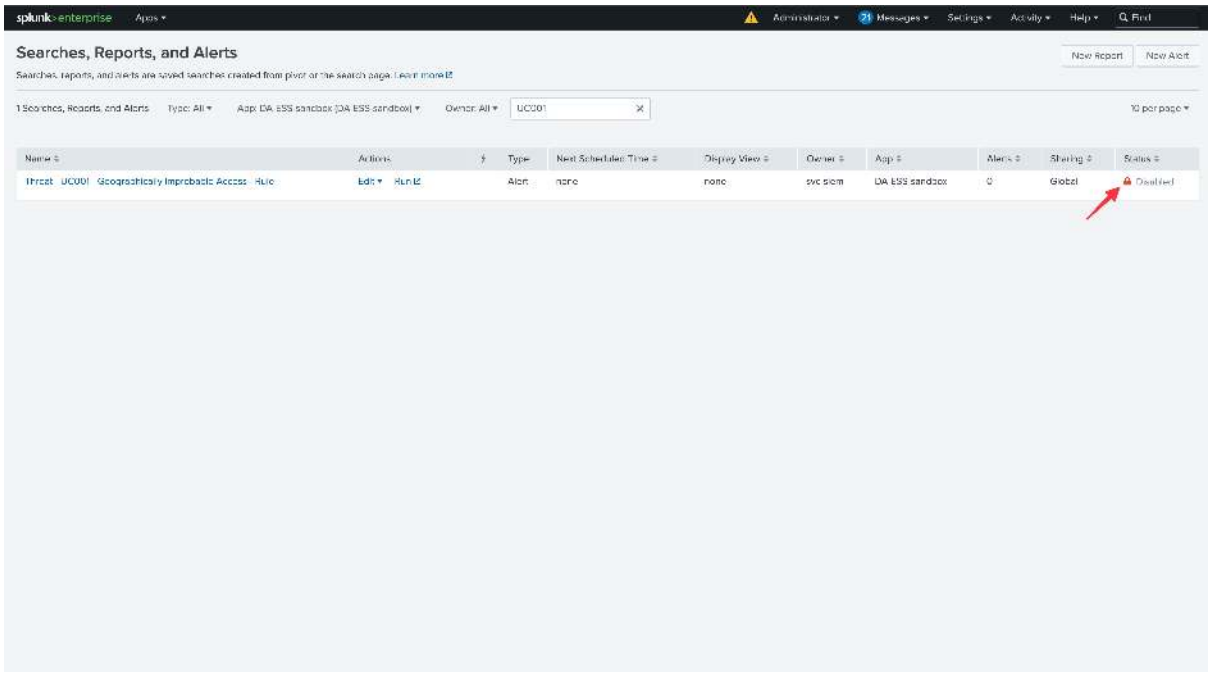


TrackMe detects that the search has stopped being executed, and tags the entity as delayed:





The search was indeed disabled at the Splunk level, which might have been intentional, or not, and should be reviewed:



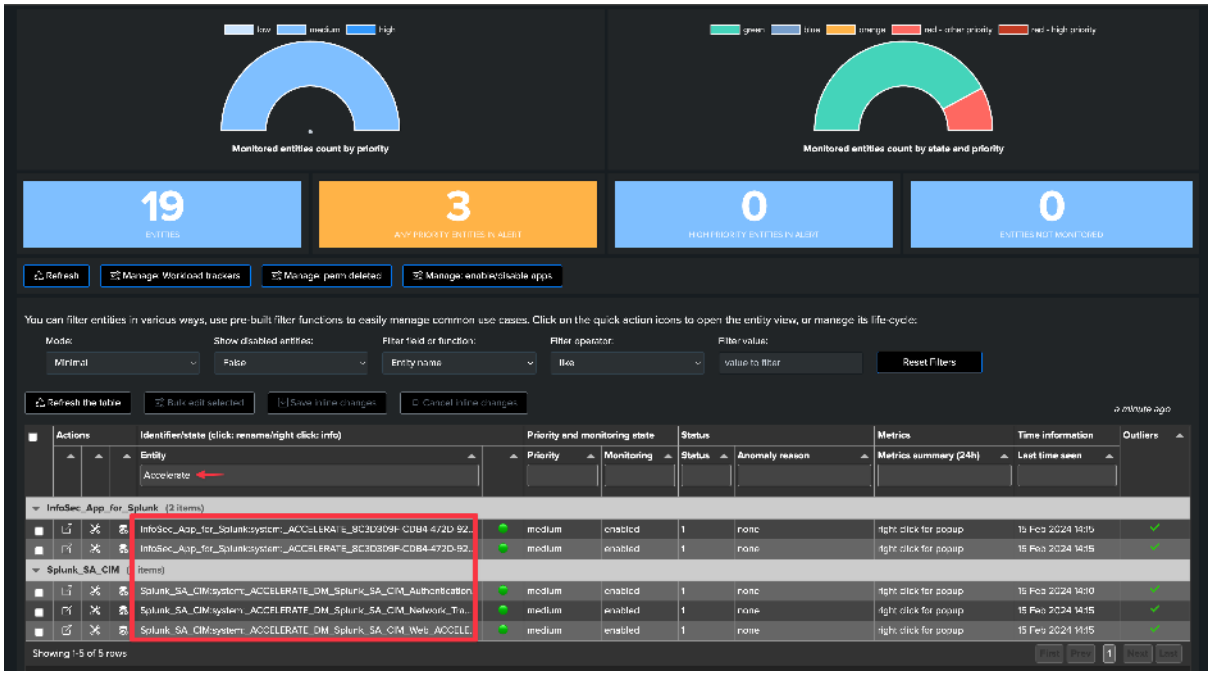
Use case: Monitoring Data Models Acceleration and Reports Acceleration behaviors and performance

Splunk Accelerated Searches in the Workload component

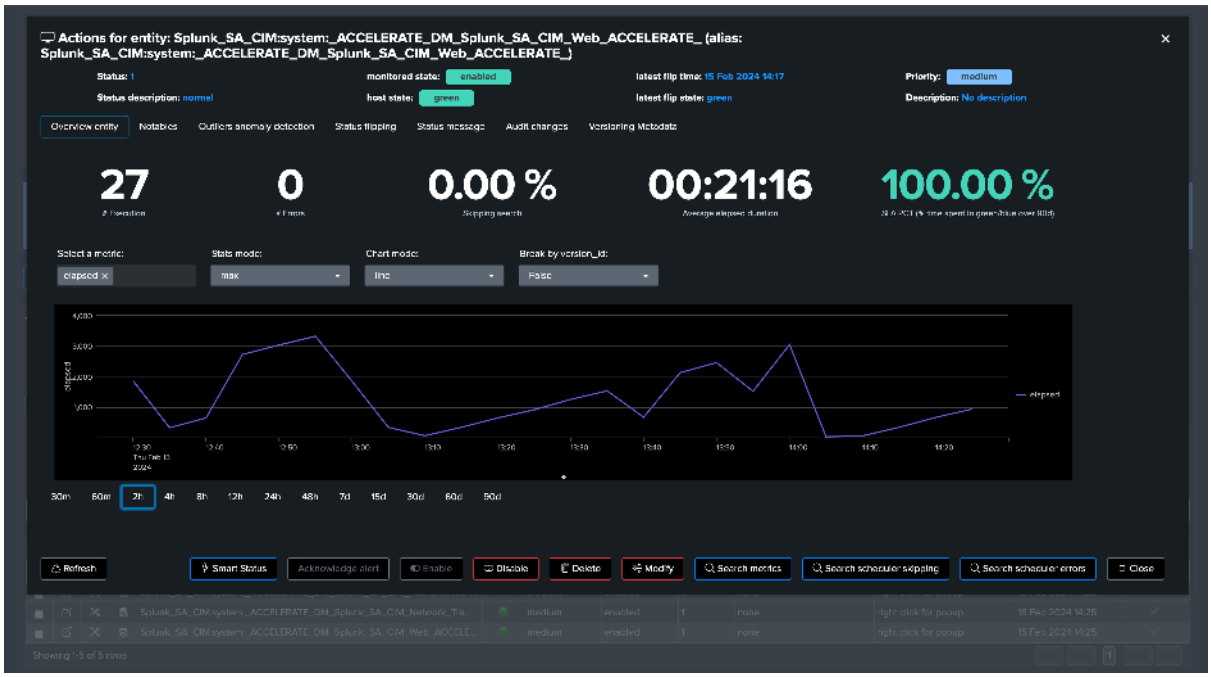
- TrackMe also discovers and monitors searches that relate to Splunk Data Models (DMA) and Reports Acceleration.
- Especially for DMA, it is important to monitor the level of Skipping searches per Data Model, as well as the elapsed time performance of these searches.
- TrackMe will, for instance, automatically alert if these are suffering from high skipping searches ratios.

- Finally, TrackMe also uses Machine Learning to detect Outliers regarding the performance of the searches and may alert if an abnormal increasing trend is detected.

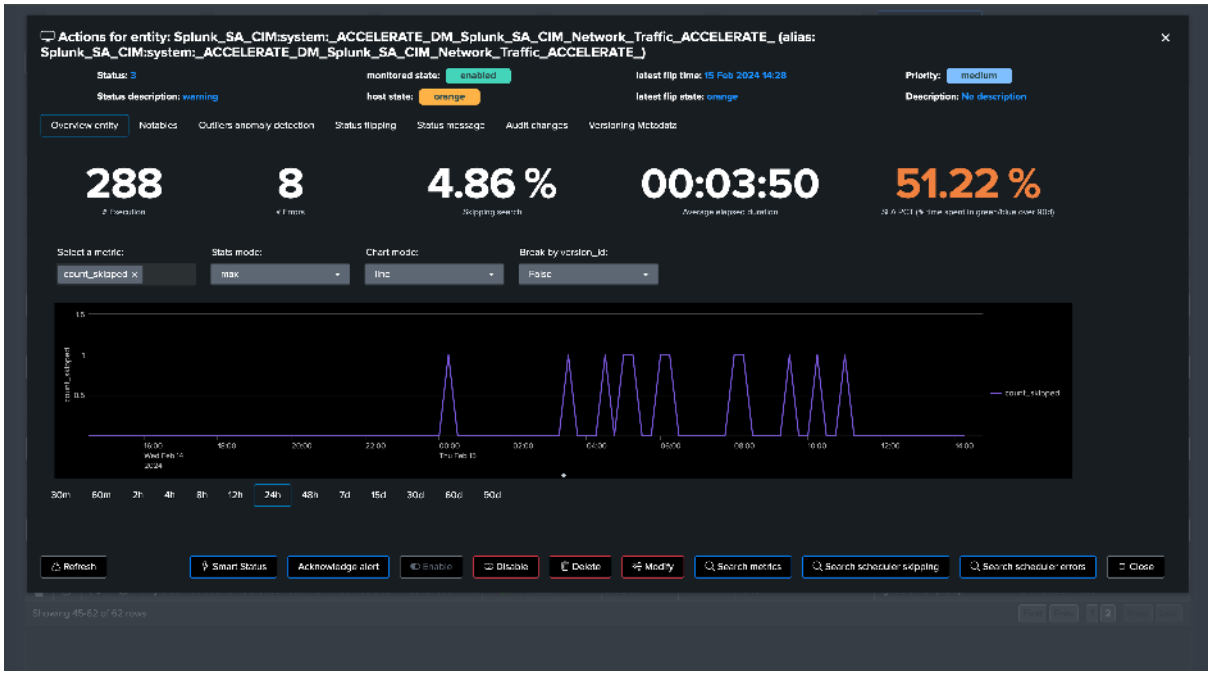
DMA and report acceleration searches are automatically discovered and monitored by TrackMe:



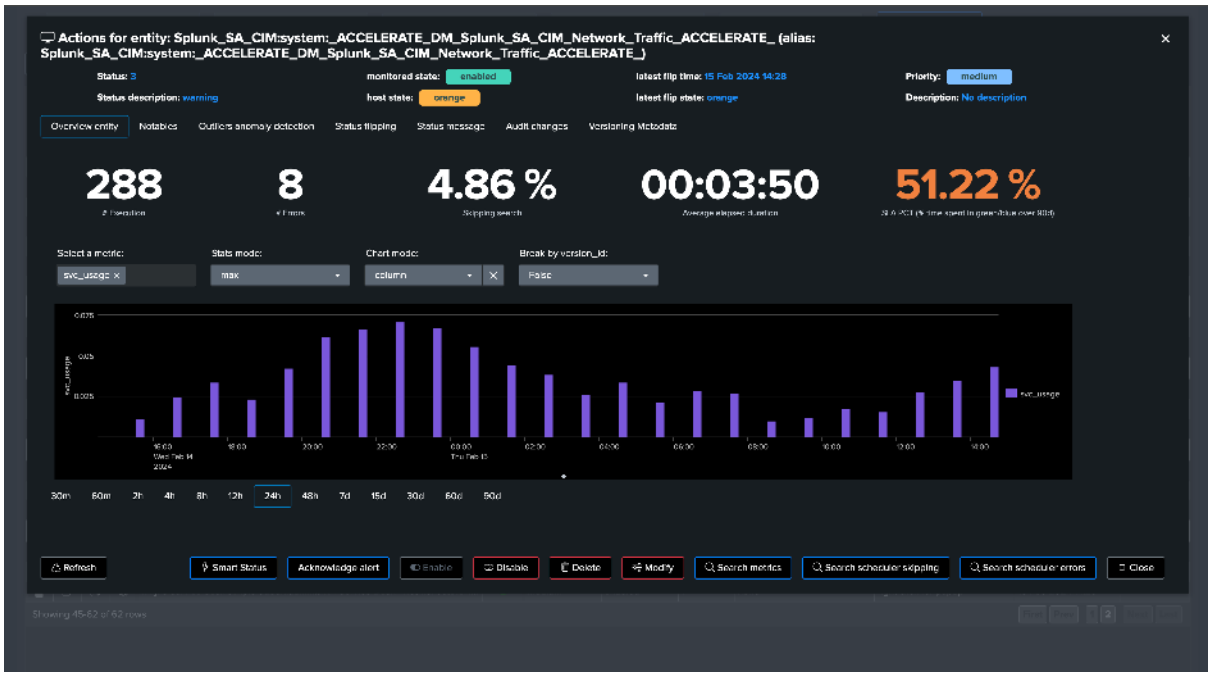
All metrics are available, such as the elapsed: (run time in seconds)



This environment is struggling to maintain DMA for instance, indicating underneath issues that can affect your security use cases:



Splunk Cloud customers can easily review related SVC usage:



Splunk Enterprise & Cloud can also review CPU usage and memory consumption metrics:





TrackMe's unique ML Outliers detection clearly catched the bad situation:



There are plenty more use cases that can be covered by the Workload component, and the above are just a few examples.

Some more documentation:

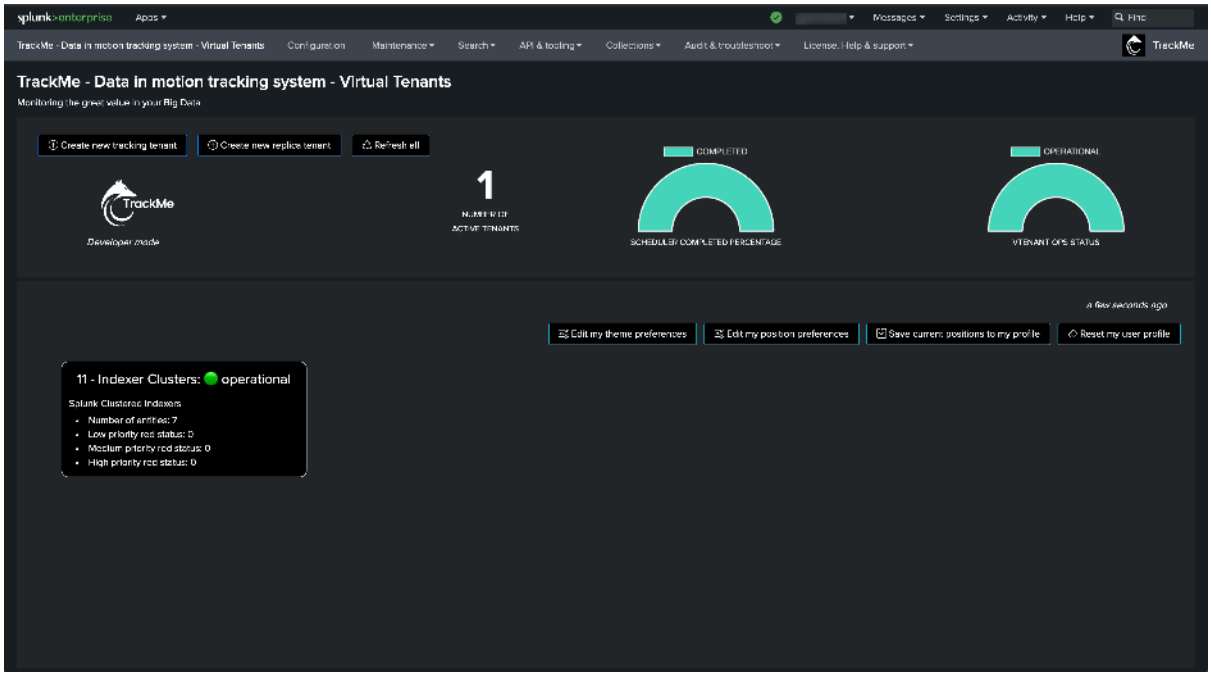
- *WLK — Workload Knowledge*
- *WLK — Workload Knowledge*

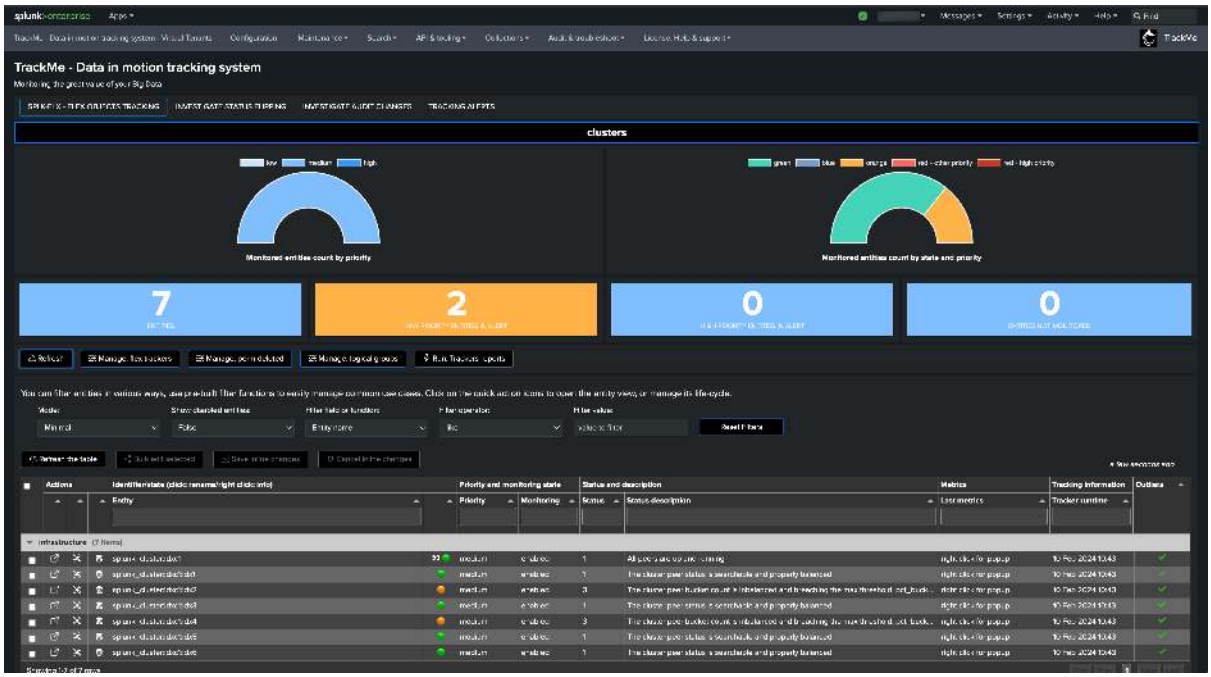
9.3.3 Monitor Splunk Indexer Clusters

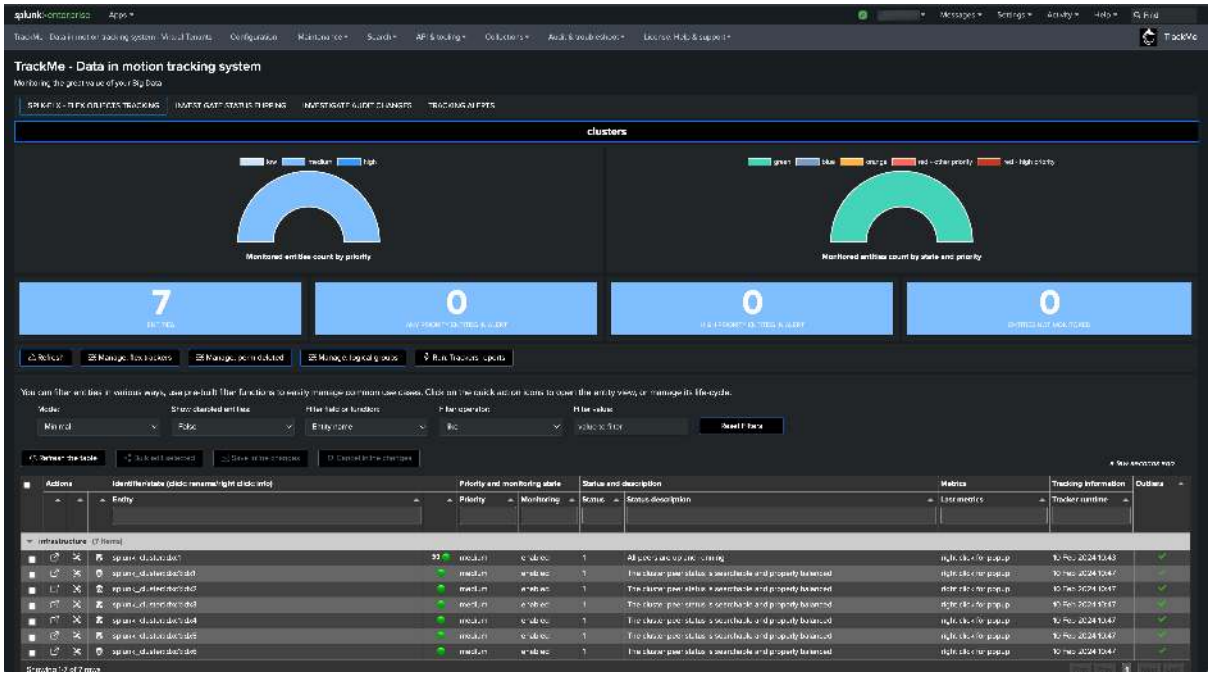
Monitoring Splunk Indexer Clusters with TrackMe Flex Objects

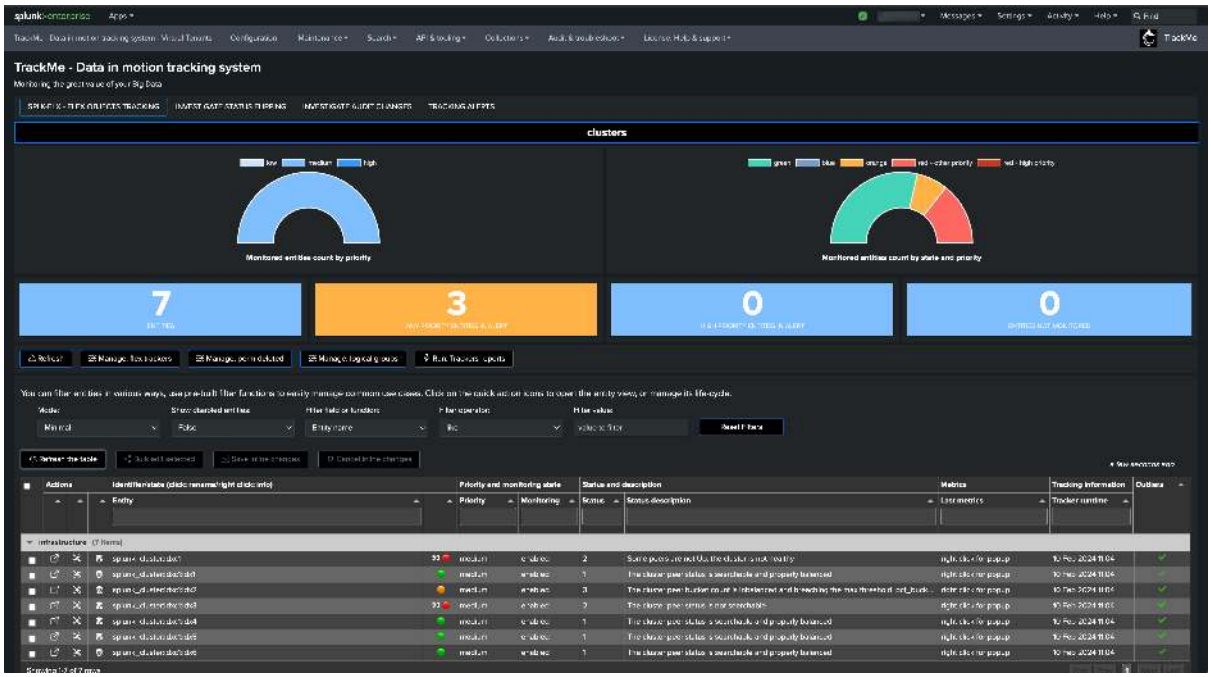
- This tutorial demonstrates the monitoring of **Splunk Indexer Clusters** with TrackMe Flex Objects.
- TrackMe Flex Objects is a component restricted to licensed customers. Please contact Splunk Sales for more information.
- Using these steps will enable TrackMe to continuously monitor the health of your Splunk Indexer Clusters and alert you when issues are detected.
- With TrackMe's remote search capabilities, you can monitor as many Indexer Clusters as you need from a single pane of glass.

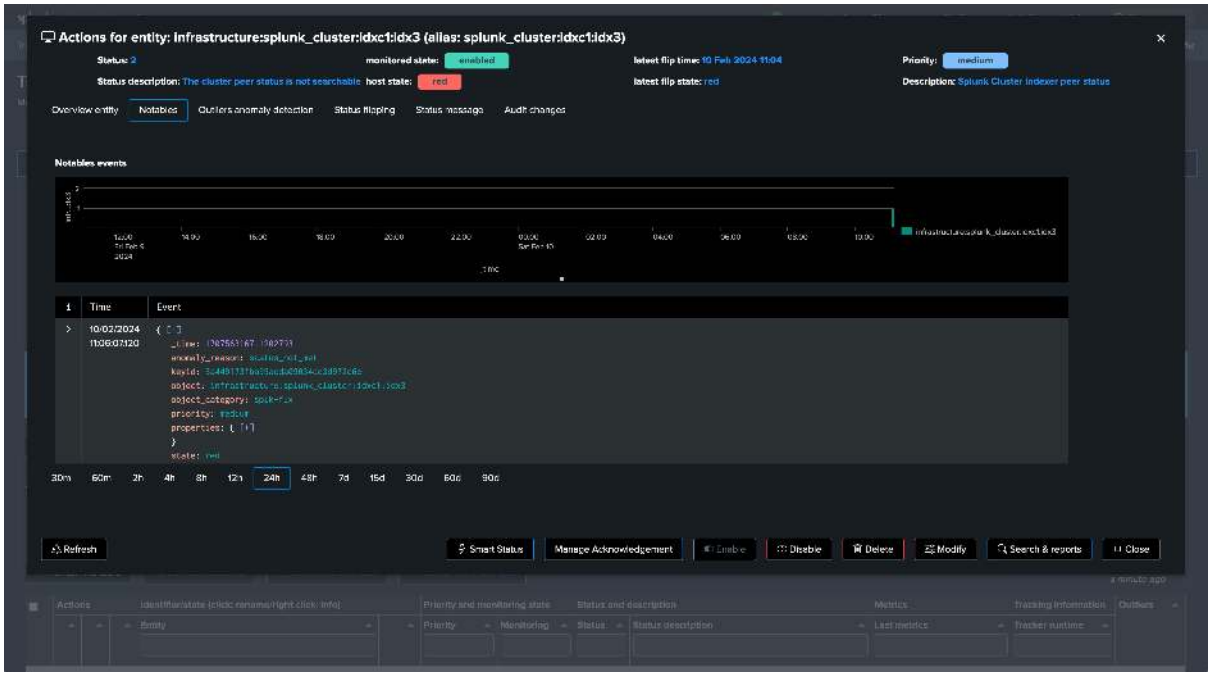
The following screen shows the final results in TrackMe, starting from Step 1 to easily implement the monitoring of your Splunk Indexers Clusters.

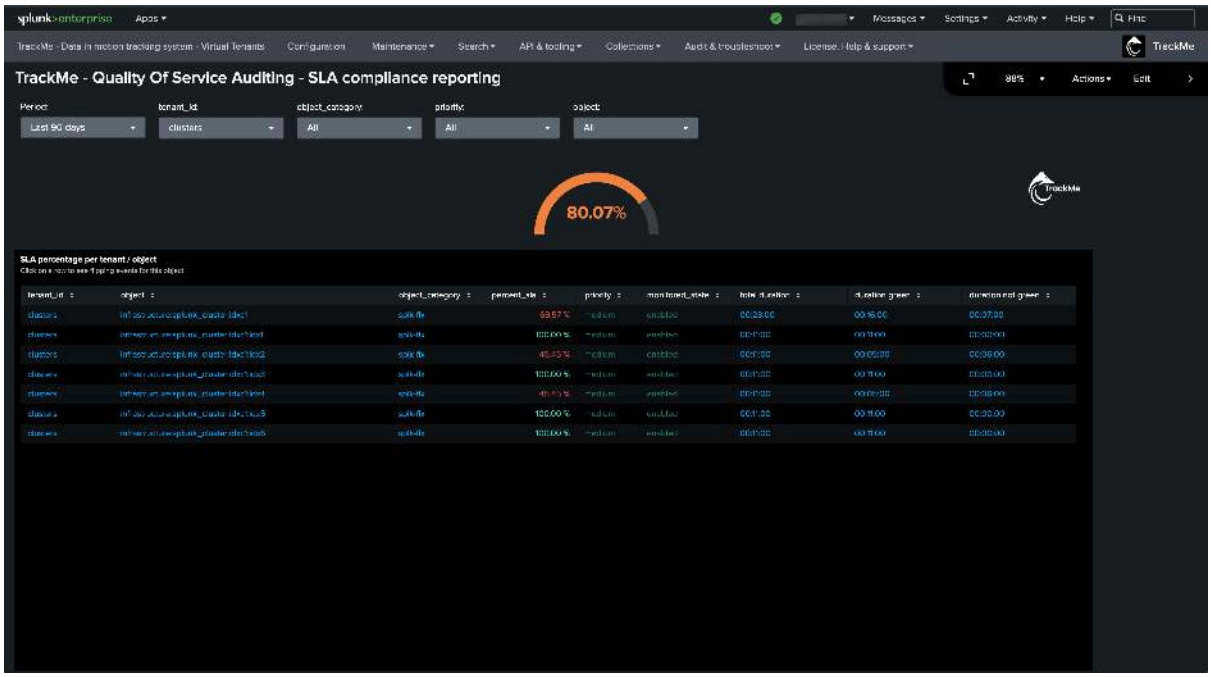

splunk>enterprise
TrackMe - Data in motion tracking system - Virtual Tenants
Create new tracking tenant
Create new replica tenant
Refresh all
Developer mode
1
COMPLETED
SCHEDULER COMPLETED PERCENTAGE
OPERATIONAL
a few seconds ago
Edit my theme preferences
Edit my position preferences
Save current positions to my profile
Reset my user profile
11 - Indexer Clusters: operational
Splunk Clusters Indexers
Number of entities: 7
Low priority red status: 0
Medium priority red status: 0
High priority red status: 0


TrackMe - Data in motion tracking system
clusters
Monitored entities count by priority
Monitored entities count by state and priority
7
2
0
0


TrackMe - Data in motion tracking system
clusters
Monitored entities count by priority
Monitored entities count by state and priority
7
0
0
0


TrackMe - Data in motion tracking system
clusters
Monitored entities count by priority
Monitored entities count by state and priority
7
3
0
0



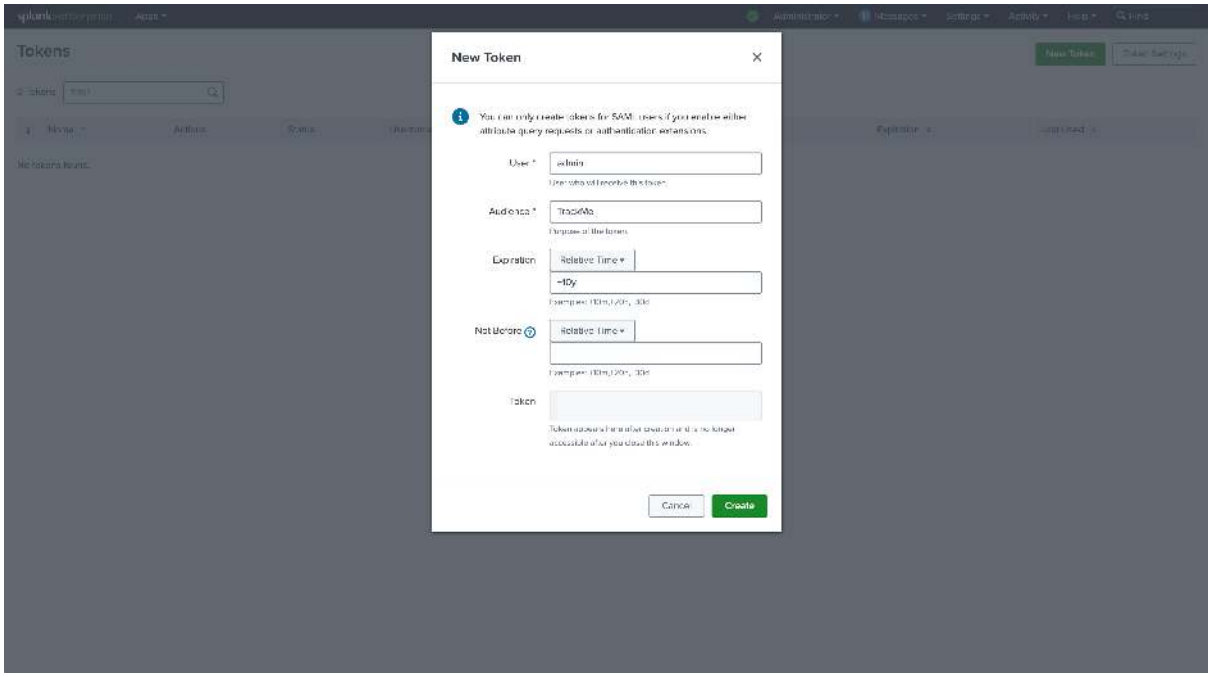


Step 1: Create a Splunk Remote Deployment Account for the Cluster Manager

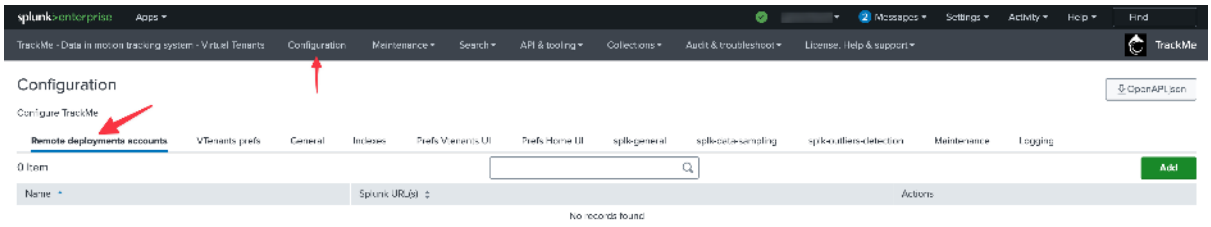
The first step is to create a Splunk Remote Deployment Account for the Cluster Manager. For more information about TrackMe Remote Search capabilities and configuration:

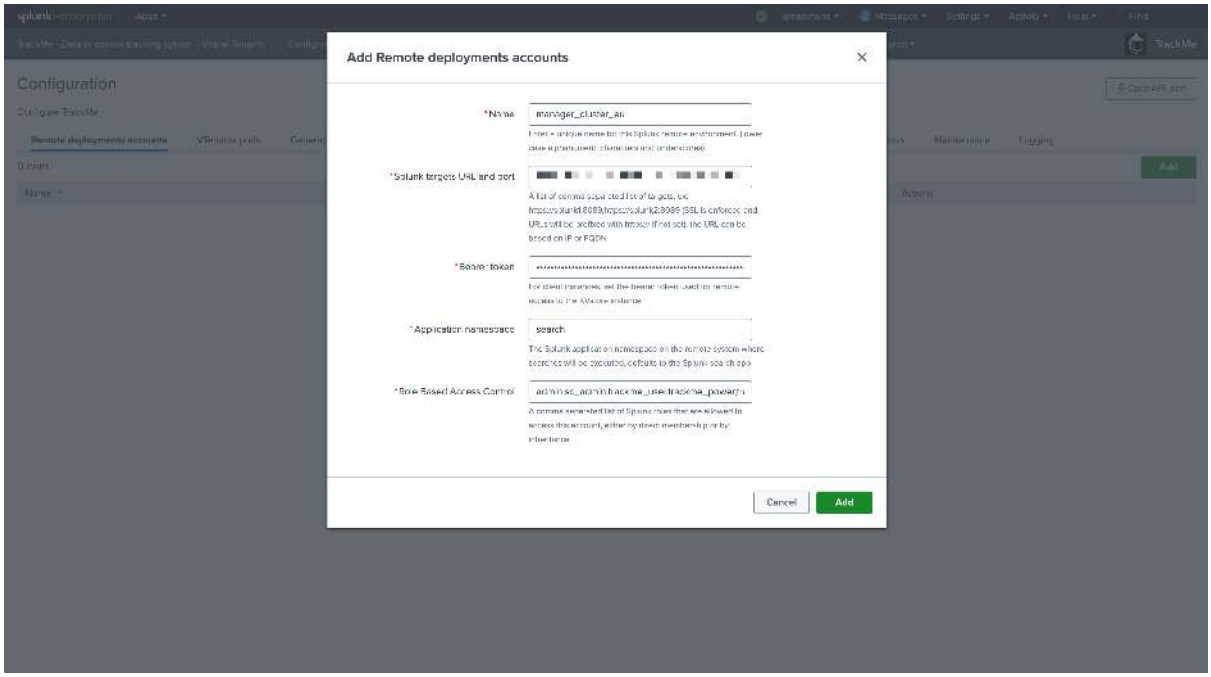
Remote Splunk deployments

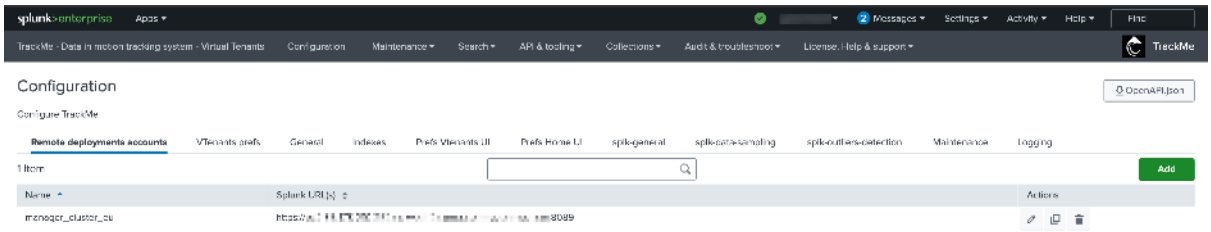
On the Cluster Manager, create a new Splunk bearer token for the TrackMe Remote Deployment Account:



In TrackMe, click on Configure / Remote Deployment Accounts and add a new account:




Add Remote deployments accounts
Cancel
Add


splunk>enterprise
Apps
Messages
Settings
Activity
Help
Find
TrackMe - Data in motion tracking system - Virtual Tenants
Configuration
Maintenance
Search
API & tooling
Collections
Audit & troubleshoot
License, Help & support
TrackMe
Configuration
Configure TrackMe
OpenAPI.json
Remote deployments accounts
VTenants prefs
General
Indexes
Prefs VTenants UI
Prefs Home UI
spk-general
spk-cara-sampling
spk-outliers-detection
Maintenance
Logging
1 Item
Add
Name
Splunk URL(s)
Actions
manager_cluster_eu

Hint

Managing multiple Indexer Clusters

- If you have multiple Indexer Clusters, you can create a Remote Deployment Account for each Cluster Manager.
- You will then be able to manage and monitor as many Indexer Clusters as you need from a single pane of glass in TrackMe.

Step 2: Create a Flex Object tenant for monitoring Indexer Clusters

Now, let's create a new tenant for the purposes of monitoring our Indexer Clusters. We can achieve this via the UI wizard or with a simple command line:

```
| trackme url=/services/trackme/v2/vtenants/admin/add_tenant mode=post body="{
↪'tenant_name': 'clusters', 'tenant_alias': '11 - Indexer Clusters', 'tenant_desc
↪': 'Splunk Clustered Indexers', 'tenant_roles_admin': 'trackme_admin',
↪'tenant_roles_power': 'trackme_power', 'tenant_roles_user': 'trackme_user',
↪'tenant_owner': 'admin', 'tenant_idx_settings': 'global', 'tenant_flx_enabled':
↪'true'}"
```

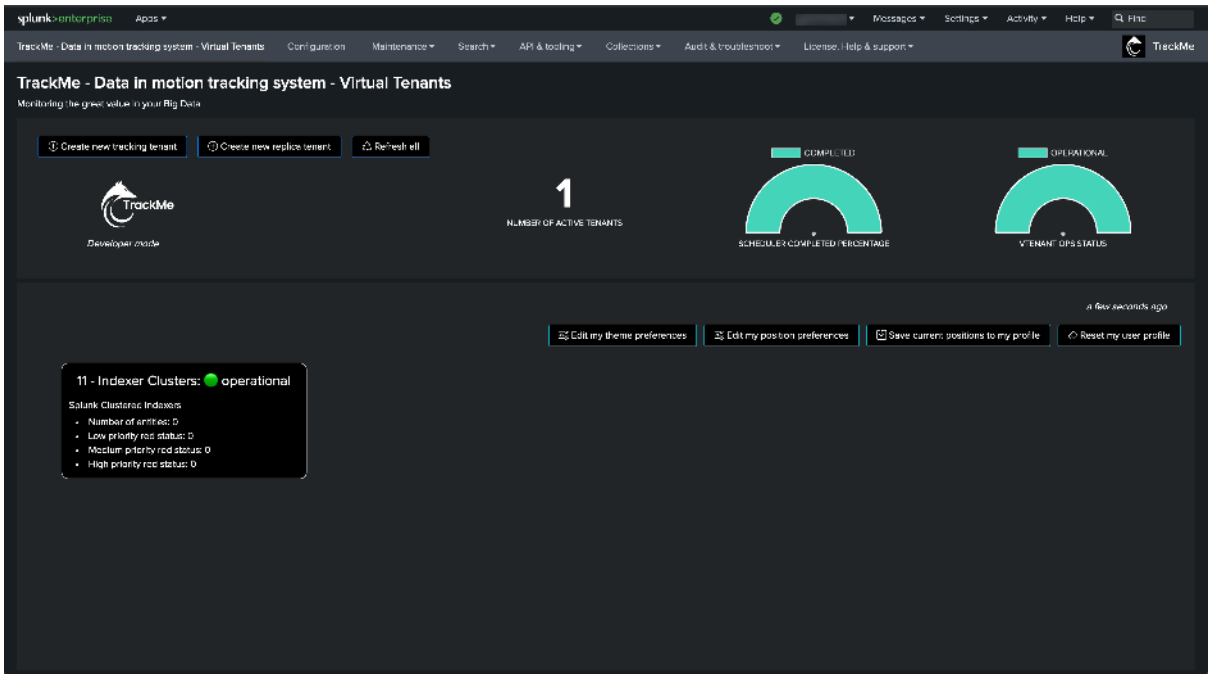
Step 3: Create Flex Object trackers using the Flex Objects library

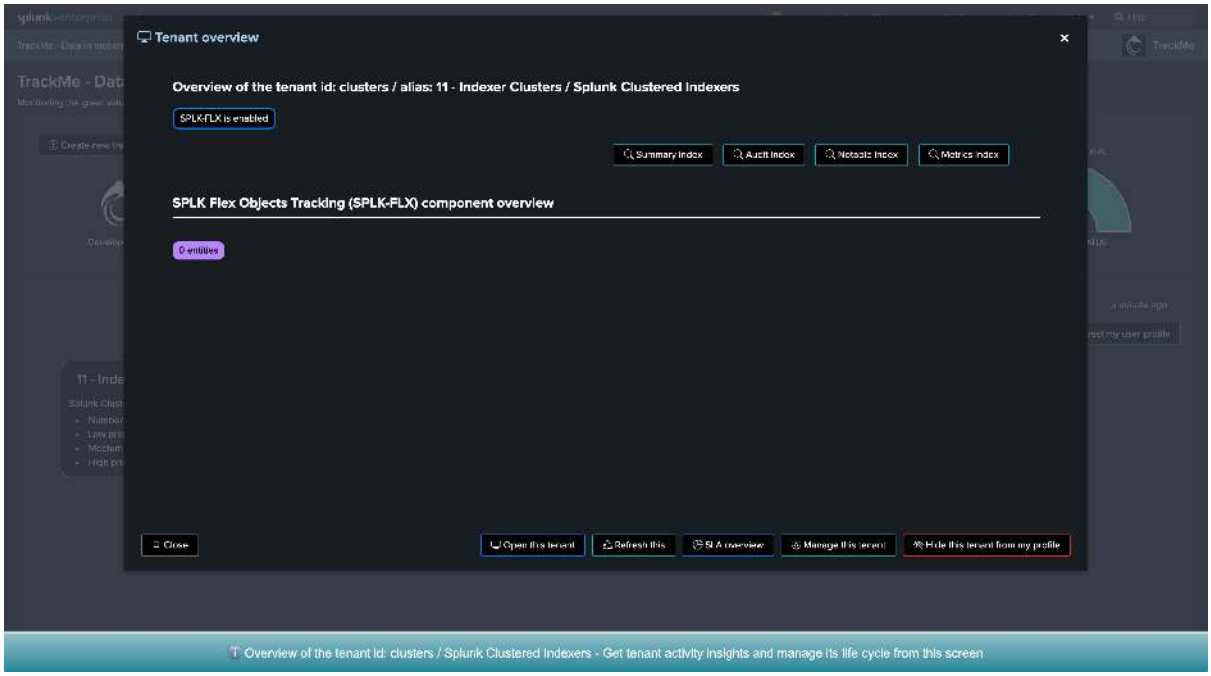
Hint

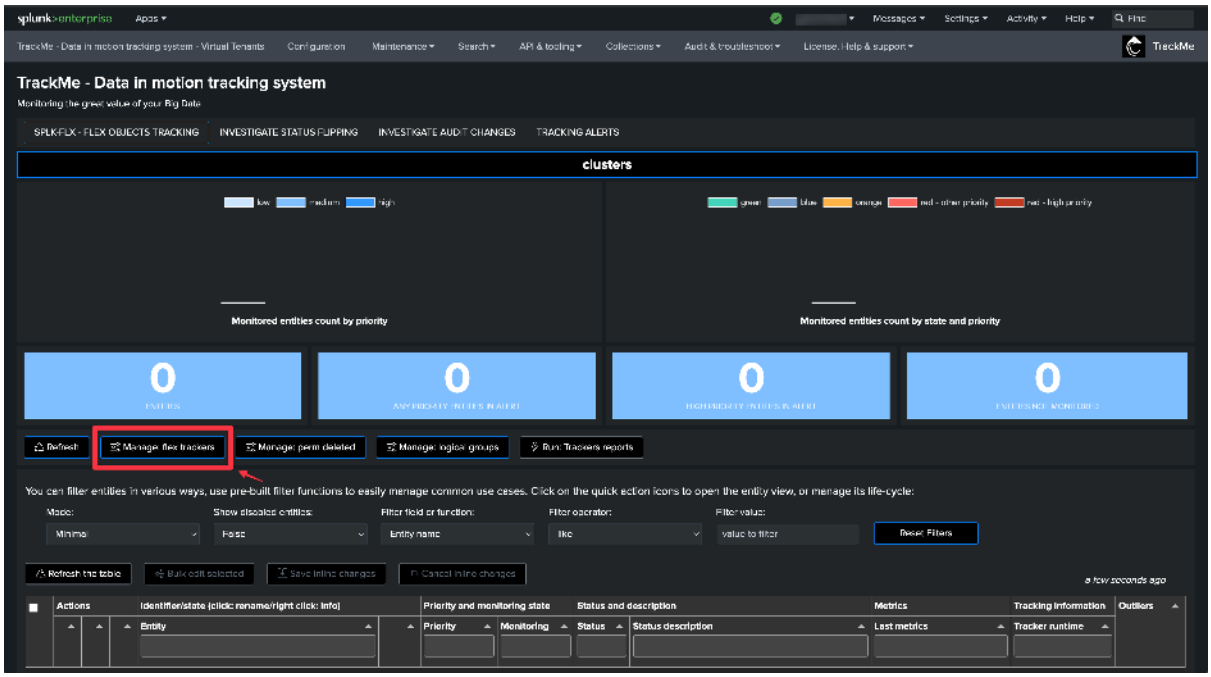
Find the use cases definition (SPL) in annex of this documentation

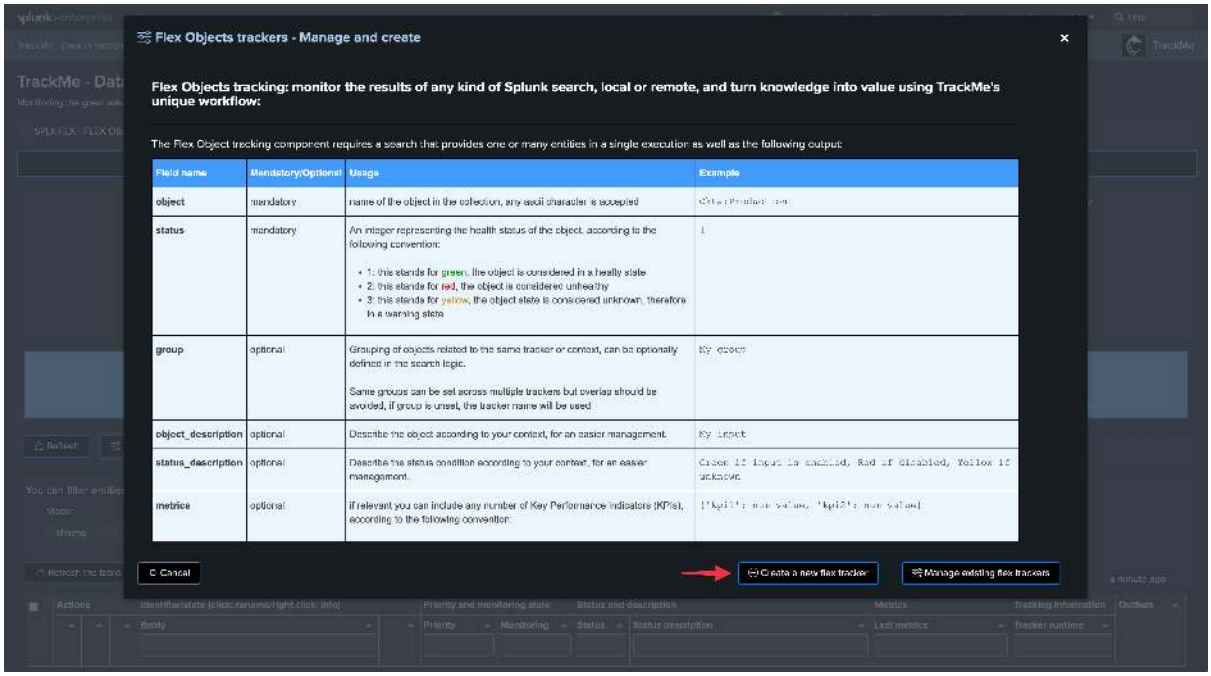
- See: *Annex: Use cases definition (SPL)*

Enter the newly created tenant and start the Flex Object wizard using the built-in Flex Object library:



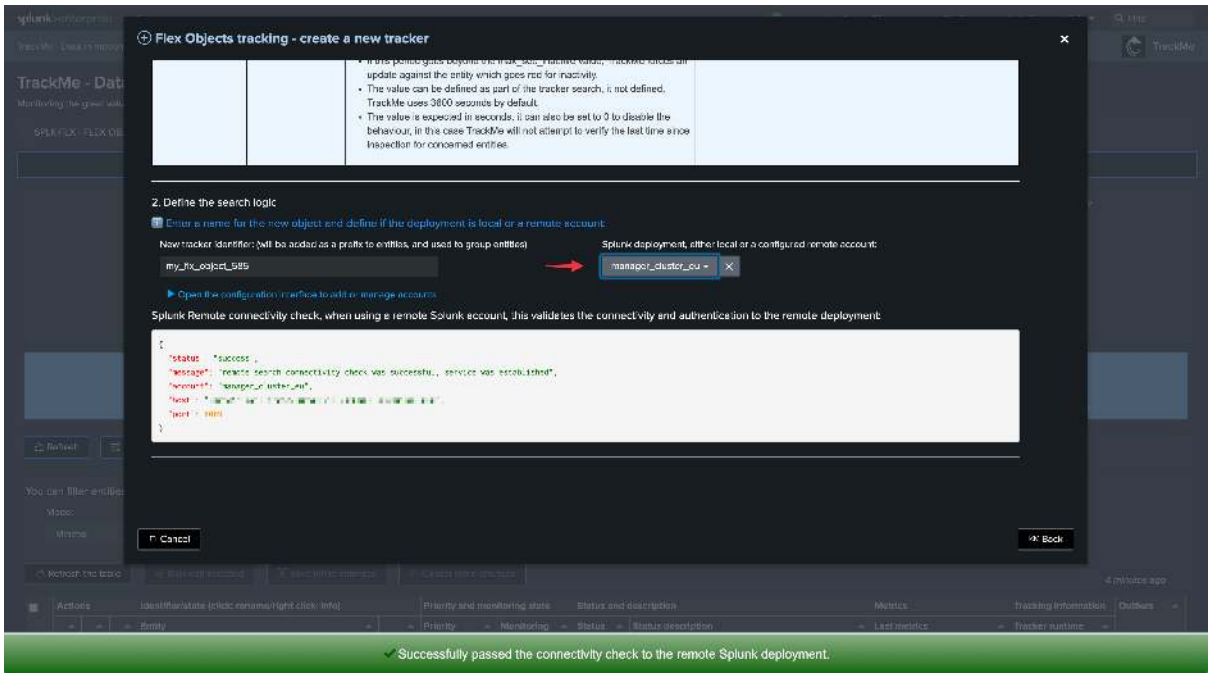

Tenant overview
Overview of the tenant id: clusters / alias: 11 - Indexer Clusters / Splunk Clustered Indexers
SPLK-FLX is enabled
Summary Index
Audit Index
Notable Index
Metrics Index
SPLK Flex Objects Tracking (SPLK-FLX) component overview
0 entities
Close
Refresh this
Manage this tenant
Hide this tenant from my profile
Overview of the tenant id: clusters / Splunk Clustered Indexers - Get tenant activity insights and manage its life cycle from this screen

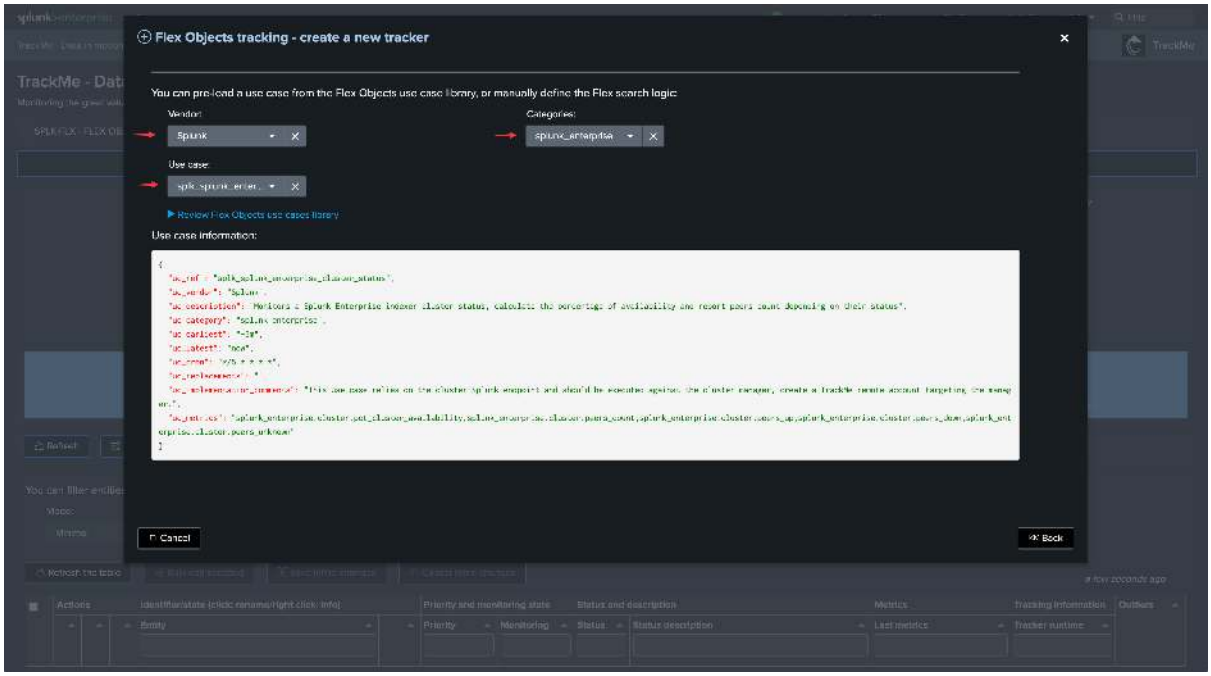

TrackMe - Data in motion tracking system
Monitoring the great value of your Big Data
SPLK-FLX - FLEX OBJECTS TRACKING
INVESTIGATE STATUS FLIPPING
INVESTIGATE AUDIT CHANGES
TRACKING ALERTS
clusters
Monitored entities count by priority
Monitored entities count by state and priority
Refresh
Manage flex trackers
Manage: perm deleted
Manage: logical groups
Run Trackers reports
You can filter entities in various ways, use pre-built filter functions to easily manage common use cases. Click on the quick action icons to open the entity view, or manage its life-cycle:
Mode:
Show disabled entities:
Filter field or function:
Filter operator:
Filter value:
Reset Filters
Refresh the table
Bulk edit selected
Save inline changes
Cancel inline changes
a few seconds ago
Actions
Identifier/state (click: rename/right click: info)
Entity
Priority and monitoring state
Priority
Monitoring
Status and description
Status
Status description
Metrics
Last metrics
Tracking information
Tracker runtime
Outliers

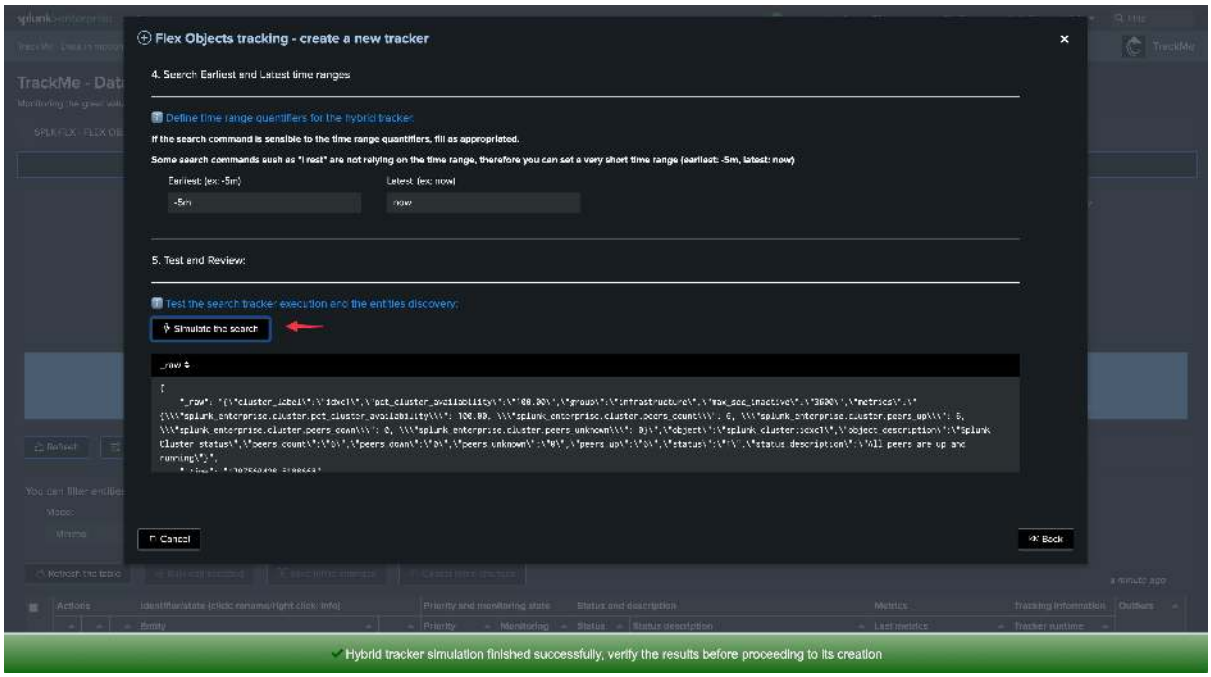


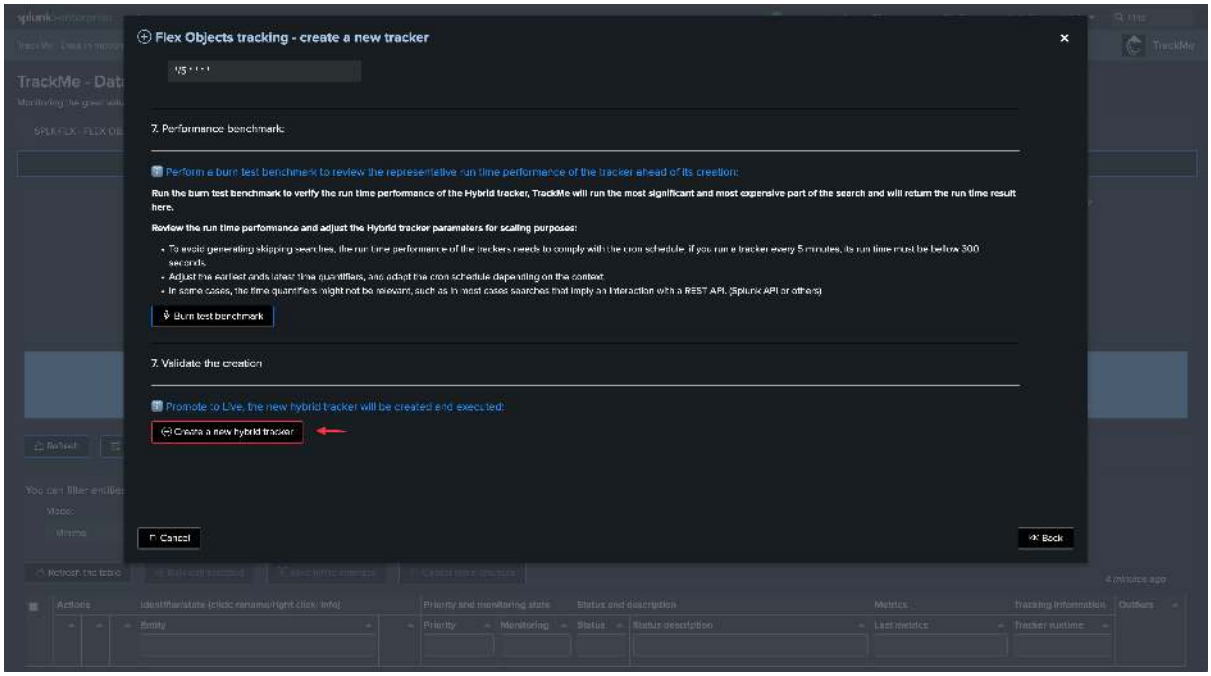
Select the use cases and remote accounts:

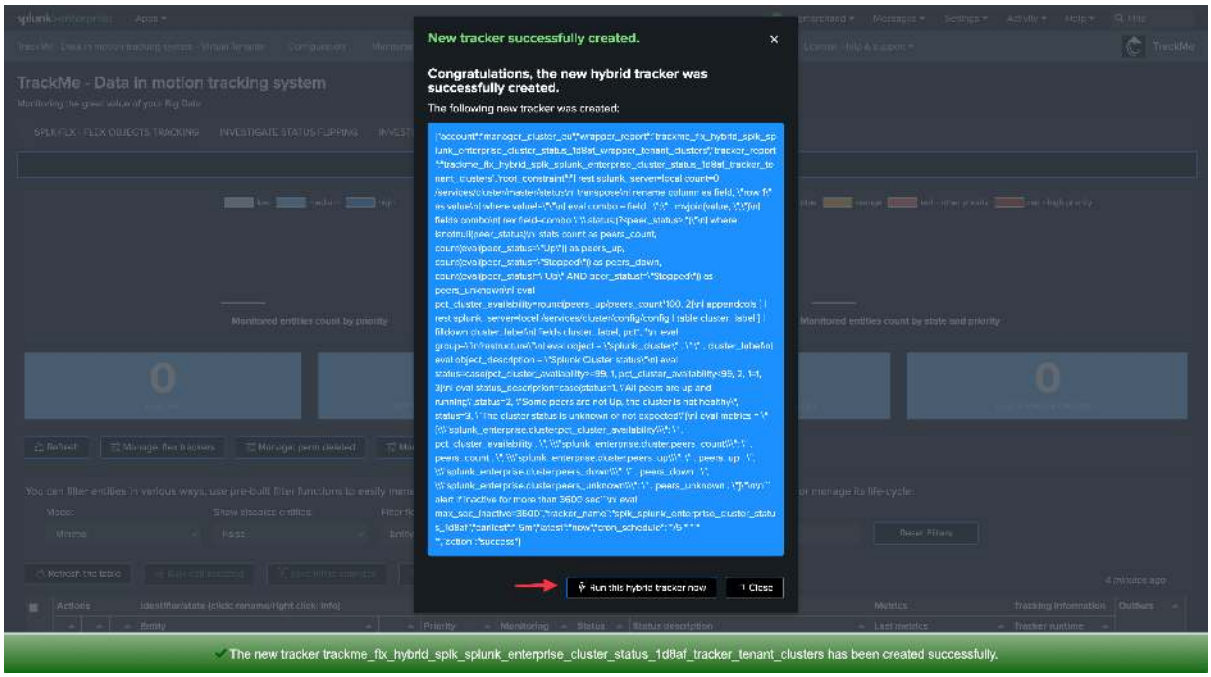
We will start with the use case "splk_splunk_enterprise_cluster_status" which tracks the global indexer cluster status using the manager API endpoints.



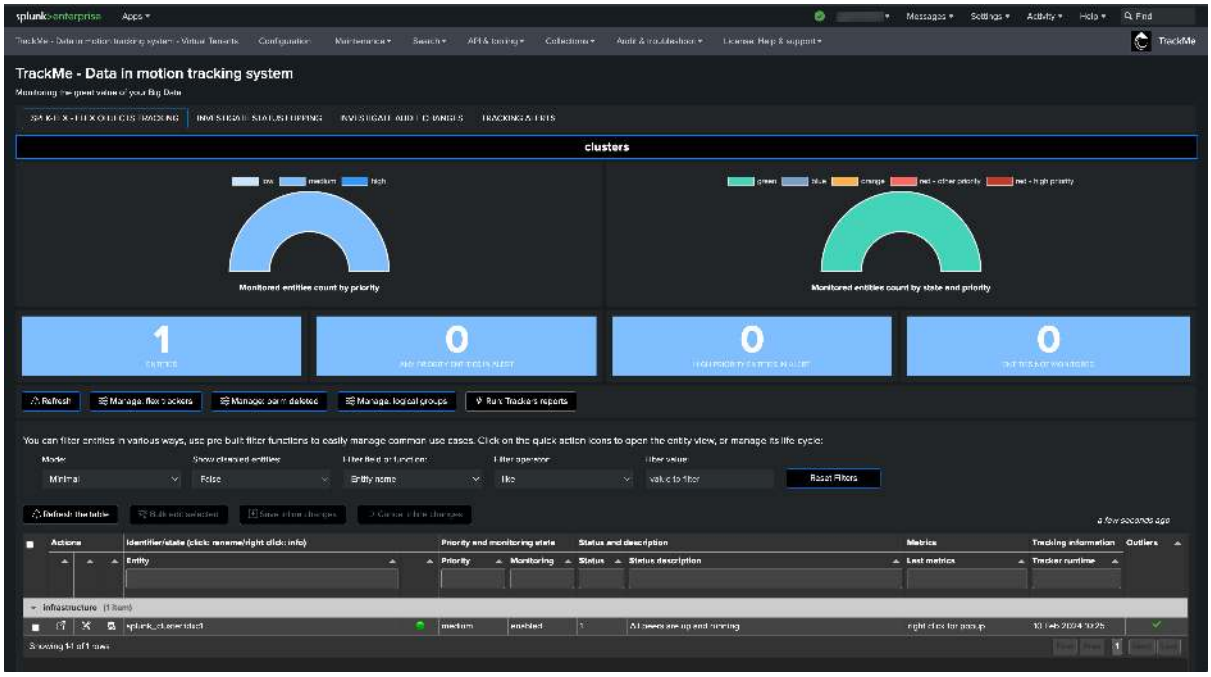

Flex Objects tracking - create a new tracker
You can pre-load a use case from the Flex Objects use case library, or manually define the Flex search logic:
Vendor:
Splunk
Categories:
splunk_enterprise
Use case:
Review Flex Objects use cases library
Use case information:
Cancel
Back

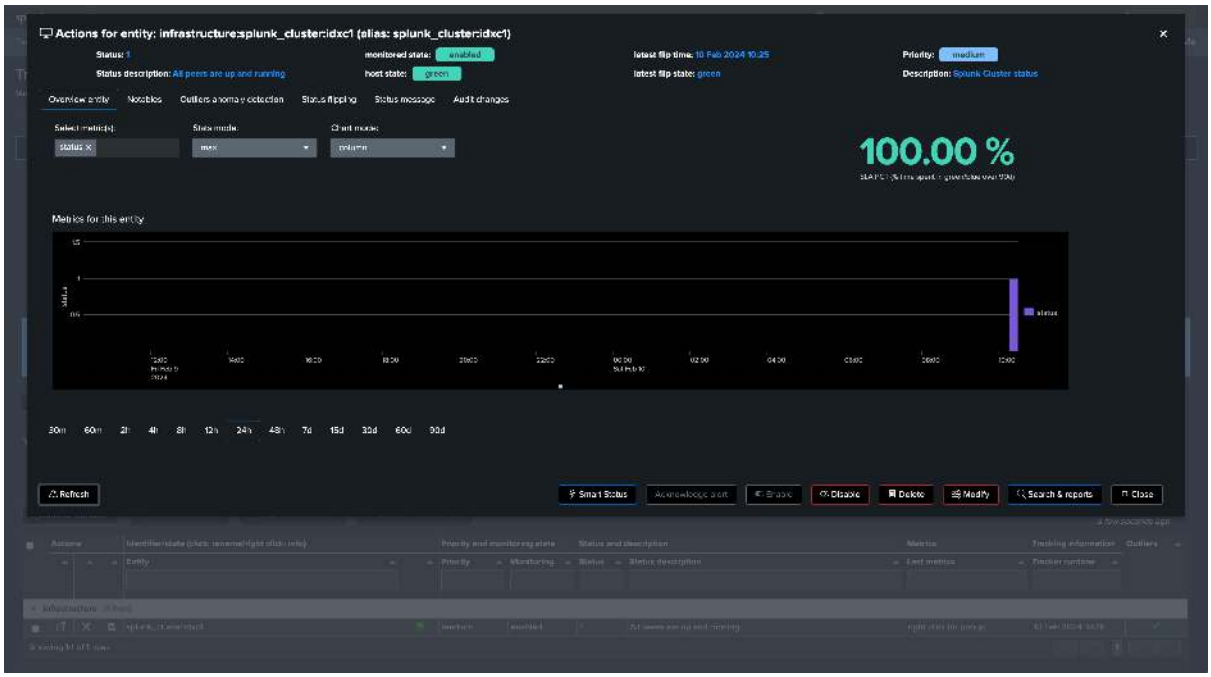

Flex Objects tracking - create a new tracker
4. Search Earliest and Latest time ranges
Define time range quantifiers for the hybrid tracker:
If the search command is sensible to the time range quantifiers, fill as appropriated.
Some search commands such as "| rest" are not relying on the time range, therefore you can set a very short time range (earliest: -5m, latest: now)
Earliest (ex: -5m)
Latest (ex: now)
-5m
now
5. Test and Review:
Test the search tracker execution and the entities discovery:
Simulate the search
Cancel
Back
Hybrid tracker simulation finished successfully, verify the results before proceeding to its creation

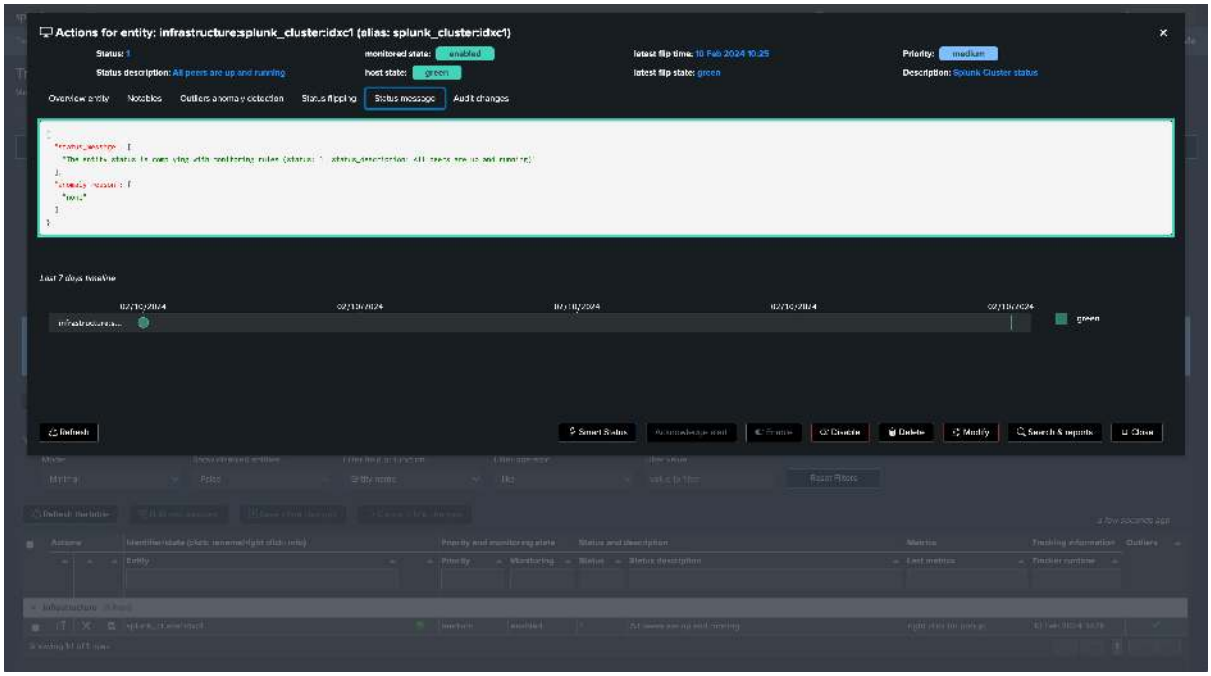


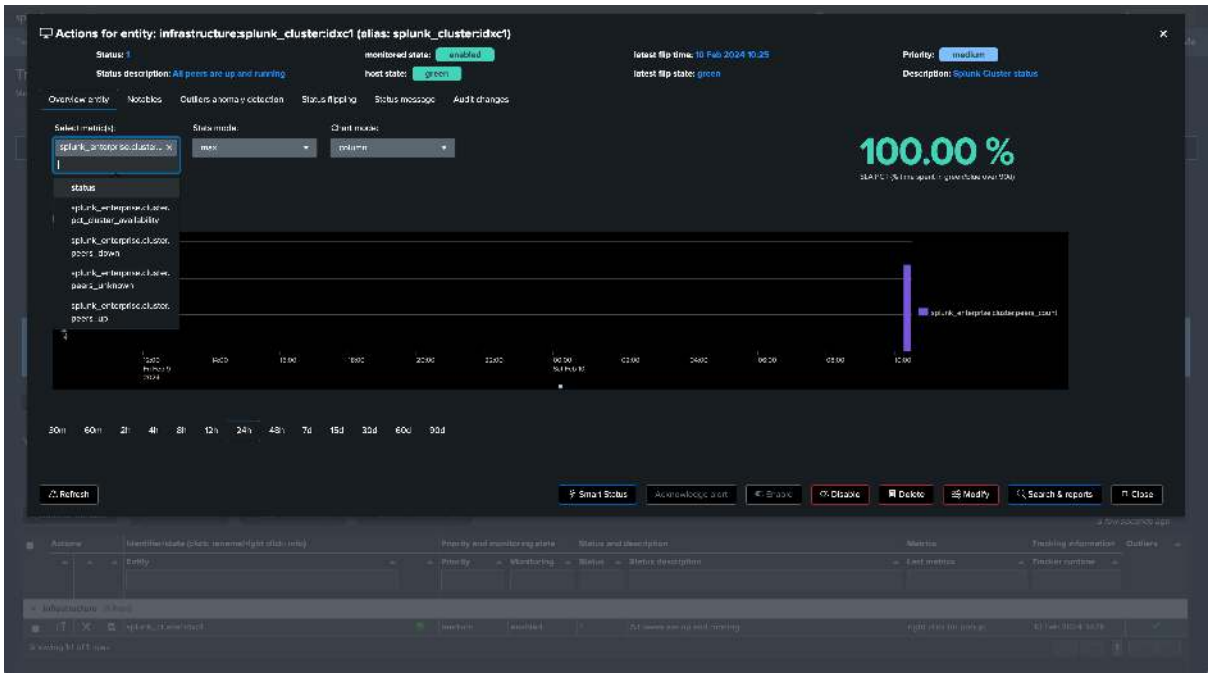


Once executed, TrackMe immediately starts monitoring the global indexer cluster status, generates KPIs for the cluster, and alerts you when issues are detected.

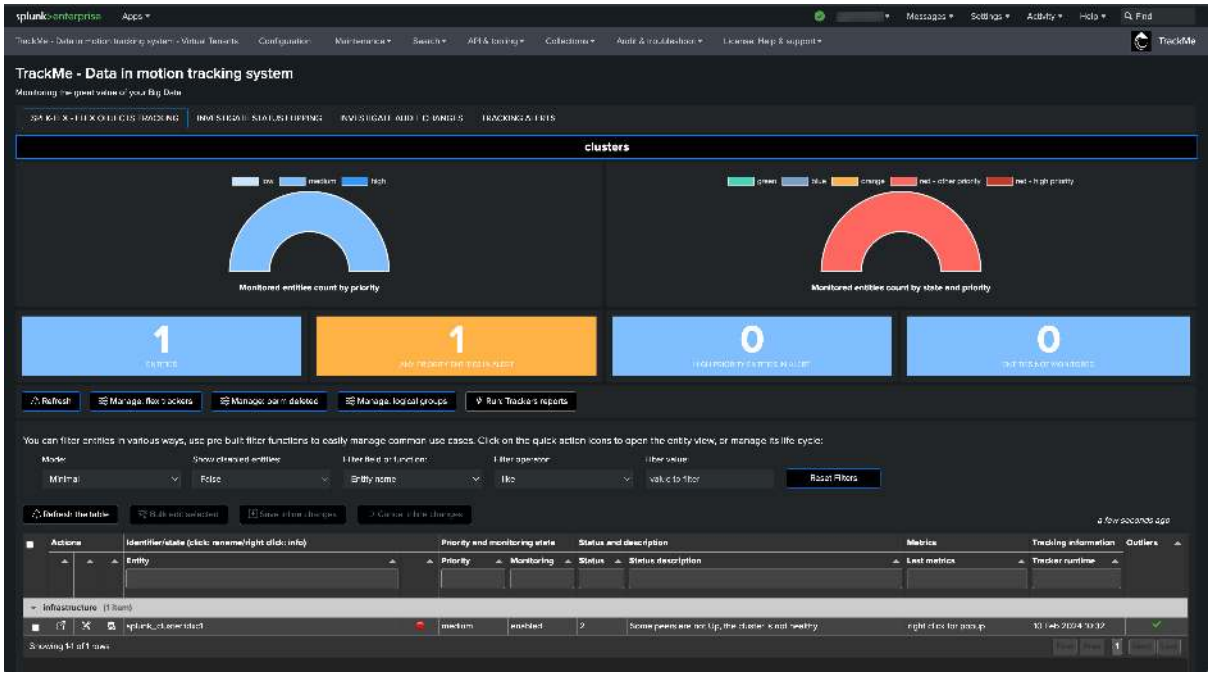

splunk>enterprise
TrackMe - Data in motion tracking system
clusters
Monitored entities count by priority
Monitored entities count by state and priority
1
0
0
0
Refresh
You can filter entities in various ways, use pre built filter functions to easily manage common use cases. Click on the quick action icons to open the entity view, or manage its life cycle:
Reset Filters
Actions
Entity
Priority
Monitoring
Status
Status description
Last metrics
Tracking information
Outliers
infrastructure
splunk_clusteridxc1
medium
enabled
All peers are up and running

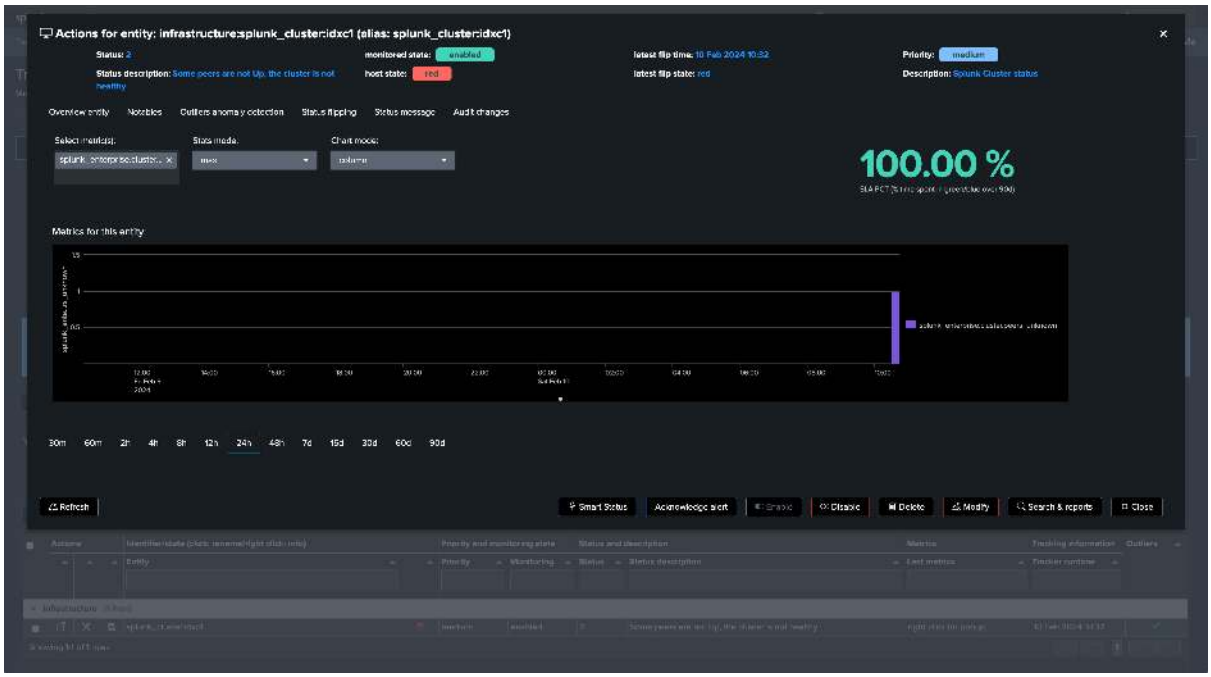

Actions for entity: infrastructure:splunk_clusteridxc1 (alias: splunk_clusteridxc1)
Status: 1
Status description: All peers are up and running
monitored state: enabled
host state: green
latest flip time: 10 Feb 2024 10:25
latest flip state: green
Priority: medium
Description: Splunk Cluster status
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
100.00 %
Metrics for this entity
Refresh
Smart Status
Disable
Delete
Modify
Search & reports
Close



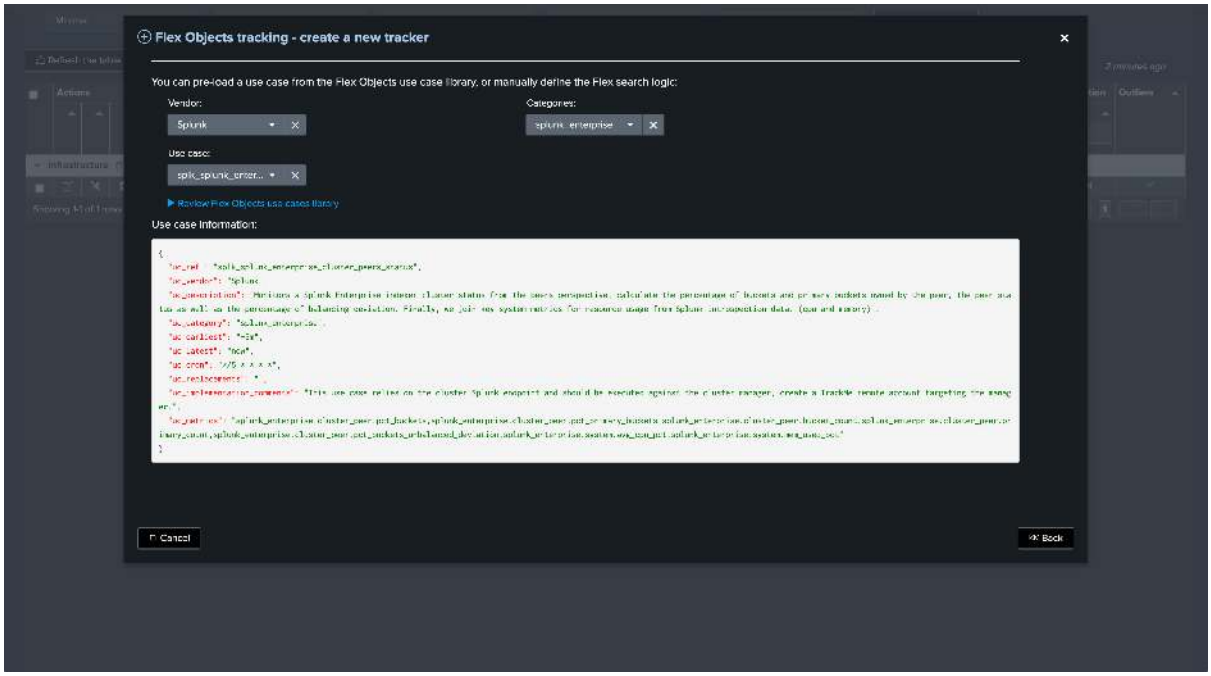


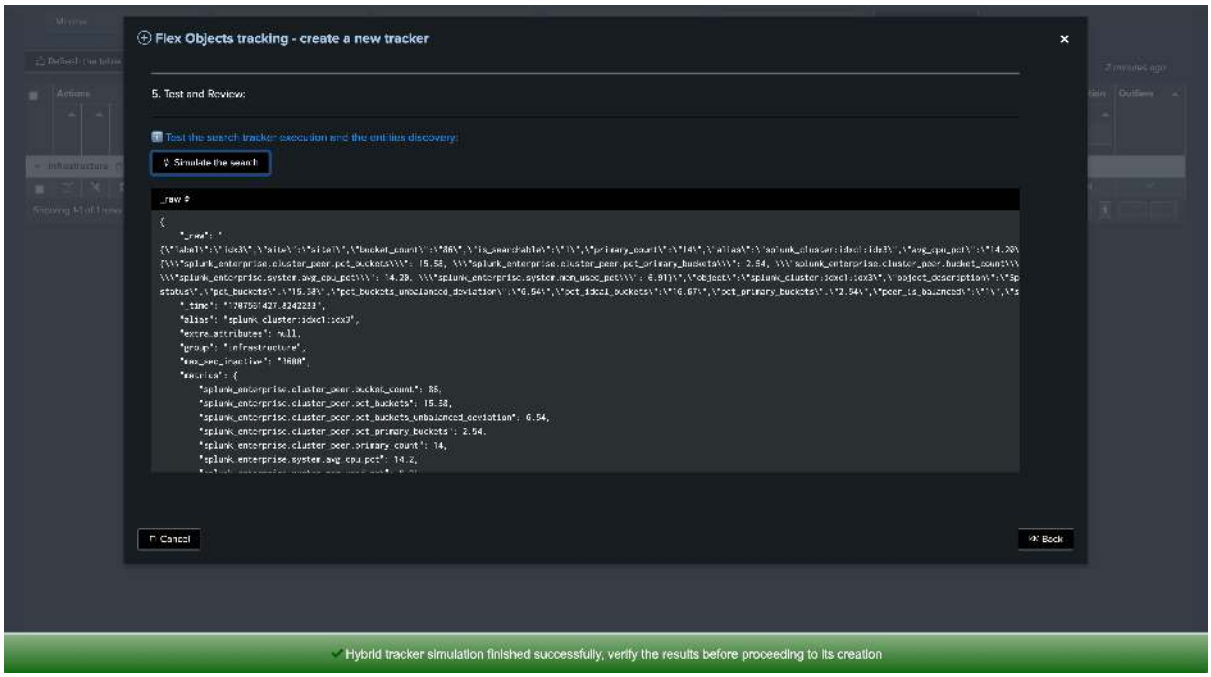
If a cluster peer goes down for any reason, the status of the TrackMe entity turns red and an alert is generated.



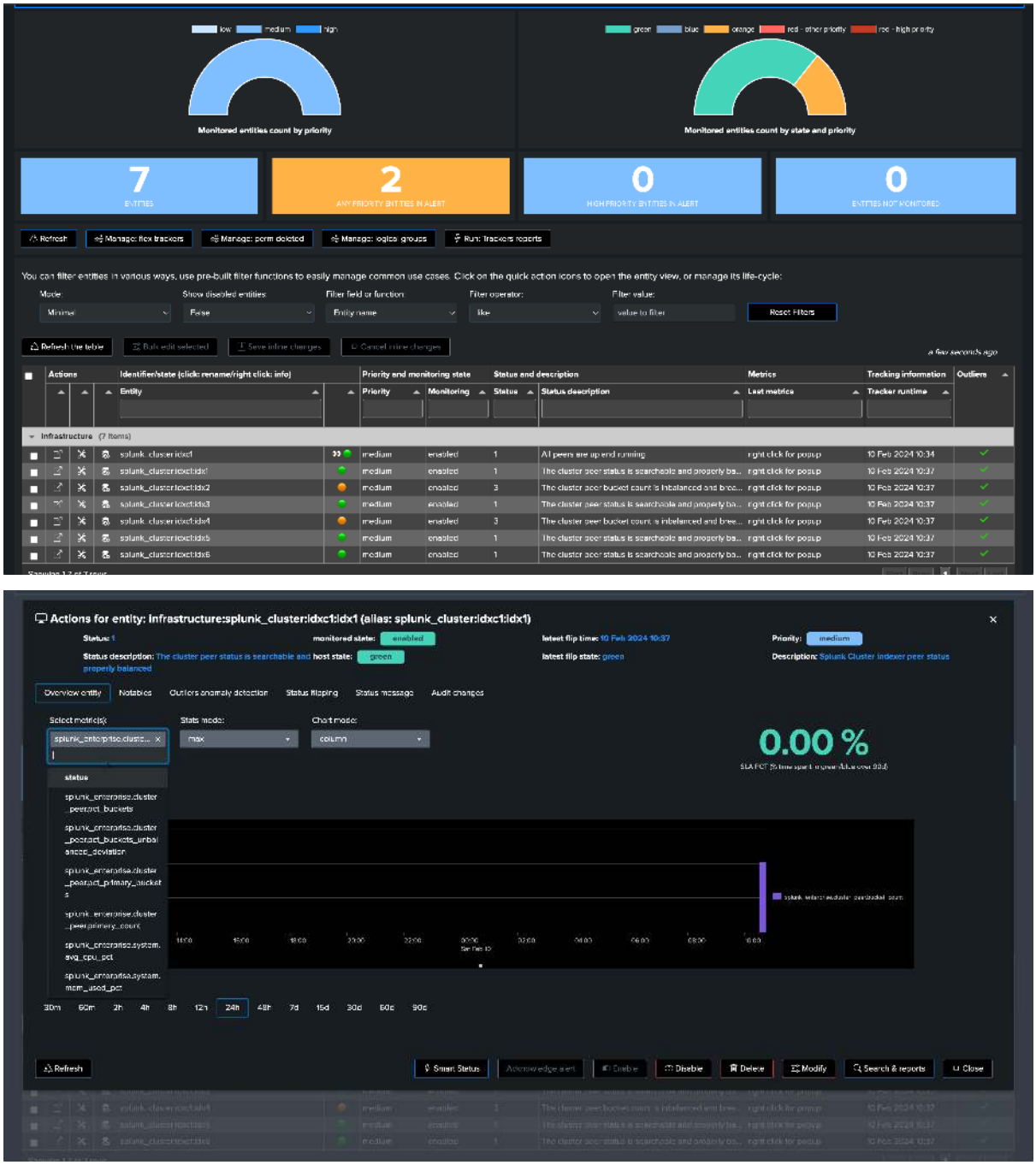


Let's now create a second Flex Object tracker using the library with the use case "splk_splunk_enterprise_cluster_peers" which tracks the peers' statuses individually.

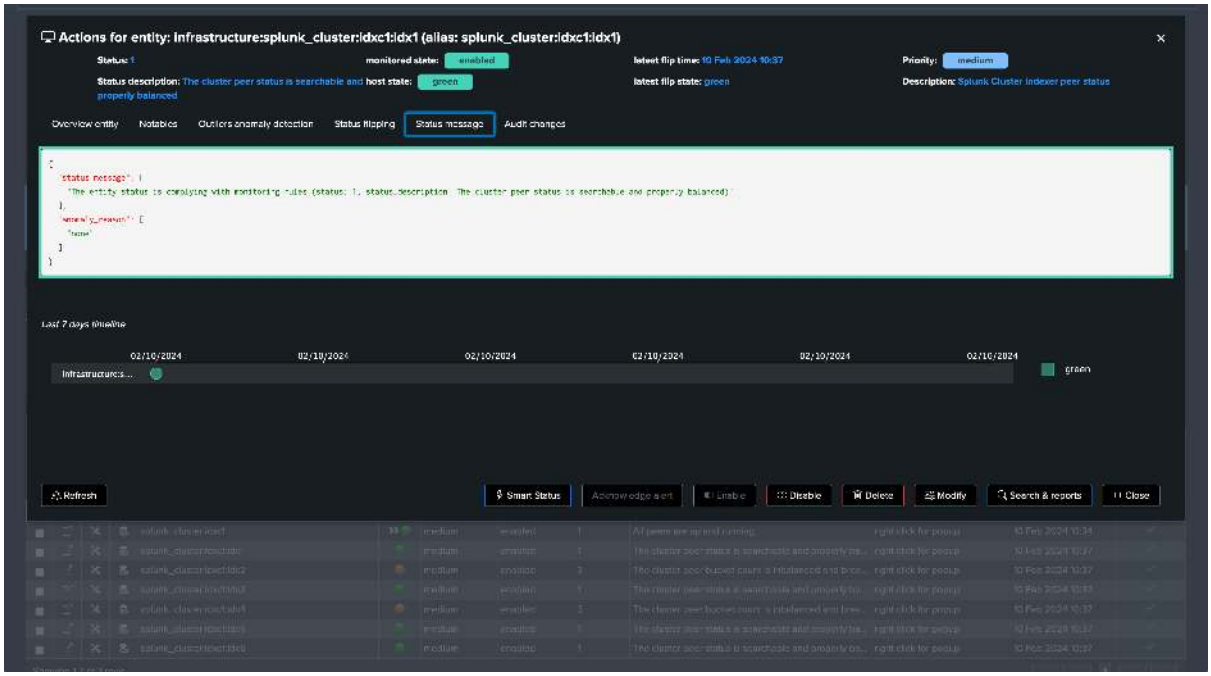


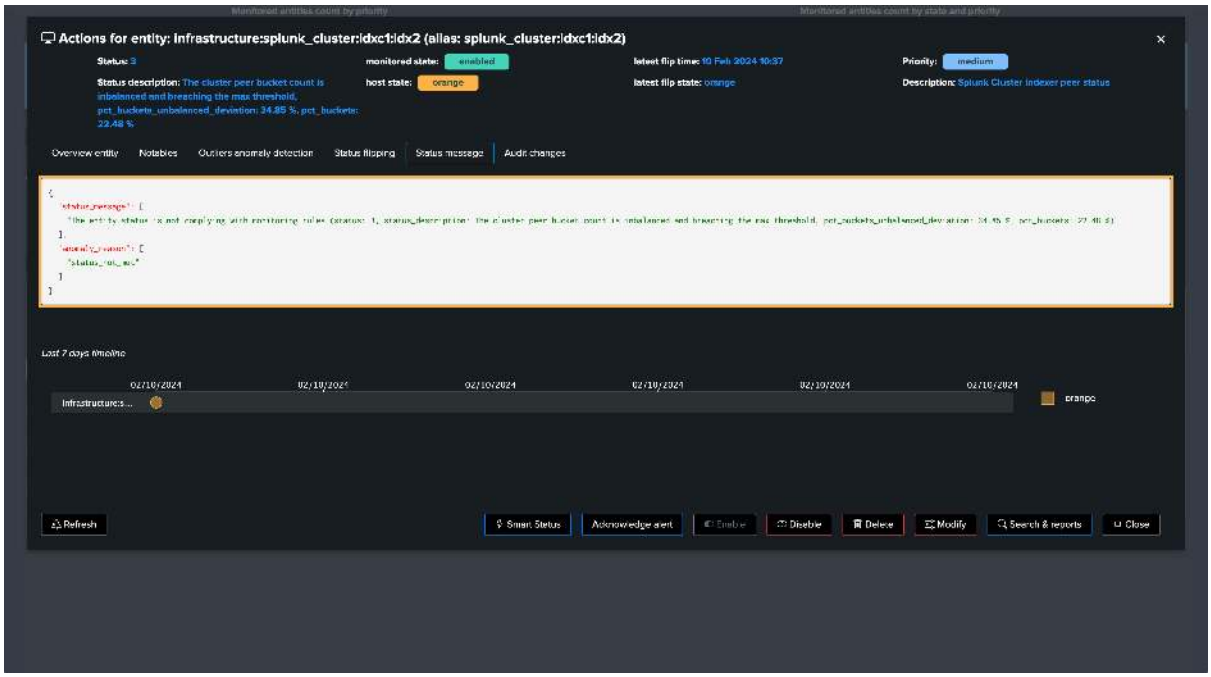


Once executed, one TrackMe entity is created per cluster peer, TrackMe starts monitoring the peer status and begins collecting main KPIs of interest for the peers.



Notably, TrackMe monitors the distribution of the buckets within the members and alerts you when the distribution is not properly balanced between the members.





You now have a fully monitored Indexer Cluster, enabling TrackMe to continuously monitor the health of your Splunk Indexer Clusters, track SLA availability, and easily alert you when issues are detected on your Splunk indexers cluster(s) infrastructure.

Annex: Use cases definition (SPL)

Use case splk_splunk_enterprise_cluster_status

```
| rest splunk_server=local count=0 /services/cluster/master/status
| transpose
| rename column as field, "row 1" as value
| where value!=""
| eval combo = field . ":" . mvjoin(value, ",")
| fields combo
```

(continues on next page)

(continued from previous page)

```
| rex field=combo "\.status:(?<peer_status>.*)"
| where isnotnull(peer_status)
| stats count as peers_count, count(eval(peer_status="Up")) as peers_up,␣
↪count(eval(peer_status="Stopped")) as peers_down, count(eval(peer_status!="Up"␣
↪AND peer_status!="Stopped")) as peers_unknown
| eval pct_cluster_availability=round(peers_up/peers_count*100, 2)
| appendcols [ | rest splunk_server=local /services/cluster/config/config | table␣
↪cluster_label ] | filldown cluster_label
| fields cluster_label, pct*, *
| eval group="infrastructure"
| eval object = "splunk_cluster" . ":" . cluster_label
| eval object_description = "Splunk Cluster status"
| eval status=case(pct_cluster_availability>=99, 1, pct_cluster_availability<99,␣
↪2, 1=1, 3)
| eval status_description=case(status=1, "All peers are up and running",status=2,
↪"Some peers are not Up, the cluster is not healthy", status=3, "The cluster␣
↪status is unknown or not expected")
| eval metrics = "{\"splunk_enterprise.cluster.pct_cluster_availability\": " .␣
↪pct_cluster_availability . ", \"splunk_enterprise.cluster.peers_count\": " .␣
↪peers_count . ", \"splunk_enterprise.cluster.peers_up\": " . peers_up . ", \
↪"splunk_enterprise.cluster.peers_down\": " . peers_down . ", \
↪"splunk_enterprise.cluster.peers_unknown\": " . peers_unknown . "}"

``` alert if inactive for more than 3600 sec```
| eval max_sec_inactive=3600
```

Use case 'splk_splunk_enterprise_cluster_peers'

```
| rest splunk_server=local count=0 /services/cluster/master/peers
| fields label site bucket_count is_searchable primary_count

``` aggreg ```
| eventstats sum(bucket_count) as total_bucket_count, dc(label) as dcount_peers

``` calculate percentage of buckets ```
| eval pct_buckets = round(bucket_count/total_bucket_count*100, 2),␣
↪pct_primary_buckets = round(primary_count/total_bucket_count*100, 2),␣
↪pct_ideal_buckets=round(100/dcount_peers, 2)
| eval pct_buckets_unbalanced_deviation = round(abs(pct_buckets -␣
↪pct_ideal_buckets)/pct_ideal_buckets*100, 2)

``` set the cluster_label automatically ```
| appendcols [ | rest splunk_server=local /services/cluster/config/config | table␣
↪cluster_label ] | filldown cluster_label

``` set group ```
| eval group="infrastructure"

``` set object, alias and object_description ```
| eval object = "splunk_cluster" . ":" . cluster_label . ":" . label, alias=object
| eval object_description = "Splunk Cluster indexer peer status"

``` set status, if the peer is not searchable, then it is red ```
| eval status=case(is_searchable=1, 1, is_searchable!=1, 2, 1=1, 3)
```

(continues on next page)
```

(continued from previous page)

```
``` if the peer buckets deviation percentage is higher than 25%, then the peer is␣
↪highly inbalanced, we set the status to yellow, but you can also set this to red␣
↪if you prefer ```
| eval peer_is_balanced=if(pct_buckets_unbalanced_deviation>25, 0, 1)
``` yellow if breached, set to red if preferred ```
| eval status=if(peer_is_balanced!=1, 3, status)

``` join external CPU metrics ```
| join type=outer label [ search index=_introspection␣
↪sourcetype=splunk_resource_usage component=Hostwide
| eval cpu_total_pct=('data.cpu_system_pct'+'data.cpu_user_pct'), mem_used_pct=(
↪'data.mem_used'/'data.mem'*100)
| stats avg(cpu_total_pct) as avg_cpu_pct, avg(mem_used_pct) as mem_used_pct by␣
↪host
| eval avg_cpu_pct=round(avg_cpu_pct, 2), mem_used_pct=round(mem_used_pct, 2) |␣
↪rename host as label ]
| eval avg_cpu_pct=if(isnum(avg_cpu_pct), avg_cpu_pct, 0),␣
↪mem_used_pct=if(isnum(mem_used_pct), mem_used_pct, 0)

``` set status_description ```
| eval status_description=case(
status=1 AND is_searchable=1 AND pct_buckets_unbalanced_deviation<=25, "The␣
↪cluster peer status is searchable and properly balanced",
status=2 AND is_searchable!=1, "The cluster peer status is not searchable",
(status=2 OR status=3) AND peer_is_balanced!=1, "The cluster peer bucket count is␣
↪inbalanced and breaching the max threshold, pct_buckets_unbalanced_deviation: " .
↪ pct_buckets_unbalanced_deviation . " %, pct_buckets: " . pct_buckets . " %",
status=3, "The cluster peer status is unknown or not expected"
)

``` set metrics ```
| eval metrics = "{\"splunk_enterprise.cluster_peer.pct_buckets\": " . pct_buckets␣
↪. ", \"splunk_enterprise.cluster_peer.pct_primary_buckets\": " .␣
↪pct_primary_buckets . ", \"splunk_enterprise.cluster_peer.bucket_count\": " .␣
↪bucket_count . ", \"splunk_enterprise.cluster_peer.primary_count\": " .␣
↪primary_count . ", \
↪"splunk_enterprise.cluster_peer.pct_buckets_unbalanced_deviation\":" .␣
↪pct_buckets_unbalanced_deviation . ", \"splunk_enterprise.system.avg_cpu_pct\":
↪" . avg_cpu_pct  . ", \"splunk_enterprise.system.mem_used_pct\": " .␣
↪mem_used_pct . "}"

``` alert if inactive for more than 3600 sec```
| eval max_sec_inactive=3600
```

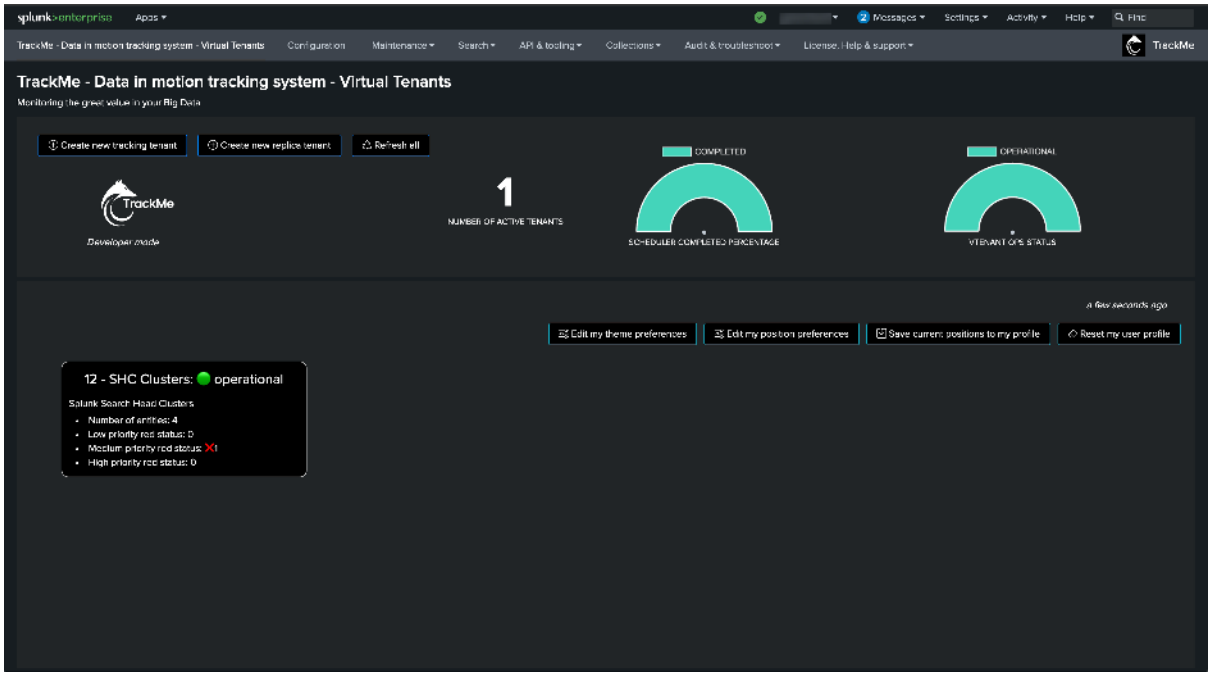
9.3.4 Monitor Splunk Search Head Clusters

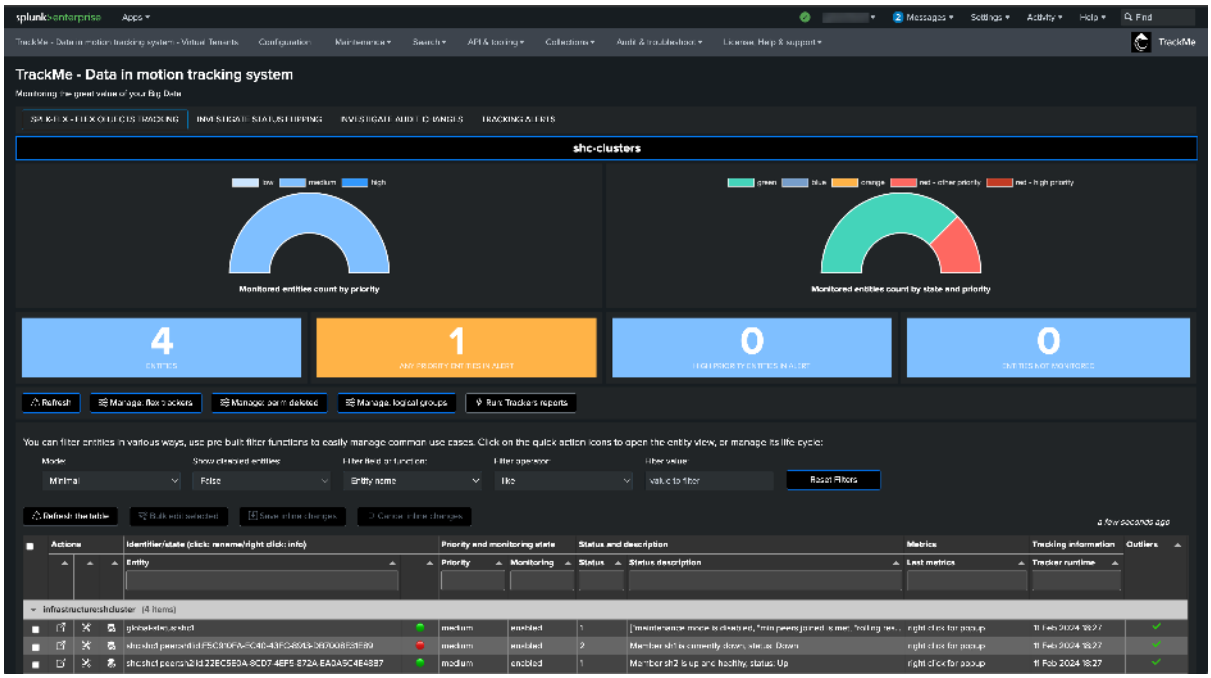
Monitoring Splunk Search Head Clusters with TrackMe Flex Objects

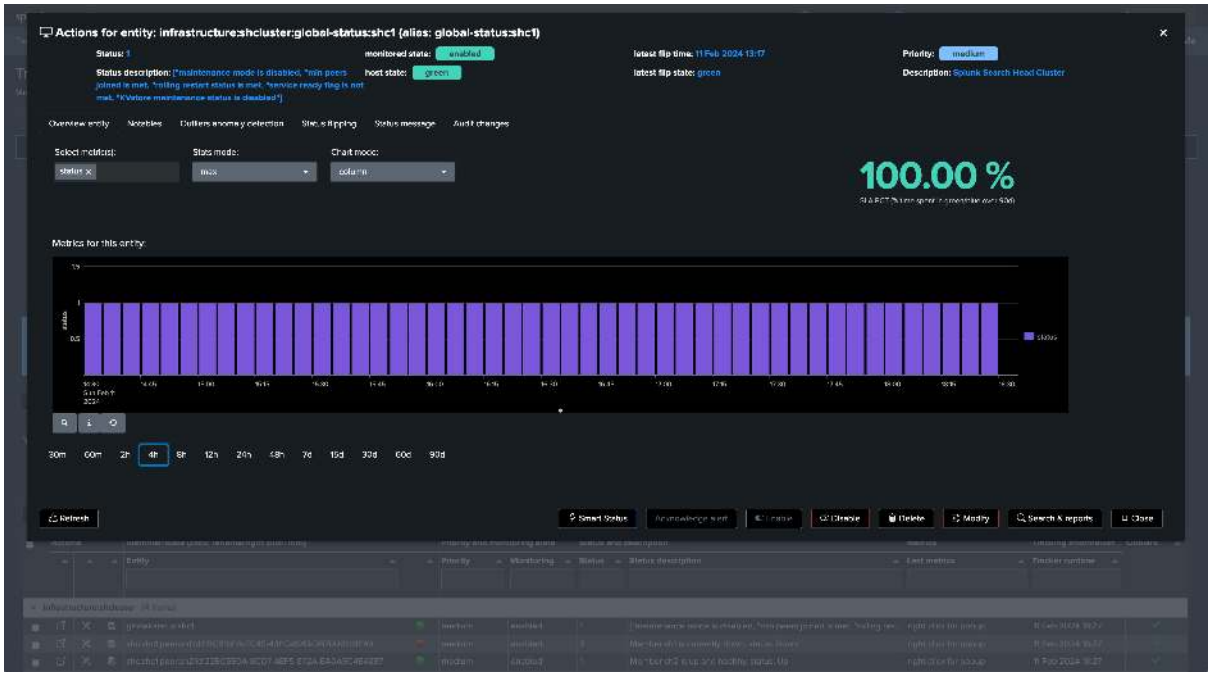
- This tutorial demonstrates the monitoring of **Splunk Search Head Clusters** with TrackMe Flex Objects.
- TrackMe Flex Objects is a component restricted to licensed customers. Please contact Splunk Sales for more information.
- Using these steps will enable TrackMe to continuously monitor the health of your Splunk Search Head Clusters and alert you when issues are detected.
- With TrackMe's remote search capabilities, you can monitor as many Search Head Clusters as you need

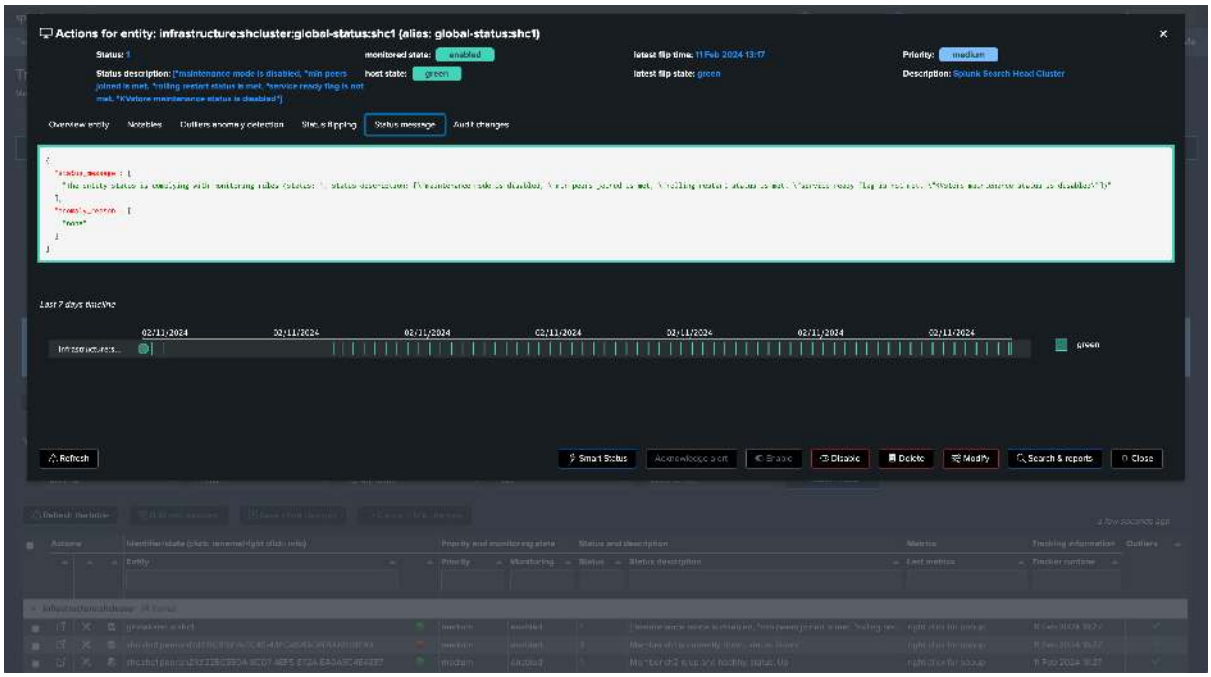
from a single pane of glass.

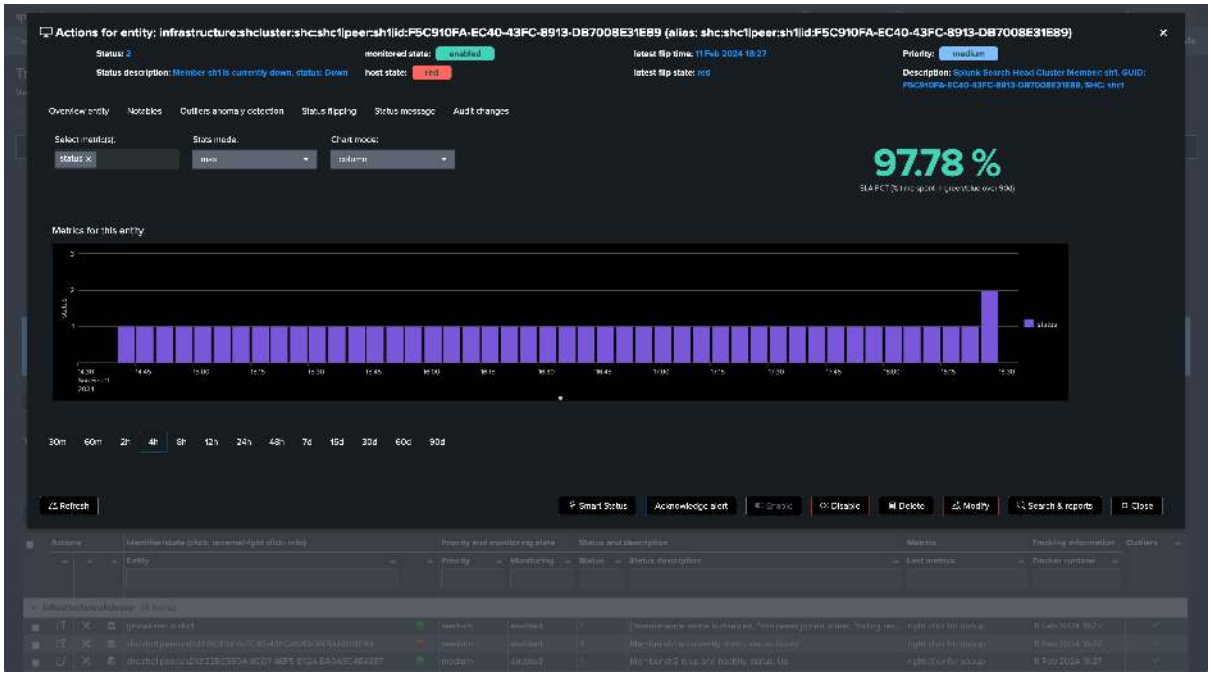
The following screen shows the final results in TrackMe, starting from Step 1 to easily implement the monitoring of your Splunk Search Head Clusters.

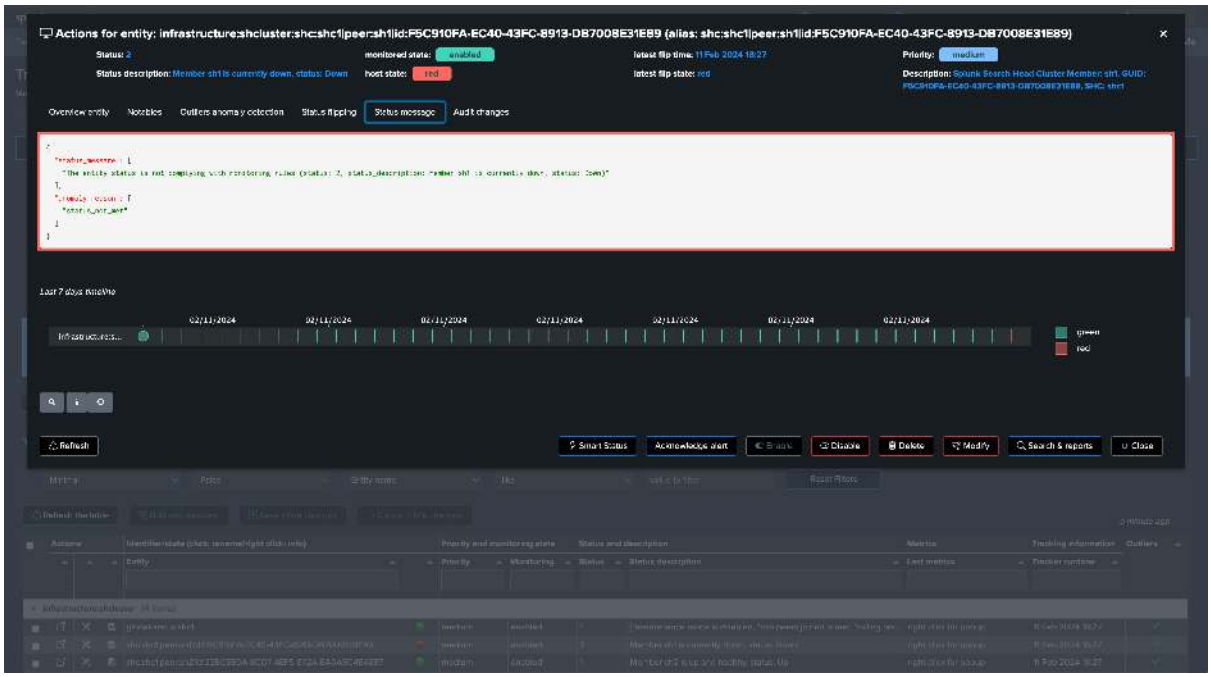




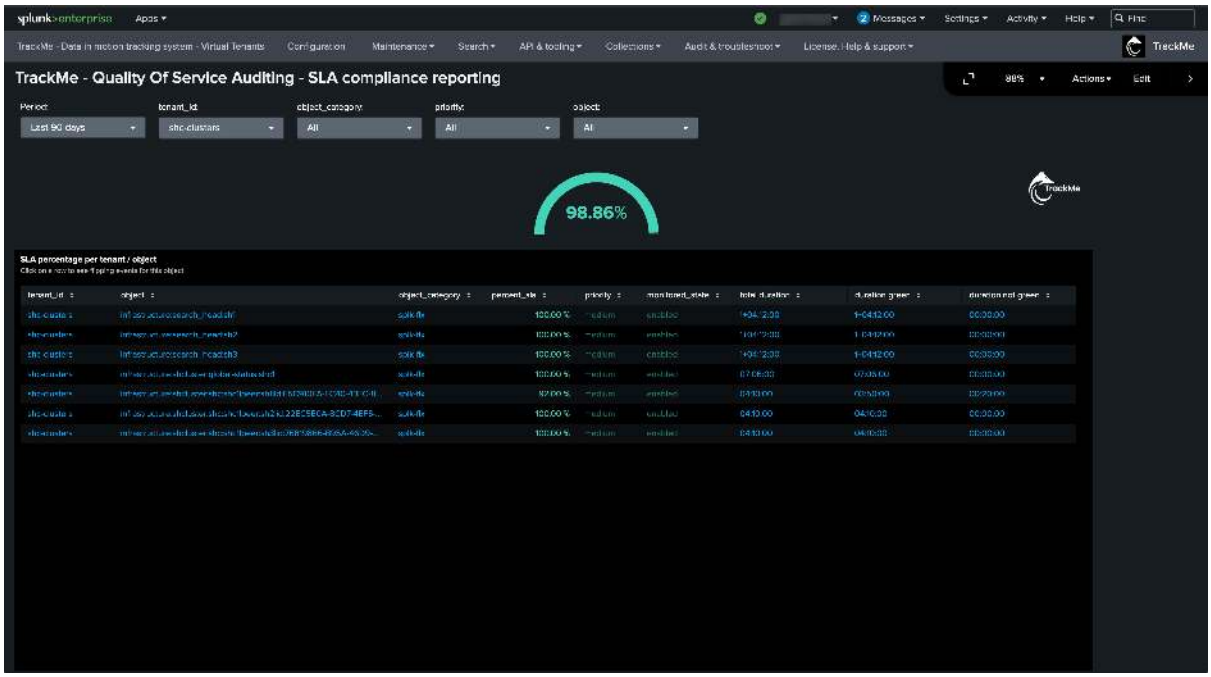

Actions for entity: infrastructure:shcluster:global-status:shc1 (alias: global-status:shc1)
100.00 %
Refresh
Smart Status
Search & reports
Close


Actions for entity: infrastructure:shcluster:global-status:shc1 (alias: global-status:shc1)
Refresh
Smart Status
Search & reports
Close


Actions for entity: infrastructure:shcluster:shc:shc1|peer:sh1|id:F5C910FA-EC40-43FC-8913-DB7008E31E89 (alias: shc:shc1|peer:sh1|id:F5C910FA-EC40-43FC-8913-DB7008E31E89)
Status: 2
monitored state: enabled
latest flip time: 11 Feb 2024 18:27
Priority: medium
Status description: Member sh1 is currently down, status: Down
host state: red
latest flip state: red
Description: Splunk Search Head Cluster Member: sh1, GUID: F5C910FA-EC40-43FC-8913-DB7008E31E89, SHC: shc1
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
Select metric(s):
Stats mode:
Chart mode:
Status
mean
column
97.78 %
Metrics for this entity
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
Refresh
Smart Status
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close


Actions for entity: infrastructure:shcluster:shc:shc1|peer:sh1|id:F5C910FA-EC40-43FC-8913-DB7008E31E89 (alias: shc:shc1|peer:sh1|id:F5C910FA-EC40-43FC-8913-DB7008E31E89)
Status: 2
monitored state: enabled
latest flip time: 11 Feb 2024 18:27
Priority: medium
Status description: Member sh1 is currently down, status: Down
host state: red
latest flip state: red
Description: Splunk Search Head Cluster Member: sh1, GUID: F5C910FA-EC40-43FC-8913-DB7008E31E89, SHC: shc1
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
Last 7 days timeline
green
red
Refresh
Smart Status
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close





Step 1: Create a Splunk Remote Deployment Account

Hint

Interacting with Splunk API through TrackMe's remote search capabilities

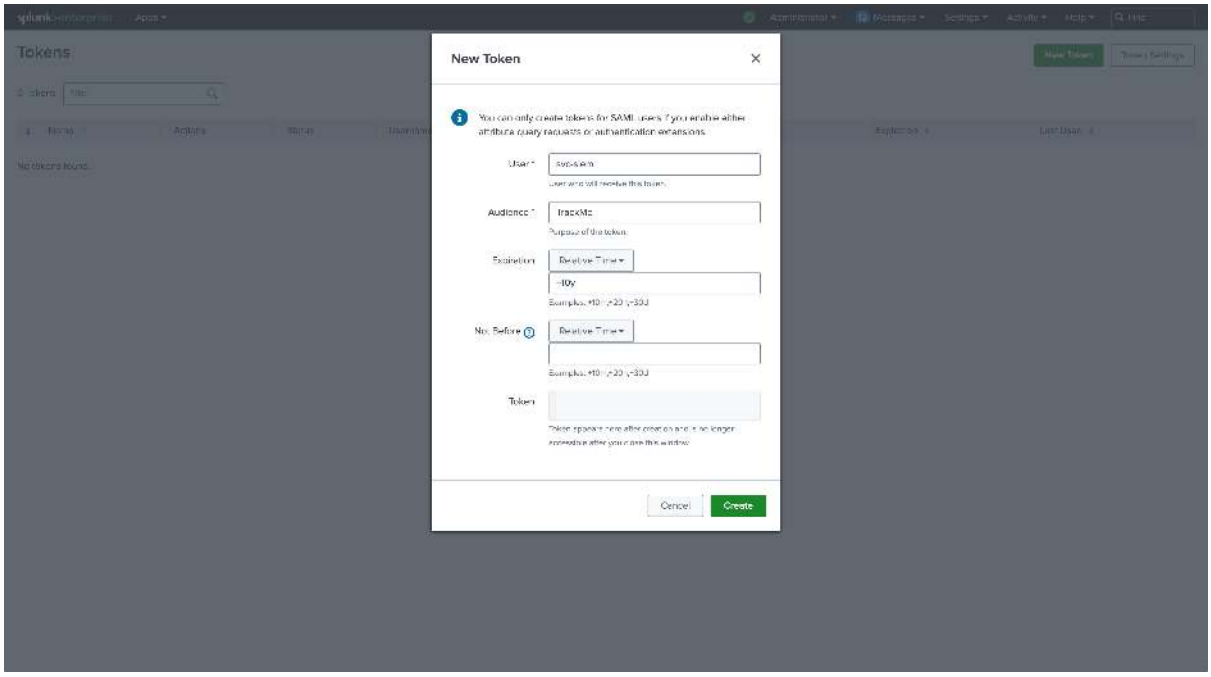
- This use case relies on a Splunk | *rest* command which is a Splunk REST API call through native SPL.
- Therefore, this command needs to be executed on a remote Splunk target, in the context of monitoring Search Head Clusters, the command can be executed on any up and running SHC member.
- Using TrackMe's remote search capabilities, you can transparently execute the command on a remote target, and process results locally on the TrackMe guest.

The first step is to create a Splunk Remote Deployment Account for the Cluster Manager. For more infor-

mation about TrackMe Remote Search capabilities and configurations:

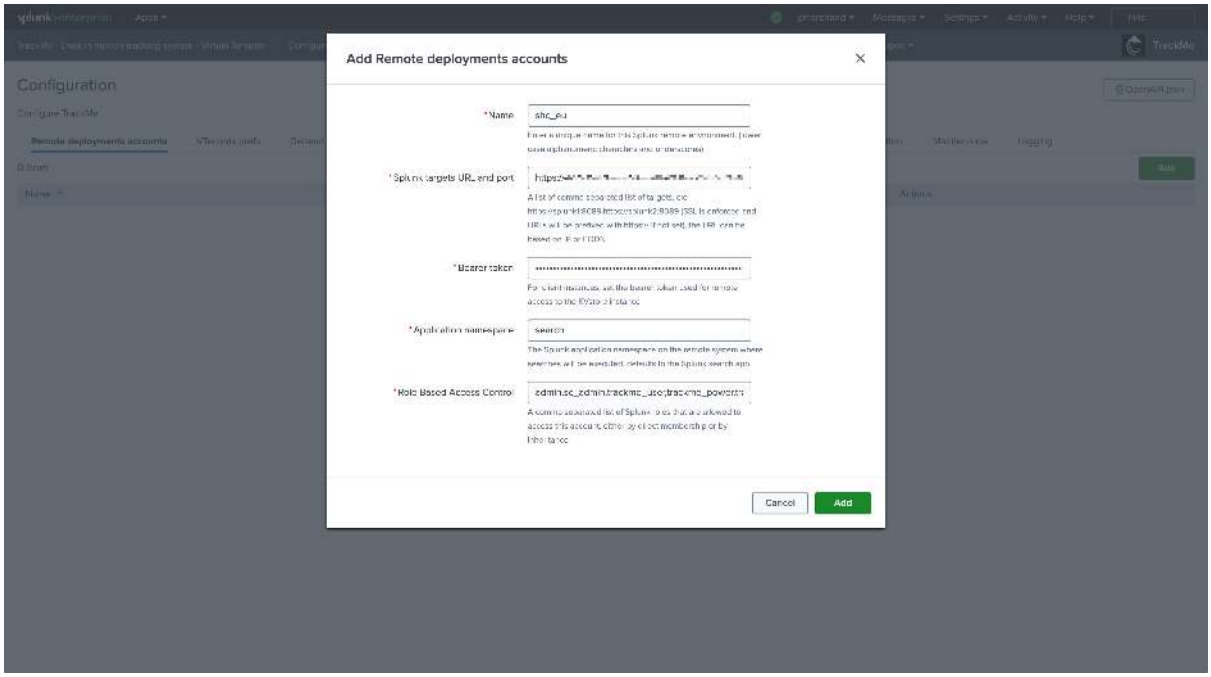
Remote Splunk deployments

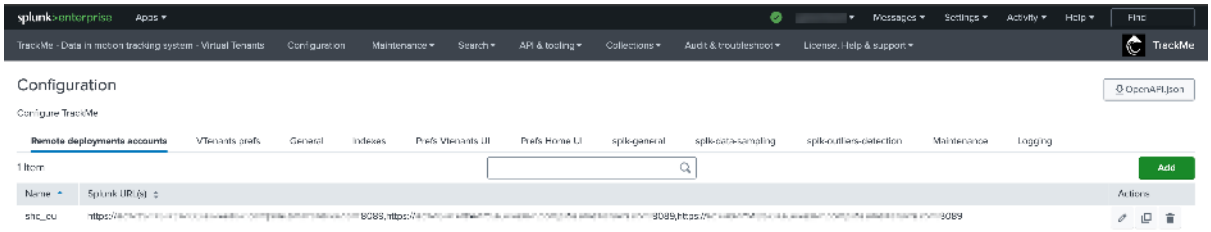
On the Search Head Cluster, create a new Splunk bearer token for the TrackMe Remote Deployment Account:



In TrackMe, click on Configure / Remote Deployment Accounts and add a new account:

- You can specify each of the SHC members in a comma separated list or use a load balancer URL.
- If multiple endpoints are specified, TrackMe automatically dispatches searches randomly amongst available and responding members (it validates connectivity and authentication)





Hint

Managing multiple Search Head Clusters

- If you have multiple Search Head Clusters, you can create a Remote Deployment Account for each SHC.
- You will then be able to manage and monitor as many Search Head Clusters as you need from a single pane of glass in TrackMe.

Step 2: Create a Flex Object tenant for Search Head Clusters monitoring

Now, let's create a new tenant for the purposes of monitoring our Search Head Clusters. We can achieve this via the UI wizard or with a simple command line:

```
| trackme url=/services/trackme/v2/vtenants/admin/add_tenant mode=post body="{
↪'tenant_name': 'shc-clusters', 'tenant_alias': '12 - SHC Clusters', 'tenant_desc
↪': 'Splunk Search Head Clusters', 'tenant_roles_admin': 'trackme_admin',
↪'tenant_roles_power': 'trackme_power', 'tenant_roles_user': 'trackme_user',
↪'tenant_owner': 'admin', 'tenant_idx_settings': 'global', 'tenant_flx_enabled':
↪'true'}"
```

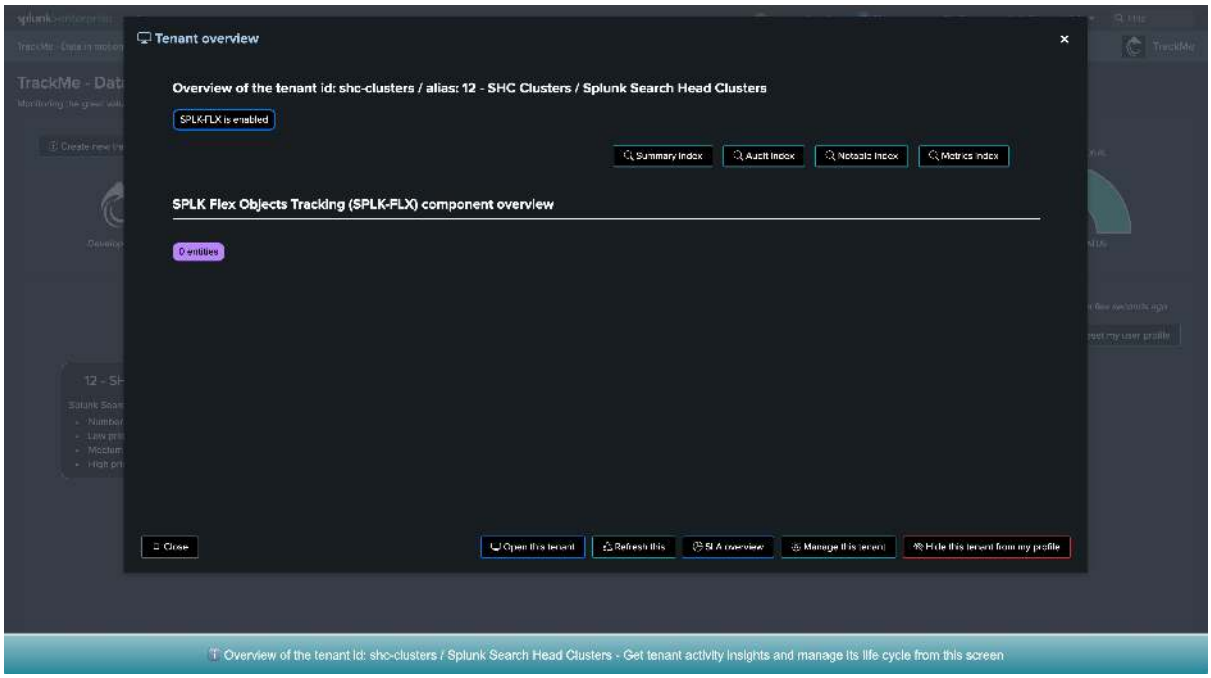
Step 3: Create Flex Object trackers using the Flex Objects library

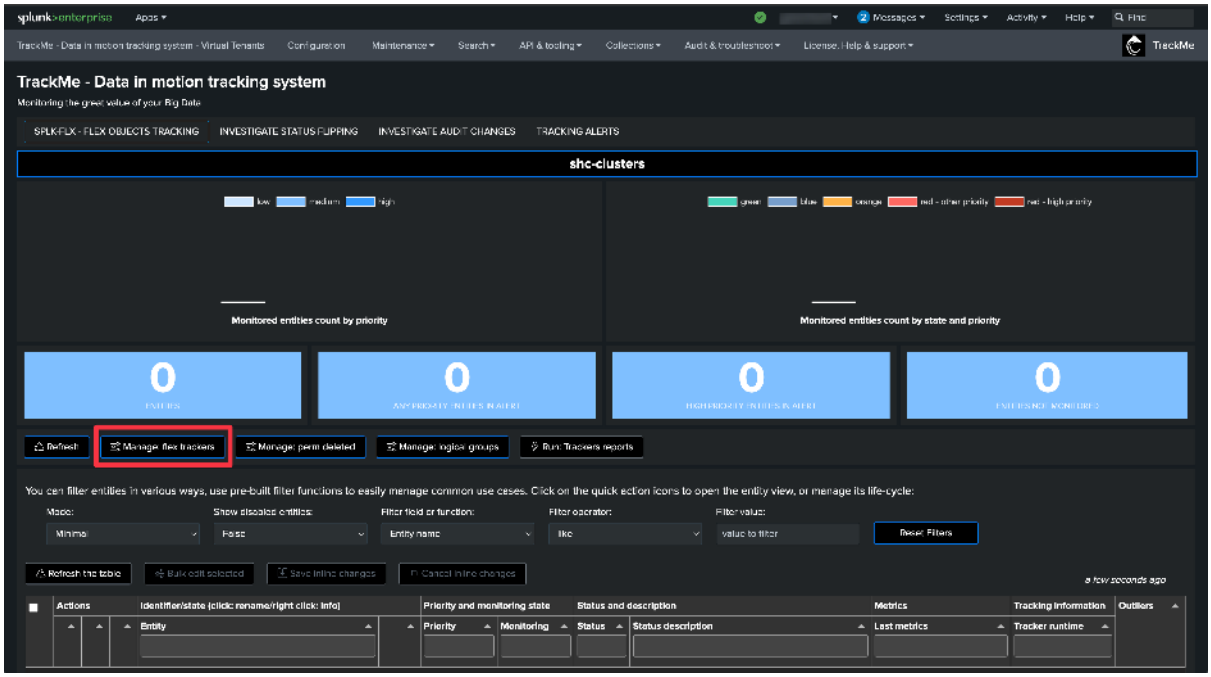
Hint

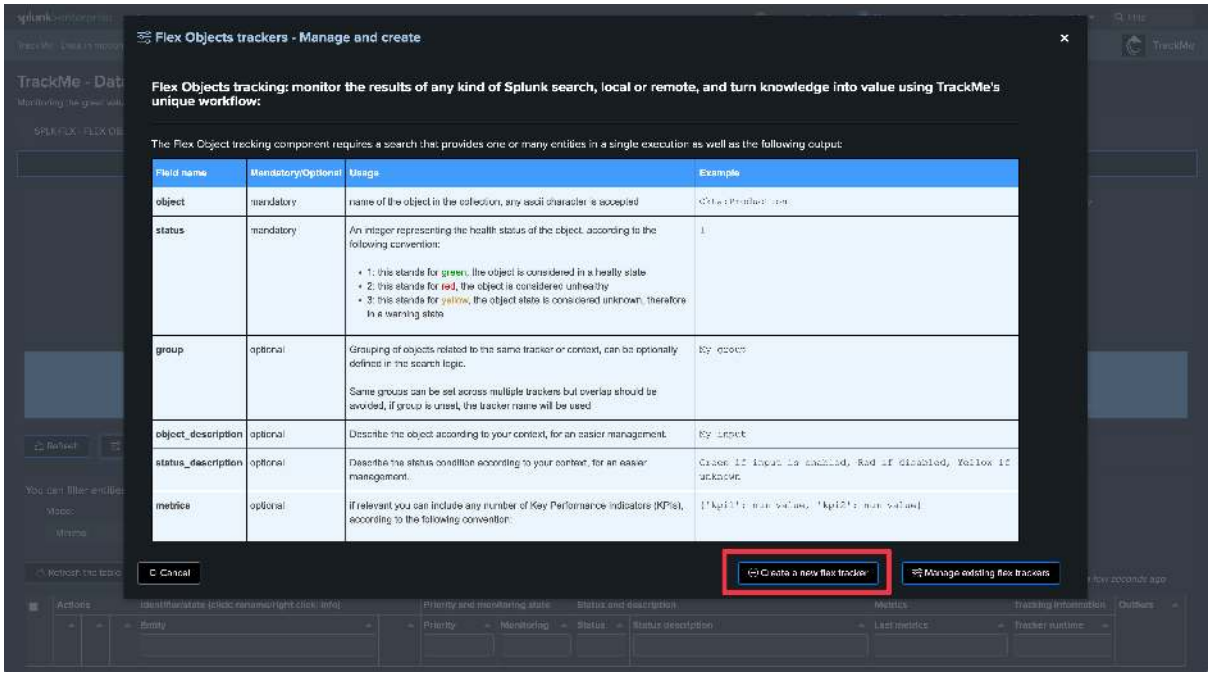
Find the use cases definition (SPL) in annex of this documentation

- See: *Annex: Use cases definition (SPL)*

Enter the newly created tenant and start the Flex Object wizard using the built-in Flex Objects library:

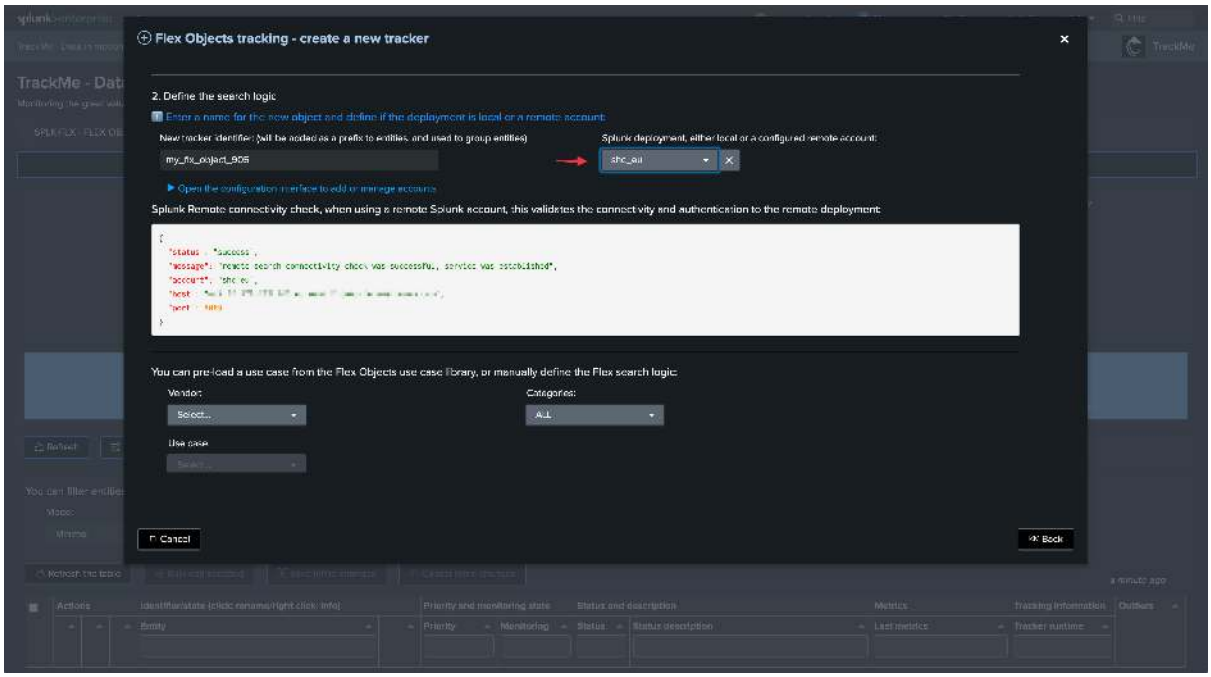


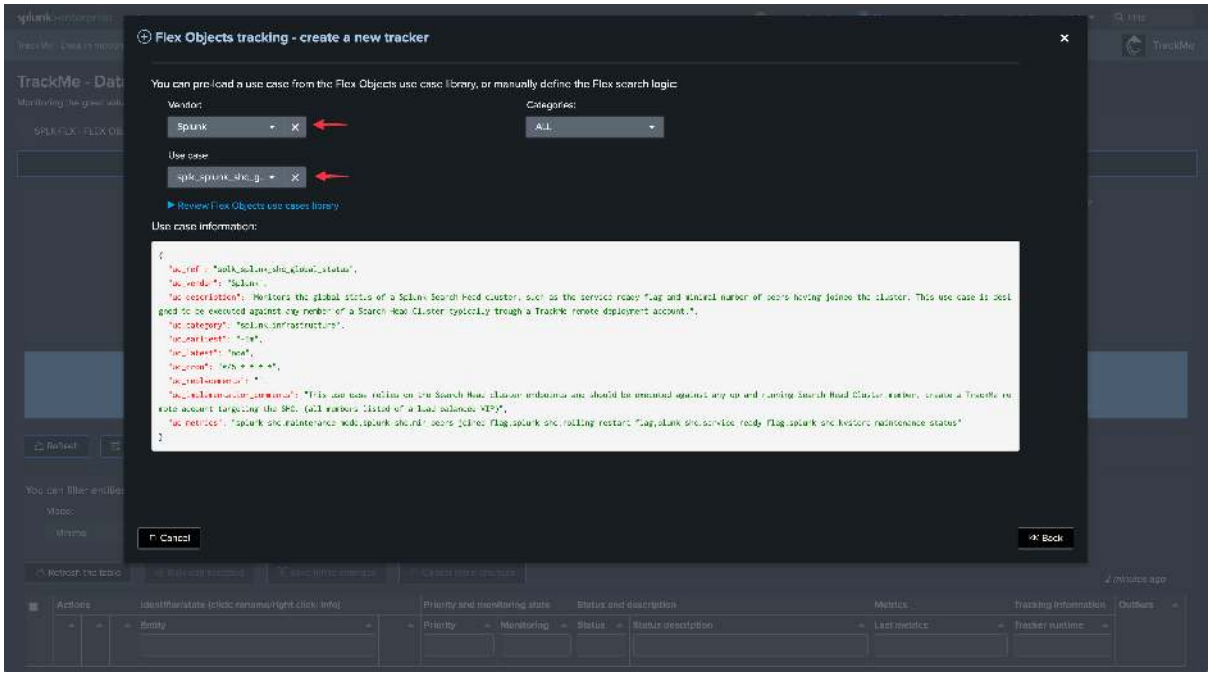


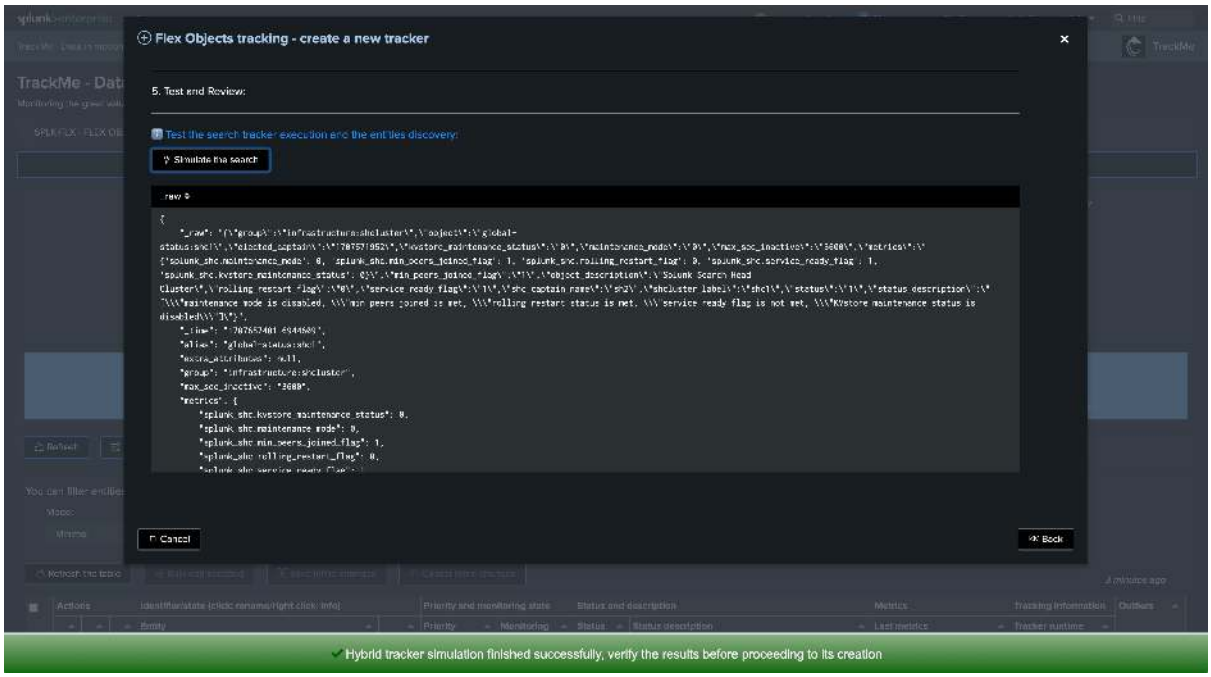


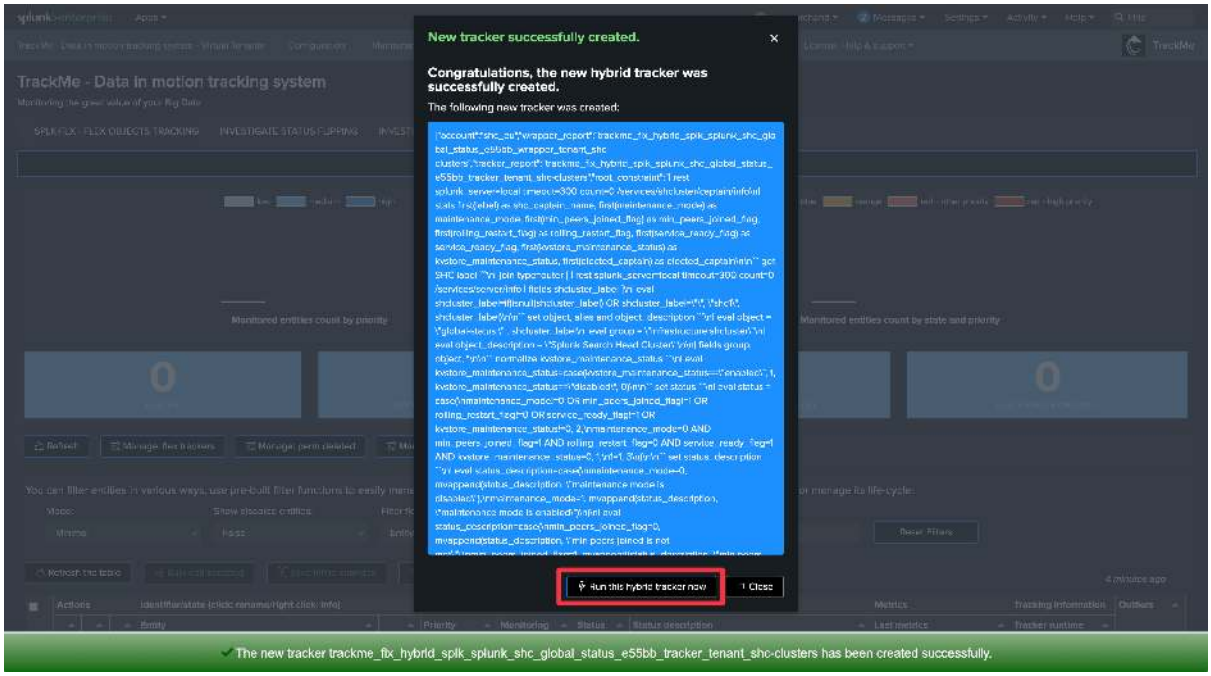
Select the use cases and remote accounts:

We will start with the use case "splk_splunk_shc_global_status" which tracks the SHC globally using the SHC API endpoints.

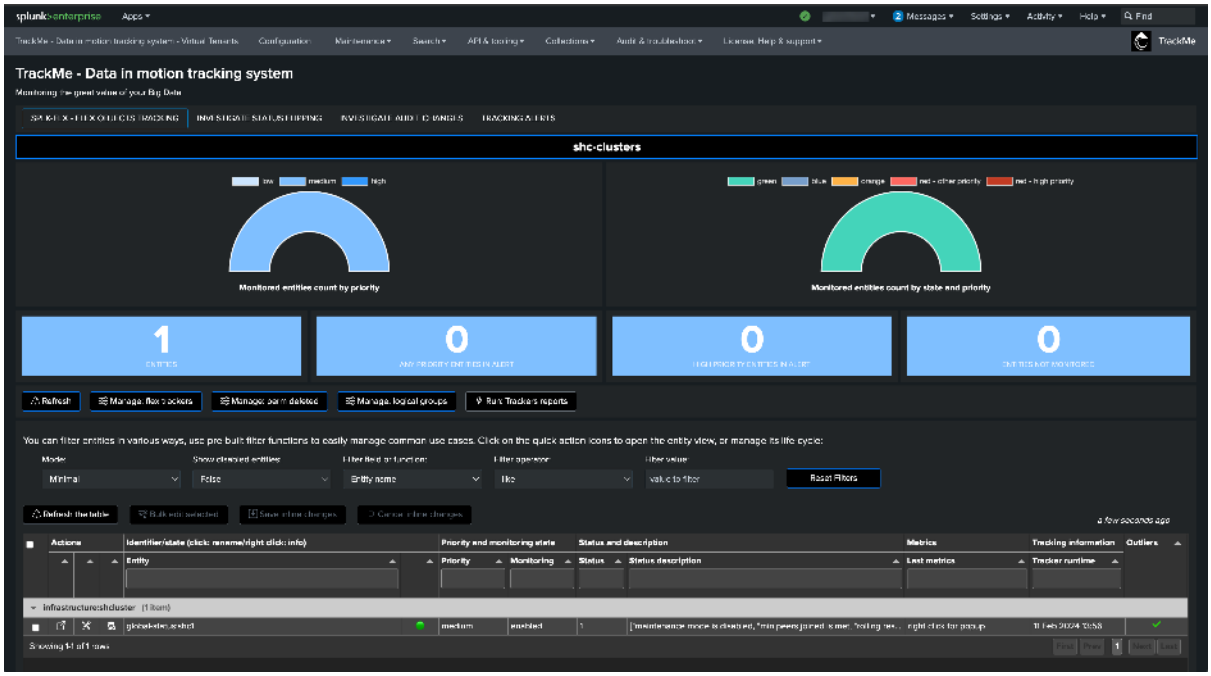


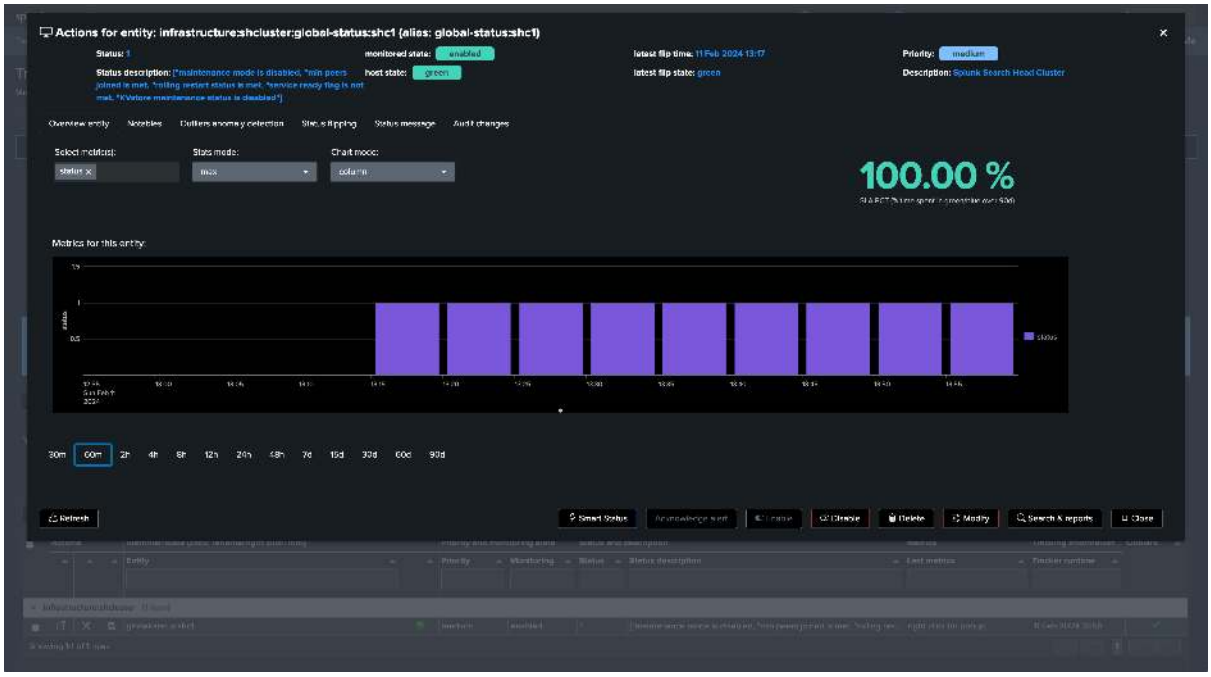

Flex Objects tracking - create a new tracker
You can pre-load a use case from the Flex Objects use case library, or manually define the Flex search logic
Vendor:
Splunk
Categories:
ALL
Use case
Review Flex Objects use cases library
Use case information:
Cancel
Back

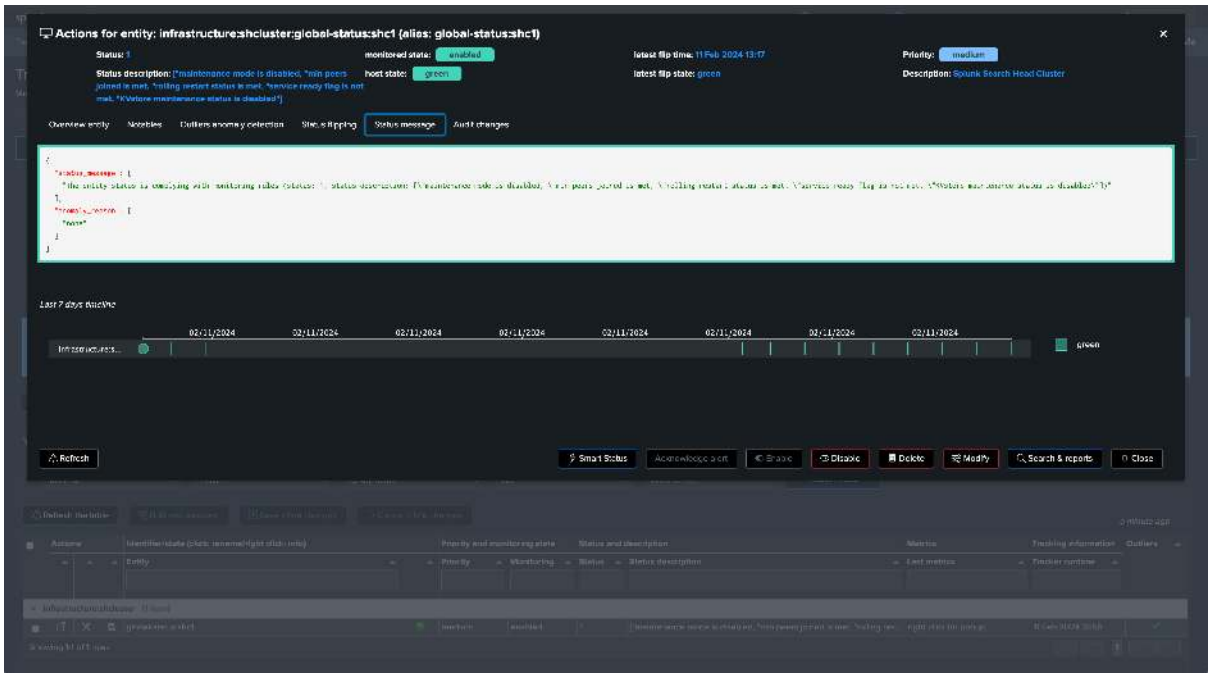

Flex Objects tracking - create a new tracker
5. Test and Review:
Test the search tracker execution and the entities discovery:
Simulate the search
Cancel
Back
Hybrid tracker simulation finished successfully, verify the results before proceeding to its creation

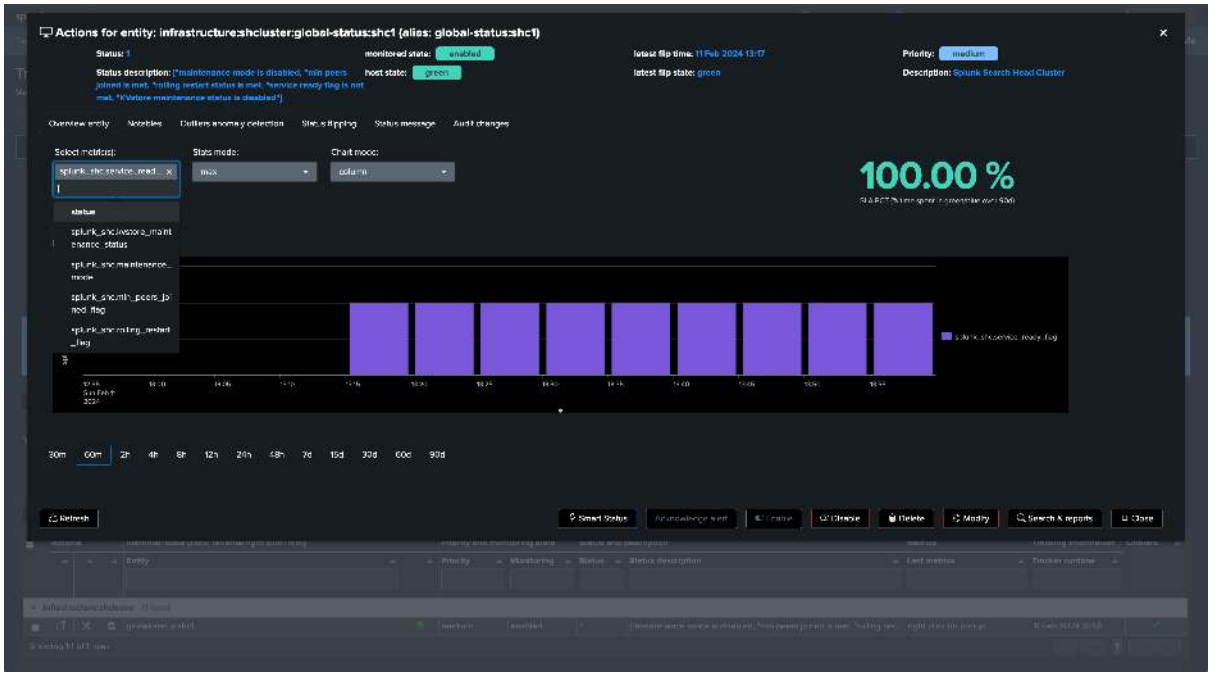


Once executed, TrackMe immediately starts monitoring the SHC global status, generates and indexes metrics, and alerts you when issues are detected.

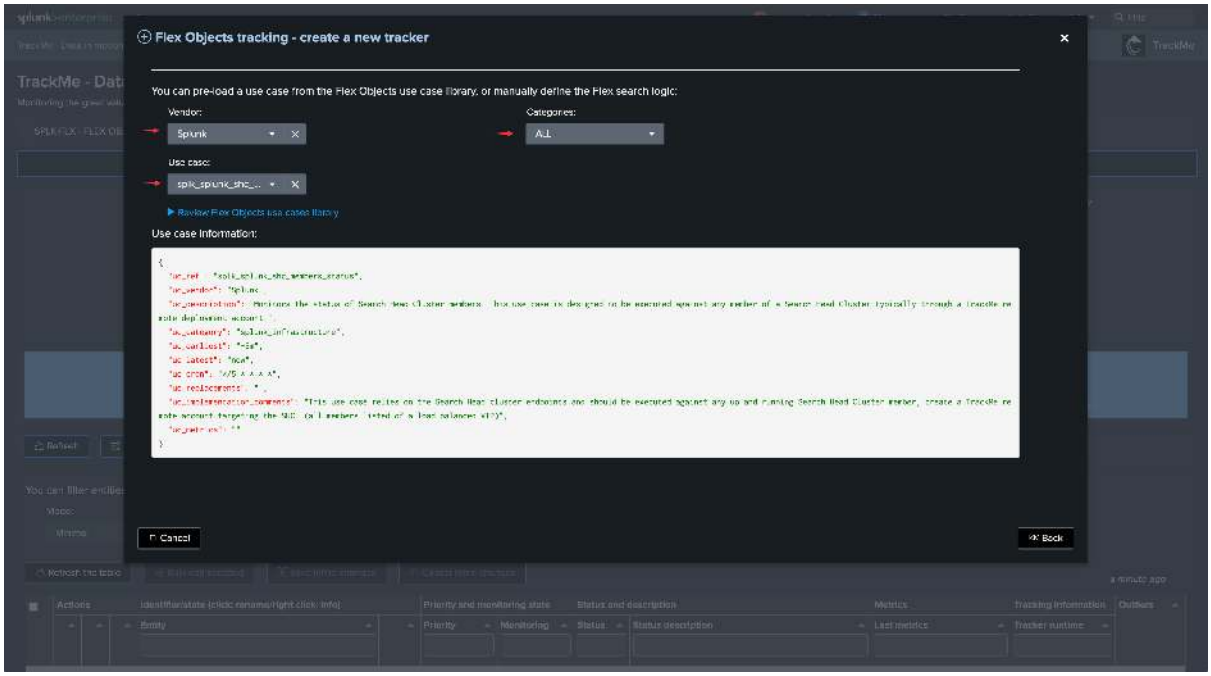



Actions for entity: infrastructure:shcluster:global-status:shc1 (alias: global-status:shc1)
Status: 1
monitored state: enabled
latest flip time: 11 Feb 2024 13:17
Priority: medium
Status description: ["maintenance mode is disabled, *min peers joined is met, *rolling restart status is met, *service ready flag is not met, *KVstore maintenance status is disabled"]
host state: green
latest flip state: green
Description: Splunk Search Head Cluster
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
Select metric(s):
Stats mode:
max
Chart mode:
column
100.00 %
Metrics for this entity:
30m
60m
2h
4h
8h
12h
24h
48h
7d
15d
30d
60d
90d
Refresh
Smart Status
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close

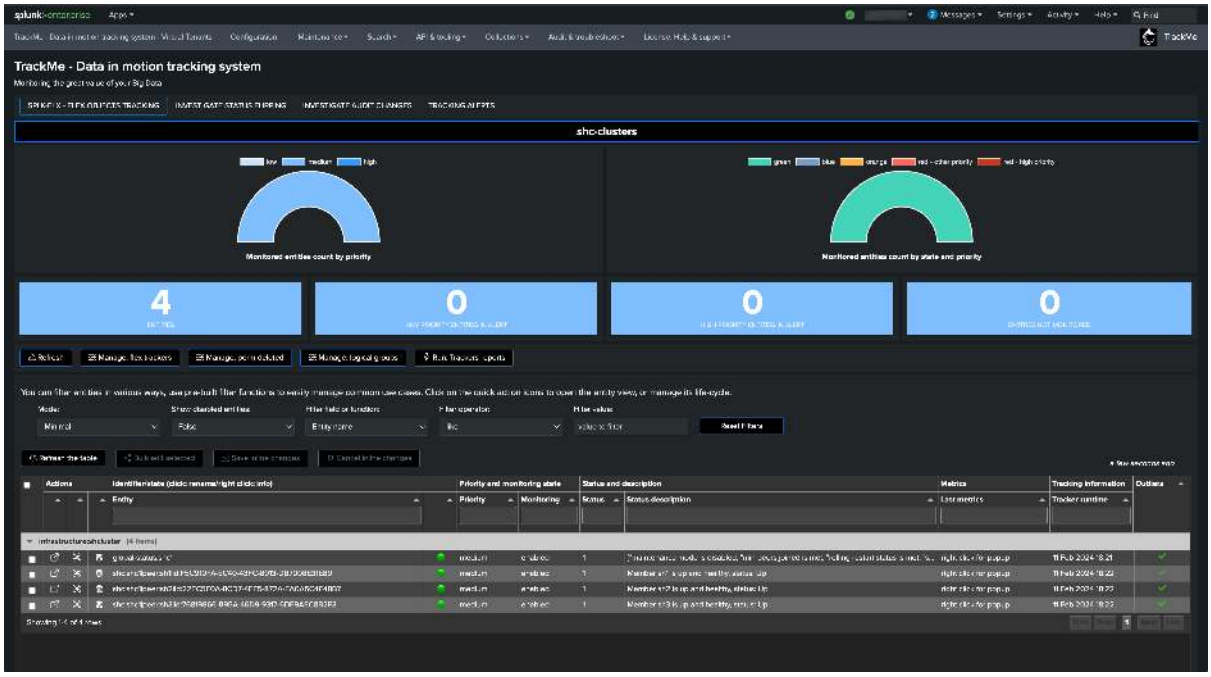

Actions for entity: infrastructure:shcluster:global-status:shc1 (alias: global-status:shc1)
Status: 1
monitored state: enabled
latest flip time: 11 Feb 2024 13:17
Priority: medium
Status description: ["maintenance mode is disabled, *min peers joined is met, *rolling restart status is met, *service ready flag is not met, *KVstore maintenance status is disabled"]
host state: green
latest flip state: green
Description: Splunk Search Head Cluster
Overview entity
Notables
Outliers anomaly detection
Status flipping
Status message
Audit changes
Last 7 days timeline
02/11/2024
green
Refresh
Smart Status
Acknowledge alert
Disable
Delete
Modify
Search & reports
Close

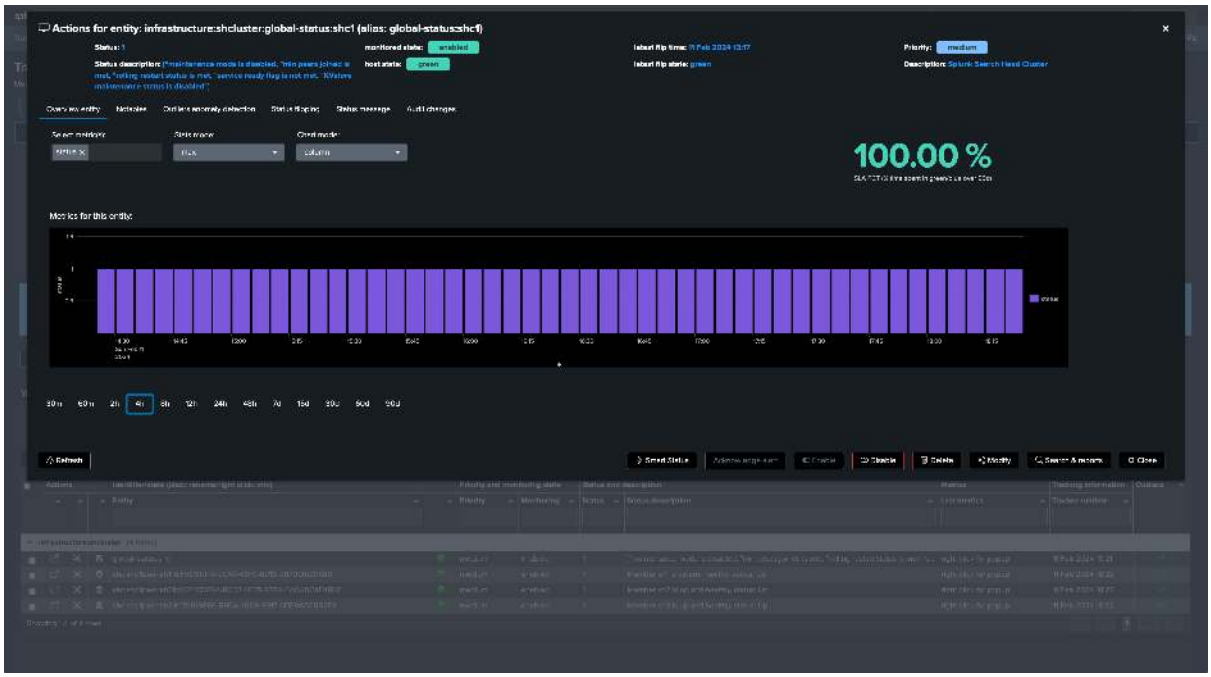


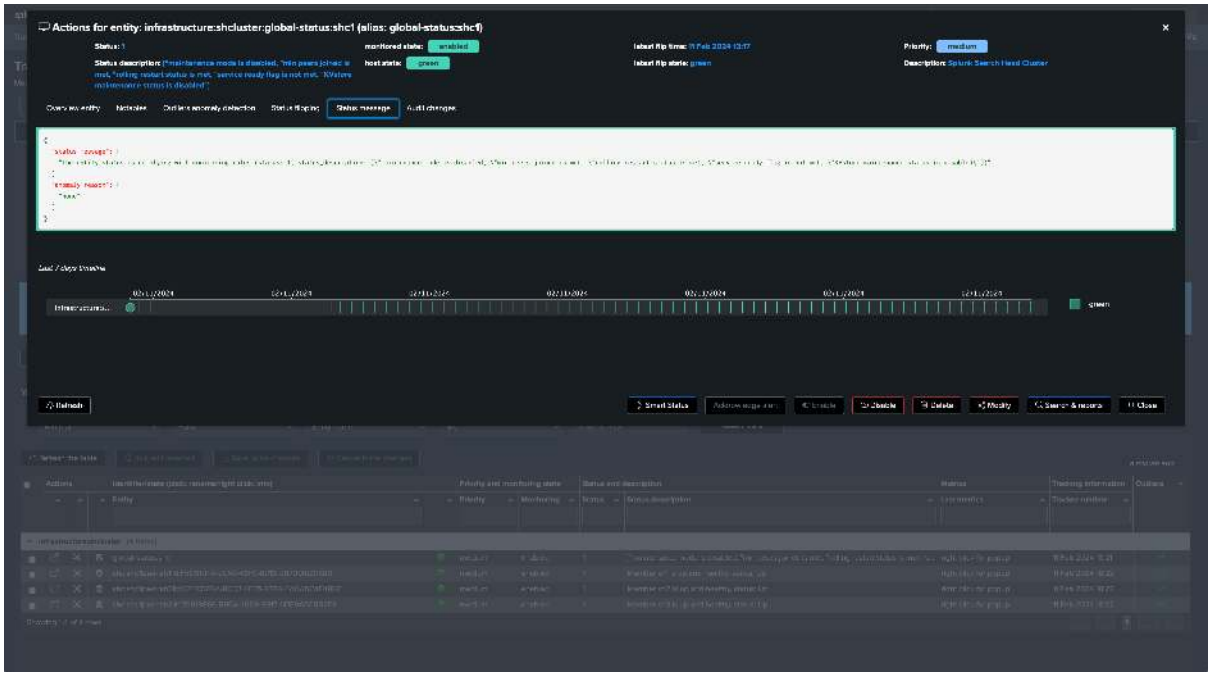
Repeat the same operation with the use case "splk_splunk_shc_members_status" which tracks the SHC members individually, so we instantly detect which members are having issues:

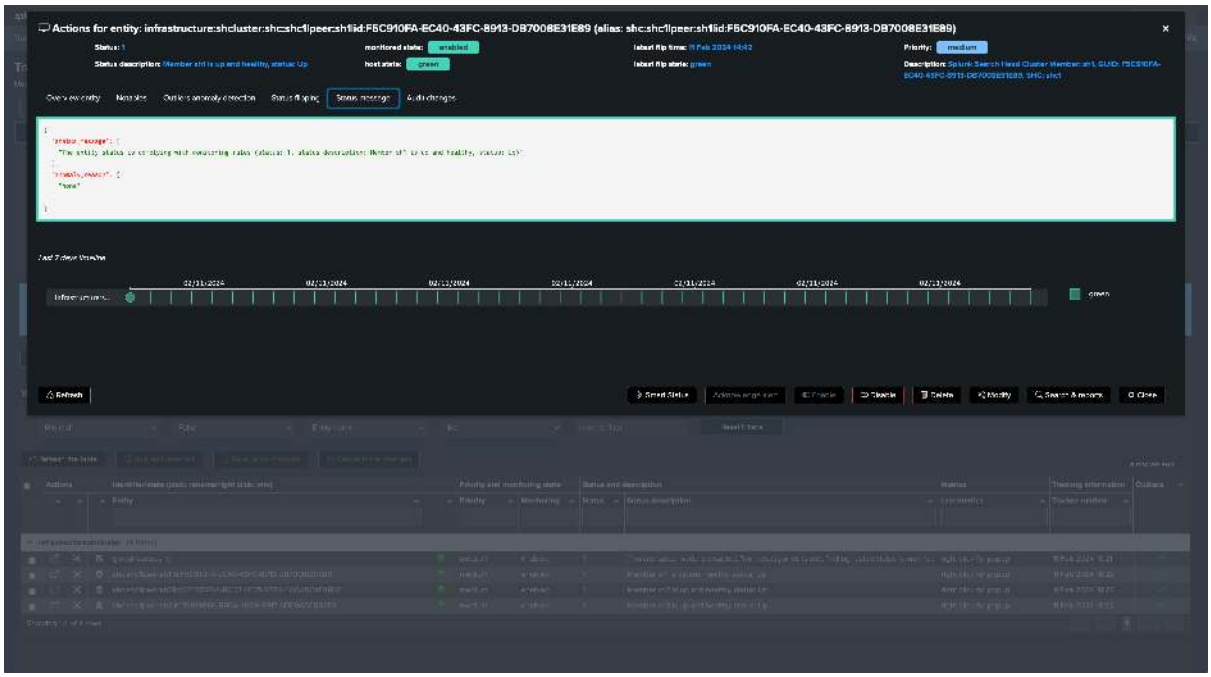


After the first execution of this second tracker, we have a global view of the Search Head Cluster, as well as the view on a per member basis, TrackMe will alert automatically when issues are detected.


TrackMe - Data in motion tracking system
shc:clusters
Monitored entities count by priority
Monitored entities count by state and priority
4
0
0
0


Actions for entity: infrastructure:shcluster:global-status:shc1 (alias: global-status:shc1)
100.00 %
Metrics for this entity





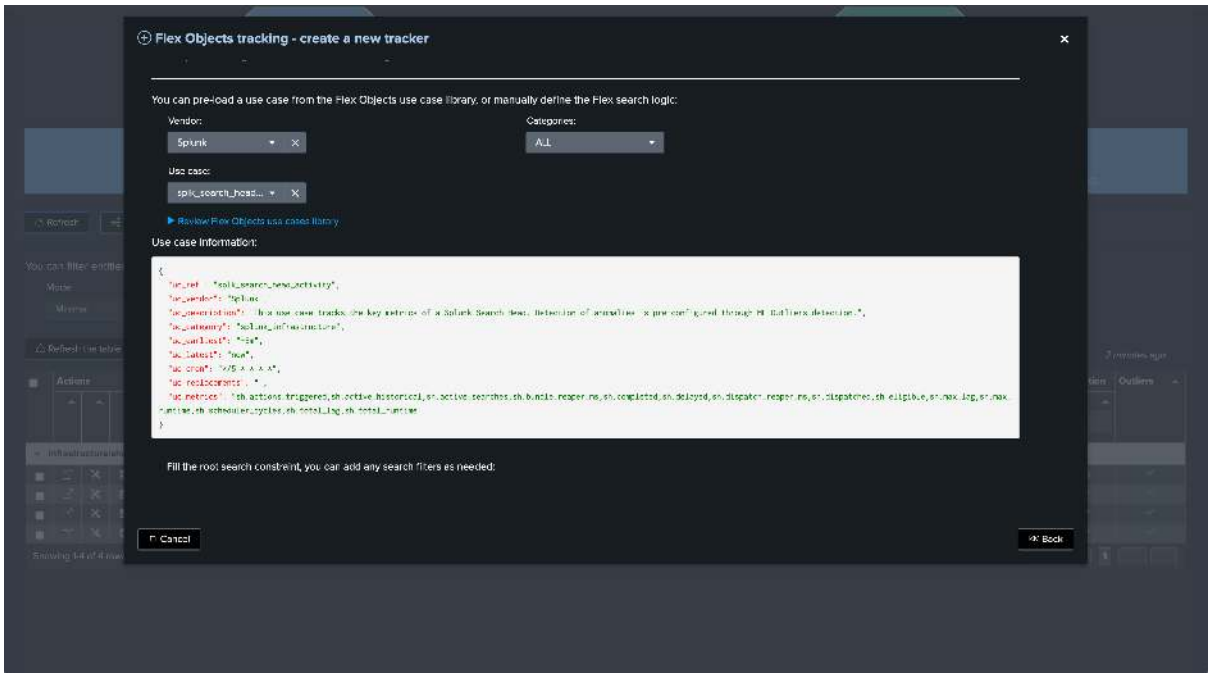
Finally, we can also opt out for the Search Head activity Flex use case, which focuses on collecting and monitoring the activity of the Search Heads:

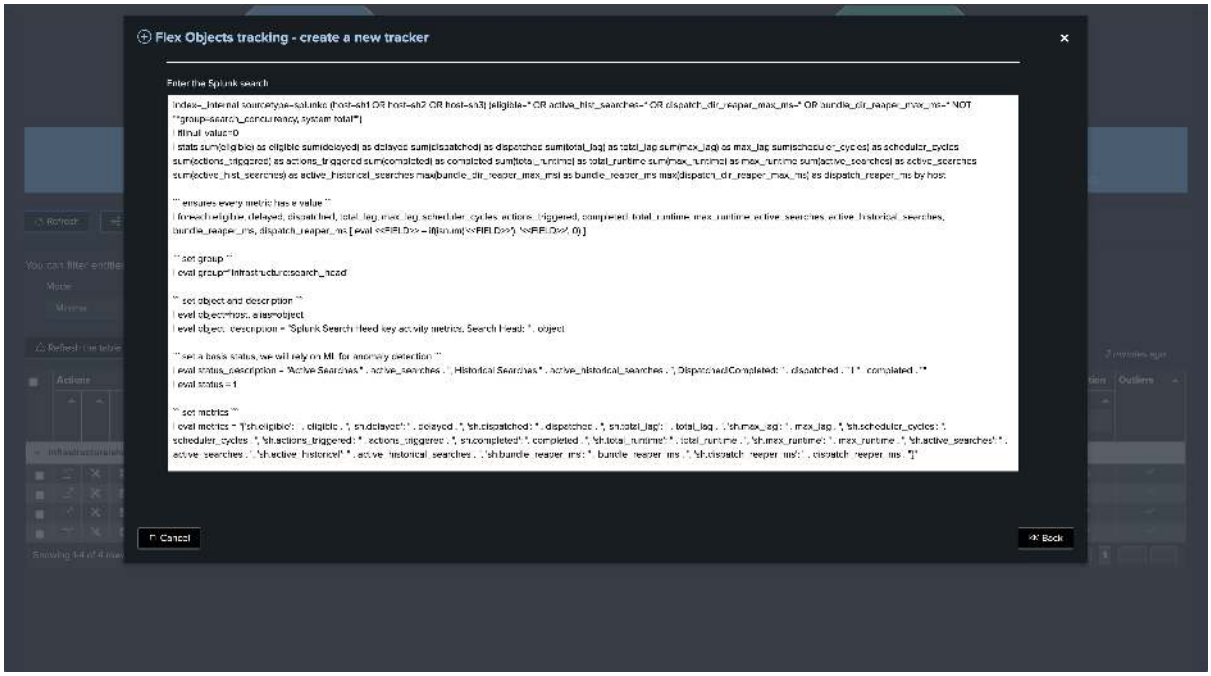
Hint

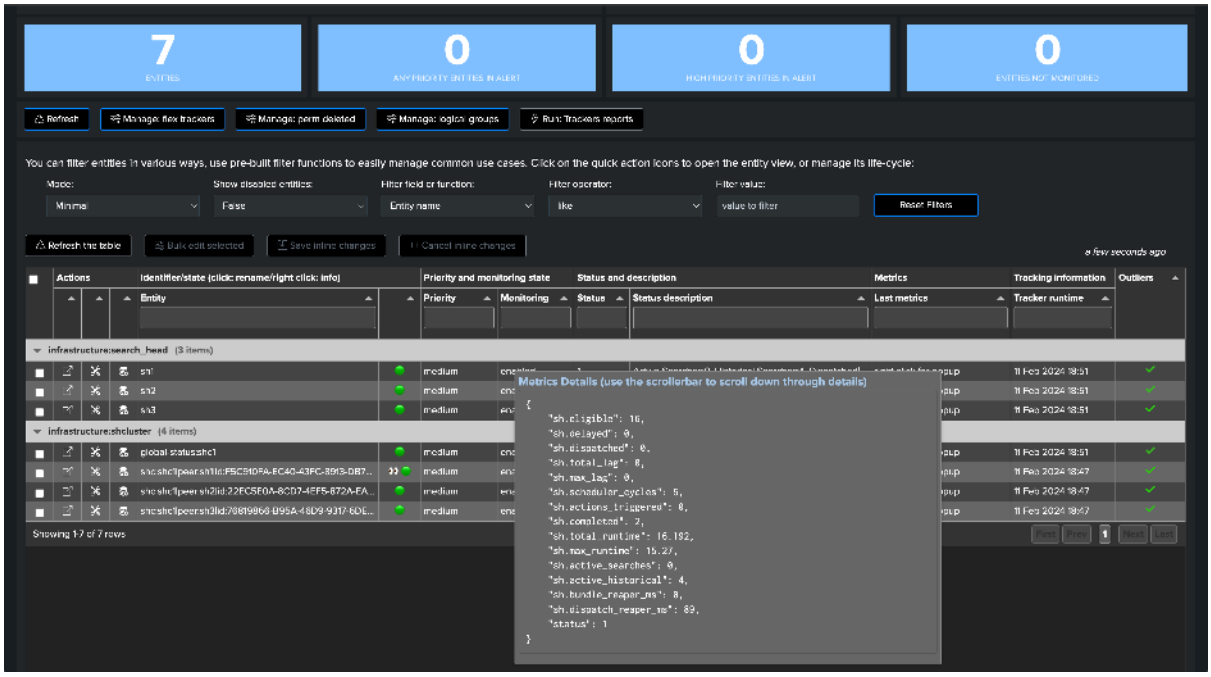
Search Head activity Flex use case

- This use case relies on Splunk indexed data, unlike the rest related use cases which are basically interactions with the Splunk REST API.
- Therefore, If TrackMe has access to the same indexers from a search perspective, it is not required to target the remote account in this case.
- On the other hand, or if you prefer to do so, if the TrackMe guest is on a different indexing layer, using the remote account is mandatory.

- Finally, make sure to restrict the list of host to the Search Heads only, or this use case will also monitor any Search Head type of activity from other Search Heads









Note that unlike the previous use cases, we alert based on the detection of abnormal behaviours of the Search Heads using Machine Learning, notably TrackMe monitors the typical volume of search executions on the Search Head.

Annex: Use cases definition (SPL)

Use case splk_splunk_shc_global_status

```
| rest splunk_server=local timeout=300 count=0 /services/shcluster/captain/info
| stats first(label) as shc_captain_name, first(maintenance_mode) as␣
↪maintenance_mode, first(min_peers_joined_flag) as min_peers_joined_flag,␣
↪first(rolling_restart_flag) as rolling_restart_flag, first(service_ready_flag)␣
↪as service_ready_flag, first(kvstore_maintenance_status) as␣
↪kvstore_maintenance_status, first(elected_captain) as elected_captain
```

(continues on next page)

(continued from previous page)

```
``` get SHC label ```
| join type=outer [ | rest splunk_server=local timeout=300 count=0 /
↪services/server/info | fields shcluster_label ]
| eval shcluster_label=if(isnull(shcluster_label) OR shcluster_label="", "shc1",␣
↪shcluster_label)

``` set object, alias and object_description ```
| eval object = "global-status:" . shcluster_label
| eval group = "infrastructure:shcluster"
| eval object_description = "Splunk Search Head Cluster"

| fields group, object, *

``` normalize kvstore_maintenance_status ```
| eval kvstore_maintenance_status=case(kvstore_maintenance_status=="enabled", 1,␣
↪kvstore_maintenance_status=="disabled", 0)

``` set status ```
| eval status = case(
maintenance_mode!=0 OR min_peers_joined_flag!=1 OR rolling_restart_flag!=0 OR␣
↪service_ready_flag!=1 OR kvstore_maintenance_status!=0, 2,
maintenance_mode=0 AND min_peers_joined_flag=1 AND rolling_restart_flag=0 AND␣
↪service_ready_flag=1 AND kvstore_maintenance_status=0, 1,
1=1, 3
)

``` set status_description ```
| eval status_description=case(
maintenance_mode=0, mvappend(status_description, "maintenance mode is disabled"),
maintenance_mode=1, mvappend(status_description, "maintenance mode is enabled")
)
| eval status_description=case(
min_peers_joined_flag=0, mvappend(status_description, "min peers joined is not␣
↪met"),
min_peers_joined_flag=1, mvappend(status_description, "min peers joined is met")
)
| eval status_description=case(
rolling_restart_flag=0, mvappend(status_description, "rolling restart status is␣
↪met"),
rolling_restart_flag=1, mvappend(status_description, "rolling restart status is in␣
↪progress or status is not met")
)
| eval status_description=case(
service_ready_flag=1, mvappend(status_description, "service ready flag is met"),
service_ready_flag=0, mvappend(status_description, "service ready flag is not met")
)
| eval status_description=case(
kvstore_maintenance_status=0, mvappend(status_description, "KVstore maintenance␣
↪status is disabled"),
kvstore_maintenance_status=1, mvappend(status_description, "KVstore maintenance␣
↪status is enabled")
)
| eval status_description="[\"" . mvjoin(status_description, ", \"") . "\"]"

``` set metrics ```
| eval metrics = "{" . "'splunk_shc.maintenance_mode': " . maintenance_mode . ",
```

(continues on next page)

(continued from previous page)

```
↪'splunk_shc.min_peers_joined_flag': " . min_peers_joined_flag . ",
↪'splunk_shc.rolling_restart_flag': " . rolling_restart_flag . ",
↪'splunk_shc.service_ready_flag': " . service_ready_flag . ",
↪'splunk_shc.kvstore_maintenance_status': " .  kvstore_maintenance_status . "}"

``` alert if inactive for more than 3600 sec```
| eval max_sec_inactive=3600
```

Use case "splk_splunk_shc_members_status

```
| rest splunk_server=local timeout=300 count=0 /services/shcluster/status
| transpose
| rename column as category, "row 1" as value

``` extract information ```
| eval cluster_member_name=case(match(category, "^peers\.[^\.]*\.label$"), value)
| rex field=category "^peers\.(?<cluster_member_guid>[^\.]*)\.label$"
| eval cluster_member_status=case(match(category, "^peers\.[^\.]*\.status$"),␣
↪value)

``` aggregate ```
| stats list(cluster_member_guid) as cluster_member_guid,␣
↪list(cluster_member_name) as cluster_member_name, list(cluster_member_status) as␣
↪cluster_member_status

``` build entity from mv fields ```
| eval cluster_member_info=mvzip(mvzip(cluster_member_guid, cluster_member_name),␣
↪cluster_member_status)

``` order ```
| fields cluster_member_info, cluster_captain, service_ready_flag,␣
↪rolling_restart_flag

``` expand entities and extract ```
| mvexpand cluster_member_info
| eval cluster_member_info=split(cluster_member_info, ",")
| eval cluster_member_guid=mvindex(cluster_member_info, 0),␣
↪cluster_member_name=mvindex(cluster_member_info, 1),␣
↪cluster_member_status=mvindex(cluster_member_info, 2)

``` order ```
| fields cluster_member_guid, cluster_member_name, cluster_member_status

``` get SHC label ```
| join type=outer [ | rest splunk_server=local timeout=300 count=0 /
↪services/server/info | fields shcluster_label ]

``` set object, alias and object_description ```
| eval object = "shc:" . shcluster_label . "|peer:" . cluster_member_name . "|id:"␣
↪. cluster_member_guid
| eval group = "infrastructure:shcluster"
| eval object_description = "Splunk Search Head Cluster Member: " .␣
↪cluster_member_name . ", GUID: " . cluster_member_guid . ", SHC: " .␣
↪shcluster_label
```

(continues on next page)
```

(continued from previous page)

```
``` set status ```
| eval status = case(
cluster_member_status="Up", 1,
cluster_member_status="Down", 2,
1=1, 3
)

``` set status description ```
| eval status_description = case(
status=1, "Member " . cluster_member_name . " is up and healthy, status: " .␣
↪cluster_member_status,
status=2, "Member " . cluster_member_name . " is currently down, status: " .␣
↪cluster_member_status,
status=3, "Member " . cluster_member_name . " status is unknown or unexpected,␣
↪status: " . cluster_member_status
)

``` alert if inactive for more than 3600 sec```
| eval max_sec_inactive=3600
```

Use case splk_splunk_shc_activity

```
index=_internal sourcetype=splunkd host=* (eligible=* OR active_hist_searches=* OR␣
↪dispatch_dir_reaper_max_ms=* OR bundle_dir_reaper_max_ms=* NOT
↪"*group=search_concurrency, system total*")
| fillnull value=0
| stats sum(eligible) as eligible sum(delayed) as delayed sum(dispatched) as␣
↪dispatched sum(total_lag) as total_lag sum(max_lag) as max_lag␣
↪sum(scheduler_cycles) as scheduler_cycles sum(actions_triggered) as␣
↪actions_triggered sum(completed) as completed sum(total_runtime) as␣
↪total_runtime sum(max_runtime) as max_runtime sum(active_searches) as␣
↪active_searches sum(active_hist_searches) as active_historical_searches␣
↪max(bundle_dir_reaper_max_ms) as bundle_reaper_ms␣
↪max(dispatch_dir_reaper_max_ms) as dispatch_reaper_ms by host

``` ensures every metric has a value ```
| foreach eligible, delayed, dispatched, total_lag, max_lag, scheduler_cycles,␣
↪actions_triggered, completed, total_runtime, max_runtime, active_searches,␣
↪active_historical_searches, bundle_reaper_ms, dispatch_reaper_ms [ eval <
↪<FIELD>> = if(isnum('<<FIELD>>'), '<<FIELD>>', 0) ]

``` set group ```
| eval group="infrastructure:search_head"

``` set object and description ```
| eval object=host, alias=object
| eval object_description = "Splunk Search Head key activity metrics, Search Head:
↪" . object

``` set a basis status, we will rely on ML for anomaly detection ```
| eval status_description = "Active Searches:" . active_searches . ", Historical␣
↪Searches:" . active_historical_searches . ", Dispatched|Completed: " .␣
↪dispatched . " | " . completed . ""
| eval status = 1
```

(continues on next page)
```

(continued from previous page)

```
``` set metrics ```
| eval metrics = "{'sh.eligible': " . eligible . ", 'sh.delayed': " . delayed . ",
↪'sh.dispatched': " . dispatched . ", 'sh.total_lag': " . total_lag . ",
↪'sh.max_lag': " . max_lag . ", 'sh.scheduler_cycles': ". scheduler_cycles . ",
↪'sh.actions_triggered': " . actions_triggered . ", 'sh.completed': ". completed .
↪ ", 'sh.total_runtime': " . total_runtime . ", 'sh.max_runtime': " . max_runtime␣
↪. ", 'sh.active_searches': " . active_searches . ", 'sh.active_historical': " .␣
↪active_historical_searches . ", 'sh.bundle_reaper_ms': " . bundle_reaper_ms . ",
↪'sh.dispatch_reaper_ms': " . dispatch_reaper_ms . "}"

``` set outliers metrics ```
| eval outliers_metrics = "{'sh.completed': {'alert_lower_breached': 1,
↪'alert_upper_breached': 0, 'time_factor': '%w%H'}}"

| eval max_sec_inactive=3600
```

9.4 Detection & alerting use cases

White papers on detection and alerting patterns — volume outliers on the Splunk license usage with the Volume Outliers component, who is doing what on your Splunk with User Activity Monitoring, event-drop detection, log-level behaviour anomalies with Machine Learning, and building a two-tier SLA alerting system.

9.4.1 Volume outliers on the Splunk license usage with Volume Outliers (VOL)

About this white paper

- This white paper walks through the **Volume Outliers (VOL)** component introduced in TrackMe **2.4.18**: the detection of abnormal **drops** and **spikes** of the licensed volume of every Splunk index, the detection of **silent** indexes, and the analytics of the license consumption — trend, month projection, top consumers, pools — that come with it.
- Updated for **2.4.19**: the Global license usage tab became a multi-tab view — named scopes, working days and holidays, days over quota, opt-in costs and a monthly chargeback per cost centre, with an AI Assistant that can configure it for you — and the backfill accepts any depth up to 400 days.
- Volume detection has always been part of TrackMe, per feed and per index. VOL is its standardised, first-class form at scale, and supersedes the earlier recipes.
- It is a **tutorial on a live deployment**: a Splunk Enterprise environment already monitored by two typical TrackMe tenants (a feeds tenant and a hosts tenant), to which we add a Volume Outliers tenant in a few clicks and follow it from its creation to its first detections.
- It **supersedes** the volume-outliers recipes of the earlier white papers — *detecting abnormal events count drops* with the splk-dsm models and the Flex Objects *license usage per index* templates — which remain valid references for custom needs (per-feed event counts, custom KPIs) and are cross-referenced where relevant.
- VOL is available on the **Foundation**, **Enterprise** and **Unlimited** editions.
- Product guide references: *VOL — Volume Outliers*, *Volume Outliers — in depth*, *Outlier detection*.

The problem: volume deviations nobody sees coming

The volume of data a Splunk index receives is one of the most telling signals of a platform, and one of the least monitored:

- **A drop** is, more often than not, a broken pipeline: a forwarder stopped, a syslog relay lost a route, a modular input silently stopped, a parsing change discards half of the events. The feed is *still arriving* — a freshness

check is green — but half of what should be there is missing, and so are the detections and dashboards built on it.

- **A spike** is a cost and a risk: a debug logging level left on, a runaway application, a new source pointed at the wrong index, a duplicate forwarder. Discovered at month end, it is a license overage; discovered within the hour, it is a ticket.
- **A silent index** — one that stops consuming license altogether — is the drop nobody correlates: the index still exists, dashboards still run over it, and nobody notices that its last event is a week old.

TrackMe has offered volume outlier detection since its first versions (the splk-dsm `eventcount` models), and, for licensed customers, the more efficient index-level recipe built on Flex Objects over the licensing logs. Both work; both demand design (which sourcetypes to exclude, which templates to deploy, how to backfill) before they deliver. The requirement was clear: **out of the box, reliable, cheap to run, and available to every customer**.

Why the license usage log

The Splunk license manager writes, every minute, one `Usage` event per index (and per pool, host and source) into `license_usage.log` — the volume it counts against your license. As a source for volume monitoring it is close to ideal:

- **Authoritative** — it is the volume you pay for, measured by Splunk itself, not an approximation derived from event counts.
- **One source for the whole deployment** — every index of every indexer, in one log, on the license manager; no per-index or per-sourcetype scan, whatever the number of indexes.
- **Cheap to read** — a few kilobytes per minute; reading only the *new* events every five minutes costs a fraction of one `tstats` over the platform.
- **The same everywhere** — Splunk Enterprise or Splunk Cloud, and on a remote deployment through a TrackMe remote account: the format and the semantics do not change.

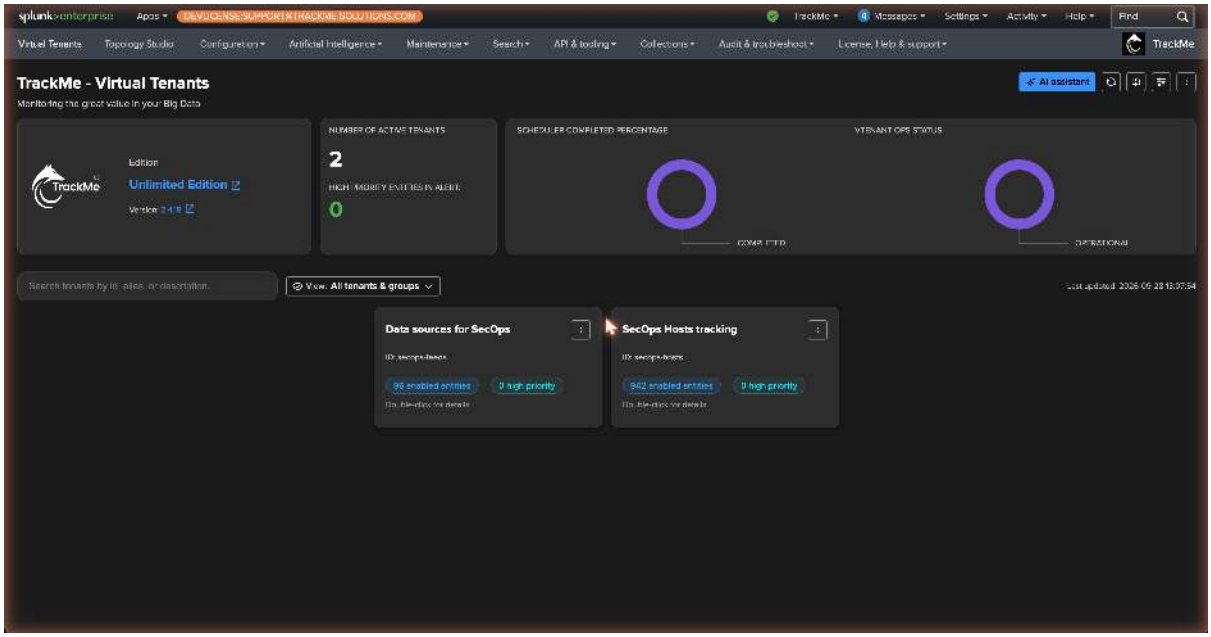
Volume Outliers builds on it with a purpose-built, incremental collector — described in *the in-depth reference* — and turns every index into a first-class TrackMe entity.

What VOL delivers

Capability	What you get
Detection	One ML model per index on its rolling 24-hour licensed volume, with a **detection direction** per index — drops, spikes, or both. Inactivity detection with a tenant policy (by day of week and hour of day) and per-index overrides.
Day one	The history is **backfilled** at creation (30 days by default), so the models are trained and the cost trend is complete immediately.
Analytics	Per index: the daily licensed volume per license day, the 30-day trend, the month to date and the month projection. For the environment: the **Global license usage (VOL)** tab — key figures, insights, top consumers, movers, pools and quotas.
Operations	Priority, tags, labels, SLA, logical groups, acknowledgments, notes, maintenance, stateful alerting with rich emails, the AI Assistant and the AI ML Advisor: every index is a regular TrackMe entity.
Cost	One bounded search every five minutes per tenant, reading only the new license usage events; a compact metrics footprint (about 216 KB per index per day).

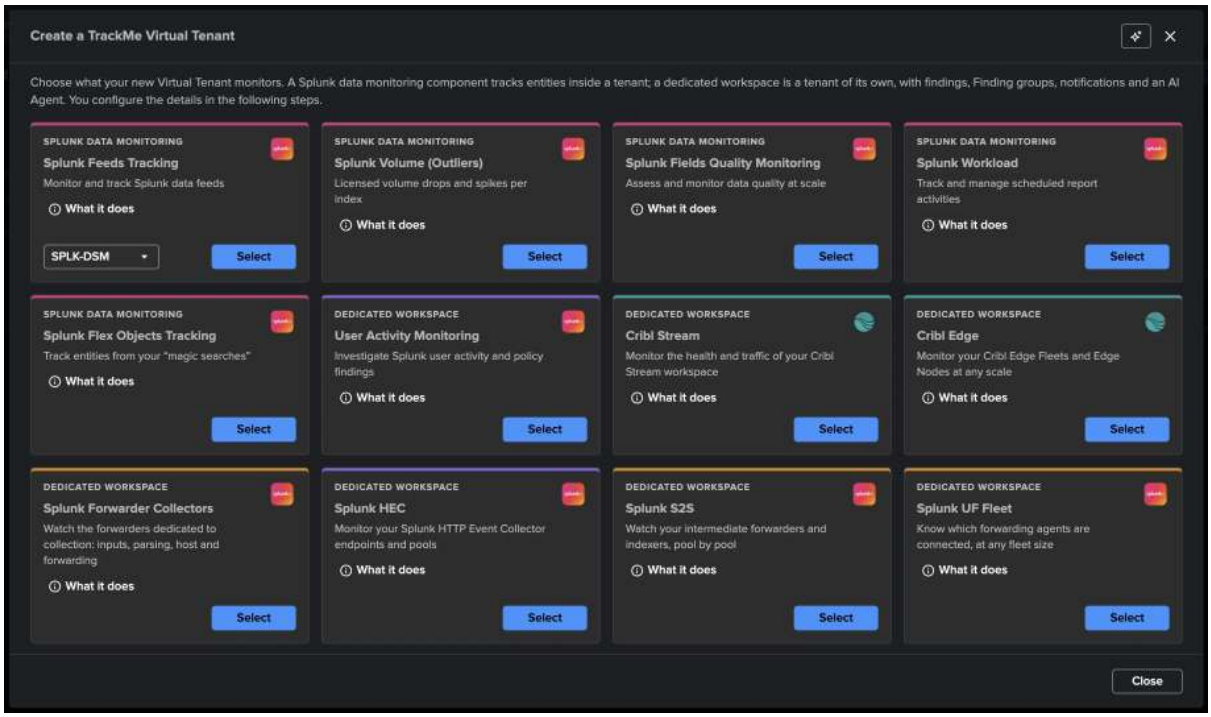
The environment of this tutorial

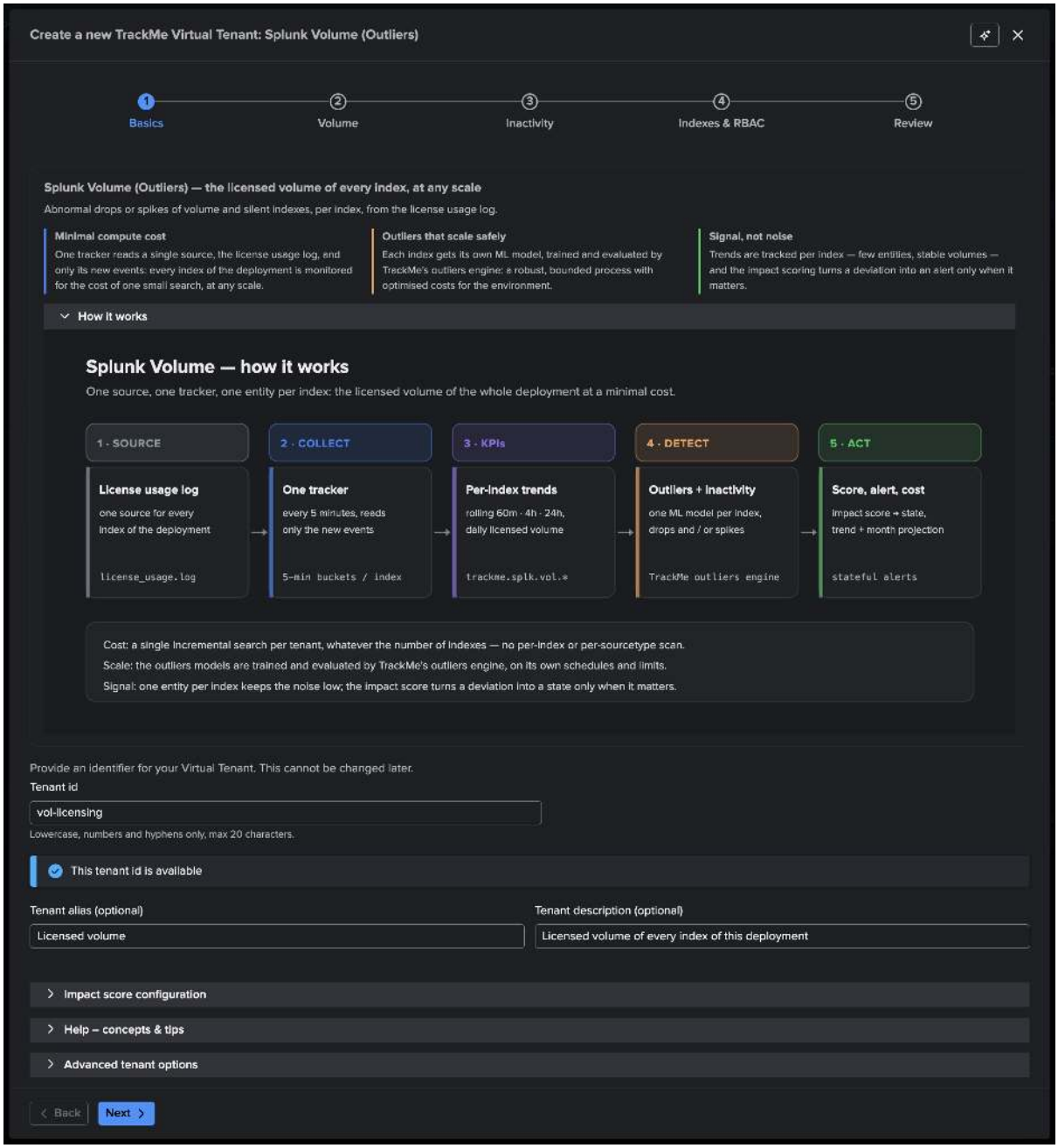
Our deployment is a typical **Foundation** setup: one Splunk Enterprise environment, monitored by two TrackMe tenants — **Data sources for SecOps** (splk-dsm, the feeds by index and sourcetype) and **SecOps Hosts tracking** (splk-dhm, the endpoints). We add a third tenant for the license volume. The environment has around forty indexes in one license pool; the license usage log is the standard `index=_internal source="*/license_usage.log"`.



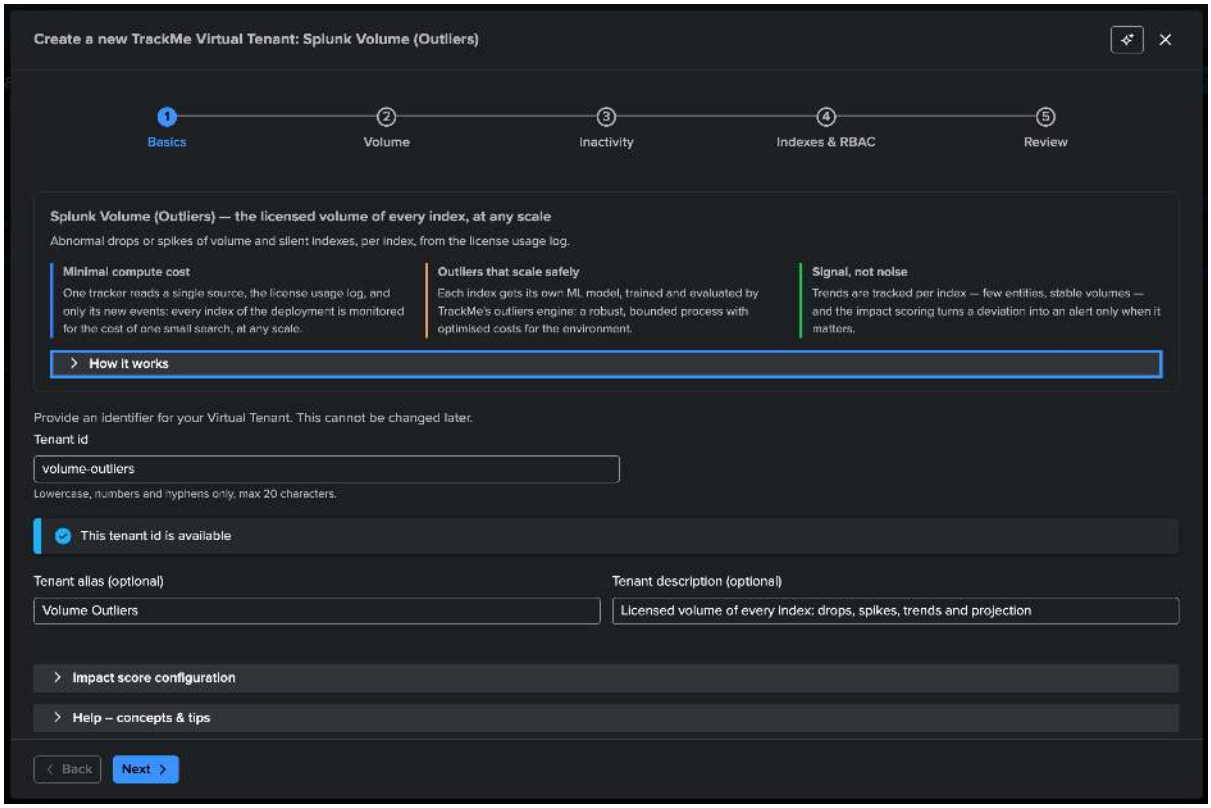
Step 1 — Create the Volume Outliers tenant

From the Virtual Tenants page, **Actions → Create a tracking tenant** opens the component chooser. The **Splunk Volume (Outliers)** card is second, right after Splunk Feeds Tracking:



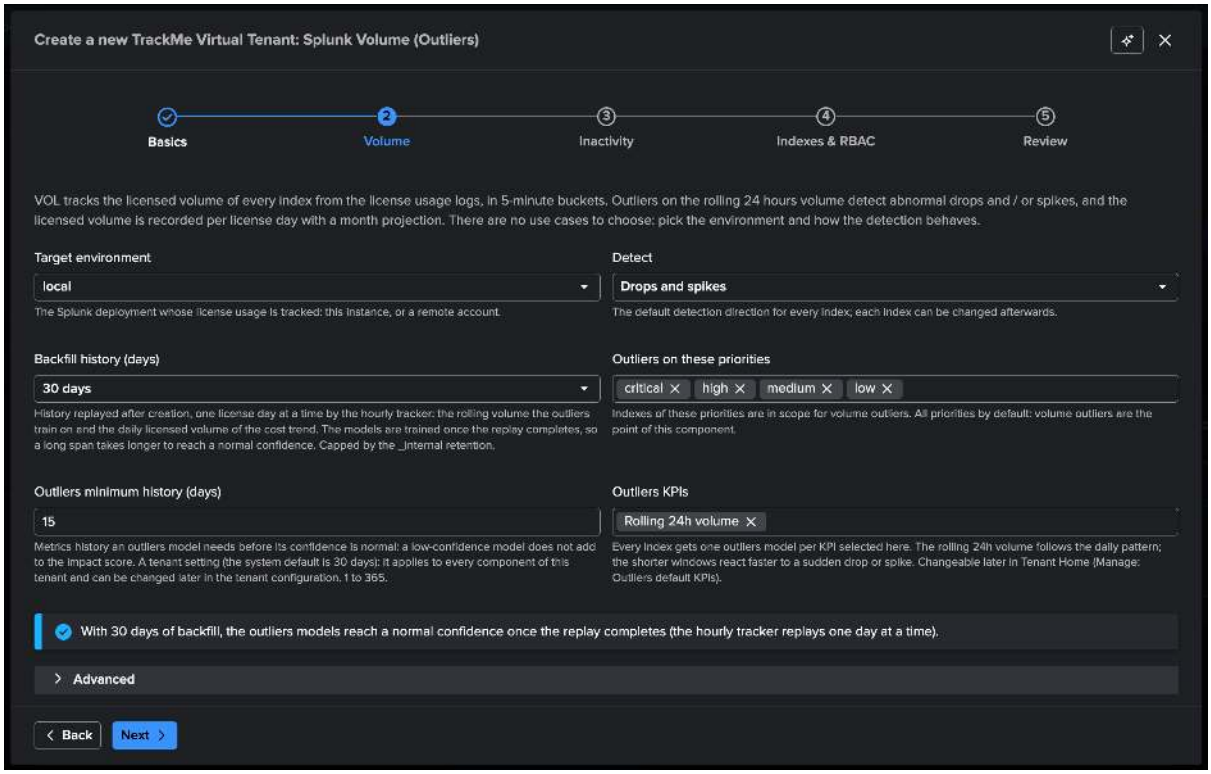


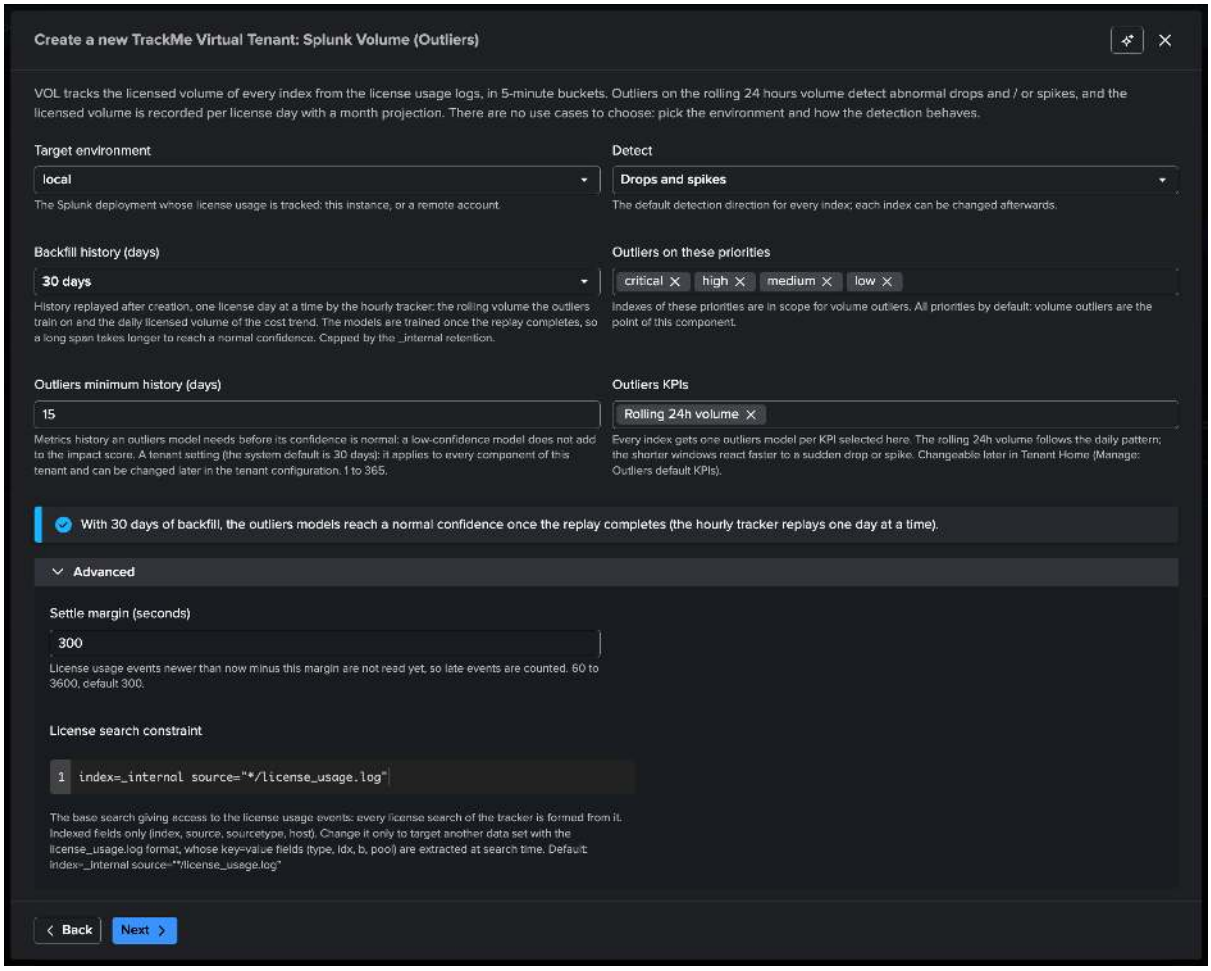
The **Basics** step introduces the component — one tracker, one entity per index, outliers that scale safely — and asks for the tenant identity. We call the tenant `volume-outliers`:

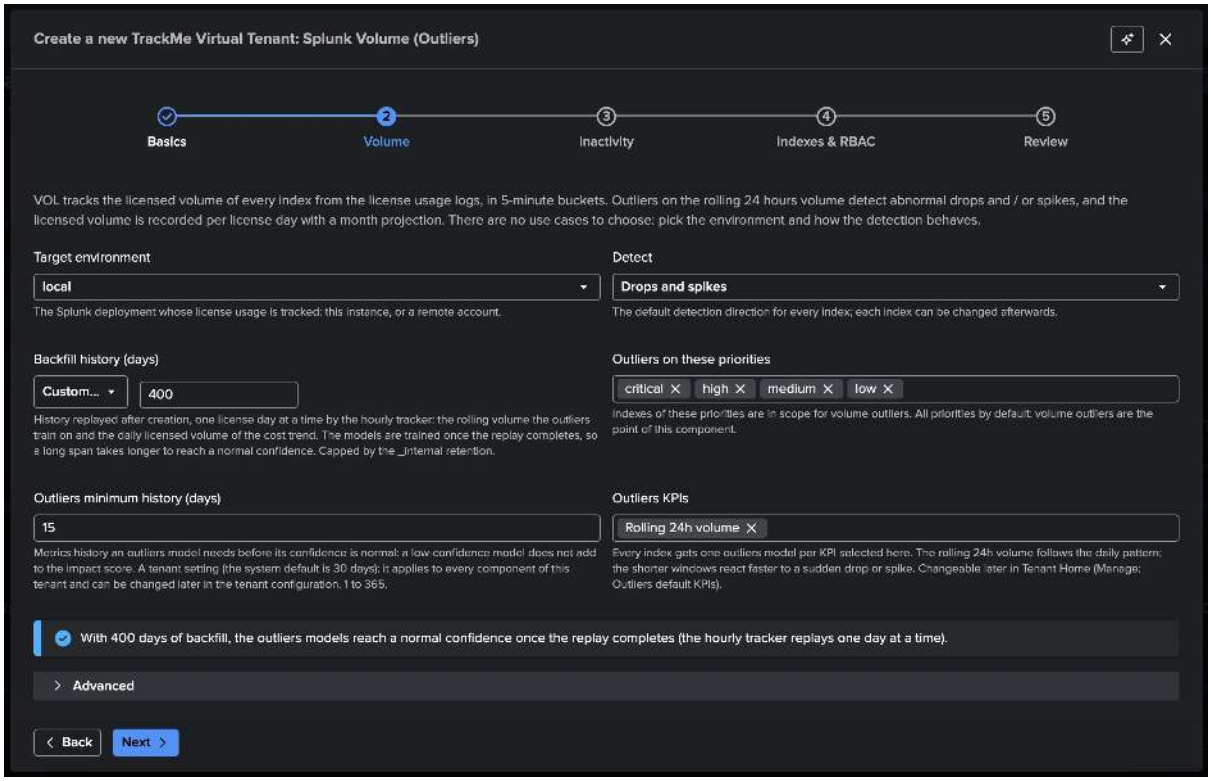


The **Volume** step is where the component is configured, and its defaults are the recommended values:

- **Target environment**: `local` — the license usage of this Splunk deployment. A **remote account** would collect another deployment's license usage from here.
- **Detect**: *Drops and spikes* — the default direction of every new index; we will change it per index later.
- **Backfill history**: **30 days** — the history replayed after creation, one license day at a time by the hourly tracker. The list offers 7 to 90 days, and **Custom…** any number of days up to 400, within the retention of the license usage log.
- **Outliers on these priorities**: every priority.
- **Outliers minimum history**: **15 days** — the history a model needs before its confidence is *normal*; the notice confirms that 30 days of backfill cover it, so the models reach normal confidence as soon as the backfill completes.
- **Outliers KPIs**: the rolling **24h volume** — one model per index.


Create a new TrackMe Virtual Tenant: Splunk Volume (Outliers)
Basics
2
Volume
3
Inactivity
4
Indexes & RBAC
5
Review
VOL tracks the licensed volume of every index from the license usage logs, in 5-minute buckets. Outliers on the rolling 24 hours volume detect abnormal drops and / or spikes, and the licensed volume is recorded per license day with a month projection. There are no use cases to choose: pick the environment and how the detection behaves.
Target environment
local
The Splunk deployment whose license usage is tracked: this instance, or a remote account.
Detect
Drops and spikes
The default detection direction for every index; each index can be changed afterwards.
Backfill history (days)
30 days
History replayed after creation, one license day at a time by the hourly tracker: the rolling volume the outliers train on and the daily licensed volume of the cost trend. The models are trained once the replay completes, so a long span takes longer to reach a normal confidence. Capped by the _internal retention.
Outliers on these priorities
critical
high
medium
low
Indexes of these priorities are in scope for volume outliers. All priorities by default: volume outliers are the point of this component.
Outliers minimum history (days)
15
Metrics history an outliers model needs before its confidence is normal: a low-confidence model does not add to the impact score. A tenant setting (the system default is 30 days): it applies to every component of this tenant and can be changed later in the tenant configuration. 1 to 365.
Outliers KPIs
Rolling 24h volume
Every index gets one outliers model per KPI selected here. The rolling 24h volume follows the daily pattern; the shorter windows react faster to a sudden drop or spike. Changeable later in Tenant Home (Manage: Outliers default KPIs).
With 30 days of backfill, the outliers models reach a normal confidence once the replay completes (the hourly tracker replays one day at a time).
Advanced
Back
Next

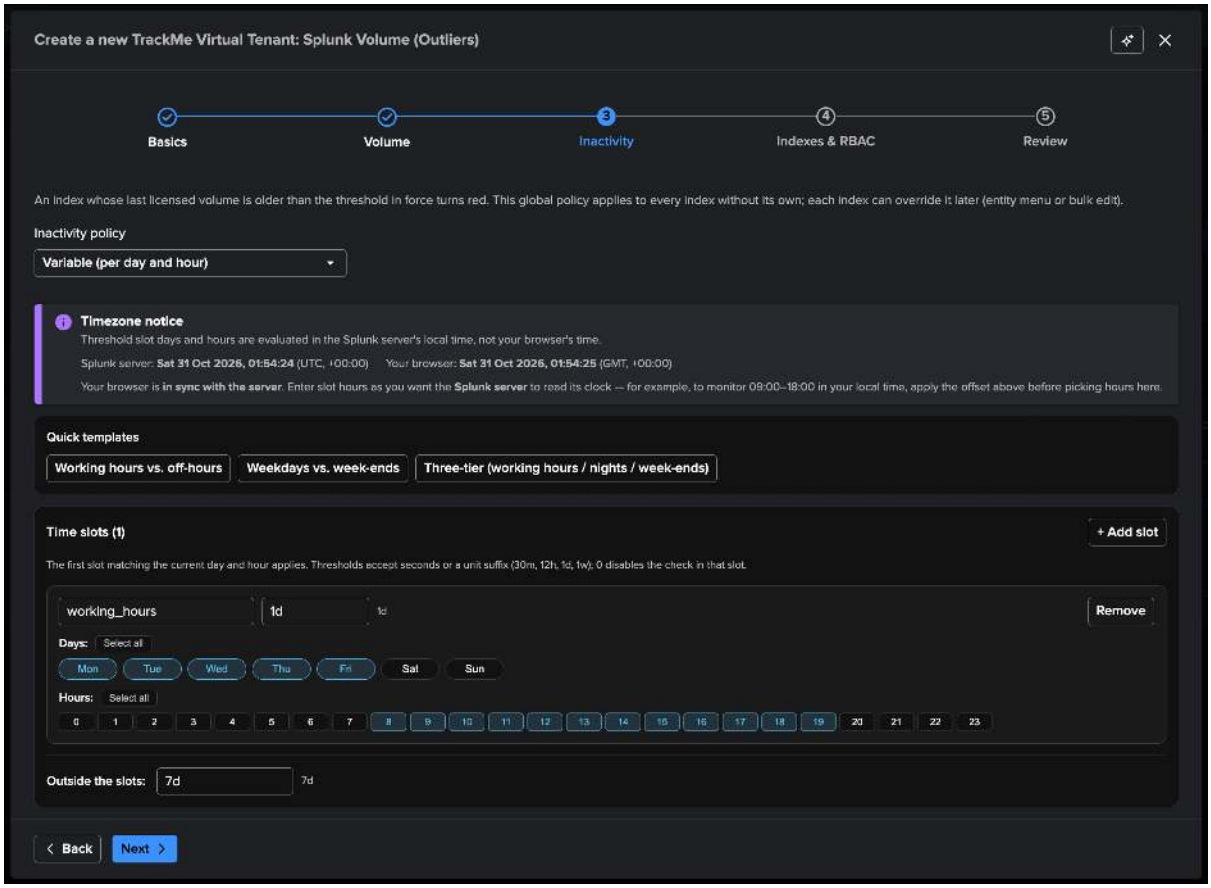

Create a new TrackMe Virtual Tenant: Splunk Volume (Outliers)
VOL tracks the licensed volume of every index from the license usage logs, in 5-minute buckets. Outliers on the rolling 24 hours volume detect abnormal drops and / or spikes, and the licensed volume is recorded per license day with a month projection. There are no use cases to choose: pick the environment and how the detection behaves.
Target environment
local
The Splunk deployment whose license usage is tracked: this instance, or a remote account.
Detect
Drops and spikes
The default detection direction for every index; each index can be changed afterwards.
Backfill history (days)
30 days
History replayed after creation, one license day at a time by the hourly tracker: the rolling volume the outliers train on and the daily licensed volume of the cost trend. The models are trained once the replay completes, so a long span takes longer to reach a normal confidence. Capped by the _internal retention.
Outliers on these priorities
critical
high
medium
low
Indexes of these priorities are in scope for volume outliers. All priorities by default: volume outliers are the point of this component.
Outliers minimum history (days)
15
Metrics history an outliers model needs before its confidence is normal: a low-confidence model does not add to the impact score. A tenant setting (the system default is 30 days): it applies to every component of this tenant and can be changed later in the tenant configuration. 1 to 365.
Outliers KPIs
Rolling 24h volume
Every index gets one outliers model per KPI selected here. The rolling 24h volume follows the daily pattern; the shorter windows react faster to a sudden drop or spike. Changeable later in Tenant Home (Manage: Outliers default KPIs).
With 30 days of backfill, the outliers models reach a normal confidence once the replay completes (the hourly tracker replays one day at a time).
Advanced
Settle margin (seconds)
300
License usage events newer than now minus this margin are not read yet, so late events are counted. 60 to 3600, default 300.
License search constraint
1 index=_internal source="*/license_usage.log"
The base search giving access to the license usage events: every license search of the tracker is formed from it. Indexed fields only (index, source, sourcetype, host). Change it only to target another data set with the license_usage.log format, whose key=value fields (type, idx, b, pool) are extracted at search time. Default: index=_internal source="*/license_usage.log"
Back
Next

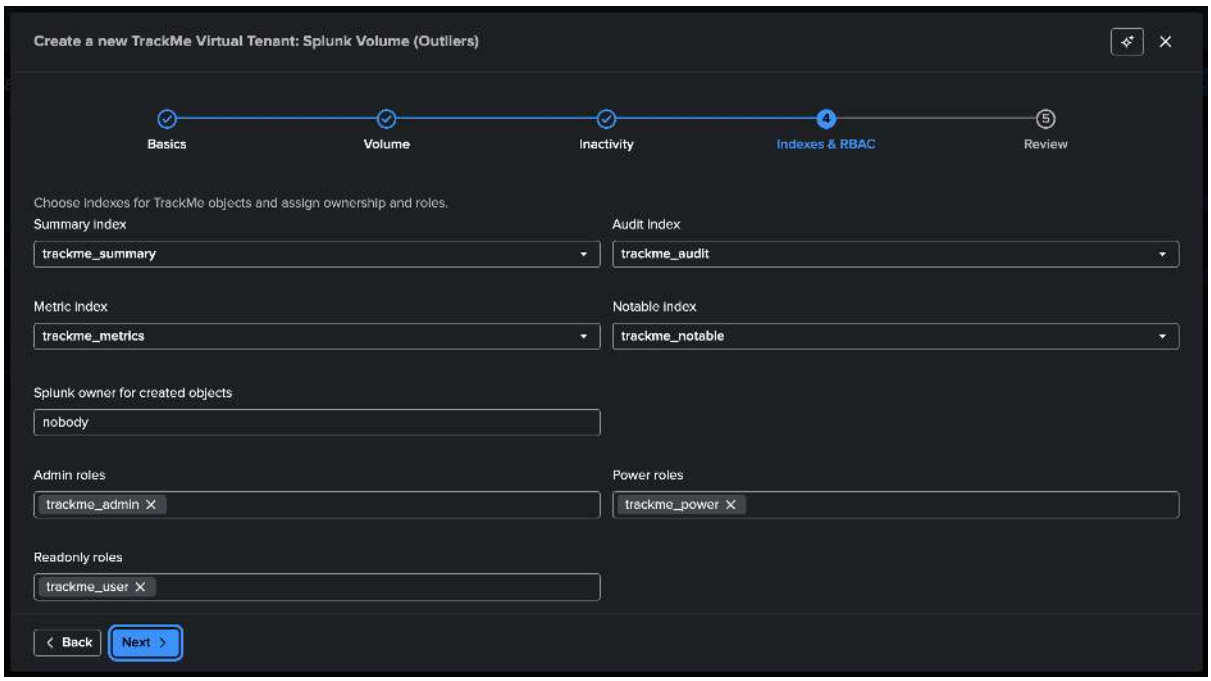


Note

Advanced: the license search constraint. Every license search of the tenant is formed from one constraint, `index=_internal source="*/license_usage.log"` by default. It can be changed here (and later in *Configure tenant*) to point the tenant at another data set holding events in the `license_usage.log` format — a copy forwarded to a dedicated index, for instance. It stays a plain base search naming an index; the wizard validates it as you type. Leave it at its default unless you have such a setup — see *The license search constraint (advanced)*.

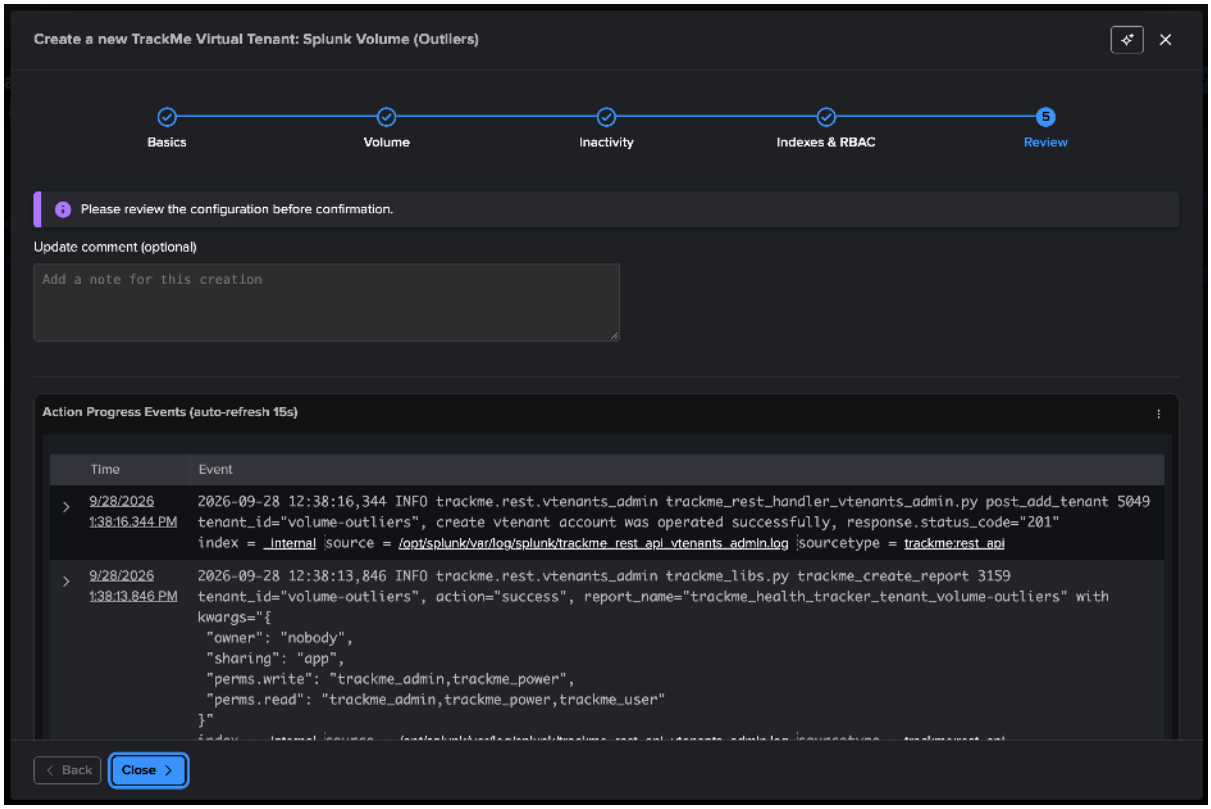
The **Inactivity** step sets after how long without licensed volume an index turns red. The default policy is **variable**: **1 day during working hours** (Monday to Friday, 08:00–19:59 in the server's time zone) and **7 days** for nights and week-ends — a business-hours feed that stops on a Tuesday morning is red the next day, a Friday-evening stop waits for Monday. Quick templates give a static threshold or other common shapes; each index can override it later:





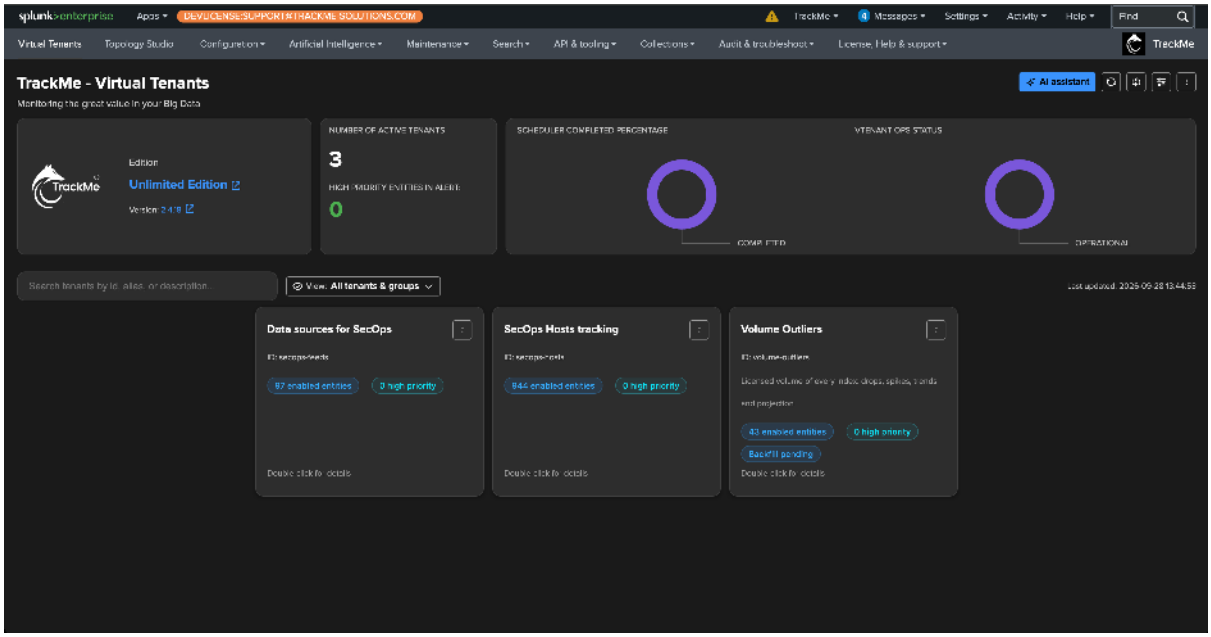
Indexes & RBAC and **Review** are the steps every tenant ends with:

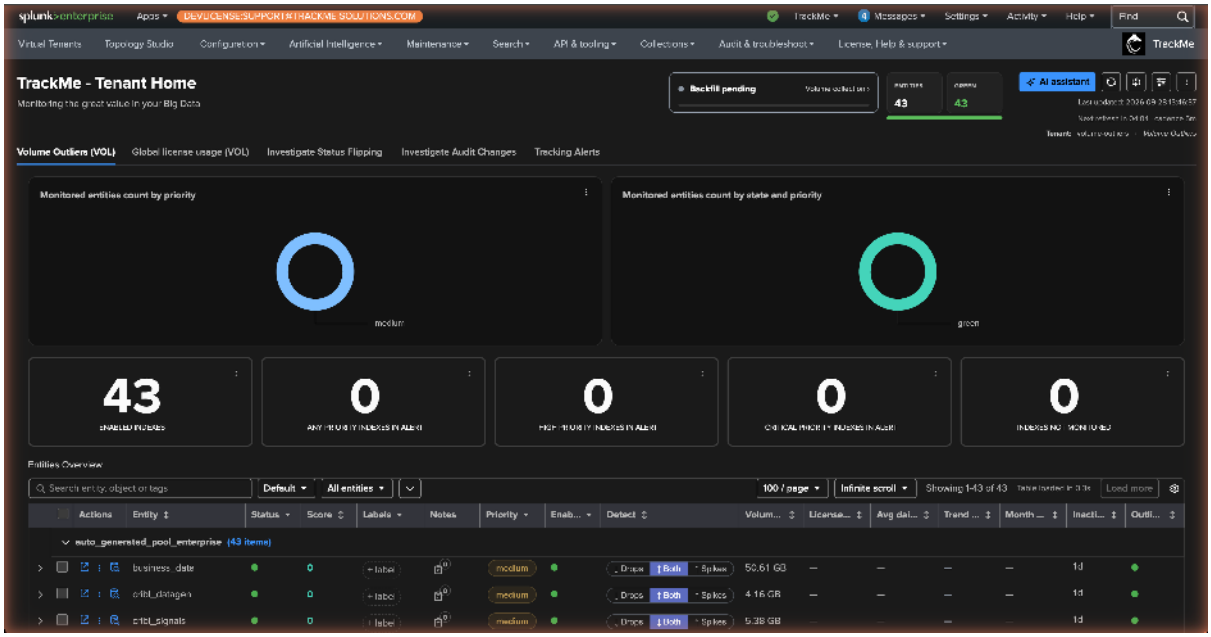




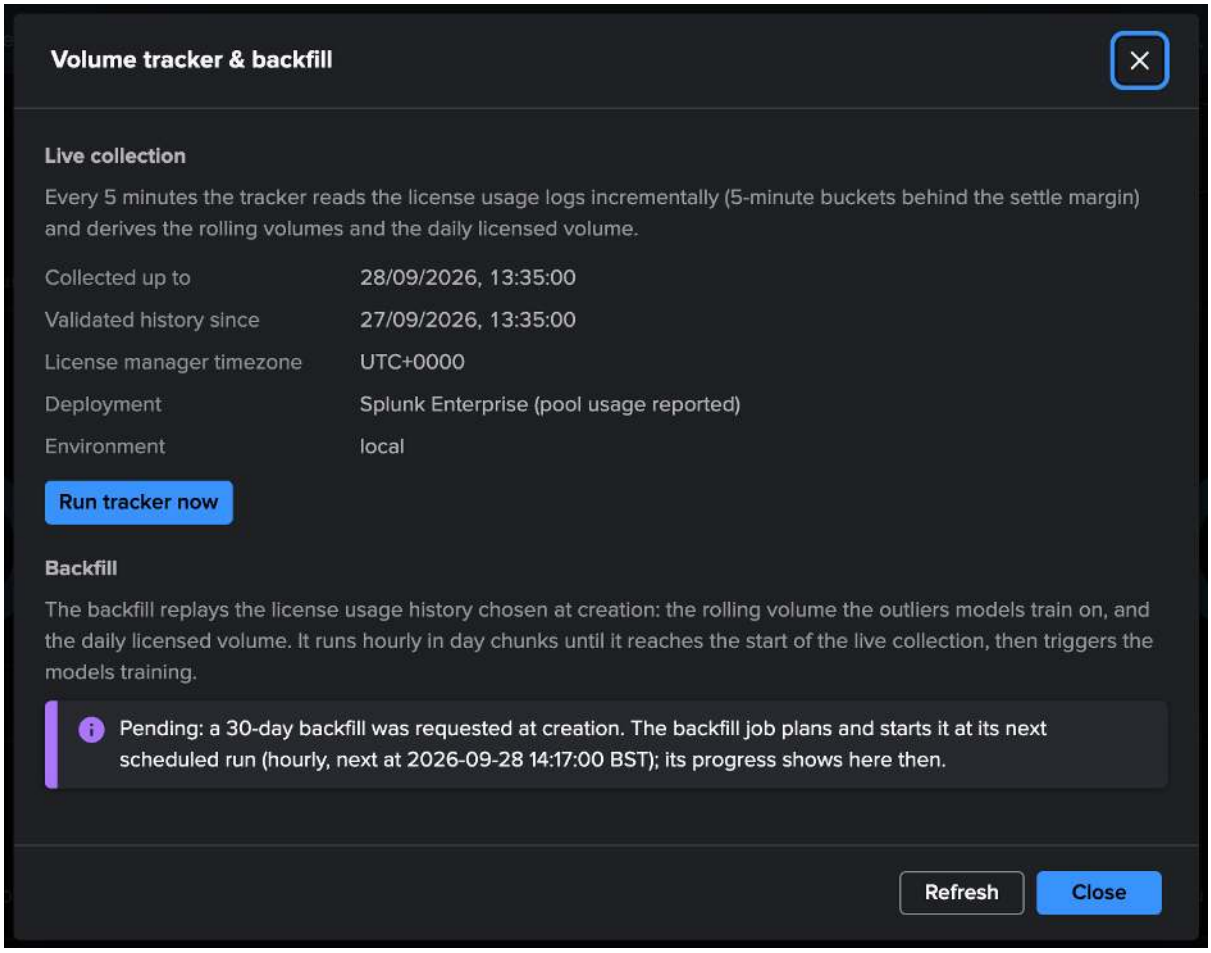
Step 2 — Day one: the first collection and the backfill

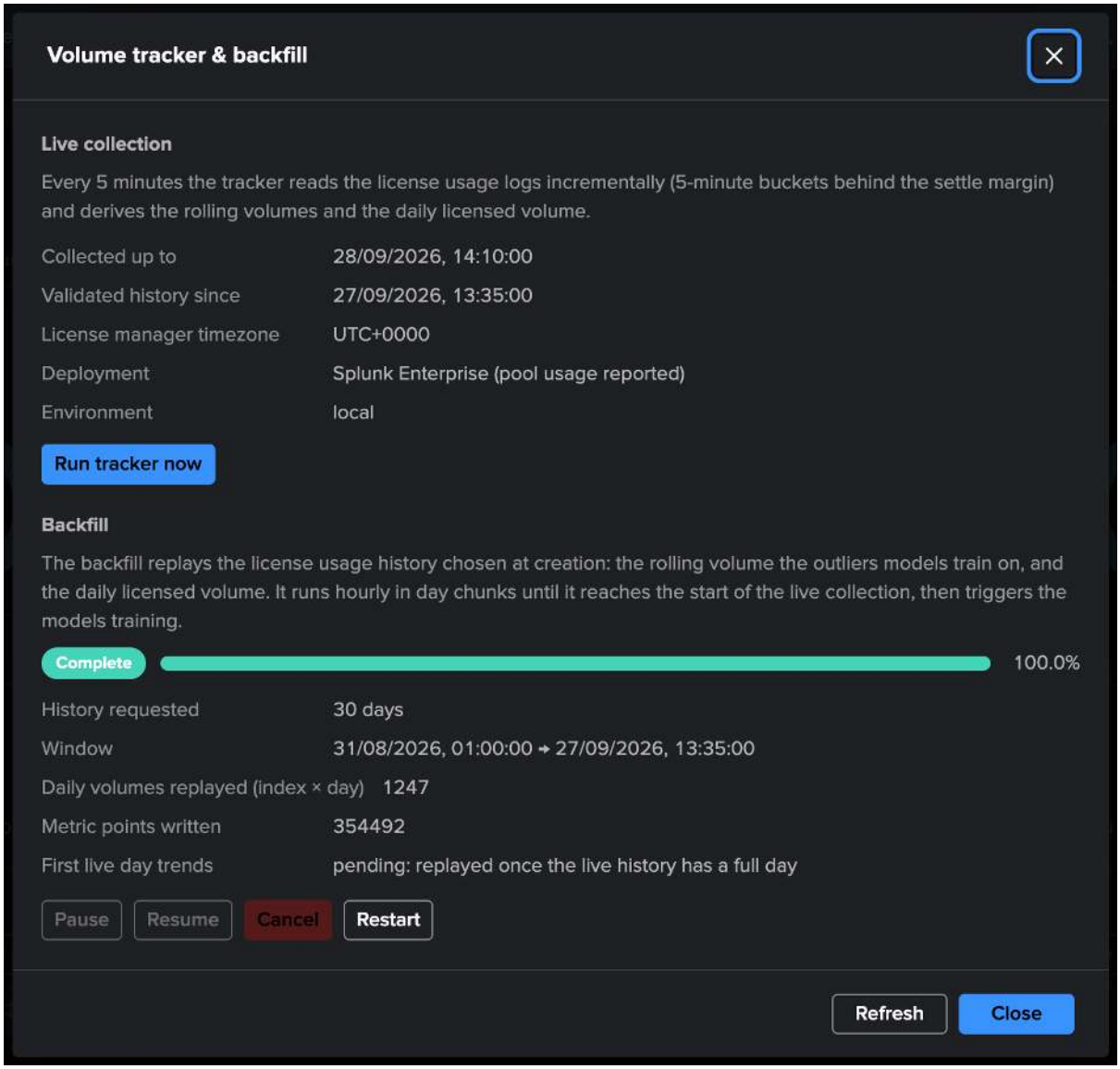
The creation runs the first collection at once: the tracker seeds the last 24 hours of the license usage log, one entity per index appears, and the rolling volumes are already there. The backfill job plans itself at its first hourly run — until then the tenant card reads **Backfill pending**, and so does the vignette in the Tenant Home header:



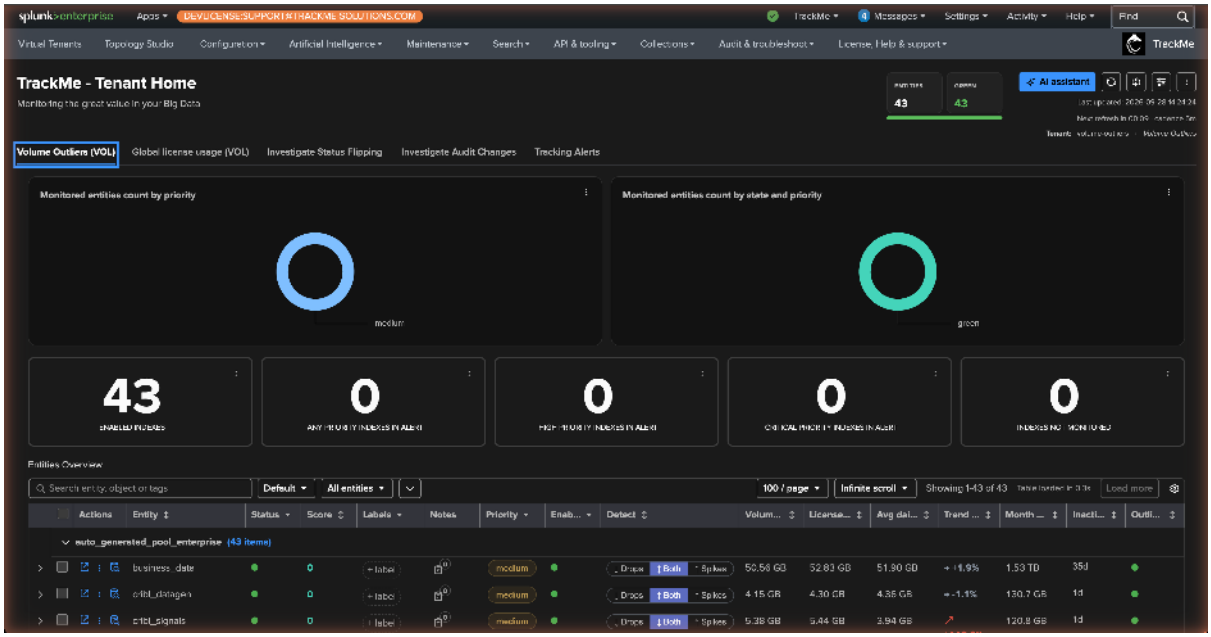


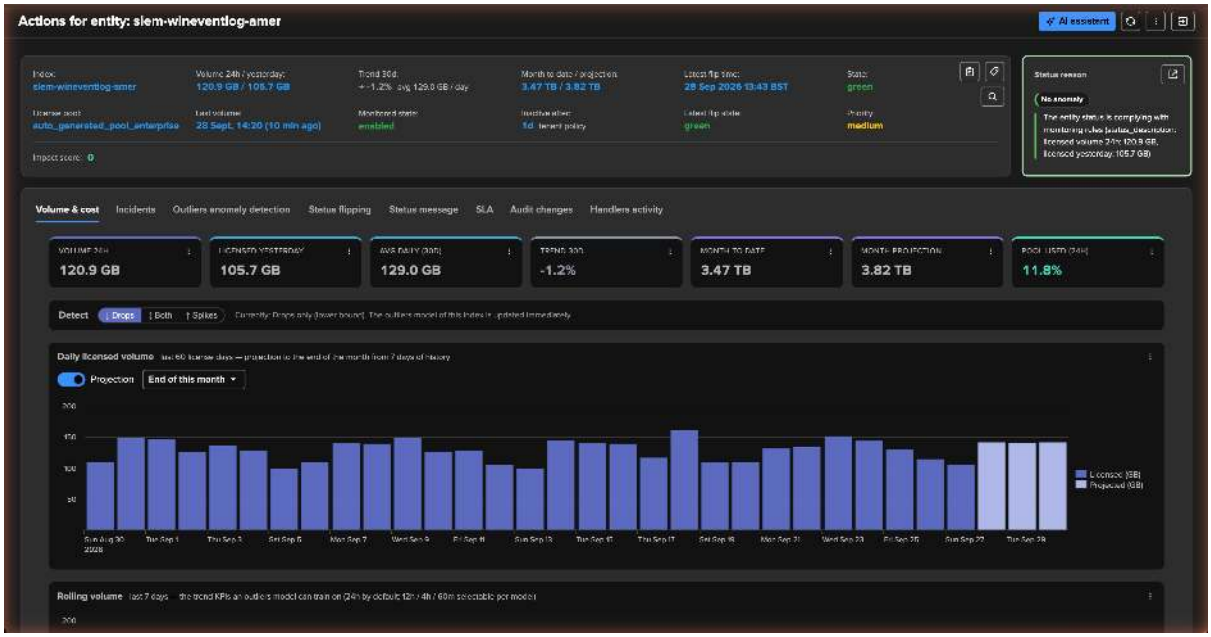
Clicking the vignette (or **Manage: volume collection** in the Tenant Home menu) opens the collection screen: the live cursor, the validated history, the license manager's UTC offset, the deployment type, and the backfill with its progress and controls. At its first hourly run the backfill job plans the 30 days and replays them day by day; on a large deployment this spans several hourly slots and the card shows the percentage — on our environment the whole month replayed in under two minutes, 28 day-chunks and about 350,000 metric points, and the screen already reports it complete:


Volume tracker & backfill
Live collection
Every 5 minutes the tracker reads the license usage logs incrementally (5-minute buckets behind the settle margin) and derives the rolling volumes and the daily licensed volume.
Collected up to
28/09/2026, 13:35:00
Validated history since
27/09/2026, 13:35:00
License manager timezone
UTC+0000
Deployment
Splunk Enterprise (pool usage reported)
Environment
local
Run tracker now
Backfill
The backfill replays the license usage history chosen at creation: the rolling volume the outliers models train on, and the daily licensed volume. It runs hourly in day chunks until it reaches the start of the live collection, then triggers the models training.
Pending: a 30-day backfill was requested at creation. The backfill job plans and starts it at its next scheduled run (hourly, next at 2026-09-28 14:17:00 BST); its progress shows here then.
Refresh
Close



When the backfill completes, its days are merged into every index and every VOL model is trained. The **Volume & cost** tab of any index now shows a full month of daily licensed volume, the 30-day trend and the month projection, and the outliers chart of the index's model with its learned bounds:





Hint

Nothing was written twice. The live collection and the backfill never overlap: the backfill ends where the live collection started, the first backfilled day is aligned on a license-day start, and every complete license day emits exactly one daily point. The rolling KPIs are emitted only once their whole window lies inside validated history, so no ramp-up ever reaches the series the models train on.

Step 3 — Reading an index

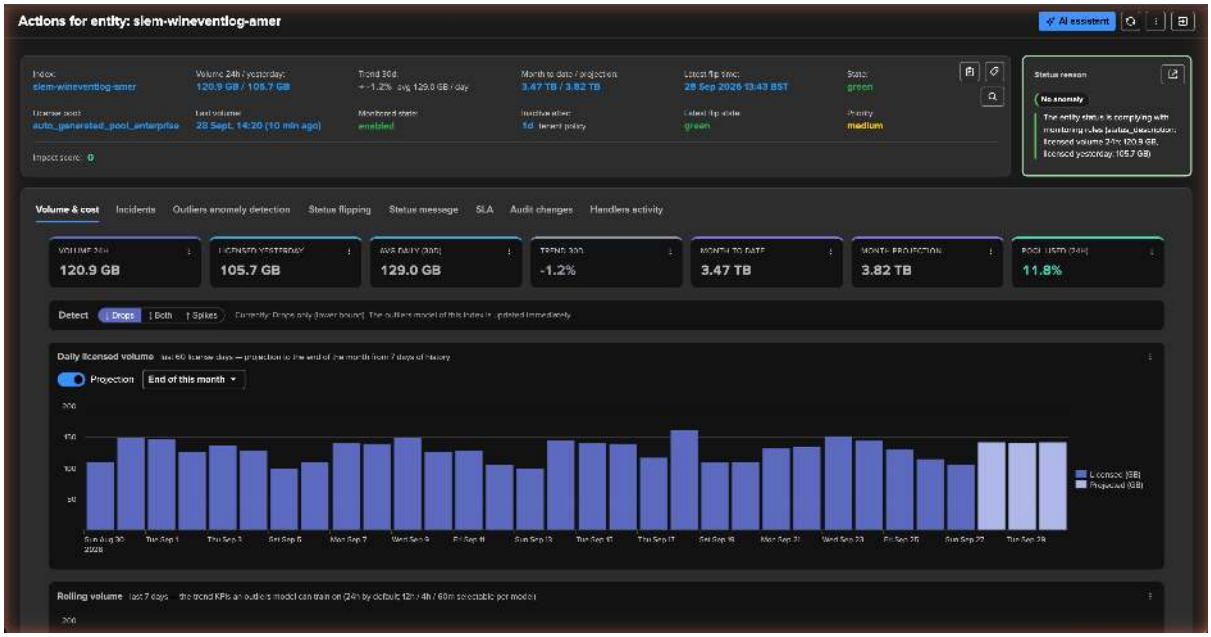
The entity modal of an index opens on **Volume & cost**:

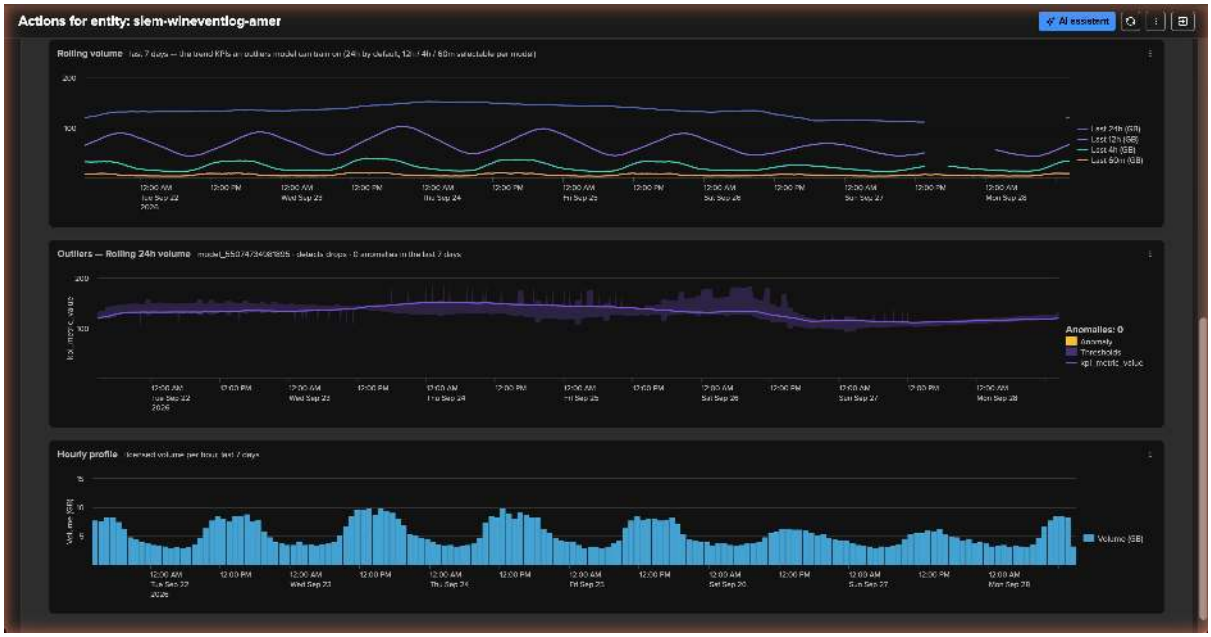
- **Key figures** — the rolling 24-hour volume, licensed yesterday, the last volume seen, the 30-day trend and daily average, the month to date and the month projection, the inactivity threshold in force and where it comes from.
- **Detect** — the direction toggle of this index.
- **Daily licensed volume** — one bar per license day (the day rolls at the license manager's midnight), with the **projection** to the end of this month or of next month, scaled to the collector's figure so the chart and the tile

agree. *Open in Search* gives the SPL, projection included.

- **Rolling volume** — the four trends (60 min, 4 h, 12 h, 24 h) together over the last week.
- **Outliers** — one chart per model: the KPI, the learned bounds, the anomalies, the confidence.
- **Hourly profile** — the volume per hour, the shape a spike or a drop deviates from.

Then the standard tabs — Outliers anomaly detection, Incidents, Status flipping, Status message, SLA — as for any entity.

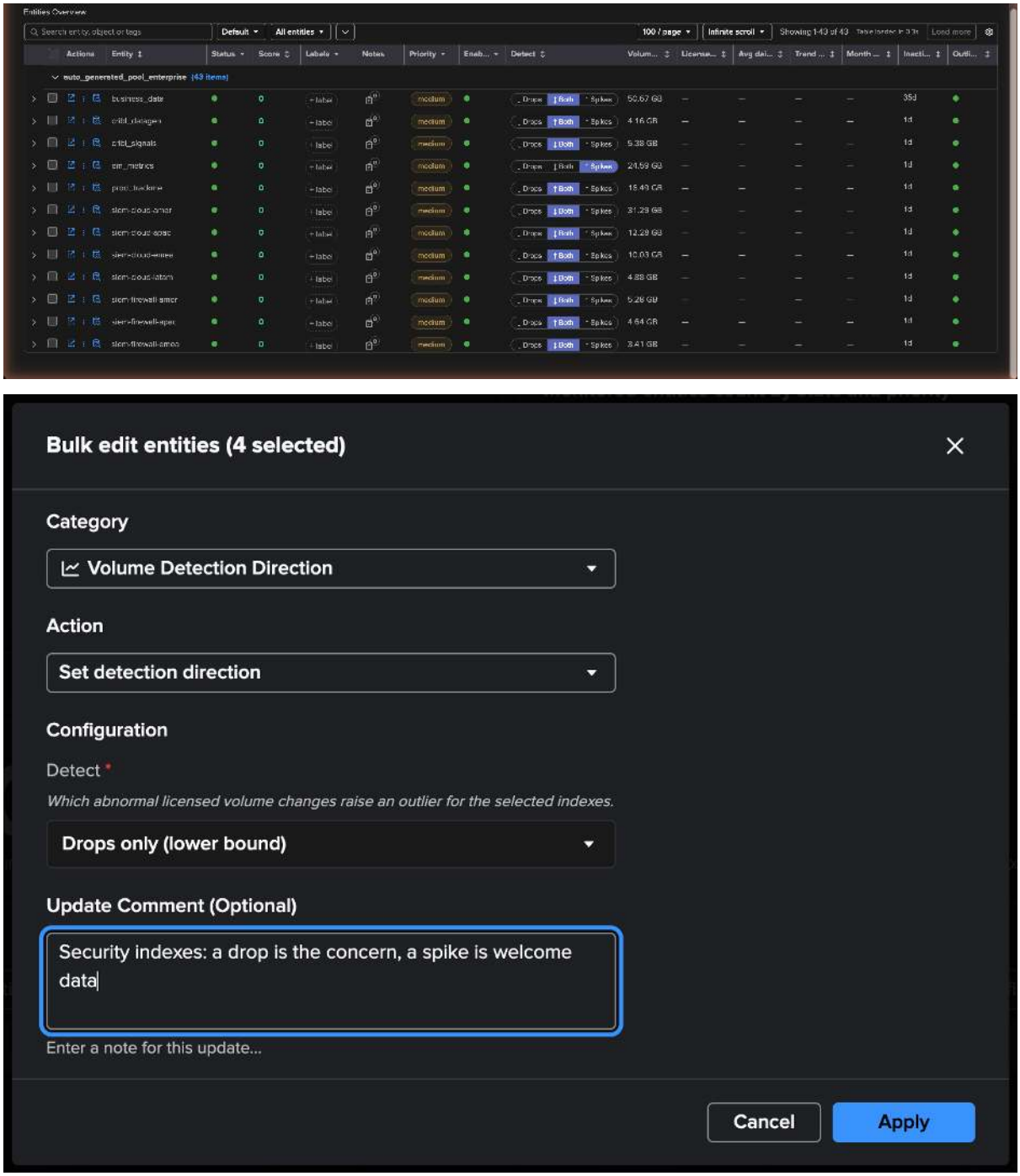




Every chart draws its volumes in one unit — MB, GB or TB — picked from its largest value, so a small index is not a flat line of `0.00 GB` and a large one does not read in millions of MB.

Step 4 — Choosing what to detect, per index

Not every index deserves both directions. A **security** index is a drop concern — a spike is welcome data; a **development** or **metrics** index is a spike concern — a drop costs nothing. The **Detect** toggle of the VOL table switches an index between *drops*, *both* and *spikes* with one click, and the **bulk edit** action *Detection direction* applies a choice to a selection:

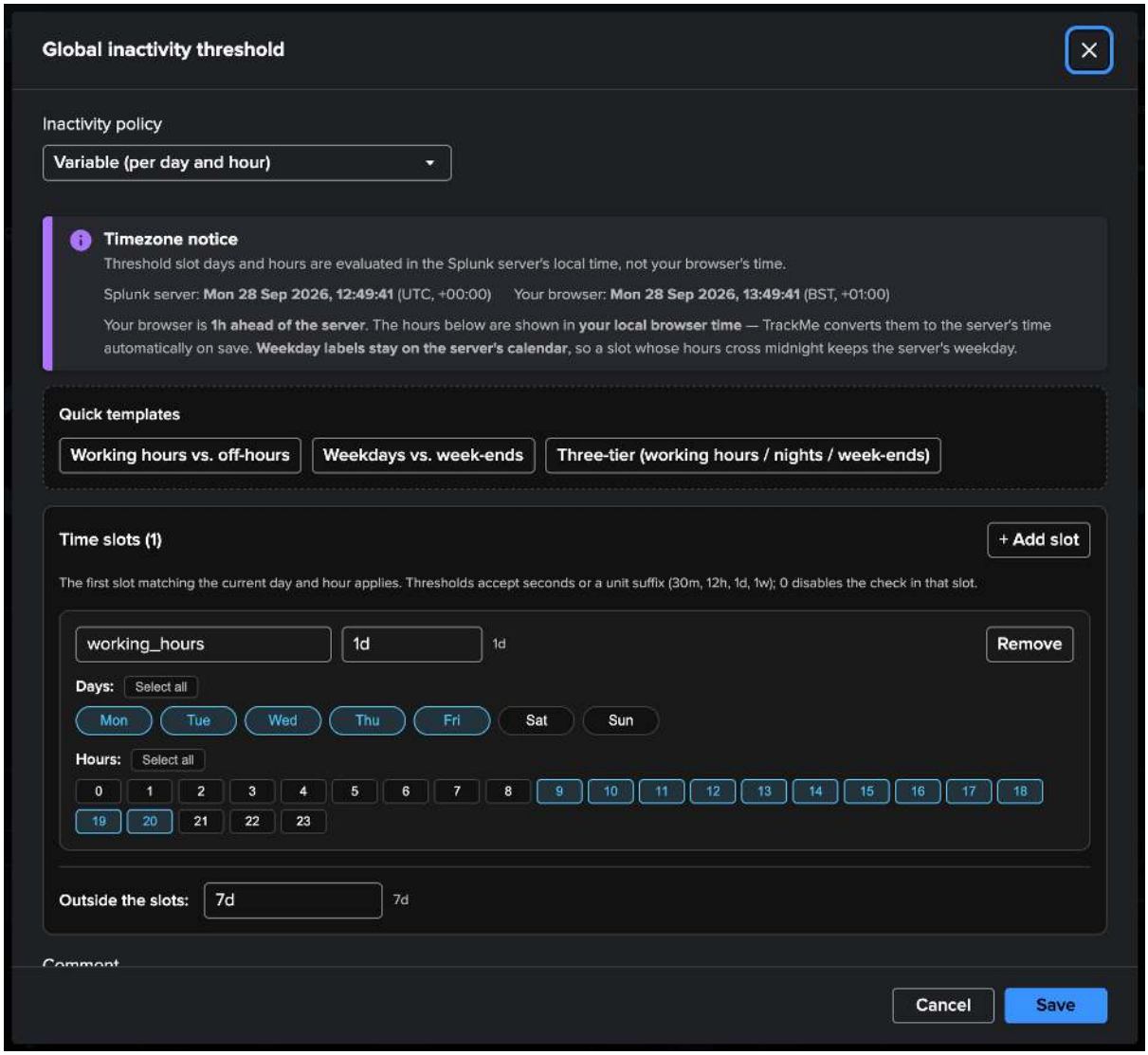


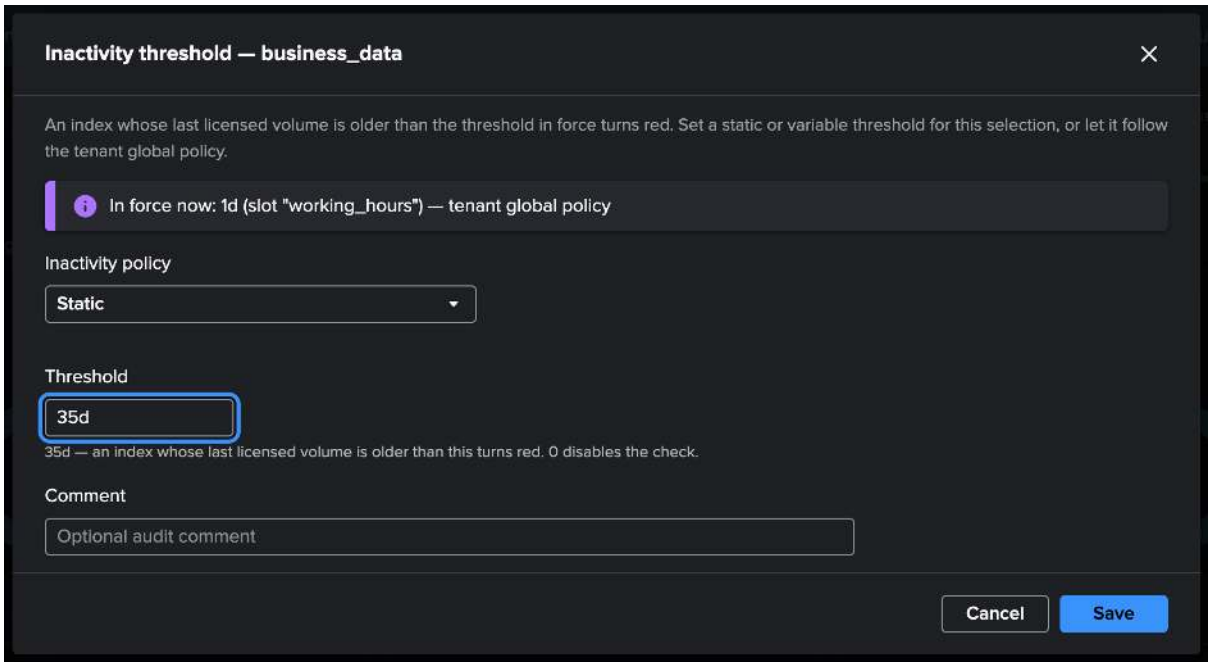
The direction is stored on the index's model (the lower bound for drops, the upper bound for spikes); the tenant default applies to every new index. Under the hood the models are the standard TrackMe outliers models — the **Outliers anomaly detection** tab manages them (simulation, retraining, false positives), and the **AI ML Advisor** reviews them like any other.

Step 5 — Inactivity: the silent index

An index that stops consuming license does not produce a drop for long: after 24 hours its rolling volume is 0 and stays there — a lower-bound outlier the first day, then a flat line the model accepts. **Inactivity** is the second detection: the index turns **red** once its last licensed volume is older than the threshold in force.

The tenant policy set in the wizard (1 day in working hours, 7 days otherwise) applies live to every index. **Manage: Global inactivity threshold** in the Tenant Home menu edits it — the slots on a day / hour grid, drawn in the server's time zone — and an index gets its own override from its menu (*Modify → Inactivity threshold*) or from the bulk edit category *Inactivity Threshold*: a monthly batch index, for instance, gets a static 35-day threshold.



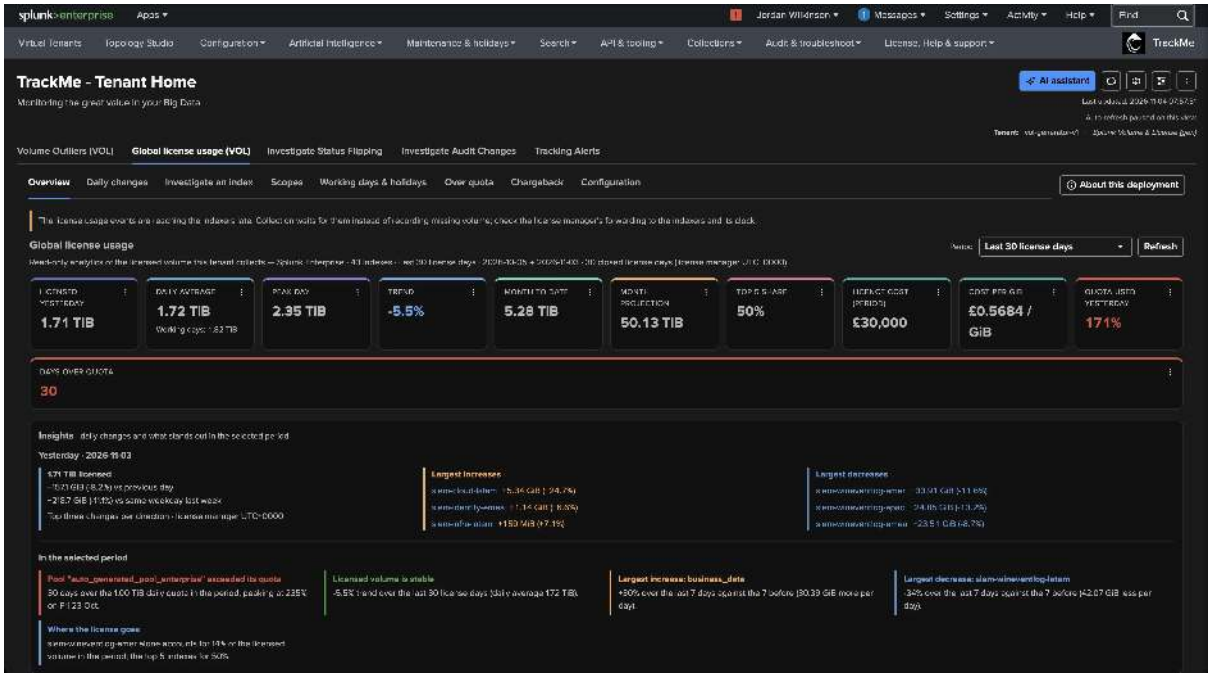


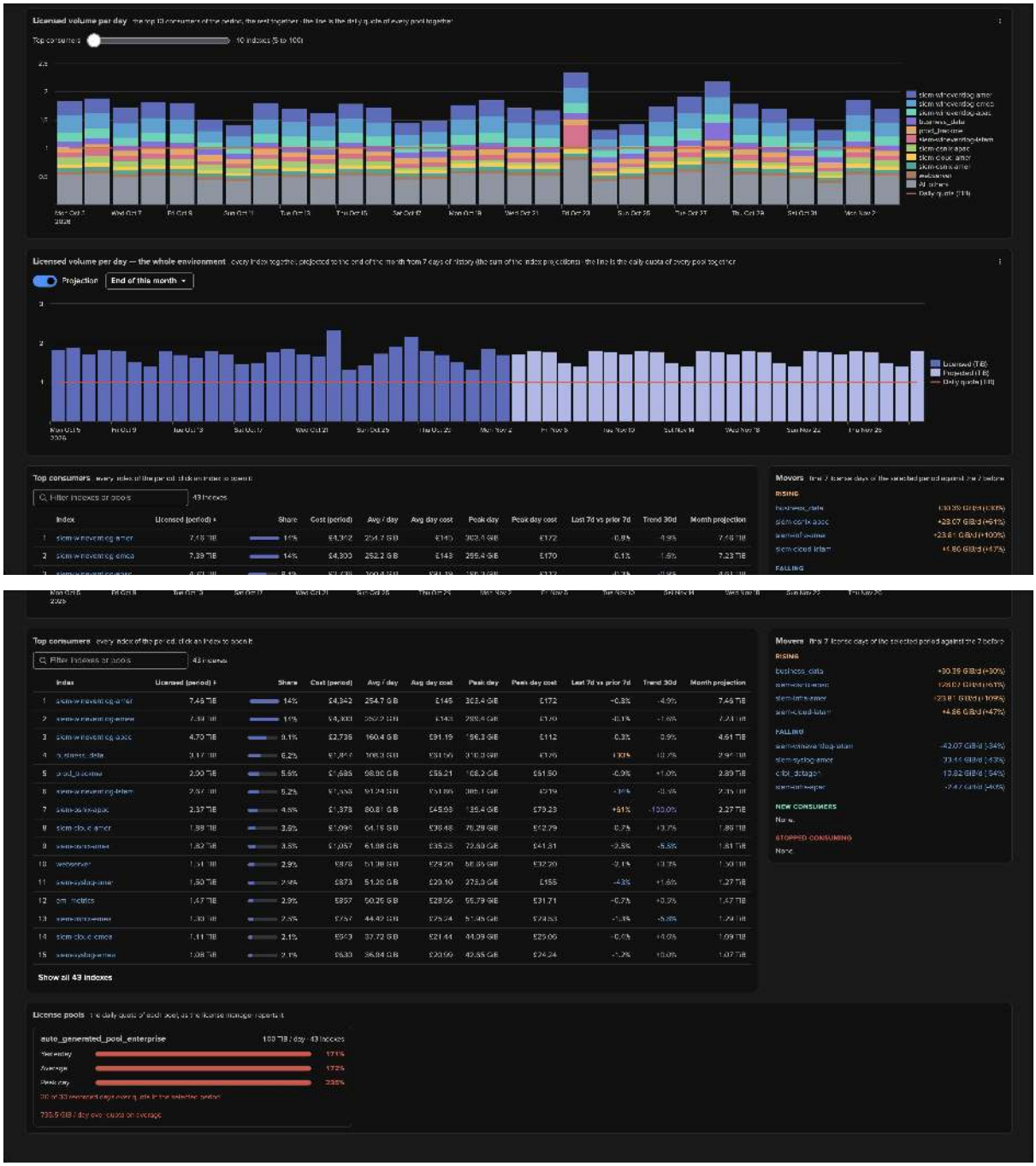
The table's **Inactivity** column shows the threshold in force per index (the slot and the source in its tooltip), and the red status message names the slot — *silent for 1d 4h, threshold 1d (working_hours), set for this index*.

Step 6 — Global license usage

The **Global license usage (VOL)** tab, next to the component tab, is the environment view — the dashboard that every Splunk administrator builds by hand at some point, computed here from what the tenant already collected (no extra search on the license usage log):

- the **key figures**: licensed yesterday, daily average, peak day, the trend of the environment, month to date and the month projection, the share of the top 5 consumers, and on Splunk Enterprise the quota used yesterday;
- the **insights**, in plain words — the trend, days over quota, the largest increase and decrease, indexes that stopped consuming, new consumers, the concentration, the weekly pattern;
- **Top consumers per day**: the stacked daily volume of the top consumers, the slider from 5 to 100, the rest as *All others*, with the daily quota line on Splunk Enterprise;
- **The whole environment per day**, with the same projection controls as the entity chart and the quota line;
- the **consumers** table, the week-over-week **movers**, and the **pool** cards with their usage meters, days over quota and headroom.





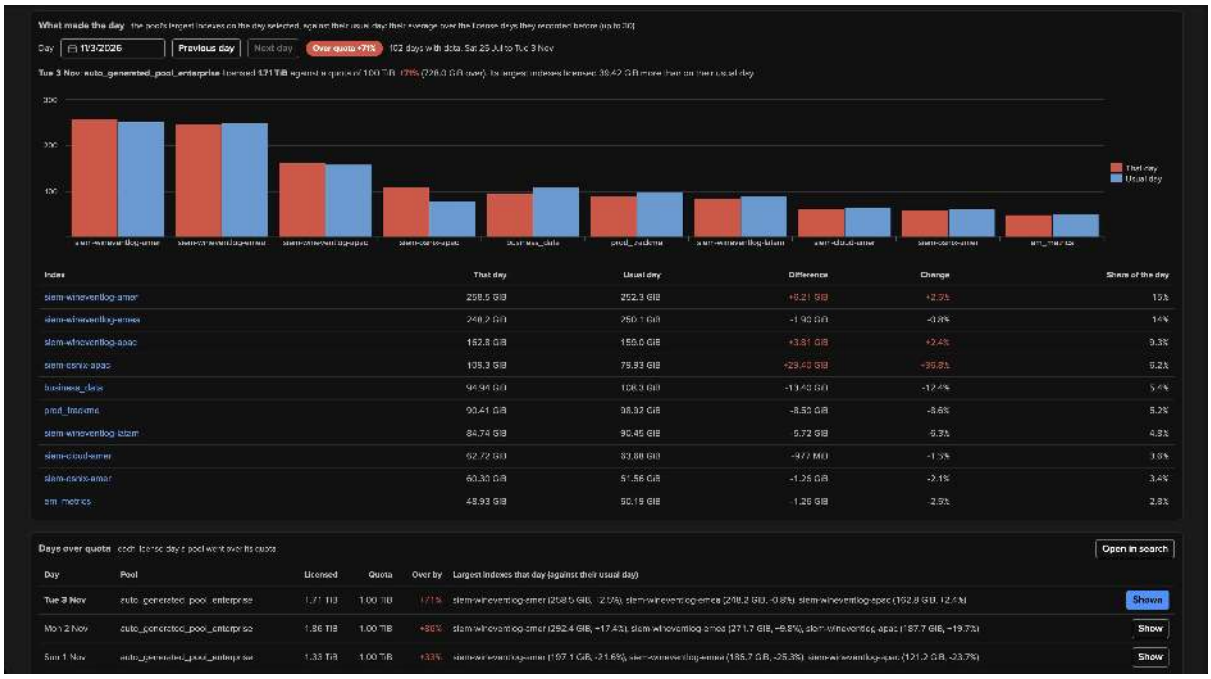
On **Splunk Cloud**, where a pool has no limit, the quota line and the pool meters are not drawn and the view says so, until you configure a daily quota per pool (below).

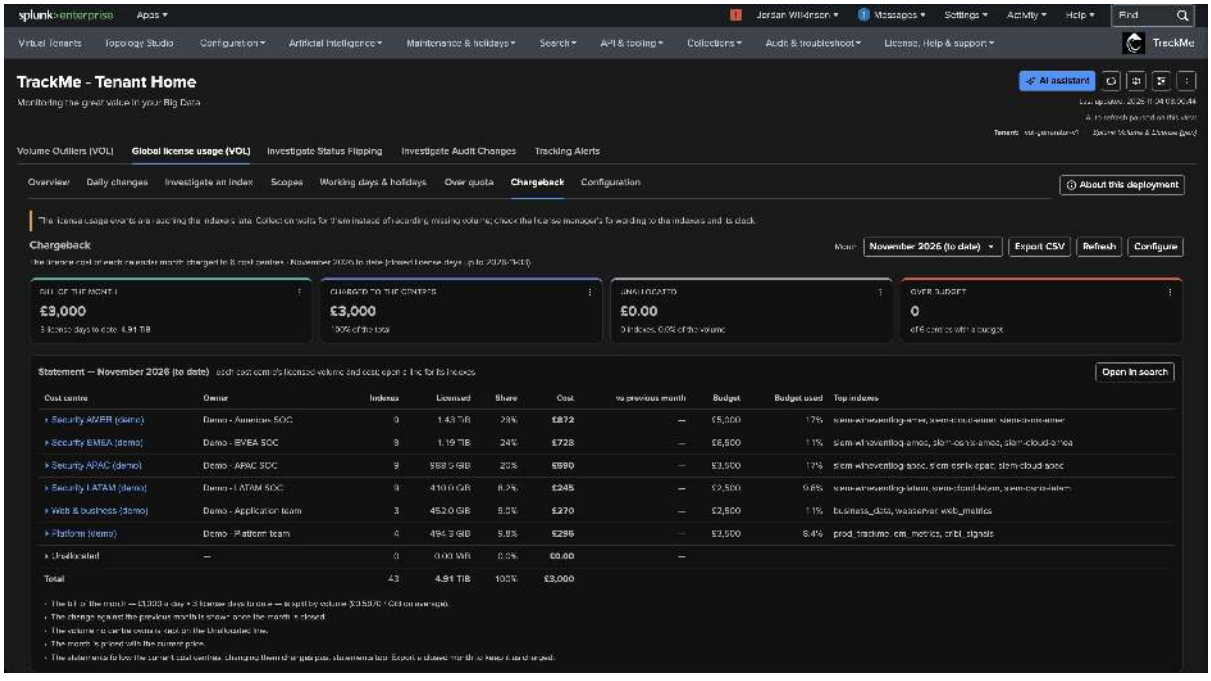
From volume to cost

The overview is one of six sub-tabs. The others turn the same figures into answers for the people who own the license and its budget; everything is opt-in, set once in the **Configuration** sub-tab (power or admin role on the tenant), or by asking the AI Assistant:

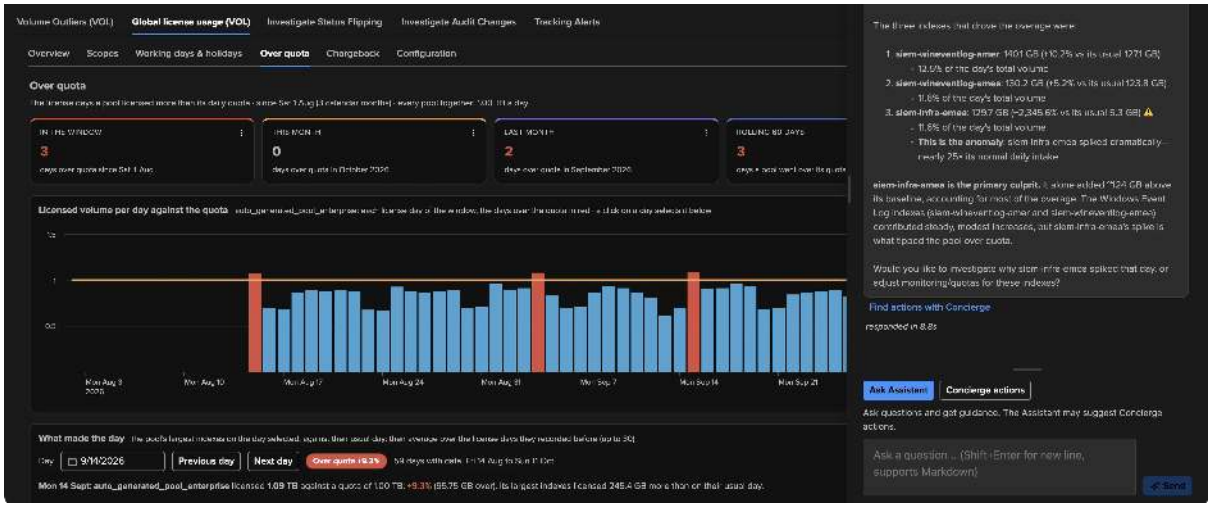
- **Scopes —** name a set of indexes (a team, a use case, the data of Enterprise Security, from index patterns or from the CIM datamodel macros) and get its share of the license, its daily average and peak, its trend and its month projection on its own tab.
- **Working days & holidays —** the working-day, weekend and holiday averages, with the holidays of the tenant's *Bank Holidays* calendars: a quiet Easter Monday no longer drags the daily average down.

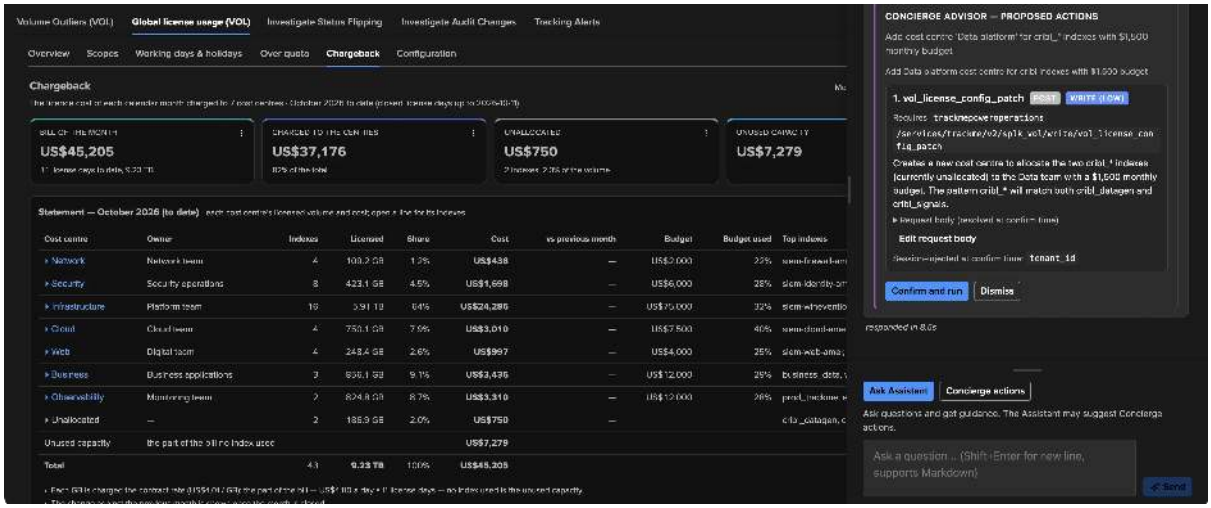
- **Over quota** — the days a pool went over its daily quota, per day, per month and over a rolling 60 days, and **What made the day**: the pool's largest indexes on the day you pick, against their usual day. On Splunk Cloud, a daily quota per pool gives the stack the volume it was sized for.
- **Chargeback** — with a price (annual licence cost or cost per GB, with its history) and a list of cost centres, a monthly statement of who pays for what over 13 calendar months, with budgets, the unallocated volume, the unused capacity and a CSV export.





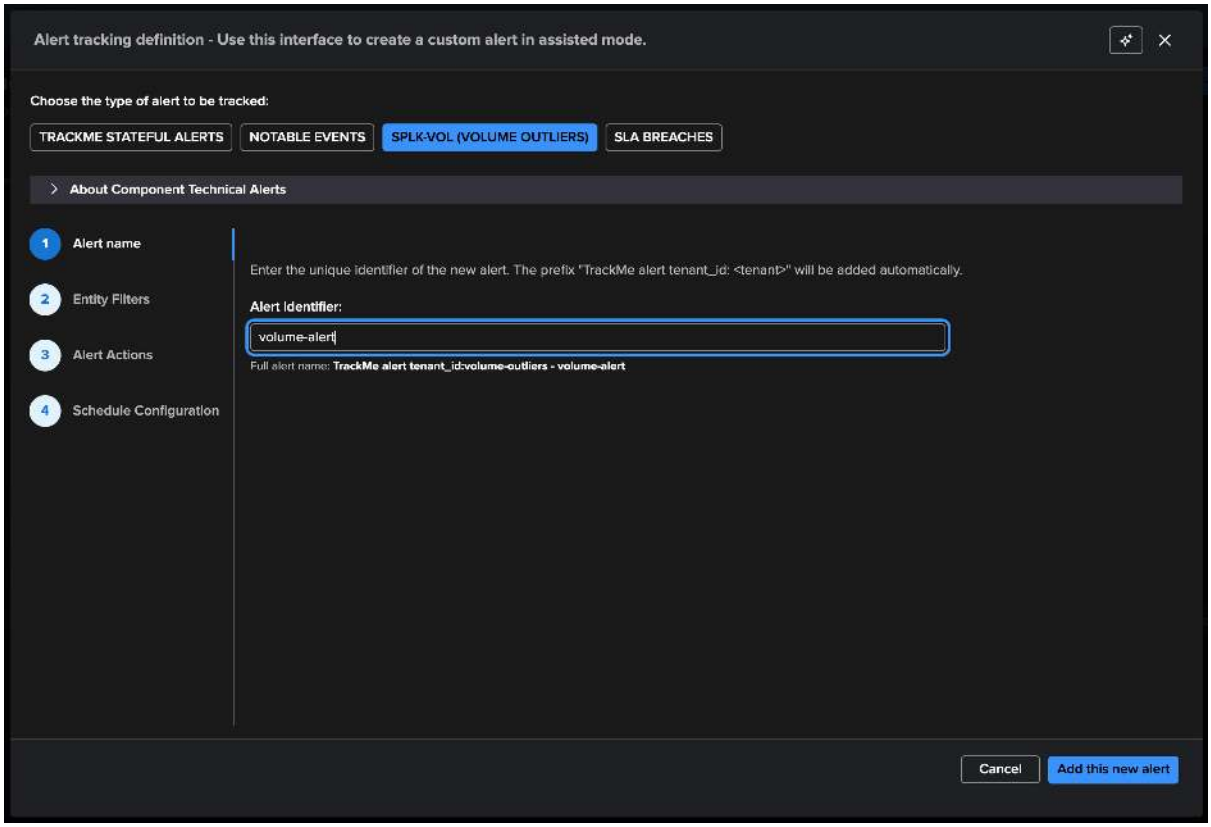
While the tab is open, the Tenant Home **AI Assistant** knows the sub-tab and the figures on screen: ask it why yesterday went over quota, or to create a cost centre for the security indexes, and it proposes the change for you to approve. The full reference is in *Global license usage (VOL)*.

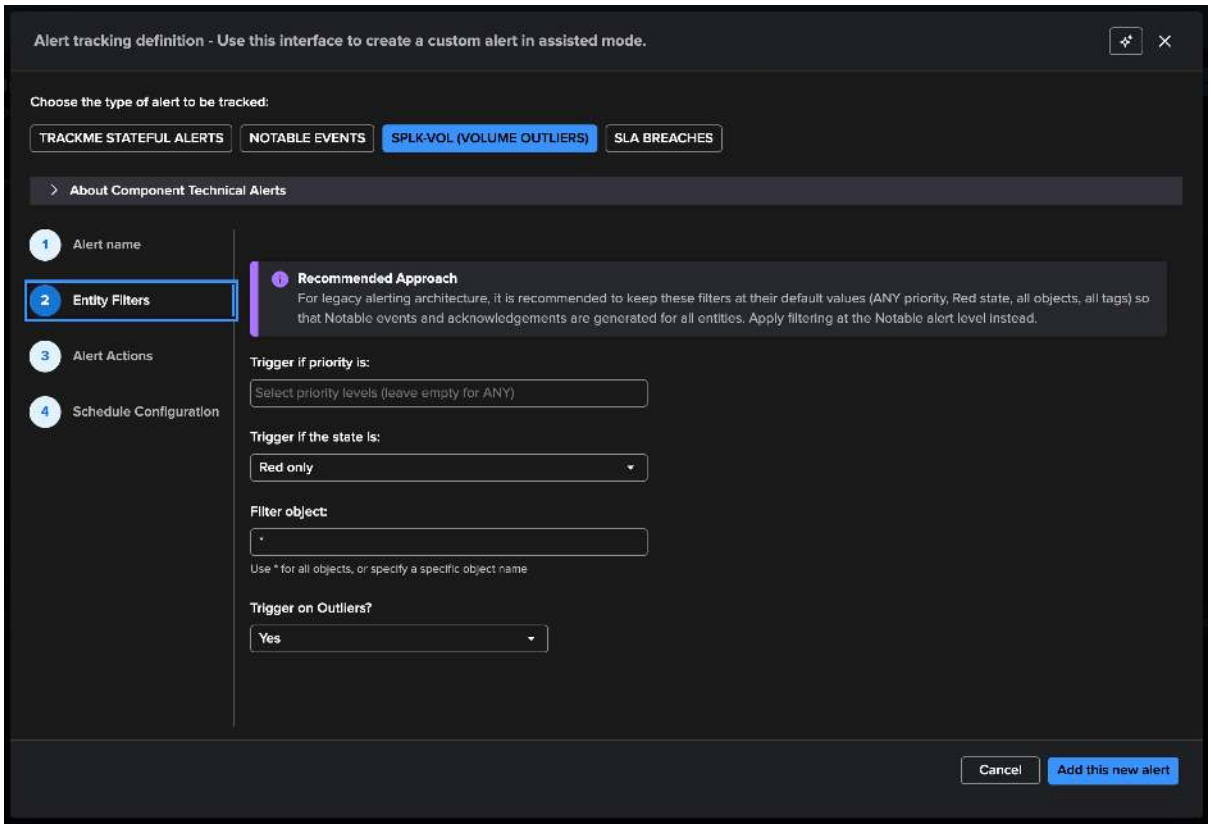




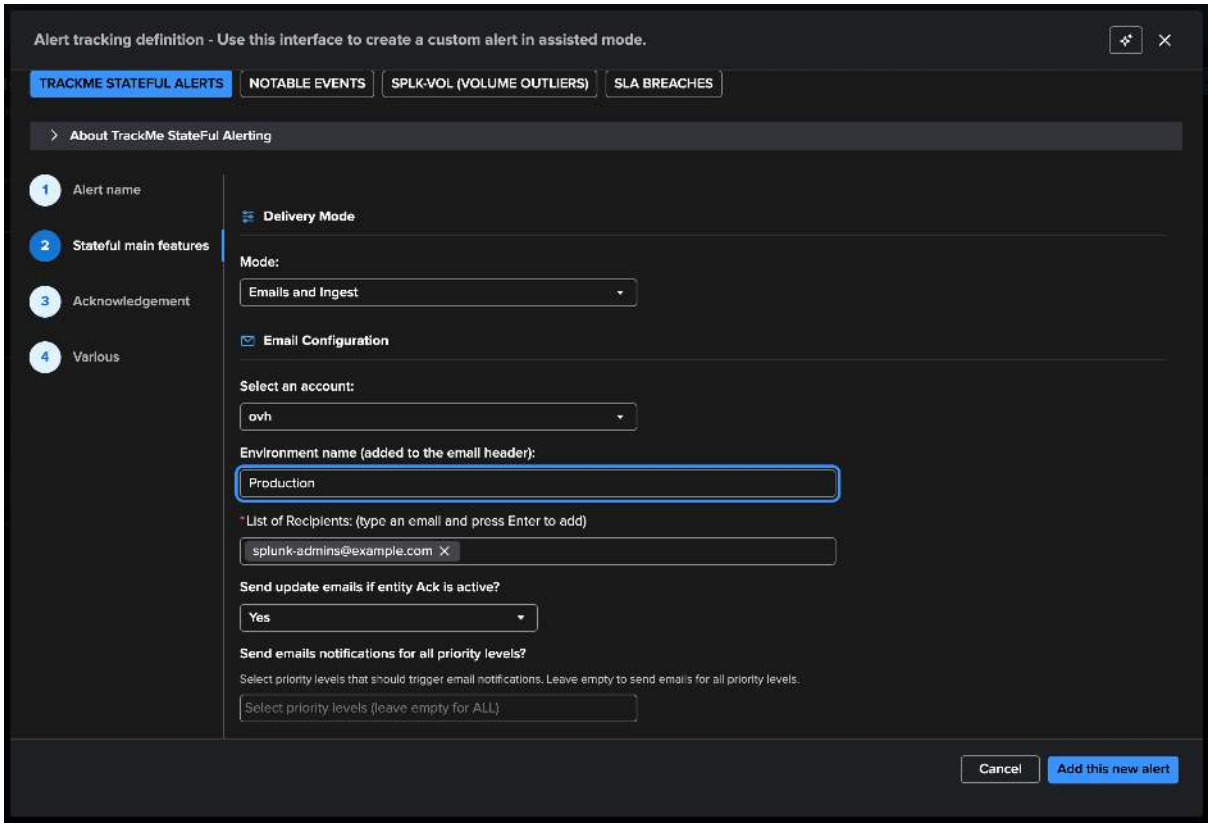
Step 7 — Alerting

A Volume Outliers tenant alerts like any other: from the **Tracking Alerts** tab, create an alert of type `SPLK-VOL`. The entity filters keep **Trigger on Outliers** on — the outliers *are* the state of this component — and the same alert covers an index turning red for inactivity. Keep the recommended pattern: the component alert generates the **Notable events** (with auto-acknowledgment), and a **Notable alert** forwards them to your ticketing or messaging tool; or send **stateful alert emails** directly, with the rolling volume charts, the outliers chart and the daily licensed volume of the index embedded:





For the notifications themselves, a **TrackMe stateful alert** with the *Emails and Ingest* delivery mode: the email delivery account, the environment name stamped in the email header, the recipients, and the charts and AI status report options — the emails of a Volume Outliers entity embed its rolling volume, its outliers chart and its daily licensed volume:



See *Alerting — in depth* for the alert types and the notification anatomy, and *SLA alerting* for a two-tier design.

Operating the tenant

- **Manage: volume collection** — the live cursor and validated history, the license manager's offset and the deployment type, *run the tracker now*, and the backfill with pause / resume / cancel / **restart** (a restart only extends the history backward; it never rewrites what was written).
- **Manage: Outliers default KPIs** — the rolling KPIs a new index gets a model on, with an opt-in apply to the existing indexes (adds the models of newly selected KPIs, removes the models of deselected ones).
- **Manage: ML Outliers scope** — on a VOL tenant it edits the VOL pair: the priorities and the filter expression (`object`, `priority`, `tags`, `labels`, `pool`).
- **Configure tenant** — every `vol_*` option; changing the target environment or the license search constraint re-seeds the collection from the new source and cancels an open backfill plan (restart it once the live collection runs on the new source).
- **Time zones** — the daily volumes follow the license manager's license day, and the inactivity slots follow splunkd's system time zone: your Splunk user's time zone changes where a bar is drawn, never what it measures.

Cost and scale

- **Collection**: one bounded search per tenant every five minutes reading only the new `Usage` events — on a large distributed deployment, about 76 thousand events per run where the previous Flex recipe read about 4.4 million (three appended scans over 24 h, 4 h and 60 min every five minutes).
- **Metrics**: 288 buckets × 5 measures per index per day, about 216 KB per index per day of metric-index licensing; the backfill writes the same series over the requested depth (the wizard shows the estimate).
- **ML**: one model per index by default, trained and monitored on the standard schedules; the VOL scope (priorities, filter expression) limits the models where needed.
- **State**: the rolling values come from the buckets TrackMe holds, not from `mstats` over its own series — a blocked metrics pipeline never reads as a drop.

VOL against the earlier recipes

	VOL (2.4.18)	Flex — license usage templates	splk-dsm volume models
Granularity	One entity per index	One entity per index	One entity per index × sourcetype (× break-by)
Source	License usage log, incremental 5-minute buckets	License usage log, three rolling scans every 5 minutes	Event counts of each feed
Setup	One tenant, a few clicks, no template	A Flex tenant + the templates per deployment type	Part of the feeds tenant; exclusions to design
History	Backfilled at creation	Accumulates from the deployment	Accumulates from the discovery
Direction per entity	Drops / spikes / both, one click	Model bounds, per model	Model bounds, per model
Silent index	Inactivity policy (by day and hour)	Not covered	Feed inactivity (delay)
Analytics	Daily volume, trend, projection, Global license usage: scopes, working days, days over quota, costs and chargeback	The template's KPIs	Event counts
Edition	Foundation, Enterprise, Unlimited	Enterprise, Unlimited	All

The Flex templates remain in the library for custom needs — a different grain, custom KPIs, a source other than the license usage log — and the splk-dsm models remain the right tool for per-feed event-count behaviour: a sourcetype dropping inside an index whose total volume holds. The two are complementary to VOL, not replaced by it.

See also

- *VOL — Volume Outliers* and *Volume Outliers — in depth* — the product guide.
- *Use TrackMe to detect abnormal events count drop in Splunk feeds* — the earlier recipes, per feed and with Flex.
- *Analyse log messages logging level to detect behaviour anomalies using TrackMe's Flex Object and Machine Learning Anomaly Detection* — behavioural detection on log levels.
- *Outlier detection* and *Outlier detection — in depth* — the outliers engine.
- *AI ML Advisor* — the AI ML Advisor, which reviews VOL models like any other.

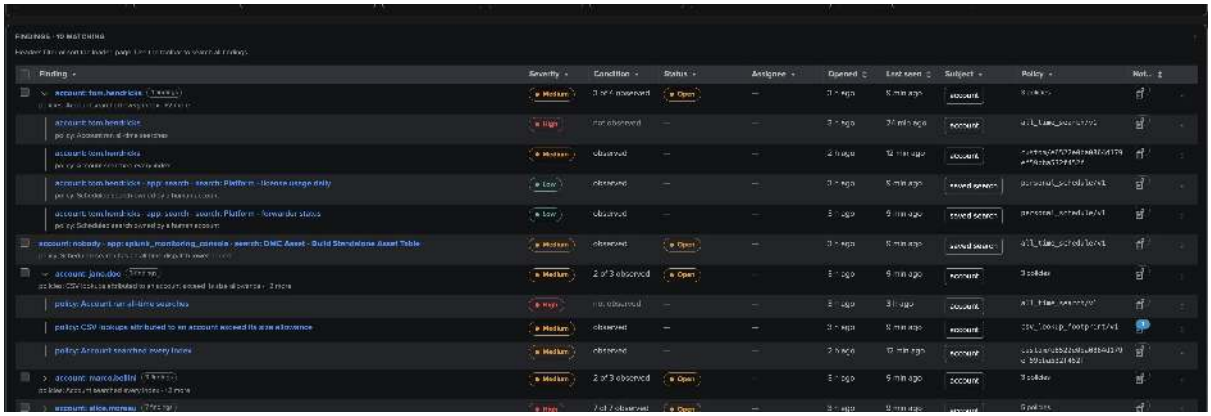
9.4.2 Who is doing what on your Splunk? User Activity Monitoring (UAM)

About this white paper

- This white paper presents **User Activity Monitoring (UAM)**, one of the seven dedicated tenant types of TrackMe, **generally available in 2.4.19**: why it exists, how it is designed, how it turns the audit trail, the search resource records and the knowledge objects of a Splunk platform into **findings**, and how to adopt it.
- It is not a procedure. The tasks — creating the tenant, reading the workspace, triaging a finding — are in the product guide, and the 36-slide hands-on deck walks through the same story screen by screen. This paper links to them where they apply.
- The screenshots come from a lab tenant monitoring two search head tiers of one indexing layer. The

accounts that appear in them — `alice.moreau`, `jane.doe`, `ben.okafor`, `tom.hendricks` and others — are fictitious.

- UAM requires a licence beyond the Foundation edition: **Enterprise**, **Unlimited**, the Unlimited trial or the **Developer** licence.
- Product guide references: *User Activity Monitoring* (the task guide), *User Activity Monitoring — reference*, *Findings, Finding groups and findings actions* and the hands-on deck *User Activity Monitoring*.



The problem: nobody sees who does what

TrackMe has always watched the data. A Splunk platform is also the people and the automation that use it, and after a few years nobody can say precisely who does what on it:

- **Personal schedules.** An analyst creates a scheduled report to feed a dashboard, then changes team or leaves. The report keeps running under an account nobody owns any more — or stops the day the account is disabled, and a dashboard silently goes dark.
- **All-time searches.** A search with no lower time bound reads every bucket of an index. Scheduled, it is a recurring cost; ad hoc, it is the `index=* earliest=0` that pins a search head for an hour. Nobody sees it until the platform slows down.
- **Knowledge objects that grow.** A CSV lookup rewritten every hour by a scheduled search reaches hundreds of MiB, is replicated with every knowledge bundle, and nobody knows which account writes it.
- **Cost without a name.** One account, a person or an integration, takes a large share of the search CPU with a handful of searches. The resource records know, per process and per node; they sit in `_introspection` for a couple of weeks and age out.

Splunk keeps the evidence of all of this — the **audit trail**, the **search resource records** and the **knowledge objects** exist on every deployment, Enterprise or Cloud — but reading it is a project: the formats differ between versions, an audit line is not a search, a search is not a person, and a naive search over `_audit` is itself a load on the platform. The requirement behind UAM was simple to state: **read those sources once, cheaply, and turn them into decisions an administrator can take** — with the result kept as evidence in the indexes and as a work queue in TrackMe, not as a dashboard someone has to remember to open.

Design principles

Six decisions shape the tenant type.

A dedicated tenant, whose object is the finding. UAM is not an eighth entity component. A tenant is either an entity tenant (DSM, DHM, MHM, FLX, FQM, WLK, VOL) or a UAM tenant, never both. There is no decision maker, no impact score and no state colour: a healthy account has no finding at all, and the work queue only contains what needs a look. UAM shares this **findings model** — policies, findings, Finding groups, triage, assignment, notifications, findings actions, the AI Agent, Collection health — with the other dedicated tenant types: Splunk HEC, Splunk S2S, Splunk UF Fleet, Splunk Forwarder Collectors, Cribl Stream and Cribl Edge (*Findings, Finding groups and findings actions*).

Read what Splunk already keeps. No agent, no forwarder change, no third-party app: the `_audit` trail, the `_introspection` resource records and the search completion records on the indexing layer, and the Splunk REST API of every search head tier.

Bounded collection, visible coverage. Every read is incremental and bounded: the audit trail from a durable checkpoint, never all time; the catalogs one dataset at a time; the history in fixed buckets. Gaps and partial coverage are recorded in receipts and shown in *Collection health*, never hidden. A run that could not cover everything may add findings; it never clears one.

Honest claims. An account is the principal **reported by Splunk**, not a verified person: shared credentials or an integration running as a person's account look like that person. Distinct search ids are a launch-count proxy. There is no machine learning: the thresholds are explicit, and the daily CPU baseline is a percentile.

Observe, never act on the platform. UAM never edits, disables, dispatches or deletes what it observes. A remediation is a suggestion; a decision on a finding stays in TrackMe.

Talk to the people concerned. A finding about an account is, in the end, a conversation with the person or the team behind it. UAM can email that account directly, next to the operators who triage.

How it works

One indexing layer, one to ten search head tiers

A UAM tenant monitors **one indexing layer** seen through **one to ten search head tiers**: the local search head TrackMe runs on and / or remote Splunk deployments reached through TrackMe remote accounts. The two kinds of data live in different places, and UAM reads each where it lives:

Data	Lives on	Read
Saved searches, users, roles, CSV lookup files and the other knowledge objects	each search head tier	**once per tier**, from the Splunk REST API
`_audit` activity, `_introspection` resource records, search completion records	the indexing layer, where every tier forwards its internal indexes	**once per tenant**, through one tier called the **route account**

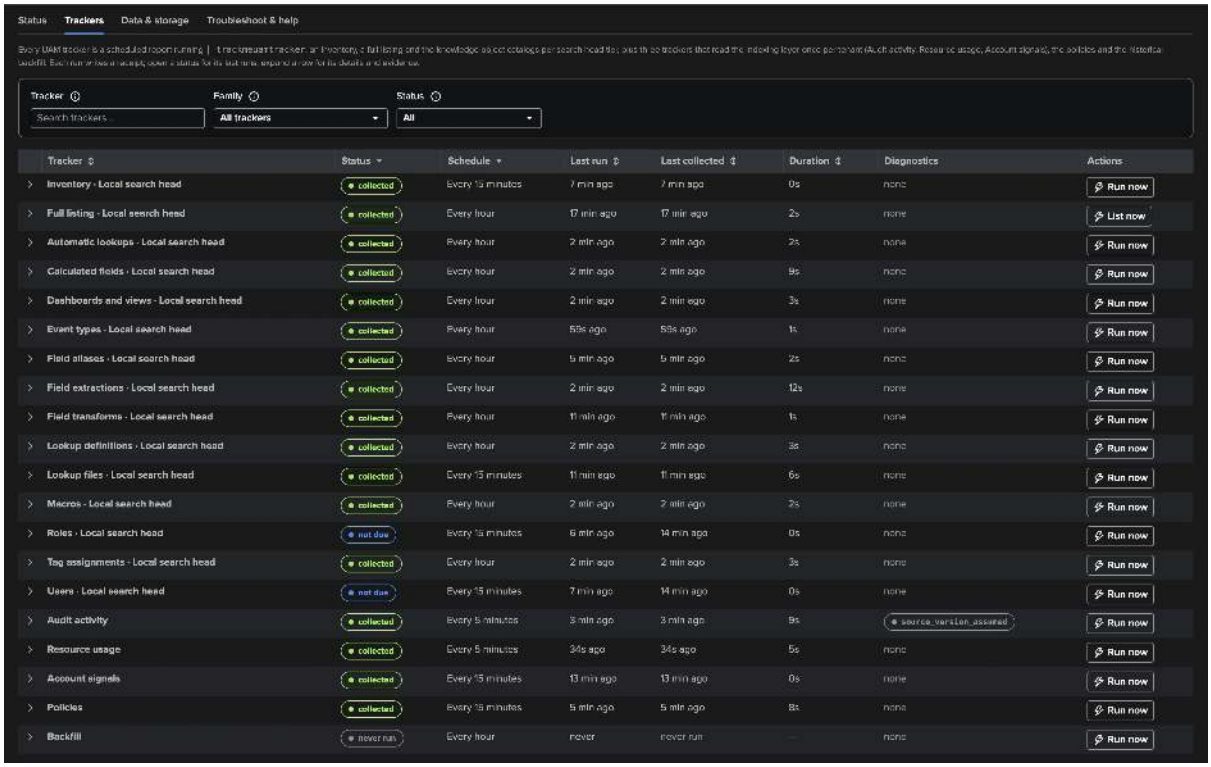
A typical Splunk Cloud shape is the ad hoc search head (the local tier, also the route) plus a remote account for a premium-app search head; on Splunk Enterprise, any number of remote accounts over the same indexers. Keeping every tier on the same indexing layer is the administrator's responsibility: two independent deployments make two tenants. Optional hostname patterns (*Activity scope*) leave out a search head that shares the indexers, such as a test instance. The details are in *Search head tiers and the route account*.

Knowledge-object catalogs from the REST API

The inventory is read from the **Splunk REST API only**, one scheduled traversal per dataset and per tier. Ten knowledge-object families sit next to the saved searches, the users and the roles: dashboards and views, macros, event types, field extractions, field transforms, field aliases, calculated fields, lookup definitions, automatic lookups and tags, plus the **CSV lookup files** — their size, owner and modification time, never their contents.

- Saved searches and the other families are collected on the catalog cadence (daily, in an off-peak window by default), users and roles every hour, CSV lookup files every 15 minutes.
- A traversal **publishes a complete generation or nothing**: a failed or partial one keeps the last good catalog, and only a complete one can mark an object removed. Every change of a saved search, user, role or lookup file — added, changed, removed, reappeared — is indexed.
- The definitions (saved-search SPL, macro bodies, dashboard source, field expressions…) are kept for administrators, from an explicit attribute list — never a credential, never the contents of a CSV file.
- The catalogs reflect what the collection account can see: the coverage of private objects depends on its permissions.

Each family surfaces where it is useful: the owner's catalogs in the account view, **Search knowledge objects** across every owner, and *Collection health → Trackers*.



Activity and resources through the route account

Three trackers read the indexing layer, each with its own schedule, time budget and health. Every five minutes, **Audit activity** reads the `_audit` trail from its durable ingestion-time checkpoint and writes **one compact record per search and window**: the account Splunk reports, the app, the search id and type, the outcome, the run time, the events and results, and whether the search covered all time. REST job access is summarised per account rather than written line by line. Two minutes later, **Resource usage** collects the resource usage of every search — CPU time, the part spent on the indexers, peak memory, elapsed time — and refreshes the environment's **search capacity** at most hourly. Every fifteen minutes, **Account signals** scans the same audit trail for the Splunk Web logins and the automated searches behind the account detection.

Three choices make the collection hold on today's platforms:

- **Time is the only limit of a run, and nothing is dropped.** A busy window is split down to a second and read whole; a range a run cannot finish continues at the next run; a range whose search fails is held and retried, never skipped; a window read while some indexers did not answer is read again once they all answer; and closed ranges are recounted about ten minutes later to catch lines indexed late. Nothing is written twice. The **Activity scope** is part of the base searches, so the indexers return only the events of the search heads in scope.
- **The audit format is the contract, not the Splunk version.** A line that parses as the classic `key=value` audit grammar or as the structured JSON record is evidence. Splunk Enterprise 9.4, 10.4 and 10.6 and Splunk Cloud 10.5 are qualified; any other version is read all the same and only adds a soft diagnostic.
- **CPU follows Splunk's own accounting where it exists.** Splunk Cloud 10.5 accounts a search's CPU at completion, on the search head and on each indexer, and runs short searches in pooled processes that the introspection samples barely catch. UAM decides per window and per tier from the data: a tier with completion figures counts its CPU once, at completion; a tier without them keeps the CPU integrated from the introspection samples. Every resource record says which source it used, and peak memory reads *not measured* rather than zero when no sample caught a search.

Every search UAM runs goes through Splunk's **Search API v2**, the API Splunk keeps as the v1 Search API is retired.

An optional **historical backfill** (none by default, up to 90 days) collects the audit and resource history that existed before the tenant, newest first, in 5-minute buckets, and stops at the retention of the sources — `_introspection`

is often kept for less time than `_audit`. It never collects a span twice. Its main value is the classification: the people and services already active are recognised on the first day instead of a week later.

Evidence in indexes, state in the KV store

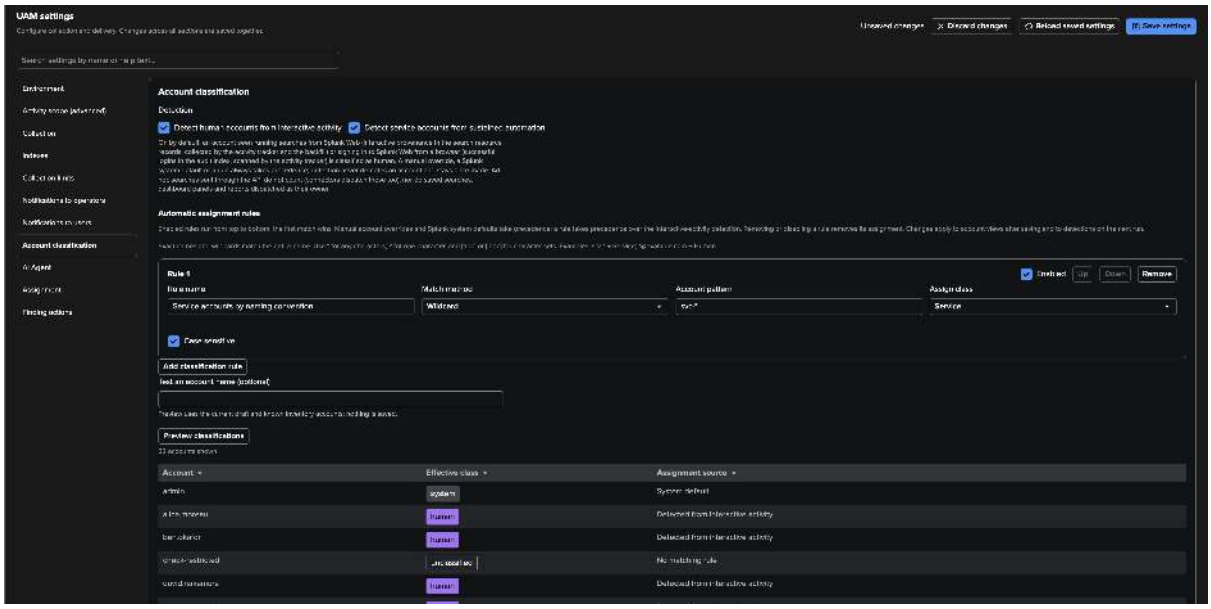
The evidence is written as `trackme:uam:*` events into the tenant's summary index (audit records, resource records, inventory changes, receipts) and its notable index (finding transitions), each family re-targetable to another index. Indexed dimensions let the per-account and timeline views run through `tstats`, in proportion to the accounts rather than the events. The state — configuration, catalogs, findings, Finding groups — lives in per-tenant KV collections. All collection runs in scheduled searches, never in a REST handler. The evidence counts against the Splunk licence like any TrackMe summary data; it is compact by design, and the full shapes are in *The evidence*.

Accounts: who is a person, who is a service

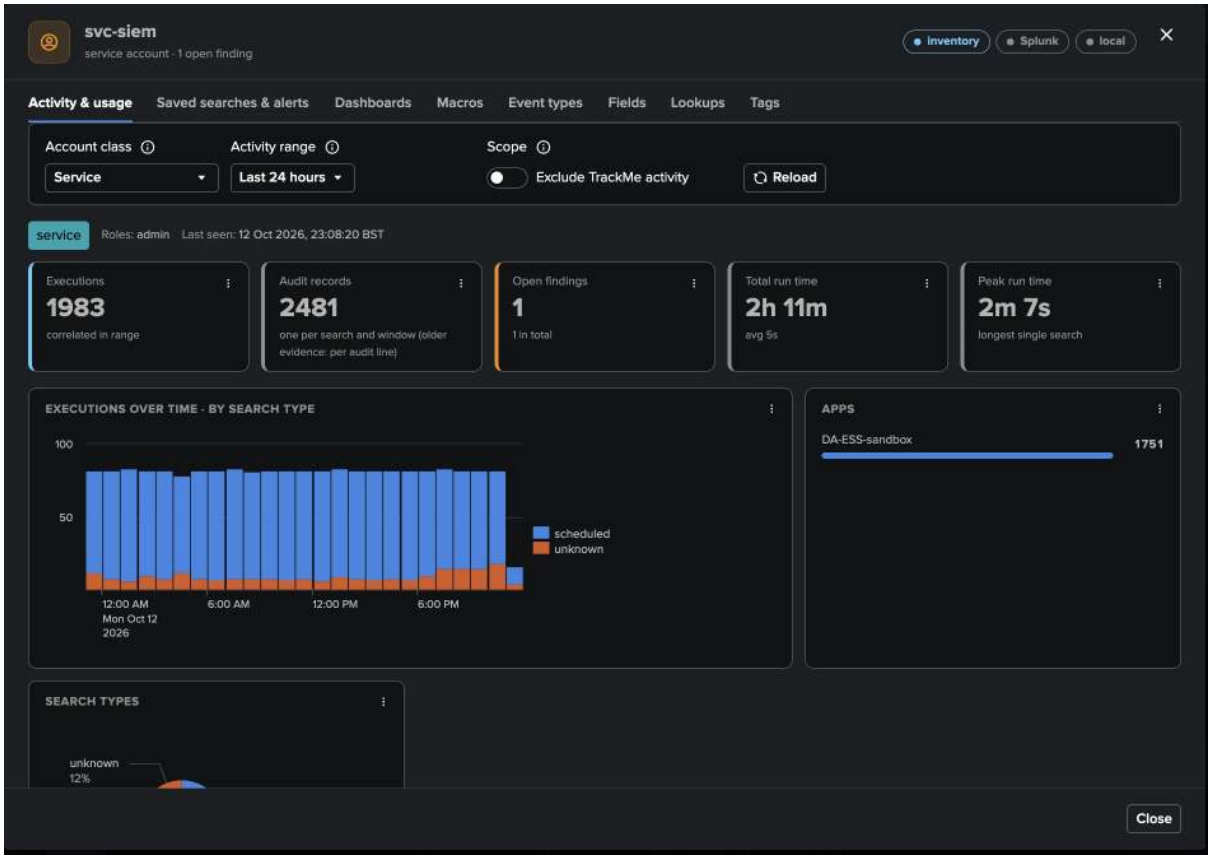
A finding about an account only makes sense once you know what the account **is**. UAM keeps three classes, and the class decides which policies apply:

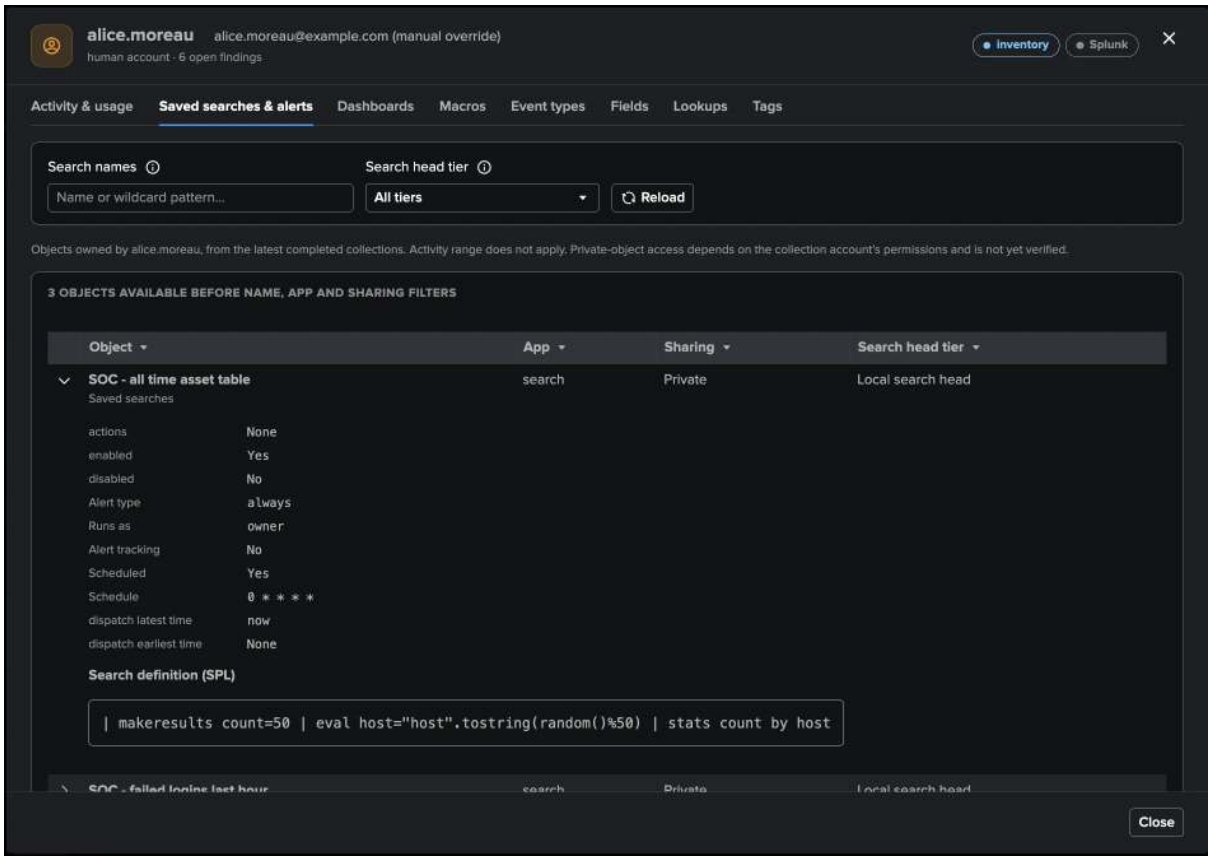
- **human** — detected from interactive activity: a search run from Splunk Web, or a successful browser login. One signal is enough, and the class stays.
- **service** — detected after **seven distinct days** of automation-only activity (API jobs, the scheduler, acceleration) and nothing a person does; withdrawn as soon as the account does what a person does. This detection needs the classic `key=value` audit format.
- **system** — `nobody`, `splunk-system-user` and the built-in `admin`, preset; never evaluated by the account policies and never emailed.

On top of the detection, ordered **classification rules** (exact names or wildcards such as `svc-*`) and **manual classes** settle the rest. Precedence is always the same: manual class, system defaults, the first matching rule, human detection, service detection. An account nothing classifies stays **unclassified** and is left out of the policies scoped to human accounts.



The **account view** brings everything known about one account together: executions and run time by search type, CPU over time and the daily CPU baseline, the most expensive and most recent searches with **Fetch SPL**, its findings — and, since 2.4.19, the knowledge objects it owns on each tier, one tab per family. **Fetch SPL** reads the text of a search live from `_audit`, as the person asking, and stores nothing; indexing search text is an opt-in (*Store SPL search*, off by default), to be pointed at a restricted index first since search text can carry literals and secrets.





Privileged users answers a question every audit asks first: which accounts hold high privileges, and why? An account is privileged when one of its roles — directly or through the roles it imports — matches a privileged role pattern (`*admin*` by default, live, so a role created tomorrow counts as soon as its name matches) or grants a privileged ca-

pability (`admin_all_objects`, `list_storage_passwords`, `edit_user` and the other well-known ones). Each reason is written in words, with the tier that grants it — `team_lead imports admin (pattern *admin*)`, `secops grants list_storage_passwords` — next to the account's class and its activity of the last 24 hours. The rules are the administrator's: add or exclude roles by name, choose the capabilities. It is computed from the users and roles catalogs UAM already collects, without any further call to Splunk.

Policies: what deserves a look

A policy reads the catalogs or the evidence and raises **one finding per policy and subject** — a saved search, an account, or a role — with its evidence, a severity and a suggested remediation.

Sixteen built-in policies in four families

Family	Policies (on by default in **bold**)
Inventory — read from the catalogs, judged per tier	**Scheduled search owned by a human account** (low); Scheduled search owner is not classified; Scheduled search exceeds the hourly schedule limit; **Scheduled search has an all-time dispatch lower bound**; **Scheduled search uses a real-time dispatch range**; **CSV lookups attributed to an account exceed its size allowance**.
Activity — read from the audit records, 15 minutes at a time	Account launched an unusual number of searches; Account ran repeated long searches; **Account ran all-time searches** (high).
Resource — read from the resource records	**Account's search resource consumption exceeded its allowance**; **Account ran repeated resource-intensive searches**; Account's CPU consumption is unusual for its daily baseline (shipped in preview mode).
Privileged — read from the privileged assessment, inheritance included	Account became privileged (high, once per change); Role gained a privileged capability (high, a finding about the role whose own definition changed); Privileged account is dormant; Privileged accounts differ from the reference (names and lookup tables you maintain).

Seven are enabled on a new tenant; the privileged policies wait until you turn them on. Every policy takes exact owner and app exclusions — the `trackme` and `Splunk_SA_CIM` apps are excluded by default, so the monitoring and the data-model accelerations do not flag themselves — and the account policies an account scope (human accounts, classified accounts or all). The full catalogue with every threshold is in *The built-in policy catalogue*.

Three design points deserve attention:

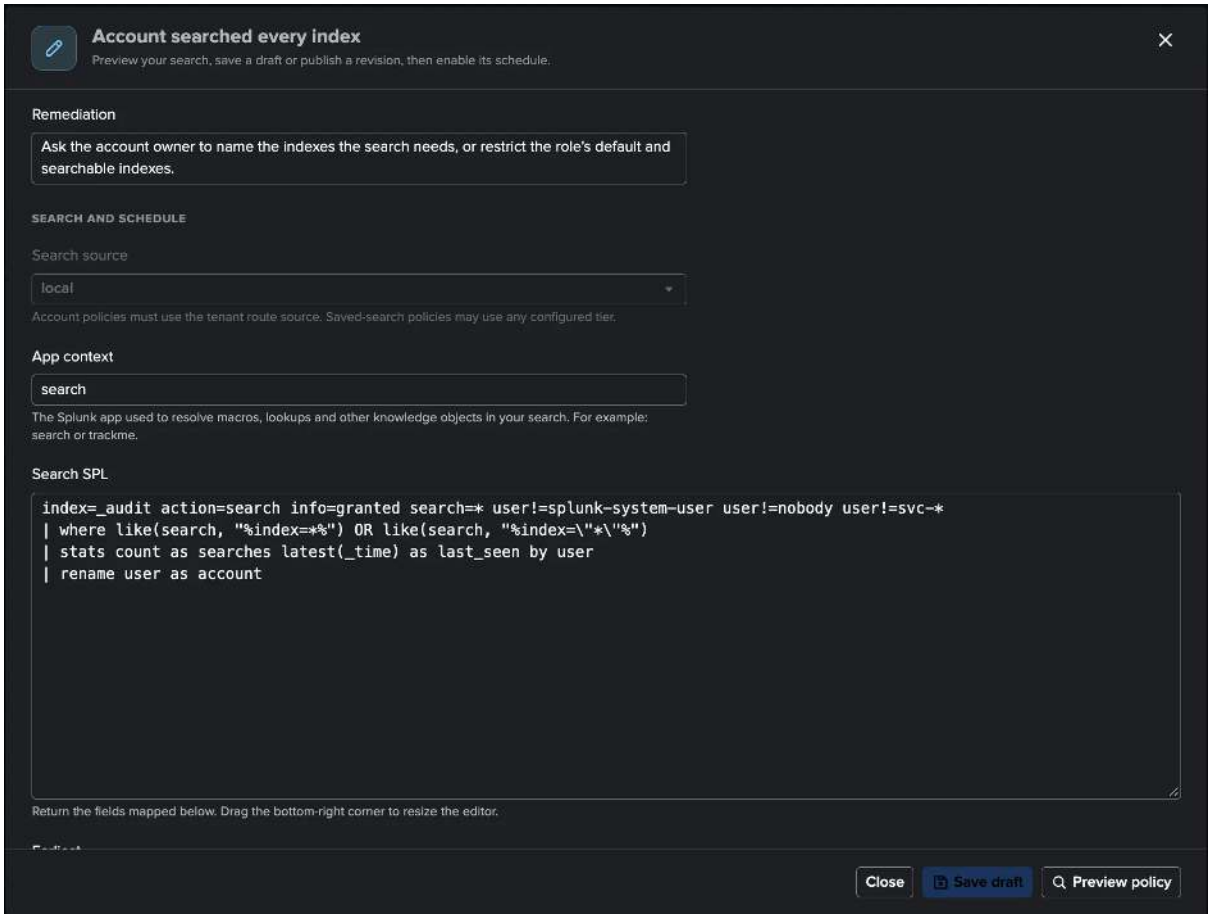
- **Budgets derived from the capacity.** The resource policies carry no absolute figure to maintain: with automatic thresholds on, an account's allowance is 10 % of the measured search CPU capacity per window, and a heavy search reaches 2 % of it or 10 % of the smallest search host's memory. An account above 40 % of the environment's search CPU has its finding raised one severity level — the share is evidence and a boost, never a finding on its own. The resource policies stay *learning* until the tenant holds 24 hours of resource history and a measured capacity.
- **The CSV lookup footprint, attributed.** On each tier, one CSV lookup over 100 MiB, or the CSV lookups of one account over 500 MiB in total, raise a finding. A file is attributed to its **probable writer** — the account the audit evidence shows writing it — and otherwise to its recorded owner; the evidence keeps the basis of the attribution and both identities. It is an estimate since monitoring began, not proof of authorship, and only the metadata is read.
- **Explained evaluations.** Each policy's last evaluation opens a **policy insight**: what the run saw, what was matched, excluded, not applicable or could not be judged, and why, with a seven-day history. *Preview policy* shows what a draft would raise before you save.

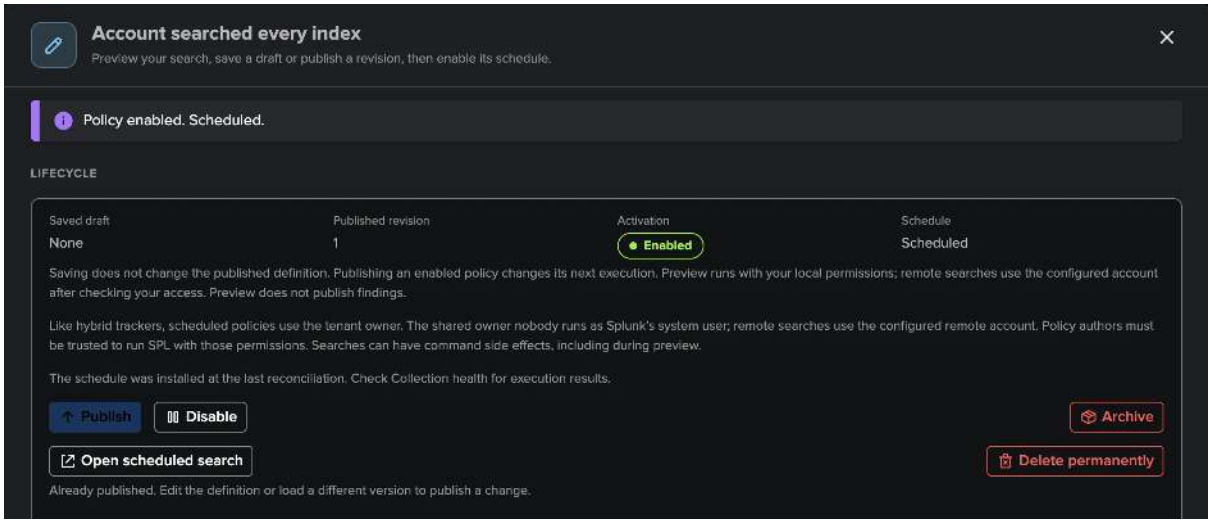
Custom policies in SPL

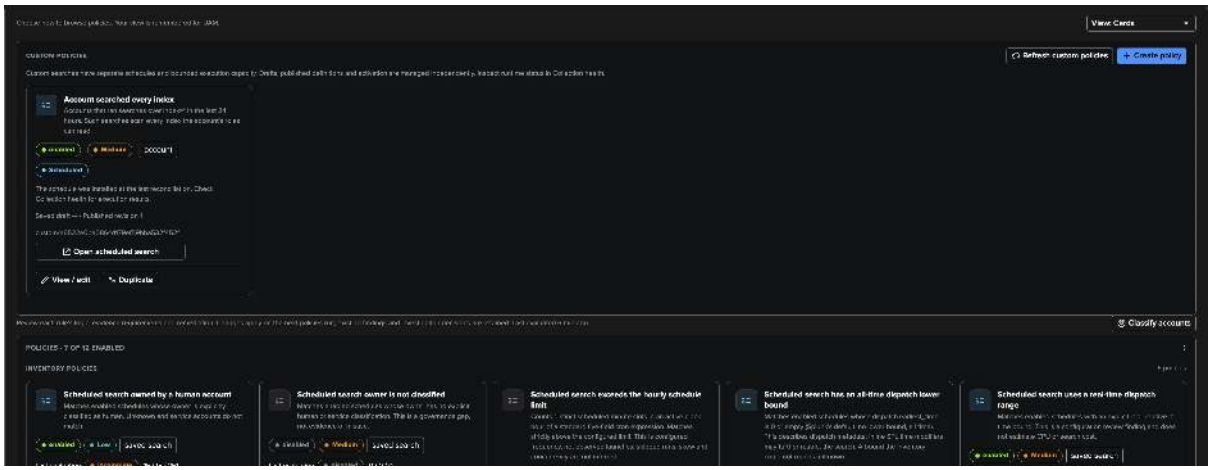
When the catalogue does not cover a rule of your own, tenant administrators write it in SPL. A custom policy is a search, a relative window ending now (at most 31 days), a cron, and a mapping of its results to findings about an **account** or a **saved search** — the subject that places them in the right Finding group. Its lifecycle is deliberate:

- **Preview** runs the current draft as its author and shows up to 20 candidate findings.
- **Publish** freezes an immutable revision; later draft edits never change what runs.
- **Enable** schedules a search for the published revision, owned by the tenant owner.
- A version history loads, compares and restores versions; *Collection health* reports the execution health of every enabled policy, and the Configuration Guardian warns when one drifts from its schedule or keeps failing.

A tenant holds up to 100 custom policies, 25 of them enabled. Custom findings appear with the built-in ones, join the same Finding groups and are triaged the same way. Because the SPL runs on a schedule as the tenant owner — and a preview as its author — only trusted tenant administrators author policies. See *Custom policies*.







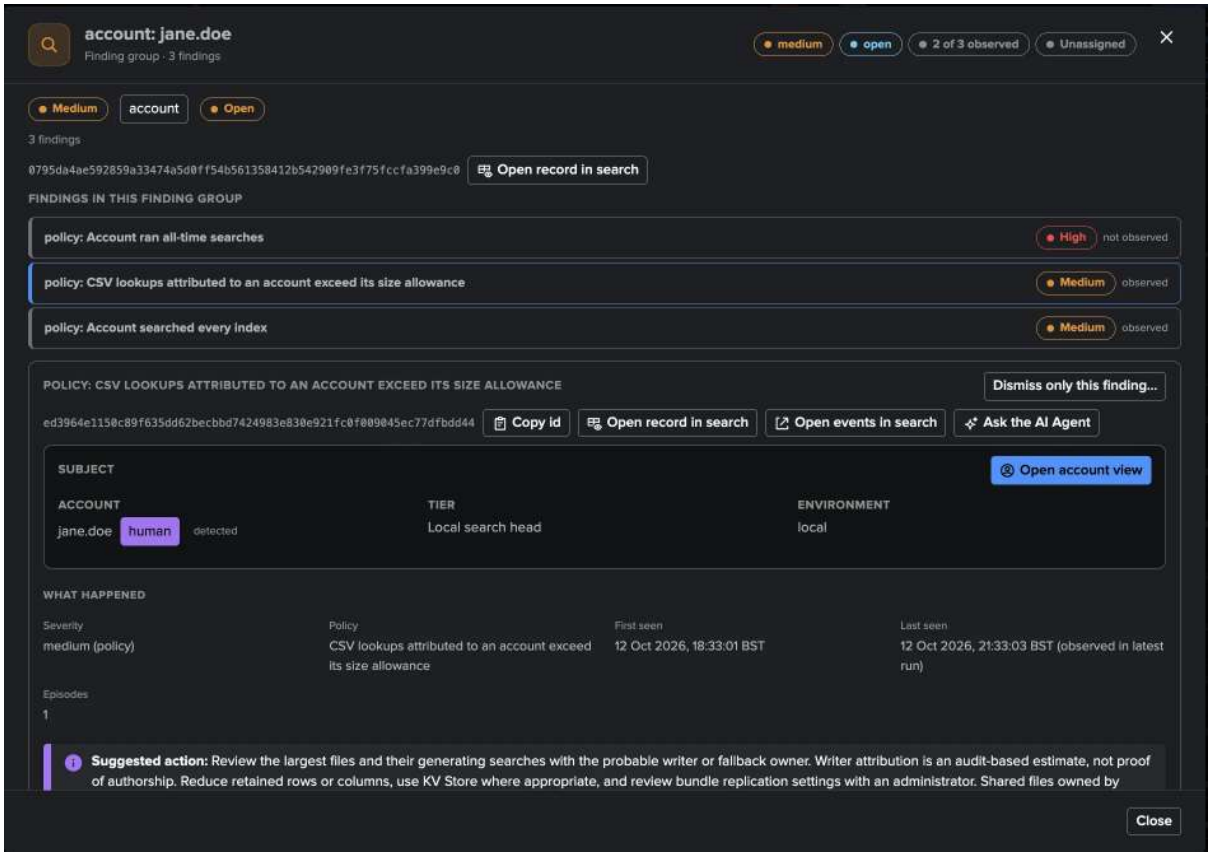
Finding groups: triage at the level of the account

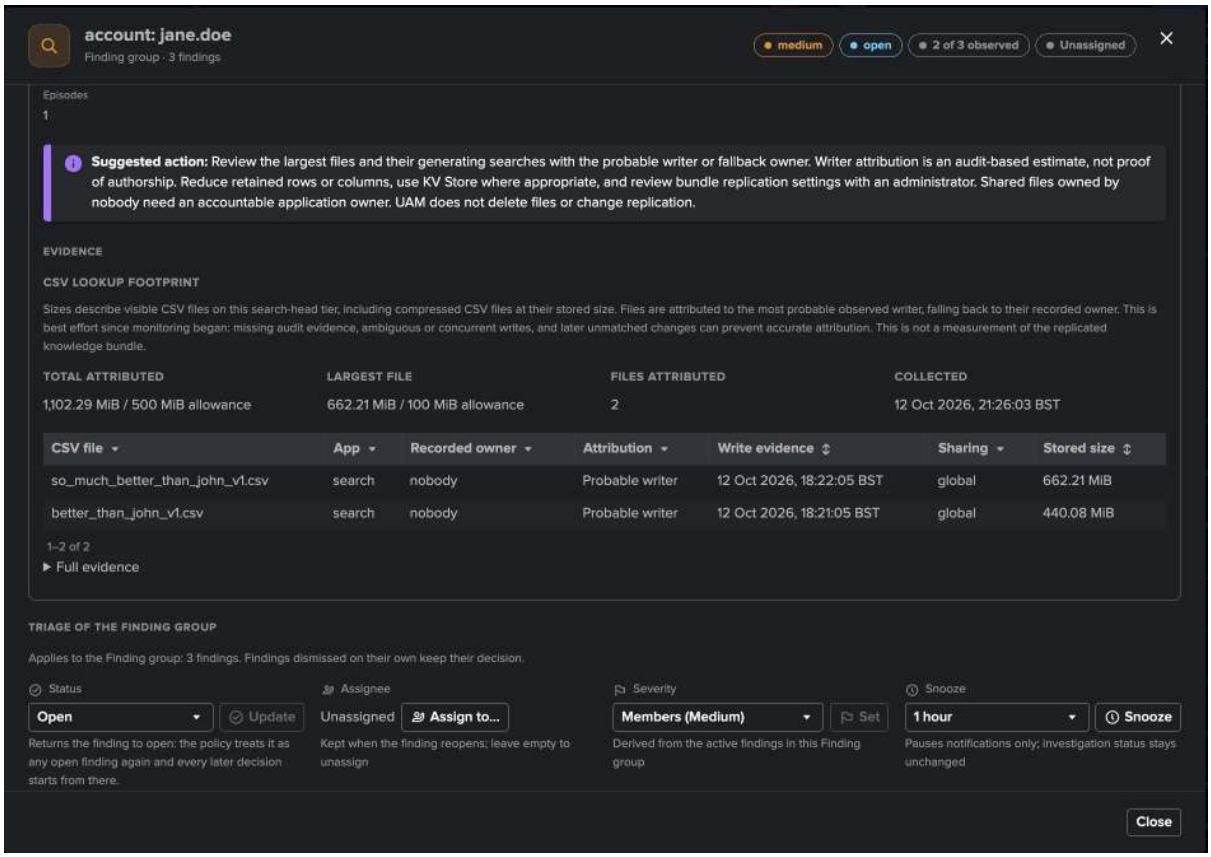
A UAM finding has two independent axes. The first is **whether the policy still sees the condition**: *observed*, or *not observed* after a complete evaluation without it. UAM never resolves a finding for you — a behaviour that disappears from one run still deserves a human decision. The second is **your decision**: open, investigating, resolved or dismissed (permanently or for 1, 7, 30 or 90 days). A resolved finding observed again reopens; an ended dismissal returns the finding to open.

The decisions are taken on **Finding groups**. All the findings about **one account and the saved searches it owns** form one group: the all-time searches and CSV footprint of `jane.doe` together with the custom policy that caught the account's `index=*` searches; the personal schedules of `tom.hendricks` with that account's own activity findings. Ownership defines the group, not classification, and every tier of the indexing layer shares it. The reasoning is practical: the person you will talk to is the owner, and the conversation covers everything at once.

- A decision on the group applies to every member, except a member **dismissed on its own** — a personal schedule you accept inside a group that also holds a real problem.
- A new kind of finding on a dismissed group reopens the group: a dismissal never silences a new problem.
- Operators can also group by hand: **merge** 2 to 25 groups into one manual group, add to or remove from it, or **ungroup** — for example to handle the accounts of one project together. The findings of one object always move together.

The **Investigate** view of a group shows its findings, the subject with its class, *What happened* in plain words, the evidence of the selected finding, the suggested action and the **Triage** of the group: status, assignee, severity and notification snooze.





Decisions are recorded in the history of the finding and of its group, with the actor and a note, and protected against concurrent changes. **Assignment groups** and ordered **routing rules** assign new Finding groups to a team; **findings actions** add drilldown searches, links, webhooks through system-wide Webhook accounts and an email action to the finding menus, by hand or through per-policy automation rules. Every finding transition is also indexed as

`trackme:uam:finding`, ready for your own Splunk alerts, Enterprise Security correlations or SOAR playbooks. See *Finding groups*, *Assignment and automatic routing* and *Findings actions*.

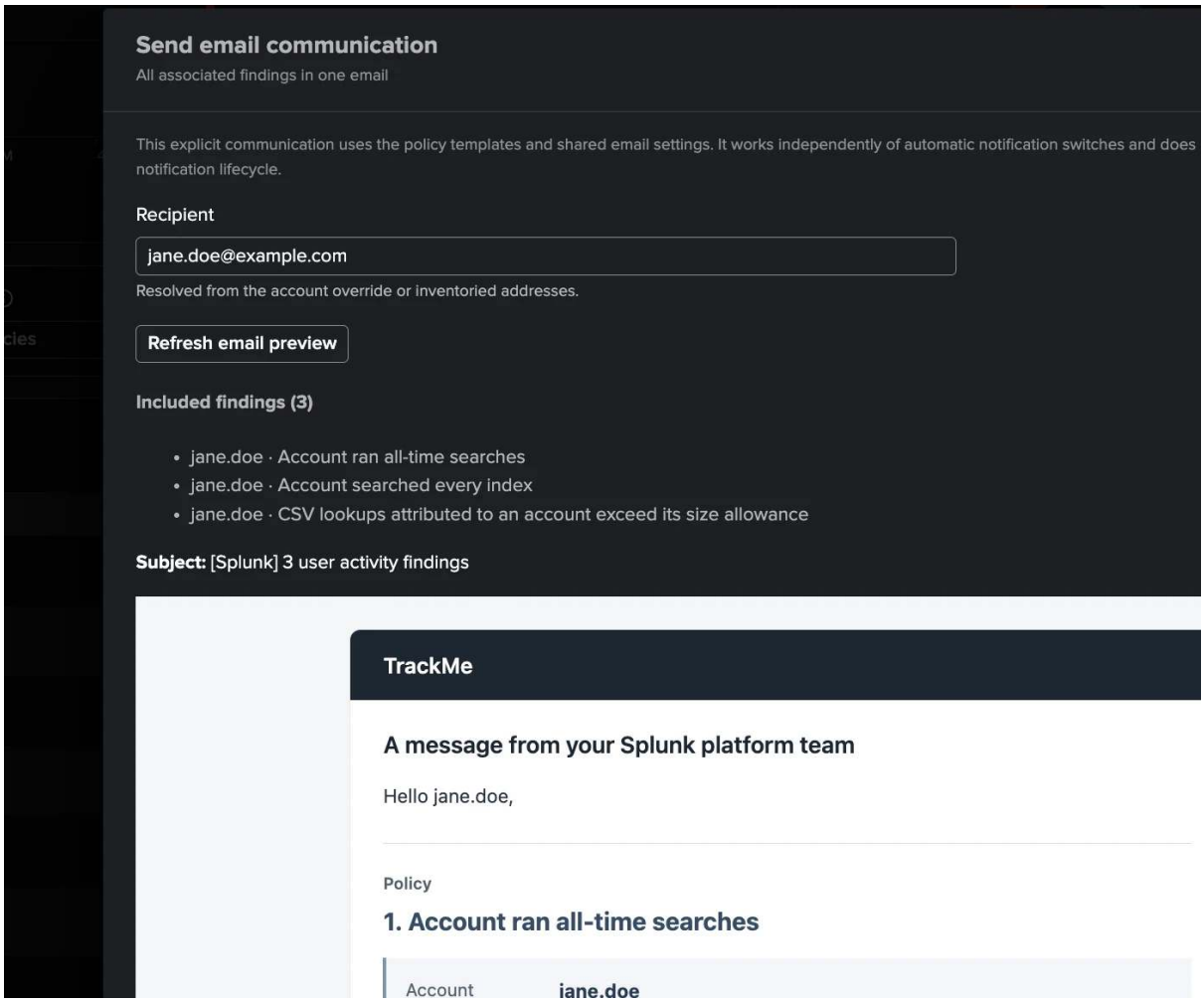
Reaching the people concerned

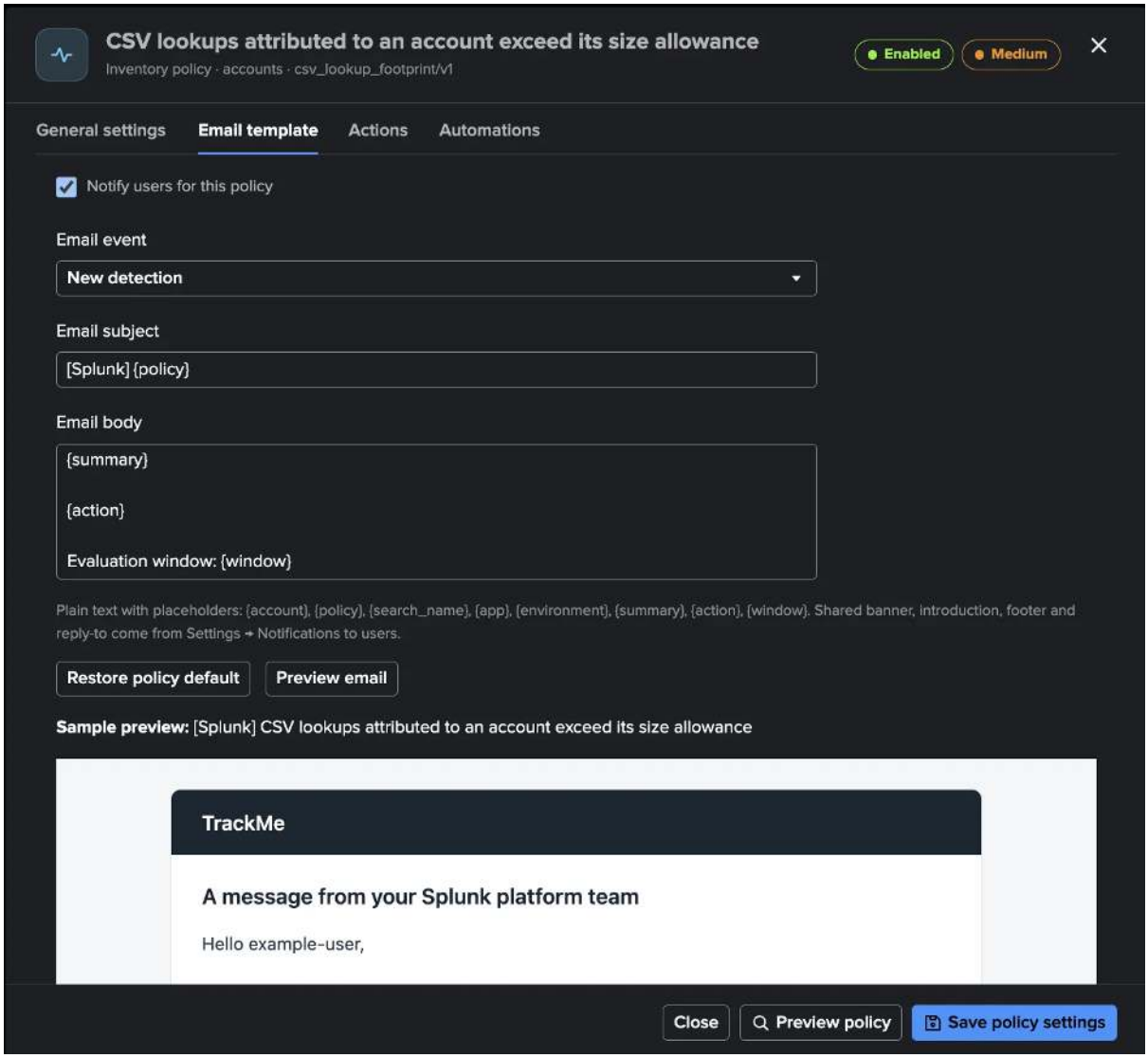
UAM writes to two audiences, through three independent channels.

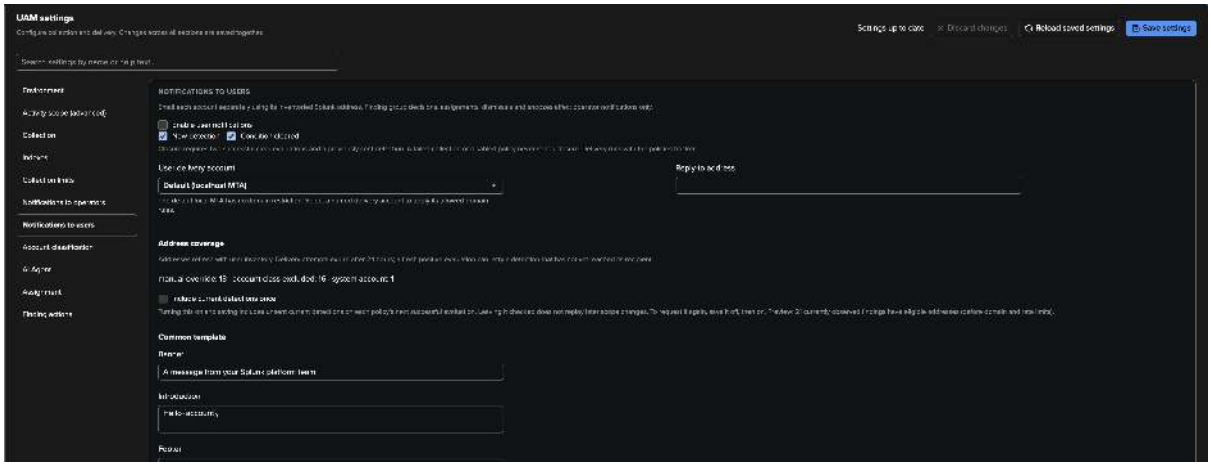
Operators — *Notifications to operators*, off by default: emails on the transitions that matter (new finding, reopened, severity raised, resolved; optionally dismissed and assigned), above a minimum severity, as a **digest** per policies run or **per finding**, threaded in the mail client. Every email links to the finding in its Investigate view. Snoozing a Finding group holds these emails without changing its status.

The account concerned — *Notifications to users*, off by default: one email to the account a finding is about, on a **new detection** and when the **condition is cleared** (after two consecutive complete evaluations without it), so that people fix their own searches. The addresses come from Splunk's user records on each tier, or from an administrator's override. System accounts never receive mail; human accounts do by default, service and unclassified accounts only on opt-in. A common banner, introduction, footer, reply-to address and support link combine with each policy's own **email template**. Turning it on records the current findings as a baseline, so nobody receives a backlog. Rate limits apply (50 emails per run, 3 per address per hour, 24 hours between repeat detections). These emails follow the policy, not the operators' triage.

A reviewed one-off email — *Send email communication*, from a finding, a member or a whole Finding group (up to 200 findings in one email): a preview, a recipient you may change for this email only, and an explicit confirmation. It is sent by a TrackMe search job, once, and never resent blindly when the outcome is uncertain.




CSV lookups attributed to an account exceed its size allowance
Inventory policy · accounts · csv_lookup_footprint/v1
Enabled
Medium
General settings
Email template
Actions
Automations
Notify users for this policy
Email event
New detection
Email subject
[Splunk] {policy}
Email body
{summary}
{action}
Evaluation window: {window}
Plain text with placeholders: {account}, {policy}, {search_name}, {app}, {environment}, {summary}, {action}, {window}. Shared banner, introduction, footer and reply-to come from Settings → Notifications to users.
Restore policy default
Preview email
Sample preview: [Splunk] CSV lookups attributed to an account exceed its size allowance
TrackMe
A message from your Splunk platform team
Hello example-user,
Close
Preview policy
Save policy settings



Important

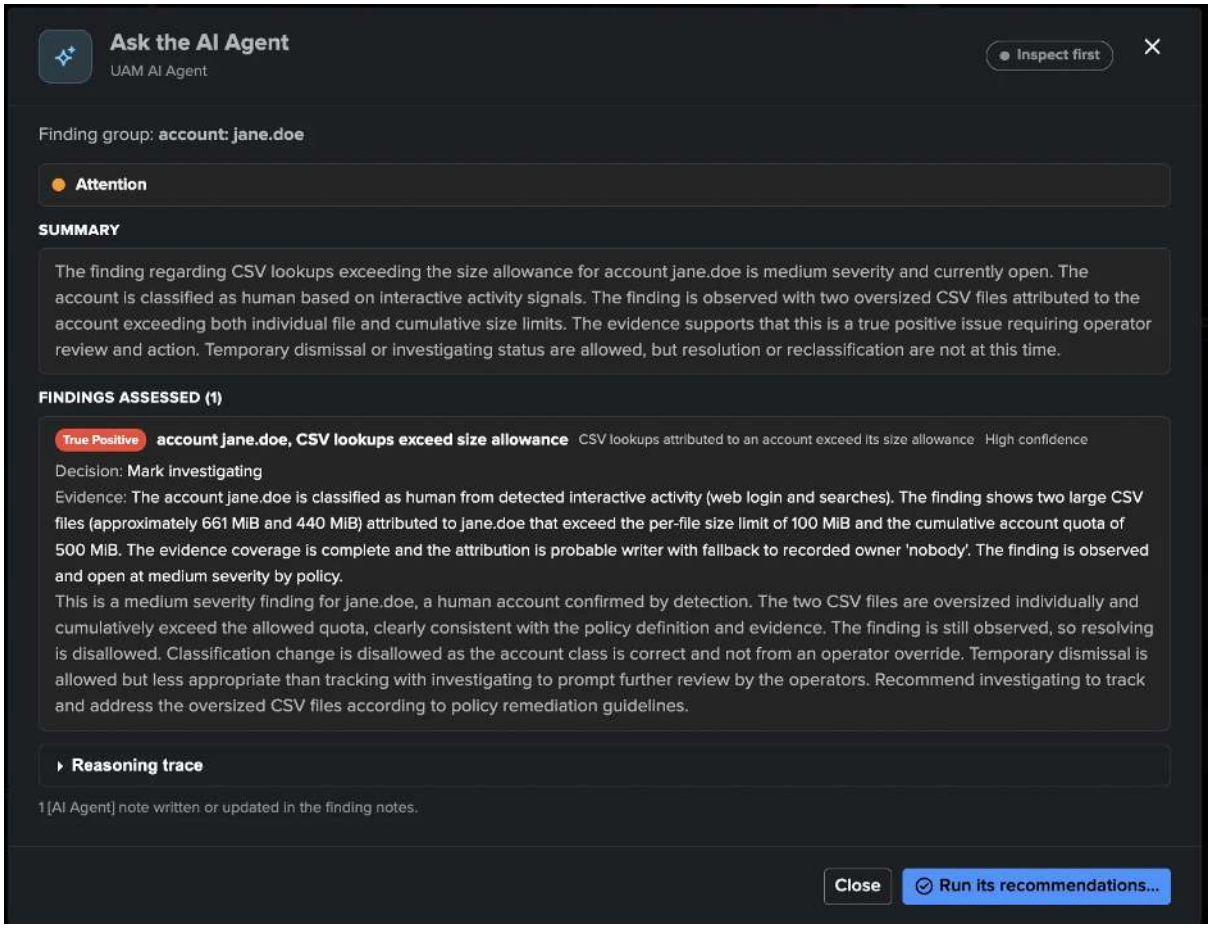
The people who receive the user emails did not ask for them. Before turning the channel on, check the address coverage, point the reply-to address and the support link at your team, and preview the email template of every enabled policy.

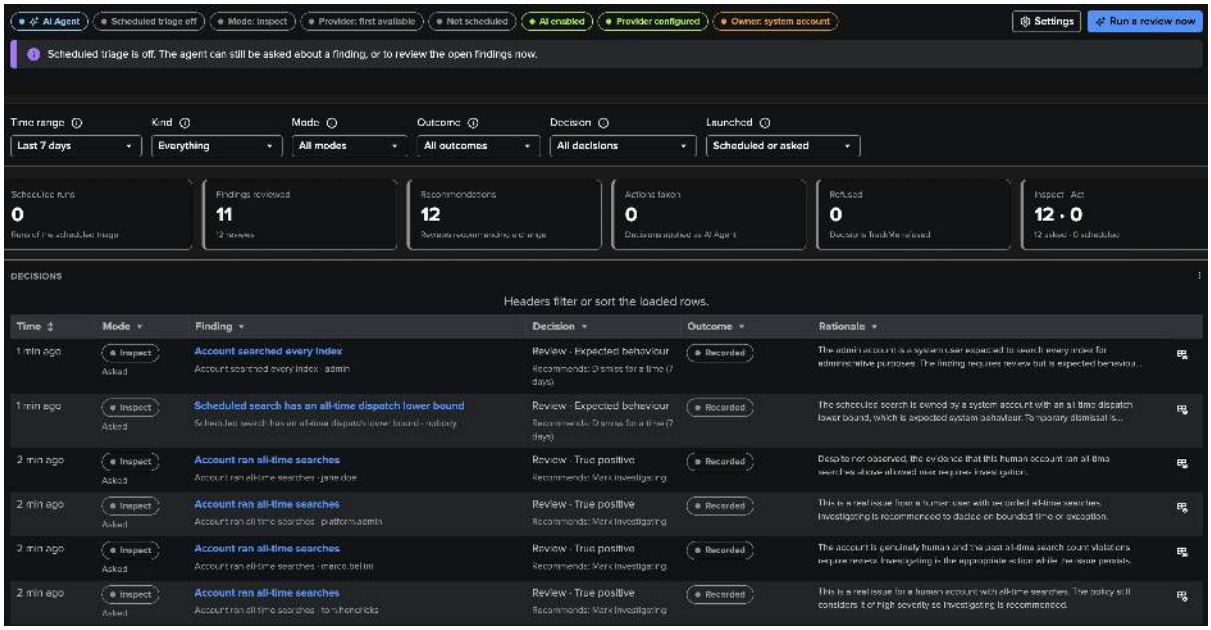
The UAM AI Agent

The **UAM AI Agent** reviews Finding groups the way an operator would, from the evidence UAM holds, and writes its assessment as one current `[AI Agent]` note per finding. It runs on the findings AI Agent framework shared by the AI Agents of every dedicated tenant type, and replaces the UAM Findings Advisor of 2.4.18.

- **Inspect** changes nothing: an assessment, a confidence and a recommendation.
- **Act** also applies the decisions the tenant allows: mark investigating and dismiss for a time (on by default, never for a critical or high finding still observed), resolve a finding its policy no longer observes and classify an unclassified account (off by default). It never dismisses permanently, assigns, or changes a severity, a policy or a setting.
- It is used three ways: **Ask the AI Agent** on one Finding group, **Run a review now**, and the **scheduled triage** — hourly, off by default, up to 10 findings per run, one per group, and by default only critical and high findings. A scheduled run never overrides a decision or an assignment made by a person.

Every decision appears under the actor *AI Agent* in the history and in the decision log of the **AI Agent** tab. The agent needs the AI Assistant to be enabled with an AI provider, and the scheduled triage a tenant owner other than `nobody` or `splunk-system-user`. See *Findings AI Agents*.





Operating and cost

Collection health is the first stop, and it answers rather than reports: **Status** says what needs attention, why, where the evidence is and what to do, with the problems of the last 24 hours and a **Flow** view of the whole collection pipeline; **Trackers** lists every report with its live schedule, last run and diagnostics, and the exact searches each run sent, ready to copy into the search bar; **Data & storage** names every index, KV collection, log and report UAM writes for the tenant; and **Troubleshoot & help** explains every diagnostic code — what it means, where it comes from, its possible causes and what to do. One run per report at a time; a run whose search provably stopped is recovered automatically after five minutes, otherwise **Recover stopped run** releases it. **Collect now** runs the whole chain on demand. The Configuration Guardian check `uam_collection_health` warns when collection is not running as configured — a missing or unscheduled report, missed schedules, repeated failures, a checkpoint that does not move, a range held, windows waiting to be read again, an overdue listing, failing email delivery, a custom policy that drifts — and clears itself once the condition is gone.



What the tenant costs:

- **Collection** — searches bounded by time: every run budget is its tracker's interval less 30 seconds, and a run dispatched late ends before the next scheduled one, so no run is skipped; the audit trail five minutes at a time from its checkpoint; one catalog traversal per dataset and tier, mostly daily and off-peak; the backfill under an hour per hourly slot. What a run cannot finish continues at the next run. The **Activity scope** is the main lever on a large indexing layer shared by several search head clusters. Several tenants watching the same tier each pay their own collection cost.
- **Evidence** — one compact record per search and window, one resource record per search and window, the inventory changes and one receipt per run. To reduce it, switch off resource collection (the resource policies then stay learning), keep *Store SPL search* off and size the backfill to your retention.
- **Reads** — the account and timeline views run through `tstats` on indexed dimensions. These index-time

settings take effect after the splunkd restart that follows the app installation or upgrade.

Every budget is a setting with a validated range, not a product cap; the defaults fit a busy cluster. They are listed in *Settings reference*.

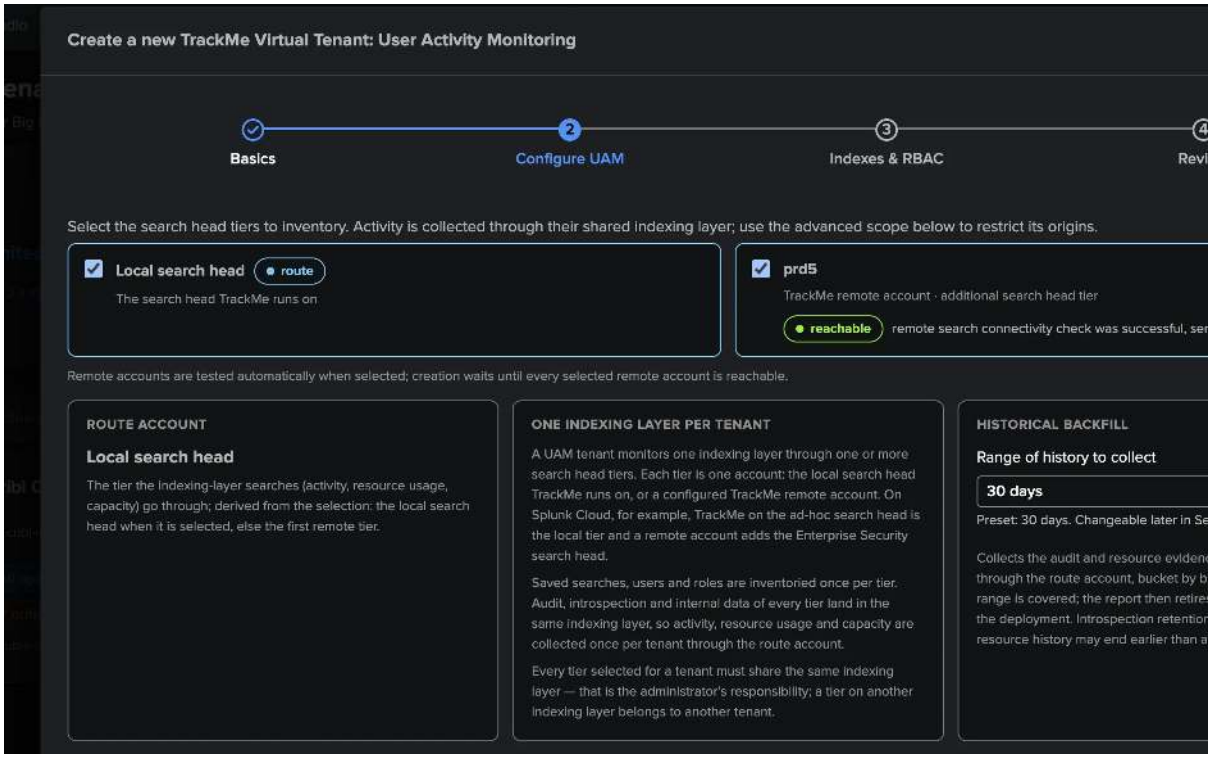
Adopting UAM

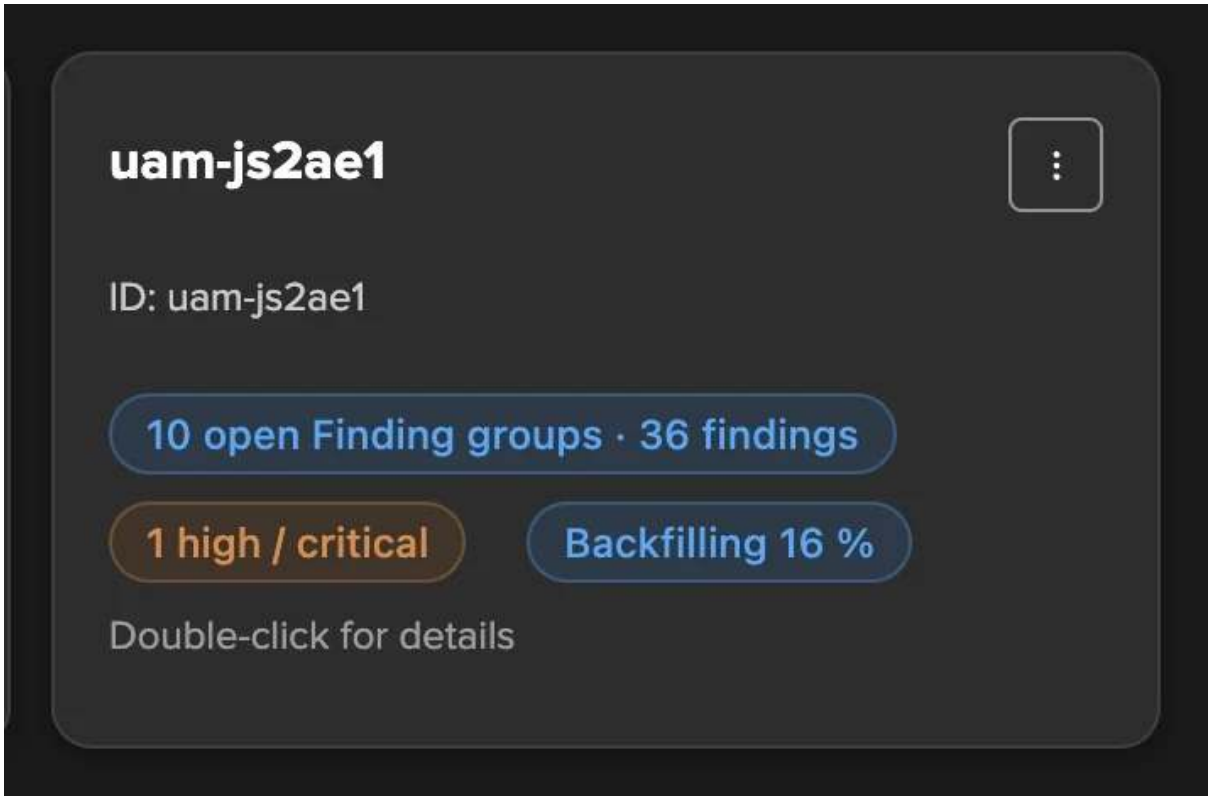
Prerequisites

- A licence beyond Foundation: Enterprise, Unlimited, the Unlimited trial or the Developer licence.
- `_audit` and `_introspection` forwarded to the indexing layer by every search head tier. A search head that keeps its internal indexes local hides its searches, logins and resource usage from the tenant.
- A TrackMe remote account per additional search head tier (*Remote Splunk deployments*).
- A dedicated **tenant owner** service account: the scheduled reports run as the owner, and the scheduled AI Agent cannot run as `nobody` or `splunk-system-user`.
- Optionally, a TrackMe email delivery account for the notifications.
- UAM evidence is audit data: the workspace is for the **TrackMe administrators of the tenant**. Power users of the tenant have read access through the REST API and can send user communications and run findings actions.

Creating the tenant

The *User Activity Monitoring* card of the tenant creation wizard asks for the tenant identity, the search head tiers (the route account is derived), the historical backfill and the optional activity scope, then the usual indexes, owner and roles. Everything is provisioned and scheduled at once; the tenant card shows its open Finding groups and findings, the high or critical ones, and the collection status. The same creation is one REST call (`POST /trackme/v2/vtenants/admin/add_tenant` with `tenant_uam_enabled` and a `uam_config` object). See *Creating a splk-uam tenant*.





A rollout path

The first hour. Check in *Collection health* (Status, then Trackers) that the first inventory, audit activity, resource usage, account signals and policies runs succeeded on every tier. Classify the monitoring itself — the tenant owner, the administrator accounts, the platform's own automation — as system or service so that it never fills the findings, and add a rule for your service-account naming convention.

The first days. Let the backfill and the detection classify the people and the services. Work through the findings of the seven default policies: the all-time searches first (high), then the all-time and real-time schedules, the CSV lookups and the resource findings once the learning period ends. Use the policy insight to understand what each run saw before changing a threshold, and exclusions for the legitimate cases — an acceleration owner, an approved investigation account.

Routine. Turn on the operator notifications at medium severity, set assignment groups and routing rules for your teams, and enable the opt-in policies whose thresholds your baselines now support. When the address coverage is good and the templates are reviewed, turn on the notifications to users — the step that makes the platform's users part of the remediation. Add custom policies for your own rules, and the scheduled AI Agent triage, in inspect mode first, for the critical and high findings.

The hands-on deck *User Activity Monitoring* follows this path screen by screen.

Upgrading from 2.4.18

- The first policies runs gather the existing findings into Finding groups; triage decisions, notes and history are kept.
- The inventory moves to the REST API catalogs: each tier gets its knowledge-object reports and its first full saved-search listing runs within the hour. The users and roles reports move from a 5-minute to a 15-minute schedule.
- The UAM Findings Advisor is replaced by the UAM AI Agent; an enabled advisor's settings are carried over once.

Limits (as of 2.4.19)

- **Identity.** An account is the principal Splunk reports, not a verified person, and UAM does not link the same person's accounts across deployments.
- **Custom policies** map to accounts and saved searches only, write their exclusions in their SPL, and their findings are not covered by the operator and user notifications, which list the built-in policies.
- **One indexing layer per tenant**, and keeping the tiers on it is the administrator's responsibility. A saved search with the same name, app and owner on two tiers shares one set of run figures.
- **Cadence.** The activity and resource policies judge the trailing 15 minutes by default; the inventory policies follow the catalog cadence, a daily saved-search listing by default. Catalogs are not atomic snapshots and depend on the collection account's permissions.
- **Service detection** needs the classic `key=value` audit format, and its seven days must lie inside the audit retention.
- **No remediation.** UAM never changes the platform; every remediation is carried out by people.

See also

- *User Activity Monitoring* — the task guide: setting up, the workspace, triage, communication and tuning.
- *User Activity Monitoring — reference* — the reference: tiers, reports and schedules, evidence, classification, the policy catalogue, every setting, REST and troubleshooting.
- *Findings, Finding groups and findings actions* — findings, Finding groups, assignment, notifications, findings actions and the findings AI Agents, shared by every dedicated tenant type.
- *User Activity Monitoring* — the hands-on deck.
- *Creating a splk-uam tenant* — the creation wizard and the REST flags.
- *Configuration Guardian* — the Configuration Guardian.
- *The AI Advisors* and *AI Agents automation* — the AI Agents and their automation.

9.4.3 Use TrackMe to detect abnormal events count drop in Splunk feeds

Detecting abnormal events count drop in Splunk feeds with TrackMe

- This TrackMe whitepaper tutorial demonstrates how you can leverage TrackMe to detect **abnormal events count drop in Splunk feeds**.
- It is a frequent use case requested by TrackMe users. The goal is to be able to easily detect when a feed in Splunk is facing a significant drop in the amount of events received.
- A massive drop in a given feed, whether you look at this from the index perspective or different factors, can be a sign of a problem in the data ingestion pipeline, or even a sign of a problem in the data source itself.
- TrackMe tackles this challenge in different ways, notably by leveraging Machine Learning driven anomaly detection.
- In this tutorial, we will cover the different options available to licensed and unlicensed TrackMe users.
- This tutorial demonstrates use cases around Machine Learning Outliers detection. For deeper insights, refer to the following documentation: *Outlier detection*.
- In this tutorial, we will review the out-of-the-box features of the **splk-dsm** component as well as an additional use case for licensed customers leveraging the **splk-flx** component.

Important

TrackMe 2.4.18 — volume outliers are now a component of their own.

- The index-level use case of this paper (**use case 2**: Flex over the Splunk licence usage) is superseded by the **Volume Outliers (VOL)** component: one tenant, one entity per index fed by the license usage log, drops and / or spikes per index, inactivity detection, the history backfilled at creation, and a global license usage view — on the Foundation edition as well. Start with the *Volume Outliers white paper* and *VOL — Volume Outliers*.
- This paper remains the reference for **per-feed event-count** behaviour (the splk-dsm models and the Flex `events count per feed` use case), which VOL does not replace: a sourcetype dropping inside an index whose total volume holds is a feed concern, not a license-volume concern.

Hint

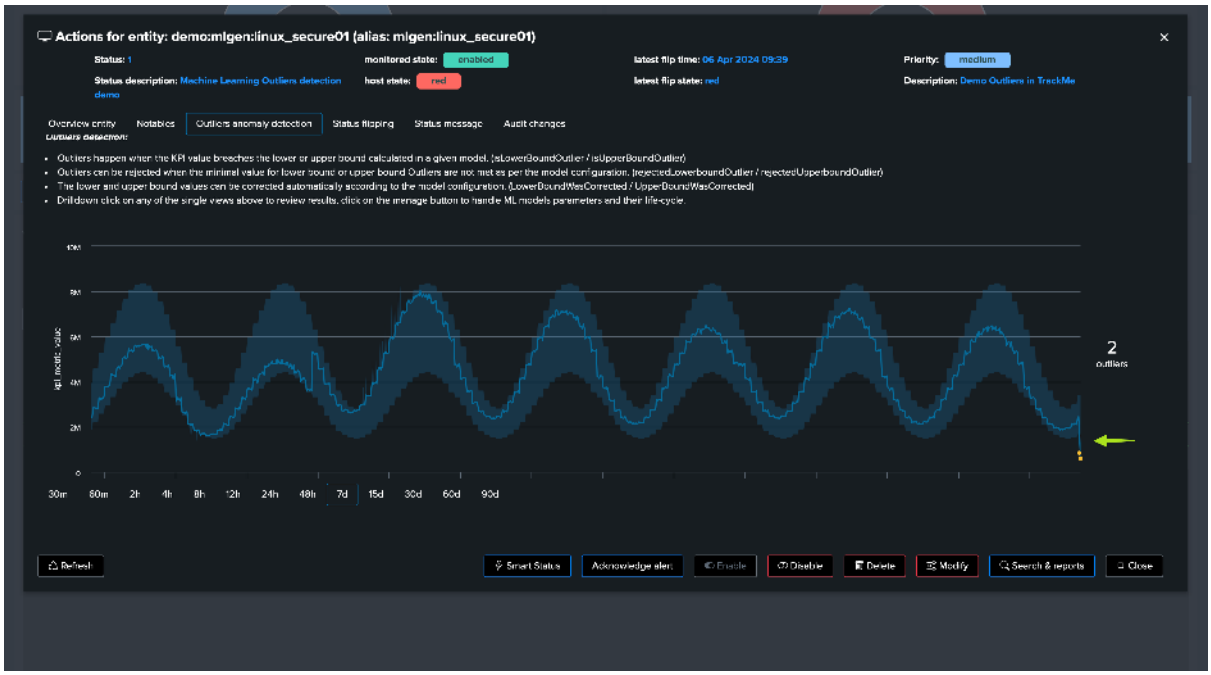
Before we start, let's summarize our 3 main options and features in TrackMe:

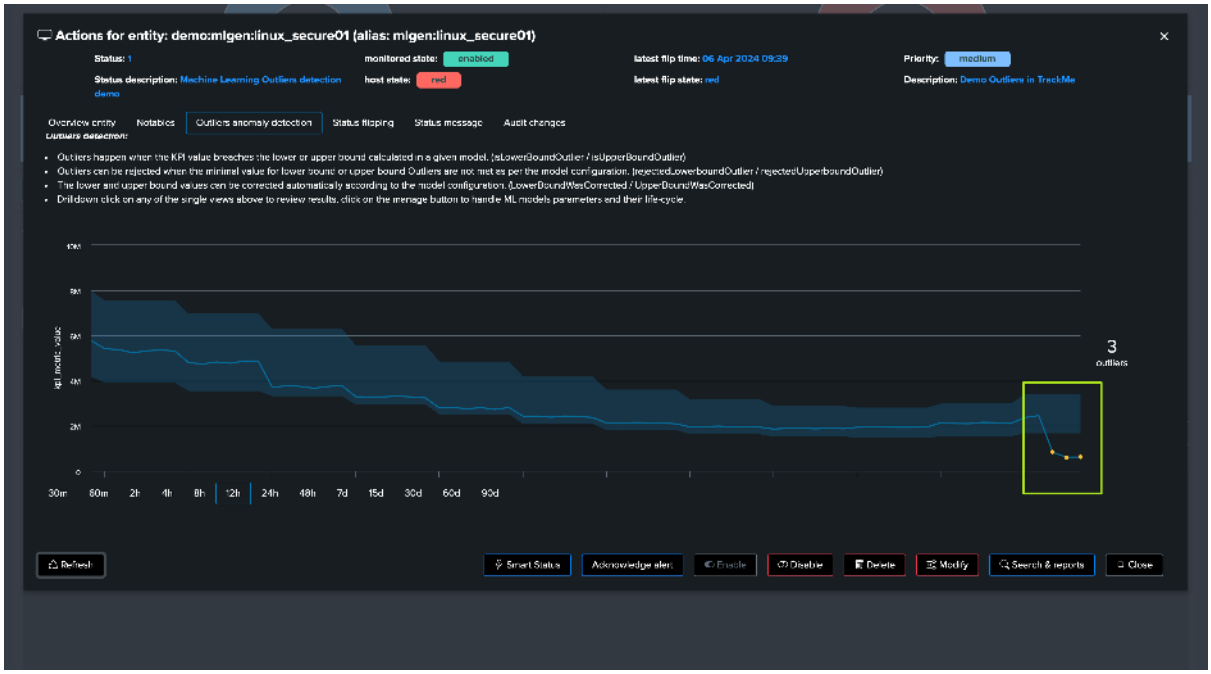
- **splk-dsm:** Out-of-the-box component for feeds tracking, performs Machine Learning based Anomaly Detection against the events count metrics driven by the ingest.
- **splk-flx (licensed customers):** Flex Object tracker component, allows you to orchestrate any kind of SPL logic, translates results into TrackMe entities, turning these into Key Performance Indicators (KPIs) and applying Machine Learning based Anomaly Detection.
- **Use case 1 with Flex:** We will demonstrate how to use the Flex Object tracker to track the events count of Splunk feeds, and apply Machine Learning based Anomaly Detection against these metrics. (These use cases have been added to the Flex Object library in TrackMe 2.0.89)
- **Use case 2 with Flex:** We will demonstrate how to use the Flex Object tracker to track the license usage volume metrics of Splunk feeds, and apply Machine Learning based Anomaly Detection against these metrics.

Objective: Detecting abnormal events count drop in Splunk feeds

Our objective can be summarised in a few charts, as bellow:

Our feed is still making it to Splunk, we get events but the number of events received has significantly decreased:





Let's highlight the main challenges:

- **Feeds behaviors:** All feeds can act differently. Some are continuous with very stable volumes, while others are much more volatile and can slightly change depending on the period of the day or time.
- **Scalability:** We need to be able to scale our detection with no restrictions.
- **Pipeline abstraction:** We also possibly need to abstract the pipeline from the feed itself. Depending on your requirements, the main factors could be simply based on indexes, combinations of indexes and sourcetypes, or even more complex factors.
- **Distinguishing between normal and abnormal behaviors:** Feeds suffering from an abnormal drop in events count are still likely sending events to Splunk. This is a different and more complex use case than tracking feeds interruption.

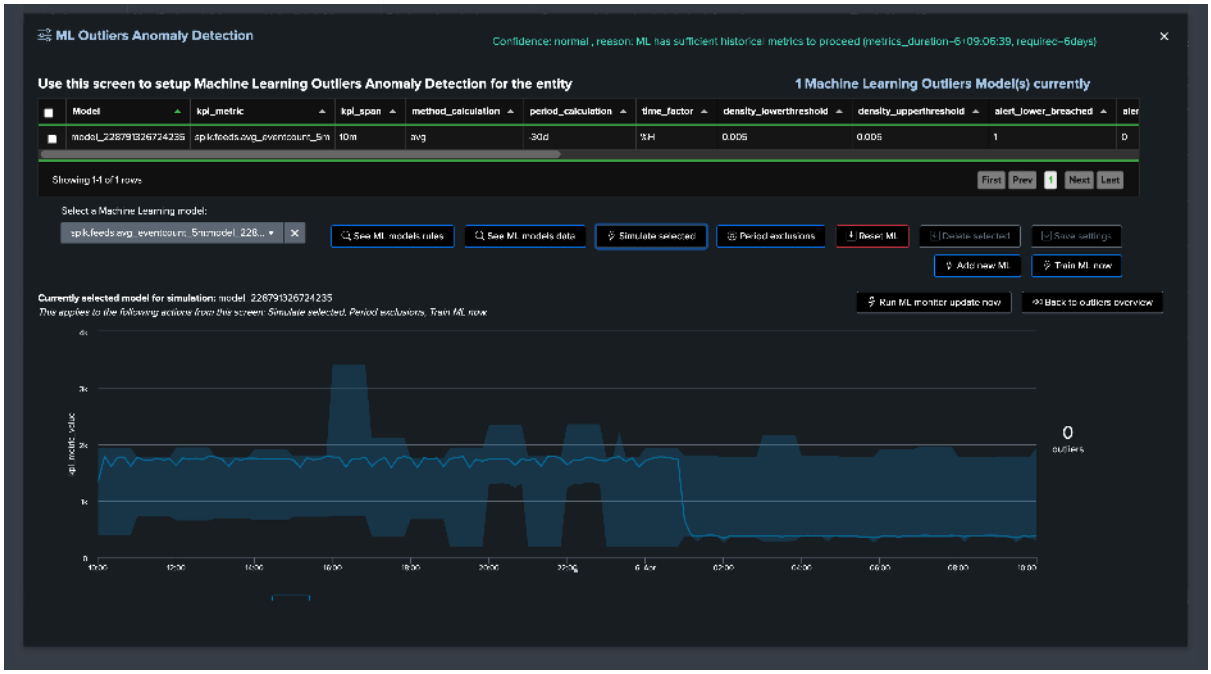
What does TrackMe do out of the box? (feeds tracking with splk-dsm)

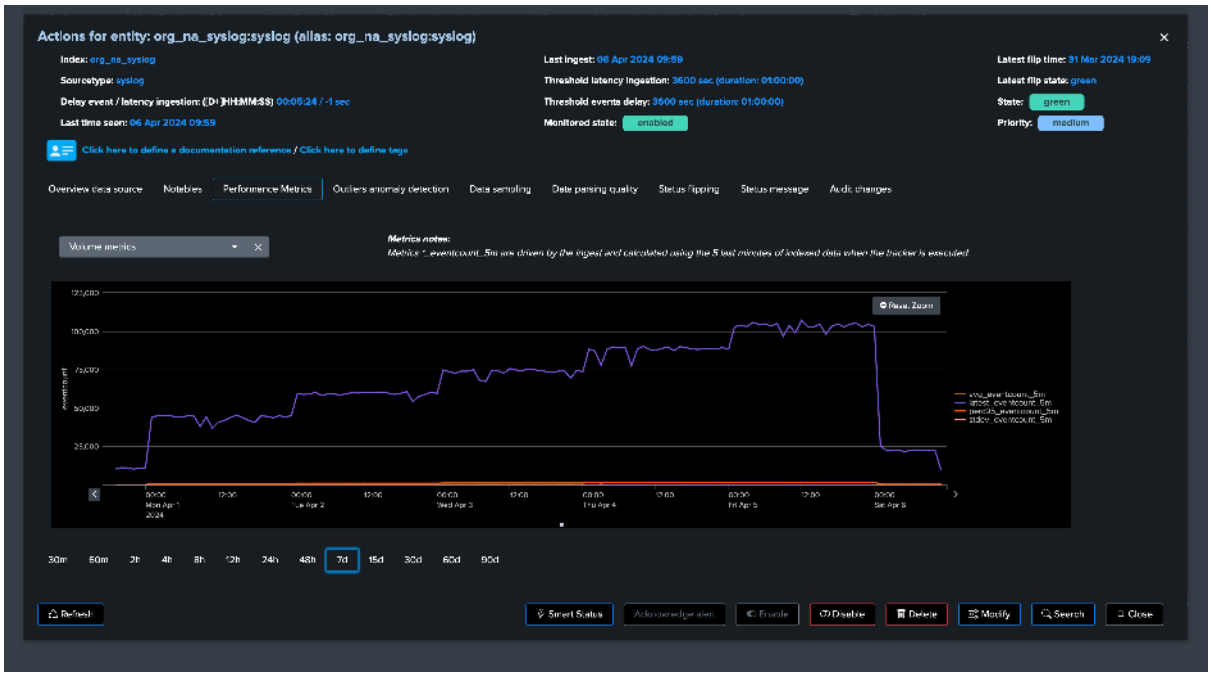
TrackMe provides a main component for data sources tracking (splk-dsm) which performs this detection out of the box:

- When you configure the splk-dsm component, TrackMe performs an association of factors (by default indexes and sourcetypes), creates and maintains entities for each factor, called TrackMe entities.
- Part of the Outliers Anomaly detection framework, TrackMe creates and maintains automatically Machine Learning models based on the events count metrics: `splk.feeds.avg_eventcount_5m`
- Default system wide settings for the Machine Learning models are applied when TrackMe discovers a new entity.
- From this stage, TrackMe will continuously monitor the events count and apply Machine Learning based Anomaly Detection against these entities.
- Machine Learning models can fine-tune on a per entity basis, and different concepts such as ML confidence are actionned by TrackMe to limit the risk of false positives.
- When a given entity is detected as an anomaly, this defines a flag `isAnomaly` which influences the global entity health status, turning possibly an entity from `green` to `red`.

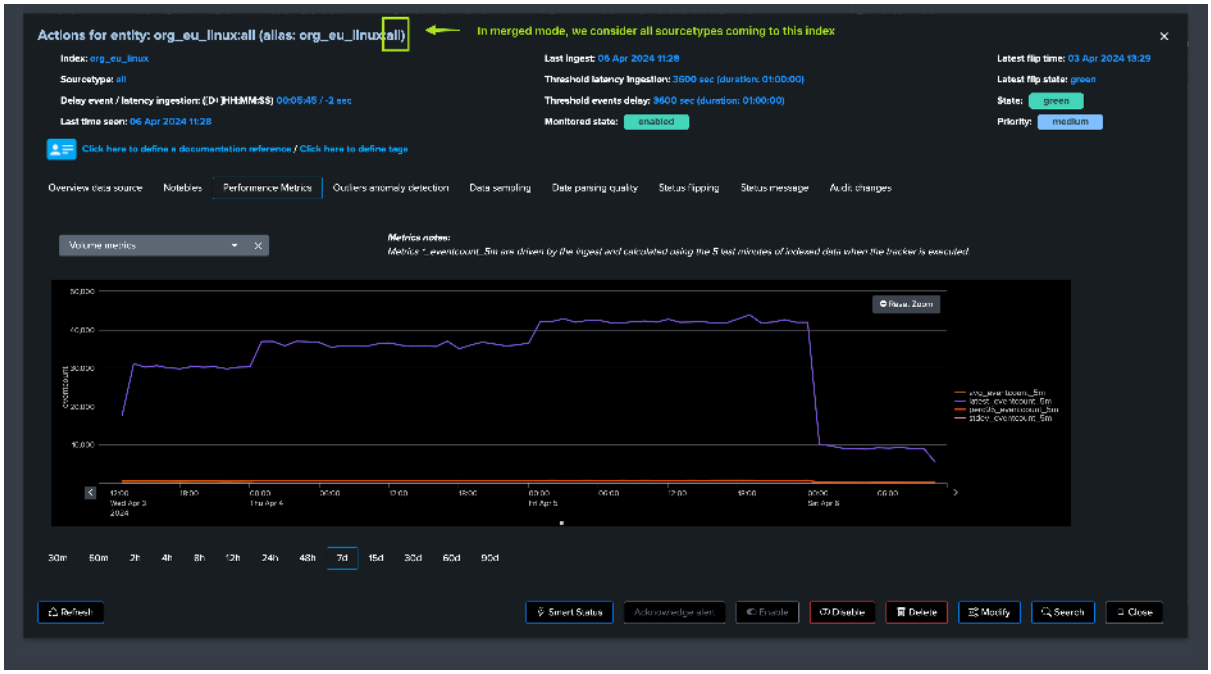
The example below shows a TrackMe data source entity, conditioned by the couple index/sourcetype:

We can observe a slight decrease in the events count. However, this may be normal behavior depending on the period of the week day:





Using merged mode, you can also consider all sourcetypes while using the splk-dsm component to track feeds:





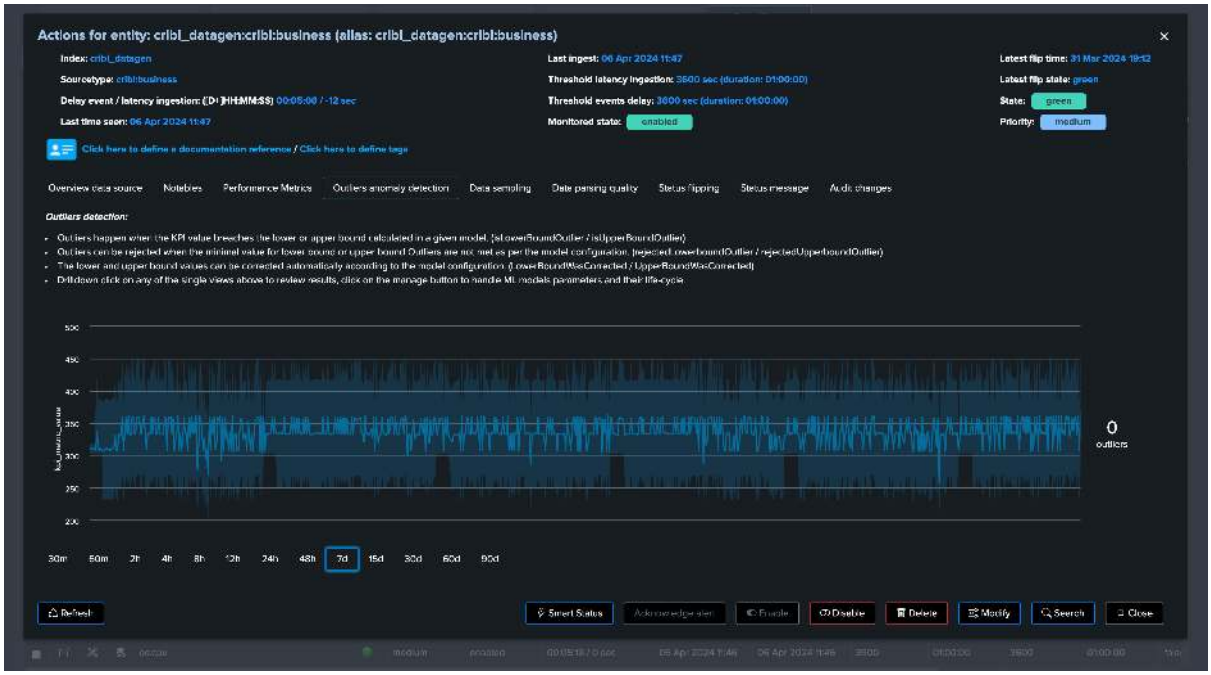
Let's take an additional use case, and let's look at a quite continuous and stable feed:

- In this example, we have a feed which is characterized by a quite stable and continuous activity in terms of events count.
- Therefore, it does not really have a seasonality concept that we should be taking into account.
- We can fine-tune the associated Machine Learning model to use **time_factor: None** which will disable the seasonality concept.

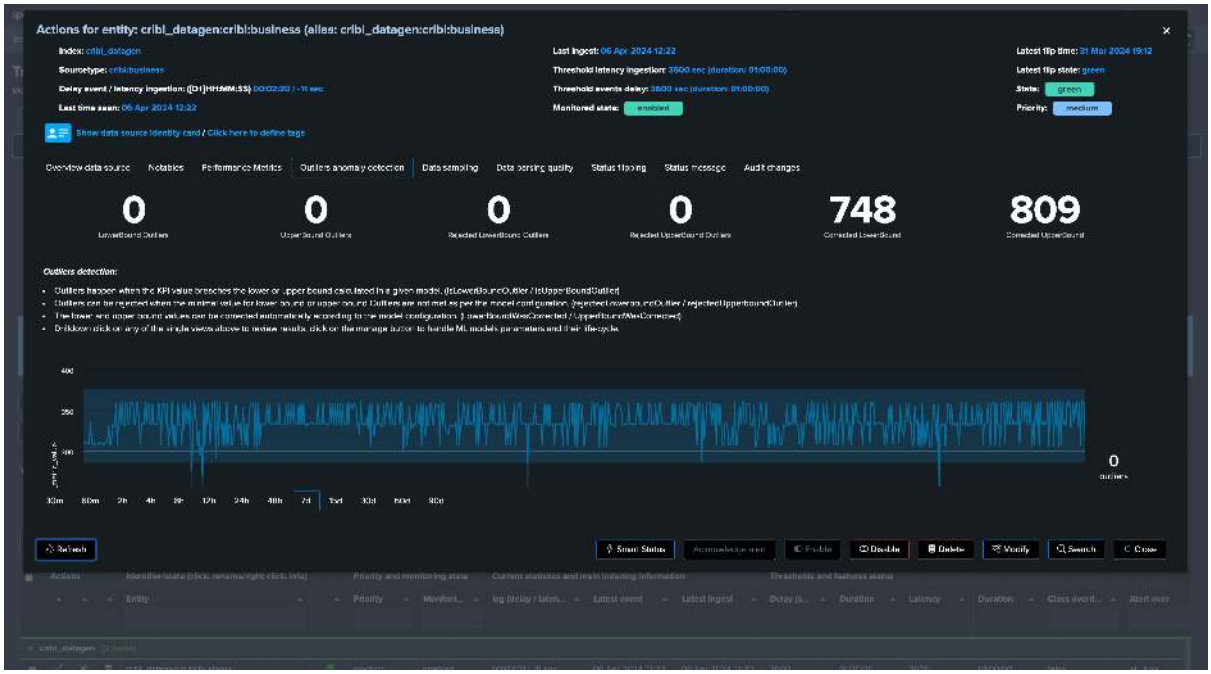
The performance metrics view shows the event count by the ingest over time:



The current Machine Learning model is configured with the default time factor: %H:



By applying time_factor: None, we can disable the seasonality concept, note that TrackMe eventually performs correction if it finds out that the drop is not significant enough to avoid false positives:



More information about this feature in TrackMe

- These metrics are driven by the ingestion. (rolling ingest per 5 minutes)
- Therefore, this is not a strict event counts for a given time period, but instead these metrics demonstrate the ingest behaviour of the feeds from the event counts perspective.
- You can define the default models definition, such as the **time_factor** which conditions how TrackMe handles the seasonality of the events in a given feed.
- You can also define the default **time period** to be considered when TrackMe trains the models.
- Finally, each model can **individually** be fine-tuned as needed, you can specify many settings on per model basis which will apply against a given entity especially.
- In conclusion, detecting abnormal events count drop is an **out of the box feature** of feeds tracking (splk-dsm), so technically TrackMe performs this activity by default and without the need for any extra configuration!

Flex Object (splk-flx): Detect abnormal events count drop using Flex

Hint

Flex Object tracker use case to monitor and detect abnormal events count drop

- The Flex Object tracker component is a very flexible and powerful restricted component of TrackMe. (**splk-flx**)
- These capabilities are available for licensed customers only.
- These use cases were added to the **Flex Object library in TrackMe 2.0.89**.
- It allows to orchestrate any kind of SPL logic, translates results into TrackMe entities, turning these into Key Performance Indicators (KPIs) and applying Machine Learning based Anomaly Detection.
- Because you have full control and within this use case context, we can use a very efficient and costs optimised `tstats` search to track the event count of Splunk feeds according to our needs.
- These metrics will be more “simple” compared to what does the spl-dsm component, these won’t be driven

by the ingest but will be strict events counts over a specific time window.

- Equally, we will define the default definition for the Machine Learning models, which can then still be fine-tuned on a per entity basis.
- Finally, because you have full control, you totally define what an entity should, whenever you want to consider indexes as whole, or more complex scenarios.
- One advantage here is that we do not have to break against granular time concepts to calculate complex metrics such as the latency, so we can create **very efficient searches with the lowest cost in compute**.

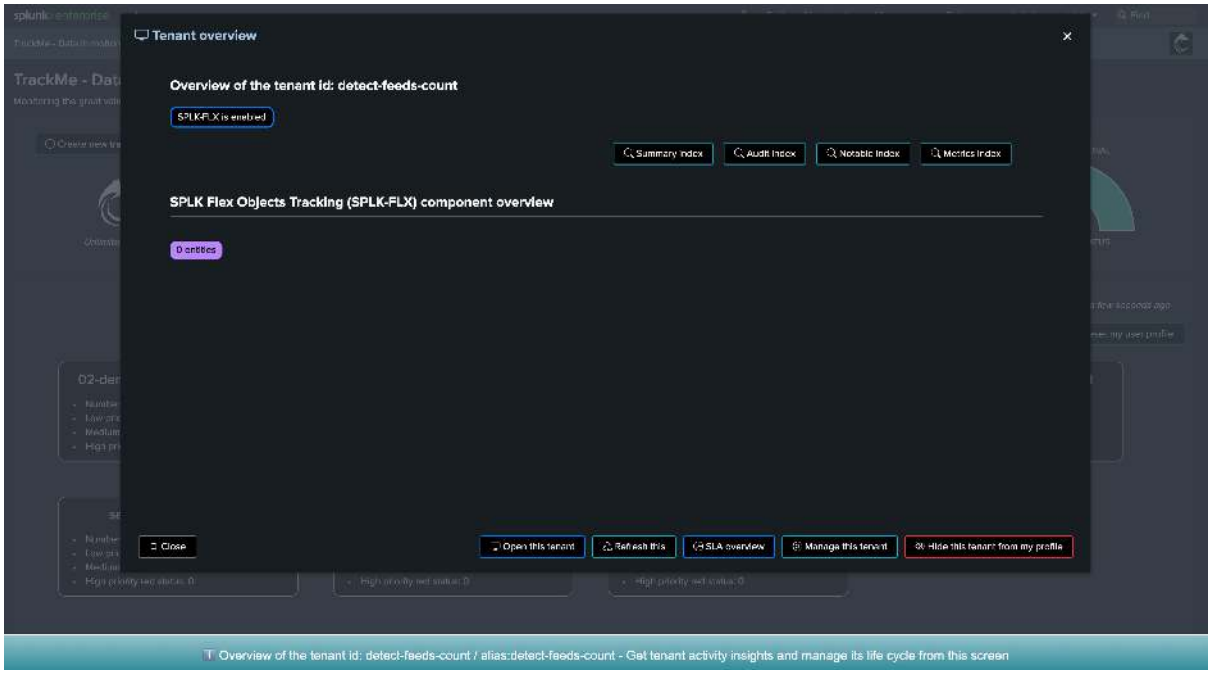
First, let's create a new Virtual Tenant

Depending on your needs and preferences, you can add the component to any existing Virtual Tenant, or dedicate a new Virtual Tenant for this use case.

For the purposes of the documentation, we will create a new Virtual Tenant called **detect-feeds-count**.

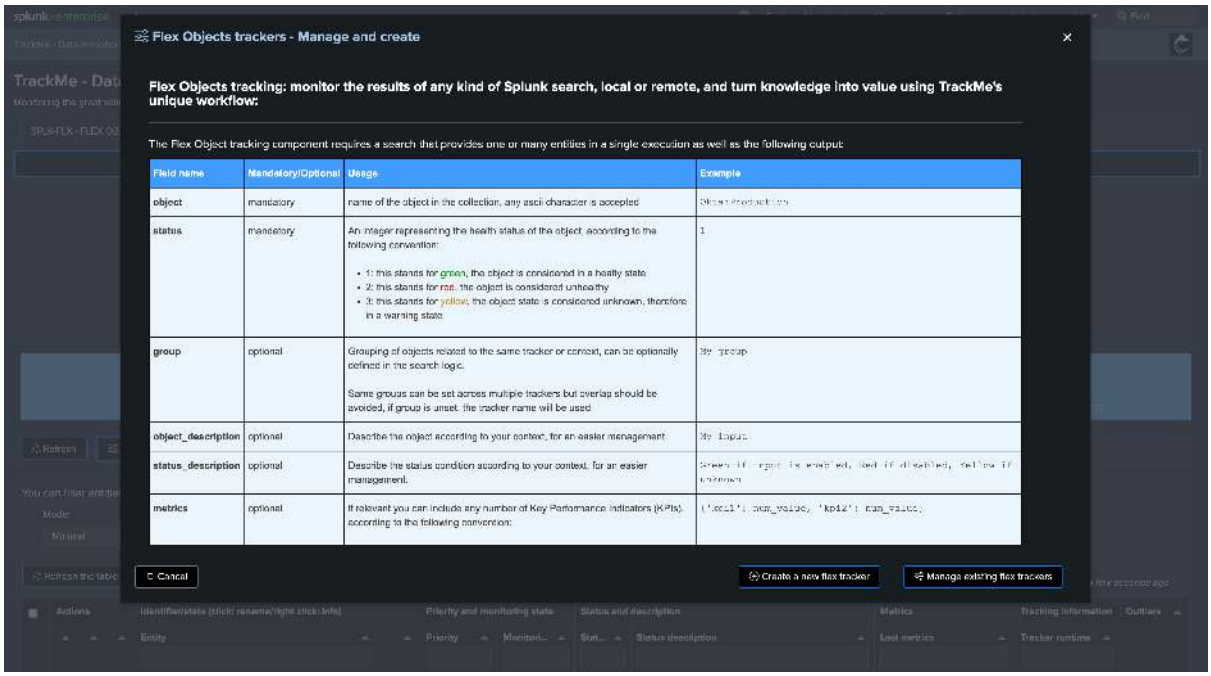


Once proceeded, we have a new Virtual Tenant ready to be used:

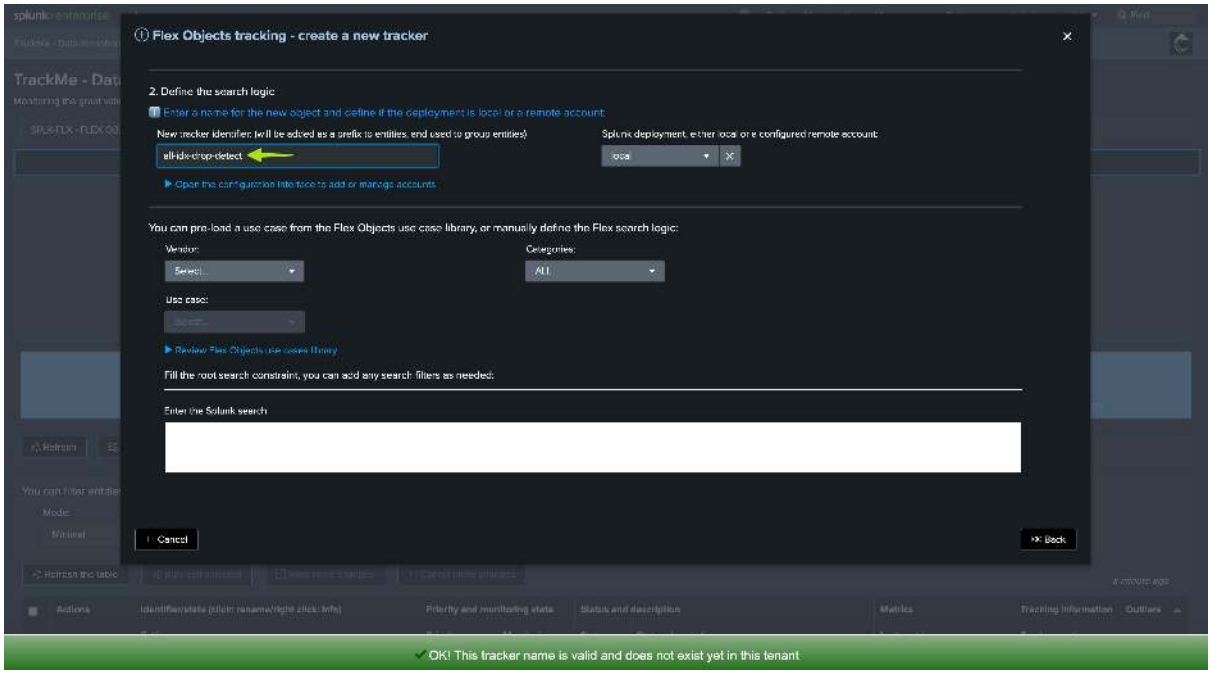


Second, create a new Flex Object tracker

Let's enter the new tenant, and define our Flex Object tracker:



We will create a new Flex Object tracker, called **all-idx-drop-detectt**:



Flex Object tracker source code

Hint

These use cases are available out of the box from TrackMe 2.0.89

- Since TrackMe 2.0.89, we have added these two use cases to the Flex Object library, so you can simply leverage the library to implement the use cases.
- In TrackMe 2.0.98, we added the dictinct count host to the use case, this is also a very valuable KPI to track.
- You can still create the use cases manually (therefore with a prior version of TrackMe), this is what we will demonstrate in this tutorial.

Creating a Flex Object tracker is pretty simple, you can take any of the use cases in the Flex Object library as an example (for instance: splk_license_usage_per_index), we will define our Flex Object tracker as:

Flex object tracker source code:

```
| tstats count as events_count, dc(host) as dcount_host where (index=*) by index
``` fine tune the query above, you can store eventually the root constraint in a␣
↪macro, or simply establish the list here ```

``` define the object, group, alias and description ```
| eval object = index
| eval group="drop-detect"
| eval alias = index
| eval object_description = "Splunk index: " . index

``` set status: in this case this is always going to be green, we will rely on␣
↪outliers detection to detect an abnormal drop in the events count ```
| eval status=1

``` set status_description ```
| eval status_description="Events count drop detection for index: " . index . ",␣
```

(continues on next page)

(continued from previous page)

```
↪latest events_count:" . events_count

``` Set metrics and outliers ```
| eval metrics = "{'splunk.feeds.idx.events_count': " . if(isnum(events_count),␣
↪events_count, 0) . ", 'splunk.feeds.idx.dcount_host': " . if(isnum(dcount_host),␣
↪dcount_host, 0) . "}"

| eval outliers_metrics = "{'splunk.feeds.idx.events_count': {'alert_lower_breached
↪': 1, 'alert_upper_breached': 0, 'time_factor': 'none'},
↪'splunk.feeds.idx.dcount_host': {'alert_lower_breached': 1, 'alert_upper_breached
↪': 0, 'time_factor': 'none'}}"

``` do not alert if the index stops receiving events entirely, setting this to 0␣
↪means this, otherwise you can define a value in seconds, if the index stops␣
↪receiving events, it will turn red for inactivity ```
| eval max_sec_inactive=0
```

We will choose a certain time period, as our search does not have to deal with a time granularity concept (we are not breaking against the time), we have a super efficient and very low cost tracker!

Hint

Fine tuning the Flex Object tracker

- You can fine tune the root constraint (the "where") to match your needs and especially filter on the indexes you want to track.
- You can also modify the break by sequence to match your needs, for instance including the sourcetype or including additional factors such as custom indexed fields.
- Finally, the group is used to logically group entities together, you can also fine tune the group definition to match your needs. (such as using a specific indexed field providing a company or perimeter concept, etc)

Rolling versus Absolute Key Performance Indicator

In next steps, we will define the time period and the frequency of the tracker, depending on these settings and depending on your preferences, we influence the meaning of the KPIs:

- **Rolling Key Performance Indicator:** This is a metric that is frequently executed by the tracker, and the time period is relative to the current time.
- **Absolute Key Performance Indicator:** The time period length matches the execution frequency (cron), so we produce a strict events count for a definitive period of time.
- The meaning of the metrics will be different depending on these settings, although the finality remains the same and both will be equally able to detect abnormal events count dropped.
- We will cover both options in the next steps of this tutorial.

Let's summarize the main ideas:

Rolling metrics pros/cons

Pros:

- Easier to understand. The metric is the real strict event count and would match manual comparison searches.
- Can be easily backfilled, so you don't need to wait for TrackMe to build the historical knowledge before the tracking is operational.

Cons:

- More sensitive to failures. If the tracker schedule is missed and not executed by Splunk (outage), the metrics will be wrong.
- Always looks at 1 hour back compared to the real-time data. We will need at least 1 hour to detect a significant drop in the events count. (However, the use case is about trend analytics, so this last argument is in reality not a real issue)

Absolute metrics pros/cons

Pros:

- Less sensitive to failures, if the tracker schedule is missed and not executed by Splunk, there will be no impact on the detection capabilities.
- Represents a different and valuable way to look at the trending of the events count rather than a basic strict events count.

Cons:

- Cannot be easily backfilled, the trend can only be calculated at point in time and backfilling it is much more challenging.
- Does not represent the basic events count, so one cannot compare with simple Splunk searches, in some cases when doing investigations, analysts can struggle to understand the metrics.

Absolute Key Performance Indicator

The time period length must match the frequency of the tracker, so we have a strict events count for a definitive period of time:

- earliest: -1h@h
- latest: @h
- cron_schedule: `0 * * * *`

Note: you can optionally be more protected about a single failure of Splunk in executing our Tracker once per hour, you can for instance execute it twice per hour, ML calculations will be affected as we do not use a sum but avg/latest and so others, so a duplicated metrics has no impacts.

Rolling Key Performance Indicator

We would for instance execute on a high frequency the Flex Tracker against a bigger time period length:

- earliest: -60m
- latest: now
- cron_schedule: `*/5 * * * *`

Defining the Machine Learning model general rules

In our Flex Object tracker, we have defined the following line:

```
| eval outliers_metrics = "{'splunk.feeds.idx.events_count': {'alert_lower_breached
↪': 1, 'alert_upper_breached': 0, 'time_factor': 'none'},
↪'splunk.feeds.idx.dcount_host': {'alert_lower_breached': 1, 'alert_upper_breached
↪': 0, 'time_factor': 'none'}}"
```

This does the following:

- Automatically define a Machine Learning model definition per entity
- The model will by default alert only for Lower Bound breaches, which means in our context an abnormal drop in the events count.

- Finally, we decided to define by default a time_factor: none, which means we disable the seasonality concept, so we have single LowerBound and UpperBound ranges per entity.
- We could also choose a different pattern between: %H, %H%M, %w%H, %w%H%M

Hint

Machine Learning time_factor

- In this example, we have chosen to define a time_factor: none, which means we disable the seasonality concept.
- You can also do the opposite and enable by default a time_factor of your choice.
- Remember that you can always update the time factor per entity as needed.
- This statement in our Flex Object tracker defines the default definition of the models and is only used once at the time of the entity discovery or if you delete the Machine Learning model for a given entity. (so it is re-created automatically at the next execution)

Backfilling the Key Performance Indicator (Absolute KPI)

ONLY AFTER A FIRST EXECUTION OF THE TRACKER: You can optionally choose to backfill the metrics. This is very useful for the Absolute KPIs:

make sure to replace the tenant_id, the root constraints and the period as per your needs. Here we will be backfilling for the past 30 days which would remain efficient even at large scale

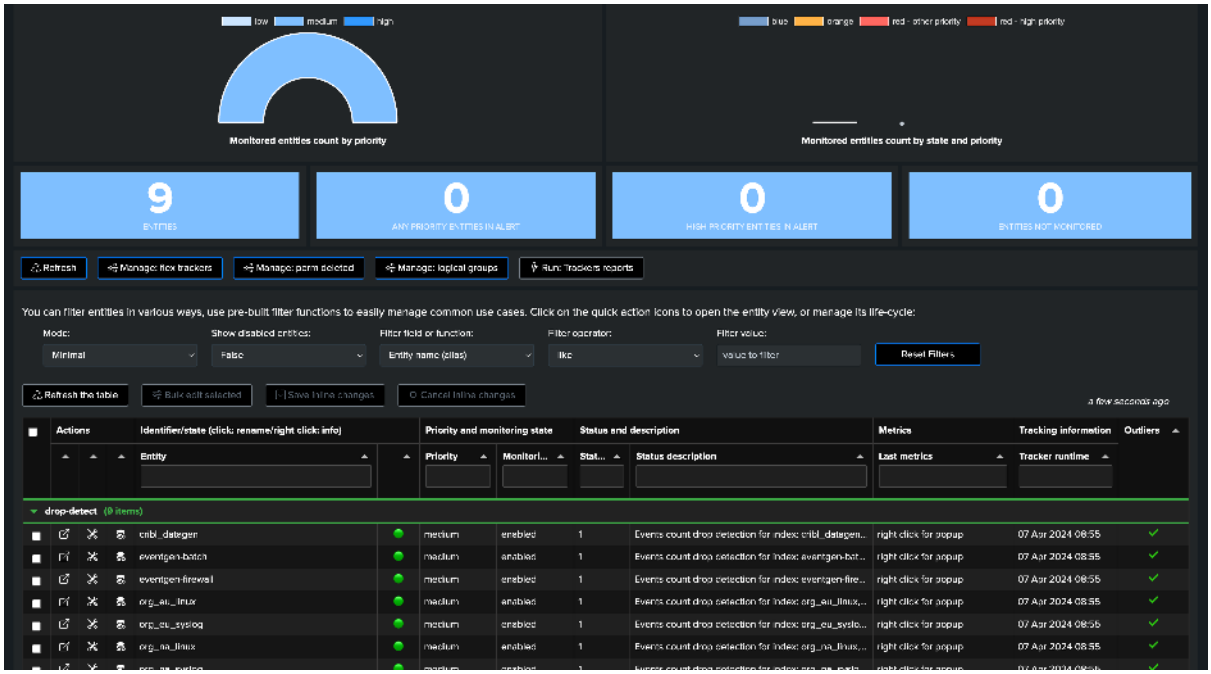
```
| tstats count as trackme.splk.flx.splunk.feeds.idx.events_count, dc(host) as␣
↪trackme.splk.flx.splunk.feeds.idx.dcount_host where (index=*) earliest=-30d␣
↪latest=@h by _time, index span=1h

| lookup trackme_flx_tenant_detect-feeds-count alias as index OUTPUT _key as␣
↪object_id, object
| eval tenant_id = "detect-feeds-count", object_category="splk-flx"
| where isnotnull(object_id)

| mcollect index=trackme_metrics split=t object, object_category, object_id,␣
↪tenant_id
```

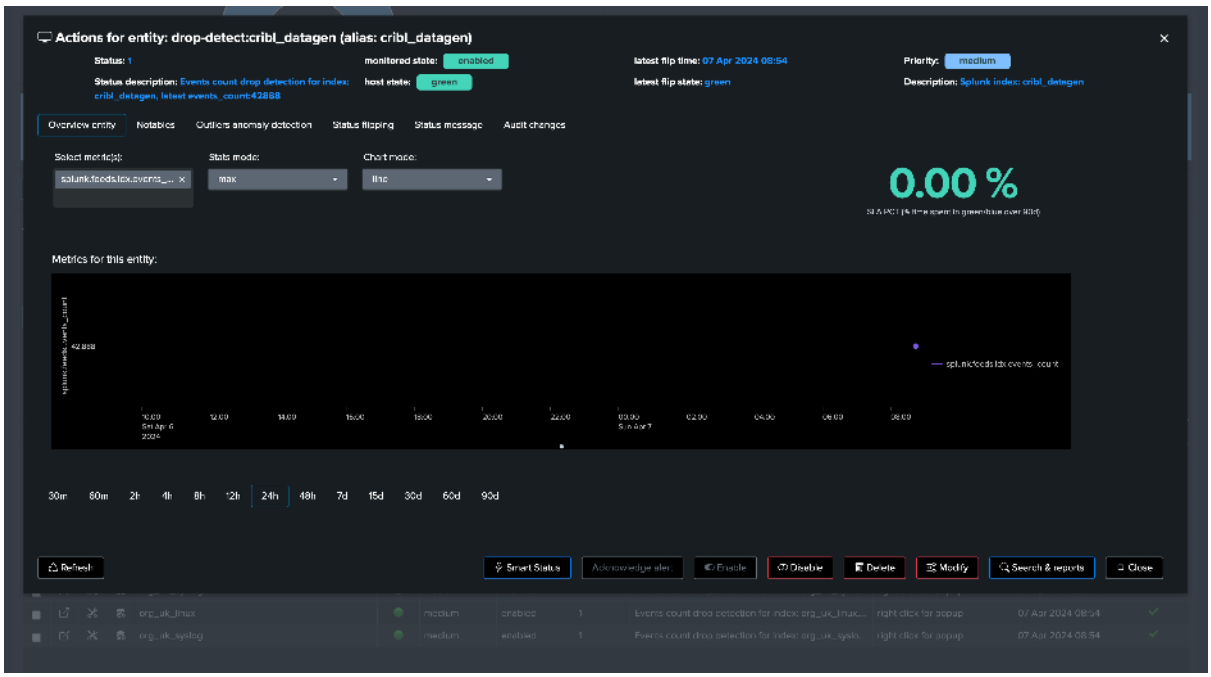
Final, execute and review entities

Finally, once we have first executed the tracker, we will be able to see one TrackMe entity per (active) index:

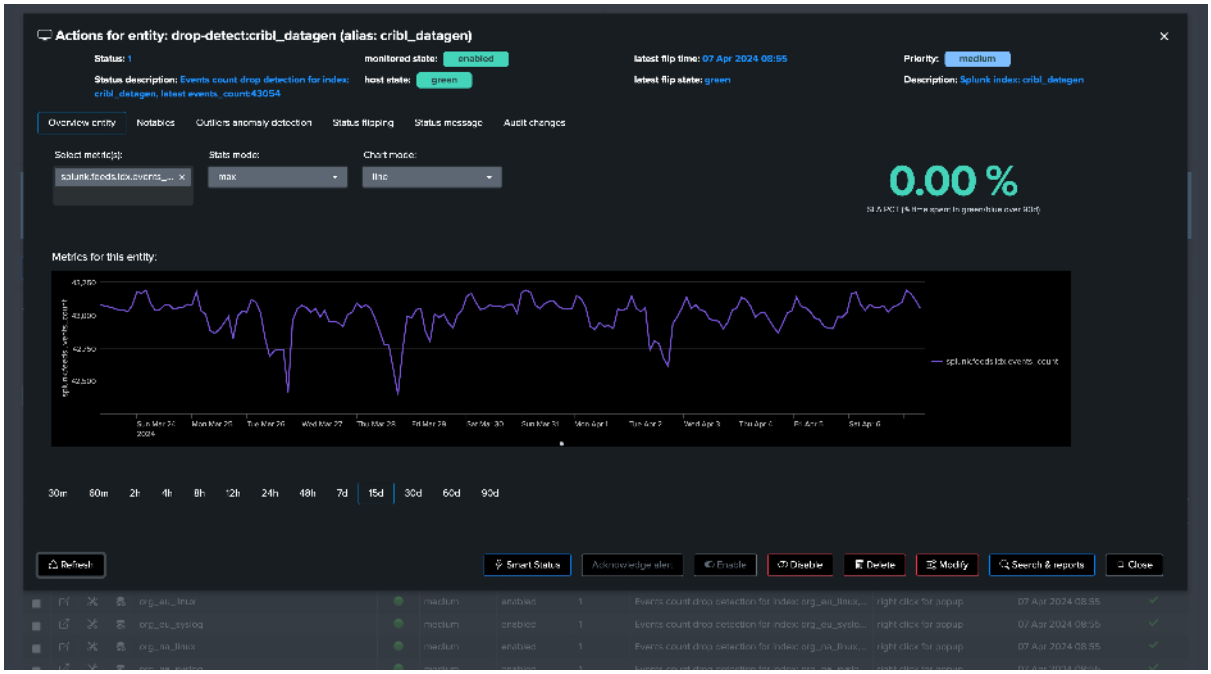


Depending if we wanted to use Rolling or Absolute KPIs, and backfill, we either see the first point of metric or a full backfilled metrics:

Not backefilled:

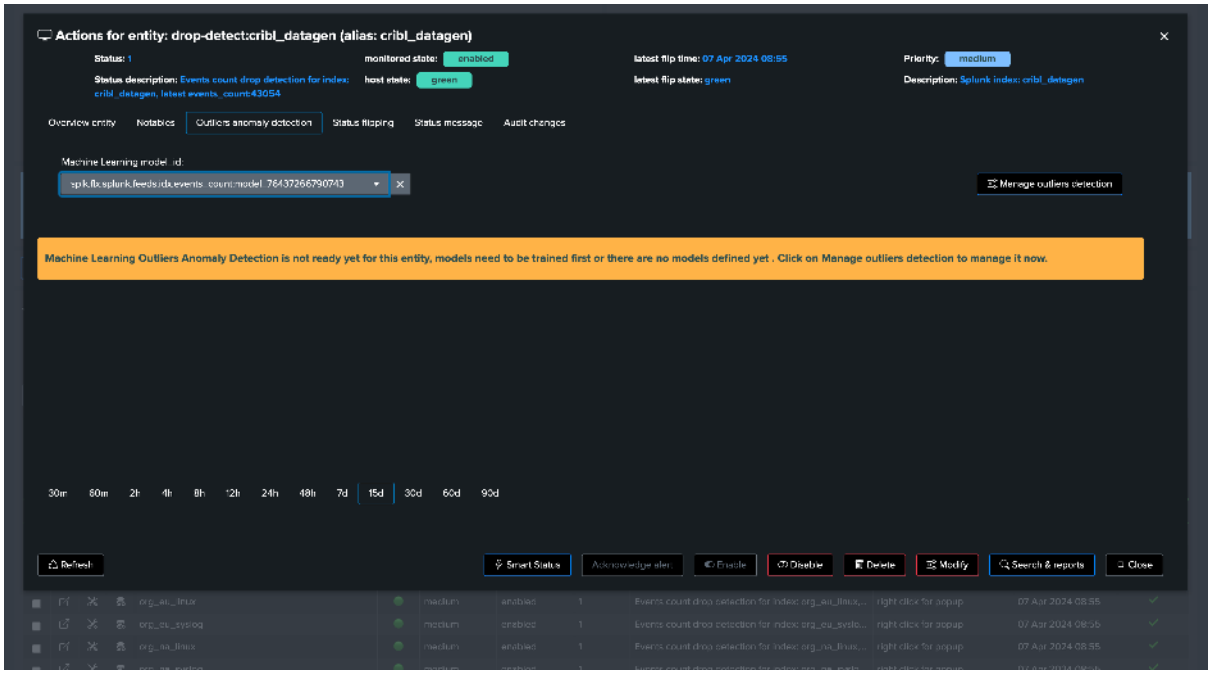


Backfilled:

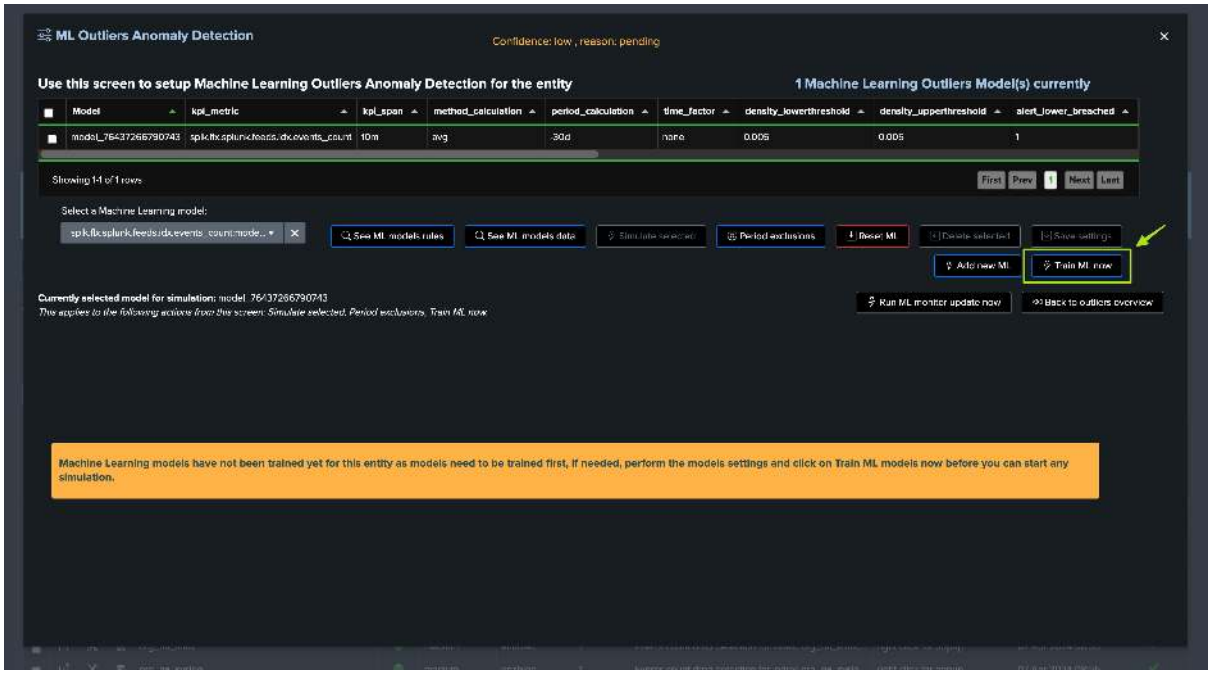


Third, let's have a look at the Machine Learning models

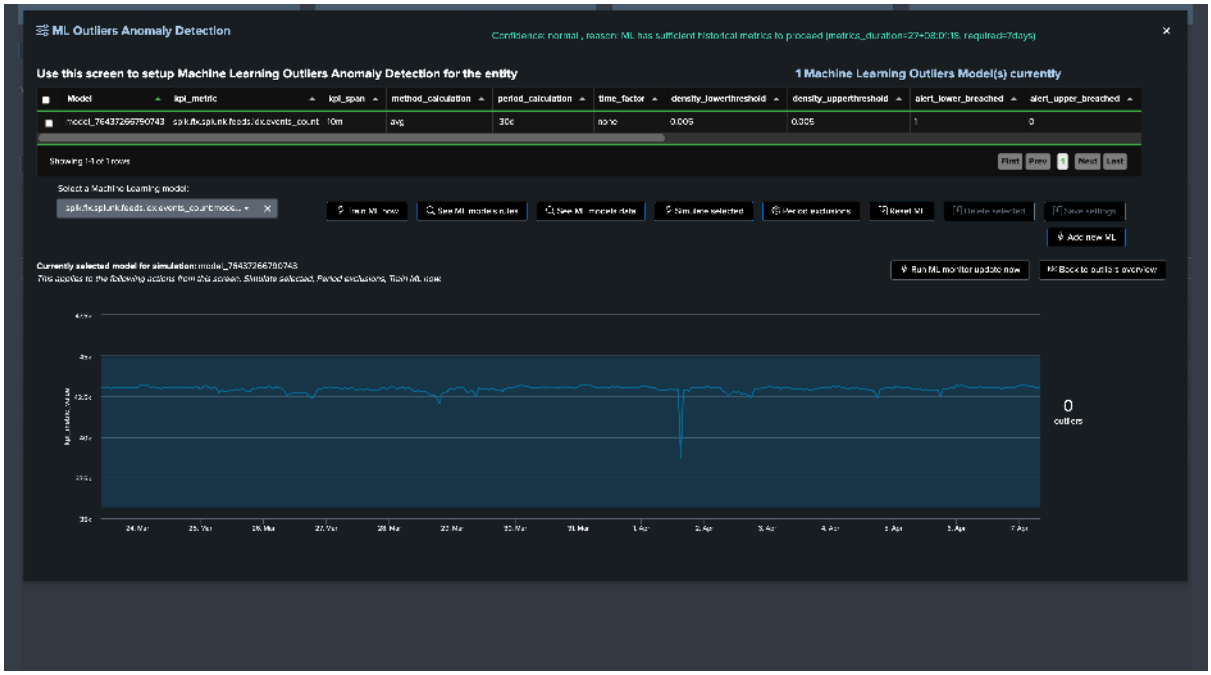
Since we've just created our tenant and entities, the Machine Learning models are not yet trained, we will have to wait for the next execution of the tracker to see the models being trained, or manually trigger it:



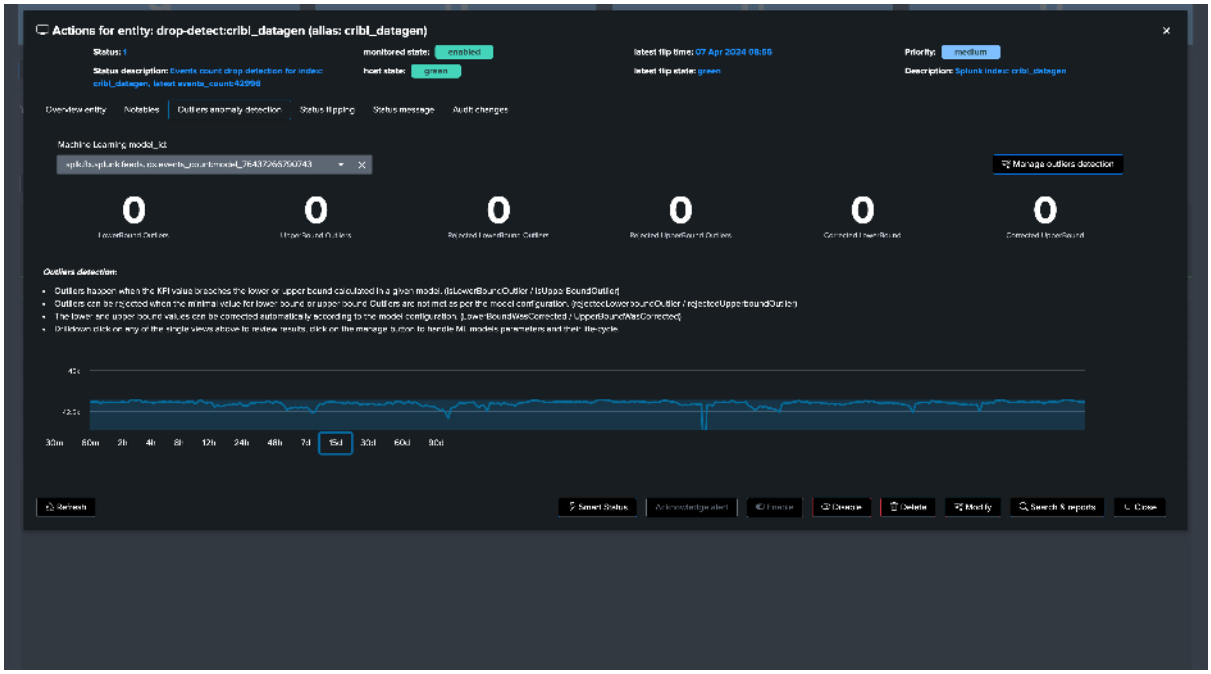
For the purposes of the tutorial, let's run the training now:

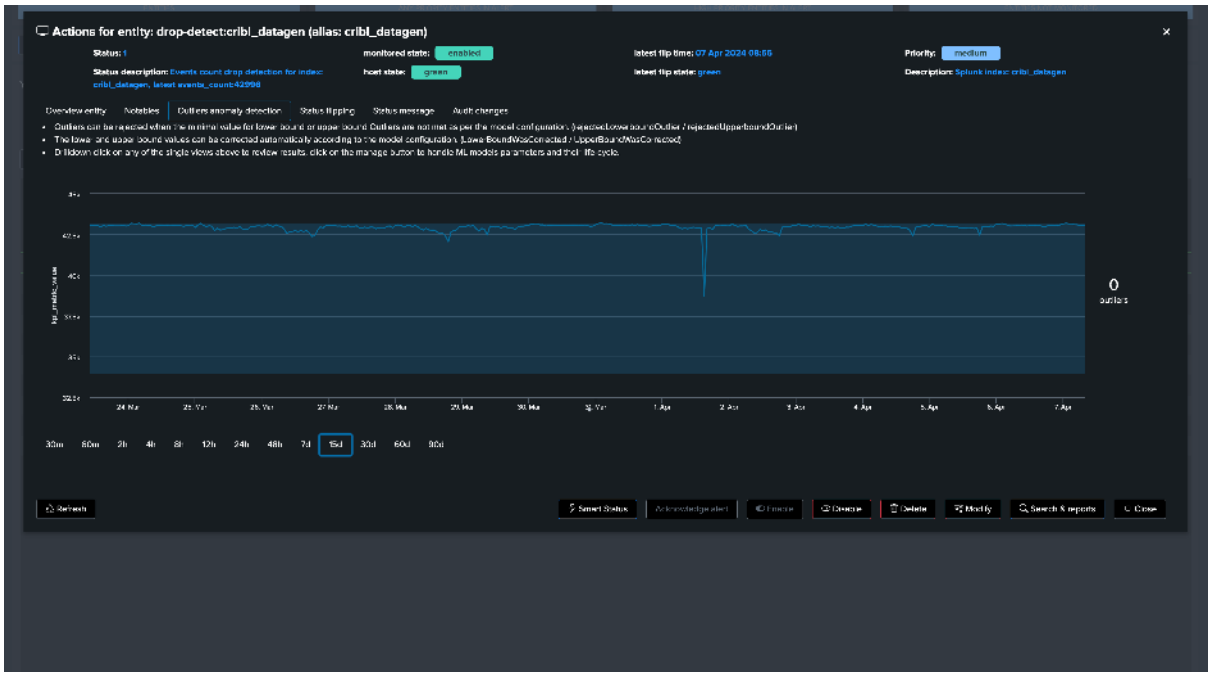


Once trained and simulated, we can see the Machine Learning models being applied:



Going back on the live Machine Learning Outliers detection screen (if the chart does not appear, refresh the page!):





note: Additional options can be set to fine tune Models, notably you can specify minimal values for the LowerBound breaches. (Anything below this value will be rejected as LowerBound Outliers)

Conclusion on drop detection

At this stage, we have:

- A ready and operational framework to continuously track and detect abnormal events count drop in Splunk feeds.
- TrackMe will automatically train and maintain Machine Learning models for each entity, and apply Anomaly Detection against these models.
- Very little fine-tuning is really required, so you can focus on the core of the use case.

- You can also fine tune the most important and critical feeds of your environments easily in TrackMe.

Flex Object use case 2 (splk-flx): Detect abnormal events count drop using Flex and Splunk licence usage

Note

Superseded in 2.4.18 by the *Volume Outliers component*, which delivers this use case out of the box — incremental collection, backfill, per-index direction, inactivity, projection — without a Flex tenant or template. The templates remain available for custom needs; the walkthrough below is kept as a reference.

Hint

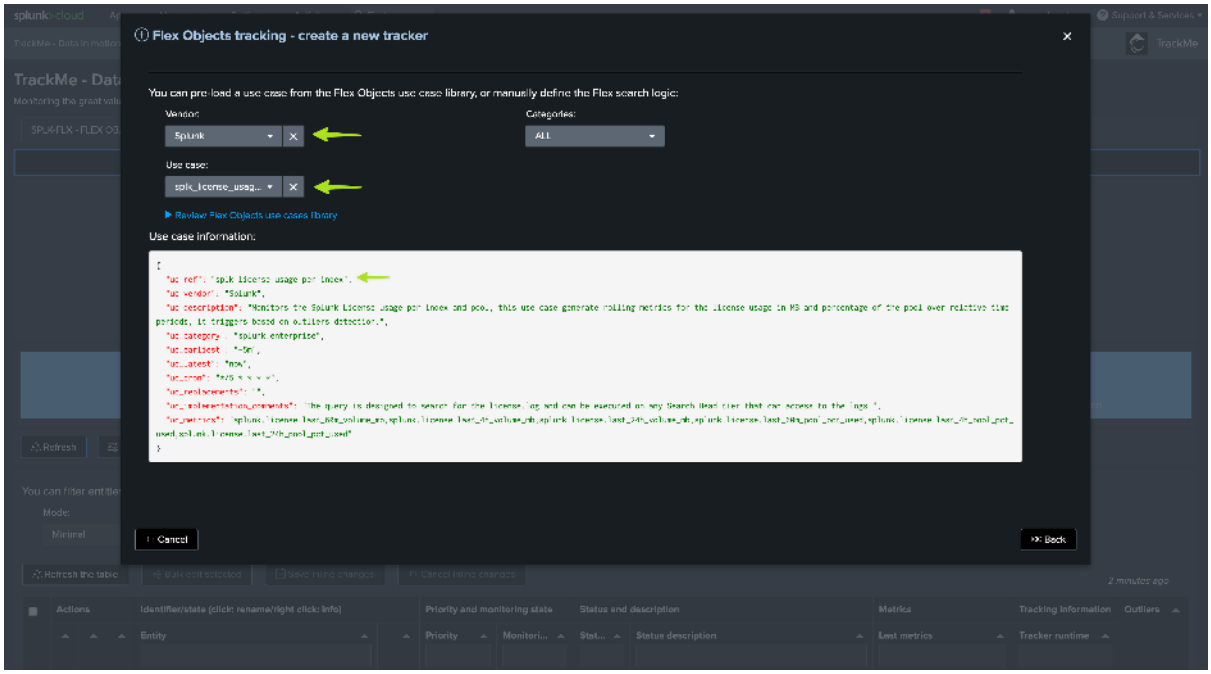
Flex Object tracker use case to monitor and detect abnormal events count drop using Splunk licence statistics

- We also have a **third option** with TrackMe FLex Object tracker and using the Splunk licence usage statistics.
- In fact, TrackMe provides an out of the box Flex Object use case called `splk_license_usage_per_index` which does exactly this!
- This use case leverages the Splunk licence usage statistics which accounts against Splunk indexes, this is available and valid for both **Splunk Enterprise customers and Splunk Cloud customers**.
- Unlike the previous use cases, the calculations are based against **Megabytes volume metrics of incoming data**, rather than events count.
- This is also a very efficient use case, the out of the box use case is based on a rolling metric approach, but you could also modify the scheduling plan as we did in the previous example if you wished to turn it into an absolute metric instead.
- This use case calculates **3 ranges of rolling metrics**: -24h, -4 hours and -60 minutes, and applies Machine Learning based Anomaly Detection against the past 24 hours metrics by default.

We will assume our Flex Object Virtual Tenant is ready to be used, let's call the Out of the Box use case

Go with the creation of a Flex Object tracker and select the out of the box use case:

See after this screenshot for an alternative version of the tracker focusing only on the volume based metrics:



Alternative version of the tracker focusing only on the volume based metrics:

```
index=_internal sourcetype=splunkd source="*/license_usage.log" earliest="-24h"␣
↪latest="now"
| stats sum(b) as b, latest(poolsz) as poolsz by idx, pool
| eval last_24h_volume_mb = round(b/1024/1024, 2),␣
↪last_24h_pool_pct_used=round(b/poolsz*100, 2)
| fields idx, pool, last_24h_volume_mb, last_24h_pool_pct_used
| eval object = "idx:" . idx

``` get last 4 hours ```
| append [ search index=_internal sourcetype=splunkd source="*/license_usage.log"␣
↪earliest="-4h" latest="now"
| stats sum(b) as b, latest(poolsz) as poolsz by idx, pool
| eval last_4h_volume_mb = round(b/1024/1024, 2),␣
↪last_4h_pool_pct_used=round(b/poolsz*100, 2)
| fields idx, pool, last_4h_volume_mb, last_4h_pool_pct_used
| eval object = "idx:" . idx
]

``` get last 60 minutes ```
| append [ search index=_internal sourcetype=splunkd source="*/license_usage.log"␣
↪earliest="-60m" latest="now"
| stats sum(b) as b, latest(poolsz) as poolsz by idx, pool
| eval last_60m_volume_mb = round(b/1024/1024, 2),␣
↪last_60m_pool_pct_used=round(b/poolsz*100, 2)
| fields idx, pool, last_60m_volume_mb, last_60m_pool_pct_used
| eval object = "idx:" . idx
]

| stats first(*) as "*" by object
| foreach last_* [ eval <<FIELD>> = if(isnum('<<FIELD>>'), '<<FIELD>>', 0) ]

| eval group="license|pool:" . pool
| eval alias = idx
| eval object_description = "Splunk license usage for index: " . idx . " / pool: "␣
```

(continues on next page)

(continued from previous page)

```
↪. pool

``` set status: in this case this is always going to be green, we will rely on␣
↪outliers detection to detect an upperBound outlier against the rolling 24 hours␣
↪metrics ```
| eval status=1

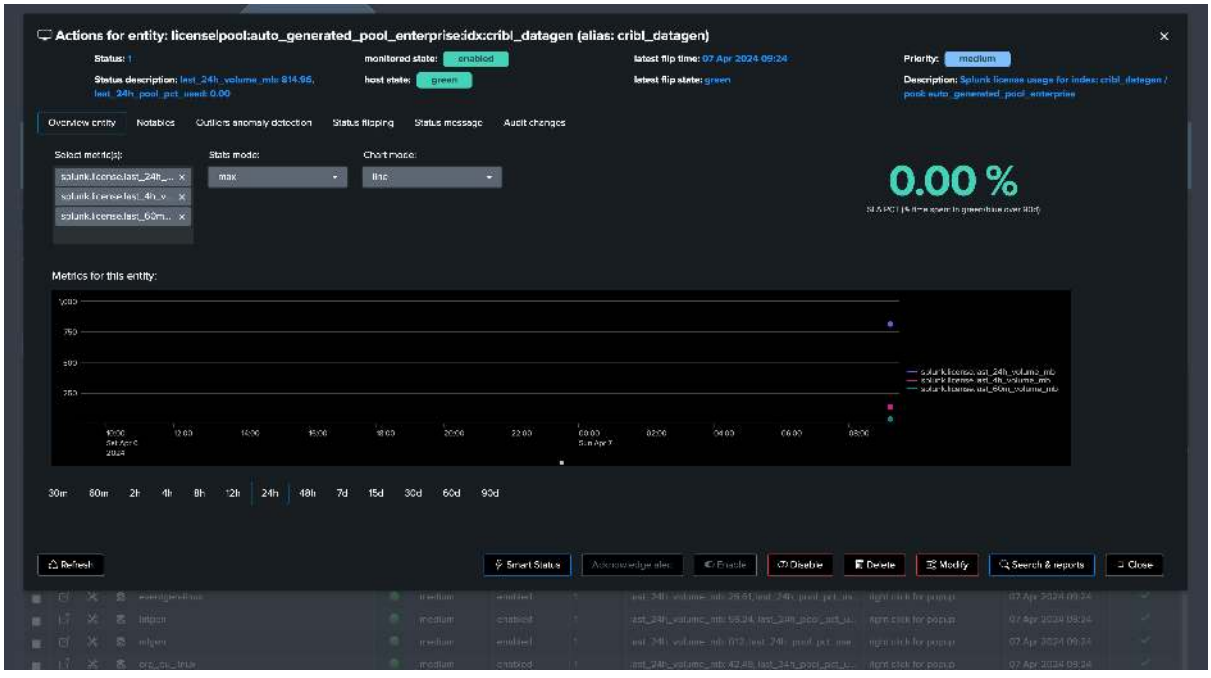
``` set status_description ```
| eval status_description=case(
status=1, "last_24h_volume_mb: " . last_24h_volume_mb . ", last_24h_pool_pct_used:
↪" . last_24h_pool_pct_used,
status=2, "last_24h_volume_mb: " . last_24h_volume_mb . ", last_24h_pool_pct_used:
↪" . last_24h_pool_pct_used,
status=3, "last_24h_volume_mb: " . last_24h_volume_mb . ", last_24h_pool_pct_used:
↪" . last_24h_pool_pct_used
)

``` Set metrics and outliers ```
| eval metrics = "{\"splunk.license.last_60m_volume_mb\": " .␣
↪if(isnum(last_60m_volume_mb), last_60m_volume_mb, 0) . ", \
↪"splunk.license.last_4h_volume_mb\": " . if(isnum(last_4h_volume_mb),␣
↪last_4h_volume_mb, 0) . ", \"splunk.license.last_24h_volume_mb\": " .␣
↪if(isnum(last_24h_volume_mb), last_24h_volume_mb, 0) ."}"

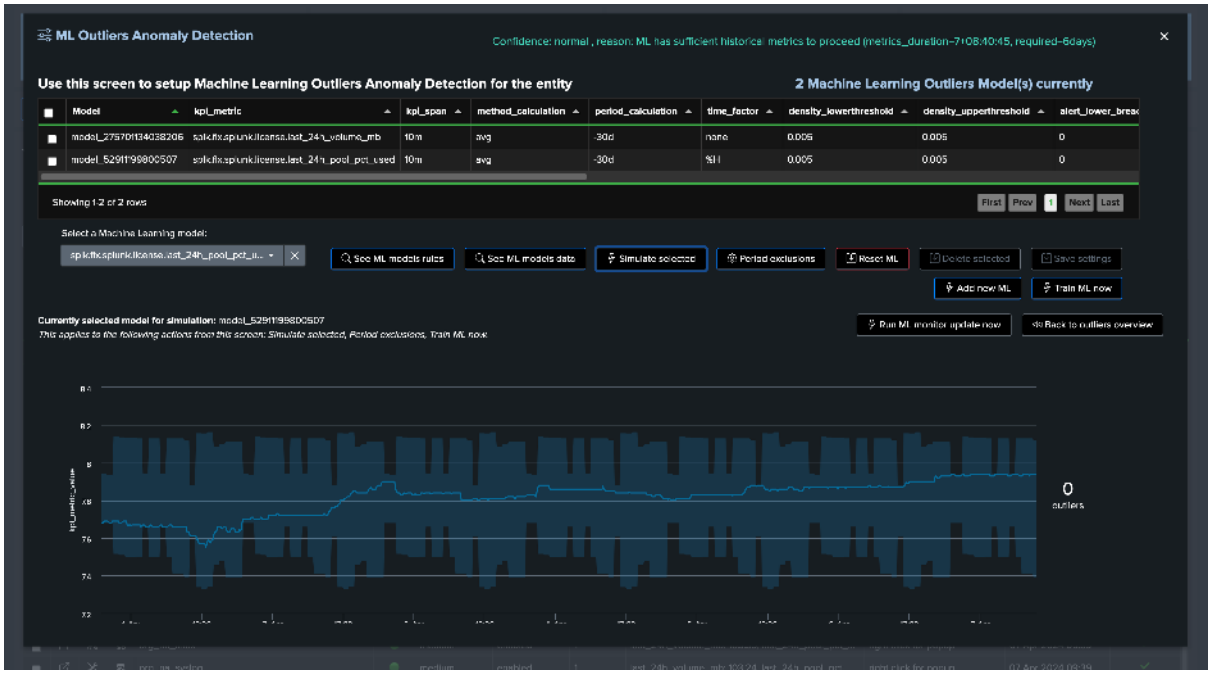
| eval outliers_metrics = "{\"splunk.license.last_24h_volume_mb\": {\
↪"alert_lower_breached\": 0, \"alert_upper_breached\": 1}}"

``` alert if inactive for more than 30 days```
| eval max_sec_inactive=30*24*60*60
```

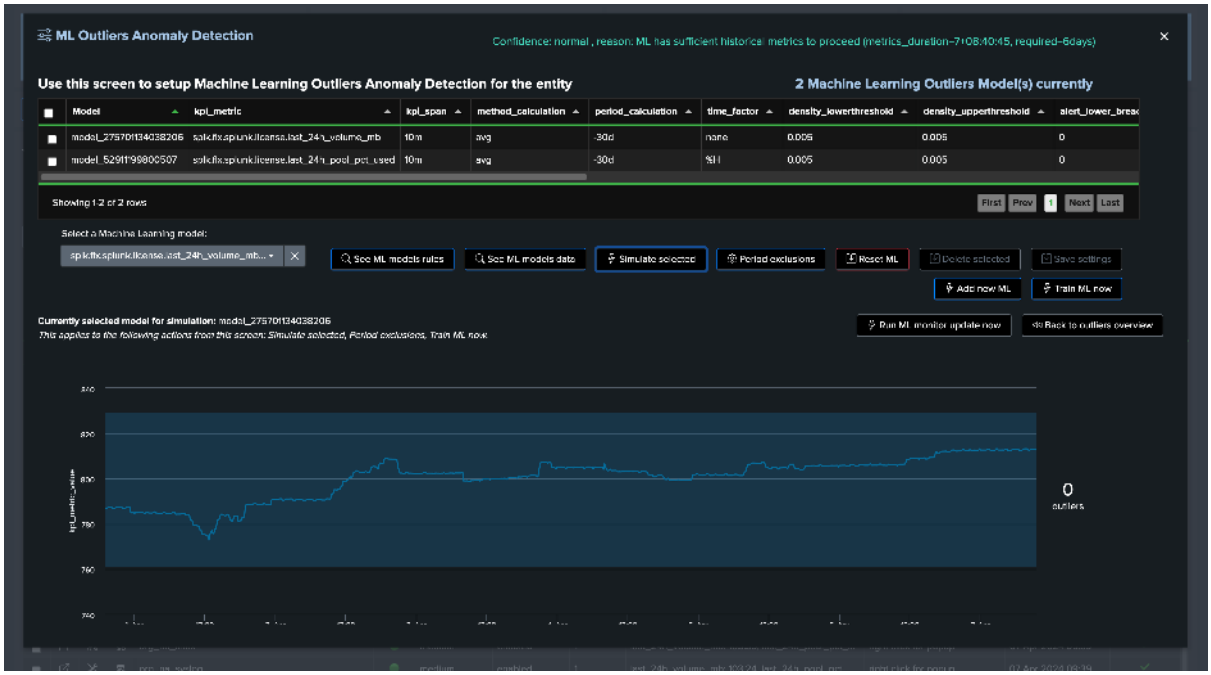
Once first executed, we have one entity per index, with 3 levels of volume metrics



This example shows Machine Learning applied against the licence pool percentage usage with seasonality concept enabled: (for Enterprise customers only)



This example shows Machine Learning applied against the volume metrics with seasonality concept disabled:

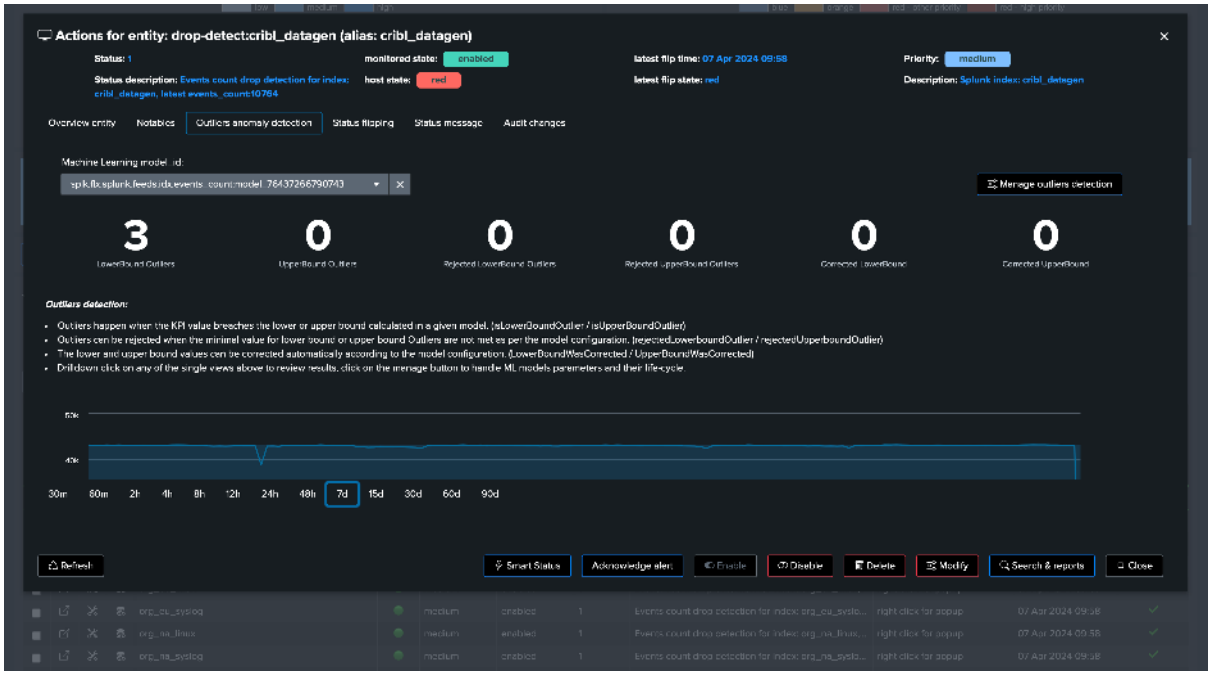


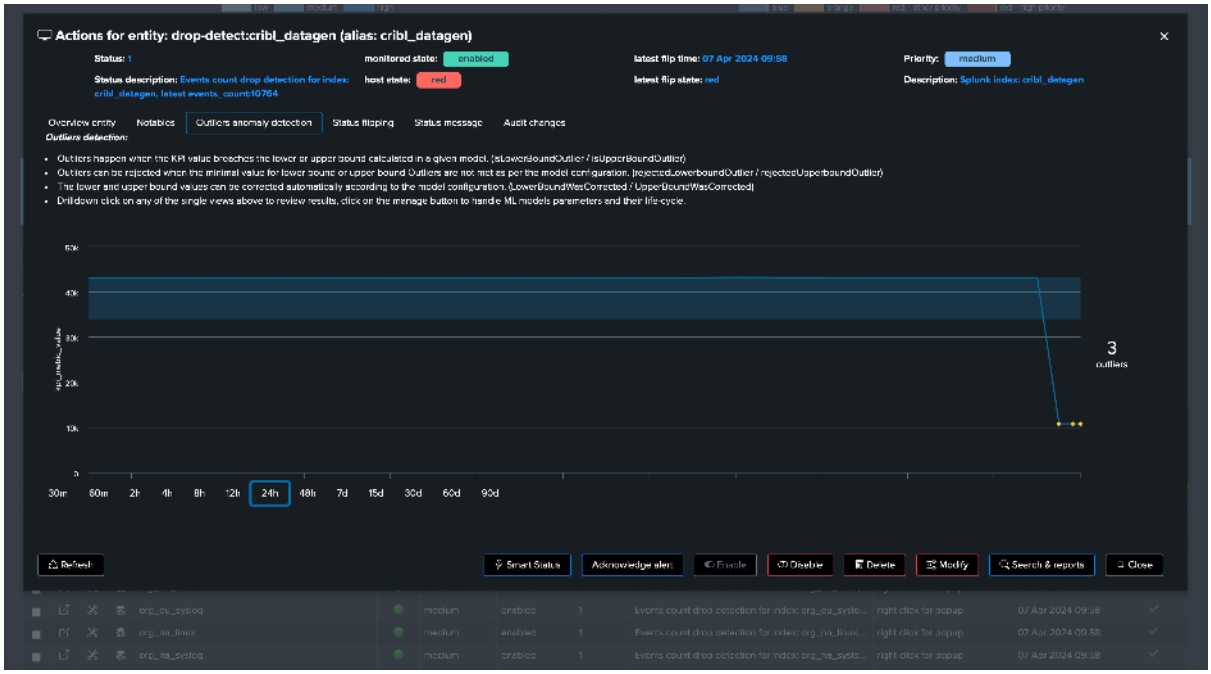
Conclusion: volume based metrics

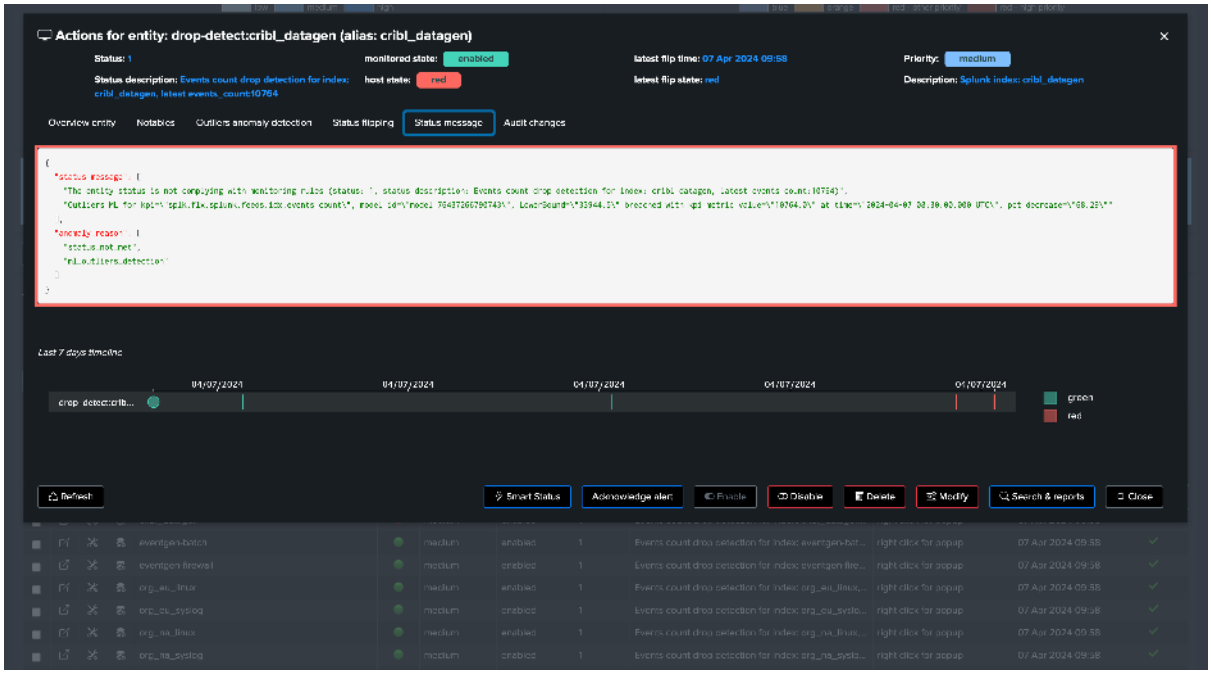
At this stage:

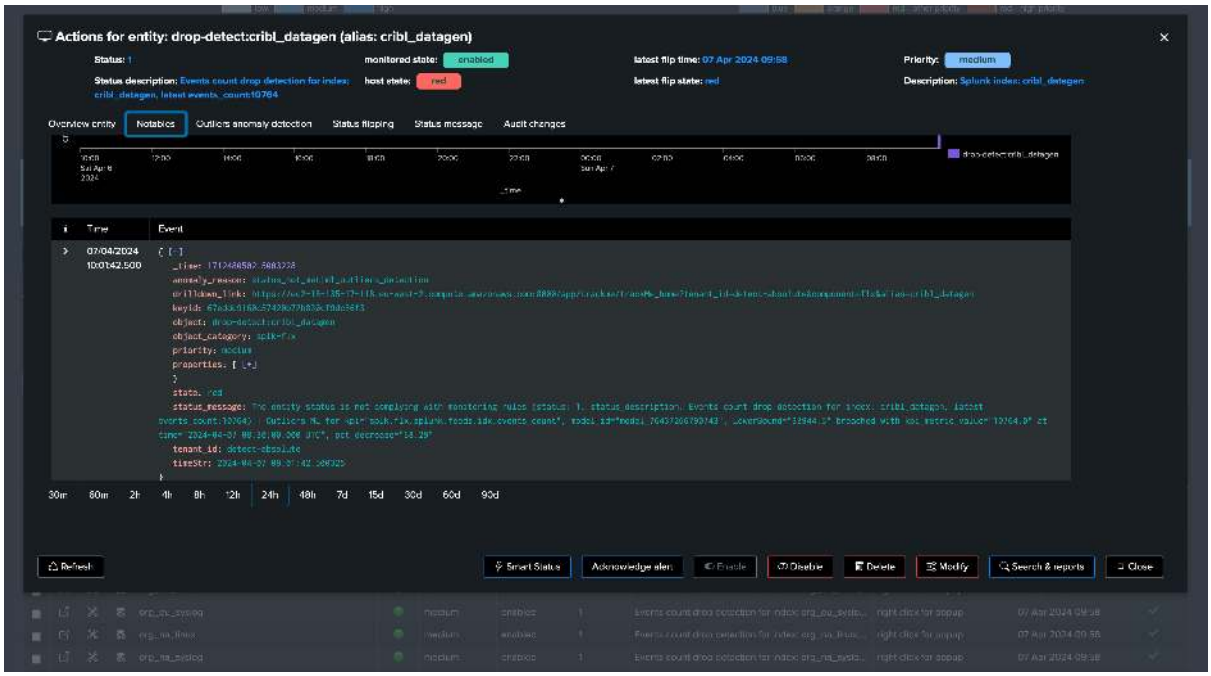
- We have a **ready and operational automated powerful framework** to continuously track and detect abnormal volume drop in Splunk feeds per index.
- TrackMe will automatically train and maintain Machine Learning models for each entity, and apply Anomaly Detection against these models.
- Very little fine-tuning is really required, so you can focus on the core of the use case.
- You can also fine tune the most important and critical feeds of your environments easily in TrackMe.

Screenshots of an entity triggering on Machine Learning detection due to abnormal events count drop:









9.4.4 Analyse log messages logging level to detect behaviour anomalies using TrackMe's Flex Object and Machine Learning Anomaly Detection

Detecting increasing volumes, globally and per index with TrackMe

- This TrackMe whitepaper tutorial demonstrates how you can leverage TrackMe to detect **Splunk instances and deployments abnormal conditions by analysing log messages and their logging level through the lens of Machine Learning Outliers detection**.
- This use case tracks Splunk internal logs and their logging level, turning these into KPIs and submitting them to TrackMe's Machine Learning Outliers detection engine.
- The objective is then to detect abnormal trends, such as an increase of the number of errors or a lack of informational messages, which would be symptomatic of potentially serious conditions affecting Splunk

instances and deployments.

- The same use philosophy can be applied to any application log messages, and detect abnormal behaviour in any application or system.

Hint

Requires TrackMe licence:

- This use case is designed using TrackMe's restricted component Flex Object.
- This component is only available to licensed users of TrackMe, and is not available with the Foundation Edition trial.

Objective: Making sense of Splunk log messages logging level and detect Splunk abnormal conditions

The basic logic is to track Splunk internal log messages through the lens of their logging level, similarly to the following Splunk search:

Let's take an example:

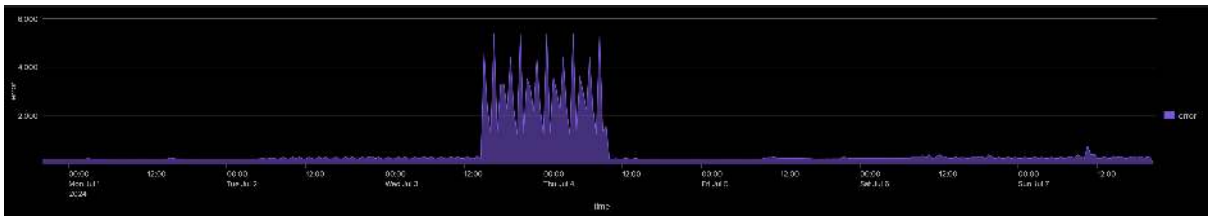
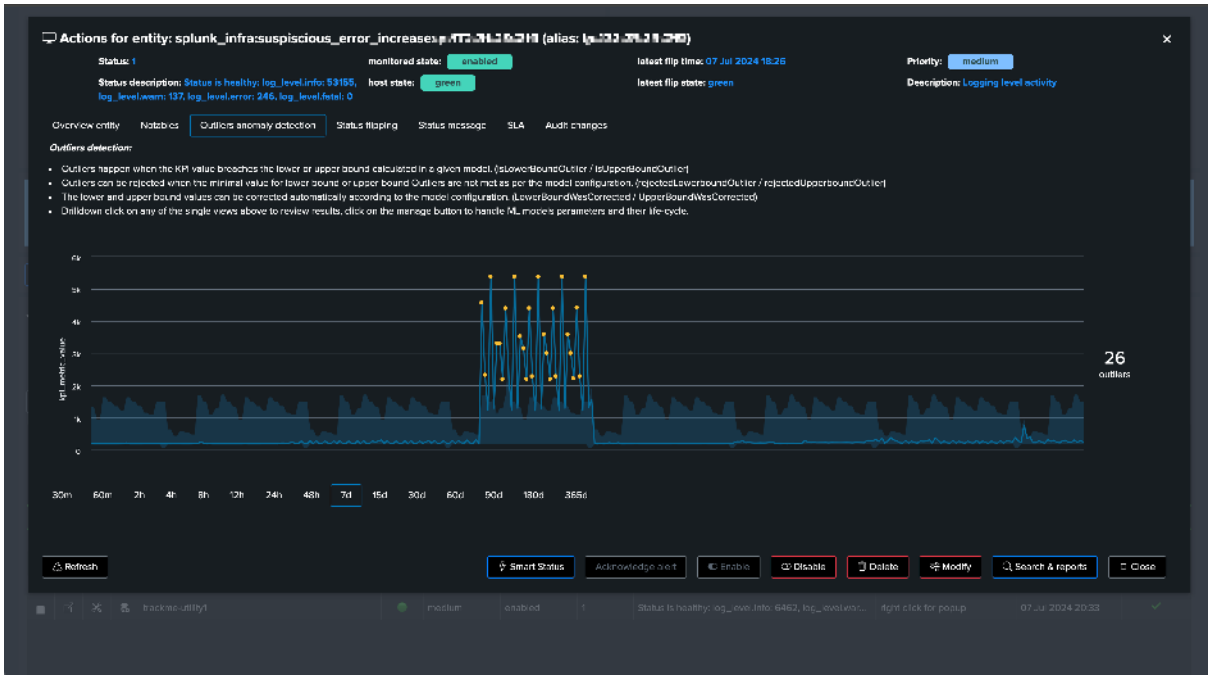
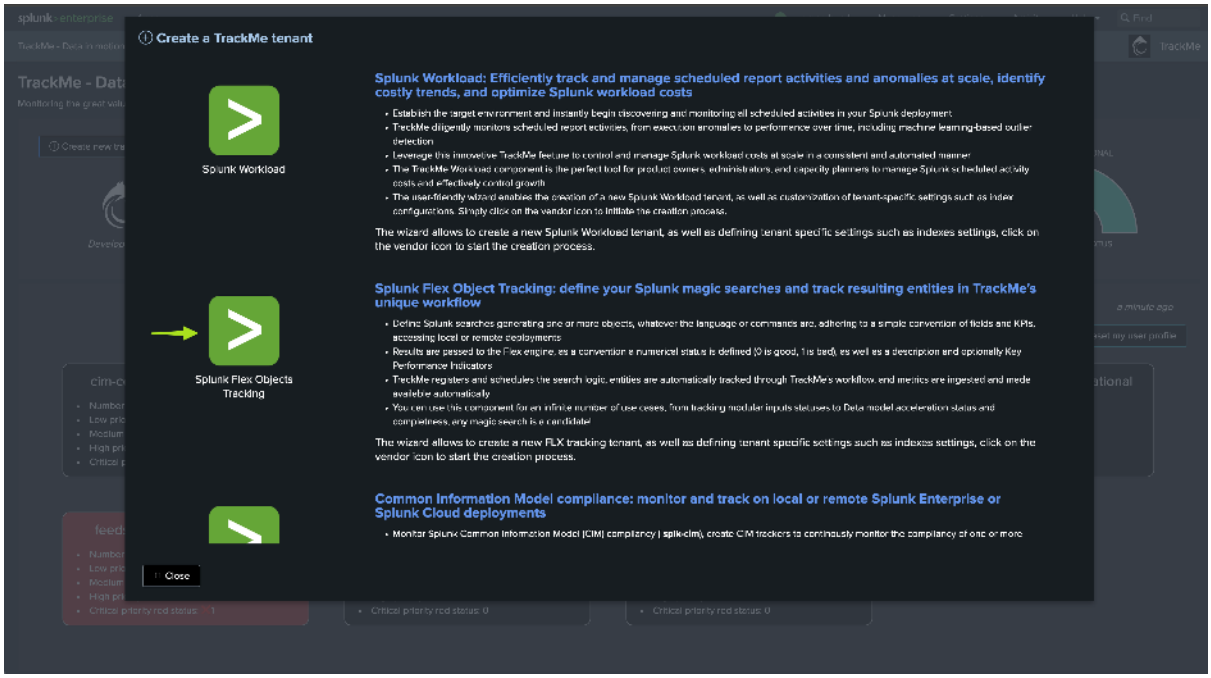
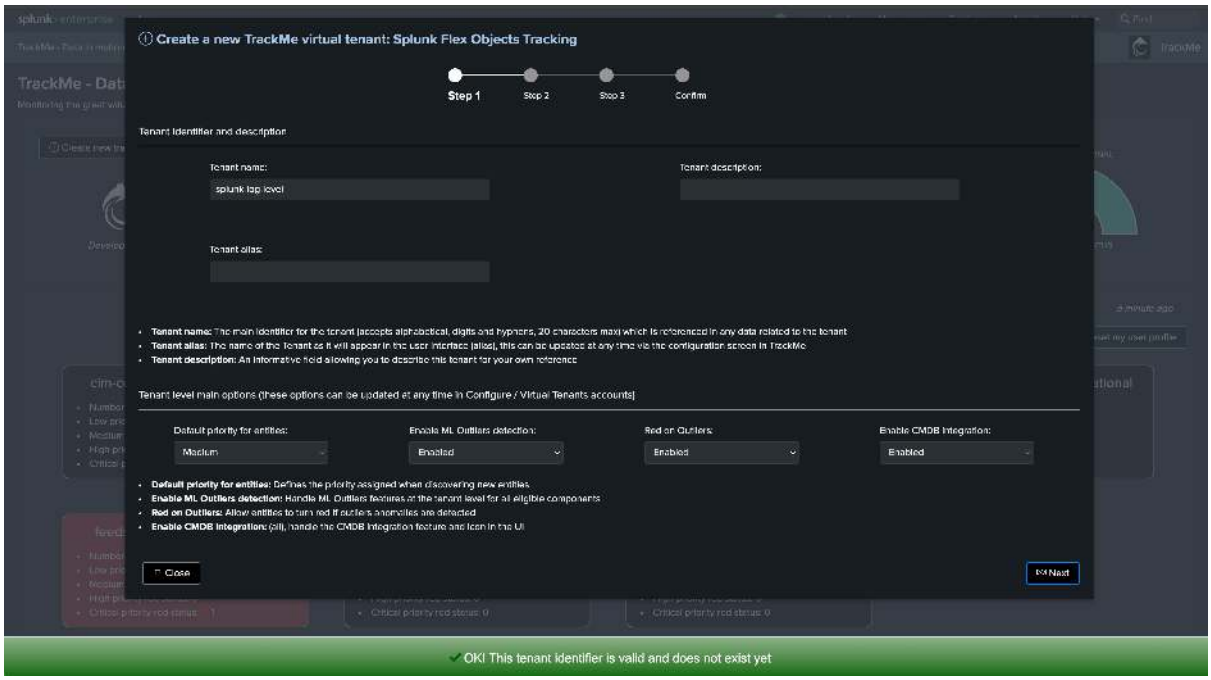
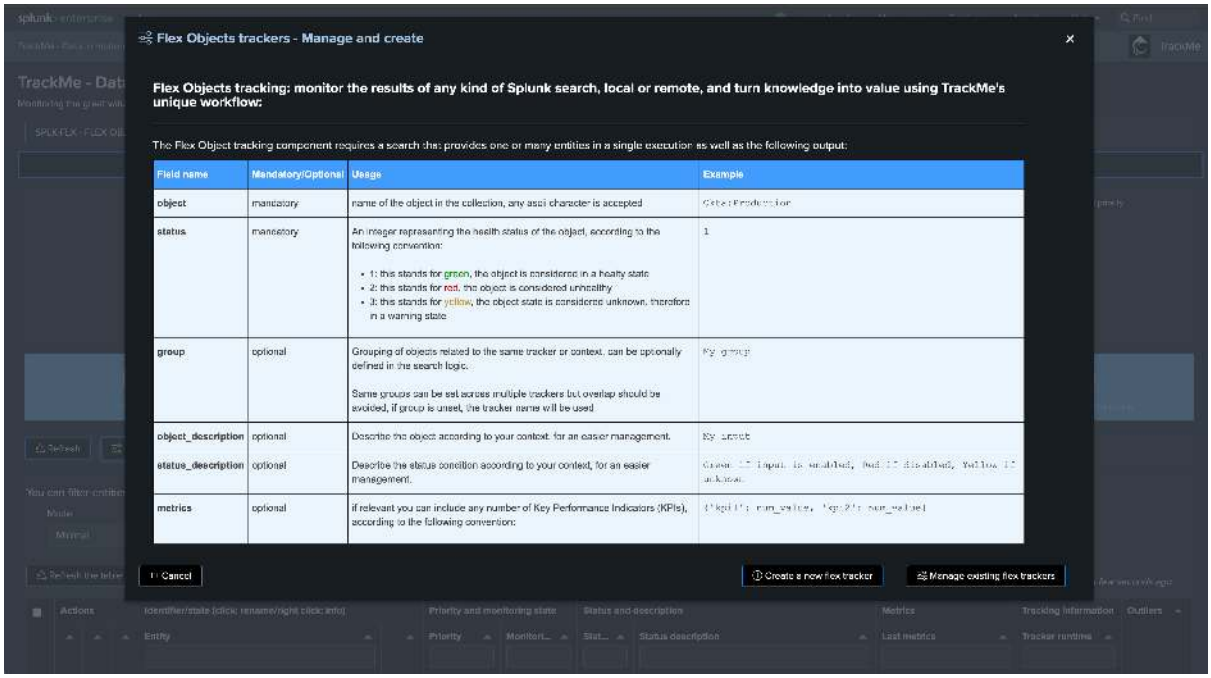
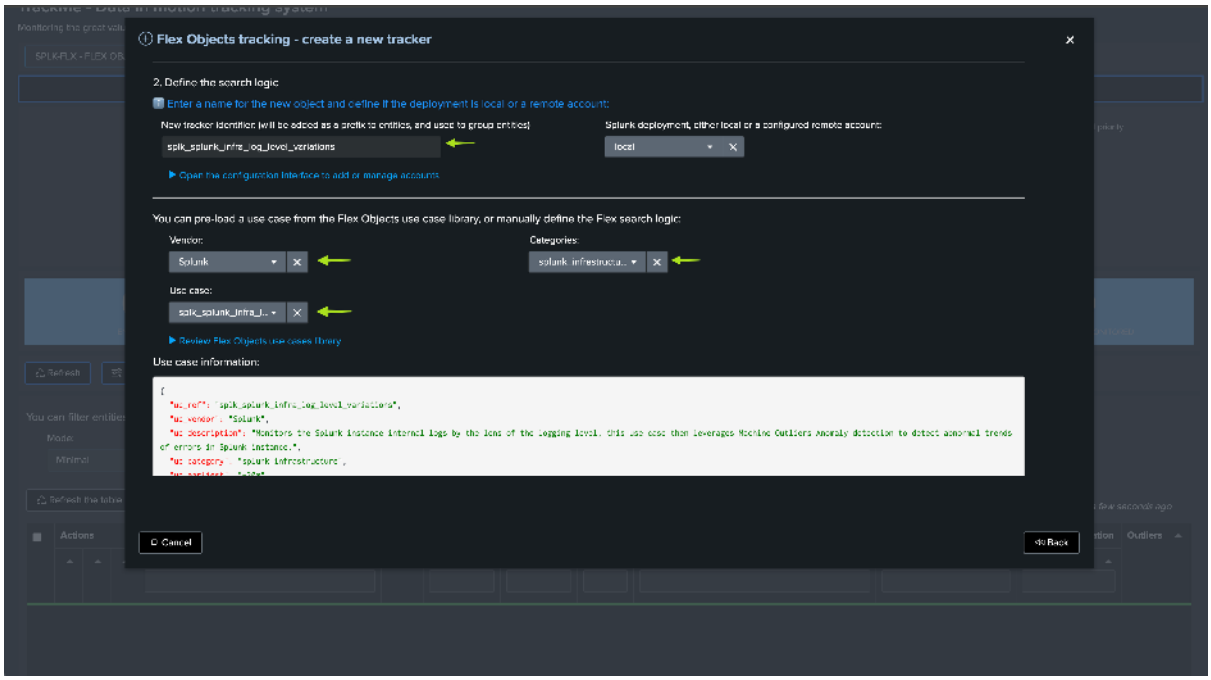
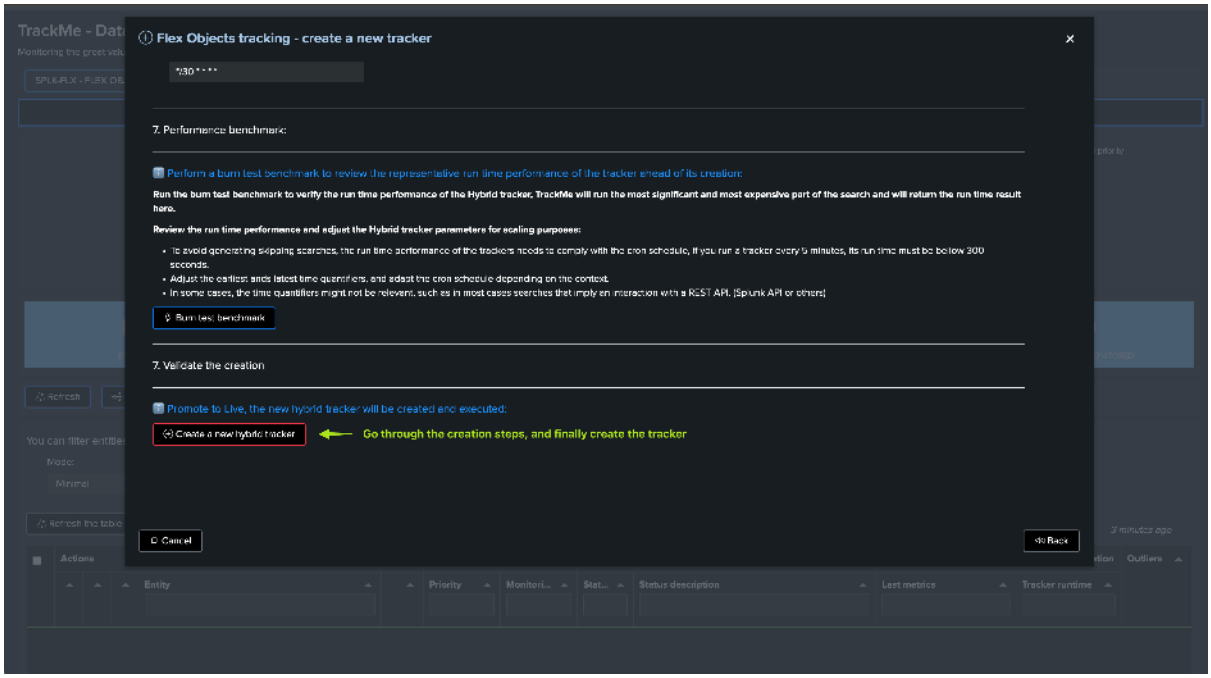
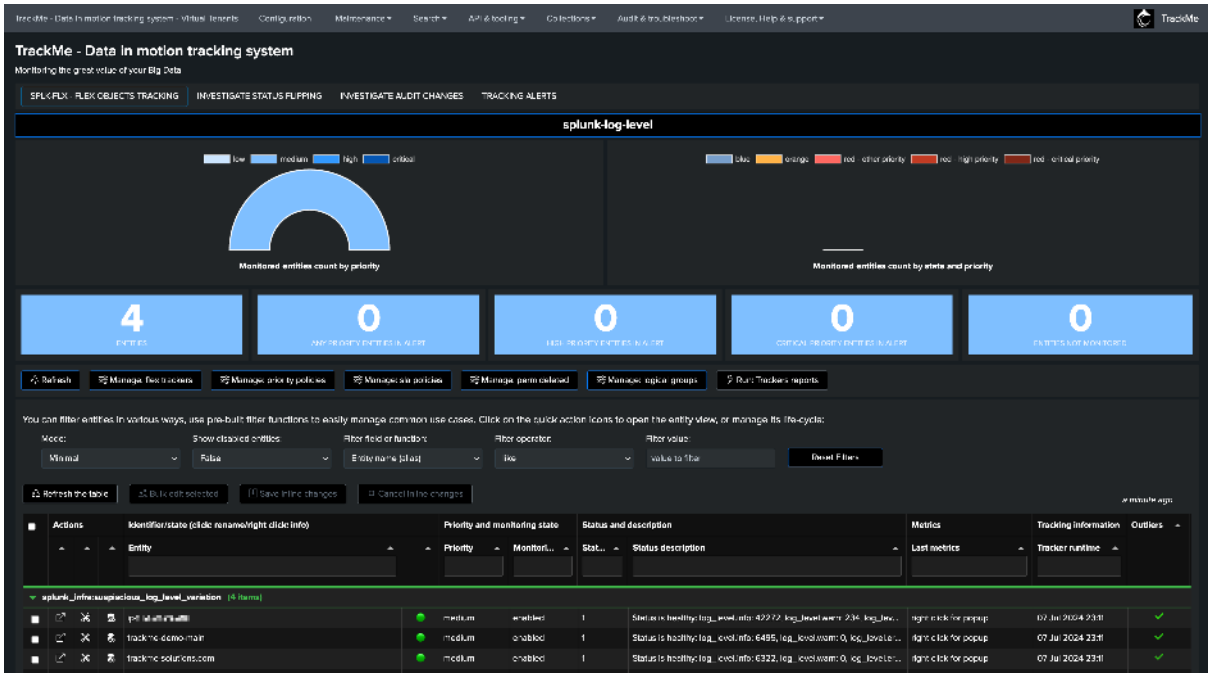
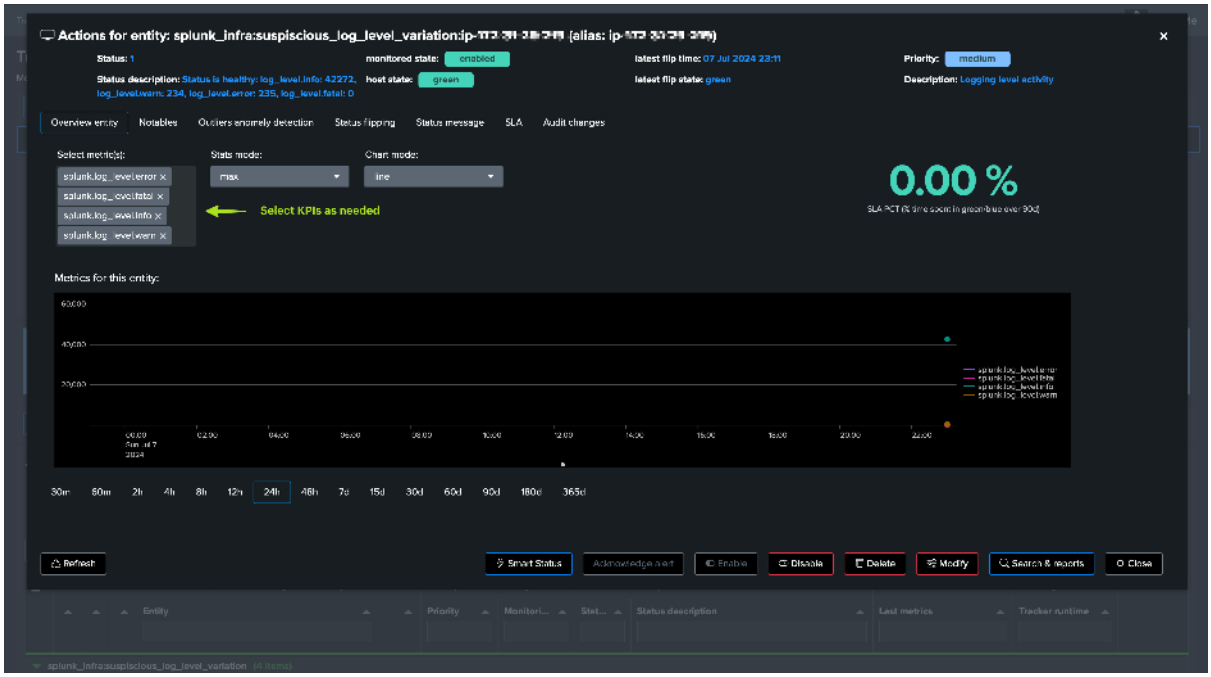
```
index=_internal host=* log_level=* (log_level=ERROR OR log_level=INFO OR␣
↪log_level=WARN OR log_level=FATAL)

``` The basis break by statement logic happens on a per host basis, you may want␣
↪to update the logic to group by tiers logic, such as grouping by indexer␣
↪cluster, Search head, Search Head Cluster and so forth ```
| bucket _time span=30m
| stats
count(eval(log_level="INFO")) as info,
count(eval(log_level="WARN")) as warn,
count(eval(log_level="ERROR")) as error,
count(eval(log_level="FATAL")) as fatal
by _time, host
| fillnull value=0

| timechart span=30m sum(info) as info, sum(warn) as warn, sum(error) as error,␣
↪sum(fatal) as fatal
```

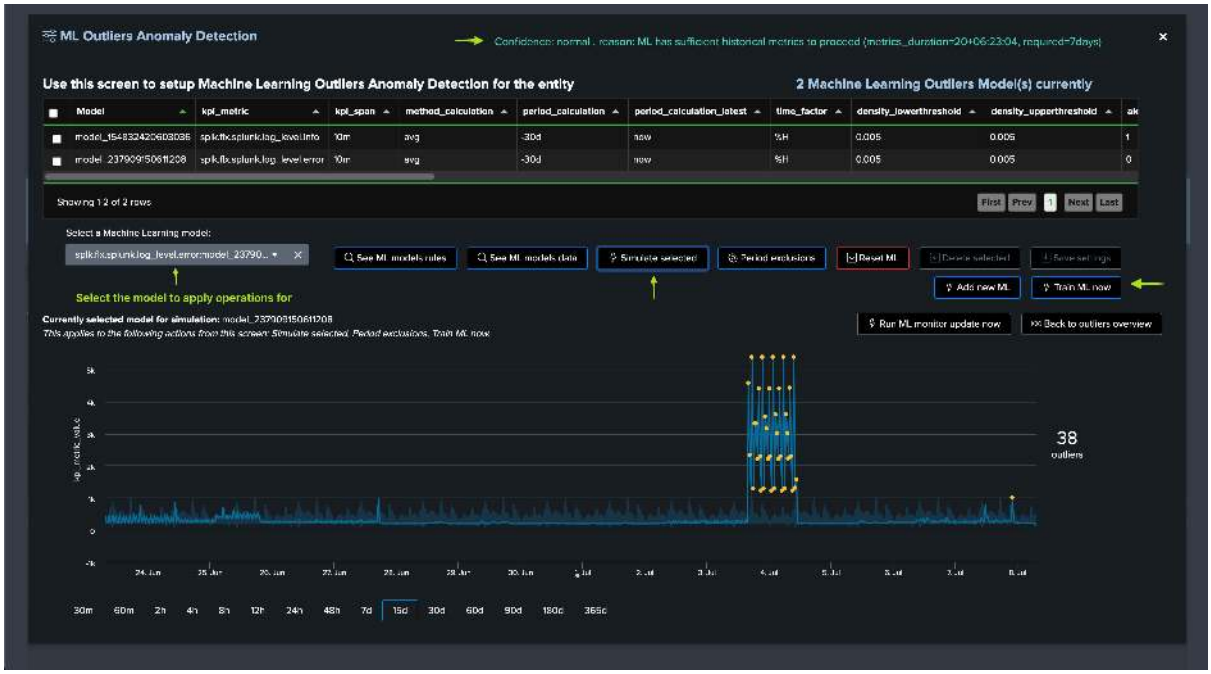
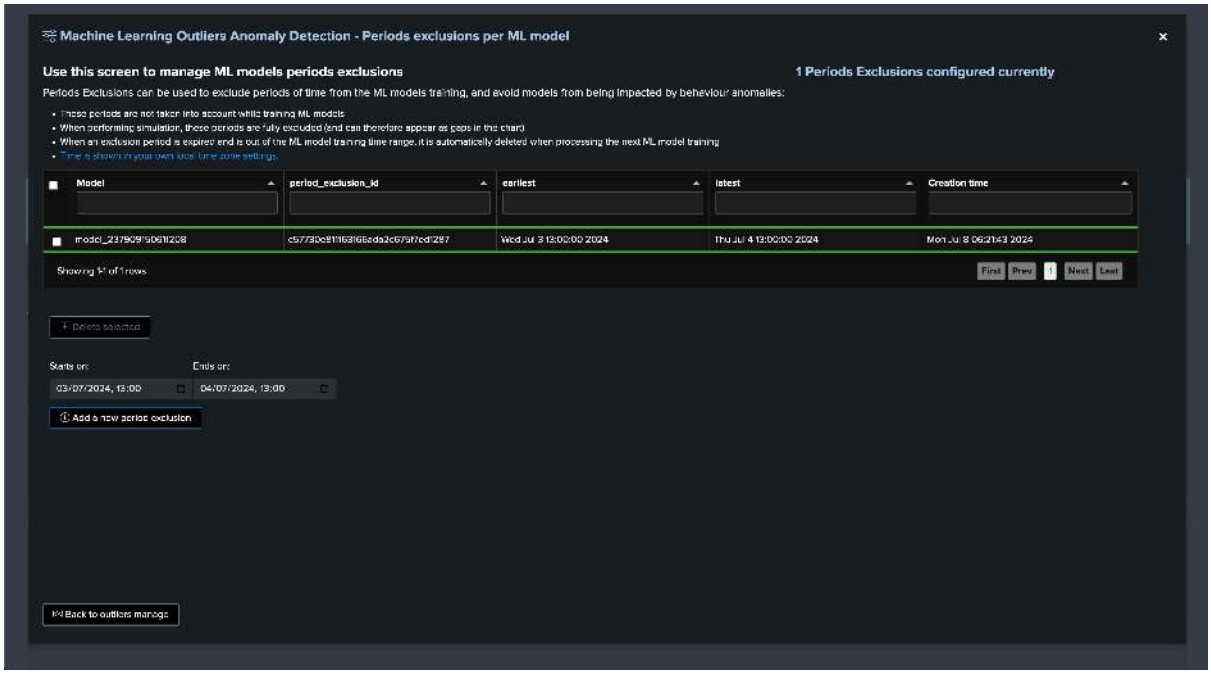
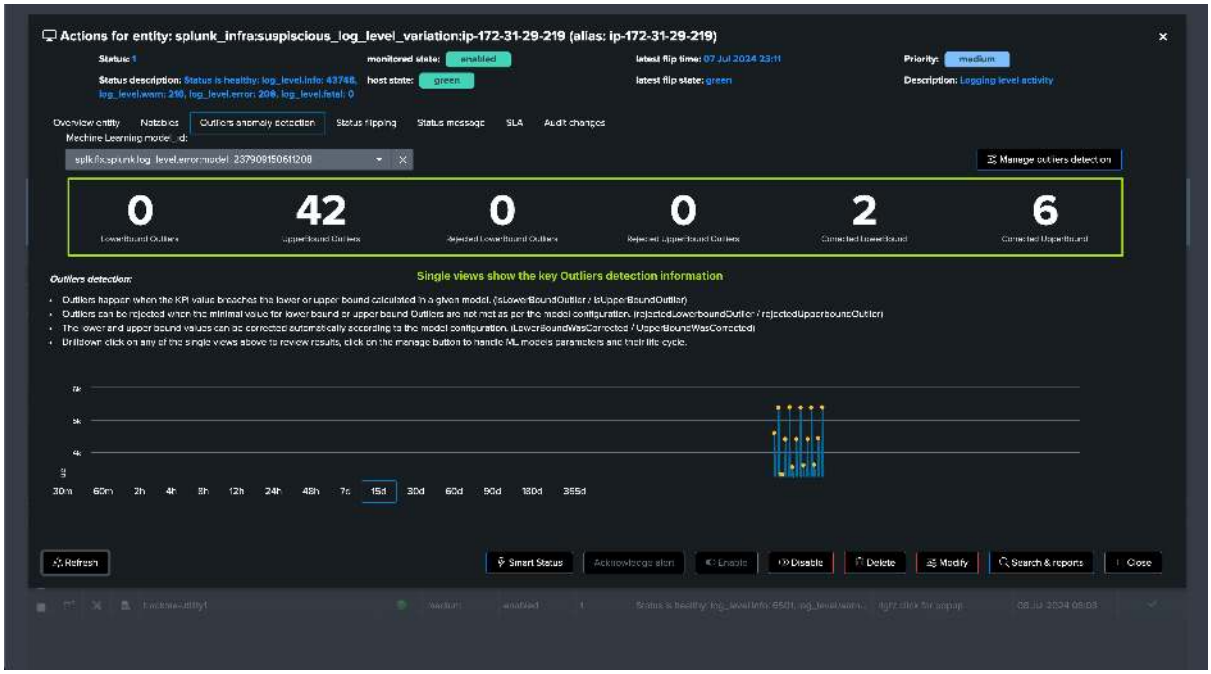
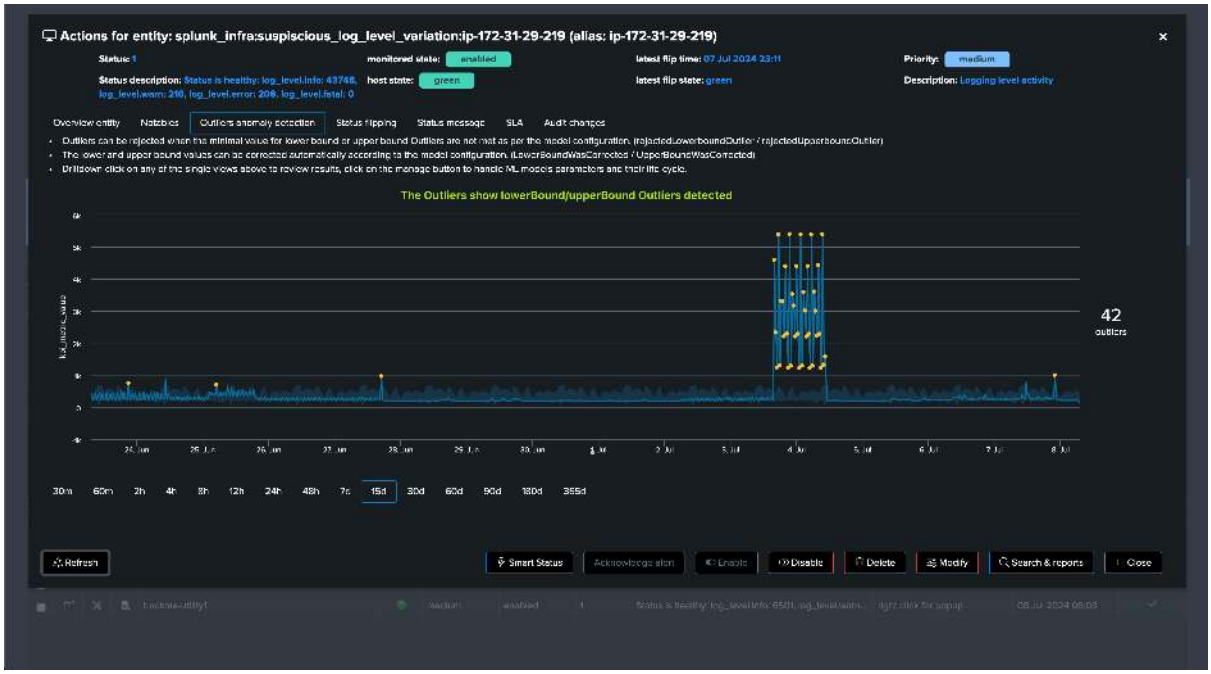
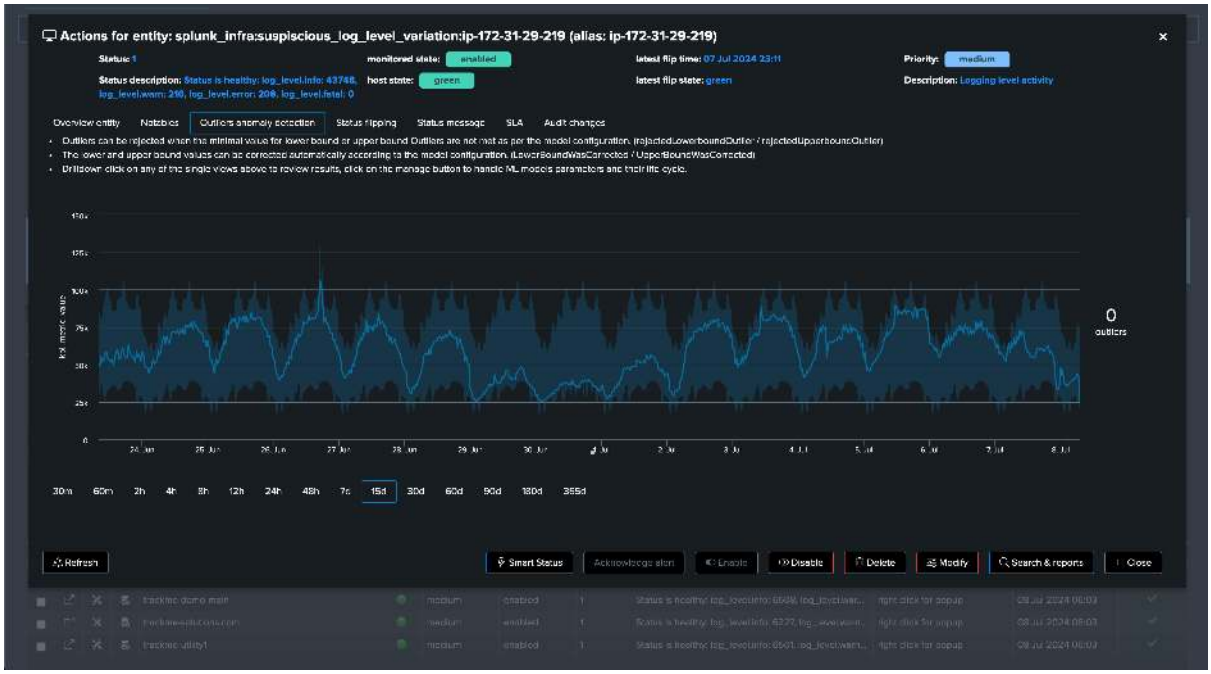
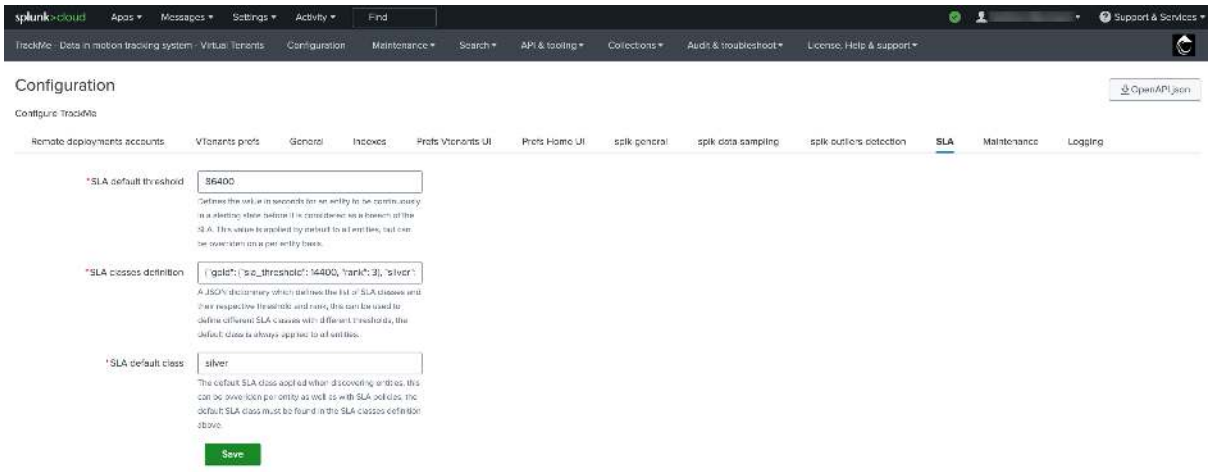
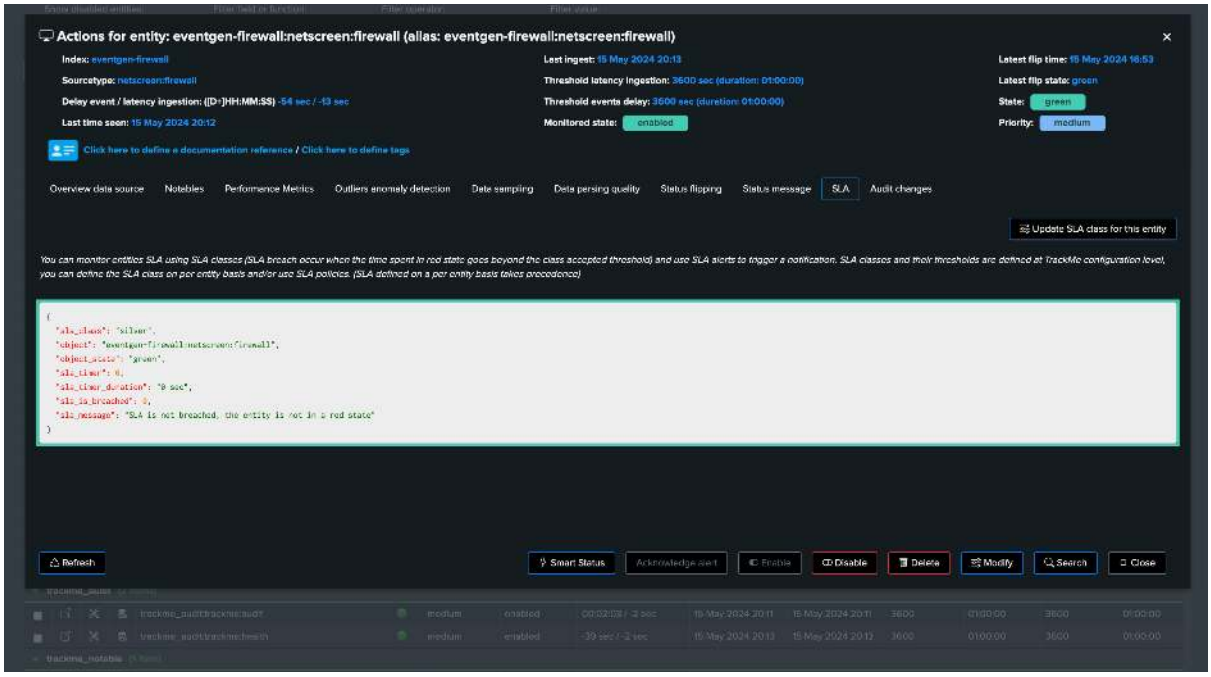
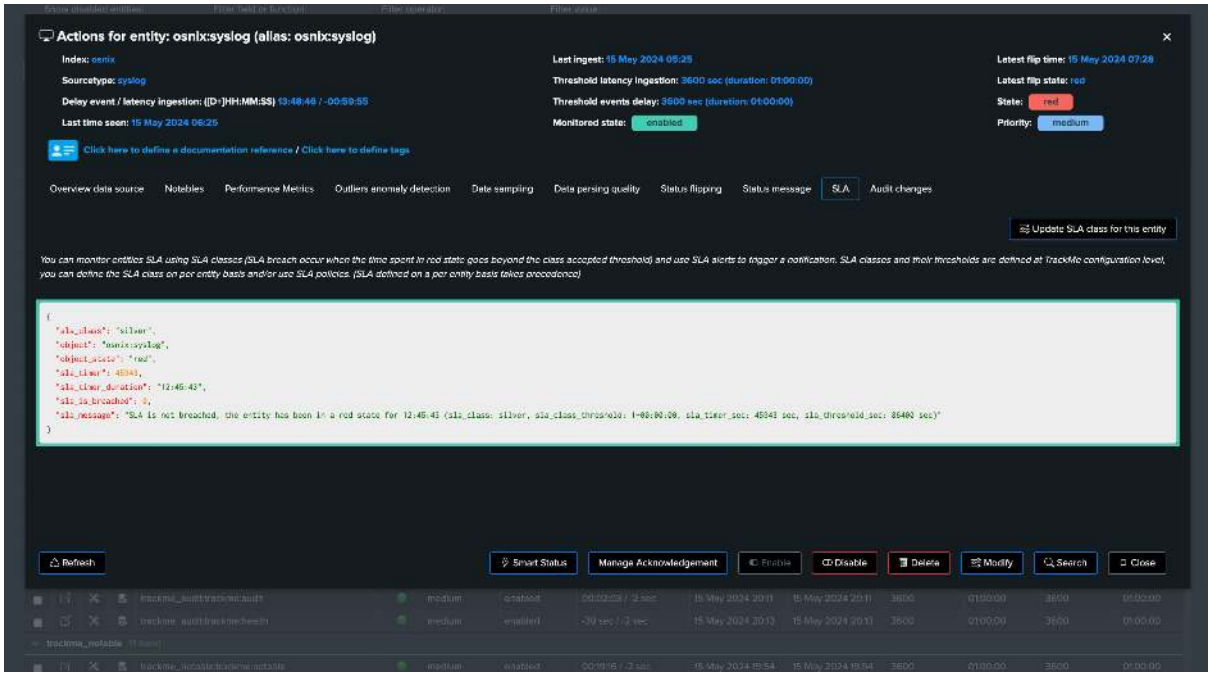
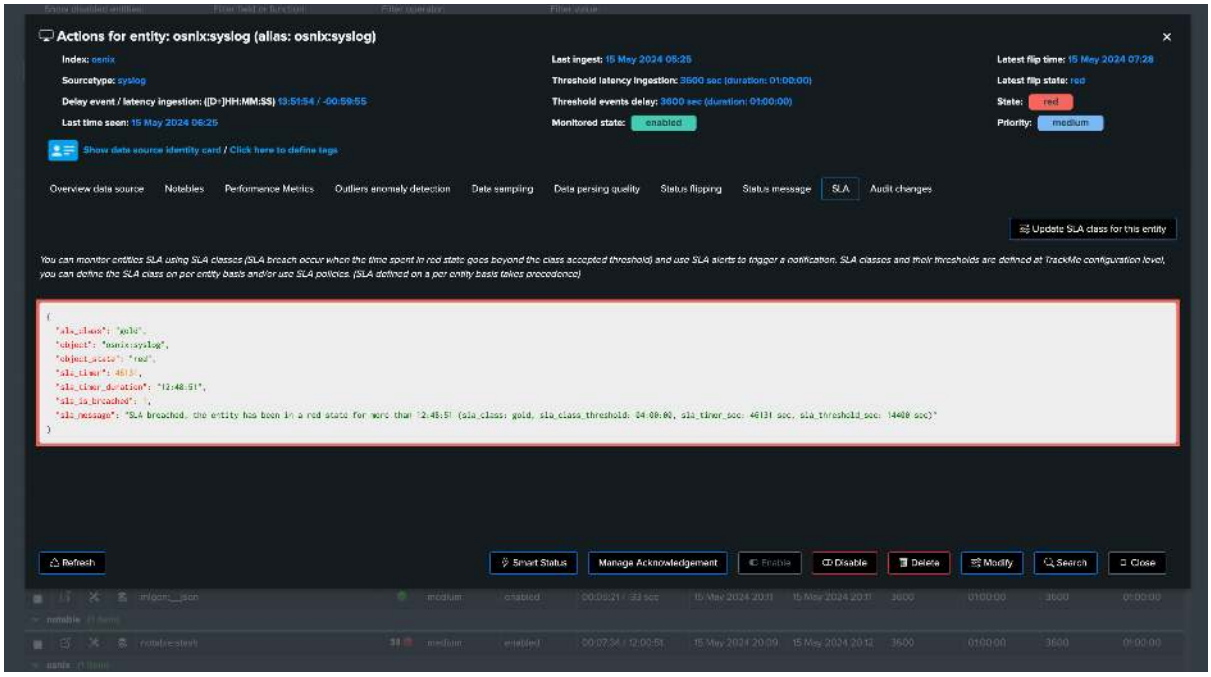
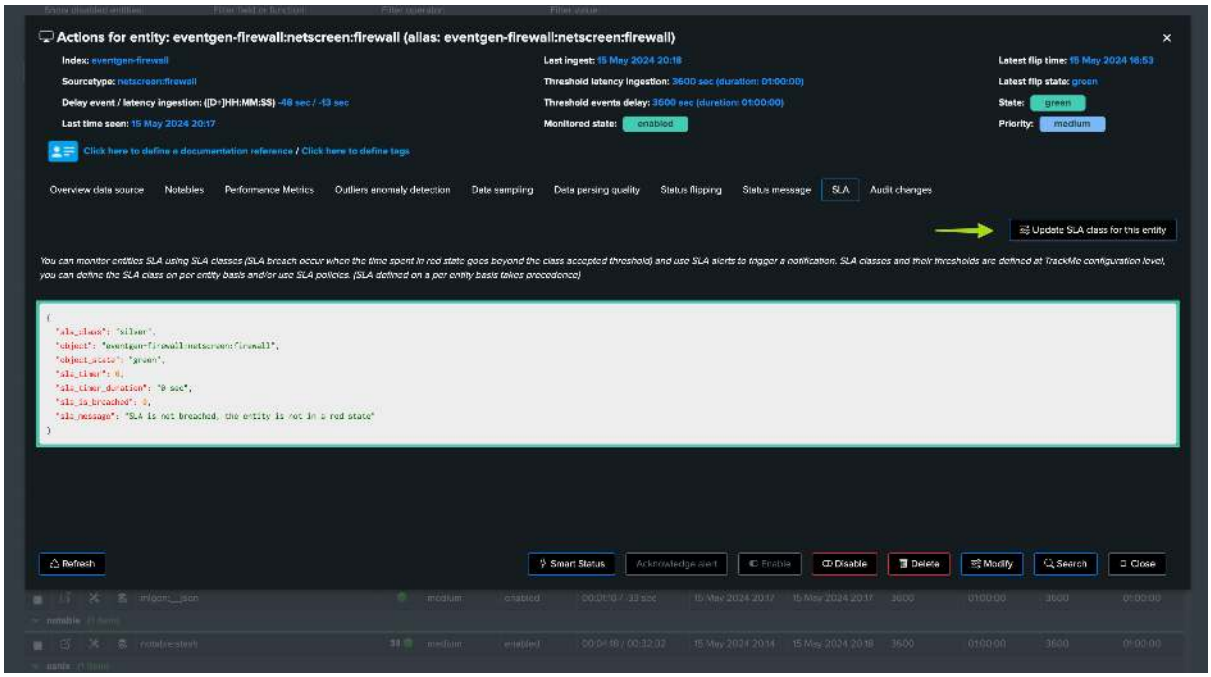
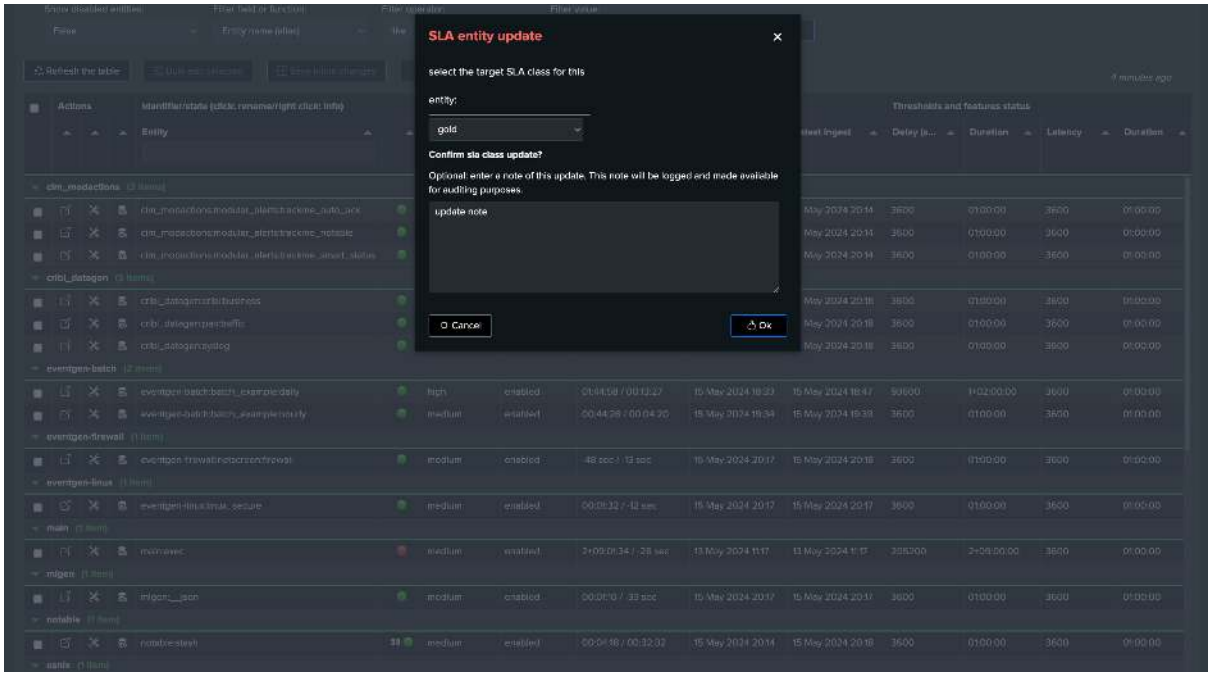
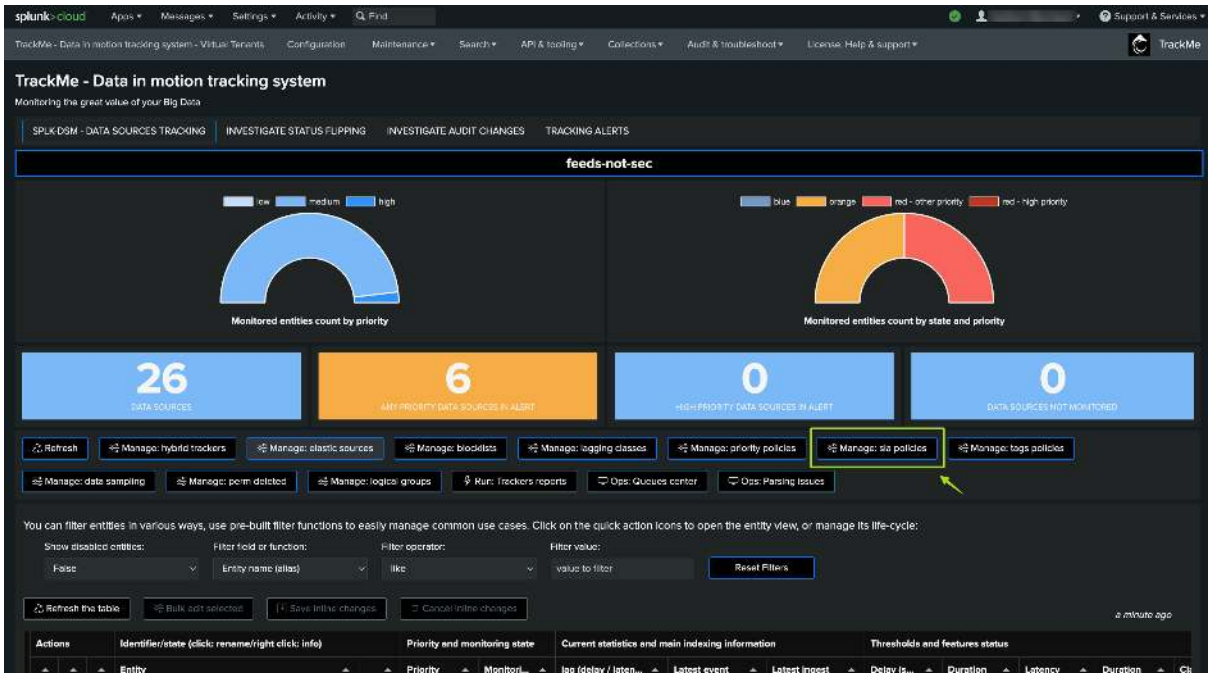
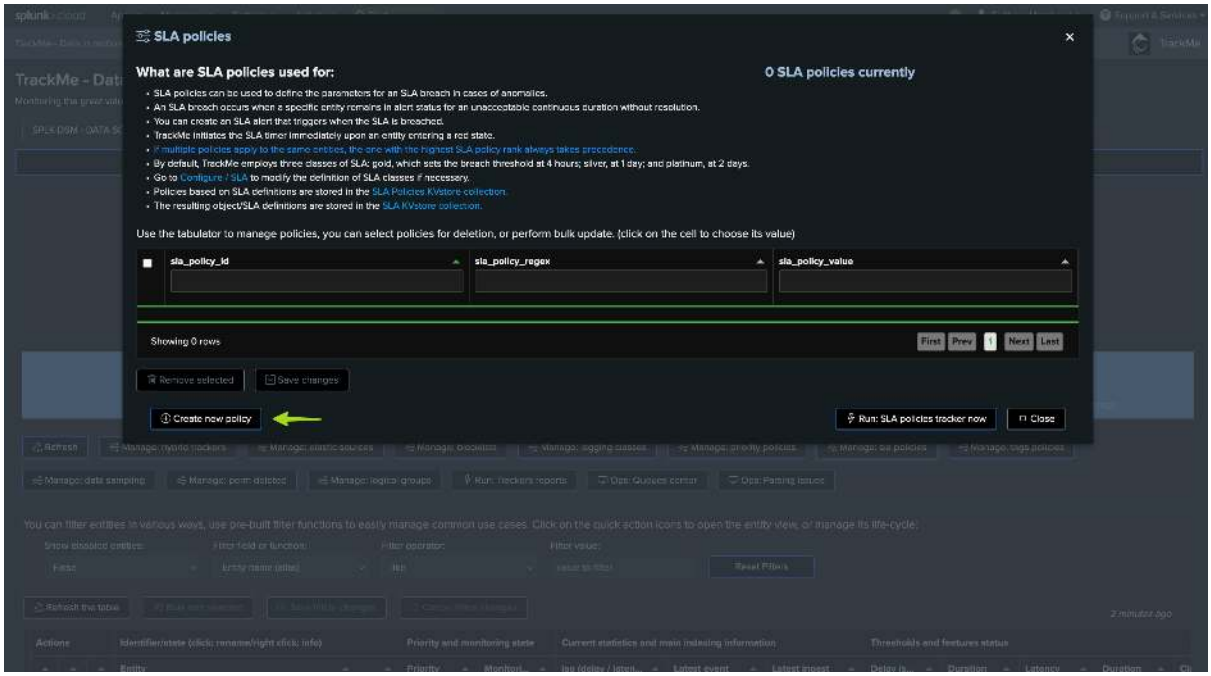
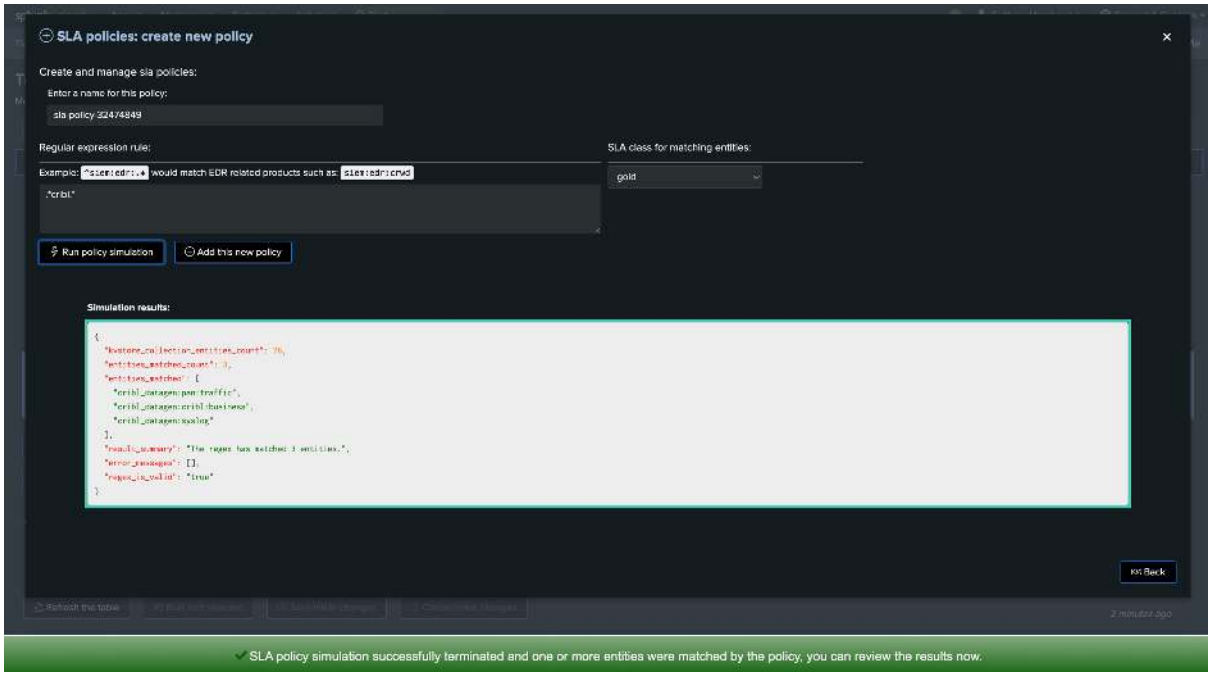
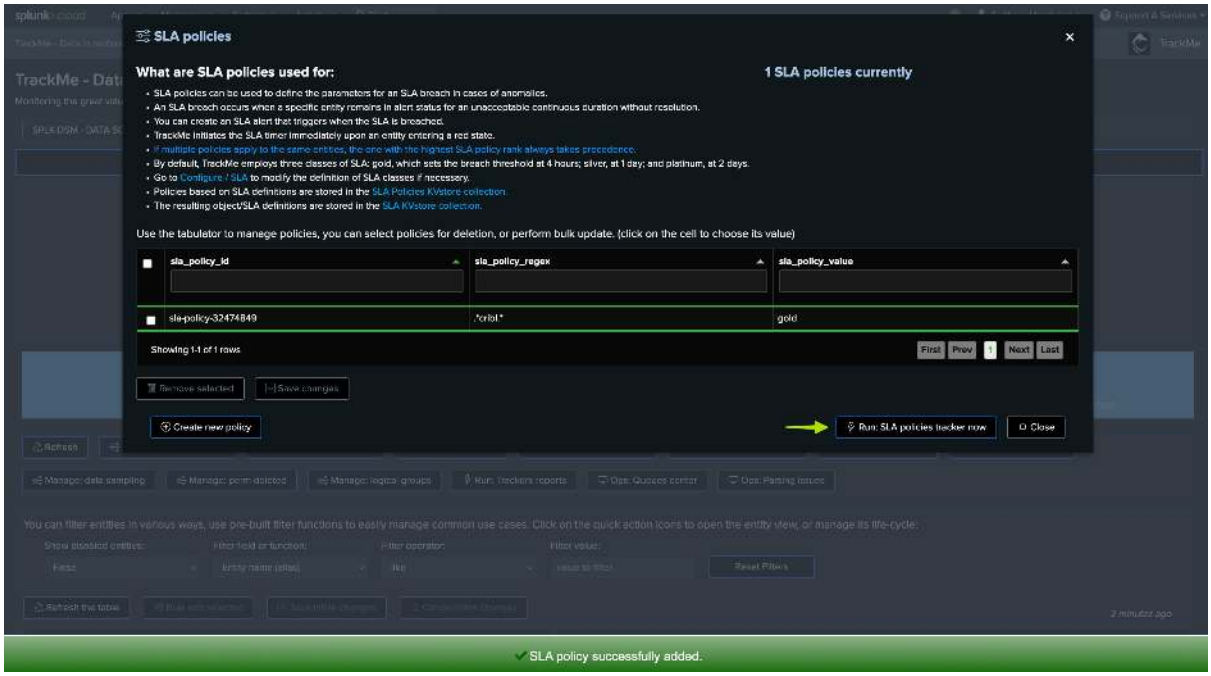
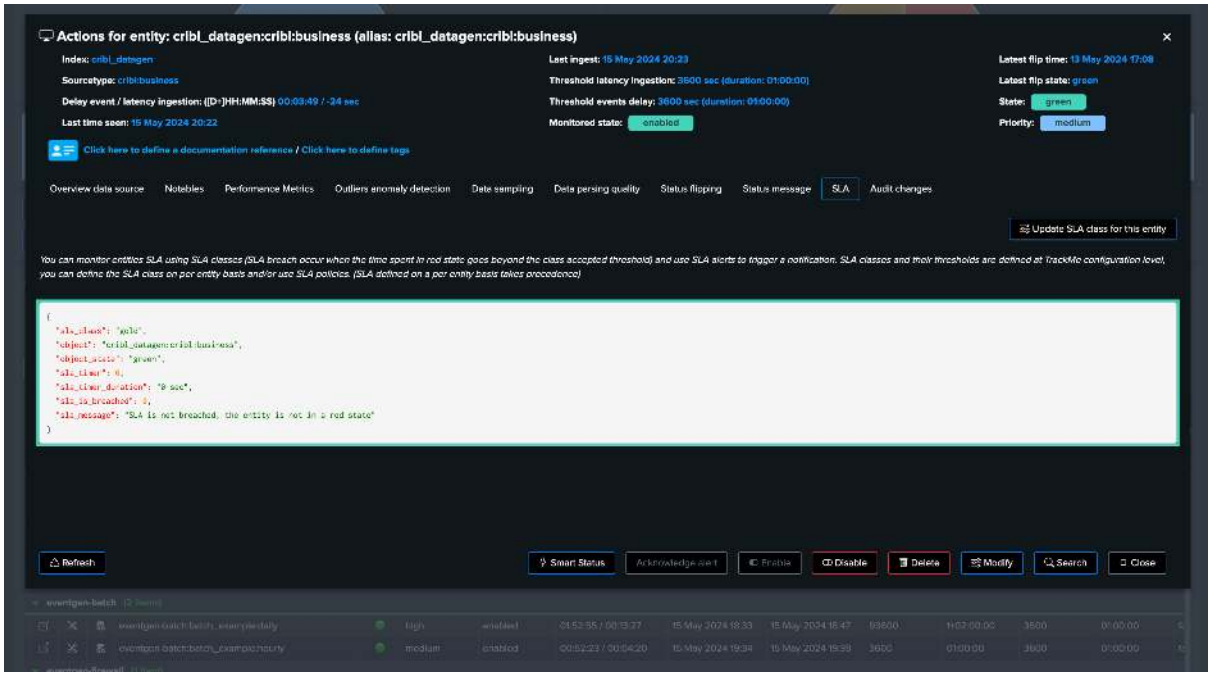
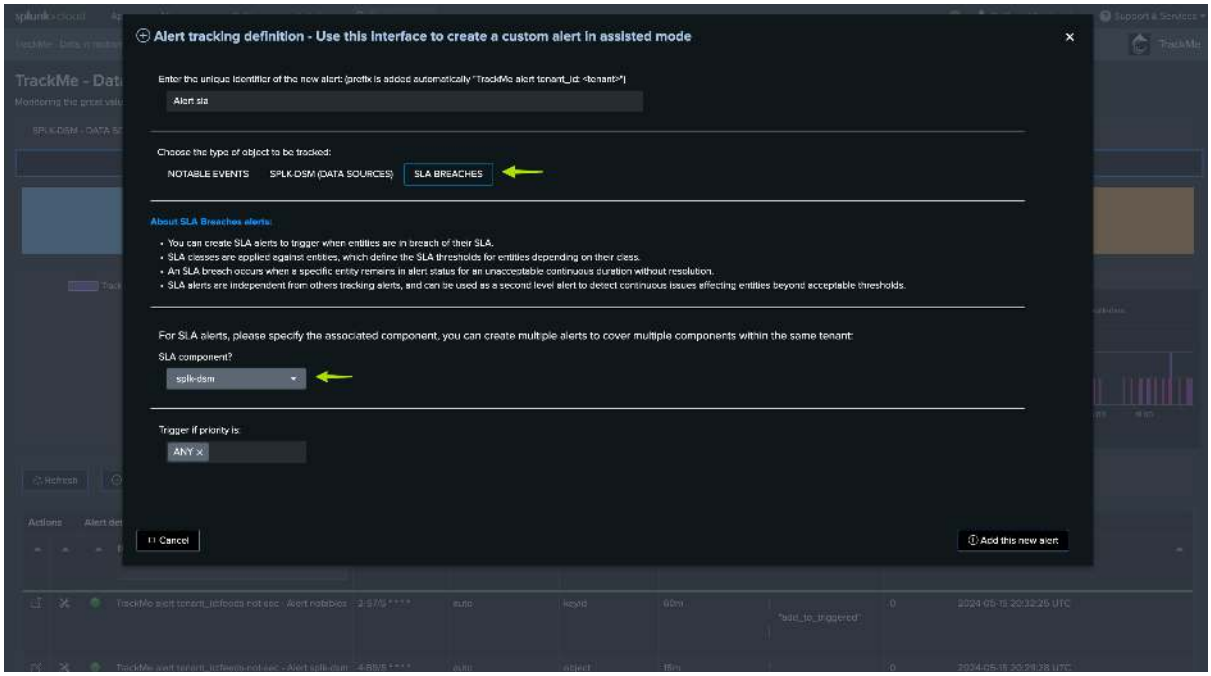
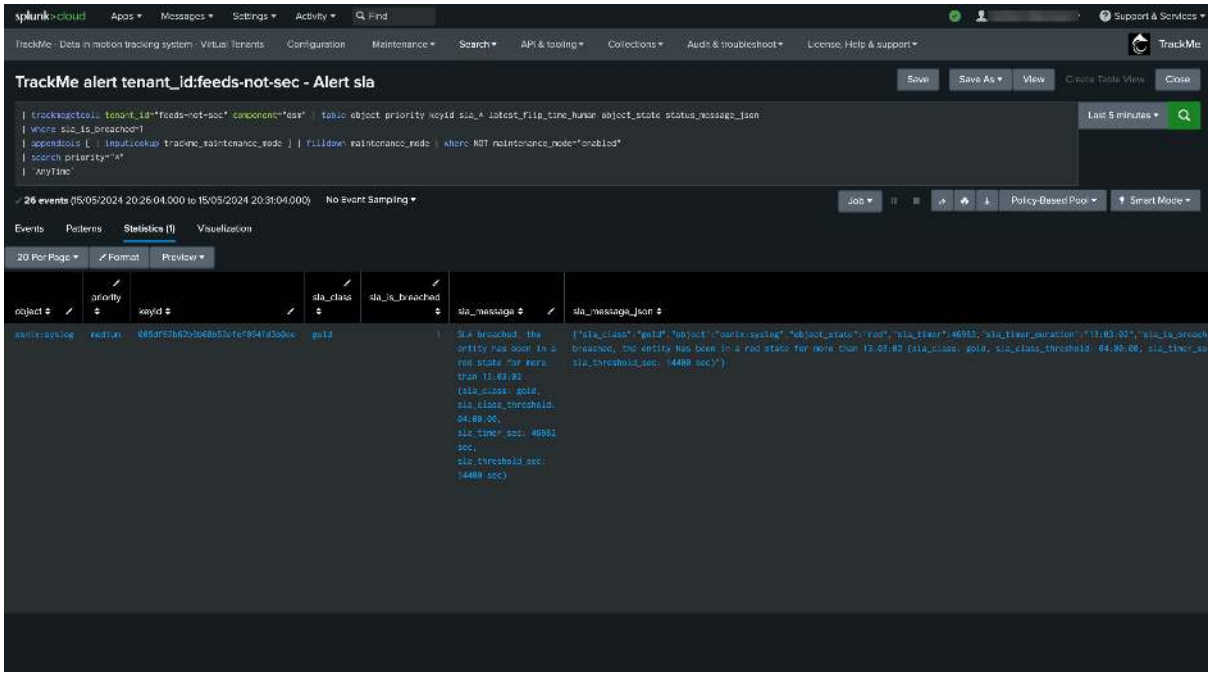
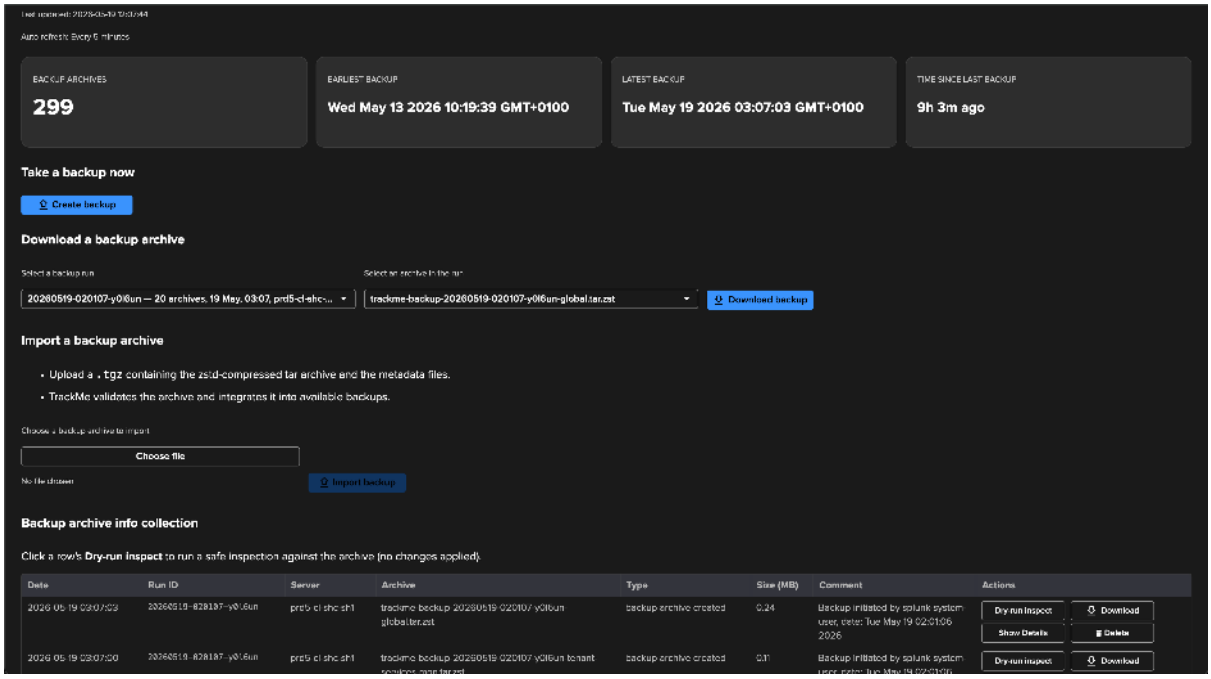
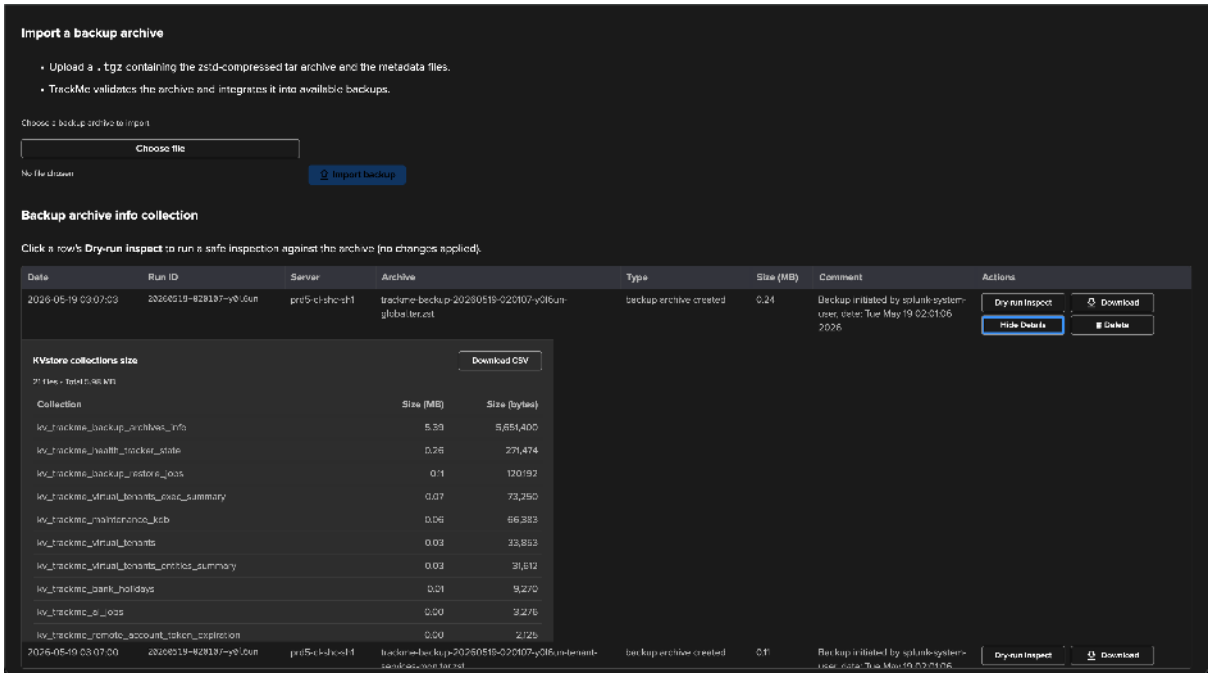
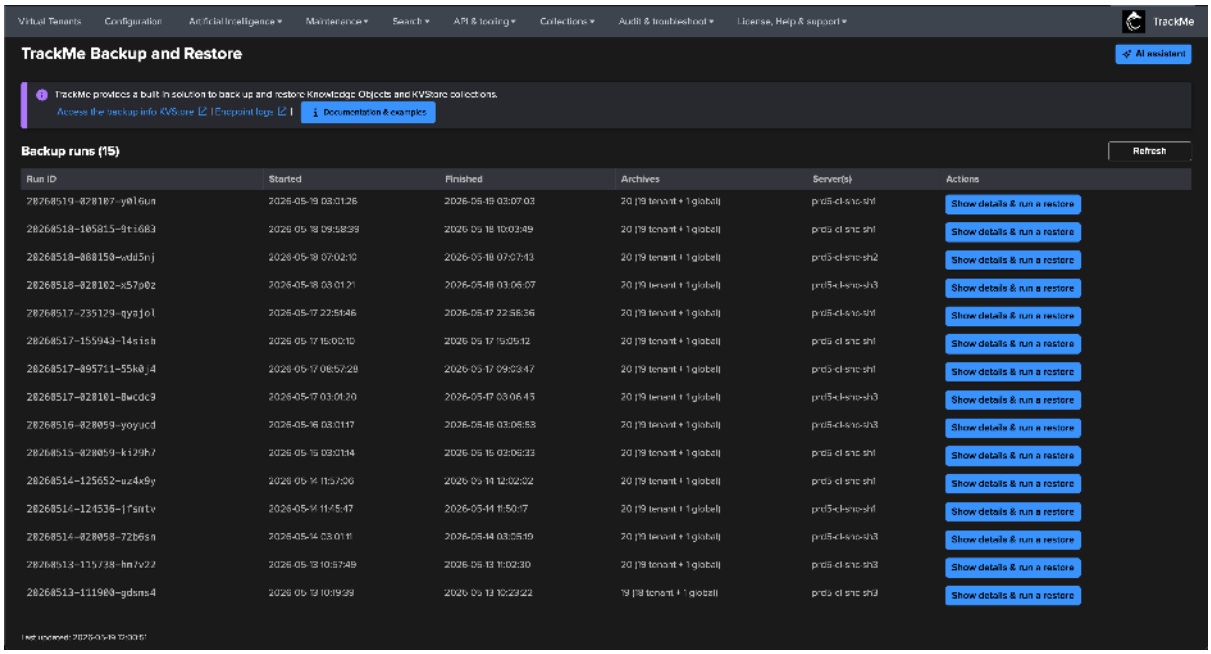
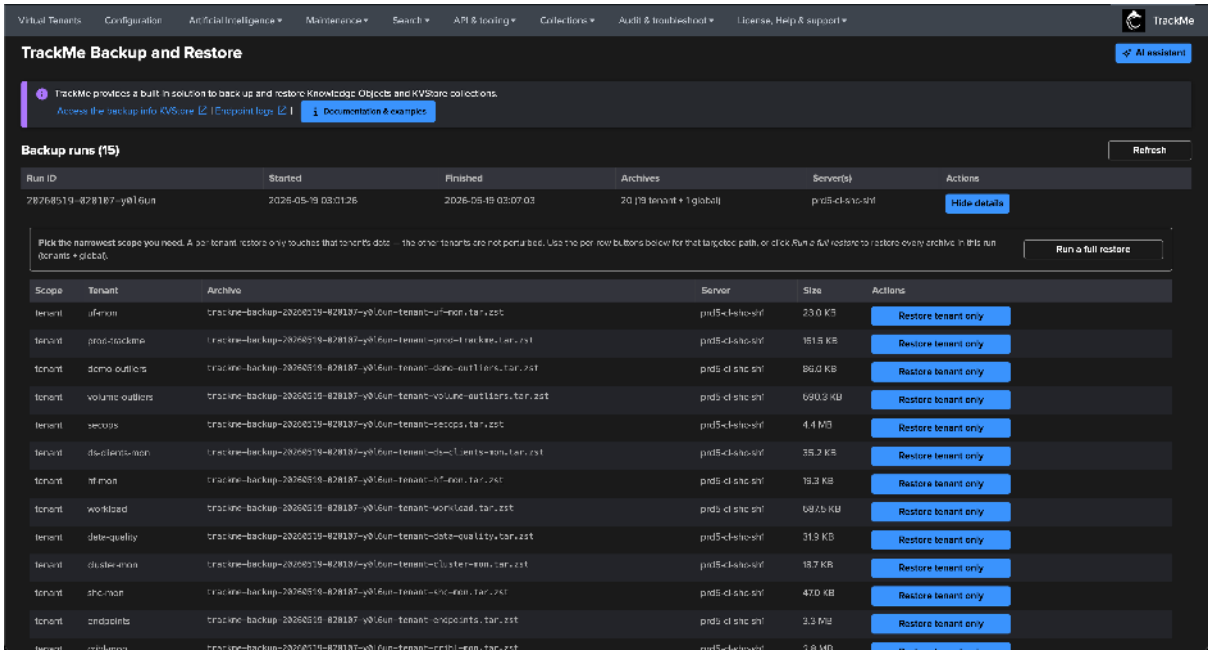
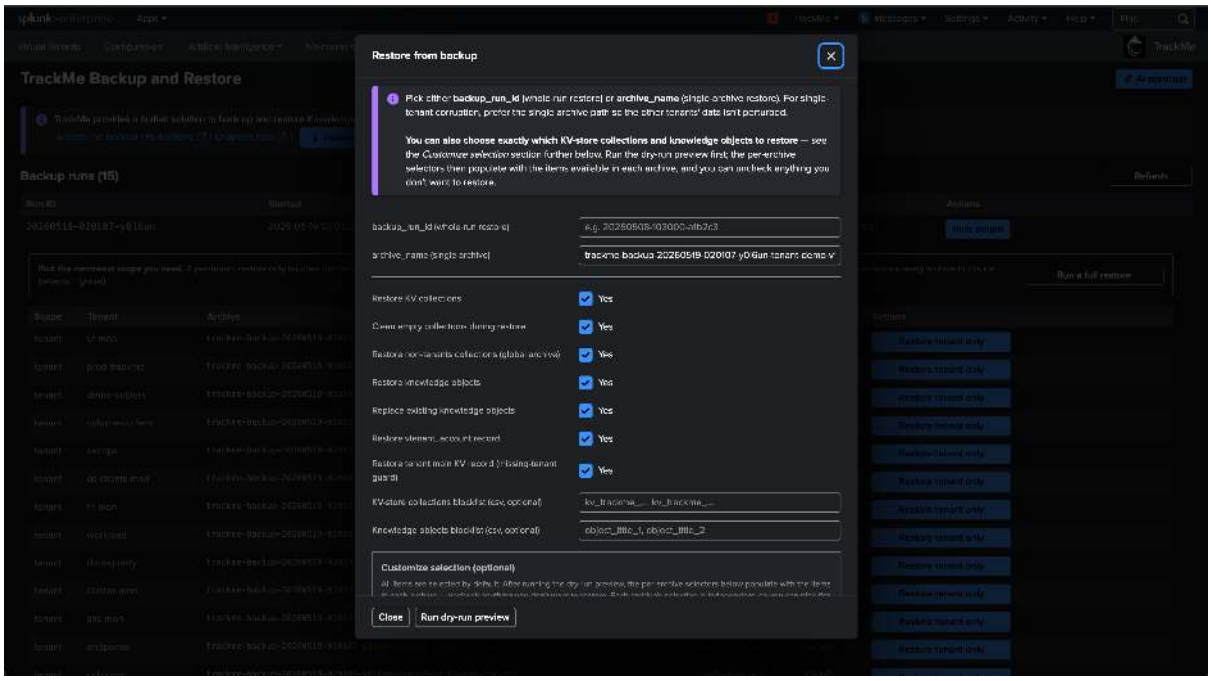
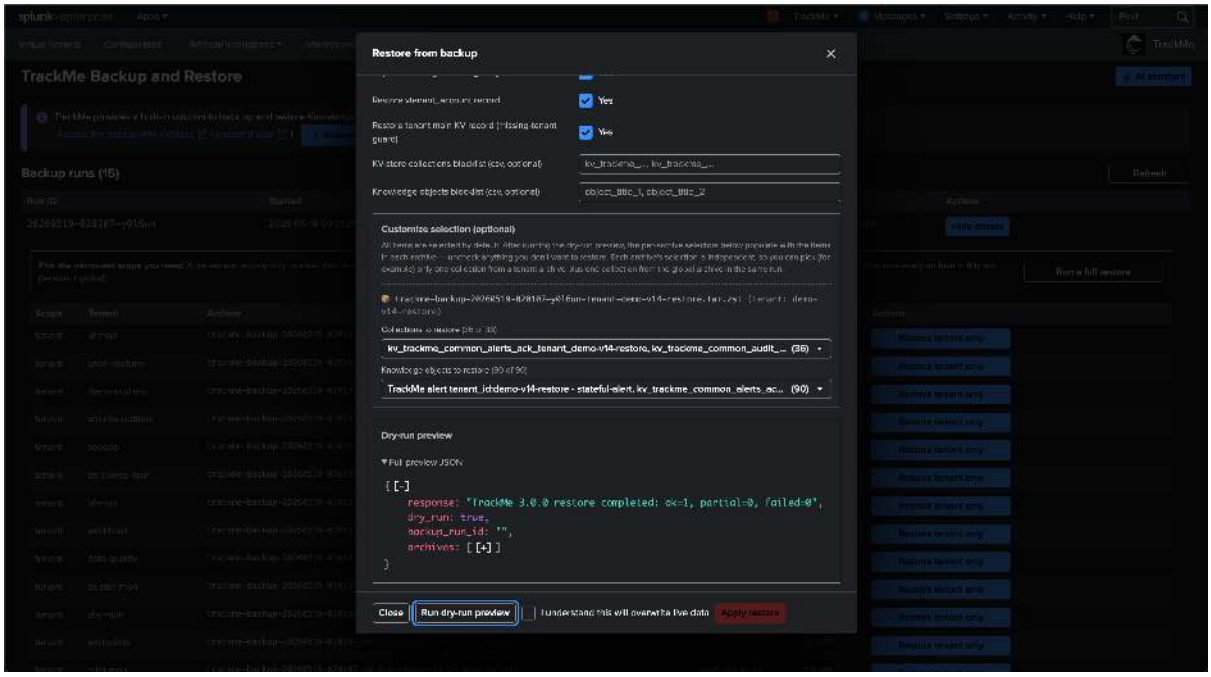
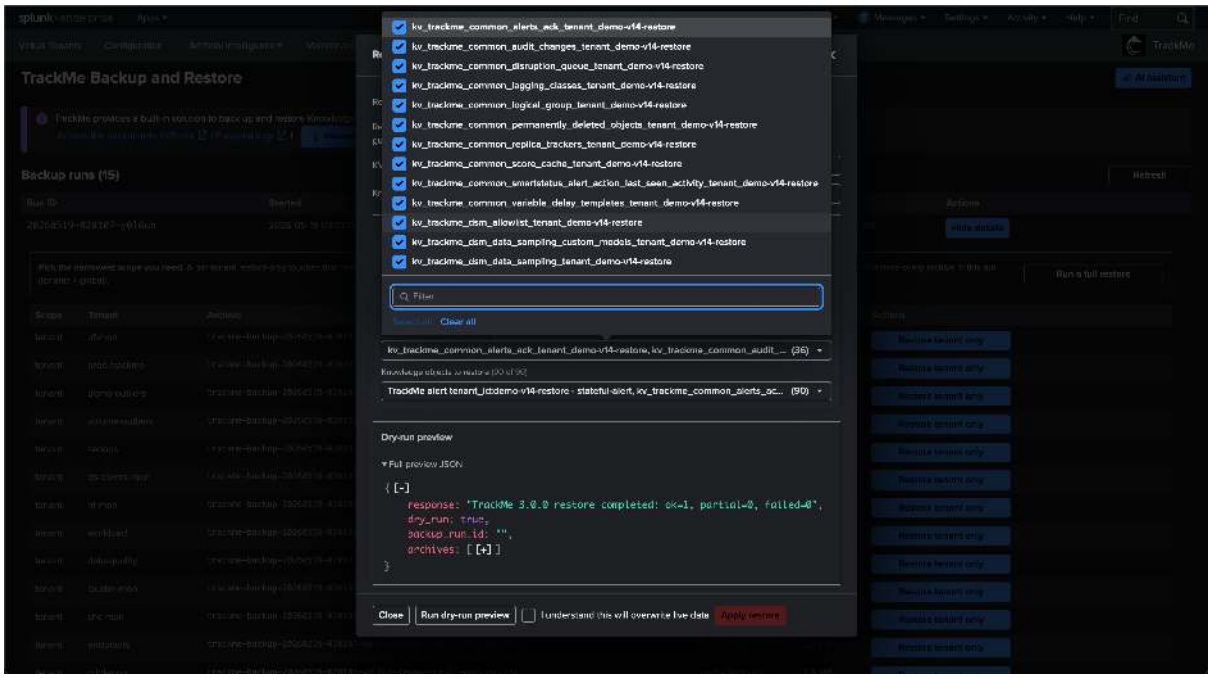
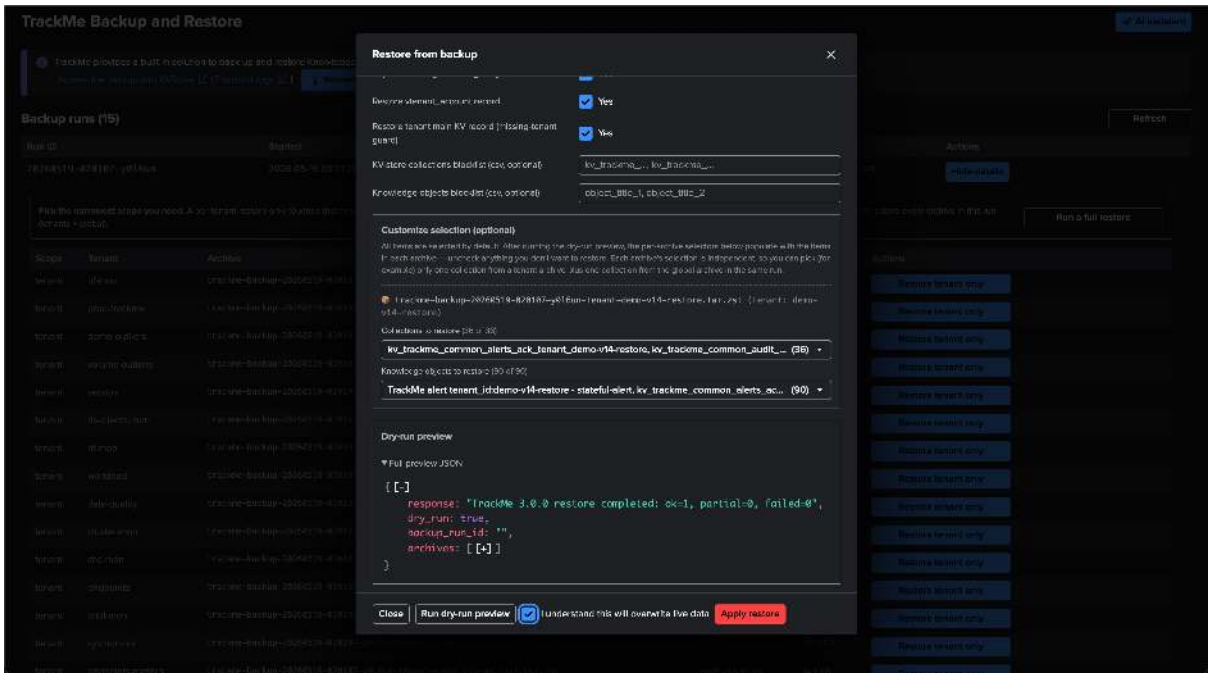
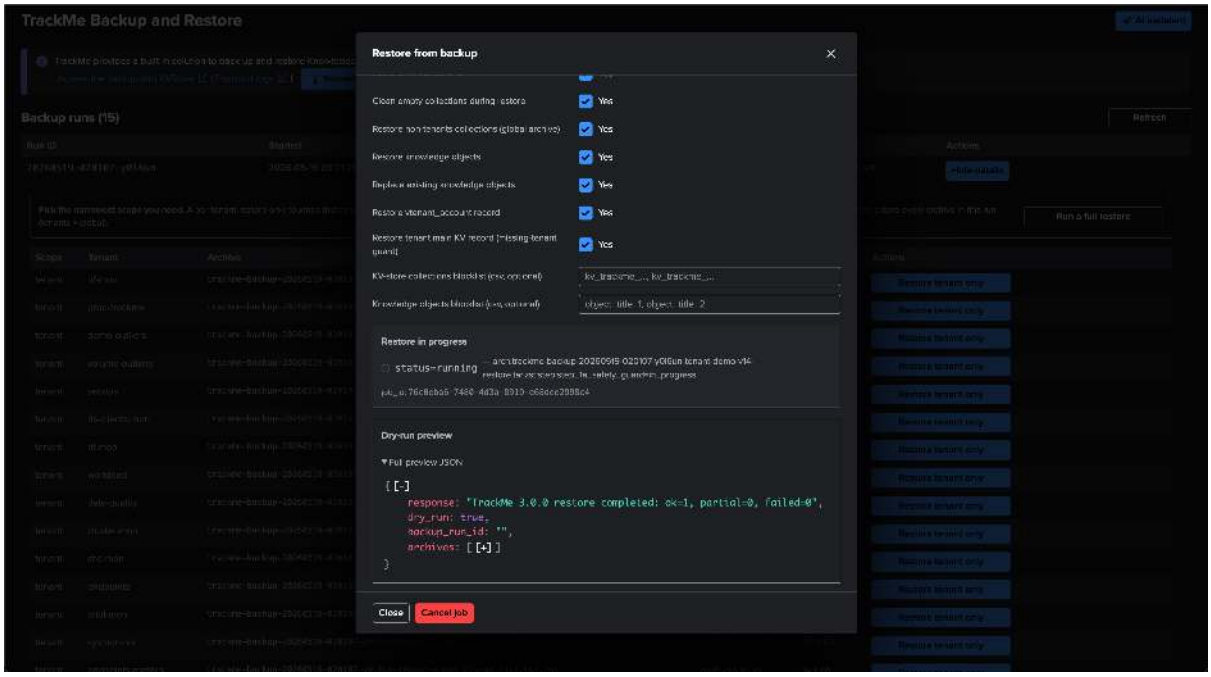
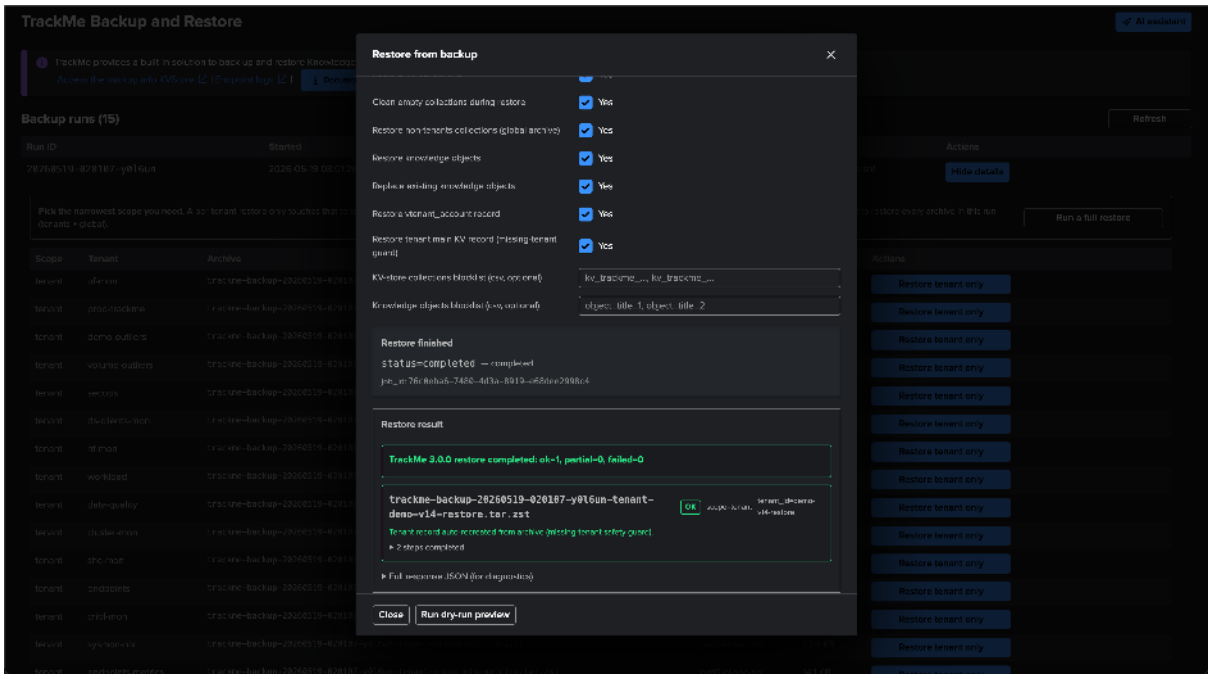
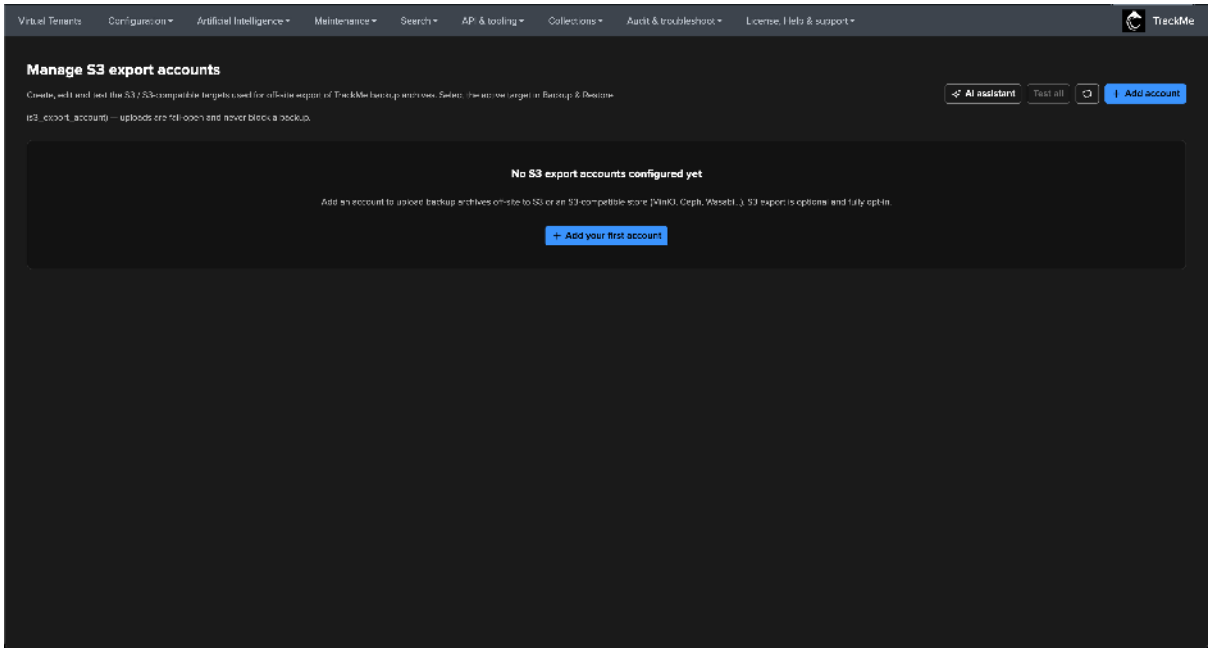
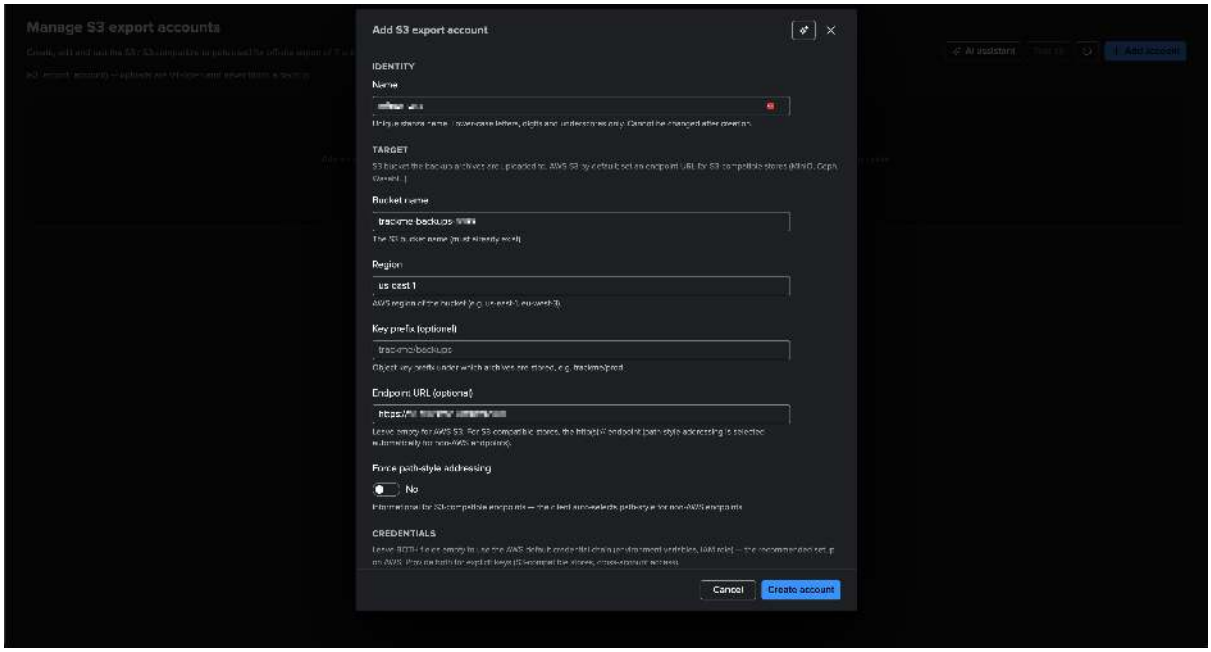
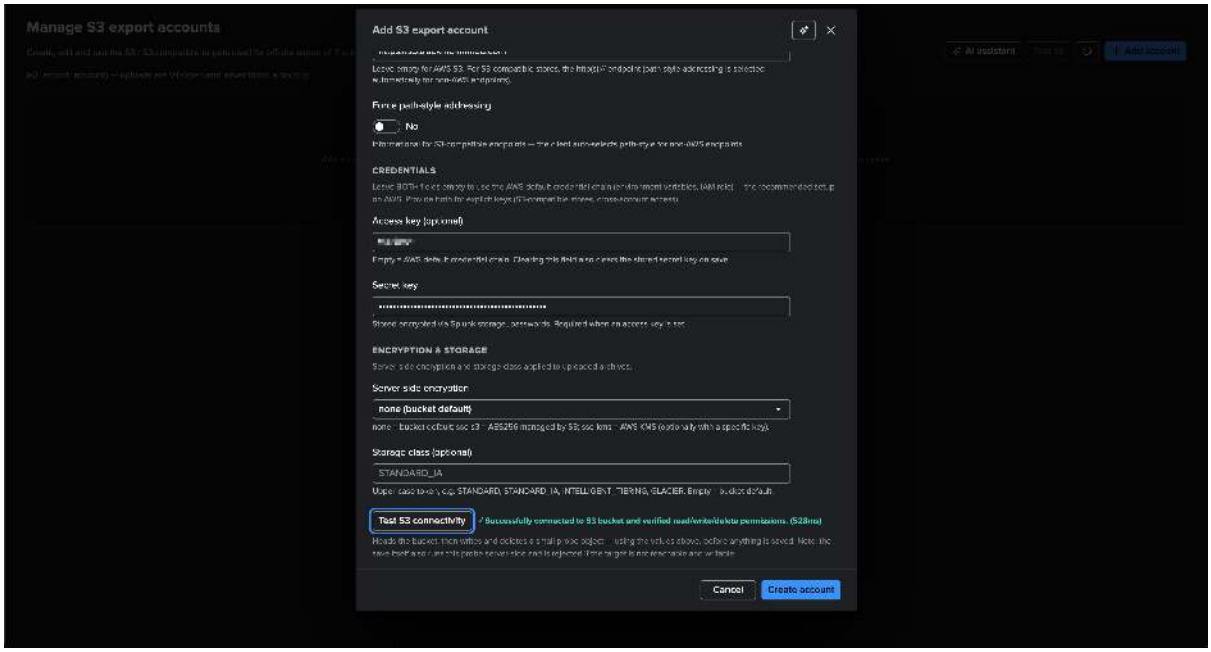
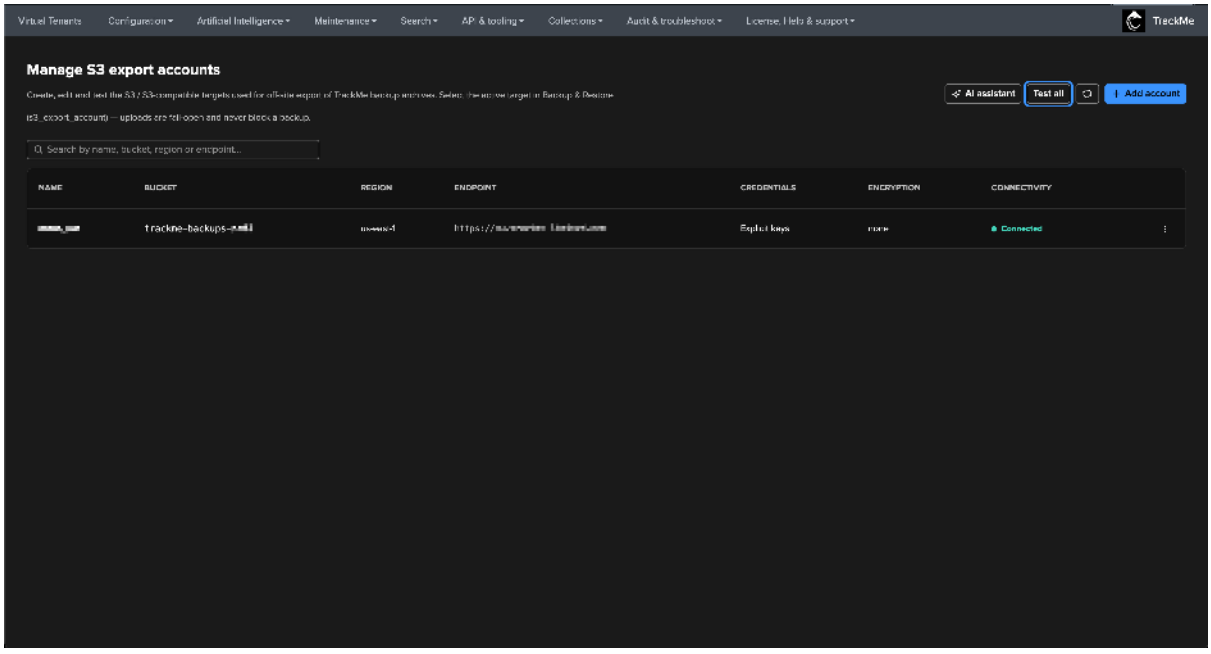
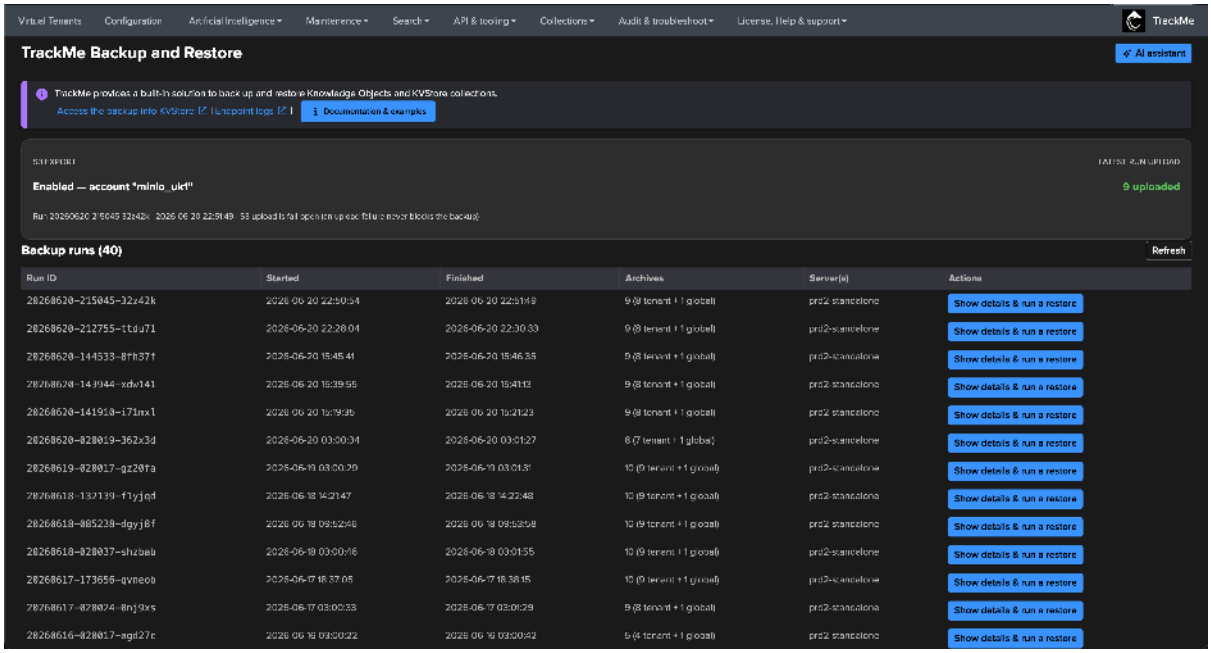
Several challenges need to be tackled:

- **volume**: the volume of log messages can be very high, running large searches can be very resource consuming, or not even realistic in a continuous monitoring context.
- **variability**: the volume of log messages can vary a lot, depending on time conditions, users activity and so forth.
- **relevance:**: Simply tracking for errors is not meaningful, errors happen all the time, and are not necessarily symptomatic of a problem. The key is to detect abnormal trends.

The following chart shows the challenges quite well:


```

An ad-hoc investigation can possibly detect an abnormal condition, our goal is to leverage TrackMe to do this work efficiently:



TrackMe's Machine Learning Outliers detection:



Implementation in TrackMe

Step 1: Create a Virtual Tenant

First, let's create a new Virtual Tenant for our use case, we may well simply use an existing Virtual Tenant, for the purposes of the documentation we will simply create a new tenant:





Step 2: Create the new Flex Object tracker

The use case was integrated in the Flex Object library in TrackMe 2.0.97, so you can simply call the template splk_splunk_infra_log_level_variations:

Note: You can find the use case search at the end of this document


Flex Objects trackers - Manage and create
Flex Objects tracking: monitor the results of any kind of Splunk search, local or remote, and turn knowledge into value using TrackMe's unique workflow:
The Flex Object tracking component requires a search that provides one or many entities in a single execution as well as the following output:
Field name
Mandatory/Optional
Usage
Example
object
mandatory
name of the object in the collection, any ascii character is accepted
group
optional
Grouping of objects related to the same tracker or context, can be optionally defined in the search logic.
Same groups can be set across multiple trackers but overlap should be avoided, if group is unset, the tracker name will be used
object_description
optional
Describe the object according to your context, for an easier management.
status_description
optional
Describe the status condition according to your context, for an easier management.
metrics
optional
if relevant you can include any number of Key Performance Indicators (KPIs), according to the following convention:
Cancel
Create a new flex tracker
Manage existing flex trackers


Flex Objects tracking - create a new tracker
2. Define the search logic
Enter a name for the new object and define if the deployment is local or a remote account:
New tracker identifier (will be added as a prefix to entities, and used to group entities)
splk_splunk_infra_log_level_variations
Splunk deployment, either local or a configured remote account:
local
Open the configuration interface to add or manage accounts
You can pre-load a use case from the Flex Objects use case library, or manually define the Flex search logic:
Vendor:
Splunk
Categories:
splunk infrastructu...
Use case:
splk_splunk_infra_l...
Review Flex Objects use cases library
Use case information:
"uc_ref": "splk_splunk_infra_log_level_variations",
"uc_vendor": "Splunk",
"uc_description": "Monitors the Splunk instance internal logs by the kind of the logging level. This use case then leverages Machine Outliers Anomaly detection to detect abnormal trends of errors in Splunk instance.",
"uc_category": "splunk infrastructure",
Cancel
Back



Once the tracker has been executed at least once, we start tracking the log messages logging level on a per Splunk instance basis (which you can customise):





Step 3: (optional) backfill KPIs

You can optionally backfill the KPIs, so ML models can be trained with proper dataset immediately:

Note: ensure to replace the name of the tenant in the Splunk search

```
index=_internal host=* log_level=* (log_level=ERROR OR log_level=INFO OR␣
↪log_level=WARN OR log_level=FATAL) earliest=-30d@d latest=now

``` The basis break by statement logic happens on a per host basis, you may want␣
↪to update the logic to group by tiers logic, such as grouping by indexer␣
↪cluster, Search head, Search Head Cluster and so forth ```
| bucket _time span=30m
| stats
count(eval(log_level="INFO")) as "trackme.splk.flx.splunk.log_level.info"
count(eval(log_level="WARN")) as "trackme.splk.flx.splunk.log_level.warn"
count(eval(log_level="ERROR")) as "trackme.splk.flx.splunk.log_level.error"
count(eval(log_level="FATAL")) as "trackme.splk.flx.splunk.log_level.fatal"
by _time, host
| fillnull value=0

| eval alias=host
| lookup trackme_flx_tenant_splunk-log-level alias OUTPUT _key as object_id,␣
↪object, object_category, tenant_id

| mcollect index=trackme_metrics split=t object, object_category, object_id,␣
↪tenant_id
```

Once processed, KPIs should be backfilled depending on if the internal index retention allowed it:
```



Step 4: Review Machine Learning Outliers detection

Once TrackMe has trained Machine Learning models, you can review the results in the Machine Learning Outliers detection dashboard:

Note: You can manually run the mltrain job to force training the models now, or train a particular model through the UI



In our demo case, we know we already have an abnormal period, which can add as an exclusion so the models are not affected by this period:



Detecting abnormal error logging increasing trend:





Detecting insufficient informational logging:



Our detection is now fully ready, and TrackMe will alert us if abnormal conditions are detected on the Splunk instances and deployments.

9.4.5 Using SLA alerting to build a 2-tier monitoring system

About SLA alerting

- **SLA alerting** concepts were introduced in TrackMe 2.0.92.
- The concepts of SLA alerting can be leveraged to design a 2-tier monitoring system, where a first alert is emitted when TrackMe entities switch to a red state, and a second independent alert is emitted when the SLA is breached.

- This can be translated as “alert: this TrackMe entity is red!”, then later on “alert: This entity has now spent too long in alert and is breaching its SLA!”.
- Entities are associated with a given SLA class, each SLA class has a threshold in seconds and a numerical rank value.
- Available SLA classes and their configuration are defined at the level of the TrackMe system configuration. (A JSON object which defines the list of classes and their parameters)
- You can then define the SLA class per entity (which if defined will take precedence over policy-based classes), or via SLA policies. SLA policies support two modes:
  - **Regex-based policies** match regular expressions against the entity name and assign an SLA class.
  - **Lookup-based policies** leverage existing Splunk lookups (CSV or KVstore) to assign SLA classes based on field mappings, making it easy to integrate with a CMDB or asset inventory.
- If multiple policies match, the **highest ranked SLA class wins** (based on the rank value defined in the SLA class configuration).
- Finally, you can create an SLA alert, which leverages the SLA information to emit an alert when the SLA is breached.

SLA Classes and Thresholds

SLA classes are defined at the level of the TrackMe system configuration, SLA classes define a threshold in seconds and a numerical rank value:



The default SLA classes are:

```
{
    "gold": {
        "sla_threshold": 14400,
        "rank": 3
    },
    "silver": {
        "sla_threshold": 86400,
        "rank": 2
    },
```

(continues on next page)

(continued from previous page)

```
    "platinum": {
        "sla_threshold": 172800,
        "rank": 1
    }
}
```

Notes:

- You can add / remove / change classes as needed.
- You can define the default class to be used when TrackMe entities are discovered. (parameter: default_sla_class)
- The **Threshold** value is in seconds. For an SLA to be breached, a given TrackMe entity must be in alert for a continuous amount of time exceeding the threshold.
- The **rank** value is a numerical value, it is used to handle any conflict when applying SLA policies, the highest rank value will always win.

SLA Tab in TrackMe UI

The SLA feature is first translated into a tab called SLA which describes the current SLA status for the selected TrackMe entity:

The entity is in green state, therefore the SLA cannot be breached:



The entity is red state, however it has not yet breached the SLA threshold by spending a continuous amount of time in alert higher than the threshold:



The entity is red state, and it has breached the SLA threshold by spending a continuous amount of time in alert higher than the threshold:



Defining the SLA class per entity

You can define the SLA class manually on per entity basis: (via the UI or via the associated endpoint)

Hint

If the SLA has been defined manually, it will take precedence against policies based classes. (see next section)





Defining SLA policies to assign SLA classes automatically

TrackMe supports the management of SLA classes via policy, which can be defined per Virtual Tenant:

- SLA policies allow you to automate SLA class assignment across entities using **regex-based** or **lookup-based** rules.
- **Regex-based policies** match regular expressions against the entity name (`object` field) and assign the configured SLA class.

- **Lookup-based policies** leverage existing Splunk lookups (CSV or KVstore) to assign SLA classes based on field mappings, making it easy to integrate with a CMDB or asset inventory.
- Matching entities are automatically updated with the SLA class defined in the policy.
- If multiple policies match a given entity, the **highest ranked SLA class takes precedence** across all policy types (based on the rank value defined in the SLA class configuration).
- An entity managed by policies can still be updated manually, and the policy will not override the manual update.

Lookup-based SLA policies

Lookup-based policies leverage existing Splunk lookup transforms (CSV files or KVstore collections) to assign SLA classes to entities based on field mappings. This is the recommended approach when integrating with an external CMDB, asset inventory, or any structured data source.

Creating a lookup-based SLA policy:

- Click **Create new policy** and select the **Lookup** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)
- Select a **Splunk lookup transform** from the dropdown — TrackMe lists all available lookup transforms in your Splunk environment
- Configure **field mappings** to map lookup fields to entity fields — for example, map the lookup field `index` to the entity field `data_index`, and the lookup field `host` to the entity field `object`
- Select the **SLA class field** — the field in the lookup that contains the SLA class value
- Optionally configure **SLA class value mappings** to translate foreign values to TrackMe's SLA classes (e.g., `tier1` → `gold`, `tier2` → `silver`)
- Select the **match mode**: **Exact** (case-insensitive string matching) or **Wildcard** (supports `*` and `?` patterns)
- Click **Simulate** to preview which entities would be matched and what SLA classes would be assigned

Lookup policy configuration details

- **Field mappings**: Define how lookup fields map to entity fields. All mapped fields must match for an entity to be selected. For example, if you map `index` → `data_index` and `host` → `object`, both conditions must be satisfied.
- **SLA class field**: The column in the lookup containing the SLA class value (e.g., a `sla_tier` or `service_level` column).
- **SLA class value mappings** (optional): If your lookup uses custom values like `tier1`, `tier2`, `tier3`, you can map them to TrackMe SLA classes (`gold`, `silver`, `platinum`, etc.). If no mappings are configured, the lookup values must already match TrackMe's SLA class names.
- **Match modes**: `Exact` performs case-insensitive string comparison. `Wildcard` supports `*` (matches any characters) and `?` (matches a single character) for flexible pattern matching.

Regex-based SLA policies

Regex-based policies match a regular expression pattern against the entity name (`object` field). When the pattern matches, the configured SLA class is assigned to the entity.

Creating a regex-based SLA policy:

- Click **Create new policy** and select the **Regex** mode
- Enter a **policy identifier** (or let TrackMe auto-generate one)

- Enter the **regex pattern** to match against entities
- Select the **SLA class** to assign to matched entities
- Click **Simulate** to preview which entities would be matched before saving

For instance, say we want to match entities containing "cribl":







SLA class hierarchy and conflict resolution

When multiple policies (regex, lookup, or a mix of both) match the same entity, TrackMe applies the **highest ranked SLA class** across all matching policies.

The rank is determined by the numerical rank value defined in the SLA class configuration (see SLA Classes and Thresholds above). The policy with the highest rank value wins.

For example, if a regex policy assigns `silver` (rank 2) and a lookup policy assigns `gold` (rank 3) to the same entity, the entity will receive the `gold` SLA class.

Running the SLA policy tracker

After creating one or more policies, you can execute the SLA policy tracker to apply all configured policies to your entities:

- Click the **Run policy tracker** button in the policy management screen
- TrackMe will evaluate all regex and lookup policies against all entities in the selected component
- A summary of the results is displayed, including the number of entities updated, matched, and any errors encountered



From this stage, all matching entities will get the highest ranked policy and its associated threshold:



SLA Alerts

From TrackMe alert tabs, you can now create an SLA alert:

Hint

The TrackMe component must be defined and match your target (there would be one alert per component in the tenant)



Notes:

- This alert is independent from the TrackMe main alerts and notable alert
- If it triggers, it means that the SLA has been breached for one or more entities
- If the SLA is breached, the concept is to say that basically we had a first alert, the issue is not fixed after an acceptable amount of time so we generate a second alert once the SLA threshold has been breached. (2 tiers alerting system)

Example:



9.5 Operations & administration

White papers on running TrackMe day to day — backup and restore, entity auto-deletion and lifecycle management, and mass operations at scale.

9.5.1 Backing up and Restoring TrackMe

Hint

TrackMe 2.3.23: multi-archive backup format and a streamlined restore UI

- From release **2.3.23**, every backup run produces **N+1 independent archives**: one **global archive** plus one **tenant archive per enabled Virtual Tenant**.
- Each archive can be **restored, downloaded, exported or deleted on its own** — a failure on one archive never blocks the others.
- Restoring is driven entirely from the **TrackMe Backup and Restore UI** (menu **API & Tooling → Backup & Restore**), including a built-in **dry-run preview** and an explicit *"this will overwrite live data"* confirmation step.
- **Search Head Cluster (SHC)** support is transparent — TrackMe automatically delegates each operation to the SH peer that owns the archive.
- **Legacy archives** produced by 2.3.21 and earlier remain fully restorable (see *Legacy archive compatibility (pre-2.3.22)*).

Hint

TrackMe 2.4.3: configurable backup storage and off-site S3 export

- The **backup directory** and **retention** are now configurable under **Configuration → System settings → Backup & Restore**, with a fail-safe fallback to the default location (see *Backup storage — retention and location (2.4.3)*).
- Completed archives can be **exported off-site to S3** — Amazon S3 or any S3-compatible store (MinIO, Ceph, …) — **opt-in** and **fail-open** (an upload failure never blocks or fails the backup; the local archives are always intact). See *Off-site export to S3 (2.4.3)*.

Note

What TrackMe backs up:

- **Knowledge Objects** created by TrackMe (KVstore definitions, transforms, macros, reports, alerts, saved searches).
- **KVstore collection records** — TrackMe stores its state, entities and configuration in KVstore collections.
- **Virtual Tenant accounts** — the per-tenant `vtenant_account` records that drive RBAC, scheduling, component enablement, etc.

Note

Limitations:

- **Disabled Virtual Tenants are not backed up.** When a backup is restored, any tenant that was disabled at backup time is not restored either. If the tenant exists at restore time but was not in the backup, TrackMe force-purges it.

- **Alert actions:** TrackMe restores alerts that were created via TrackMe, but only TrackMe built-in alert actions and Splunk email alert actions are supported. External alert actions (custom modular alerts) must be re-configured manually after restore.

What's new in 2.3.23 — the multi-archive model

Earlier releases produced a single monolithic `.tgz` archive containing every tenant and every system collection. At scale, that single file became impractical to restore reliably — a corruption in one tenant could block recovery of all the others.

From 2.3.23, each **backup run** is decomposed into independent archives:

- **One global archive** — `trackme-backup-<RUN_ID>-global.tar.zst` — contains the non-tenant KV collections (`kv_trackme_virtual_tenants`, maintenance mode, bank holidays, native ML models, license, etc.) and global knowledge objects.
- **One tenant archive per enabled tenant** — `trackme-backup-<RUN_ID>-tenant-<tid>.tar.zst` — contains that tenant's `_tenant_<tid>` KV collections, its tenant-scoped knowledge objects and the tenant's `vtenant_account` snapshot.

The `RUN_ID` has the form `YYYYMMDD-HHMMSS-<6 chars>` and groups every archive produced by one invocation.

Why this matters in practice:

- **Per-tenant recovery.** When a single tenant's data is corrupted, you can restore just that tenant's archive — every other tenant stays untouched.
- **Per-archive isolation.** A failure on one archive (zstd corruption, disk issue, etc.) does not abort the run. The other archives are produced, validated and restorable independently.
- **Per-archive validation.** Each archive is sha256-checksummed and zstd-tested at creation time, so corruption is caught on the producing peer rather than at restore.
- **No opt-in.** Once you upgrade to 2.3.23, scheduled backups produce the new format automatically.

Accessing the TrackMe Backup and Restore user interface

The Backup and Restore page is reached from the **API & Tooling** menu. The top section shows the global state — how many archives exist, when the most recent one was produced, and quick actions:



From here you can:

- **Create backup** — trigger a backup run on demand. Produces one global + one per-tenant archive on the local SH (or on whichever SH the request lands on).
- **Download a backup archive** — pick a run, then pick one archive within the run, then download. SHC-transparent: the file is fetched from the owning peer automatically.
- **Import a backup archive** — upload a `.tar.zst` (3.0.0) or legacy `.tgz` (pre-2.3.22) file. TrackMe validates it and registers it as an available archive on the receiving peer.
- **Backup archive info collection** — flat table of every known archive with dry-run inspect, download, delete and "Show Details" actions.

Inspecting an archive before restoring

Two options are available depending on how much detail you need:

Quick check — Dry-run inspect from the table:
The **Dry-run inspect** button on each row of the backup archive info table runs the archive through extraction and metadata parsing **without applying any change**. Use it to confirm the archive opens cleanly before triggering a real restore.

Detailed breakdown — Show Details on an archive:



The **Show Details** action expands an archive and lists every KVstore collection it contains with its on-disk size. Helpful for sanity-checking what is actually inside the archive, and for spotting unexpectedly small or large collections before a restore.

For SPL-driven inspection (CI checks, scripted audits), the built-in command still works against both legacy and 3.0.0 archives:

```
| trackmecheckbackups archives_list="*"
```

Restoring TrackMe — UI walkthrough

The recommended workflow lives entirely in the UI. The example below walks through restoring a single tenant from a multi-archive run — the typical use case when one tenant has been corrupted or accidentally cleared.

Step 1 — Pick a backup run

The middle section of the page lists every known backup run:



Each row shows the run ID, when it started, when it finished, how many archives it contains (e.g. *20 tenants + 1 global*) and which SHC peer(s) produced it. Click **Show details & run a restore** on the run you want to use.

Step 2 — Pick a scope (single archive vs. whole run)

The run expands into its per-archive table:



You now have two choices:

- **Restore tenant only** (per row) — restores just that tenant's archive. The recommended path when only one tenant is affected.
- **Run a full restore** (top right) — restores every archive in the run, both the global archive and every tenant archive. Reserve for full disaster recovery.

Step 3 — Configure the restore

Clicking **Restore tenant only** opens the restore modal pre-populated with the target archive's filename:



The toggles let you decide what to restore:

- **Restore KV collections** — write back the KVstore records contained in the archive.
- **Clean empty collections during restore** — when a collection was empty at backup time, empty the live collection too. Leave on unless you have a specific reason to keep stale data.
- **Restore non-tenants collections (global archive)** — only applies when restoring the global archive; controls whether system-wide collections (`kv_trackme_virtual_tenants`, maintenance mode, bank holidays, etc.) are touched.
- **Restore knowledge objects** — write back the saved searches, macros, transforms and reports captured in the archive.
- **Replace existing knowledge objects** — when on, an existing KO with the same name is deleted and re-created from the archive. When off, the existing live KO wins.
- **Restore vtenant_account record** — write back the `vtenant_account` JSON for the tenant (component enablement, scheduling, RBAC roles, …).
- **Restore tenant main KV record (missing-tenant guard)** — when on, if the tenant is **absent** from `kv_trackme_virtual_tenants` at restore time, TrackMe auto-recreates the tenant record from the archive's own snapshot. Leave on — this is what makes per-tenant restore of a deleted tenant possible.

The **blocklists** at the bottom let you exclude specific KV collections or specific knowledge objects by name (comma-separated). Useful for fine-grained surgical restores ("restore everything except these two collections").

Step 4 — Restrict which collections / knowledge objects to restore (optional)

Scrolling down reveals the **Customize selection** panel:



By default every collection and every knowledge object in the archive is selected. To exclude items, open the multi-select dropdowns and uncheck what you want to skip:



A search box on each dropdown helps when an archive contains hundreds of collections. **Clear all** lets you start from an empty selection and pick just the few you want.

Step 5 — Run a dry-run preview

Always click **Run dry-run preview** before applying anything. The preview output shows exactly what the restore would do:

```
{
  "response": "TrackMe 3.0.0 restore completed: ok=1, partial=0, failed=0",
  "dry_run": true,
  "backup_run_id": "...",
```

(continues on next page)

(continued from previous page)

```
  "archives": [ ... per-archive summary ... ]
}
```

Read the per-archive summary carefully: how many records would be loaded into each KVstore collection, which knowledge objects would be replaced, whether the missing-tenant safety guard would fire. **No live data is touched at this stage** — the dry-run is safe to run repeatedly while you tune the selection.

Step 6 — Apply the restore

Once the dry-run output looks right, tick the **"I understand this will overwrite live data"** checkbox. Only then does the **Apply restore** button activate:



The button is rendered in red on purpose — applying a restore is a destructive operation on the selected scope.

Step 7 — Monitor progress

The restore runs asynchronously. The modal switches to a live progress view:



The panel shows the job ID, the current status and the step in progress. The restore is **resumable**: if the underlying `splunkd` subprocess is recycled during a long run, TrackMe's watchdog auto-resumes from the last checkpoint — you do not need to re-trigger the restore manually.

Cancel job is available if you need to abort. Cancel takes effect at the next checkpoint between archive steps, not in the middle of a single step.

Step 8 — Read the result

When the restore terminates, the modal shows the result panel:



What to look at:

- **status=completed** in the *Restore finished* panel.
- The green `ok=N, partial=N, failed=N` summary in the *Restore result* panel. `ok` is the number of archives that fully succeeded.

- Per-archive boxes for any archive that ended in `partial` or `failed` state, with the per-archive error included inline.
- The **"Tenant record auto-recreated from archive (missing-tenant safety guard)"** annotation is normal when the target tenant was missing from `kv_trackme_virtual_tenants` at restore time — TrackMe rebuilt the central record from the archive's own snapshot.
- **Full response JSON (for diagnostics)** — expandable, contains the complete per-step structured response. Useful when raising a support ticket.

Restoring a full backup run

The **Run a full restore** button (visible after expanding a run, top-right of the per-archive table) opens the same modal but pre-targets the **whole run**: every tenant archive plus the global archive. The same toggles, customize-selection and dry-run preview apply.

Use this path when:

- You are restoring into a fresh deployment.
- You are running a disaster-recovery drill.
- Multiple tenants have been corrupted or the system-wide collections are damaged.

For single-tenant corruption, prefer **Restore tenant only** — the blast radius is much smaller.

Automated backups

TrackMe runs backups automatically in two situations:

- **Scheduled backup** — the saved search `trackme_backup_scheduler` runs every night and produces a fresh backup run (one global + one per enabled tenant archive).
- **Schema-upgrade backup** — when a TrackMe release upgrade is applied and the first Virtual Tenant goes through the schema upgrade, TrackMe produces a backup run beforehand. The run comment records the version transition.

Backup storage — retention and location (2.4.3)

Retention and the on-disk location of backup archives are configured under **Configuration → System settings → Backup & Restore** — the retention (in days), the backup directory and the active S3 export account.

Backup retention

Backup retention (days) (`[backup_restore]` → `backup_retention_days`, default `30`) controls the age-based purge of local backup archives. The retention purge removes both 3.0.0 multi-archive runs and legacy archives older than the threshold.

On-disk location

By default, TrackMe stores backup archives under `$SPLUNK_HOME/etc/apps/trackme/backup/`. From **2.4.3** this is configurable with **Backup directory (optional)** (`[backup_restore]` → `backup_directory`):

- **Leave it empty to use the default.** To override, provide an **absolute** path (e.g. `/data/backups`) — relative paths are rejected. Environment variables and ~ are expanded.
- **Validated at use.** TrackMe creates the directory if needed and write-probes it (write + flush + fsync). If the configured directory cannot be created or written to, TrackMe **logs an error, raises a** `backup_directory_unwritable` **Configuration Guardian alert, and falls back to the default location** — so backups never silently stop.
- **New backups only.** Changing the directory affects *new* runs; existing archives are not moved or re-keyed.

Note

Search Head Cluster: point `backup_directory` at a path that exists and is writable by `splunkd` on **every** SHC peer (a shared mount, or an identical local path on each peer). Each peer writes the archives it produces to this location.

Off-site export to S3 (2.4.3)

From release **2.4.3**, completed backup archives can be **uploaded off-site to S3** — either Amazon S3 or any S3-compatible object store (MinIO, Ceph, …). This gives you a disaster-recovery copy outside the Splunk deployment without any external scripting.

Note

Why this is safe to enable

- **Opt-in** — nothing is uploaded until you configure an S3 Export account and select it in the Backup & Restore settings.
- **Fail-open** — if S3 is unreachable, the backup still completes; the local archives are always intact. The upload failure is recorded per-archive and surfaced (see *Monitoring failures* below), never raised.
- **Local backups are unchanged** — S3 export is an *additional* copy of the same archives, uploaded after each run completes.

Step 1 — Create an S3 Export account

S3 targets are managed from a dedicated page reachable from the top navigation: **Configuration → Manage S3 export accounts.** The page lists every target with its bucket, endpoint and last connectivity status, and offers per-row **Edit / Clone / Test connectivity / Delete** actions plus **Test all** and **Add account**. When no target exists yet, the page shows a short getting-started guide:



Click Add account and name a bucket, region, prefix and (for S3-compatible stores) an endpoint:



Key fields:

- **Bucket name —** the destination bucket (it must already exist).
- **Region —** the AWS region. Optional for S3-compatible stores; defaults to `us-east-1` for request signing.
- **Prefix —** an optional key prefix. Archives are stored under `<prefix>/<run_id>/<filename>`.
- **Endpoint URL —** leave empty for AWS S3; set it (e.g. `https://s3.example.com`) to target an S3-compatible store such as MinIO. Addressing style is auto-selected (path-style for non-AWS endpoints).
- **Access key / Secret access key —** the credentials. **Leave both empty** to use the host's IAM instance / ECS / EKS role (the AWS default credential chain) for keyless deployments. The secret is stored encrypted in Splunk `storage_passwords`.
- **Server-side encryption —** optional: SSE-S3 (AES256) or SSE-KMS (with an optional KMS key id).
- **Storage class —** optional: e.g. `STANDARD_IA`, `GLACIER`, `INTELLIGENT_TIERING`.

Note

Connectivity is verified on save. When you create or update an S3 Export account, TrackMe runs a live probe (a bucket check plus a write/delete of a tiny test object). A save whose target fails the probe is **rejected** — mirroring how remote Splunk accounts are validated — so a broken account cannot be saved silently.

Use Test S3 connectivity in the modal to run that probe on demand — it verifies read, write and delete access and reports the round-trip time:



Once saved, the target appears in the list with its live connectivity status:



Step 2 — Enable S3 export for backups

S3 export is activated with the **S3 Export account (enables S3 upload)** dropdown in the **Backup & Restore** settings (`[backup_restore]` → `s3_export_account`) — the same settings section shown in *Backup storage — retention and location (2.4.3)*. Leaving it empty disables S3 export. Once an account is selected, every subsequent backup run — scheduled, schema-upgrade or on-demand — uploads its archives automatically.

Step 3 — Verify uploads

The top of the Backup & Restore page shows the live S3 export status — the selected account, the latest run and how many archives were uploaded:



For each run, every archive (`.tar.zst`), its `.full.meta` / `.light.meta` sidecars and the run `manifest.json` are uploaded under `<prefix>/<run_id>/`. The per-archive upload outcome is also recorded on the archive's metadata record — the `s3_export_status` / `s3_export_uri` / `s3_export_error` fields in `kv_trackme_backup_archives_info`.

Two REST endpoints back this feature: `POST /trackme/v2/backup_and_restore/test_s3` tests connectivity on demand, and `POST /trackme/v2/backup_and_restore/export_s3` uploads existing archives (unlike the fail-open auto-upload, the on-demand endpoint *reports* failures). See the *API Reference* dashboard inside TrackMe for the exact request shape.

Monitoring failures

Because S3 export is fail-open, a silent upload failure would otherwise go unnoticed. TrackMe surfaces it two ways:

- The **Configuration Guardian** raises an `s3_export_failing` alert when the configured account is unreachable (a proactive daily probe), the configuration is invalid, or the latest run failed to upload one or more archives. It self-heals once a subsequent run uploads cleanly.
- The per-archive `s3_export_status` field (`ok` / `failed`) and the `post_backup` response `s3_export` summary record exactly which archives uploaded and which did not.

Retention of S3 objects

TrackMe **does not delete objects from the S3 bucket** — S3-side retention is intentionally left to a **bucket lifecycle policy**. This is a deliberate security and reliability choice:

- **Least privilege.** TrackMe's S3 credentials can be scoped to **write-only** (`PutObject`). The whole point of an off-site copy is to survive the primary deployment being compromised; granting delete rights on the DR copy

would let a compromise of the Splunk host (or the stored secret) wipe your off-site backups too.

- **Immutability.** Combine a lifecycle expiration rule with **Object Lock (WORM)** for ransomware-resistant, immutable-until-expiry retention — something a TrackMe-driven delete could not provide.
- **Reliability.** Expiration runs server-side and asynchronously, with no extra failure mode in the backup path.

Scope the lifecycle rule to the TrackMe prefix so it never affects unrelated objects. For example, on MinIO:

```
mc ilm rule add --expire-days 30 <alias>/<bucket> --prefix "<prefix>/"
```

On AWS S3, configure an equivalent **Lifecycle configuration** with an `Expiration` rule filtered to the same prefix (and `NoncurrentVersionExpiration` if bucket versioning is enabled).

Backup archive metadata KVstore

Every archive's metadata is recorded in the cluster-replicated `kv_trackme_backup_archives_info` collection. You can inspect it directly with SPL:

```
| inputlookup trackme_backup_archives_info | sort 0 - mtime
```

Notable fields (3.0.0 onwards):

- `backup_run_id` — groups archives produced by the same backup run.
- `archive_scope` — `tenant` or `global`.
- `tenant_id` — set when `archive_scope=tenant`, empty for global.
- `archive_schema_version` — `3.0.0` for the multi-archive format.
- `archive_sha256` — checksum captured at archive creation time.
- `server_name` — the SHC peer that produced the archive. Used by the cross-peer delegation logic when you trigger an operation from a different peer.
- `mtime` — archive timestamp.
- `s3_export_account` / `s3_export_status` / `s3_export_uri` / `s3_export_error` — set when *S3 export* is enabled: the target account, the per-archive upload outcome (`ok` / `failed`), the resulting `s3://` URI and any failure detail.

TrackMe itself does **not** restore the records of `kv_trackme_backup_archives_info`. The collection is auto-rebuilt from archive metadata, so a missing record (e.g. after an import) is recreated on the next scan.

Importing and exporting backup archives

Use **Import a backup archive** to move a backup file from one TrackMe instance to another — useful for migrations, clones or for replaying a customer-supplied backup in a lab. Both 3.0.0 `.tar.zst` (single archive) and legacy `.tgz` files are accepted.

Use **Download a backup archive** in the top section of the page to pull a specific archive out of the TrackMe deployment. The UI lets you pick the run first, then the archive within the run.

The download path uses a memory-efficient two-step token flow under the hood (POST to get a token, GET to stream the file). The UI handles this transparently. When scripting against the REST API directly, the same two-step flow applies; see the *API Reference* dashboard inside TrackMe for the exact endpoint shape.

Search Head Cluster considerations

In an SHC, archives physically live on the SH peer that produced them, but every operation (list, dry-run, restore, download, delete, export) can be initiated from **any** peer. TrackMe handles this transparently:

- Each archive's record carries the producing peer's `server_name`.
- When you trigger an operation from a peer that doesn't own the archive, TrackMe delegates the work to the owning peer via an internal REST call.

- Listing and grouping work on any peer because `kv_trackme_backup_archives_info` is cluster-replicated.

You don't need to know which peer owns which archive — the UI works the same way regardless.

Legacy archive compatibility (pre-2.3.22)

Archives produced by TrackMe 2.3.21 and earlier (single-archive `.tgz` or `.tar.zst` with `archive_schema_version` 1.0.0 / 2.0.0) remain **fully restorable indefinitely**:

- They appear in the UI under a clearly-labelled **Legacy archives** grouping.
- The legacy restore code path is preserved unchanged — restoring an old archive uses the same logic as a 2.3.21 deployment would have done.
- Import accepts legacy `.tgz` files transparently.
- `trackmecheckbackups` recognises both legacy and new filenames.

Recommended after upgrade: trigger a fresh backup once on 2.3.23 to gain per-tenant restore granularity for the new run. Existing legacy archives remain available for disaster recovery from cold storage.

Logs and troubleshooting

All backup and restore operations are logged under `trackme:rest_api`:

```
index=_internal sourcetype=trackme:rest_api source=
↪"*trackme_rest_handler_backup_and_restore.py"
```

For a failed restore, the **Full response JSON** block at the bottom of the restore modal carries the per-step structured response — that's the first place to look. The same structured payload is also written to the restore job record in `kv_trackme_backup_restore_jobs` and can be retrieved later by job ID even after the modal is closed.

9.5.2 Auto deletion or management of TrackMe entities

Introduction to automated management of TrackMe entities

About this paper:

- In some use cases, you may want to automatically manage entities in TrackMe based on different criteria.
- For instance, if you track hosts in TrackMe, you could automatically delete machines that have been tagged as decommissioned in your CMDB.
- This way, you would automatically keep your TrackMe entities in sync with your CMDB and purge entities as needed.

Automatically deleting entities in TrackMe with a custom search

In this example, we are managing hosts in TrackMe through the splk-dhm component. We can access the real-time view very easily using the following command:

Note: Click on the "Search table" button in TrackMe's UI to access this search.

Hint

Replace tenant and component name:

- Replace "mytenant" with the tenant ID you want to delete entities from.

```
| trackmegetcoll tenant_id=hosts-dsm-tracking component=dsm
```

Then, let's add a table to focus on specific fields:

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state
```

Now, let's assume that we want to filter on entities that have stopped forwarding data to Splunk, therefore are currently in a red state.

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"
```

Then, we will cross with our CMDB which is available in Splunk, we will use the alias field which contains the true host as seen in Splunk, you obviously can add any additional logic and/or customise the logic to match your context and requirements:

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""
```

The next step is to turn this into a comma-separated list of keyids, so that we can use it in a delete command using TrackMe's API endpoints and built-in pure SPL. We will also use the map command from Splunk:

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""

``` build the keyid comma separated list ```
| stats values(keyid) as keyid
| eval keyid=mvjoin(keyid, ",")
```

Last, and final step, call the endpoint:

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""

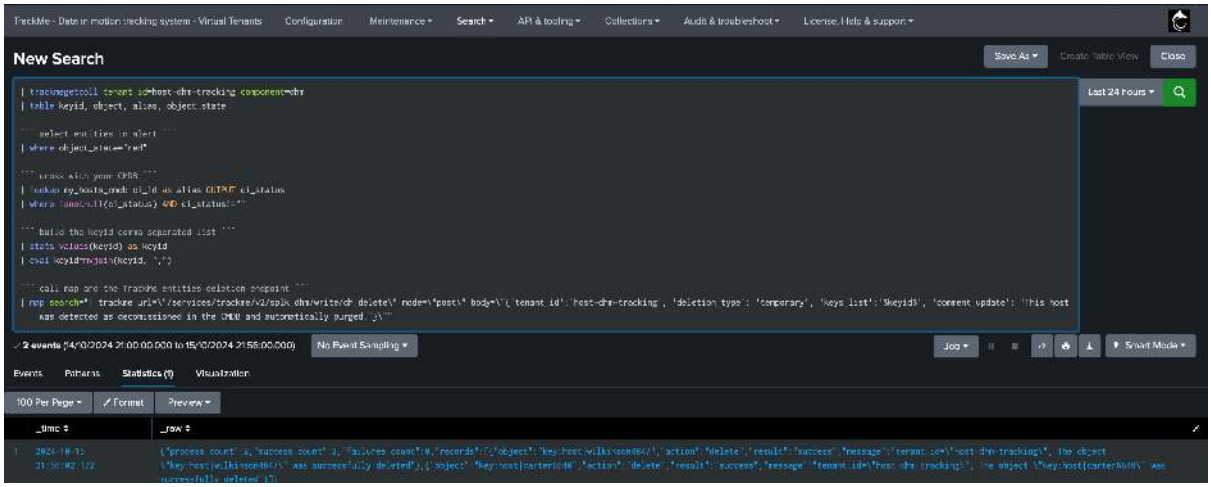
``` build the keyid comma separated list ```
```

(continues on next page)
```

(continued from previous page)

```
| stats values(keyid) as keyid
| eval keyid=mvjoin(keyid, ",")

``` call map and the TrackMe entities deletion endpoint ```
| map search="| trackme url=\"/services/trackme/v2/splk_dhm/write/dh_delete\"␣
↪mode=\"post\" body=\"{'tenant_id':'host-dhm-tracking', 'deletion_type':
↪'temporary', 'keys_list':'$keyid$', 'comment_update': 'This host was detected as␣
↪decommissioned in the CMDB and automatically purged.'}\""
```

We will schedule this search to run automatically. It doesn't need to run frequently, and most likely once per day or every few hours would be more than enough.

Any entities that are detected as decommissioned in the CMDB will be automatically purged from TrackMe. We also have TrackMe audit events should we need to review the deletion history.



Alternative using the Flex object component (splk-flex)

Licensed TrackMe users can also leverage instead the Flex object component to track hosts, the same logic can be used very easily too:

```
| trackmegetcoll tenant_id=hosts-flx-tracking component=flx
| table keyid, object, alias, object_state, object_description

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""

``` build the keyid comma separated list ```
| stats values(keyid) as keyid
| eval keyid=mvjoin(keyid, ",")

``` call map and the TrackMe entities deletion endpoint ```
| map search="| trackme url=\"/services/trackme/v2/splk_flx/write/flx_delete\"␣
↪mode=\"post\" body=\"{'tenant_id':'hosts-flx-tracking', 'deletion_type':
↪'temporary', 'keys_list':'$keyid$', 'comment_update': 'This host was detected as␣
↪decommissioned in the CMDB and automatically purged.'}\""
```
```

Alternative logic: disabling rather than deleting entities

Alternatively, we may want to disable entities rather than deleting them. The same logic would apply, and we would simply call a different endpoint.

With our splk-dhm example:

```
| trackmegetcoll tenant_id=host-dhm-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""

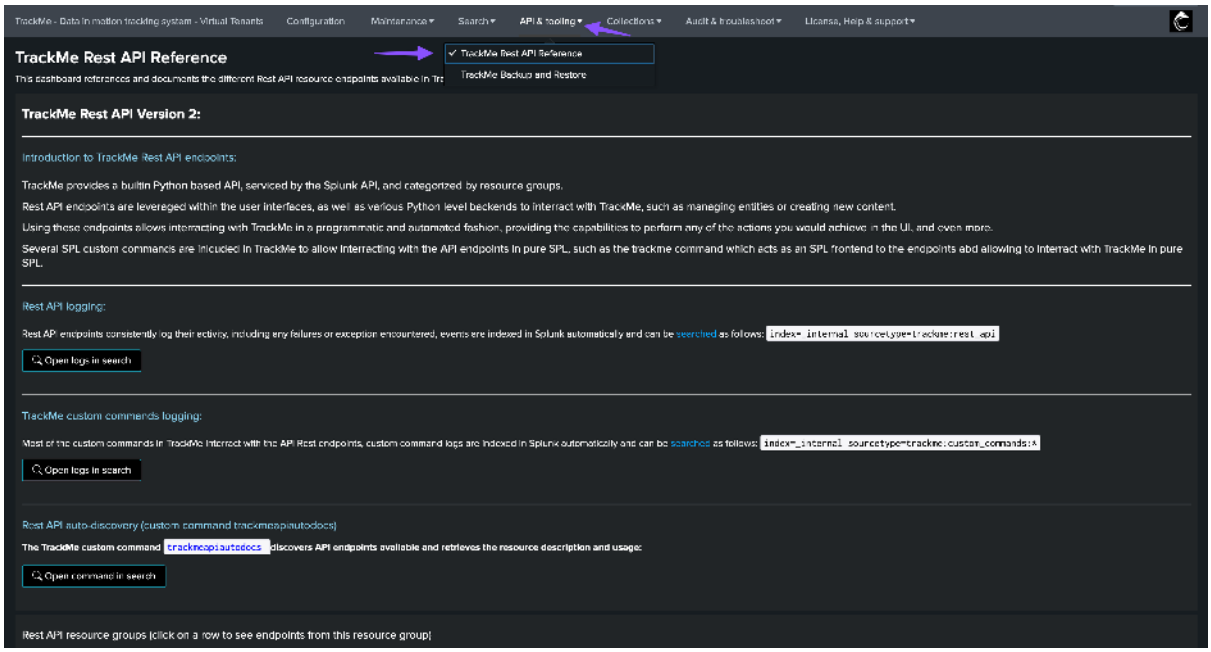
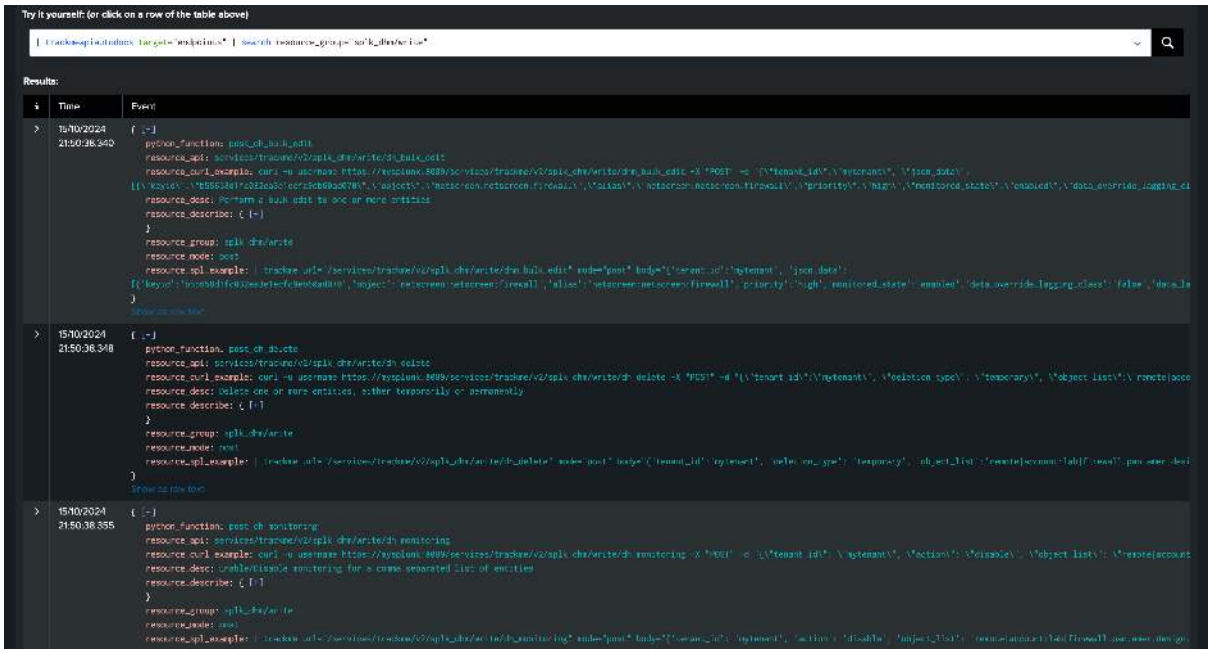
``` build the keyid comma separated list ```
| stats values(keyid) as keyid
| eval keyid=mvjoin(keyid, ",")

``` call map and the TrackMe entities deletion endpoint ```
| map search="| trackme url=\"/services/trackme/v2/splk_dhm/write/dh_monitoring\"␣
↪mode=\"post\" body=\"{'tenant_id':'host-dhm-tracking', 'action': 'disable',
↪'keys_list':'$keyid$', 'comment_update': 'This host was detected as␣
↪decommissioned in the CMDB and automatically disabled.'}\""
```

With our splk-flx example:

```
| trackmegetcoll tenant_id=host-flx-tracking component=dhm
| table keyid, object, alias, object_state

``` select entities in alert ```
| where object_state="red"

``` cross with your CMDB ```
| lookup my_hosts_cmdb ci_id as alias OUTPUT ci_status
| where isnotnull(ci_status) AND ci_status!=""

``` build the keyid comma separated list ```
| stats values(keyid) as keyid
| eval keyid=mvjoin(keyid, ",")

``` call map and the TrackMe entities deletion endpoint ```
| map search="| trackme url=\"/services/trackme/v2/splk_dhm/write/flx_monitoring\"␣
↪mode=\"post\" body=\"{'tenant_id':'host-flx-tracking', 'action': 'disable',
↪'keys_list':'$keyid$', 'comment_update': 'This host was detected as␣
↪decommissioned in the CMDB and automatically disabled.'}\""
```

Hint

The same type of logic applies to anything in TrackMe:

- In fact, you can use the same type of logic in many variations, whatever component you are dealing with.
- You can, for example, decide to have entities with a monitored state defined as disabled when entities are discovered, then wait for at least 7 days of historical knowledge to be collected before enabling the monitoring, or any other concept that makes sense for your context.

- Consult the REST API reference dashboard in TrackMe, menu **API & Tooling** to find which endpoints are available for each component, as well as their usage and usage examples.





9.5.3 Performing mass operations in TrackMe

Introduction to mass operations in TrackMe

About mass operations in TrackMe

- There are different contexts in which mass operations can be performed in TrackMe.
- Depending on the context, you can perform mass operations directly in the TrackMe UI (bulk edit), in some contexts you can also directly use TrackMe REST API endpoints and eventually take advantage of Splunk SPL techniques.

- The purpose of this whitepaper is to document various typical mass operations that can be performed in TrackMe, and to provide guidance on how to perform them.

Mass deleting entities

The following example demonstrates how to mass delete entities in TrackMe:

Hint

Replace tenant and component name:

- Replace "mytenant" with the tenant ID you want to delete entities from.
- "dsm" is the component name, simply replace it with the component you want to delete entities from. (dsm/dhm/mhm/wlk/flx)

Context:

- In this example, we assume that we have a large list of entities that should be removed from splk-dsm, for instance due to an issue in Splunk which has led to the unwanted creation of a large number of sourcetypes.

Step 1: Get entities

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to delete all entities related to the main index,␣
↪which unexpecty was fed with a large number of unwanted entities ```
| search data_index="main"
```

Step 2: Collect keys and turn into a CSV list

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to delete all entities related to the main index,␣
↪which unexpecty was fed with a large number of unwanted entities ```
| search data_index="main"
| table keyid
| stats values(keyid) as keys
| eval keys=mvjoin(keys, ",")
```

Step final: Delete entities

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to delete all entities related to the main index,␣
↪which unexpecty was fed with a large number of unwanted entities ```
| search data_index="main"
| table keyid
| stats values(keyid) as keys
| eval keys=mvjoin(keys, ",")
``` finally call the endpoint, which supports multiple entities deletion using the␣
↪Splunk map command```
| map search="| trackme url=/services/trackme/v2/splk_dsm/write/ds_delete␣
↪mode=post body=\"{'tenant_id':'mytenant', 'deletion_type': 'temporary',
↪'keys_list':'$keys$'}\""
```

Run the search above, if dealing with a very large number of entities, this search can take some time to complete, send the search to background and monitor the progress.

Mass enabling/disabling entities

Here is a customer use case:

- TrackMe is configured to automatically disable the monitoring state when new entities are discovered.
- The purpose is to avoid creating noise until we have enough historical knowledge of the feeds' behaviors, so that TrackMe's features such as the adaptive delay will eventually adapt the delay values if needed.
- Then, a custom scheduled search inspects entities, and when we have reached a state of having enough historical knowledge, disabled entities are enabled automatically.

Configuring TrackMe to disable the monitoring status at the time of the feed discovery:

The default monitored state is handled via a Splunk macro in TrackMe, which definition in macros.conf is as follows:

```
[trackme_default_monitored_state]
definition = eval monitored_state=if(isnull(monitored_state), "enabled",␣
↪monitored_state)
iseval = 0
```

To disable the monitoring state at the time of the feed discovery, you can override the macro definition to:

```
eval monitored_state=if(isnull(monitored_state), "disabled", monitored_state)
```

Creating a scheduled report which leverages TrackMe's metrics to define a confidence status, then call map and TrackMe's REST API to enable entities:

The following search would be scheduled at your convenience, and handle entities enabling based on a confidence status:

Hint

Replace tenant:

- Replace "mytenant" with the tenant ID value.

```
| mstats latest(trackme.splk.feeds.lag_event_sec) as lag_event_sec where␣
↪index=trackme_metrics tenant_id="mytenant" object_category="splk-dsm" by object␣
↪span=1d

| stats min(_time) as first_time by object
| eval metrics_duration=now()-first_time
| eval confidence=if(metrics_duration<(7*86400), "low", "normal")
| where confidence="normal"

| lookup trackme_dsm_tenant_mytenant object OUTPUT _key as keyid, monitored_state,␣
↪priority
| where (isnotnull(keyid) AND monitored_state="disabled")

|  stats values(keyid) as keys
|  eval keys=mvjoin(keys, ",")

|  map search="| trackme url=\"/services/trackme/v2/splk_dsm/write/ds_monitoring\"␣
↪mode=\"post\" body=\"{'tenant_id': 'mytenant', 'action': 'enable', 'keys_list': '
↪$keys$'}\""
```

Maintaining entities priority in a Splunk lookup file

A frequent question from TrackMe users is how they can maintain the priority of entities in a lookup file, and how to update the priority of entities in bulk.

In this example, we will demonstrate how to update the priority of entities in bulk for splk-dhm (Data Hosts tracking) component.

Step 1: Prepare the lookup file

Create a lookup file which contains at the minimum the following fields:

- host: the machine hostname as it is seen in Splunk (the host metadata)
- priority: the priority value you want to set for the entity.

While it is not fully mandatory, calling the lookup through a lookup definition is recommended so you can run a case-insensitive lookup call, you should be able to run:

```
| inputlookup hosts_lookup_cmdb
```

About case-sensitive match and lookup share

- Ensure to create a definition for the lookup so you can configure Splunk to allow case-insensitive lookup matches.
- Also, make sure both the lookup file and lookup permissions are set to be shared so TrackMe can access these.

Step 2: Create a report that will be scheduled and will update and maintain the priority of entities based on the lookup content

This is really simple, TrackMe stores the priority in the main KVstore collection, updating it based on a third party lookup can therefore be handled through a simple SPL logic:

Hint

Replace tenant:

- Replace “mytenant” with the tenant ID value.

Hint

About the alias field and components:

- In the example below, we use the TrackMe internal field called `alias`.
- In the context of splk-dhm, the alias will be the exact host value as seen in Splunk.
- In fact, TrackMe stores the unique object value in a field called object and is prefixed with a TrackMe metadata *key:*`, which is why we use the alias field instead to perform the lookup call.

```
| inputlookup trackme_dhm_tenant_mytenant | eval keyid=_key
| lookup hosts_lookup_cmdb host as alias OUTPUT priority as lookup_priority
| table keyid, lookup_priority, priority, *
| where isnotnull(lookup_priority)
| where priority!=lookup_priority
``` At this stage only entities to be managed will be updated ```
```

(continues on next page)
```

(continued from previous page)

```
| eval priority=lookup_priority, mtime=now()
``` Update ```
| outputlookup append=t key_field=keyid trackme_dhm_tenant_mytenant
```

Save this search as a report and schedule it at your convenience, it doesn't need to be executed very often so once every 4 hours for instance is likely enough.

Finally, note that this report will not return any results unless a priority update is needed, not returning any results simply indicates that all entities are up to date according to the lookup file.

Maintaining priority through a third party lookup would override any change made through the UI or REST API:

- Note that this logic above would override a priority change a TrackMe power user would make through the UI or REST API.
- Therefore, this needs to be known and documented to avoid any risks of confusion.

About other components:

- The logic or variations of this logic can be applied to any TrackMe component.
- Simply replace the component suffix, in this example `_dhm` with the associated component suffix. (dsm, mhm, flx, wlk, cim)

Mass updating entities priority

The following example demonstrates how to mass update entities priority in TrackMe:

Hint

Replace tenant:

- Replace "mytenant" with the tenant ID value.
- Replace the component, in this example we use splk-dsm, replace it with the component you want to update the priority for. (notably the KVstore collection prefix, dsm/dhm/mhm/flx/wlk/cim)

Step 1: Get entities

This example updates entities associated with indexes starting with "org", replace for your context

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to update all entities related to indexes starting by␣
↪org* ```
| search data_index="org*"
```

Step 2: Collect keys and turn into a CSV list

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to update all entities related to indexes starting by␣
↪org* ```
| search data_index="org*"
| table keyid
| stats values(keyid) as keys
| eval keys=mvjoin(keys, ",")
```
```

Step final: Update entities priority

```
| inputlookup trackme_dsm_tenant_mytenant | eval keyid=_key
``` in this example, we want to update all entities related to indexes starting by␣
↪org* ```
| search data_index="org*"
| table keyid
| stats values(keyid) as keys
| eval keys=mvjoin(keys, ",")
``` finally call the endpoint, which supports multiple entities using the Splunk␣
↪map command```
| map search="| trackme url=/services/trackme/v2/splk_dsm/write/ds_update_priority␣
↪mode=post body=\"{'tenant_id': 'mytenant', 'priority': 'high', 'keys_list': '
↪$keys$'}\""
```

Hint

API endpoints per component: (look into menu API & Tooling / TrackMe REST API Reference for all API endpoints documentation)

- splk-dsm: /services/trackme/v2/splk_dsm/write/ds_update_priority
- splk-dhm: /services/trackme/v2/splk_dhm/write/dh_update_priority
- splk-mhm: /services/trackme/v2/splk_mhm/write/mh_update_priority
- splk-wlk: /services/trackme/v2/splk_wlk/write/wlk_update_priority
- splk-flx: /services/trackme/v2/splk_flx/write/flx_update_priority
- splk-cim: /services/trackme/v2/splk_cim/write/cim_update_priority

CHAPTER

TEN

TROUBLESHOOT & FAQ:

10.1 Troubleshooting TrackMe

TrackMe is heavily instrumented: every REST API handler, custom command backend and alert action writes structured, indexed logs, and a family of audit dashboards turns those logs into actionable views. When something does not behave as expected, you almost never need to guess — the answer is in the logs, and this page is the map to find it.

10.1.1 How TrackMe logs

All TrackMe components log in a consistent, structured format. Events carry a `log_level` field (`INFO`, `WARNING`, `ERROR`, `DEBUG`), and most carry contextual fields such as `tenant_id`, `component`, `task`, `run_time` and `instance_id` so you can pivot quickly from a symptom to a root cause.

The table below summarises where each class of events is indexed:

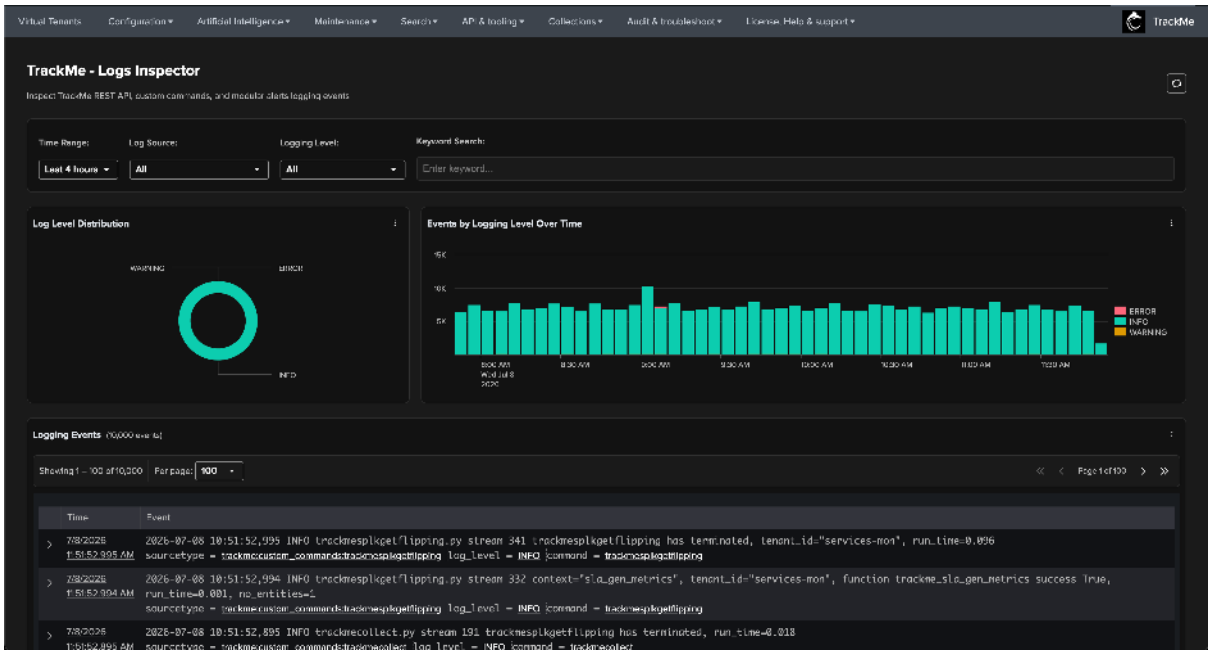
Log source	Sourcetype	Index
REST API endpoints	`trackme:rest_api`	`_internal`
Custom commands (trackers and backends)	`trackme:custom_commands:*`	`_internal`
Alert actions (Notable, Auto-ack, Stateful, etc.)	`modular_alerts:trackme_*`	`_internal` and `cim_modactions`
Health events (tracker healthy status)	`trackme:health`	TrackMe audit index (`trackme_audit*`)

Note

Ingest-time parsing is carefully handled with `TRUNCATE=0` on the TrackMe sourcetypes, so even very large events are indexed in full and never suffer from truncation.

10.1.2 Logs Inspector (recommended starting point)

The **Logs Inspector** dashboard is the fastest way to inspect TrackMe activity without writing any SPL. It is available from the navigation bar under **Audit & troubleshoot > Logs Inspector**, and it unifies the REST API, custom commands and alert action logs into a single point-and-click experience.



Use the filter bar at the top of the dashboard to narrow down exactly what you need:

- **Time Range** — any Splunk time range, defaulting to the last 60 minutes.
- **Log Source** — `All`, `Custom Commands` (`trackme:custom_commands:*`), `REST API` (`trackme:rest_api`) or `Modular Alerts` (`modular_alerts:trackme_*`). The dashboard automatically selects the correct index(es) for each source.
- **Logging Level** — `All`, `INFO`, `ERROR`, `WARNING` or `DEBUG`.
- **Keyword Search** — free-text filter applied to the raw events (for example a `tenant_id`, a component name or an error message fragment).

The dashboard then renders, for the selected filters:

- **Log Level Distribution** — a donut chart summarising the balance of `INFO` / `WARNING` / `ERROR` events, so a spike in errors is visible at a glance.
- **Events by Logging Level Over Time** — a stacked column chart (5-minute spans) revealing when errors or warnings occurred.
- **Logging Events** — a paginated table of the most recent events (up to 10,000), where each row can be expanded to reveal the full event, and the sourcetype, log level and command are surfaced as dedicated columns.

Tip

Every panel provides an **Open in Search** action, which hands the underlying SPL over to the Splunk search view so you can extend the investigation with your own commands.

10.1.3 Reviewing logs in Splunk Search

When you need the full power of SPL, you can query the TrackMe logs directly. The following searches are the building blocks used throughout support investigations.

REST API endpoints logging

All TrackMe REST API handlers log to a dedicated, automatically indexed log file:

```
index=_internal sourcetype=trackme:rest_api
```

TrackMe Broker logging

The add-on on a forwarder sends its log to `_internal`, and TrackMe's own side of a broker has its log on the stack:

```
index=_internal sourcetype=trackme:broker
index=_internal sourcetype=trackme:rest_api source=*trackme_rest_api_broker*
```

See *where to read what happened* and *the failure codes*.

Custom commands logging

Each custom command backend (trackers and helper commands) logs to its own log file, tied to a specific sourcetype following the `trackme:custom_commands:*` convention. Review all of them at once with:

```
index=_internal sourcetype=trackme:custom_commands:*
```

To target a single backend, append its name, for example the Health tracker:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth
```

Alert actions logging

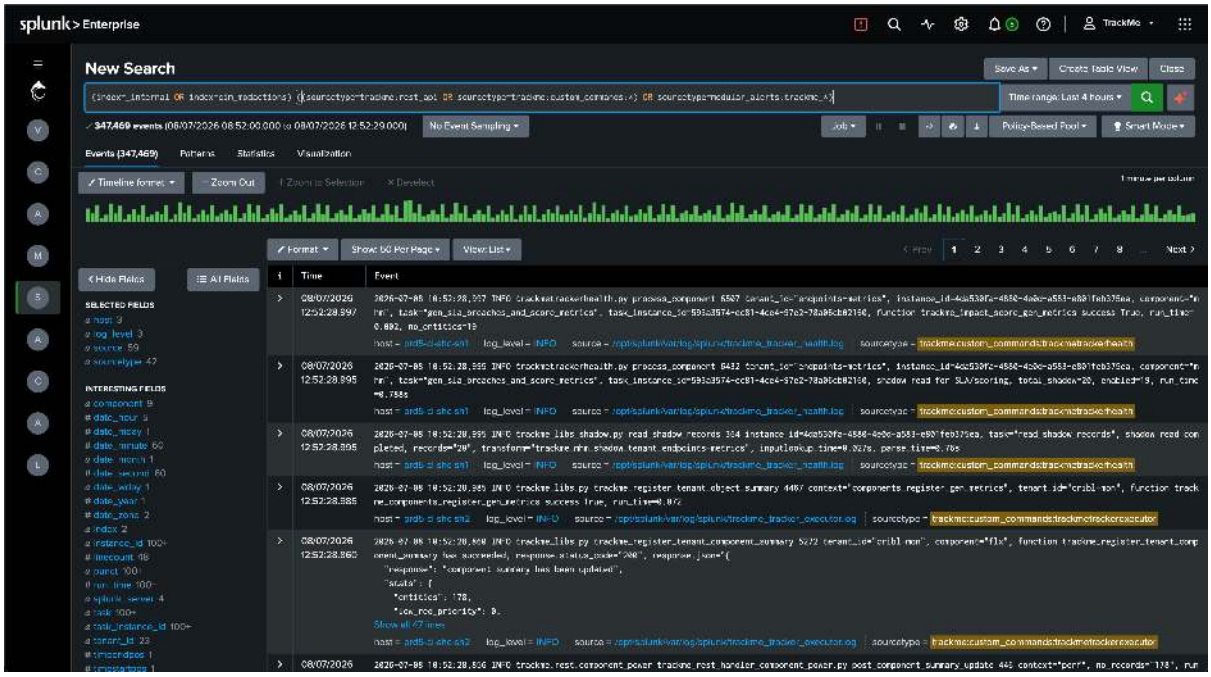
TrackMe ships multiple alert actions (Notable, Auto-acknowledge, Stateful alerts, Free-style REST call, and more). Each writes to its dedicated log file, and the Splunk modular alert framework additionally records execution in `cim_modactions`. Review all TrackMe alert action logs with:

```
(index=_internal OR index=cim_modactions) sourcetype=modular_alerts:trackme_*
```

The all-in-one search

To review **everything** TrackMe logs — REST API, custom commands and alert actions — in a single search, use the combined query below. This is the same search that powers the Logs Inspector dashboard:

```
(index=_internal OR index=cim_modactions) ((sourcetype=trackme:rest_api OR␣
↪sourcetype=trackme:custom_commands:*) OR sourcetype=modular_alerts:trackme_*)
```



Focusing on errors

Because every TrackMe event carries a `log_level` field, isolating problems is straightforward. Add `log_level=ERROR` to any of the searches above to surface only errors, for example across all custom commands:

```
index=_internal sourcetype=trackme:custom_commands:* log_level=ERROR
```

Or, to catch errors across the whole product in one pass:

```
(index=_internal OR index=cim_modactions) ((sourcetype=trackme:rest_api OR␣
↪sourcetype=trackme:custom_commands:*) OR sourcetype=modular_alerts:trackme_*)␣
↪log_level=ERROR
```

10.1.4 Health events and the Health tracker

Health events

TrackMe produces and indexes **health events** that track the healthy status of its trackers on a per Virtual Tenant basis. These events are stored in the TrackMe audit index under the `trackme:health` sourcetype and carry a `last_status` field (`success` / `failure`):

```
index=trackme_audit* sourcetype=trackme:health
```

The live, aggregated view of these statuses is surfaced by the **Audit - Operational Statuses** dashboard described below.

The Health tracker

TrackMe runs an important tracker that is automatically created for each Virtual Tenant. Among other responsibilities, it is in charge of triggering **schema upgrade** procedures when TrackMe is upgraded, and of generating a number of per-tenant metrics.

Its execution is logged under the `trackmetrackerhealth` sourcetype, with rich per-task detail (one line per task, including its run time). For example, to review the schema upgrade activity:

```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth task=
↪"schema_upgrade"
```

To track the run time of every task handled by the Health tracker:

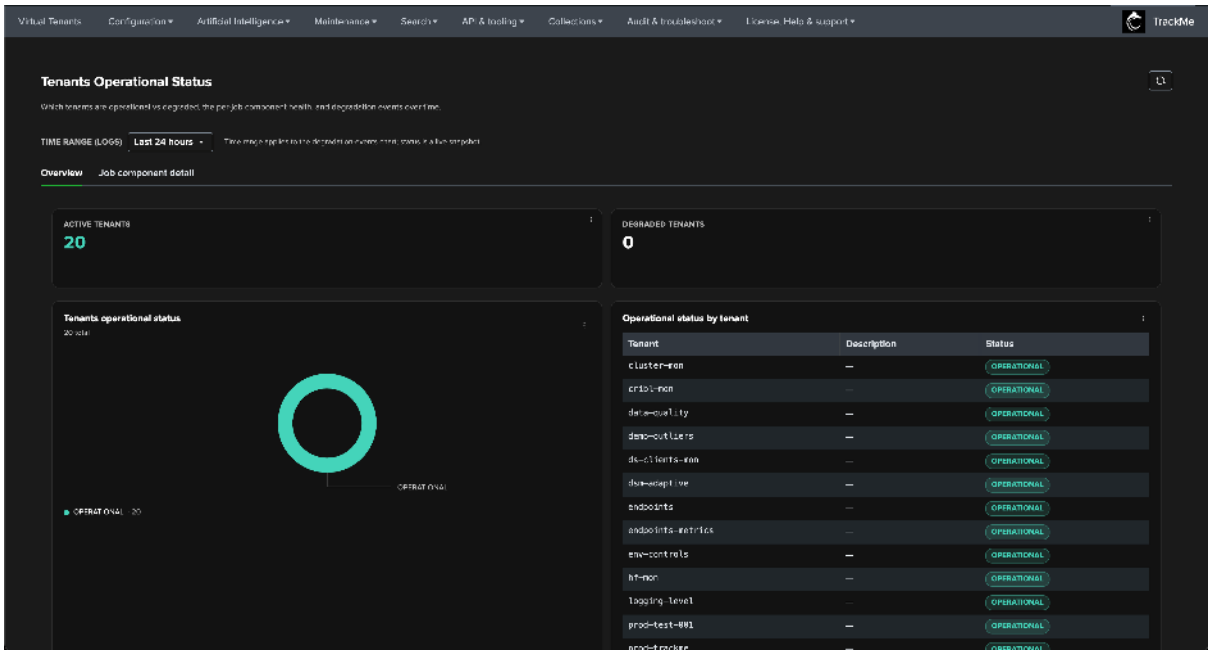
```
index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth␣
↪instance_id=* task_instance_id=* task=* run_time=* tenant_id=*
| table _time tenant_id instance_id task task_instance_id run_time _raw
| sort 0 - _time
```

10.1.5 Audit dashboards

Beyond the Logs Inspector, TrackMe provides a set of audit dashboards for troubleshooting and auditing specific behaviours. They are all reachable from the **Audit & troubleshoot** menu in the navigation bar.

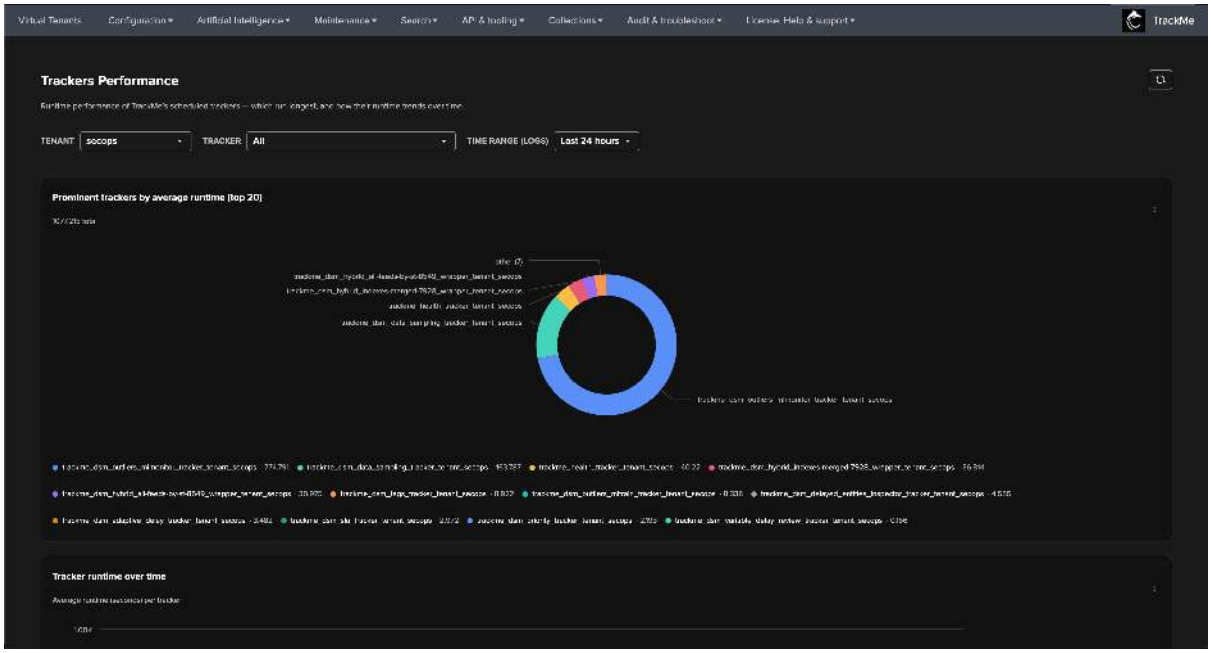
Audit - Operational Statuses

Shows which tenants are operational versus degraded, built from the components register and the health events. An **Overview** tab summarises the count of active and degraded tenants, the operational-status breakdown and a per-tenant status table, while a **Job component detail** tab drills into per-job component health. The status is a live snapshot; the selected time range applies to the degradation-events history:



Audit - Trackers Performance Deep Dive

Provides a comprehensive review of the trackers' run time performance — which trackers run longest and how their runtime trends over time. Filter by tenant, tracker and time range to see the most prominent trackers by average runtime:

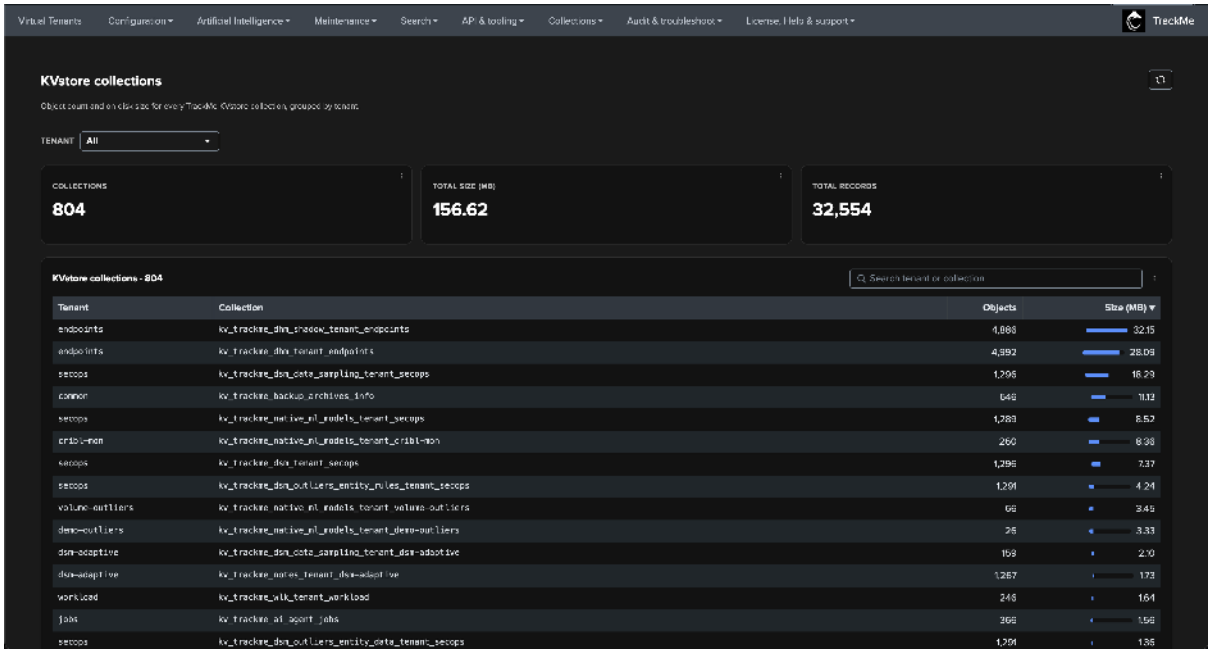


The lower panels chart the average runtime per tracker over time and provide a per-tracker performance table with the average, p95, max and latest runtime (in seconds), making runtime regressions and outliers easy to spot:



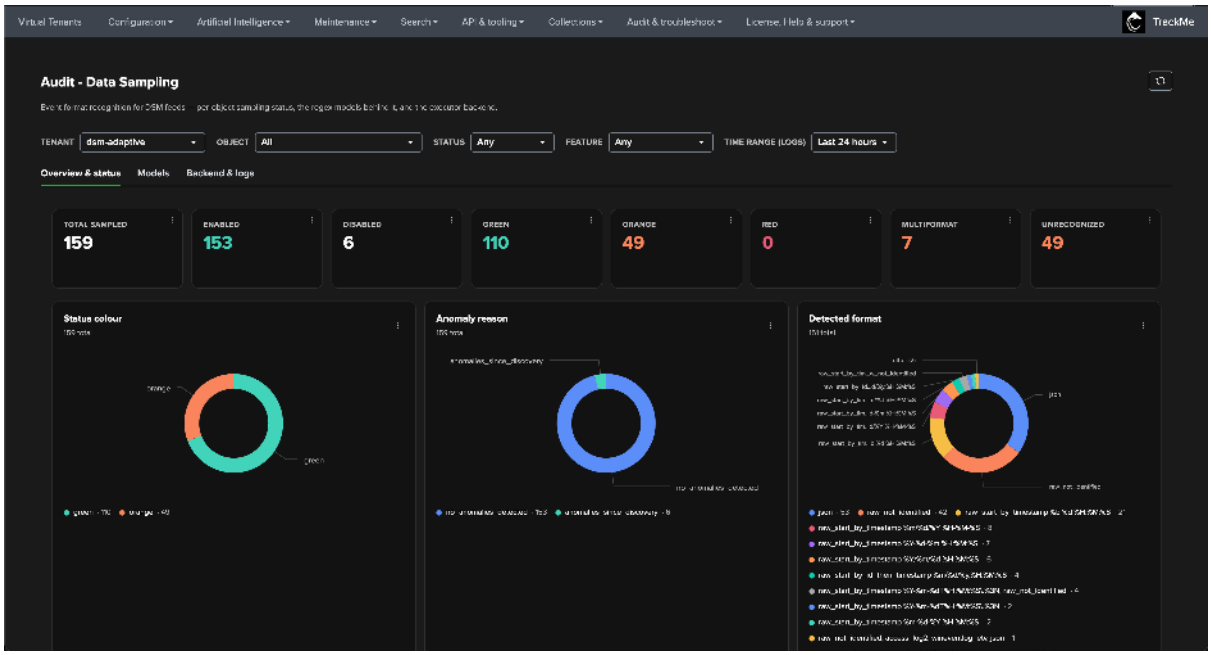
Audit - KVstore Collections

Reports the object count and on-disk size for every TrackMe KVstore collection, grouped by tenant. Use it to review the total number of collections, their aggregated size and records, and to identify the largest collections — filter by tenant or search by collection name:



Audit - Data Sampling

Investigates the event-format recognition (Data Sampling) feature for the splk-dsm component (part of splk-feeds): per-object sampling status, the regex models behind it, and the executor backend. Filter by tenant, object, status and feature, then review the sampling status distribution (green / orange / red), the anomaly reasons and the detected event formats across its **Overview & status**, **Models** and **Backend & logs** tabs:



10.2 Frequently asked questions (FAQ)

10.2.1 Virtual Tenants were created but the Virtual Tenants UI does not show any content

As of today, we do not support role inheritance when it comes to the RBAC configuration of the Virtual Tenants.

This means that a user needs to be a proper explicit member of either any of the administrative roles or the user roles for TrackMe to grant access to the tenant(s).

In short, you can review the workflow as follows:

trackmeload custom command

What TrackMe does is first calling the custom command called `trackmeload`, which can be called as follows:

```
| trackmeload
```

The custom command verifies the role membership of the current user and compares these with the roles defined at the level of the Virtual Tenant KVstore collection:

```
| inputlookup trackme_virtual_tenants | eval keyid=_key
| table tenant_id, tenant_owner, tenant_roles_admin, tenant_roles_user
```

If you enable the `DEBUG` logging level, the `trackmeload` custom command logs activity traces, similar to:

```
index=_internal sourcetype=trackme:custom_commands:trackmeload trackmeload.py
↪"check if username_role" OR "username="
```

Example:

```
2023-02-13 21:03:37,101 INFO trackmeload.py generate 105 username="admin", roles="[
↪'admin']"
2023-02-13 21:03:37,122 DEBUG trackmeload.py generate 147 check if username_role=
↪"admin" is in {'trackme_admin'} or {'trackme_user'}
```

If the current user is not an explicit member of the roles defined in the Virtual Tenant, or admin or sc_admin, the custom command will not grant access to these tenants.

10.2.2 How can I manage priority using a third-party source?

External priority management since TrackMe 2.1.10

- Since this release, TrackMe supports natively the external management of priority levels.
- This means that you can use any Splunk logic to define the levels of priority for your entities, and TrackMe will take this information into account and display it in the UI.
- Finally, you will also be able to manually update the priority level of an entity, and the external management will not override the manual update.
- See: *Managing priority externally*

10.2.3 How can I use TrackMe's entity status counts or entity records in my own dashboard?

Yes, everything the UI shows is available to your own searches and to external tools: the precomputed status counts through the `cached_stats` REST endpoint, the live entity table through the `trackmegetcoll` command, and, for KV-store access, the pre-processed **shadow** collections through `inputlookup` (recommended over the raw per-tenant collections, which are the decision maker's input rather than its output). If your dashboard lives outside the TrackMe app, first share the application globally and set TrackMe's default sharing level to global.

See *Consuming TrackMe data from your own dashboards* for the walkthrough.

10.2.4 Python error with module splunklib.results has no attribute JSONResultsReader

If you encounter this error when using TrackMe, then you have an issue in your environment which is improperly overriding the Splunk Python SDK used by TrackMe.

The symptoms lead to TrackMe various failures with the message:

```
module 'splunklib.results' has no attribute 'JSONResultsReader'
```

This happens if TrackMe is forced to use an incompatible and outdated version of the Splunk SDK for Python which does not have the JSONResultsReader module.

You can verify the version of the SDK which TrackMe uses by opening a Python interpreter and running the following command:

```
/opt/splunk/bin/splunk cmd python
```

Then paste the following code and press enter:

```
import sys, os
splunkhome = os.environ["SPLUNK_HOME"]
sys.path.append(os.path.join(splunkhome, "etc", "apps", "trackme", "lib"))

import splunklib.results as results
import pkg_resources

def get_package_version(package_name):
    try:
        return pkg_resources.get_distribution(package_name).version
    except pkg_resources.DistributionNotFound:
        return None

sdk_version = get_package_version("splunk-sdk")
if sdk_version:
    print("Splunk SDK for Python Version:", sdk_version)
else:
    print("The Splunk SDK for Python is not found or its version can't be␣
↪determined.")
```

The expected output should show the SDK version which should match what TrackMe is shipped with. The SDK version can be found in:

```
trackme/lib/splunklib/__init__.py
```

Example:

```
__version_info__ = (1, 7, 4)
```

This is also the case for every other Splunk application shipping the Python SDK. Once you have identified the application causing this, you can either manually upgrade the SDK or identify where a custom configuration such as a modification of environment variables (ex: PYTHONPATH) leads to this unexpected behavior.

You can also execute the following Python code in the interpreter to show the value of the PYTHONPATH from the perspective of Splunk:

```
import os
import sys

# Print the PYTHONPATH environment variable
pythonpath = os.environ.get('PYTHONPATH')
if pythonpath:
    print("PYTHONPATH:", pythonpath)
else:
    print("PYTHONPATH is not set.")
```

(continues on next page)

(continued from previous page)

```
# Print the current sys.path
print("\nsys.path:")
for path in sys.path:
    print(path)
```

CHAPTER

ELEVEN

VERSIONING AND BUILD HISTORY:

11.1 Release notes: trackme

TrackMe has transitioned to a new licensing model.

- The Free Community Edition has been discontinued.
- What this means for you:
  - Existing licensed customers are not impacted. Your current license continues unchanged.
  - Community users are automatically upgraded to a Foundation Edition trial for 90 days, with full functionality enabled.
- After the trial period:
  - TrackMe remains active and fully usable for monitoring
  - The platform enters read-only mode
  - You can continue to view data, dashboards, alerts, and history
  - Creation of new tenants, trackers, or entities will be disabled
  - To continue using TrackMe with full capabilities after the trial, a Foundation license is required.

Buy Foundation Edition: https://trackme-solutions.com/get-foundation/

Contact us: contact@trackme-solutions.com

11.1.1 Version 2.4.25 - build 1795090855 (19/11/2026)

Hint

TrackMe 2.4.25 — a maintenance release: fixes for Cribl Stream and for search head clusters, Fleet versions for Cribl Edge, collector selection for Splunk Forwarder Collectors and TrackMe Broker improvements.

- **Cribl Stream live metrics.** A Route with steady, low-volume traffic could make live runs fail with `invalid_metric_value`. Fixed: the tenant publishes again from its first run. See *Cribl Stream*.
- **Search head clusters.** Findings tenants' collection runs no longer fail now and then with `inventory_persistence_failed`, `workers_unavailable` or `uf_policies_unavailable`.
- **Cribl Edge: Fleet versions.** Edge versions, operating systems and architectures by Fleet, compared with the Leader, with recent upgrades. Every count of the Fleets table opens its nodes. See *Cribl Edge*.
- **Splunk Forwarder Collectors.** Choose the hosts the tenant follows, and pick universal forwarders as collectors from a list. See *Splunk Forwarder Collectors*.

- **TrackMe Broker.** Readable failure reasons, refusals reported at once, and brokered searches that use the memory of a direct one. See *TrackMe Broker* and its *failure codes*.
- **TrackMe Broker client API.** Another Splunk app can call an approved private HTTPS API through a broker. Disabled by default. See *Broker client API*.
- **Email delivery accounts.** A **Send test email** action.
- **Interface.** The app's pages share their common code, and the package is about 30 MB smaller. Topology Studio fixes.
- **Dependencies.** The regular update of the Splunk UI toolkit and other libraries.

SHA-256: 7f49c35a95c05799257e76d303e234970ee6368dac36441698378e9028d11280

Table 1: New Features

Issue Number	Description	Details
#4166	feature - Cribl Edge: Fleet versions	The Edge counterpart of the Splunk UF Fleet view, opened from the **Fleet versions** button of the Fleets panel, the *Not on the Leader version* tile or a Fleet's version drift. It shows the Edge versions, operating systems, releases and architectures of the selected Fleets or of one Fleet, the versions compared with the Leader, the version mix of each Fleet, the nodes that changed version over the last 24 hours, 7 days and 30 days with their from and to versions, and the history of versions and operating systems over time. The figures are counted when the node directory is built, never by a scan of the nodes, so the view opens at once at any fleet size. A Fleet published before the upgrade is counted by the next live run.
#4178	feature - Splunk Forwarder Collectors: choose the followed hosts and pick collectors in the wizard	The discovery table gains an **Include** column: unchecking a collector excludes it by its exact name, and checking a utility node adds it to the new *Included utility nodes* setting (names or patterns). A host excluded by a pattern stays listed and says why it is locked. **Add collectors** lists the universal forwarders seen sending, the busiest first, page by page with a name filter, and the forwarders picked become collectors of a declared group. This is how intermediate universal forwarders used as syslog or file collectors are added: they are declared by the operator, never detected. The same dialog serves the wizard, Settings and the Inventory. In the wizard, hosts and groups share one Collectors card: groups show as pills that open their settings, and a Group column places each host. Existing tenants keep their source and history.
#4173	feature - TrackMe Broker: a client API for other Splunk apps, disabled by default	A separate Splunk app on the stack can call a private HTTPS API that an administrator approved, through a paired TrackMe Broker, with its own destination credentials: TrackMe stores none of them. The client API is disabled by default. An administrator turns it on under **Configuration → System settings → TrackMe Broker → Enable the broker client API**, or with `POST /services/trackme/v2/broker/client/admin/settings`. Client pools and accounts stay on that administrative API. Each account names one broker, an origin, a path prefix, the methods and the Splunk roles allowed to call it; callers also need the new `trackme_broker_client` role, which grants no access to any tenant. Client calls use their own queues and pools, bounded sizes and sealed exchanges, and the broker add-on's protocol is unchanged. Nothing changes until an administrator enables it. See *Broker client API*.

Table 2: Enhancements and Changes

Issue Number	Description	Details
#4166	enhancement - Cribl Edge: every count of the Fleets table opens the nodes behind it	Disconnected, a version, Not on configuration and Without metrics open the list of those nodes, where each node opens its detail and **Show in Edge Nodes** carries the filter to the tab. Not on Leader version opens the Fleet's versions, and I/O unhealthy opens the Fleet's warning and trouble Sources and Destinations with the nodes reporting them. Edge Nodes gains architecture, release and metrics filters.
#4166	enhancement - Cribl Edge: System activity reads the System Metrics index first	When the tenant has a System Metrics index, a node's System activity reads it with one search you run, instead of asking the Leader to query the node: the same charts, tiles and facts, in seconds, with *Open in Search* on each chart. Without a System Metrics index, or when it holds nothing for the node, the read goes through the Leader as before.
#4157	enhancement - TrackMe Broker: clear failures, readable reasons and a live Manage Brokers page	Manage Brokers refreshes every 30 seconds, so a forwarder's last seen time and the broker's state stay current while the page is open. Each failed or expired exchange shows a readable reason and when it ended, and those that ended before the broker's last successful exchange are marked *Before the last success*. An administrator can clear a broker's failures: the action is confirmed and audited, and it leaves queued and in-flight calls untouched. A late progress call from an exchange that already finished is answered normally rather than refused. The add-on and its protocol are unchanged.
#4183	enhancement - Splunk Cloud wording names TrackMe Broker	Help texts of Splunk Forwarder Collectors and Splunk UF Fleet said that a Splunk Cloud search tier cannot reach on-premises collectors or Deployment Servers. Since 2.4.24 a remote account set to connect through a TrackMe Broker reaches them, and the texts now say so.
PR #4195	enhancement - The app's pages load through shared chunks	Each page script is now a small loader, and the code several pages have in common is downloaded once instead of being copied into every page. A first visit downloads about the same as before, later pages reuse what the browser already holds, and the package is about 30 MB smaller. Chunk names are stable, so an upgrade overwrites them and nothing piles up in the app directory.
#4169	enhancement - Dependencies	Splunk UI toolkit (`@splunk/react-ui` 5.18.0, `@splunk/react-icons` 5.17.0, `@splunk/react-page` 8.5.0 and related packages), `date-holidays` 3.39.2 with the regenerated bank holiday catalog (saved calendars keep their rules), and development tooling (PRs #4170, #4193, #4205, #4228).
#4210	enhancement - TrackMe Broker: a brokered response is read chunk by chunk	The transport held a whole remote response in memory, peaking at more than twice its size before the caller read a byte, and concurrent brokered remote searches on Splunk Cloud could be stopped by the platform's memory limit. A response is now read one chunk at a time as the caller consumes it, so a brokered call uses the memory a direct one does. The add-on and the wire format are unchanged.
#4217	enhancement - Email delivery accounts: send a test message	**Test connectivity** stops after the connection, TLS, login and a no-op command, so a refused sender or a recipient policy only showed when a real alert email was sent. The row menu of the email delivery accounts now offers **Send test email**: one message to one recipient, which must match the account's allowed recipient domains before any connection is made. The stored password never leaves the server. A completed attempt updates the account's last connectivity check.
#4221	enhancement - Topology Studio: the view title in the page header, the legend shown or hidden from the toolbar	On an open view, the title and the unsaved changes marker sit in the page header beside the release chip, so the toolbar keeps its room; a long title is shortened. A **Show legend / Hide legend** button in the toolbar shows or hides the legend of a view holding Cribl, Splunk HEC, Splunk S2S, Splunk UF Fleet or Splunk

Table 3: Bug Fixes

Issue Number	Description	Details
#4199	bug - Cribl Stream: live metrics rejected on Splunk's rounding of its aggregates (`invalid_metric_value`)	Every live run of a Cribl Stream tenant could fail with `invalid_metric_value`. Collection health then showed *Live metrics: Failure* and *Monitoring policies: Failure*, and the policies evaluated stale evidence. TrackMe checked Splunk's aggregates against each other: the sum between the count times the minimum and the count times the maximum, the minimum below the maximum, the latest value between them. Splunk computes these figures together over the same samples, so they always agree, but it prints them rounded to six decimals, and the check could only fail on that rounding. It did with a Route carrying steady, low-volume traffic, and one such row withheld the live figures of every object, from the moment that Route started reporting. The comparison is removed, both when the figures are read and when the stored snapshot is validated. Each value is still checked on its own: a number, not negative, with at least one sample. No action is needed after the upgrade: the tenant publishes again from its first run.
PR #4212	bug - TrackMe Broker: a refusal is reported at once, with its reason	A remote account whose endpoint the broker's forwarders do not declare as a target was retried for 30 seconds, then reported as a network failure and an unexplained `HTTP 500` in the account test. Refusals that asking again cannot change (`target_not_allowed`, an unknown or disabled broker, the account refusals) now stop at the first answer and say what to do, for example to add the destination to the targets on the TrackMe Broker add-on page of each forwarder. Failures that may recover (`broker_offline`, `target_unreachable`, timeouts) still retry. Webhook accounts gain readable reasons for the broker codes.
PR #4209	bug - Fixes from the review of the 2.4.25 release	Cribl Edge: *Open in Search* of a node's System activity showed the CPU only, and now keeps memory, network and disk; in Fleet versions, nodes without a known version (not reported, or past a Fleet's crossing cap) are no longer counted as off the Leader version and are shown apart. Splunk Forwarder Collectors: the wizard's preview declares a forwarder exactly as the saved configuration would (assignments first, then, when collectors join by their patterns, the first matching group); moving a host no longer leaves a group the save refuses; the group dialog asks before discarding an Ungroup. TrackMe Broker: Manage Brokers no longer fails for a broker named like `constructor`.
#4213	bug - Findings tenants: collection runs failed from time to time on a search head cluster	On a search head cluster, Cribl Stream and Cribl Edge inventory runs failed with `inventory_persistence_failed`, Cribl live runs with `workers_unavailable` and Splunk UF Fleet policies runs with `uf_policies_unavailable`, the next run succeeding with nothing wrong. A fresh Splunk Cloud stack could fail the first inventory run of a new Cribl Stream tenant the same way. These stores verify a KV write by reading it back right away; each cluster member reads from its own KV copy and a change reaches the others asynchronously, so the read could come before the change. A read-back that disagrees is now read again after short waits, up to a few seconds; a write that was really lost still fails.
PR #4215	bug - TrackMe Broker: users with an `@` in their name were refused (`account_access_denied`)	An interactive action through a brokered remote account, such as a Flex Objects wizard simulation, was refused for a user named like `user@example.com`, typically a SAML user: the name was encoded twice when TrackMe read the user's roles, so Splunk did not find the user. It is now encoded once. This also applies to the broker client API and the Cribl tenant owner check.
#4223	bug - Topology Studio: View status and View details opened	In full screen, the View status and View details dialogs opened behind the canvas and only appeared after leaving full screen

11.1.2 Version 2.4.24 - build 1794268494 (10/11/2026)

Hint

TrackMe 2.4.24 — Splunk Forwarder Collectors, a new tenant type for the forwarders that collect your data, and TrackMe Broker, which takes TrackMe on Splunk Cloud to your on-premises destinations.

- **Splunk Forwarder Collectors.** A new dedicated tenant type for the forwarders used as collection points: DB Connect, cloud and API add-ons, scripted and modular inputs, and the forwarders that monitor the files written by rsyslog or syslog-ng. Every input is listed as it is configured, for any add-on, and TrackMe answers two questions separately: does it run, and does it work. Errors and rhythm are judged against each input's own past, nothing is installed on the collectors, and the tenant brings the findings model of the other dedicated tenants with its own AI Agent. See *Splunk Forwarder Collectors*.
- **TrackMe Broker.** TrackMe on Splunk Cloud cannot open a connection into your network. An add-on on a Heavy Forwarder inside it now opens every connection itself and carries TrackMe's calls, so remote Splunk accounts, self-managed Cribl Leaders and webhook receivers can be reached from Splunk Cloud or from a tightly controlled network. Pairing uses a short-lived code and a fingerprint check, every call is signed, requests are sealed, and several forwarders form one broker. See *TrackMe Broker* and the *add-on release notes*.
- **Splunk S2S, Splunk UF Fleet and Splunk HEC.** An Inventory with stop and resume monitoring of any instance, Splunk's own health report as a Health column, Open in search on every setup discovery, and no more false findings from Splunk 9.4 telemetry. Collection reads no longer stop at a row ceiling, so Splunk UF Fleet keeps working on estates above 100,000 agents. See *Splunk UF Fleet*.
- **At scale.** Every scheduled search has what is left of its run's time instead of a fixed number of seconds, a Splunk UF Fleet inventory rebuild reads only what changes by the hour (four to six searches instead of eleven), and a Cribl Stream tenant no longer stops at 20,000 monitored objects.
- **User Activity Monitoring.** No false partial coverage from an Activity scope, and a coverage chip and time budget help that match what the runs did. See *User Activity Monitoring*.
- **Security and platform.** Everything TrackMe computes itself uses SHA-256 or stronger, for FIPS deployments. Turning Verify SSL certificate off no longer raises a Configuration Guardian alert, and dependencies carry every available security fix.

SHA-256: 79c88e9cfe1f82ae18b49cdb5d39c44db0d400a341d66440cd0274d1b572d1b8

Table 4: New Features

Issue Number	Description	Details
PR #4095	feature - Splunk Forwarder Collectors, generally available	A new dedicated tenant type on the findings runtime for the forwarders used as collection points: heavy forwarders running DB Connect, cloud and API add-ons, scripted and modular inputs, and the forwarders, universal forwarders included, that monitor the files written by rsyslog or syslog-ng. Those file collectors are declared by the operator on a group, never detected. Monitoring lists every input as it is configured, for any add-on, from Splunk's own records and the optional API accounts, while what runs has its own Processes view. Whether an input runs and whether it works are two answers, and its errors and rhythm are judged against its own past, so a feed that goes quiet or starts failing is reported without a threshold to tune. An operator can select the inputs that must stay enabled, and take one input out of one input policy while it stays monitored. Nothing is installed on a collector and its remote accounts are optional, so every policy works from the telemetry alone, as a Splunk Cloud search tier needs. Collector API accounts are checked as they are chosen. The tenant brings built-in and custom policies, Finding groups, notifications, findings actions, traffic models, Collection health, an Inventory with stop and resume, Topology Studio, backup and restore, and its own AI Agent. It runs 12 scheduled reports per tenant, the most of any tenant type, so the Splunk Cloud search concurrency quota applies (PRs #4109, #4111, #4122, #4135, #4137, #4139, #4141).
PR #4092	feature - TrackMe Broker: reach on-premises destinations from Splunk Cloud	The TrackMe Broker add-on runs on a Heavy Forwarder inside your network. It opens every connection itself, to the stack's REST API, and carries the calls TrackMe queued for it. Remote Splunk accounts, self-managed Cribl Leaders and webhook receivers select a broker in their Connectivity setting, and an account that selects none keeps its direct connection. A forwarder is paired by a short-lived code entered in Manage Brokers and a fingerprint check. Every call it makes is signed, and requests and responses are sealed, so a destination's credentials are readable only by the paired forwarders. Several forwarders form one broker and share its queue, so it keeps working when one stops, and Configuration Guardian reports a broker that is degraded or offline. The stack renews the forwarder's token itself, and Remove and Delete revoke it. The forwarders' account needs only the new `trackme_broker` role, and on Splunk Cloud the forwarder's egress address goes on the `search-api` allow list. Validated on a Splunk Cloud stack with forwarders on Splunk 9.4 and on a FIPS deployment (PRs #4110, #4114).
PR #4126	feature - Splunk S2S and Splunk HEC: an Inventory with stop and resume monitoring	Splunk S2S (Receivers, Intermediates) and Splunk HEC (Endpoints) gain an Inventory tab, aligned with Splunk Forwarder Collectors. Each row shows whether the instance is monitored, and a power or admin user can stop or resume its monitoring. A stopped instance leaves every monitoring view at once and gets no live read or decision, its open findings are resolved as the operator's decision, and it stays in the Inventory so it can be resumed.
PR #4134	feature - Splunk UF Fleet: an Inventory with stop and resume monitoring of agents and Deployment Servers	Splunk UF Fleet gains an Inventory tab with Deployment Servers and Agents, and stop or resume monitoring of any of them. Unlike Forget, which retires a disconnected agent until it connects again, stopping monitoring works in any state and stays until it is resumed.
PR #4129	feature - Splunk's health report as a Health column on Collectors, S2S and HEC	A Health column on Splunk Forwarder Collectors, Splunk S2S (Receivers, Intermediates) and Splunk HEC (Endpoints) shows Splunk's own health report of each instance, the tree behind the Splunk Web health icon. Each pill opens a Health view with the state, the yellow and red indicators, every feature of the latest

Table 5: Enhancements and Changes

Issue Number	Description	Details
PR #4125	enhancement - Splunk UF Fleet: the operating system agents report when they phone home	A start line names only an agent's kernel. Each agent also names its system when it phones home, on the Deployment Server's access log, so a new hourly job reads it and Splunk UF Fleet can say `Windows Server 10` rather than a kernel. It is shown in Fleet versions (Operating systems tab), the agent's Identity tab and the fleet lookup, and Collection health gains the *Deployment Server history* tracker. Fleet versions also groups the operating systems by family, type, release and Splunk version (PR #4120).
PR #4156	enhancement - Splunk UF Fleet: an inventory rebuild reads only what changes by the hour	A rebuild of the agent directory sent eleven searches about every hour, among them 30 days of Deployment Server client records and a week of phonehomes, and one that did not finish failed the whole rebuild. The client records are now read forward from where the last read stopped, and the phonehome baseline once a day, by the hourly *Deployment Server history* job (formerly *Reported systems*). The inventory takes both from it without a search, and reads them itself only while the job has none for it. The receivers' versions are read once a day, and at once for a receiver whose agents run a newer release. The probe of the `_metrics` mirror is held a day except while the mirror is complete, which is checked at every rebuild. A scheduled rebuild now sends four searches on the internal form and six with the mirror.
PR #4145	enhancement - Open in search and one discovery contract for every Splunk tenant's setup	The setup discoveries of Splunk S2S, Splunk Forwarder Collectors, Splunk HEC and Splunk UF Fleet, in the creation wizards and in Settings, gain Open in search next to Discover again: it opens the discovery's main search exactly as it ran, with its time window. The four discoveries now follow one contract (PR #4138).
PR #4132	enhancement - Splunk HEC: the endpoint view in tabs	The Splunk HEC endpoint view uses tabs, like a Splunk S2S host and a collector, instead of one long scroll. A tab's searches run only when it opens, and the Refused requests and Queues tiles open their tab.
#4084	change - FIPS: SHA-256 or stronger in everything TrackMe computes itself	The User Activity Monitoring resource search and the Workload orphan check were the last searches using SPL `md5()`. Neither was a security use, but TrackMe honours FIPS in everything it does itself, so both now use SHA-256, and a test guard keeps TrackMe's own code to FIPS-approved primitives. After the upgrade, at most one UAM resource window per tenant is read twice if a run was cut at that moment (PRs #4087, #4088, #4093).
PR #4121	change - Configuration Guardian: Verify SSL certificate off no longer raises an alert	Turning Verify SSL certificate off on a Cribl connection or a Webhook account is the administrator's choice, not a misconfiguration. The checks `cribl_connection_tls_unverified` and `webhook_account_tls_unverified` are retired, and their stored alerts, settings and snoozes are removed by the next daily health run.
PR #4115	enhancement - Splunk UF Fleet: what Excluded receivers does and does not do	The help of Excluded receivers now says that it removes receivers from the agent discovery and the connection measures, not a destination: failures in an agent's own logs are still reported.
#4085	dependencies - Dependency updates and security fixes	Consolidated dependency updates with every security fix available, `@babel/core` 7.29.13 across every package, and a build check that fails when the frontend lock file drifts from the manifests (PRs #4086, #4112, #4113, #4144).

Table 6: Bug Fixes

Issue Number	Description	Details
PR #4094	bug - Splunk UF Fleet, Splunk S2S and Splunk HEC: collection failed with `search_row_limit` on large estates	An estate above 100,000 agents upgraded to 2.4.23 and every Splunk UF Fleet job still failed, now with `search_row_limit`: the search runner refused any result above 120,000 rows. A read that grows with the estate now has no row or byte limit, every row Splunk returns is read, and the job's time budget is the only bound. This also applies to the identity and HEC receiver reads, the agent discovery's sender counts, and each agent's receivers, which are now its latest ones with their exact count (PRs #4101, #4136, #4140).
#4100	bug - Splunk forwarding: false findings from Splunk 9.4 telemetry	Splunk 9.4 writes quoted values in `metrics.log`, which the indexed reads cannot see. A 9.4 sender no longer drops out of the writers of its destination, so a fleet mostly on 9.4 no longer reads a destination as unreachable, and a sender with no readable GUID no longer counts for receiver churn. Splunk Forwarder Collectors reads the volume of a 9.4 host again by its lines at search time, where it read 0 (PRs #4108, #4111).
#4096	bug - Splunk S2S: a 9.4 intermediate forwarder read as a receiver	On the internal telemetry form, an instance that receives and forwards now reads as an intermediate, whatever version wrote its `metrics.log` (PR #4099).
#4102	bug - Splunk S2S: fully qualified intermediates each got a pool of their own	Numbered, fully qualified intermediate forwarders of one domain now share one automatic pool, while another domain stays another tier (PR #4103).
#4131	bug - Settings listed the instances an operator stopped	The Settings discovery of Splunk S2S, Splunk Forwarder Collectors and Splunk HEC no longer lists the instances stopped in the Inventory, nor counts them in the proposed pools, and says how many it left out. A stopped instance keeps its hand assignment for when it resumes (PRs #4130, #4133).
PR #4120	bug - Splunk UF Fleet: Windows agents read `Windows 0`	The start line of a Windows agent was parsed against the wrong layout, so many agents read `Windows 0` or `Windows 3`. They now read `Windows (kernel 10.0)`, and stored values are corrected by the next directory rebuild, up to an hour after the upgrade.
#4097	bug - User Activity Monitoring: an Activity scope reported false partial coverage	On a tenant with an Activity scope, a search head that the scope stopped reading could make a whole window read Partial on Resource usage. A name Splunk states now stays named when its search has no evidence from that head (PR #4098).
#4106	bug - User Activity Monitoring: the coverage chip said complete, and the time budget help gave wrong advice	The coverage chip now also reads the latest Resource usage, Account signals, Data usage and Policies runs, and names the one that failed or stopped short. The time budget help was checked against the trackers: its figures now match them, and it no longer advises raising a budget a scheduled run cannot use or changing the Activity scope to catch up (PR #4107).
#4104	bug - The Collect now and Run now messages of the findings tenants stayed on screen	The Collect now banner of Splunk S2S and Splunk UF Fleet now clears on a manual refresh, and the Run now notice in Collection health clears once the job shows a newer run (PR #4105).
#4117	bug - REST handlers answered 500 right after a token was revoked	A request that arrives with no session token, as seen for a few seconds after a token is revoked, now answers 401 instead of 500 (PR #4118).
PR #4127	bug - History charts showed raw figures with no unit	Hovering a history chart point showed figures such as `74,144.31288888889`. A chart now reads in the unit of its largest value, named in the series and on the axis.
PR #4147	bug - Help texts that left out some tenant types or named a retired policy	The Manage AI Agents automation page and the Splunk Web links help now name every dedicated tenant type, and the Bank Holidays help for Splunk UF Fleet no longer lists a retired policy among those held by default (PR #4146).
PR #4155	bug - Scheduled searches stopped at a fixed number of seconds on large estates	Some reads of scheduled jobs had a fixed limit, from 30 seconds to 15 minutes, whatever their job's budget: Cribl Stream traffic model training, Cribl Edge node CPU and memory, the policy lookup reads, and several User Activity Monitoring reads (Data usage confirmations, backfill scans, privileged users, resource

11.1.3 Version 2.4.23 - build 1793812942 (04/11/2026)

Hint

TrackMe 2.4.23 — a hotfix release: Splunk UF Fleet, Splunk S2S and Splunk HEC collection at scale, and User Activity Monitoring Data usage linked to any VOL or DSM tenant.

- **Splunk UF Fleet and Splunk S2S: collection as dedicated jobs.** At scale, every live run of a Splunk UF Fleet tenant failed with `search_timeout`: each search had a fixed cap of 60 seconds, one read past it cancelled the run, and the next run repeated the same read. The reads now run as dedicated jobs, each with its own report, schedule, time budget and Collection health line, and no search has a fixed timeout any more on Splunk UF Fleet, Splunk S2S or Splunk HEC: a search may use what is left of its job's budget. A live run never builds the directory, and the day-long reads run hourly or every 15 minutes instead of every 5 minutes. See *Splunk UF Fleet* and *Splunk S2S*.
- **Splunk S2S, Splunk UF Fleet and Splunk HEC: a short setup discovery.** The creation wizards ran up to eight searches over 24 hours to 60 days, and the first timeout failed the whole discovery. They now run at most two searches per product over the last 15 minutes and answer with counts and roles. **Propose groups** is its own action in the Splunk UF Fleet settings. See *Splunk HEC*.
- **User Activity Monitoring: Data usage links any VOL or DSM tenant.** The Data usage settings list every VOL and DSM tenant you can see, whatever account it collects through, and select one VOL tenant and every DSM tenant automatically; a VOL tenant on a remote account now returns its volume and cost. See *Data usage*.

SHA-256: 0ae4b21a158b1fbbd918db089235454b1cc172e322569e09c59234308ad50877

Table 7: Bug Fixes

Issue Number	Description	Details
PR #4075	bug - Splunk UF Fleet and Splunk S2S: the collection failed with `search_timeout` at scale	The first production feedback on Splunk UF Fleet: every live run failed with `search_timeout`, by schedule or by hand, with most of its budget unused. Each search of a Splunk UF Fleet or Splunk S2S run had a fixed cap of 60 seconds (45 for the qualification and the probes), one read past it cancelled the job, nothing carried over, and no setting could change it. The collection is now split into dedicated jobs, each with its own report, schedule, time budget (its interval less 30 seconds), lock, receipt, published facts and Collection health line: `trouble`, `destinations`, `identity`, `deployment` and `deployment_detail` on Splunk UF Fleet (seven scheduled reports per tenant instead of two), `pipeline`, `health` and `destinations` on Splunk S2S (six instead of three). The live run keeps the connection and traffic reads, assembles the latest facts into the one snapshot every consumer already reads, and evaluates the policies: views, topology, evidence and findings read what they read before. No search has a fixed timeout any more on Splunk UF Fleet, Splunk S2S or Splunk HEC: it may use what is left of its job's budget. A live run never builds the directory (the inventory job does, with its own budget): Splunk S2S and Splunk HEC live runs wait for it, a Splunk UF Fleet live run admits the connected agents and its policies judge nothing until the first directory; a first manual live run starts the inventory beside it. The day-long reads (the day's input errors, the Deployment Servers' own errors, the deployment failures) run hourly or every 15 minutes, after the window reads, and are kept between reads. A job that misses a slot makes the policies hold, never count twice; facts older than three runs read as not read. Collection health gains `job_never_ran`, `job_not_running` and `job_failing`. Splunk HEC, Splunk S2S and Splunk UF Fleet tenants run the new `trackmesplunkcollect` command (log `trackme_splunk_collect.log`), Cribl tenants keep `trackmecriblcollect`; the health tracker rewrites the reports, no migration. Splunk HEC keeps its one live job: it gets the budget rule and the directory rule, not the split. Known: with Scheduled collection off, it takes two **Collect now** before anything is judged; a least-privilege tenant owner may need a higher search quota, since up to five jobs can overlap.
PR #4074	bug - Splunk S2S, Splunk UF Fleet and Splunk HEC: the setup discovery timed out at scale	The discovery of the creation wizards ran eight or more sequential searches, mostly over 24 hours and up to 60 days, with 60 seconds each, and the first timeout failed the whole run (Splunk HEC ran two searches, over 24 hours too). It is now a short, high-level read: at most two searches per product over the last 15 minutes, 5 minutes per search. Splunk S2S lists one row per receiver (role, senders counted on the indexers, inbound volume) and reads the indexer clusters; Splunk UF Fleet counts the senders and receivers per sender type and lists the Deployment Servers; Splunk HEC lists the endpoints and the tokens active in the last 15 minutes. The Splunk UF Fleet wizard shows counts instead of an agent table (no versions, platforms or last connection: the tenant's own collection reads further back and lists every agent), the Splunk S2S rows no longer carry the heavy or universal type, and a receiver or endpoint silent for 15 minutes is not in the preview. **Propose groups** is its own action in the Splunk UF Fleet settings (three reads; server classes keep their 60-day window), and a manager account is checked when it is chosen and on **Discover again**. Removed: `trackmehecdiscover hours=`, `trackmeufdiscover heartbeat_fallback=` and `connected_window_seconds=`. The tenants' scheduled collec-

11.1.4 Version 2.4.22 - build 1793748743 (03/11/2026)

Hint

TrackMe 2.4.22 — Splunk S2S and Splunk UF Fleet, two new tenant types for Splunk forwarding, and User Activity Monitoring Data usage: who reads which index, and what the unread data costs.

- **Splunk S2S and Splunk UF Fleet.** Two new dedicated tenant types monitor how a Splunk deployment forwards its data, from what Splunk already writes: nothing is installed on a forwarder and nothing new is ingested. Splunk S2S watches the receivers, intermediate forwarders and pools (an indexer cluster is a pool, one per site of a multisite cluster, and the intermediate forwarders are grouped by host-name family), their traffic and their TLS and acknowledgement posture. Splunk UF Fleet follows the forwarding agents, their groups and the Deployment Servers that manage them. Both bring the findings model of the other dedicated tenants, with their own AI Agent. See *Splunk S2S* and *Splunk UF Fleet*.
- **User Activity Monitoring: Data usage.** Every audited search is resolved to the indexes it actually reads, through its knowledge objects, and weighed with the volume and cost of each index: globally, per index, per account and per knowledge object. An index is only called unread when the evidence supports it. See *Data usage*.
- **VOL: exact license figures.** Late license events are never lost, closed license days are reconciled against Splunk's own Usage records with every difference explained, and volumes use binary units by default. See *Volume Outliers*.
- **Cribl Edge and Cribl Stream.** Cribl Edge gains object monitoring with activity trends; Cribl Stream routes, pipelines and destinations open with their connected objects. See *Cribl Edge* and *Cribl Stream*.
- **Every findings tenant.** Custom policies can drive findings actions and automations, links in notifications use a configurable Splunk Web root URL, and searches trust Splunk's own job state, so a busy or restarting indexer no longer causes re-reads. See *Findings actions*.
- **Workload.** Entities whose scheduled search was deleted from Splunk are flagged and retired automatically. See *Workload*.
- **Reliability and compatibility.** Lighter User Activity Monitoring on large search head clusters, restore of deleted findings tenants, a clearer Configuration Guardian, Splunk 9.x TLS contexts, the openai SDK 3.x, and a DOMPurify security update.

SHA-256: a46821075d99999b3021a7340ead7f980804863a224a23a836e1e99ddc408314

Table 8: New Features

Issue Number	Description	Details
PR #3947	feature - UAM: Data usage — which indexes are read, by whom, and what the unread data costs	A new Data usage tab on User Activity Monitoring tenants. A dedicated collector resolves every audited search to the indexes it names (literal terms, macros, event types, saved searches, data models; wildcards kept as references), counted once, with the indexers' own confirmations on distributed deployments and the knowledge objects that declare each index. The account view gains a Data accesses tab: which indexes the account read, and when. Five views follow: where the accesses or the data go (flow or treemap, naming the top readers and indexes), value and cost (volume against reads, cost of disuse, a decision list), index activity with each index's readers and access timeline, knowledge objects declared against observed (in use, not run, not due in this period, building blocks, dashboards), and the evidence. Review candidates are decided across the whole indexing layer, and no index is called unread before the collection covers the period or seven days of it. Volume and cost come from VOL and DSM tenants matched automatically per search head tier. Every read runs as the caller's own search jobs, without a time budget. Reads follow the readers' Splunk role search restrictions: a wildcard reaches only the indexes its reader can search, a search the roles deny is never a read, and a search naming no index shows what its account's roles give it. Writes made through summary indexing and the Log event alert action are recorded, the views open in Search, and How Data usage works is an in-depth reference (PRs #4030, #4033, #4044, #4047, #4054, #4055).
PR #3973	feature - Splunk S2S and Splunk UF Fleet, generally available	Two new dedicated tenant types on the findings runtime. Splunk S2S watches the receiving side of forwarding: receivers, intermediate forwarders and pools, indexer clusters as pools (one per site of a multisite cluster), automatic pools of intermediate forwarders, back pressure and its bottleneck, traffic models per pool, and the TLS and acknowledgement posture (22 policies). Splunk UF Fleet follows the forwarding agents (connected, disconnected, dormant, expected), their groups and enrichment, and the Deployment Servers that manage them (28 policies). Both come with creation wizards, custom policies, notifications, findings actions, Collection health, Topology Studio, backup and restore, and their own AI Agent. Hosts can be assigned to a pool from their pool pill, Splunk Cloud receivers form automatic pools, a blocked host shows its queue history, a disconnected agent can be forgotten, and the older identity of a rebuilt host is forgotten by TrackMe (PRs #4001, #4002, #4003, #4004, #4005, #4010, #4011, #4015, #4017, #4019, #4025, #4028, #4032, #4060).
#3968	feature - Workload: flag entities whose scheduled search was deleted and retire them automatically	Entities of scheduled searches deleted from Splunk stayed red, and a temporary deletion brought them back. The hourly inactive entities tracker now lists each account's saved searches as the tenant owner (through the remote account for a remote deployment). An entity whose definition the metadata tracker once read and whose search is no longer listed is flagged with a red icon in the Overview table, whose tooltip gives the detection time and the retirement date, and is retired (temporary deletion) after `deleted_search_retirement_days` (default 7, 0 flags only) once it has been quiet for a day. A listing that fails or comes back empty never flags nor clears; `_ACCELERATE` searches, apps disabled in Splunk and apps whose WLK monitoring is off are never checked; a flagged search that runs again is never flagged again until it is listed. Both delays are set in the new Manage: entity lifecycle (admins save, every user can view) and in the Entity lifecycle panel of the tenant wizard. Existing tenants get the 7-day default from the first hourly run

Table 9: Enhancements and Changes

Issue Number	Description	Details
#3854	enhancement - Cribl Edge: object monitoring and activity trends	A Fleet-scoped Monitoring tab lists the sources, destinations, routes, pipelines and packs of the Fleet with their health, event and byte sparklines, KPI and chart modals and affected-node drilldowns. Everything is emitted from the published Fleet aggregates: no additional request to the Leader, whatever the number of nodes.
#3844	enhancement - Cribl Stream: connected route, pipeline and destination detail views	A route, pipeline or destination opens with the objects it is connected to, their measured health and activity, and the exact connected set highlighted in Topology.
#4023	enhancement - Cribl Edge: add Edge Nodes to the expected nodes, and an Expected nodes dialog	The Edge Nodes table gains a selection and Add to expected, which adds the host names of the selected nodes to the Node connectivity policy. An Expected nodes dialog manages the list as on Splunk UF Fleet: the expected nodes with their connection, names never seen, patterns, lookup tables and a test of the selection. The Fleets tab also gains an Edge Nodes now panel with the nodes by state (PRs #4024, #4017).
PR #4039	enhancement - Cribl Edge and Splunk UF Fleet: lookups of the Edge Nodes and agents, Open in Search	New lookups (`trackme_cribl_edge_nodes_tenant_<tid>`, `trackme_splunk_uf_agents_tenant_<tid>`) expose the Edge Nodes and forwarding agents a tenant knows, to reuse in any Splunk search, and the Edge Nodes and Agents tables gain Open in Search. They are lookup definitions over the stores the collectors already maintain: nothing more is written and no run time is added.
PR #4016	enhancement - Cribl Stream, Cribl Edge and Splunk HEC: Collection health aligned on Splunk S2S and Splunk UF Fleet	The Collection health tab gains the same views: Status, Trackers, Data & storage, Troubleshoot & help and Custom policies, with the Cribl Leader view on Cribl. A run whose work simply continues on the next run reads Continues next run instead of a failure (PR #4020).
PR #4031	enhancement - Findings tenants: search the Policies catalogue	A search field in the Policies panel of User Activity Monitoring, Cribl Stream, Cribl Edge, Splunk HEC, Splunk S2S and Splunk UF Fleet finds a policy by name, description, category, severity, status or identifier.
PR #4014	enhancement - Splunk HEC and Splunk S2S: Traffic ML models with trends	The models table gains banded trend sparklines and a trend range picker, as on Cribl Stream.
PR #3841	enhancement - Findings: custom policies in actions and automations, UAM Email template, Cribl Edge policy actions	Findings actions and automation rules can name custom policies the same way as built-in policies, on every findings component. User Activity Monitoring custom policies get an Email template, and Cribl Edge policies gain the actions already available on Cribl Stream.
#3880	enhancement - Configure tenant shows only the settings each tenant type uses	The Configure tenant modal is organised in sections modelled on System settings, and only shows the settings read by the tenant's components: a Cribl, UAM or Splunk HEC tenant no longer lists entity-component settings it never uses.
#3906	enhancement - Configuration Guardian: modern alert card with step-by-step remediation on Virtual Tenants	The Guardian notification on the Virtual Tenants page is a card styled like the workspace kit: the severity sets the accent, the remediation is rendered as numbered steps taken from the check's recommended actions, each alert of a group keeps its own remediation, and Dismiss closes the card only once every alert of the group is dismissed.
#3915	enhancement - Keep the last connectivity check of every account type	Remote deployment, email, S3, webhook and AI provider accounts keep the result and time of their last connectivity check and show it in their management pages, as Cribl connections already did.
#3963	enhancement - Tenant Home: the browser tab is named after the tenant	Every Tenant Home tab read `TenantHome`; it now reads `<tenant> – TrackMe Tenant Home`, so several tenants open in separate tabs can be told apart (PR #3965).
#3992	enhancement - Findings tenants: Splunk Web root URL for the links to TrackMe	The links of the notification emails and the `finding.link` token of findings actions were built from the detected search head address, which cannot see a reverse proxy, a load balancer, a

Table 10: Bug Fixes

Issue Number	Description	Details
#4021	bug - Splunk 10.6.2 and later: KV store writes refused for field names with a dot or a leading $	From Splunk 10.6.2, on Splunk Enterprise and Splunk Cloud alike, the KV store refuses a batch holding such a field name and the whole batch is lost. TrackMe no longer writes one: maps keyed by data are stored as pairs and archived maps are converted at restore. Readers accept both shapes, so no migration is needed and each writer rewrites its rows on its next run (PR #4022).
PR #4041	bug - ML Outliers: models without an impact score	Models created before the impact score (2.3.21) have no score: a bulk edit skipped them while reporting success, and they were always scored 36. The bulk edit now adds the score, training adds the tenant default score, and the ML monitor and the UI use the tenant default instead of a fixed 36. A failed training call now skips that model only, writes nothing for it, and lists it in the run's result (PRs #4043, #4046, #4051).
PR #4027	bug - Health tracker: a closed connection stopped the tenant's health cycle	Mostly on search head cluster members, a request sent on a connection splunkd had closed while idle failed, and the remaining tasks of that health cycle did not run. Every splunkd call of the tracker now uses its own connection.
PR #4018	bug - User Activity Monitoring: a brief splunkd failure failed a tracker step	One dropped connection, 503 or slow answer failed the step. Every splunkd call of every UAM tracker now follows one transport policy: time is the only bound, only what is safe to repeat is repeated, and absorbed repeats are counted on the receipt. Resource usage runs on a tenant scoped to some search heads could end incomplete; they now read a search's origin from Splunk's own fields and absorb the seconds a run drifts (PRs #4026, #4029).
PR #4049	bug - License Management: a dialog could be closed or submitted again during a request	The license file, license key, trial, developer mode and reset dialogs now wait for the backend: they show progress, refuse a second click or a close while the request is pending, and unlock on a failure so it can be retried.
PR #3936	bug - VOL: license events that reach the indexers late were lost	On deployments where the license manager's `license_usage.log` reaches the indexers late, the collector's window closed before the events arrived and most of the volume was missing from one moment on. Each run now ends its window at the license writer's own source watermark and re-reads a late overlap before its cursor, raising the committed buckets to the re-read totals without re-emitting a point. A held collection is measured on the entity and is not reported as inactivity.
#3848	bug - VOL: effective licensing and verified closed days	Licensed daily figures use the effective license weight, one Usage writer is established per tenant, and closed days are verified against the raw and effective rollover totals. The view discloses the effective coverage, the provisional days and the source scope of its figures.
#3847	bug - VOL: binary unit preferences and honest license figures	Volumes are labelled in IEC binary units (MiB, GiB) by default, with a system setting and a browser override for Splunk-style labels (MB, GB). Missing data is handled on exact dates, and the headers expose the license-day frame and the collection scope.

continues on next page

Table 10 – continued from previous page

Issue Number	Description	Details
PR #3943	bug - VOL: Daily changes table sorts change by sign on demand and follows the displayed precision	The Change column always ranked by size, mixing increases and decreases; its header now cycles largest change, increases first and decreases first, and the card subtitle names the current order. The sign and the percentage follow what is displayed: a change that reads 0.00 shows `0.00` and `0.0%`, never `-0.00`, and a comparison volume that reads 0.00 gets no ratio instead of a phantom `+2000%`.
PR #3853	bug - Cribl: retained single-member Finding groups did not render	A Finding group retained with a single member is rendered like any other group.
PR #3925	bug - Custom policies: one object named two ways counted as two	A custom policy result naming the same object by its id and by its host name is merged into one observation, and the rows of both names are counted together.
PR #3948	bug - VOL: a healthy entity read "the license source is delivering late", and the index investigation showed its evidence twice	Ordinary forwarding lag across a 5-minute boundary measured a one-bucket hold, and the entity status said the license source was delivering late; the message now starts at the collector's own delay threshold (15 minutes), and every hold is still discounted from inactivity. Investigate an index showed the deployment's license evidence twice (two About this deployment buttons, alerts repeated); it shows it once. The multiple license writers diagnostic now says why collection is held and how to scope the license search to one host, including after a license manager is replaced.
#3838	bug - UAM: audit lines framed by the audit.log layout were read as invalid	On a cluster indexing `_audit` from the audit.log file, every record was read as invalid with an unknown account. Both audit readers accept the framed line.
PR #3944	bug - UAM: searches started from the UI were cut by a time budget	The account deep-dive, Activity, timelines, policy and tracker histories and the stored search text were answered within a 45 to 60 second budget and marked incomplete beyond it. These reads now run as the caller's own Splunk search jobs, polled by the UI, with a generous safety timeout only, and are cancelled when the view closes or its filters change. A read is partial only when a search failed, Splunk reported a cut, or the read's own row bound was reached. API and AI tool callers keep their synchronous contract and now wait for the whole search. A read whose plan changed between two polls starts again by itself, and a refused fallback search no longer fails the whole read (PR #3949).
PR #3958	bug - UAM: search head cluster qualification fixes	Found while qualifying an upgraded UAM tenant on a Splunk Cloud search head cluster; no evidence was lost. Run now waits only for the tracker it started, so a Data usage catch-up no longer keeps the whole workspace disabled. A tracker run on another member records a lease on its lock and reads as running within it, so a live run is no longer reported as an unverifiable lock with a Recover that fails. A span read during a catch-up is due its recount from when it was read, not from when it ended, so recounts are no longer reported overdue by hundreds of hours. With a backfill, a new Data usage history starts at its depth but never before the oldest audit event the deployment holds.

continues on next page

Table 10 – continued from previous page

Issue Number	Description	Details
PR #3960	bug - UAM Data usage: indexes weighed 0 until the VOL tenant had a complete day	Data usage weighs each index by the linked VOL tenant's 30-day average, which needs a complete day. On a new VOL tenant without a backfill, or with an `_internal` retention under a day, every index weighed 0 and the flow, treemap and value views were empty. Until the average exists, the index's last 24 hours are used and the views say so; the 30-day average takes over as soon as VOL has a complete day.
#3964	bug - Workload: a long saved search name overlapped the entity header	Saved search names such as data model acceleration searches wrapped over several lines, overlapped the neighbouring values and made the header card scroll. The name now holds on one line, with an icon that opens the full name when it is cut, and Application and Owner move to the card's last line (PR #3967).
PR #3971	bug - Final qualification fixes: Cribl restore state, UAM Data usage wording and units, Guardian alerts on new tenants, native ML log volume, Cribl route drop label	Found during the final 2.4.22 qualification. Cribl Stream, Cribl Edge and Splunk HEC: a tenant restored from a backup and waiting for Repair workspace showed an error that a reload could not fix, the last loaded findings and summary stayed on screen, and the health tracker logged a warning every cycle; the Findings tab now says the collection is paused until an administrator runs Repair workspace, and the health tracker logs it at INFO. The Routes table showed a route's pipeline drop rate as the route's own (a route on a dropping pipeline read `Dropped 100%`); it now reads Pipeline dropped. UAM Data usage: a missing volume names its actual cause (several VOL tenants matching a search head tier, too many DSM matches, an incomplete match, a linked tenant that cannot be read) instead of always saying the volume could not be read, and volumes follow the VOL unit preference (GiB/day by default). Configuration Guardian: `health_tracker_not_executing` no longer raises a critical alert for a tenant created less than 30 minutes ago, and `backup_run_incomplete` no longer reports a tenant created after the latest run started. Native ML fit: one summary line per fit instead of a warning for every group still building its history.
PR #3983	bug - User Activity Monitoring on large search head clusters	Found on a 14-member search head cluster. User Activity Monitoring rewrote unchanged permissions on its collections and lookups every few minutes, and each write was replicated to every member; it now writes configuration only when it changes. Its locks expire, empty policy windows are skipped, and the Resource usage search aggregates before it correlates and reads indexed fields once they are proven, so it no longer runs as long as its own schedule (PRs #3984, #3994).
PR #3993	bug - Findings tenants: Splunk's job state is the verdict of every search	User Activity Monitoring, Cribl Stream, Cribl Edge, Splunk HEC, Splunk S2S and Splunk UF Fleet judged how a search was served by the search peers, which led to re-reads, recounts and runs refused while an indexer was busy or restarting. Every search now follows one rule: a finished job's results are the answer, a failed job is a failure, and Splunk's messages are logged without deciding anything. VOL keeps its exact license accounting (PR #3988, issue #3986).

continues on next page

Table 10 – continued from previous page

Issue Number	Description	Details
#3974	bug - Virtual Tenants: a tenant being created could be reported faulty	A Virtual Tenants refresh during a tenant creation could hide the new tenant with a configuration issue warning, and in rare cases recover its account with default settings. A tenant created less than 10 minutes ago is never reported faulty nor repaired, and the creation now reports an account it could not write (PR #3980).
PR #3990	bug - Splunk HEC: a setting saved in Settings waited for the hourly rebuild	A pool or endpoint setting saved in the tenant's Settings now applies at the next inventory run.
PR #3999	bug - User Activity Monitoring: a deleted tenant could not be restored nor its id reused	Deleting a User Activity Monitoring tenant left a marker that blocked every later tenant of the same id: restoring the tenant from its backup archive left it unable to collect, and creating a new tenant with that id failed. The marker now applies only to the deleted tenant, a restore (single archive or full run) or a new creation retires it once the deletion has completed, and a deletion still running is waited for, never bypassed. Ids blocked by earlier versions can be reused.
PR #4000	bug - REST API: WLK tenant without outliers settings, FQM collect jobs and their schedule	A Workload tenant created through the REST API without outliers settings could not get its main tracker; it now gets outliers disabled, as in the wizard. The FQM collect job endpoints accept any valid schedule, keep the default of a daily run, and remove the reports and dictionary a failed creation made instead of leaving them behind after the tenant is deleted.
#3831, #3832	bug - Splunk HEC: Configure tenant hid the shared settings, Overview had two time ranges	Configure tenant shows the shared tenant settings again on a Splunk HEC tenant, and the Overview owns a single time range for all of its panels.
#3870	bug - Splunk HEC: Guardian did not check findings-action deliveries, Cribl-only commands accepted other products	The `findings_action_delivery_failing` Guardian check covers Splunk HEC tenants, and the Cribl-only search commands refuse other products by name instead of failing later.
#3843	bug - Virtual Tenants: tenant cards flickered during a summary refresh	When a collection published a new summary snapshot while the page was reading it, the tenant badges flickered; the race is recovered without a visible change.
PR #3941	bug - The trackme_virtual_tenants lookup did not list every tenant record field	A `| inputlookup` / `| outputlookup` round trip on the tenants lookup, a common way to force a migration in a lab, erased the seven fields the lookup did not list: the Cribl Edge and Splunk HEC product flags (an empty Tenant Home and backend gates no longer treating the tenant as a dedicated one), the ML outlier filters, the automatic labels rules and the licence enforcement marker. The lookup now lists every field and a guard keeps it so.
#3923	bug - Webhook account save failed with HTTP 500 on Splunk 9.x (Python 3.9 / OpenSSL 1.0.2)	The webhook and custom-CA TLS contexts are built with the attributes available on Python 3.9 and OpenSSL 1.0.2, and a Python 3.9 stdlib guard refuses attributes that exist only on newer versions.
#3927	bug - AI: the provider's own error text was lost with the openai SDK 3.x	The openai SDK 3.x no longer puts the provider's message in the exception text; the AI Advisors and the AI Assistant read it from the response body so the provider's own words reach the operator.
#3928	bug - AI agents: HTTP objects handed to the openai SDK under openai 3.x	Every HTTP client, timeout, limit or transport handed to the openai SDK (Azure OpenAI, Splunk-hosted) is built from the SDK's own HTTP library, which changed in openai 3.x.

continues on next page

Table 10 – continued from previous page

Issue Number	Description	Details
#3932	bug - Editing a remote deployment or S3 account without re-entering its secret failed the save	The save-time connectivity check used the empty secret of the form; the stored secret is kept and used when the account is edited without it.
#3850	bug - FIPS: the expected TLS group fallback flooded splunkd.log	On FIPS-enabled deployments the fallback of the splunk-sdk TLS group guard is expected and is no longer logged on every connection.
PR #3945	bug - Backup & Restore: a deleted Cribl Stream, Cribl Edge or Splunk HEC tenant could not be restored, orphaned KV tables and leaked UAM locks	The deletion marker of a Cribl Stream, Cribl Edge or Splunk HEC tenant was permanent, so every restore of that tenant was refused; the restore now retires it, and a run of the deleted tenant still in flight stops instead of writing into the restored one. On a Splunk 10.6 search head cluster, a member could recreate the table of a deleted collection and every later creation of that name failed (HTTP 500); collection creation now re-attaches such a table. A UAM lock could leak and block later restores and the tenant's findings runs; it is now released by its token and recovered once its process is gone. A restore also waits for a busy UAM tenant, and a tenant record that lost its product flag is still cleaned up at deletion and written back correctly at restore.

11.1.5 Version 2.4.21 - build 1792629142 (22/10/2026)

Hint

TrackMe 2.4.21 — Splunk HEC monitoring, User Activity Monitoring rebuilt for scale and transparency, Cribl Stream and Cribl Edge improvements, license usage investigation for VOL, and disaster recovery for Backup & Restore.

- **Splunk HEC: a new dedicated tenant type.** Monitor the HTTP Event Collector of a Splunk deployment, local or through a remote account, from the telemetry Splunk already logs. Endpoints are discovered automatically and grouped in pools, tokens are followed by name (never by value), and 18 built-in policies watch refused requests, back pressure, ACK pressure, queues, output and traffic, with traffic models per pool and token. It is built on the same model as User Activity Monitoring, Cribl Stream and Cribl Edge: custom policies, Finding groups, notifications, assignment, findings actions, a Splunk HEC AI Agent, generated topologies and Topology Studio, and bank holidays. On Splunk Cloud, the wizard recognises the stack's indexers and creates a replaceable Splunk Cloud pool for them. See *Splunk HEC*.
- **User Activity Monitoring: rebuilt for high scale, clarity and transparency.**
  - Collection never drops data: time is the only bound of a run, what a run cannot finish carries over to the next, and the row limits are retired.
  - Three indexing-layer trackers, Audit activity, Resource usage and Account signals, each with its own schedule, budget, lock and health.
  - Collection health that answers: Status, Trackers, Data & storage and Troubleshoot & help, a glossary of every diagnostic code with its meaning and what to do, the exact searches each run sent, and a Flow view of the tenant.
  - Privileged users: a tab showing which accounts hold high privileges and why, with role inheritance resolved, and four privileged-user policies, all off by default.
  - Users and roles are read with the system token on the local tier, so a least-privilege tenant owner no longer limits the catalogs to its own account, and a limited listing is flagged.
  - Smaller evidence, Settings that stay responsive with thousands of accounts, Splunk 9.x and SSO logins recognised, no scheduler skew on Splunk Cloud.

See *User Activity Monitoring*.

- **Cribl Stream and Cribl Edge: stronger operations.** A How it is evaluated tab on every built-in policy, Finding groups listed most recent first, a source that never qualified is retried automatically, a tenant deletion is never refused, run budgets aligned with UAM, longer-lived UI searches, and creation wizards that open with a How it works diagram and close with review cards. See *Cribl Stream* and *Cribl Edge*.
- **VOL Global license usage: investigate.** Daily changes rank the increases and decreases across every index, Investigate an index gathers everything about one index, and an Analysis period selector covers rolling, calendar and year-to-date periods. A scope selector that stays readable with many scopes, an All scopes overview, CIM datamodel macros read locally or from a remote account, correct pool quotas on multi-pool deployments. See *Volume Outliers*.
- **Backup & Restore: disaster recovery.** Import backup archives from S3, restore a whole backup run onto a rebuilt deployment, Cribl and UAM tenants included, and a Cribl, Cribl Edge or Splunk HEC tenant restored in place resumes its collection by itself. See *Backup & Restore*.
- **Others.** A tenant-level default index destination for FQM collect jobs, per-state counts on the entity table group headers, Update alias from the entity actions menu, Topology Studio labels entity nodes by their alias, findings tenants in the Saved View and Virtual Group wizards, a Python 3.9 import fix, and a dependency hygiene pass.

SHA-256: aaeed6effc3673e031d9526851c65e6e2aed1242781dc2c09a204ef2bd7e8a5b

Table 11: New Features

Issue Number	Description	Details
PR #3791	feature - Splunk HEC: a dedicated tenant type for the HTTP Event Collector	A Splunk HEC tenant monitors the HTTP Event Collector endpoints of a Splunk deployment, locally or through a TrackMe remote account, read only, from the `_introspection` HEC metrics and the `_internal` failures, queues and output. It covers: • endpoints discovered automatically and grouped in pools (patterns, hand assignments, replaceable members), an endpoint directory and token names, never token values; • live collection every 5 minutes with catch-up windows, and history, trends and Open in Search; • 18 built-in policies on the shared findings runtime (Finding groups, notifications, assignment, findings actions) with a How it is evaluated tab, plus custom policies; • traffic models of the incoming traffic per pool and token, with an expected range, tuning, relearn and a learning history; • evidence screens behind the refused and back pressure figures, ACK pressure on Splunk's own signals, token silence from Splunk lookup tables; • an optional splunkd API enrichment, a Splunk HEC AI Agent and the AI Assistant context; • Topology Studio: pools, endpoints and tokens as nodes with live status, dynamic groups, templates and Build with AI; • bank holidays, backup and restore; • generated topologies from the Overview, Endpoints and Tokens tabs, host CPU and memory in the endpoint table, and a Splunk Cloud pool with replaceable members created by the wizard when it finds Splunk Cloud indexers (PR #3820, PR #3822, PR #3823, PR #3826); • the release QA fixes on a Splunk Cloud stack: a live-run failure on the pool version histogram (PR #3827), readable times in the AI Agent evidence (PR #3816, PR #3825), the Beta badge removed from the AI Assistant context (PR #3814), HEC wording in the aggregate panel and the AI Agent act card (PR #3829). Available on the Enterprise and Unlimited editions (trial and Developer licenses included). See *Splunk HEC* and *Create a Splunk HEC tenant*.
#3763	feature - User Activity Monitoring 2.4.21: collection that never drops data, Collection health that answers, three indexing-layer trackers, Privileged users	The 2.4.21 UAM milestone, built from production feedback on a high-scale deployment: • **No-drop collection.** The seven row limits are retired. A run is bounded by its time budget only: paged audit reads proved whole, a range that keeps failing is held and halved rather than skipped, a window read while search peers did not answer is read again, closed ranges are recounted, listings out of time continue in a staging generation, and the policies evaluate every window up to the collected evidence. • **Three indexing-layer trackers.** Audit activity, Resource usage and Account signals run as separate scheduled reports with their own budget, lock, receipt and health, so a heavy resource read never holds back the audit trail. Existing tenants get the two new reports automatically. • **Collection health that answers.** Four views: Status (one row per tracker that needs attention, with every reason, its evidence and its action), Trackers (live schedule, last run, diagnostics, Run now / Recover), Data & storage (every event, collection, log and report UAM writes, with an in-

Table 12: Enhancements and Changes

Issue Number	Description	Details
#3759	enhancement - Cribl Stream and Cribl Edge: How it is evaluated tab for every built-in policy	Every built-in Cribl Stream (16) and Cribl Edge (10) policy, plus Leader availability, has a How it is evaluated tab: the rule and how it decides with the current settings, the data it judges with ready-to-run searches built from the tenant's own indexes, Investigate, Causes and resolutions, Related policies and Good to know.
PR #3758	enhancement - Cribl Stream and Cribl Edge: Finding groups listed most recent first	The Findings tab lists Finding groups most recent first across every page, as the UAM Findings tab does: groups still failing first, newest opened first, then the groups that ended, most recently ended first.
PR #3799	enhancement - Cribl Stream, Cribl Edge and Splunk HEC: run budgets and scheduler skew aligned with UAM	The collection reports of the three products run under a budget of their interval less 30 seconds, end before their next slot, and no longer skew on Splunk Cloud, as the UAM trackers do.
#3705	enhancement - VOL Global license usage: read a scope's CIM datamodel macros locally or from a remote account	A scope that uses the CIM datamodel macros can read them where Enterprise Security configures them: this search head (the default) or a TrackMe remote account, per scope, as the caller and with the account's role restrictions enforced.
#3706	enhancement - VOL Global license usage: scalable scope selector and an All scopes overview	The scope row stays readable from 1 to 50 scopes (tabs while they fit, a filterable selector beyond), and a new All scopes view compares the scopes side by side: source, indexes, licensed volume, share or cost, last 7 days against the prior 7 days, month projection.
#3710	enhancement - VOL Global license usage: no auto-refresh on the license view	Tenant Home's auto-refresh is paused while the Global license usage view is open, so an analysis or a configuration draft is never reloaded under the user. The Refresh buttons still reload, and the entity tab keeps its auto-refresh.
#3798	enhancement - Entity tables: per-state entity counts on group headers	Every collapsible group header of the entity tables shows one coloured dot and count per non-empty state, problems first, for every component and every Group by option, counted on the group's full membership.
#3803	enhancement - Entity components: update the entity alias from the entity actions menu	Update alias is the first item of the Modify section of the entity menu (table row and entity view) for DSM, DHM, MHM, FLX, FQM and WLK, with an optional audit comment, replacing the legacy inline cell edit.
#3809	enhancement - Topology Studio: label entity nodes by their alias	An entity node without a custom label is labelled by the entity's alias, as the members of an expanded aggregate are; a per-view setting keeps the entity name instead.
#3731	enhancement - UAM: use the account name as email when Splunk exposes none	When Splunk exposes no email for an account, notifications use the account name as the local part, so a deployment whose mail relay resolves names still reaches the account.
#3735	change - UAM backfill: 5-minute buckets and no backfill preset in the tenant creation wizard	The backfill reads 5-minute buckets, as the live collection does, and the wizard no longer proposes a backfill period: the backfill is started from the workspace once the tenant collects.
#3821	enhancement - FQM: tenant-level default index destination for new collect jobs	The FQM tenant creation wizard and the Configure tenant modal set a default index destination for the tenant's collect jobs (`summary` by default). The collect-job wizard presets it for the CIM and raw tracker types, and a monitor-only tracker reads where the tenant's collect jobs write.
#3795	enhancement - Dependencies for version 2.4.21	The two follow-redirects advisories were resolved by removing dead lock resolutions: the package was never installed. webpack 5.112.0 and baseline-browser-mapping 2.11.30 (PR #3819), with the Mako template minification guard verified.

Table 13: Bug Fixes

Issue Number	Description	Details
PR #3828	bug - UAM: users and roles were listed as the tenant owner could see them	Splunk lists every user and role only to an account holding the list-all capabilities; with the recommended least-privilege tenant owner, the local users catalog held one account and Privileged users read 1 of 1. The local tier's users and roles catalog pages are now read with the system token through a tenant-admin operation, never falling back to the owner's own view, and a limited listing (a remote tier whose account lacks the capability) is reported on the receipt, in Collection health and on the Privileged users tab.
PR #3815	bug - Backup & Restore: a Cribl Stream, Cribl Edge or Splunk HEC tenant restored in place stayed paused	A tenant archive restored into its live tenant reported OK but left the tenant's collectors unscheduled until a Repair, and nothing said so. An archive restored into its existing tenant without error now runs the tenant's Repair itself; when a Repair is still required, Collection health says so. A refused archive stops instead of being reported as restored.
PR #3829	bug - Creation wizards: the How it works lightbox was clipped by the dialog	The lightbox of the HEC, UAM, Cribl Stream, Cribl Edge and Splunk Volume wizards is rendered over the whole window, and Escape closes the lightbox only, never the wizard behind it.
#3807	bug - Backup & Restore: full-run restore onto a rebuilt deployment	A whole backup run now restores onto a rebuilt search head or cluster: the global archive runs first and its tenant registry is fenced row by row, an existing collection is kept and cleared rather than dropped and recreated (Splunk 10 could refuse the recreate), restored trackers are scheduled only once the tenant's records are back, Cribl tenants are recreated with their archived identity, UAM tenants are recreated as UAM, the archive registry is never restored, Topology Alert searches are regenerated, and the dry run carries a per-archive preflight. A pending cancel is reported.
#3766	bug - Backup & Restore: S3 upload of UI-started backups failed for least-privilege TrackMe admins	A backup started from the page runs with the user's own session; the S3 secret is now read as the caller with the least-privilege endpoint, so an admin without the storage passwords capability uploads to S3.
#3800	bug - Cribl Stream, Cribl Edge and Splunk HEC: a tenant deletion is never refused by the runtime cleanup	A deletion was refused whenever the cleanup raised, with no way out. The cleanup is now best effort: the tenant record is always deleted, leftovers are reported and removed by a daily orphan purge.
#3808	bug - UAM and Cribl: the orphan purge could delete the resources of a tenant being created	Tenant creation writes a creation fence before provisioning, and the daily purge treats a fenced tenant as owned and re-reads the registry before removing anything.
#3783	bug - VOL: an index reporting under several pools got another pool's size	Each index's pool is paired with that pool's own size, with hysteresis so a shared index does not flap between pools, and the license view reads each pool's quota from its own rows. A pool the license manager has not reported for 4 days is forgotten (#3788).
#3774	bug - VOL: the hourly profile attributed each hour's last 5-minute bucket to the next hour	The hourly profile bins on the bucket start, not the point time.
#3771	bug - VOL Global license usage: sub-tab outline clipped and Scope caption too close to the first scope	Layout fix.
#3769	bug - Policies screens: the regex pattern editor did not fill the Pattern / Lookup column	The inline editor of a regex Tags, Priority or SLA policy now fills the whole column.
#3743	bug - Saved Views and Virtual Groups wizards: findings tenants silently did nothing when added	A Cribl Stream, Cribl Edge or UAM tenant is added as a whole card in a Saved View, and the Virtual Group wizard explains that findings tenants have no entity component to aggregate.
#3736	bug - UAM backfill: a deferred bucket ended the hourly	A deferred bucket no longer ends the run; the next bucket is read

11.1.6 Version 2.4.20 - build 1792051492 (15/10/2026)

Hint

TrackMe 2.4.20 — a maintenance release for User Activity Monitoring, Cribl Stream and Cribl Edge.

- **User Activity Monitoring: collection no longer stops on same-name knowledge objects.** Splunk can keep a private knowledge object next to a shared one with the same owner, app and name on the same search head tier. The catalog collection refused that pair on every run, so the affected dataset never published again. Both objects are now collected, each with its own identity, and a duplicate never stops a run: it is reported in Collection health instead. See *User Activity Monitoring*.
- **User Activity Monitoring: operations hardening.** Findings-action webhook rules on resolved, dismissed and assigned now fire on operator decisions. A UAM tenant deletion is never refused by its cleanup. A tenant RBAC update reports an incomplete UAM reconciliation instead of a silent success. Clearer wording when reopening a finding of a dismissed Finding group.
- **Cribl Stream and Cribl Edge: no more spurious failures when writers overlap.** Report reconciliation and live-run publication wait briefly for the tenant lifecycle guard instead of failing at once, so reports stay reconciled and the evidence no longer lags one interval behind. See *Cribl Stream* and *Cribl Edge*.
- **Cribl Stream and Cribl Edge: custom policies.** Write your own SPL policy on the tenant's Splunk data. Each result row names a Cribl object and opens a finding that behaves like a built-in one: Finding groups, triage, notifications, findings actions and the AI Agent. See *Cribl Stream custom policies* and *Cribl Edge custom policies*.
- **Findings: bulk actions are fast again.** One request and one transaction per bulk edit, batched Assign to… and Apply rules, cheaper Cribl writes, and webhook deliveries queued in one batch. Automatic webhooks are delivered when the tenant owner is a least-privilege account, and a delivery that keeps failing records why before it is dropped.
- **Others.** Bank Holidays hardening: the legacy Maintenance KDB copies purge works from a backed-up register, stale scope rows are removed, a failed scope sync is recorded as such, and VOL keeps its legacy holidays behaviour (see *Bank Holidays*). Tenant Home score action and lag policy modals read the REST response. A dependency refresh with security fixes.

SHA-256: 75bd18fef848d854c5abf4c90471b505d50ae4acbeb21aa25cba88e30282bee3

Table 14: New Features

Issue Number	Description	Details
#3652	feature - Cribl Stream and Cribl Edge: custom policies	An administrator of a Cribl Stream or Cribl Edge tenant authors an SPL search on the tenant's own Splunk data, for example `mstats` over the Cribl metrics or the Edge System Metrics index. Each result row names a Cribl object (Source, Destination, Pipeline, Route, Worker Group, Worker machine or process on Stream; Edge Node on Edge) and opens a finding of the policy. The finding joins its object's Finding group and gets triage, notes, notifications, findings actions, finding events and the AI Agent. Policies are drafted, previewed, published as revisions and enabled; one scheduled report per enabled policy runs as the tenant owner. Collection health follows each policy's schedule and runs. See *Custom policies* and *Custom policies*.

Table 15: Enhancements and Changes

Issue Number	Description	Details
#3665	enhancement - Findings: one request and one transaction per bulk edit (Cribl Stream, Cribl Edge, UAM)	Bulk actions on findings (assign, status, snooze, severity, dismiss) sent one request per selected row and paid the full write cost each time, so a bulk edit of a few rows took minutes. A bulk edit is now one request and one transaction, and every Cribl write no longer re-reads the license status and the tenant record at each fence.
#3681	enhancement - Findings: batched Assign to… and Apply rules, cheaper Cribl write fence, one look for the assignment dialogs	Assign to… and Apply rules are one request for the whole selection. Each Cribl write fence makes about 4 live reads instead of about 13, with the same checks and error codes. The Assign to… dialog reads like the bulk edit, and the bulk edit's Assign offers an assignment group as well as a user.
#3690	enhancement - Findings actions: webhook deliveries are queued in one batch	When a webhook rule matched, each finding transition of a bulk action was queued on its own, with about 11 KV requests each. Deliveries are now queued in one batch for the whole action, on UAM, Cribl Stream and Cribl Edge.
#3697	enhancement - Findings actions: a webhook delivery that keeps failing records why before it is dropped	A failed automatic webhook is retried for up to 6 attempts or 24 hours. Each failed attempt now records its reason and HTTP status on the outbox row, and a refused delivery grant keeps its real code instead of a generic failure, so a delivery problem is visible at once rather than a day later.
#3666	enhancement - Dependencies refresh for version 2.4.20	Splunk UI toolkit 5.17.0 (react-ui), 5.16.0 (react-icons), 1.13.0 (themes) and 4.1.0 (splunk-utils) across every package, and the security advisories with a fix applied (brace-expansion, source-map-js, serialize-javascript, postcss-selector-parser).

Table 16: Bug Fixes

Issue Number	Description	Details
#3683	bug - UAM: a private and a shared object with the same owner, app and name blocked the whole catalog collection	Splunk keeps a private knowledge object (`sharing=user`) next to an app or global shared one with the same owner, app and name: two objects, one identity for the UAM catalogs. The traversal refused the whole generation on every run (`duplicate_identity_during_scan`), so Full listing or a knowledge-object dataset never published again. The shared object keeps its identity; the private copy gets its own, scoped identity. A duplicate never stops a collection anymore: the run publishes and reports the pairs in its diagnostics and in Collection health. Fetch definition reads the copy matching the record's sharing level. Objects without a twin are unchanged, so there is no finding or group churn at upgrade.
#3693	bug - UAM: operator decisions never queued findings-action webhooks	On a UAM tenant, only the scheduled policies run queued webhooks, so a rule on resolved, dismissed or assigned never fired. Operator decisions (UI, bulk edits, assignment, automatic rules, the AI Agent) now queue their webhooks, as on Cribl Stream and Cribl Edge.
#3686	bug - UAM: a tenant deletion is never refused by its cleanup	A busy lifecycle lock, a running job or a report that could not be unscheduled refused the deletion of a UAM tenant. The cleanup is now best effort: every report, lookup definition and collection is deleted on its own, leftovers are reported and removed by a daily task, and the deletion always completes.
#3455	bug - UAM: a tenant RBAC update reported success when the UAM reconciliation failed	When the UAM report reconciliation raised after the tenant record was saved, the endpoint still answered success. It now answers HTTP 409 with an explicit retry instruction, as for Cribl tenants, and the UI shows the error.
#3678	bug - UAM: "Reopen this finding" wording on a dismissed Finding group	Reopening a finding that was set aside from a dismissed group opens the group for that finding and keeps the dismissal on the others, as designed. The menu description and the toasts said the finding takes the status of its group again; they now describe what happens. A regression test covers the sequence.
#3688	bug - Cribl Stream and Cribl Edge: report reconciliation failed when another writer held the tenant lifecycle guard	The report reconciliation, run every ten minutes by the health tracker, took the guard without waiting and failed with `tenant_lifecycle_busy` whenever a collection, a triage or a custom policy run held it. It now waits up to 12 seconds, like the other writers of the tenant.
#3696	bug - Cribl Stream and Cribl Edge: live-run publication failed when another writer held the tenant lifecycle guard	A live run completed, but its publication failed at once when an operator decision, a custom policy run or a reconciliation held the guard, and the evidence stayed one interval behind. Publication now waits a bounded time, as report reconciliation does.
#3691	bug - Findings actions: automatic webhooks were refused when the tenant owner is a least-privilege account	The delivery grant asked for a tenant role before checking the admin capability, so a tenant owned by an account holding only the TrackMe capabilities never delivered its automatic webhooks. Automatic grants now need the admin capability, not a tenant role.
#3417	bug - Tenant Home: several modals read a payload envelope the REST response never has	The false positive, score influence and DSM / DHM lag policy modals always fell back to their defaults. They now read the REST response: the false positive modals say when there is nothing to offset, the score influence toast shows the applied score, and the lag modals show the tenant's configured impact scores.
#3473	bug - Bank Holidays: the legacy Maintenance KDB copies purge works from a backed-up register	The opt-in purge told a legacy holiday copy from an administrator's record by its creation time against an install marker kept in a cache that backups exclude and that can be reset. The purge now deletes only the records of a copies register, taken once and kept in a backed-up collection; a record created later or restored from another deployment is never deleted.
#3472	bug - Bank Holidays: stale scope rows after a concurrent settings change	A delayed scope refresh could keep obsolete suppression rows. It now keeps exactly the rows valid in the state it reads again.

11.1.7 Version 2.4.19 - build 1791915833 (13/10/2026)

Hint

TrackMe 2.4.19 — Cribl Stream and Cribl Edge monitoring, User Activity Monitoring generally available, one findings experience across them, license usage management and chargeback for VOL, and Bank Holidays rebuilt on calendars.

- **Cribl Stream: monitor your Cribl estate end to end.** A dedicated tenant follows one Cribl Stream deployment, on Cribl.Cloud or self-managed: the Leader, Worker Groups, Workers, Sources, Destinations, Pipelines and Routes. It flags what is down, drifting, silent, backpressured or behaving unusually, with machine learning traffic models that learn what normal looks like. TrackMe uses the Cribl metrics already in Splunk and the Leader API in read-only mode. Nothing is written to Cribl, and every TrackMe request to a Leader is paced by one shared budget. A Cribl Stream AI Agent can review and act on the findings. See *Cribl Stream*.
- **Cribl Edge: every node of every Fleet, at any fleet size.** A dedicated tenant follows the Fleets of a Cribl Edge Leader: connectivity, version and configuration drift, coverage, the health of Sources and Destinations, and node disk, CPU and memory. TrackMe reads the node list in bulk, so a fleet of any size is covered without per-node API calls. See *Cribl Edge*.
- **User Activity Monitoring is generally available.** Know who is doing what on your Splunk platform: accounts, their searches, their cost and their knowledge objects, across up to ten search head tiers. New in this release:
  - custom policies that you write, preview, version and publish;
  - Finding groups, so all the findings about an account and its searches are handled as one;
  - a full knowledge-object inventory read from the Splunk REST API;
  - oversized CSV lookups and owner quotas;
  - direct notifications to the account owners, and reviewed user communications;
  - exact run time and CPU accounting, including on Splunk Cloud.
- **One findings experience across UAM, Cribl Stream and Cribl Edge.**
  - Policies raise findings, related findings form Finding groups, and every finding has a severity on the TrackMe scale and a triage state.
  - Assignment groups route findings automatically to the right team.
  - Findings actions run your own drilldown searches, links and webhooks, by hand or automatically, through new system-wide Webhook accounts.
  - The findings AI Agents of UAM, Cribl Stream and Cribl Edge investigate and act on a schedule or on demand, following the custom instructions your administrators write for them.
- **VOL: license usage management and chargeback.** The Global license usage view becomes a complete management space:
  - named scopes, opt-in costs per index and scope, and working-day averages based on your holiday calendars;
  - a daily quota per pool, with an Over quota view that explains what made the day;
  - chargeback by cost centre, with monthly statements, budgets and effective-dated prices;
  - an AI Assistant that answers questions about the view and can change its configuration for you;
  - history backfill of up to 400 days.

  See *Volume Outliers*.
- **Bank Holidays rebuilt on calendars.** Import any of 207 countries and their states and regions, or define your own company closures. Holidays are computed from rules, so floating holidays always land on the

right day. Each tenant chooses its calendars and what holidays do: hold alerts, exclude from SLA, exclude from ML training, or treat the day as outside monitoring windows. On UAM, Cribl Stream and Cribl Edge, each policy chooses whether a holiday holds its new findings. Existing holiday periods are migrated automatically. See *Bank Holidays*.

- **Others.** Deep links to every Tenant Home tab and a personal table height. Regex tags and SLA policies can match any entity field. Claude Haiku 5.5 is the default for new Anthropic providers. Targeted fixes, and a dependency refresh with a clean security audit.

SHA-256: 23d702656dc82acc5e5a569352e94a29ff3d7944140f2761075b080f2184361e

Table 17: New Features

Issue Number	Description	Details
PR #3532	feature - Cribl Stream: a dedicated tenant type for Cribl Stream deployments	A Cribl Stream tenant follows the Worker Groups of one Cribl Stream deployment through a Cribl connection, either a Cribl.Cloud workspace or a self-managed Leader. It monitors: • the Leader (API availability, CPU, memory and filesystem); • Worker Groups and Workers (availability, version drift against the running Leader, configuration drift, machine CPU, memory and disk); • worker processes; • Sources and Destinations (reported health, traffic, backpressure); • Pipelines and Routes. Traffic and health measurements come from the Cribl internal metrics already in a Splunk metric index, read in place. The configuration comes from the Leader REST API, read-only, under a shared request budget. Sixteen built-in policies raise findings, which are grouped into Finding groups, triaged, assigned and notified. The tenant has its own workspace, wizard and Topology Studio integration. Available on the Enterprise and Unlimited editions (trial and Developer licenses included). See *Cribl Stream*.
PR #3631	feature - Cribl Edge: a dedicated tenant type for Cribl Edge Fleets	A Cribl Edge tenant follows the Fleets of one Cribl Edge Leader, on Cribl.Cloud or self-managed. It monitors: • Fleet connectivity, version drift, configuration drift, coverage and node identity; • the health and traffic of the Sources and Destinations reported by the nodes; • Edge Nodes: connection and disk, plus CPU and memory when a System Metrics index is set. The node directory is read in bulk, with no API call per node and no limit on the number of nodes. Findings, Finding groups, notifications, assignment, findings actions and Topology Studio work as for Cribl Stream. Edge uses no machine learning: per-node activity is too sparse for traffic models. See *Cribl Edge*.
PR #3536	feature - Cribl Stream AI Agent	An agentic operator for the Cribl Stream findings. It reviews a finding or a Finding group from evidence TrackMe checks (when the object was first seen, whether it is disabled in Cribl, the routes that reference it, its traffic and model) and writes an AI Agent note. In act mode it applies only what the tenant allows: investigating, a time-bounded dismissal, Not an anomaly. It runs on demand from a finding (inspect, then a confirmed act), or as a scheduled triage that is off by default.
PR #3538	feature - Cribl Stream: silent objects stay known and monitored	Sources, Destinations, Pipelines and Routes that stop sending metrics are no longer forgotten: they keep their name and their findings and stay monitored. If a large part of a Worker Group goes silent at the same time, that is reported as a single collection problem rather than a flood of findings.
PR #3546	feature - Cribl Stream: objects removed from Cribl are closed and purged	When an object is removed from the Cribl configuration, its findings close and its records are purged quickly. Administrators can also forget an object explicitly.
PR #3539	feature - Cribl Stream: Expected traffic automatic inclusion	Resources with steady traffic are treated as expected without being selected one by one, so a traffic stop on them raises a finding automatically. Manual selections and exclusions still apply.
PR #3620	feature - Cribl Stream: Leader resource and filesystem monitoring	The Leader's CPU, memory and filesystems are monitored with their own policies. Leader API availability is monitored independently of the collection (PR #3615).
#3552	feature - User Activity Monitoring is generally available	UAM is out of Beta. Highlights of this release: • twelve built-in policies, plus custom policies; • Finding groups; • an API-only knowledge-object inventory; saved searches

Table 18: Enhancements & Changes

Issue Number	Description	Details
#3423	enhancement - VOL: opt-in costs per index and scope	Set an annual license cost or a cost per GB, in your currency, to see costs next to or instead of the percentages, per index and per scope. Costs are allocated at the contract rate or as a share of the bill, which suits Splunk Cloud. The view shows no cost until a price is set.
#3426	enhancement - VOL: working-day averages and the Working days & holidays tab	Averages can be computed on working days only, excluding weekends and the holidays of your Bank Holidays calendars: none, the tenant's own calendars, or a selection (#3447). A bank holidays tile shows the holidays in the period (#3453).
#3436	enhancement - VOL: AI Assistant on the Global license usage view	The AI Assistant knows the sub-tab you are looking at and its figures. With your consent it can change the configuration through the Concierge, for example adding or changing scopes and cost centres.
#3459	enhancement - VOL: custom backfill depth up to 400 days	The tenant wizard offers a Custom… backfill depth, up to 400 days, so chargeback and trends can cover the last thirteen months from day one, within the retention of your internal index.
#3588	enhancement - VOL: outliers chart history from 24 hours to 90 days	Each outliers model chart has a history selector from 24 hours to 90 days.
#3453	change - The Maintenance menu becomes Maintenance & holidays	Bank Holidays is reached through Maintenance & holidays, then Manage Bank Holidays.
PR #3647	enhancement - Cribl Stream: bank holidays for traffic models and Traffic outliers findings	A Cribl Stream tenant uses the bank holiday calendars like every tenant, with two effects on by default: its traffic ML models do not learn the hours of its holidays, and while a holiday is active the Traffic outliers policy opens no new finding (since PR #3648, each policy chooses whether it holds, Traffic outliers by default). See *Bank Holidays*.
PR #3648	enhancement - Bank holidays chosen per policy on UAM, Cribl Stream and Cribl Edge	Each policy of a findings tenant chooses whether a bank holiday of the tenant's calendars holds its new findings: Hold new findings or Keep opening findings, in the policy editor. The defaults suit each policy: the configuration reviews of UAM, Traffic outliers on Cribl Stream, Fleet connectivity and Fleet coverage on Cribl Edge hold; every other policy keeps opening findings. A finding already open keeps being evaluated and can recover and close. Cribl Edge and UAM tenants now use bank holidays (Suppress new alerts), and UAM custom policies can opt in. See *Bank Holidays*.
PR #3646	enhancement - Cribl Edge: Node connectivity from Splunk lookup tables, with Test selection	The nodes that must stay connected are selected from typed host names, with no count limit, and from up to 20 Splunk lookup tables on the local search tier or a remote deployment (up to 500,000 host names each, Exact or Wildcard match, read again every 15 minutes, last good names kept when a lookup cannot be read). Test selection shows which nodes the names select before saving. Collection health reports an unreadable, oversized or empty lookup (`node_lookups_failing`). See *Built-in policies*.
PR #3535	enhancement - Cribl Stream: live status of a Source or Destination	The detail of a Source or Destination shows its live status as reported by the Cribl Leader.
PR #3540	enhancement - Cribl connections: TLS options	Self-managed Cribl connections accept a custom CA certificate chain and a Verify SSL certificate toggle. A Guardian check warns while verification is off.
PR #3547	enhancement - Cribl Stream: Finding groups and the TrackMe severity scale	Cribl findings are grouped by object into Finding groups and use the TrackMe severity scale (informational, low, medium, high, critical). Every Cribl table can be sorted.
PR #3560	change - Cribl: least privilege	Cribl operations are authorized on the TrackMe capabilities of the tenant (user, power, administrator), and the request is escalated only after authorization. Splunk administrator roles are no

Table 19: Bug Fixes

Issue Number	Description	Details
#3444	bug - UAM reads search results through the v2 Search API	UAM collection reached search jobs through the deprecated v1 Search API. It now uses v2, with a guard against any return to v1.
#3448	bug - UAM Fetch SPL returns the right search text	When read live from the audit index, the search text is taken from the raw audit line instead of the automatically extracted field. The real denial reasons are reported instead of a generic failure (#3463).
#3466	bug - UAM run time and CPU of pooled and short searches	A pooled search's run time is the run time of its completion, not the age of the pooled process. Sub-second and unmeasured CPU no longer show as 0 s (#3465), and unsampled memory shows as not measured (#3483).
#3482	bug - UAM saved search view finds the runs of non-ASCII names	Saved searches with non-ASCII names now match their runs.
#3461	bug - UAM duration rounding at unit boundaries	Durations no longer show as 1m 60s or 1h 60m.
#3411	bug - UAM Collection health: a run with nothing to do shows its reason	A full listing that is not due shows as not due, in a neutral colour, instead of skipped.
#3427	bug - Bank holidays: pick full days on a calendar	Manual entry selects full days on a calendar instead of start and end hours.
PR #3650	bug - Manage AI Agents shows the Cribl Stream and Cribl Edge AI Agents	A Cribl Stream or Cribl Edge tenant was listed as a monitoring tenant, with the ML Advisor marked N/A and a Components Advisor switch that did nothing for it. Each findings tenant now shows its own AI Agent, with its enable toggle and a tab for its core settings. See *AI Agents automation*.
PR #3651	bug - UAM, Cribl Stream and Cribl Edge: every email delivery failure is logged with its cause	User communications, notifications, test emails and assignment emails now log the reason a send failed (the failed step and the transport, never a credential or an address) instead of a bare code or nothing. Assignment emails have their own log, `trackme_assignment_email.log`. The reasons shown in the UI are unchanged.
PR #3649	bug - FLX converging tracker: VOL tenants can be a source, dedicated tenants are left out	The tenant picker of the converging tracker (creation wizard and edit) offered no component for a VOL tenant and refused it. VOL is now offered; User Activity Monitoring, Cribl Stream and Cribl Edge tenants, which hold findings and not entities, are left out with a note saying why. The picker is searchable.
#3644	bug - VOL: a single outlier day no longer dominates the month projection and the 30-day trend	The month projection uses the median of the same weekday instead of its mean, and the 30-day trend is a robust slope as a percentage of the median, never below -100 %. The outliers card counts anomaly windows like the chart legend (PR #3645).
#3435	bug - VOL license configuration reload keeps the right revision	An untouched configuration draft adopts a reload together with its revision, so a later save is not refused as stale.
#3653	bug - FLX converging trackers can source from VOL tenants	The tenant picker of the converging tracker wizard and of Modify converging tracker refused a VOL tenant with "No enabled components". VOL tenants can now be selected. User Activity Monitoring, Cribl Stream and Cribl Edge tenants hold findings, not entities, so they are no longer offered; a message names them. The tenant list can be searched. See *Converging Trackers* (PR #3649).
PR #3615	bug - Cribl: Leader availability, worker versions and final review fixes	Leader API availability is monitored independently of the collection. Worker versions are compared with the running Leader version (PR #3597). Also fixed: AI Agent run isolation, Finding group repair, removal catch-up and disabled Workers (PR #3551); findings name their measurement, and notifications default to a configured account (PR #3544).
#3548	bug - Invalid metric events are rejected before they reach the component metric logs	Metric events are validated before they are written; a malformed point is skipped with a warning, and numeric measures are preserved in mixed payloads.

11.1.8 Version 2.4.18 - build 1790709110 (28/09/2026)

Hint

TrackMe 2.4.18 — Topology Studio grows up, a brand new Volume Outliers component, User Activity Monitoring in Beta, FQM at scale, Cribl signals, and much improved AI Routines.

- **Topology Studio.** Auto aggregates compose service health from connections: link entities, aggregates or views to one node and it becomes a live health parent, with nothing to duplicate. Build with AI generates them and previews every draft in a full-screen workspace you can refine before creating the view. Full screen is now a place to work, with drag, bulk selection, alignment, Undo and Save. Built-in help documents the whole product. The members map works for every composite node, view archives carry their version history, categories are picked or typed, and list filters survive navigation.
- **A brand new component: Volume Outliers.** Volume deviation detection remained a challenge for our customers, so we set out to make it out of the box, fully reliable and cheap to run. The new component tackles it like never before from a unique and controllable source, the Splunk licensing data, for Splunk Enterprise as for Splunk Cloud, with efficient low-cost logic. In a couple of clicks the tenant is set up, history is backfilled automatically, and detection and analytics are live from day one. Fully automated, it shows the consumers per index and for the whole environment, the major players and their trend, detects inactivity, makes drop and peak detection easy, and projects volume over time. Available to Foundation, Enterprise and Unlimited customers: understanding and identifying volume deviation, done simply and efficiently, once and for all. See *Volume Outliers*.
- **User Activity Monitoring (Beta): who is doing what on your Splunk?** TrackMe has always watched the data; the other half of a platform is the people and the automation that use it, and nobody looks at that half. UAM is a new kind of tenant, beside the entity tenants, that turns the audit trail and the introspection of a deployment into answers: the inventory of every scheduled search, user and role of each search head tier, every account classified as human, service or system, the search activity and the resource usage of each of them. A catalogue of eleven policies raises findings you can investigate, assign, resolve or dismiss: scheduled searches owned by a person, dispatched all time or in real time, accounts running all-time searches, launch bursts, costly searches, and the accounts that consume the search capacity, with thresholds derived from the measured capacity. Notifications follow the findings as a digest or per thread, the Collection health tab keeps the collection explainable and self-recovering, and the AI Findings Advisor investigates a finding on demand or on a schedule. The evidence is indexed compactly, collection is bounded and checkpointed, and a historical backfill makes the classification usable on day one. Works on Splunk Enterprise and Splunk Cloud, local or remote, on the Enterprise and Unlimited editions. Published as a Beta: the policies, their defaults and the evidence shapes may still move between beta releases, and your feedback shapes the GA. See *User Activity Monitoring* and the white paper *Who is doing what on your Splunk? User Activity Monitoring (UAM)*.
- **FQM: large upscaling.** Fields Quality Monitoring scales far beyond its former 10k-event ceiling: a single-pass aggregation makes monitor jobs 3 to 10 times faster, simulations of 100k events that used to fail now complete, and existing trackers are migrated automatically. The wizards are much improved: dictionaries stay editable during creation, with field types, a field-level AI advisor and Auto define sampling; the simulation runs as one job, twice as fast; Field Details shows a quality breakdown chart; entity Overview drilldowns went from minutes to seconds.
- **Cribl signals for DSM and DHM.** A new tracker source monitors data sources and hosts from activity metrics in a metrics index instead of searching the original logs: event delay, last observation and event counts, with the standard entity lifecycle. It reports activity observed in Cribl, not proof of indexing in Splunk. See *Cribl signals*.
- **AI Routines.** Routine evaluation is much improved: a live progress card shows what the model is doing, evaluations are bounded and evidence-checked, and a corrected retry recovers the run instead of failing it. Reliability across the board: Evaluate and Run-now run in a search process and survive REST handler recycling, routine context reaches the tools, and transient provider errors never restart the conversation.
- **Others.** Entity status messages render as readable structured views with a Raw JSON toggle. AI provider suggestions refreshed with GPT-6, Claude Opus 5.5, Fable 5.1 and Grok 4.7. Backup import survives

reverse-proxy timeouts. Two tenant RBAC gates hardened. Multi-line email footers accepted. Dependencies refreshed with a clean security audit.

SHA-256: a7417518bd9b353d94c83697faac0503094555f0ef02144819a539804b51dc66

Table 20: New Features

Issue Number	Description	Details
#3383	feature - Volume (Outliers): the seventh TrackMe component	Volume (Outliers) tracks the licensed volume of every Splunk index from the license usage logs, one entity per index in 5-minute buckets, read incrementally rather than by rescanning the internal index. It provides ML outlier detection on the rolling volume (drops and/or spikes) with tenant default KPIs and a scope by priority or filter expression, inactivity detection with a tenant policy and per-index overrides, the daily licensed volume per license day with a 30-day trend and a month projection, stateful alerting, and a history backfill at creation so detection and the cost trend work on day one. A Global license usage tab shows the whole environment: key figures, top consumers, movers, and pools and quotas on Splunk Enterprise. Works for Splunk Enterprise and Splunk Cloud, local or remote, and is available on Foundation, Enterprise and Unlimited. See *Volume Outliers*.
#3389	feature - User Activity Monitoring (Beta): a dedicated tenant type for who is doing what on the platform	A UAM tenant monitors one indexing layer through one or more search head tiers (the local search head and / or remote accounts). Five trackers collect: the inventory and the full listing of the saved searches, users and roles per tier; the activity tracker reads the audit trail from a durable checkpoint, five minutes at a time, and indexes one compact record per search and window with the REST access of an account summarised per batch, plus the resource usage of every search from the introspection and the measured search capacity; the policies tracker evaluates the catalogue; the backfill collects the recent history bucket by bucket. Accounts are classified system, human or service by detection, ordered rules or manual classes. Eleven policies in three families (inventory, activity, resources) raise one finding per policy and subject, six on by default, the resource thresholds derived from the capacity. The workspace has eight tabs: Findings with the Investigate view, Users with the account deep-dive, Activity, Scheduled searches, Policies with the policy insight, Collection health with recovery, Settings, Audit changes. Notifications are state-aware (digest or per-finding threads), a Guardian check watches the collection, and the AI Findings Advisor investigates findings. Available on Enterprise and Unlimited; the contracts may still move between beta releases. See *User Activity Monitoring*.
#3277	feature - Auto aggregates compose live service health, with AI generation and refinement	Add an auto aggregate or convert an existing label to combine connected entities, aggregates and saved views without repeating their tenant scopes or filters. Incoming child-to-parent links define membership; bidirectional links contribute either way, and hidden arrows retain their stored direction. Parent health follows the worst effective child state, including each child's health policy, while the healthy-entity KPI counts the deduplicated union of underlying entities. Nested composition and contributor drilldown are supported; empty or incomplete composition remains unavailable. Build with AI generates these parents and preserves them through refinement, and the AI Assistant can explain or propose edits using the same rules. Delivered in PRs #3274 and #3276; auto-arrange places a connection-defined parent before its contributors (#3288) and an emptied parent lets its alert resolve (#3278).
#3295	feature - Members map for auto aggregates and referenced view nodes	The inspector's members map (mini-map, member list, click-to-preview card, display cap) now appears for auto aggregates, showing the union of their contributors, and for referenced view nodes, showing every entity the view evaluates. The list is rebuilt under the reader's own permissions; restricted nodes stay inert placeholders.
#3229	feature - Preview and refine	Build with AI opens directly into a full-screen workspace. Au-

Table 21: Enhancements & Changes

Issue Number	Description	Details
#3330	enhancement - FQM Field Details shows the quality category breakdown as a chart	The Field Details modal of the FQM wizard shows each field's quality categories as a donut chart with a legend of counts and percentages, using green shades for passing categories and red, orange or pink for failures.
PR #3397	enhancement - Personal per-tenant preferences to show or hide pie charts and single values	The Tenant Home Settings panel gains a Personal preferences for this tenant section with Show / Hide toggles for the priority pie charts and the KPI single-value tiles. The choice applies to the current tenant across all of its component tabs, takes effect immediately, and is remembered per user.
#3391	enhancement - AI providers: Claude Sonnet 5.5 suggested and set as the Anthropic default	The Anthropic quick-pick list offers Claude Sonnet 5.5 as the default for new accounts. Forced tool selection is avoided for every Claude model newer than 5.0 by a version rule rather than a fixed list, so future releases are covered without a change.
#3325	enhancement - AI providers: refreshed suggested models	The quick-pick model lists and defaults in Manage AI Provider accounts add GPT-6 (astra, sol, luna) with gpt-6-sol as the OpenAI default, Claude Opus 5.5 and Claude Fable 5.1 for Anthropic, and Grok 4.7 as the xAI default.
#3304	enhancement - Export and import a view's version history	View archives can carry the view's version history, controlled by an Include version history toggle that is on by default on both export and import. Export and import open dedicated screens: the export screen explains what the archive carries and reports the history included; the import screen offers the toggle plus an optional new view ID and name, and handles an existing ID in place. The import summary reports versions imported, trimmed for retention and skipped with reasons; imported versions show an Imported origin in the version history. Archives from older instances remain compatible.
#3303	enhancement - Category field as a pick-or-type dropdown in the view form	The Category field of the Create and Edit topology view modals lists the categories already used by the views the user can see, so an existing grouping is picked without retyping it; typing still filters the list or creates a new category on save. The Tenant Home Save as topology view flow offers the same suggestions.
#3306	enhancement - Add auto aggregate from the canvas right-click menu; icons on every context menu	The canvas right-click menu now offers Add auto aggregate next to Add aggregate, opening the same modal as the toolbar. Every right-click menu row carries a leading icon matching the toolbar, choice rows show an accent check mark instead of a text prefix, and the menus are exposed to assistive technology as proper menu items.
#3282	enhancement - View-list filters are preserved and categories are a multiselect	Opening a topology view no longer discards the list search and category selection; both are restored when returning through the in-app Back button or browser navigation. The category control is a searchable multiselect with Select and Clear all; an empty selection means all categories. Filters last for the page session.
#3231	enhancement - Pre-check the tenant's enabled components in the aggregate scope builder	In the Add an aggregate node modal, choosing a tenant in a scope row now pre-selects every component that tenant has enabled, so authors untick exceptions instead of ticking every box. Switching the row to another tenant rebuilds the selection; the existing pair cap and completeness gate are unchanged.
#3234	enhancement - Display name pre-filled from the View ID when creating a view	The Create topology view modal pre-fills the display name from the View ID input until the author types a display name of their own; clearing it re-links the two. Edit mode and pre-seeded names never follow.
#3308	enhancement - AI Routines show live evaluation progress	Evaluation opens a progress card inside the routine form: the latest activity in readable language, the advisors' live tool activity with outcomes and durations, elapsed time, completed tool-call count, time limit, job reference and a Stop evaluation action. Closing or resetting the editor cancels the job.
#3292	change - Claude Opus 5.5 and other models rejecting forced tool selection	AI agent workflows use automatic tool selection for models that reject a forced selection, such as Claude Opus 5.5, while keeping the tools, output instructions and validated result parser. For

Table 22: Bug Fixes

Issue Number	Description	Details
#3241	bug - AI Routines Evaluate and Run-now survive REST handler recycling; routine context reaches the tools	Interactive Evaluate and Run-now now launch through the search-hosted agent executor, so a recycled REST handler process no longer kills the run; the synchronous Run-now lock conflict, cancellation and orphan detection are preserved. The bound routine is exported to the tool subprocess for the duration of the run, so the routine-aware tools work instead of refusing, which removes several ad-hoc search rounds per evaluation. Editing a contract field during an in-flight Evaluate now cancels it visibly instead of leaving the spinner on.
#3307	bug - AI Routine evaluations are bounded and evidence-checked	Evaluate no longer consumes its whole budget repeating searches or waiting through hidden provider retries: investigation is bounded, completed evidence is retained, time is reserved for a finalizer, and a failed model request is retried within the remaining deadline. Totals and distinct entities are aggregated exactly before sampling, evidence references and tenant coverage are validated, and an inconclusive verdict keeps Create disabled and suppresses notifications. A corrected retry recovers the original failure when it covers the same evidence scope, a real fire's model-call timing is honoured, and transient provider errors are retried by the middleware without restarting the conversation (#3297).
#3393	bug - AI agents: step-budget soft landing no longer edits the system prompt mid-run	The soft-landing instruction that asks an agent to finalize near the end of its step budget is now delivered through tool results instead of a rewritten system prompt. Claude Opus 5.5, Sonnet 5.5 and Fable 5.1 bind their reasoning to the conversation prefix and rejected the next request when the prompt changed mid-run.
PR #3395	bug - AI ML Advisor no longer exceeds the token limit on its first tool call with Claude models	The entity description returned to advisor tools no longer carries the chat-only knowledge channel, and embedded entity notes are bounded to the most relevant ten, so an entity with hundreds of automated notes no longer produces a half-megabyte tool result. Claude runs that stopped within seconds on the token limit now complete like the other providers.
PR #3396	bug - Concierge proposal audit events carry the entity identifier from the entity chat	Audit events for Concierge proposals made from an entity conversation now populate the entity identifier, which was read from the wrong level of the description payload.
#3386	bug - AI agents: a second run in the same process failed with "Event loop is closed"	Every AI agent run now gets its own HTTP client, so scheduled advisor batches and the REST fallback path no longer fail intermittently on their second run in a process.
#3354	bug - Security: tenant RBAC gates hardened on two loopback paths	The entity table endpoints and the component summary update endpoint looped back to an internal load with the system token, which let a user read the entity table or the aggregate counts of a tenant their roles do not grant, and rewrite that tenant's summary. Both paths now gate the real caller (#3363).
#3364	bug - Virtual tenant administration hardening	A forced enable or disable of a missing tenant answers HTTP 409 instead of crashing; the RBAC update initialises the WLK enablement flag instead of duplicating the FLX default (#3350); force clearing tags through the API no longer requires the manual tags field (#3358); creating a tenant without a name answers HTTP 400 with an explicit message; a deleted tenant no longer leaves or regains an execution summary record.
#3385	bug - Tenant lifecycle audit events indexed with the right outcome	Successful tenant deletions were indexed as failures and failed ones as successes, and disable events were indexed as deletions. The audit event now matches the outcome of its branch.
#3327	bug - Topology Studio entity drilldown waits out Splunk search-concurrency limits	The inspector sparklines and View status charts retry when Splunk temporarily refuses a search for concurrency reasons, with growing waits, never more than three drilldown searches at once per page, and cancellation of abandoned searches, instead of failing immediately.
#3332	bug - FQM allowed-missing and allowed-empty categories	Field quality categories are matched exactly in the decision maker, so a field allowed to be missing or empty is no longer

11.1.9 Version 2.4.17 - build 1789363319 (14/09/2026)

Hint

TrackMe 2.4.17 — A faster AI Assistant, Concierge actions from within the Assistant, and targeted fixes for AI Advisors on Splunk-hosted models, Topology Studio, Backup & Restore, Tenant Home and the policy engine.

- **AI Assistant: faster and cheaper.** Streaming answers no longer wait on KV Store on every token. **Response times drop** from tens of seconds to close to the model's own generation time. Chat contexts are rendered **static knowledge first**, so **prompt caching now hits** on every turn after the first for Anthropic, OpenAI, Azure and Gemini providers.
- **Concierge actions from within the Assistant.** A new **Ask Assistant / Concierge actions** selector lets users request **action proposals** directly. The Concierge returns validated proposals in the usual **confirmation cards**; nothing runs without confirmation. The Assistant itself **suggests actions more reliably** and no longer promises execution from a conversational reply. The Concierge agent now completes in **two tool calls** for common requests.
- **AI Advisors on Splunk-hosted models.** Requests through the Splunk-hosted SDK bridge **honour the input token cap and the configured completion allowance**. This fixes gateway errors seen in particular with the **ML Advisor**.
- **Topology Studio.** Dynamic views created from Tenant Home now include **monitored entities only**; disabled entities no longer degrade aggregate health. The **custom hierarchy** dialog is refined: aligned rows, levels added on demand, up to ten grouping fields.
- **Backup & Restore on Splunk Cloud.** When the Python sandbox reports a jailed hostname, TrackMe now resolves the **real SHC member name** and stores it in the archive metadata, so download, export, restore and delete target the right member.
- **Tenant Home.** The **lagging policy** and **variable delay** screens now open on the **current entity values** immediately after an update, with additional shadow synchronisation fixes. **Bulk label** actions are much faster, submitting up to 100 entities per request.
- **Priority, tags and SLA policies.** Policy changes now **show up immediately** on Tenant Home, including on shadow-served tenants. An assignment **exists only while a policy matches**: deleting the last tags policy clears the auto tags, and priority or SLA rows are removed when an entity stops matching, with the entity keeping its value. New **Force clear all tags** action, per entity and in bulk. Entity **tags now survive tracker cycles** on every component.
- **Health tracker and AI audit.** Daily **collection inspections** now run for **every component**, and a **corrupted entity record** is skipped instead of aborting a run. **Token usage** is now captured for OpenAI-compatible streaming chats, so AI audit dashboards report real counts.

SHA-256: 27c7f37ff4559e0d8f6b768c970e18c800cd00d94fb25db904fbf0c3fb01d2f3

Table 23: New Features

Issue Number	Description	Details
#3198	feature - Concierge actions available from within the AI Assistant	The Assistant composer gains a persistent Ask Assistant / Concierge actions selector. In Concierge actions mode, the request launches the Concierge agent with the current entity or page scope and the recent conversation, shows progress, and renders the validated proposals in the existing confirmation cards; sending a request never executes a change. Automatic routing in Ask Assistant mode now distinguishes concrete changes (monitoring, priority, acknowledgements, maintenance, labels, deletion) from specialist investigation, and the Assistant no longer implies execution from a conversational confirmation. Each mode carries concise guidance. Concierge actions mode requires Splunk 10.2 or later with Python 3.13.
#3221	feature - Force clear all tags, manual and policy-derived	A new tags operation resets an entity's manual tags, policy-derived tags and its tags assignment row in one action, so the decision maker cannot re-apply them on read. It is available from the Tags modal behind its own confirmation and as a bulk edit action with an explanatory warning. Policies stay authoritative: the next tags policy run re-tags whatever still matches.

Table 24: Enhancements & Changes

Issue Number	Description	Details
#3206	change - AI Assistant chat path performance	Streaming answers are persisted to KV Store at most twice per second instead of on every token chunk, the KV collection handle is resolved once per stream, the chat request returns its job immediately and builds the context in the worker, and the per-collection configuration lookups on the request path are removed. Chat wall time is now close to time-to-first-token plus model generation time, and concurrent chats no longer degrade each other.
#3207	change - Prompt caching renders static knowledge first	Every chat context is rendered as the static knowledge reference first and the live state last, with the Anthropic cache marker on the static block only. The static block, about 90 percent of the prompt, is now read from cache on every turn after the first; the stable prefix also lets OpenAI, Azure and Gemini automatic caching engage. Cached token counts are reported in the AI chat log.
#3208	change - Concierge agent held to its two-tool-call path	Prompt discipline, a duplicate-call breaker on the static catalog tools and a step-budget countdown remove redundant model rounds from the Concierge agent. Common single-target requests complete faster with no change to what reaches the confirmation card.
#3202	enhancement - Refined custom hierarchy dialog for Tenant Home topologies	In the embedded Topology Studio, the custom hierarchy dialog aligns each level with its control, adds levels on demand instead of showing three fixed rows, allows removing intermediate levels, and supports up to ten distinct grouping fields.

Table 25: Bug Fixes

Issue Number	Description	Details
#3194	bug - AI Advisors on Splunk-hosted models honour input limits and completion configuration	Agent requests through the Splunk-hosted SDK bridge bypassed the input token cap and omitted the configured completion allowance, which could fail the request on the gateway, in particular for the ML Advisor. The shared transport now estimates the full prompt, shortens only local read-tool results when needed, retries a rejected request a bounded number of times without replaying tools, and forwards the configured completion allowance with the correct parameter for reasoning models.
#3211	bug - Splunk-hosted chat trims the static knowledge block first	When a chat prompt exceeds the Splunk-hosted input cap, the trimmer now removes content from the static knowledge block before touching the live entity state and guidance, so the answer keeps the context it depends on.
#3200	bug - Dynamic views from Tenant Home include monitored entities only	The embedded Topology Studio applies an implicit enabled monitored-state filter when generating the temporary topology. Disabled entities are excluded from selection and aggregate health, their count is disclosed in the preview, and saved views carry the filter in each aggregate's condition.
#3209	bug - Backup & Restore resolves the real SHC member name on sandboxed Splunk Cloud stacks	When the Python sandbox reports a jailed placeholder hostname, Backup & Restore and Support Diagnostic now resolve the real member name from the request context or from the Splunk server information endpoint. Archive rows carry the correct server name, so download, export, restore and delete delegate to the right member. Healthy deployments are unaffected.
#3212	bug - Lag monitoring policy modal opens on current thresholds on shadow-served tenants	Derived delay and lag duration values are recomputed whenever the underlying thresholds are patched on the shadow record, and the DSM and DHM lag-policy modals are seeded from the raw values. The table, the entity inspector and the modal now reflect an applied policy immediately.
#3215	bug - Lag monitoring policy modal reflects a static or variable delay switch immediately	The variable-delay policy is mirrored onto the shadow record as part of the write, and the lag-policy modals resolve the effective policy from the variable-delay collection when they open. Applying, enabling, disabling or reverting a variable-delay configuration is reflected without a manual refresh.
#3220	bug - Policy changes refresh the shadow read cache; deleting the last tags policy clears auto tags	Running a priority, tags or SLA policy tracker, calling the policy apply endpoints or editing manual tags now refreshes the shadow read cache when entity records were rewritten, so the change is visible on Tenant Home immediately instead of after an unrelated enrichment. Deleting the last tags policy of a component now clears the policy-derived tags from every entity: the tags tracker proceeds with zero policies while assignment rows still carry auto tags, and the delete action dispatches it right away.
#3221	bug - Priority and SLA assignments exist only while a policy matches	Priority and SLA assignment rows of entities that no longer match any policy are removed on the next apply, and the entity is handed back to entity-managed while keeping its current value. The cleanup is fail-safe: when any policy could not be evaluated, for example an unavailable lookup or remote account, existing rows are kept for that run. Deleting an SLA policy now removes its assignment rows.
#3222	bug - Entity tags survive tracker cycles on every component	On DHM, MHM, FLX, FQM and WLK, the entity tags field was dropped on the next tracker cycle, which made the tags policy rewrite every tagged entity on every run and hid the field from raw KV Store and SPL consumers. The tags are now carried through the persistent-field pipeline of every component. The force clear tags action also reports a failure when the assignment row could not be removed, and the priority un-stamp recognises the persisted reason spelling.
#3217	bug - Token usage captured on OpenAI-compatible streaming chats	The OpenAI-compatible streaming chat path now requests and parses the final usage chunk, so AI audit dashboards no longer report zero completion tokens for OpenAI, Azure OpenAI, xAI

11.1.10 Version 2.4.16 - build 1789035950 (10/09/2026)

Hint

TrackMe 2.4.16 — Create live Topology Studio views from Tenant Home in a couple of clicks, with configurable hierarchies, more reliable tracker charts and refreshed dependencies.

- **Topology Studio is now embedded across Tenant Home.** From the topology action in any TrackMe monitoring component, users can open an automatically generated, temporary Studio canvas based on the current tenant, component or entity selection. The preview shows live health immediately, starts with a clear aggregate view and can optionally include individual entities. Users can search, filter, move and arrange the graph, then save it as a normal Topology Studio view when they want to keep and refine it. The existing Link graph and Network graph remain available as alternate tabs. See *Creating dynamic views from Tenant Home* for the walkthrough, including a Cribl Logstream filtering example.
- **Useful topology structure in a couple of clicks.** Automatic views use component-aware grouping by default. Users can switch to an existing Tenant Home grouping preset or build an ordered hierarchy from up to three available fields, including tags and labels. Root and intermediate groups show their own live aggregate health, and overlapping memberships are counted once.
- **More reliable KPI timelines for scheduled trackers.** TrackMe-generated timestamps now follow the intended scheduled run time across the tracker execution chain. This prevents successful runs from appearing as gaps or duplicate neighbouring buckets when scheduler delay or processing time crosses a chart boundary. Timestamps provided by the monitored search remain unchanged.
- **Two focused administration fixes.** Tenant RBAC reassignment no longer requests a nonexistent shared native-ML object, and macro shortcuts from hybrid tracker management and Tenant Knowledge Objects work with the updated manager interface in Splunk Cloud 10.5 while retaining compatibility with earlier supported releases.
- **Dependency and security maintenance.** The bundled Splunk Python SDK moves to version 3.0.1. JavaScript and Python dependencies were refreshed for the release, including security updates for the mapping and configuration-processing libraries; the final JavaScript audit reported no known vulnerabilities.

SHA-256: 49eb589d5941ad531002f408bc1f301516cf905a3e9d13d54d541ce7ceb269f4

Table 26: New Features

Issue Number	Description	Details
#3174	feature - Embedded Topology Studio previews in Tenant Home	The topology action available across TrackMe monitoring components now opens a temporary Topology Studio canvas generated from the current tenant, component or entity selection. It shows live health without creating anything automatically, starts with aggregates only and can optionally include individual entities. Users can apply a condition, search the canvas, move nodes, undo changes, auto-arrange, export a PNG and explicitly save the result as a normal Topology Studio view. The saved view retains the represented membership, condition and layout for continued authoring in Studio; the existing Link graph and Network graph remain available.

Table 27: Enhancements & Changes

Issue Number	Description	Details
#3177	enhancement - Configurable Tenant Home topology hierarchies and aggregate roots	Automatically generated topologies now show live aggregate health at the root and at every grouping level. Users can keep the component-aware default, choose an existing Tenant Home grouping preset or create an ordered hierarchy from up to three available fields, including tags and labels. Membership is deduplicated across overlapping groups, and excessive graph expansion is reported clearly instead of producing a partial view.
#3180	change - Dependency and security refresh for 2.4.16	JavaScript and Python dependencies were refreshed within the release compatibility boundaries, including security updates for MapLibre, colord and js-yaml and updated platform libraries. The final JavaScript dependency audit reported no known vulnerabilities.
#3183	change - Splunk Python SDK 3.0.1	The bundled Splunk Python SDK is updated from 3.0.0 to the corrected 3.0.1 patch release. Existing TrackMe AI, search-command and connection behavior remains compatible.

Table 28: Bug Fixes

Issue Number	Description	Details
#3169	bug - Scheduled tracker timestamps no longer create phantom KPI chart gaps	When TrackMe must generate a timestamp for a scheduled FLX, FQM or WLK tracker result, it now anchors that timestamp to the intended scheduled run instead of the later processing time. Successful runs therefore remain in the expected chart bucket even when scheduler delay or processing latency varies. A timestamp supplied by the monitored search is still preserved.
#3171	bug - Scheduled time is preserved through tracker wrapper jobs	The tracker executor now carries the intended scheduled time into the child job that performs parsing and metric generation. This completes the timestamp correction for normal scheduled trackers while ad-hoc runs retain wall-clock behavior.
#3175	bug - Tenant RBAC reassignment no longer reports a native-ML transform error	Reassigning a Virtual Tenant's owner or roles no longer requests a nonexistent shared native-ML transform. Native model permissions continue to be applied to the correct tenant-scoped objects, while real permission failures remain visible.
#3185	bug - Macro manager shortcuts work on Splunk Cloud 10.5	Macro shortcuts from hybrid tracker management and Tenant Knowledge Objects now use the current manager parameters, so they open the intended filtered macro listing on Splunk Cloud 10.5 while preserving compatibility with earlier supported releases.

11.1.11 Version 2.4.15 - build 1788693415 (06/09/2026)

Hint

TrackMe 2.4.15 — Topology Studio, the complete picture: views inside views, view integrity checks, a redesigned views browser, and Tenant Home stability at very high scale.

- **Views inside views — cross-view topologies.** A saved topology view can now be placed on another canvas as a **fourth node type**, next to entities, aggregates and labels. The referenced view keeps ownership of its graph and policies; the parent shows its live health (the worst effective state of the child's health nodes, nested views included) and a KPI counting the unique underlying entities. View references are alertable,

exportable and understood by **Build with AI** — the step from *a map* to *a map of maps*. See *Node types*.

- **TrackMe now watches the views themselves.** A new Configuration Guardian check detects views that reference deleted or disabled tenants, removed components or aggregates that no longer resolve — and the Topology Alerts silently broken by it. Affected views carry an integrity badge, an integrity panel explains every finding, and an editor banner offers **in-place remediation**. See *View integrity checks*.
- **A redesigned views browser. Cards**, **Compact cards** and a sortable **List** layout, a **category filter** combined with search, a deployment-wide default display mode set by administrators, and a per-user choice that persists. See *Views browser*.
- **A better editor.** Large views show a continuous loading progress instead of a greyed-out canvas; the inspector opens at a viewport-derived size and is clearly resizable; aggregate members gain **View status**, **View details** and **Open in Tenant Home** drilldowns; aggregates scale to **500 auto-generated members** with an independent auto-entities filter and a configurable **auto-arrange**.
- **Topology Alerts honour global maintenance mode.** Enabling maintenance — immediate or scheduled, all tenants or a scope — freezes every Topology Alert whose selection touches a covered tenant: no phantom incidents or emails during the window.
- **Tenant Home — stability at very high entity counts.** Deep browser-memory work removes repeated scans and retained datasets, parses large responses in batches, and preserves the last good data with a Retry when a refresh fails, so the browser tab stays healthy on very large tenants.
- The release also brings image decoration background colours, a Guardian counter fix, two tenant metric-index fixes, and the cycle's dependency refresh with four security updates.

SHA-256: cce6d8be9c77f224e383e854d520bb8fa42a3aac408c8bc4a6330385758b90a0

Table 29: New Features

Issue Number	Description	Details
#3160	feature - Topology Studio: views inside views (cross-view topologies)	A saved topology view can be placed on another canvas as a first-class **view node**. The child keeps ownership of its graph, aggregate policies and alerts; the parent stores only its identifier and shows its live health — the worst effective state of the child's health nodes, nested views and dependency propagation included — with a KPI counting the unique underlying entities across aggregates, direct entities and nested views. A view reference participates in alert selection, scoped maintenance, email rendering, history, duplication and portable export. Per-hop view and tenant permissions, cycle detection and bounded traversal are enforced server-side, and Build with AI can compose view references from a caller-authorised view directory.
#3136	feature - Topology Studio: view integrity linter with Guardian check and in-app remediation	A new Configuration Guardian check, `topology_view_stale_references`, detects views referencing deleted or disabled tenants, components removed from a tenant, or aggregates that no longer resolve, and the Topology Alerts broken by it (orphaned, selecting nodes that no longer exist, not evaluating, or emailing through a missing account). Verdicts refresh on every view and alert write and on tenant lifecycle changes. In the app, affected views carry an integrity badge in the views browser, an integrity panel explains each finding, and an editor banner lets authors remove or re-target stale items in a few clicks. Users see what is wrong on their own views; administrators get the fleet-wide picture through the Guardian.
#3156	feature - Topology Studio: views browser layouts and default display mode	The views browser offers three layouts — **Cards** grouped by category, **Compact cards** and a sortable **List** table (view, category, nodes, edges, updated, access, integrity, with open/edit shortcuts and a permission-aware actions menu). Administrators set the deployment-wide default display mode; each user's own choice persists and takes precedence.
#3142	feature - Topology Studio: category filter and remembered display mode	The views browser gains a **category filter** that combines with the text search, and remembers the selected display mode per browser.
#3095	feature - Topology Studio: auto-generated members up to 500 per aggregate	Auto-populated aggregates can expand up to 500 member nodes, with an aggregate-wide default icon for generated members and a toggle for the auto-generated marker.
#3097	feature - Topology Studio: independent auto-entities filter	A second-layer filter chooses which auto-generated members appear on the canvas, while the aggregate's health stays computed on its full scope.
#3092	feature - Topology Studio: configurable auto-arrange	Auto-arrange offers four hierarchy directions, three spacings, wrapping and explicit canvas handling for generated members.
#3144	feature - Topology Studio: image decoration background colours	Image decorations can carry a solid background colour so transparent artwork stays readable on any canvas background.

Table 30: Enhancements & Changes

Issue Number	Description	Details
#3157	change - Topology Studio: continuous loading progress	Large views no longer sit greyed out while live status loads: one compact progress bar tracks graph preparation and status transfer, never moves backward across request restarts, and keeps the canvas visible and usable throughout.
#3090	change - Topology Studio: inspector sizing and resize affordance	The details inspector opens at a size derived from the viewport instead of a fixed width, and is clearly resizable by pointer and keyboard with an accent-coloured grip.
#3118	change - Topology Studio: responsive member-map labels	Member-map labels in the inspector adapt to the available width and stay identifiable.
#3124	change - Topology Studio: aggregate member entity actions	Aggregate member previews gain **View details** and **Open in Tenant Home**, with explicit tenant identification — the same destinations as entity nodes.
#3129	change - Topology Studio: focused entity status shortcut	A **View status** action opens a compact in-Studio status card for an entity, expandable in place to the full details.
#3150	change - Tenant Home: browser-memory work at very high entity counts	Repeated refreshes no longer pin obsolete datasets through render closures; repeated table scans and optional-data allocations are removed; large shadow-cache responses are parsed in batches of about 1 MiB instead of holding the whole decoded string next to the parsed objects; and a failed refresh preserves the last good data with a stale-data warning and Retry instead of emptying the table.
#3155	change - Tenant Home: bounded bulk host-blocklist suggestions	Bulk host-blocklist suggestions load through lazy, cancellable workers that retain only distinct index and sourcetype names.
#3086	change - Dependency maintenance and security updates	Third-party dependencies were refreshed for the 2.4.15 cycle across two consolidated waves, with security-driven updates to `postcss-selector-parser` (CVE-2026-9358), `qs`, `@humanfs/node`, `fflate` and `browserslist`, and `splunk-sdk` frozen to an exact reviewed pin.

Table 31: Bug Fixes

Issue Number	Description	Details
#3152	bug - Topology Alerts: evaluation frozen while global maintenance covers the selection	Enabling maintenance mode (immediate or scheduled, all tenants or a scope) now freezes every Topology Alert whose selection touches a covered tenant, so no incidents, transitions or emails are produced during the window; the pause reason is visible on the alert.
#3153	bug - Topology Alerts: bounded and consistent email rendering at 500 members	Alert emails stay readable and cheap after the 500-member expansion ceiling: member rendering and spark series are bounded, and one shared member ordering is used by live status and the email render.
#3101	bug - Topology Alerts: email PNG lost the graph after a text decoration	The rendered topology image in alert emails no longer loses the graph when a text decoration is present — a clip path was masking everything drawn after it.
#3122	bug - Topology Studio: complete metric list for aggregate members opened from the mini-map	An aggregate member opened from the mini-map now shows its complete metric list and sparklines instead of a clipped subset.
#3117	bug - Topology Studio: inspector resize handle stays visible	The inspector's resize handle stays sticky and reachable while long inspectors scroll.
#3140	bug - Configuration Guardian: available checks count on page load	The *Available checks* counter is populated on page load, independently of the selected tab and permissions.
#3154	bug - Tenant effective metric index honoured in the FLX group-rename migration and tracker-health self-heal	The FLX group-rename metrics migration and the tracker-health metric-datatype self-heal now resolve the tenant's effective metric index; the self-heal had compared the wrong slot key since its introduction.

11.1.12 Version 2.4.14 - build 1788188553 (31/08/2026)

Hint

TrackMe 2.4.14 — Topology Studio shapes, images and text, real email conversation threading, Configuration Guardian improvements, critical FIPS fix.

- **Topology Studio — the canvas becomes a diagramming surface.** Draw **shapes**, drop **images** and place **free text** directly on the canvas. Style everything from the context menu, align decorations as part of bulk selections, and let image-aware size limits keep large decorated views safe. Set an explicit **canvas size** per view — existing views keep their current presentation — and find all creation actions grouped in a compact toolbar.
- **Stateful alert emails — real conversation threading.** Alert emails now thread correctly in standards-compliant mail clients: RFC 5322 compliant `Date`, `Message-ID` and `References` headers, plus a **static or templated incident subject** for clean Outlook/Exchange conversation grouping.
- **Configuration Guardian — three ways better.** The sourcetype-explosion safeguard becomes a first-class managed Guardian check with enable, snooze and threshold tuning; SHC member discovery works on newer Splunk Cloud stacks; and UI updates preserve your catalog context.
- **Critical — FIPS-enabled Splunk 10.4.x+ compatibility.** A TLS call rejected by the OpenSSL FIPS provider broke every TrackMe page with "Splunk Service Unavailable". The call is now guarded — FIPS deployments on Splunk 10.4.x+ should prioritise this upgrade.
- The release also brings policy application at very large scale, an SLA manual-override contract fix, subgroup-aware group-by options for FLX and WLK, and the cycle's dependency refresh.

SHA-256: 64f2985a2c53f7ebd9b3438a00340a7eab40aa0a9e87098d677f729de9202606

Table 32: New Features

Issue Number	Description	Details
#3066	feature - Topology Studio: canvas shapes and images	Topology views become genuine diagrams: draw shapes and drop images directly on the canvas, then move, resize and style them like any other canvas element. Shapes offer multiple forms with configurable line style, colour and width; images embed into the view itself (PNG and JPEG, with transparency control and aspect-ratio lock), so duplicated views, version history and portable exports remain fully self-contained. Decorations participate in bulk selections and alignment actions alongside nodes, and the whole-graph size limit is image-aware, keeping heavily decorated views within safe storage and rendering bounds. Server-side validation enforces the same size, byte and pixel limits as the editor.
#3072	feature - Topology Studio: text canvas decorations	Free text can be placed anywhere on the canvas: titles, zone labels, annotations. Text decorations support font family, style, size and colour, all adjustable inline from the context menu, and render identically in the editor, in PNG exports and for view-only users.
#3078	feature - Topology Studio: canvas size control	Authors can set an explicit logical canvas size per view from a new **Canvas size** dialog. Fully backward compatible: existing views keep the historical fit-to-content presentation until an author applies a size, and the dialog suggests a size derived from the view's current visible canvas, so accepting it preserves the presentation. Once a size is applied, centered and full-screen modes fit the complete canvas boundary, content stays visible when shrinking, and automatic canvas growth, undo and save/reload behave as before.
#3037	feature - Alerting: static incident subject for Outlook/Exchange conversation grouping	Stateful alert emails can use a **static subject** across the whole incident lifecycle (opened, updates, closed), so clients that group conversations by subject — Outlook and Exchange in particular — keep the entire incident in one conversation thread.
#3044	feature - Alerting: configurable incident subject template	The incident email subject becomes a configurable template with token substitution, so teams can shape the subject line to their routing and filtering conventions while preserving the threading behaviour.
#3062	feature - Configuration Guardian: managed sourcetype-explosion safeguard	The DSM sourcetype-explosion safeguard (which blocks runaway per-index sourcetype creation) becomes a first-class managed Guardian check: enable or disable it, snooze it, and tune the per-index threshold from the Guardian management UI, with the standard Guardian alert lifecycle including email notifications. The legacy hidden setting is imported automatically during the upgrade and historical alerts are migrated to the canonical Guardian collection.
#3050	feature - Policies: expose `_key` as a selectable entity field in policy field mappings	Lookup-based and search-based policy field mappings can now target the entity `_key` directly, enabling exact-identity mappings where object names are ambiguous.
#3053	feature - Entities tables: subgroup-aware group-by options for FLX and WLK	When FLX or WLK entities carry a subgroup, the entities tables offer subgroup-aware group-by options, so grouped views follow the structure of your Flex use cases and workload apps.

Table 33: Enhancements & Changes

Issue Number	Description	Details
#3077	change - Topology Studio: compact authoring toolbar	The six canvas authoring buttons are grouped into two keyboard-accessible menus — **Add items** (entities, aggregates, labels) and **Add decoration** (shapes, images, text) — keeping the full toolbar on one line across more viewport widths.
#3048	change - Policies: policy application at very large scale	Policy application now runs in-process in the policy trackers instead of a blocking REST call, removing the 600-second ceiling that aborted runs on very large tenants. Reads are projected to the fields the apply needs and writes are delta-only, so applying policies over hundreds of thousands of entities completes reliably and efficiently.
#3034	change - Dependency maintenance	Third-party dependencies were refreshed for the 2.4.14 cycle across two consolidated waves, including a security-driven update to `brace-expansion`.

Table 34: Bug Fixes

Issue Number	Description	Details
#3061	bug - Critical: FIPS-enabled Splunk 10.4.x+ compatibility	On FIPS-enabled Splunk 10.4.x+ deployments, an unguarded TLS `set_groups` call in the bundled splunk-sdk was rejected by the OpenSSL FIPS provider, breaking every TrackMe page with "Splunk Service Unavailable". The call is now guarded at build time with a safe fallback. FIPS deployments on Splunk 10.4.x+ should prioritise this upgrade.
#3035	bug - Emails: RFC 5322 `Date` and `Message-ID` headers on all outgoing emails	Every email sent by TrackMe now carries RFC 5322 compliant `Date` and `Message-ID` headers, improving deliverability and giving mail clients a reliable identity for every message.
#3043	bug - Emails: spec-correct `References` chain on stateful alert lifecycle emails	Stateful alert lifecycle emails (opened, updates, closed) now build a spec-correct `References` chain containing ancestors only, so mail clients thread the incident reliably. Legacy threading records are healed automatically and chain growth is capped for long-running incidents.
#3049	bug - Policies: SLA manual-override contract honoured	Manually assigned SLA classes are no longer overwritten by SLA policies: the manual-override guard is enforced, policy writes are no longer stamped as manual assignments, and existing installations are cleaned up automatically during the upgrade.
#3055	bug - Configuration Guardian: SHC member self-heal discovery on newer Splunk Cloud stacks	Search Head Cluster member discovery for the Guardian self-heal checks is now dual-source and system-configurable (SPL and earliest time), fixing empty member discovery on newer Splunk Cloud stacks.
#3064	bug - Configuration Guardian: catalog context preserved during updates	Guardian UI updates no longer reset the catalog context: the current filters and view state are preserved while alerts refresh.
#3068	bug - Topology Studio: context-menu inspect state aligned with the selection	The context menu's inspect state now stays aligned with the current selection, removing cases where the menu offered inspect actions for a previously selected element.

11.1.13 Version 2.4.13 - build 1787307707 (21/08/2026)

Hint

TrackMe 2.4.13 — Topology Studio sparklines and better AI, notes management improvements, label assignments at scale and in bulk, security fixes.

- **Topology Studio — Sparkline KPI display style: 24-hour trend lines on the canvas.** A new Topology Studio KPI display style renders each selected KPI as its label, current value and a 24-hour mini trend line directly on the node — trends at a glance across the whole topology without opening a single panel. Scalable by design: the complete view's trend history is fetched by one single batched search per refresh, regardless of how many nodes carry the style. The aggregate member preview card gains the same Metrics (24h) sparklines.
- **Topology Studio — Build with AI.** Describe the topology you want in plain language and TrackMe generates a complete draft: aggregates with scopes and filters, pinned entities, dependency edges — laid out top-down and validated server-side. The draft lands on the canvas as a pending change you can review, adjust and undo before saving. Available from the editor toolbar, from the views index (create the view and open it in one step), and proposed by the AI Assistant in conversation when a topology would answer your question. Generation runs as a dedicated AI Builder agent, RBAC-scoped server-side to the tenants you select, and works across all supported AI providers.
- **EU AI Act Article 50(2) — machine-readable marking of AI-generated content.** AI-generated email content and every AI Assistant chat response now carry a machine-readable "artificially generated" marking; deterministic (non-AI) content remains unmarked by design.
- **Label assignments can no longer lose data under pressure.** An important fix closes a path where label edits performed under concurrent write pressure could drop existing assignments and automatic-label metadata. Deleting a label that an enabled automatic rule still references is now refused, so a deletion can no longer silently break the rule.
- **Backup & Restore portability.** Convert a zstd archive to a portable gzip `.tgz` with automatic download, and export or import a **complete run bundle** — a whole backup run as one portable artifact — for deployment migrations.
- **FQM collect-job wizard: richer benchmarking.** The benchmark step gains event-distribution timecharts and a retrieved-events metric above the charts, so you can judge coverage and volume before creating the tracker; the generating-search type no longer breaks the CIM simulation, and the data-dictionary Test regex search now honours the trackers' summary index(es).
- **Email delivery: one SMTP transport, per-account certificate control.** Every send path now selects the SMTP transport the same way, and each email account gains an SSL certificate verification toggle for environments with internal certificate authorities.
- **Notes management: clear automated notes without touching yours.** TrackMe's system-generated notes (adaptive-delay summaries, threshold-lock reconciliations) can now be cleared selectively — all of them, or all except the most recent — and notes gain bulk management, so user-authored notes are never collateral of a cleanup.
- **Tenant isolation, completed server-side.** The remaining sibling read endpoints (cached summary counts and shadow-cache probes) now enforce the caller's tenant visibility exactly like the main entity reads — completing the server-side read-gate work started in 2.4.9.
- The release also carries the cycle's dependency refresh, including security fixes.

SHA-256: f411c03e5b3f3e3bd233c167e558d44e01986cfb8084e52e5d1b86b9758d6e2f

Table 35: New Features

Issue Number	Description	Details
#2947	feature - Topology Studio: Build with AI — natural-language topology generation	Describe the topology you want in plain language and TrackMe generates a complete draft: aggregates with tenant scopes and filters, pinned entities and dependency edges, laid out top-down and validated server-side before anything reaches the canvas. The draft is applied as one pending change — review it, adjust it, undo it, and nothing is stored until you Save, where the normal version-history flow applies. Build with AI is available from the Topology Studio editor toolbar, from the views index (the view is created and opened in one step), and through the AI Assistant, which proposes the Builder in conversation when a topology would serve the discussion. Tenant scoping is enforced server-side against your visible tenants, and entity resolution honours per-tenant permissions. Generation runs as a dedicated AI Builder agent rather than the chat assistant, making it fast and reliable across all supported AI providers; the Topology Studio chat context also became dramatically lighter, making topology conversations faster on every provider. Requires a configured AI provider; interactive only. A REST surface is available for automation users and documented in the REST API Reference.
#2948	feature - Topology Studio: Sparkline KPI display style — 24-hour trend lines on the canvas	A new entity KPI display style, **Sparkline**, renders each selected KPI as its label, current value and a 24-hour mini trend line directly on the canvas — the same at-a-glance history as the node inspector's Metrics (24h) panel, across the whole topology at once. Scalable by design: the complete view's trend history is fetched by one single batched search per refresh regardless of how many nodes carry the style, so it is safe on wall-of-screens deployments and large canvases; the fast live-status refresh stays search-free and keeps the KPI values current. Available on entity nodes and on expanded-aggregate member nodes through member KPI defaults, with honest degradations: a dashed placeholder when a KPI has too little recorded history, real trend lines in PNG exports, and a stat-card fallback in Topology Alert emails.
#2946	feature - Topology Studio: Metrics (24h) sparklines in the aggregate member preview card	The aggregate member preview card now renders 24-hour metric sparklines for the previewed member, bringing the node inspector's trend view into the preview itself.
#2958	feature - Backup & Restore: portable run bundles — export and import a complete backup run	A backup run produces one archive per tenant plus a global archive. Backup & Restore can now export a **complete run bundle** — every archive of a run packaged as one portable artifact — and import such a bundle on a target deployment, restoring the run as a coherent set. Portable-archive workflows are hardened end to end (working-directory-independent compression, bounded conversion output, accurate restore error summaries), and the Backup & Restore actions gain a clearer, accessible layout. Whole-run bundle import runs under a bounded memory envelope, so importing a large run cannot pressure the search head.
#3014	feature - Notes: selective clearing of automated notes and bulk notes management	TrackMe's automated notes (adaptive-delay summary notes and threshold-lock reconcile notes) accumulate on entities over time, and the only cleanup was **Clear all notes** — which also deleted user-authored notes. The Notes modal now offers clearing automated notes selectively (all of them, or all except the most recent), and notes can be managed in bulk. User-authored notes are never affected by the automated-notes actions.

Table 36: Enhancements & Changes

Issue Number	Description	Details
#2959	change - Alerting: machine-readable marking of AI-generated email content (EU AI Act Art. 50(2))	When an alert email embeds AI-generated content (such as the AI status report), the message now carries a machine-readable marking: an `X-TrackMe-AI-Generated` MIME header, a `data-ai-generated` attribute on the HTML section containing the generated content, and a visible plain-text label mirrored in the HTML heading. Deterministic email content remains unmarked. This implements the EU AI Act Article 50(2) transparency requirement for synthetic content.
#2950	change - AI Assistant: per-response machine-readable AI-generated marker (EU AI Act Art. 50(2))	Every AI Assistant chat response now carries a machine-readable "artificially generated" marker, present both in the rendered interface and in the REST payload, extending the Article 50(2) marking convention from email content to interactive chat.
#2924	change - Backup & Restore: convert a zstd archive to a portable gzip archive	An administrator can now upload a zstd-compressed TrackMe backup archive, convert it to a gzip-compressed `.tgz`, and automatically download the converted file — enabling restores on deployments where the `zstd` binary is not available. A compression-mode selector accompanies the conversion workflow.
#2923	change - Dependency maintenance	Third-party dependencies were refreshed for the 2.4.13 cycle across four consolidated waves, including security-driven updates to `brace-expansion`, `dompurify`, `nanoid` and `js-yaml`. The release validation reported zero known audit vulnerabilities.
#2925	change - Tenant visibility is enforced on the remaining sibling read endpoints	The server-side tenant-visibility enforcement introduced in 2.4.9 for the main entity reads now also covers the sibling read endpoints: cached per-component summary counts and the shadow-cache probes (single, and bulk with per-scope handling). A caller outside a tenant's RBAC scope receives the same not-found response as for a nonexistent tenant, and internal system callers are unaffected. This completes the tenant-RBAC read-gate work across the component read surface.
#3007	change - FQM collect-job wizard: event-distribution timecharts in the benchmark step	The benchmark step of the FQM collect-job wizard renders event-distribution timecharts, showing how the events feeding the field-quality analysis distribute over time before the tracker is created.
#3009	change - FQM collect-job wizard: retrieved-events metric above the benchmark charts	The benchmark step reports the number of events retrieved against the configured limit above the distribution charts, so an undersized sample is visible at a glance.
#3003	change - Entities table: the number of visible labels per entity is a system-wide setting	The number of labels shown inline per entity in the entities table is now a system-wide setting (default 4), adjustable from the system settings.
#3006	change - Timestamps render in the caller's Splunk timezone	Entity-overview timestamps now render in the caller's own Splunk timezone, and backend-rendered timestamps are suffixed with the server timezone label — removing the ambiguity of unlabelled server-local times.
#2991	change - AI providers: refreshed public model suggestions and a new xhigh reasoning-effort level	The AI provider configuration refreshes its public model suggestions to the current generations (including Grok 4.6, Claude Opus 5, GPT-5.6 and Gemini 3.x) and adds an `xhigh` reasoning-effort level for providers that support it.
#3012	change - Email delivery: unified SMTP transport selection and a per-account SSL certificate verification toggle	All email send paths now select the SMTP transport through one shared decision, removing per-path behaviour differences, and each email delivery account gains an SSL certificate verification toggle for deployments using internal certificate authorities.

Table 37: Bug Fixes

Issue Number	Description	Details
#3016	bug - FQM: pie chart drilldowns honour the clicked slice	Clicking a slice of an FQM pie chart always drilled down on the first category regardless of which slice was clicked — on the global-entity chart and on the field-entity charts alike (#3018). Both drilldowns now filter on the clicked slice's category.
#2995	bug - Remote accounts: tokens rotated for accounts that had opted out of rotation	`token_rotation_enablement="0"` was parsed as truthy, so bearer tokens were rotated for remote accounts that had explicitly disabled rotation. Opt-out is now honoured.
#2989	bug - Topology Studio: auto-arrange no longer overlaps rows when KPI chips are shown	The auto-arrange layout now reserves vertical clearance for visible KPI chip stacks, so arranged rows no longer overlap on views showing per-node KPIs — most visibly on auto-created member nodes.
#3001	bug - FQM collect-job wizard: simulation failed with a generating search	Using a generating search type in the CIM collect-job wizard broke the simulation with an HTTP 500. Generating searches now simulate correctly.
#3011	bug - FQM: the data-dictionary Test regex search honours the trackers' summary index(es)	The data dictionary's Test regex search queried the default summary index regardless of the index(es) the tenant's trackers actually write to; it now targets the trackers' configured summary index(es).
#3005	bug - Labels: dangling automatic-rule label identifiers are contained and repairable	Automatic-label rules could retain references to label identifiers that no longer exist. Batch operations now filter dangling identifiers, and the UI surfaces repair chips to clean the affected rules.
#2994	bug - Remote accounts: configuration defaults aligned and backfilled	Three remote-account defaults had drifted from their declared configuration values and are realigned; the connectivity-check timeout fallback is aligned to 15 seconds; and `splunkcloud_member_autoheal_enablement` is backfilled onto existing remote accounts so the auto-heal behaves consistently for accounts created before the option existed.
#2982	bug - Flex Objects: status 3 (unknown) no longer escalates to red, and status messages report the tracker-produced status	A Flex Objects entity reporting status 3 — the Flex convention for an unknown or unexpected status — was escalated straight to red, and the entity's status message was overwritten by the derived state instead of reporting what the tracker actually produced. Status 3 now lands orange by default through a new configurable impact score (`impact_score_flx_status_unknown`, default 50 — set 100 to force red or 0 to ignore), and status messages carry the tracker-produced status.
#2985	bug - Labels: deleting a label still referenced by an enabled automatic rule is refused	Deleting a label definition that an enabled automatic-label rule still references silently broke the rule. The deletion is now refused with an explanatory message; disable or retarget the rule first, then delete the label.
#2935	bug - Labels: edits could drop existing assignments and automatic-label metadata	Label edits performed under concurrent write pressure could lose existing label assignments and automatic-label metadata. The write path is now serialised and budget-bounded so concurrent activity can no longer cause assignment loss.
#2929	bug - Topology Studio: the AI-describe context now applies per-tenant graph redaction	The graph context provided to the AI Assistant for Topology Studio conversations did not apply the per-tenant redaction enforced on every other read surface, so a restricted user's chat context could include tenant content hidden from them in the interface. The AI-describe context now applies the same tenant-visibility redaction as saved views, version previews and portable exports.
#2956	bug - Flex Objects: license usage templates no longer double-count rollover summaries	The Flex Objects license usage tracking templates counted rollover summary events alongside the primary usage events, double-counting usage. The templates now filter license usage events by type.

11.1.14 Version 2.4.12 - build 1785703548 (03/08/2026)

Hint

TrackMe 2.4.12 — Topology Studio opens to Foundation Edition, with critical stability hardening for large logical-group deployments.

- **Topology Studio is now open to Foundation Edition.** The feature was introduced in TrackMe 2.4.9 but was mistakenly restricted to Enterprise and Unlimited customers. Foundation customers can now use the complete authoring experience and manage Topology Alerts. Existing security boundaries and the normal licence read-only state are unchanged.
- **Critical memory-stability fix for large logical-group deployments.** In versions 2.4.6 through 2.4.11, logical-group processing could trigger a repeating cycle of overlapping background refreshes. At scale, this could put severe memory pressure on `splunkd` and cause an out-of-memory termination. TrackMe 2.4.12 prevents the cycle and limits duplicate refresh work while retaining automatic updates. Customers with large numbers of logical groups should treat this as a critical upgrade and prioritise deployment of 2.4.12.
- **No external machine-learning application dependency.** TrackMe no longer requires the Splunk Machine Learning Toolkit or the Splunk Scientific Python add-on because outlier detection runs on TrackMe's native engine. Outdated checks for those applications could report missing requirements and interfere with Virtual Tenant creation when they were absent; the checks are now removed.
- **Incremental outlier baselines.** Supported native KV Store models can retain representative historical observations within a configured bound across training cycles. Full retraining remains the default; incremental learning can be inherited from the system setting or selected per model, and switching back to full performs the established full refit.
- **Median and better guidance for outlier analysis.** Median is available as an outlier calculation method, making models less sensitive to occasional volume spikes than an average. The Outliers page now includes an accessible visual explanation of learning, detection, qualification and impact-based scoring, and its settings guidance reflects the shipped defaults.
- **More expressive and securely shared Topology Studio views.** Authors can select a saved link, inspect its source and target, edit its label, choose one-way, bidirectional or hidden arrows, reverse a one-way relationship and decide which relationships carry impact propagation. Bidirectional links carry impact in both directions immediately. The two-step connection workflow is also more prominent and easier to follow. Tenant visibility is enforced consistently when opening saved views, previewing their history and creating or importing portable archives.
- **More precise aggregates and propagation.** An aggregate can optionally be constrained to an exact subset of the entities matching its scope and filter. Generated members, live status, alerts and rendered emails all use the same membership. Red, orange and informational blue states propagate consistently, aggregate-member arrows remain visible, and member KPI choices reflect the generated entity set.
- **Complete policy and coverage results at scale.** Priority, tag and SLA policies based on SPL searches or lookups now evaluate the complete result set instead of stopping at 50,000 rows. Exceptionally large policy sources stop explicitly before partial results can be applied. Host and data-source coverage analysis and expected-entity injection now read the complete tracked inventory, expose duplicate reference rows separately and fail clearly if that complete inventory cannot be loaded.
- **Median volume KPIs for DSM and DHM.** The median five-minute event count is available in Performance Metrics, stateful-alert charts and ML Outliers KPI selection. Existing defaults are unchanged and history starts after upgrade; earlier values are not backfilled.
- **Cleaner fresh installations.** Hybrid tracker creation now tolerates the short configuration-refresh window that can follow creation of a search macro. A deployment with no AI provider or no Virtual Tenant configured also no longer records a misleading error for that normal state.
- **Dependency maintenance.** Third-party JavaScript dependencies were refreshed for the release cycle, with no known vulnerabilities reported by the release audit.

- SHA256: caf23ca3a0cc4b60a22f4cf51ff23451c6460f53ea32f62197f8606f4bfc4859

Table 38: New Features

Issue Number	Description	Details
#2895	feature - Outlier detection supports bounded incremental baseline learning	Supported native KV Store outlier models can now retain representative history across training cycles without allowing their stored baseline to grow without bound. Full retraining remains the factory default. Administrators can enable incremental learning globally, override it for selected models, see the configured and effective mode in the interface and AI-assisted workflows, and switch back to full training when recent data should replace the retained history. Unsupported model or storage combinations remain safely full-only.
#2897	feature - Topology Studio aggregates can target an exact entity subset	Aggregate scopes and filters remain dynamic by default, but authors can now optionally select the exact matching entities that should participate. The same resolved subset is used by the live canvas, previews, alerts, emailed topology images and automatically generated member nodes, so every surface represents the same topology. Candidate selection respects the user's tenant visibility.
#2898	feature - Interactive topology link authoring and impact controls	Saved topology links are now first-class authoring objects. Select a link to edit its label, inspect or reverse its stored source and target, choose one-way, bidirectional or hidden arrows, and decide whether it participates when a node propagates impact through selected links. Clearer source/target guidance and synchronised unsaved previews make the relationship easier to understand while editing.

Table 39: Enhancements & Changes

Issue Number	Description	Details
#2896	change - Median outlier calculation and an in-product guide to outlier analysis	Median joins the existing outlier calculation choices, providing a measure of typical activity that is less influenced by occasional spikes than an average. The Outliers page now explains the complete learning, detection, qualification and scoring journey with accessible visual guides, and the settings documentation and examples have been aligned with the defaults actually shipped by TrackMe.
#2902	change - DSM and DHM expose median five-minute event volume	Data Source Monitoring and Data Host Monitoring now calculate and display the median five-minute event count in Performance Metrics, stateful-alert charts and ML Outliers KPI selection. Existing KPI defaults remain unchanged, lookup-specific behavior is preserved and the new history begins after upgrade rather than being backfilled.
#2854	change - Dependency maintenance	Third-party JavaScript dependencies were refreshed for the 2.4.12 cycle. The release validation reported no known vulnerabilities in the JavaScript dependency audit.

Table 40: Bug Fixes

Issue Number	Description	Details
#2914	bug - Prevent memory exhaustion during large-scale logical-group processing	In versions 2.4.6 through 2.4.11, logical-group processing could repeatedly schedule overlapping background refreshes. Deployments with many logical groups could experience severe `splunkd` memory pressure and an out-of-memory termination. TrackMe now prevents the repeating refresh cycle and bounds duplicate refresh work, preserving automatic updates without uncontrolled concurrency. This is a critical upgrade for customers operating large numbers of logical groups.
#2901	bug - Topology Studio is now available in Foundation Edition	Topology Studio was introduced in 2.4.9 but was mistakenly restricted to Enterprise and Unlimited customers. Foundation customers can now create, edit, duplicate, import, customise and share topology views and fully manage Topology Alerts. Capability checks, tenant isolation, per-view access controls and the normal read-only state of an expired licence continue to apply. Tenant visibility is enforced consistently across saved views, version previews and portable exports.
#2899	bug - Expanded aggregate-member links retain visible direction	Automatically generated links now point visibly from an aggregate to each expanded member, and that direction remains when a generated member is pinned. A directionless automatic link previously caused arrowheads to disappear across the entire canvas. User-authored bidirectional and directionless links keep their chosen appearance.
#2900	bug - Topology impact propagation carries every non-green state consistently	Red, orange and informational blue can now propagate through selected topology relationships; only healthy green is excluded. More important red or orange impact can replace an aggregate's derived blue state, while a protected blue entity remains protected. The inspector preserves and explains the target's own status alongside the inherited status, and informational blue remains non-alerting.
#2915	bug - Bidirectional topology links propagate impact in both directions immediately	A bidirectional relationship now contributes both propagation paths as soon as it is selected. Either endpoint can repaint the other when its propagation mode and native status permit it; authors no longer need to reverse the link temporarily or save it as a one-way relationship first. One-way and hidden-arrow links retain their stored source-to-target behavior.
#2885	bug - Bidirectional topology links cannot be reversed accidentally	Reversing source and target is now disabled for a bidirectional link, where the visual relationship makes that action misleading even though it would change the stored dependency direction. The link inspector also presents distinct Source and Target cards with a direction-aware connector for one-way, bidirectional and hidden-arrow modes.
#2880	bug - Aggregate member KPI choices reflect the generated members	The member KPI picker now derives its choices from the entities that remain after the aggregate scope, filter, optional exact selection, ordering and expansion limit are applied. It no longer offers KPIs that none of the generated member nodes can provide, and stale choices are cleared while membership inputs change.
#2903	bug - Coverage analysis and expected-entity injection support inventories above 50,000 records	Host and data-source coverage analysis previously compared a complete reference inventory with only the first 50,000 tracked entities on very large deployments, and the related injection workflows inherited the same incomplete view. They now load the complete tracked collection, keep host asset matching consistent between analysis and injection, distinguish unique reference entities from duplicate rows and report an incomplete read instead of presenting incorrect counts.
#2877	bug - Search-based policies no longer stop after 50,000 results	Priority, tag and SLA policies driven by an SPL search now consume the complete search result in both simulation and live application. Exceptionally large sources stop explicitly before partial results can be applied. The interactive field-discovery preview remains intentionally limited for responsiveness.

11.1.15 Version 2.4.11 - build 1785321072 (29/07/2026)

Hint

TrackMe 2.4.11 — hotfix: AI Advisors fail to start on recent builds due to MCP 2.0.0 publication.

- This hotfix release restores the **AI Advisors**, which fail to start on affected builds. **Only the AI layer is impacted** — data monitoring, tracking, alerting, notable events and every other TrackMe capability continue to operate normally, and deployments that do not use the AI features see no functional difference.
- **Which builds are affected:** any TrackMe package built on or after **28/07/2026** — this includes the publicly released **2.4.10** build. Builds produced before that date are not affected. On an affected build, every advisor surface fails to start: the interactive AI Advisors, the Flex Objects tracker-generation wizard, the AI Assistant's advisor actions, and the scheduled advisor batches.
- **Cause:** a third-party library bundled with TrackMe (`mcp`, used by the Splunk Agent SDK that powers the advisors) published a new **major version** on 28/07/2026 which removed part of the API the SDK depends on. TrackMe's dependency declaration had no upper bound, so builds produced from that date automatically picked the new major version up. The dependency is now pinned to the supported 1.x line, and will only move in step with a Splunk SDK release that supports the new API.
- **Upgrading is enough — no manual cleanup is required.** Splunk application upgrades never delete files that the new version no longer ships. An installation that has run an affected build therefore keeps incompatible leftovers from that library, which would continue to break the advisors even after upgrading to a fixed build. TrackMe now detects and removes those leftovers automatically, logging what it removed. This matters in particular for Splunk Cloud, where operators cannot clean the application directory by hand.

- SHA256: 0727bf8201ce64ae4c2e85ce01f553b9f74d95b17481320f98aa625d75f61464

Table 41: Bug Fixes

Issue Number	Description	Details
#2849	bug - AI Advisors fail to start on builds produced from 28/07/2026 (including the public 2.4.10 build)	The `mcp` library bundled with TrackMe — a dependency of the Splunk Agent SDK that powers the AI Advisors — released a major version (2.0.0) on 28/07/2026 that removed API elements the SDK still uses. Because TrackMe's requirement carried no upper bound, every build produced from that date bundled the incompatible version, and all advisor surfaces failed at start-up with `AttributeError: 'Server' object has no attribute 'list_tools'`. The dependency is now pinned below the breaking major version. A second-order effect is also handled: because Splunk upgrades extract over the existing installation and never remove files dropped by a newer version, an installation that ran an affected build retains incompatible leftovers that shadow the corrected ones — producing `ImportError: cannot import name 'TASK_STATUS_COMPLETED' from 'mcp.types'` even after upgrading. TrackMe now verifies the integrity of that bundled library at runtime and removes such leftovers automatically, so upgrading to this release is sufficient and no manual intervention is needed. Only the AI Advisor surfaces were affected; all other TrackMe functionality was unaffected.

11.1.16 Version 2.4.10 - build 1785285034 (29/07/2026)

Hint

TrackMe 2.4.10 — hotfix: chart images in alert emails on Splunk Cloud 10.x.

- This hotfix release addresses a chart-rendering problem affecting **Splunk Cloud 10.x only** — on-premise deployments and Splunk Cloud 9.x are unaffected.
- On Splunk Cloud 10.x, chart images embedded in alert emails were delivered as **SVG instead of PNG**; because most email clients refuse to render inline SVG, recipients effectively received their alert emails **without chart images**. The alert text, links and all other content were unaffected — alerts continued to fire and be delivered on time.
- For **stateful alerts**, this is a **regression** introduced by the Splunk Cloud 10.x platform environment (Python 3.13): chart images worked on Splunk Cloud 9.x and stopped working on 10.x. For **Topology Alerts**, the embedded topology image was affected as a **known issue in the initial release** of the feature (Topology Studio shipped in 2.4.9, so there is no prior behaviour to regress from).
- The PNG rendering backend is now fully self-contained (it no longer depends on system libraries that minimal Splunk Cloud hosts do not provide) and is resolved deterministically, so chart images render as PNG again on every supported platform.
- The release also carries the cycle's dependency refresh, with a **clean security audit**.

- SHA256: baf84a10d3a5f9ddd5211a3e2f4e7df98aa4f589e65f5be9e3847cb0d496cc88

Table 42: Enhancements & Changes

Issue Number	Description	Details
#2836	change - Dependency maintenance	Third-party dependencies were refreshed for the 2.4.10 cycle in a consolidated wave, resolving all outstanding security advisories — the JavaScript dependency audit reports **zero vulnerabilities** at release. All updates are lockfile-level version bumps with no functional impact.

Table 43: Bug Fixes

Issue Number	Description	Details
#2837	bug - Splunk Cloud 10.x: chart images in alert emails delivered as SVG instead of PNG	**Affects Splunk Cloud 10.x only** — on-premise and Splunk Cloud 9.x are unaffected. Chart images embedded in stateful alert emails (a **regression** relative to Splunk Cloud 9.x, introduced by the 10.x platform environment) and the topology image embedded in Topology Alert emails (a **known issue in the initial 2.4.9 release** of the feature) silently fell back from PNG to SVG, which most email clients do not render — so the emails arrived without their chart images. Impact was cosmetic: alerts fired and were delivered normally, and charts remained viewable in the TrackMe UI via the links in each email. Affected environments show `task=charts, PNG conversion dependencies missing, falling back to SVG` in the alert action log. The PNG rendering backend is now built fully self-contained for the Splunk Cloud 10.x Python runtime (no dependency on system libraries that minimal Cloud hosts do not provide), the rendering library is pinned to that bundled backend, and a rendering failure is now reported with its real underlying cause in the logs instead of a generic message.

11.1.17 Version 2.4.9 - build 1785244560 (28/07/2026)

Hint

TrackMe 2.4.9 — Topology Studio: the map of your environment becomes yours — live, alerting, AI-powered.

- **Topology Studio — a major new product feature, the headline of this release.** Author your own topology maps from TrackMe entities: a full canvas editor (entities from any tenant and component, live aggregate nodes with filters and auto-expanding members, labels, edges, icons, background images, KPI cards, top-down auto-arrange), shared under per-view RBAC, watched live, consumable full-screen on a wall of screens, deep-linkable, exportable as PNG and as portable archives.
- **Topology Alerts — smart alerting and correlation on your maps.** A state-aware alerting engine (open / update / resolve) correlates everything a topology covers: notable events on every transition, and rich email delivery embedding the **server-rendered topology image** — same icons as the canvas — with a drilldown link back to the live view (customisable root URL for reverse proxies).
- **AI-integrated end to end.** The AI Assistant understands your topologies and alerts in context — and can **generate or modify topologies for you** through consented, audited actions. Entity KPIs cover every component, including **dynamic per-entity metrics for Flex Objects, FQM and WorkLoad**.
- **Enterprise-grade security.** Per-view read/write RBAC with creator guarantees, and **server-side per-node tenant isolation** on the live-status surface — a node outside your tenant permissions reveals nothing, and the isolation is enforced by mutation-verified CI guards.
- **Alert Enrichment (new).** Enrich your alerts with your own data: a configurable SPL enrichment whose fields are merged into stateful alerts and notable events, so every notification carries the context that matters to you (CMDB attributes, ownership, anything your SPL returns).
- **The built-in Topology view gets a full enhancement arc.** Admin-configurable default and maximum node counts, a resizable and maximizable graph modal that remembers its size, an **Availability vs Health** display mode, click-a-node entity details (including converging members), and a live summary of the current selection.
- **A more visual TrackMe.** Tracker wizard simulation results, alerting entity overviews and the Tenant Home header now use visual cards, stat strips and actionable tiles instead of dense tables — plus a visual diagram of the DSM/DHM break-by logic.
- **Faster and safer at scale.** The Support Diagnostic is rearchitected to run in a search process, the Virtual Tenants drill-down modals load slim and render windowed, and tenant read endpoints are now gated server-side on the caller's visible tenant set.
- **Also in this release:** DSM lookup-monitoring refinements, Configuration Guardian fixes, first-class AI agent step budgets, the FLX tracker-name length bound, tolerant non-UTF-8 reference-lookup reads, two security dependency bumps, and a dependency refresh including a Splunk UI toolkit upgrade.
- See the new *Topology Studio* documentation chapter.

- SHA256: 40d213f715a4193946299c6df5241231210e3e78aca4d5342ce4af2dcda77f80

Table 44: New Features

Issue Number	Description	Details
#2733	feature - Topology Studio: user-authored live topology views + Topology Alerts	**A major new product feature — the key announcement of TrackMe 2.4.9.** Topology Studio lets you author your own topology maps of the environment from real TrackMe entities — place, connect, share, and watch live. The canvas editor supports entity nodes from any tenant and any of the six components (picked through a cross-tenant searchable picker), live **aggregate nodes** (multi-tenant scopes plus the Virtual Groups filter grammar, folded to worst state with per-state counts, and optionally **auto-populated** — expanding their current members as linked nodes), and free-text labels; visuals include 74 selectable node icons plus a **deployment-wide catalog of user-uploaded custom icons** (any image, normalized automatically), custom background images, **stacked per-node KPI displays** (up to five per node, each renamable — aggregates apply a default stack to every expanded member), and a hierarchical top-down auto-arrange. A **canvas keyword search** finds any node by label, entity, tenant, component or aggregate scope — matches light up, Enter cycles and pans to each. Views are shared under **per-view RBAC** with a read/write split, consumed live (honest refresh countdown, rich node inspector with per-metric sparklines and one-click Entity Overview), full-screen for wall-of-screens displays, deep-linkable, exportable to PNG and as **portable archives** (graph, background and alert definitions travel together). **Topology Alerts** turn any view into an alerting object: a state-aware engine (open / update / resolve — no notification spam) with per-alert scheduling, explicit or automatic node selection, **notable events on every transition**, and opt-in email delivery embedding the server-rendered topology image — the same node icons as the canvas — with a drilldown link back to the live view. Entity KPIs cover **every component**: fixed catalogs for DSM/DHM/MHM/WLK and **dynamic per-entity metrics** for Flex Objects, FQM and WorkLoad (hybrid). The rich node inspector's aggregate **mini-map is interactive** — adjustable member count, click-to-preview, and one-click jump to an expanded member's node. The **AI Assistant** is integrated end to end — a dedicated topology chat context, per-alert AI, and the ability to **generate or modify topologies through consented, audited actions**. Security is enterprise-grade by design: server-side per-node **tenant isolation** on the live-status surface, privilege-safe alert evaluation, and an independent security review of the isolation gate before release.
#2674	feature - Alert Enrichment: merge your own contextual data into alerts and notables	A new **Alert Enrichment** capability lets you attach your own data to TrackMe alerting. Define an SPL enrichment returning the fields you care about — CMDB attributes, service ownership, escalation contacts, anything searchable — and TrackMe merges those fields into stateful alerts and notable events. The enrichment is configured per tenant, applies across every delivery target, and makes each notification self-contained: the responder gets the context with the alert instead of having to look it up.

Table 45: Enhancements & Changes

Issue Number	Description	Details
#2627	change - DSM lookups: tracker wizard picker scoped to the selected namespaces	The lookups picker in the tracker wizard now lists only the lookups belonging to the application namespace(s) selected earlier in the wizard, instead of every lookup on the deployment — making the right lookup easier to find and avoiding accidental cross-namespace selection.
#2624	change - Support Diagnostic: rearchitected and resilient at scale	The global Support Diagnostic is rearchitected to run its collection **in a search process rather than a REST handler**, which removes the whole class of stalling and timeout problems seen on large environments. Supporting searches run as asynchronous jobs under a true global deadline, searches abandoned by a timeout are cancelled instead of being left running, and the job of a completed run is retained for inspection.
#2747	change - Support Diagnostic: tenant anonymisation is now complete and fails safe	When a diagnostic archive is generated with **tenant anonymisation** enabled, tenant identifiers are now replaced in the archive manifest (`metadata.json`) as well as in the collected files — previously the manifest still listed the real tenant identifiers and referenced the pre-anonymisation file names. **Behaviour change to be aware of:** anonymisation now *fails safe*. If it cannot be completed — in full, across every file — the archive is **not** produced and the run reports an error explaining why, instead of publishing an archive that would still contain real identifiers. If the identifiers are not sensitive in your environment, re-run with anonymisation disabled. In addition, a completed run whose result cannot be written to the KV Store no longer appears to hang: it now reports the failure immediately, and the generated archive is recorded in `trackme_support_diag.log` so it can still be retrieved.
#2820	change - Backup & Restore: rearchitected — backup and restore jobs run in a search process	UI-triggered backup and restore jobs now execute **inside a splunkd search process** (via a new generating command dispatched server-side) instead of background threads in the Backup & Restore REST-handler process — the same rearchitecture applied to the Support Diagnostic in this release, and for the same reason: splunkd recycles handler processes at will, which could silently kill a long backup or restore mid-flight and, in the worst case, turn a very large restore into an endless retry loop that never completed. The job is claimed exclusively (a duplicate dispatch can never double-run), the UI now shows **live progress** — a per-archive step feed for backups and a per-task feed for restores, with durations and sizes, persisted after completion — and a watchdog transitions jobs that can never finish to an explicit failed state instead of leaving them appearing to run forever.
#2640	change - Topology view (built-in): configurable size, resizable modal, new display mode and node details	The built-in per-component topology view gains a full round of enhancements: administrators can configure the default and maximum number of entity nodes; the graph opens in a resizable and maximizable modal that remembers its size across opens; a new **Availability vs Health** display mode switches what the node colours represent; clicking a node opens its details (including for converging members, from where the entity overview can be opened directly); and the current selection is summarised live without needing a click.
#2668	change - A more visual TrackMe: cards, stat strips and actionable tiles	Several dense tabular surfaces are replaced by visual, scannable presentations. Tracker wizard **simulation results** (Feeds hybrid, Flex Objects, FQM and the tenant wizard) now render as topology-style cards with conditional colours and a summary strip, so what a tracker will actually create is obvious before you create it. The **alerting entities overview** modals gain entity cards and a summary strip (#2669), the **tenant entities overview** gains actionable header stat tiles that filter the table when clicked, and the **Tenant Home header** gains a compact

Table 46: Bug Fixes

Issue Number	Description	Details
#2821	bug - Backup & Restore: ephemeral job, lock and cache collections are no longer captured in the global archive	The global backup archive included ephemeral job-tracking, lock and cache KV collections. Restoring such an archive into another environment imported the source's job rows — including stranded "running" entries the automatic purge never removes — plus lock rows that blocked the next acquisition until their staleness timeout, and cache-validity tokens referencing files that do not exist on the target. These collections are now excluded from the global archive, and bulk-restore scope lists can no longer force-restore them.
#2817	bug - Backup & Restore: importing a gzip archive no longer requires the zstd binary	Importing a valid gzip `.tgz` backup archive through the Backup & Restore UI failed with `HTTP 400 "Invalid or unsupported archive format"` on search heads without the `zstd` binary — exactly the hosts that produce gzip archives in the first place, making standalone-to-standalone migrations fail. Archive detection now recognises the format from the file content itself, and a genuinely zstd-compressed archive imported on a host without `zstd` is rejected with an actionable message naming the missing binary instead of a generic error.
#2770	bug - Scoring: repeat manual-influence and false-positive actions now reflect immediately	Score actions (manual influence / false positive) taken repeatedly on the same entity did not reflect in the UI after the first one: the score-cache merge deduplicated per score source instead of per entry, so the newest action was silently dropped in favour of the older indexed record. The merge now deduplicates per entry by recency, and repeated actions refresh immediately.
#2776	bug - Maintenance KDB: recurring KeyError in the expiration sweep	The maintenance knowledge base expiration sweep raised a recurring `KeyError: 'time_expiration'` every 5 minutes on records missing that field. The sweep now handles such records gracefully.
#2799	bug - Robustness: repaired latent errors on rarely-taken code paths	A systematic static sweep identified latent `NameError` defects on rarely-exercised error-handling paths across the backend; all fixable occurrences were repaired, hardening behaviour on unusual error conditions.
#2618	bug - FLX: long tracker names overflowed Splunk's 100-character saved-search limit	Creating a Flex Object tracker whose derived saved-search name exceeded Splunk's 100-character limit failed with a gateway timeout instead of a clear message. The derived tracker name is now bounded to fit the limit, and a name that genuinely cannot fit fails fast with an explicit error.
#2622	bug - Configuration Guardian: member-drift false positive on multi-search-head-tier Splunk Cloud stacks	The remote-account member-drift check could raise a false positive on Splunk Cloud stacks that run multiple search-head tiers under a single domain. Member drift is now assessed by cluster membership only, eliminating the false alert.
#2629	bug - DSM lookups: data sampling no longer offered or applied to lookup entities	Data sampling is not meaningful for lookup entities. The Data sampling tab is now hidden for lookups, and the sampling executor skips lookup entities entirely.
#2625	bug - Configuration Guardian: run buttons now show a busy state	The per-check "Run this check" / "Run now" buttons gave no feedback while a check was running. They now show a busy "Running…" state until the check completes.
#2638	bug - Coverage gap analysis failed with HTTP 500 on reference lookups containing non-UTF-8 bytes	Coverage gap analysis (and expected-entity injection) returned an HTTP 500 when the reference lookup it read contained a non-UTF-8 byte. The reader now tolerates such bytes and completes the analysis instead of failing.
#2671	bug - Virtual Tenants drill-down modals were sluggish and could exhaust the browser at scale	Opening a drill-down modal from the Virtual Tenants vignettes rendered every record eagerly with no progress indication, which was slow and could kill the browser tab on very large tenants. These modals now load a slim payload, render progressively in windows, and show a progress bar while loading.
#2737	bug - AI Routines: next-run time could be computed in the wrong timezone	The next-run enrichment shown for AI Routines did not honour the UTC token and could present a next-run time already in the past. The computation now honours UTC and guards against past values.

11.1.18 Version 2.4.8 - build 1784017070 (14/07/2026)

Hint

TrackMe 2.4.8 — configuration management, reinvented: the Splunk UCC configuration page is replaced by native TrackMe UIs, and lookups become first-class citizens of DSM.

- **A brand-new configuration experience.** Every configuration surface that used to live on the Splunk-generated (UCC) setup page is now a native TrackMe interface: **Email delivery accounts**, **Remote deployment accounts**, **S3 export accounts**, and a completely redesigned **System settings** page — docs-style sidebar navigation, built-in documentation for every one of the **122 settings**, connectivity tests at your fingertips, insufficient-permissions detection, and the AI Assistant available in context. Faster, clearer, and consistent with the rest of TrackMe.
- **DSM lookup monitoring, first-class.** Alert when a lookup runs **empty** (configurable minimum record count), see the last-known record count as a KPI tile, and enjoy a fully normalised experience for lookup entities — status messages, charts, overview tabs and alert emails now speak "records and update frequency" instead of feed metrics. Creating lookup trackers is easier too, with dynamic pickers for application namespaces and lookups.
- **Alerting UX.** The alert wizard gains theme-aware, zoomable diagrams of the stateful lifecycle and the legacy alerting architecture, a recipients multi-select with address validation, and resizable full-width command editors.
- **Built for very large scale.** The Virtual Tenants overview and Virtual Group views now load slim entity payloads and fetch details on demand — no more browser-tab exhaustion on very large environments — and a reliability fix ensures background summary/KV updates in search commands can no longer be silently dropped.
- **Remote accounts.** Rotate a deployment's bearer token on demand with a per-row **Rotate token now** action.
- **Flex Objects — richer insight.** A new rich **Last-metrics charts modal** replaces the raw-JSON inspector, converging trackers gain a wider tracked-entities view with colour-coded availability tiles, and the tracker simulation now lists the covered sub-objects — so you can see exactly what a tracker covers before you create it.
- **And more.** Configurable tstats time span for Elastic trackers, explicit Unlock action for DSM/DHM threshold locks, resizable data-preview tables, a licensing safety fix so a transient KV read error can never downgrade a valid license to a Foundation trial, and a batch of fixes.

- SHA256: e880b5b4adae5cab780032cd3bd839e06619ff7911c43d29bb5ddb1e0d3ad8ed

Table 47: New Features

Issue Number	Description	Details
#2538	feature - Configuration: native TrackMe management UIs replace the Splunk UCC configuration page	The Splunk-generated (UCC) configuration page is retired and replaced by native TrackMe React interfaces, accessible from the Configuration menu: **Manage Email delivery accounts**, **Manage Remote deployment accounts** and **Manage S3 export accounts** provide modern account management with inline connectivity tests, clear validation and secure credential handling; a new **System settings** page groups all **122 system settings** behind a docs-style sidebar navigation, with per-setting documentation modals, drift-proof schema generation, insufficient-permissions detection, and the AI Assistant integrated in context. Every reference and link across the application now points to the native interfaces.
#2498	feature - DSM: first-class lookup monitoring, including empty-lookup alerting	Lookups monitored as DSM entities gain a fully normalised experience. TrackMe can now alert when a lookup has **zero records** — or fewer than a configurable **minimum record count** (per-entity setting, with its own impact score weight) — closing the gap where an emptied lookup went unnoticed. The entity overview surfaces the last-known record count as a dedicated KPI tile, status messages and the delay-management UI are reframed for lookup semantics (staleness rather than feed delay/latency), entity charts and the "Overview data source" tab show records and update frequency, and stateful alert emails embed lookup-appropriate charts. The lookups hybrid-tracker wizard adds dynamic pickers that discover application namespaces and lookups and auto-translate the selection into tracker parameters.
#2570	feature - Remote accounts: on-demand token rotation	The Manage Remote deployment accounts interface gains a per-row **Rotate token now** action (with confirmation) to force the rotation of a deployment's bearer token on demand, and newly created accounts can be force-rotated in a single call. The rotation flow is fail-safe by construction: the new token is generated and validated against the remote deployment before it is stored, and the previous token is only revoked last.

Table 48: Enhancements & Changes

Issue Number	Description	Details
#2531	change - Elastic trackers: configurable tstats time span	Elastic sources (shared and dedicated) running in tstats mode can now configure the search time span, allowing the tracking behaviour to be tuned to the data profile instead of relying on a fixed window.
#2548	change - Alerting: theme-aware, zoomable architecture diagrams	The static alerting diagrams are replaced by theme-aware, zoomable components: the alert creation wizard now embeds an interactive diagram of the stateful alerting lifecycle, and the legacy alerting architecture is illustrated with a matching zoomable SVG diagram (#2551). Both render crisply in dark and light themes and support click-to-zoom inspection.
#2544	change - Stateful alerting: recipients multi-select and resizable command editors	The alert modal replaces the free-text recipients field with a multi-select accepting one address per entry (with format validation), and the command editors become full-width and vertically resizable for comfortable editing of longer searches.
#2596	change - Alerting: email delivery account pre-selected in the alert wizard	When creating a stateful alert, the wizard now pre-selects the first configured email delivery account instead of leaving the field empty, so email alerting works out of the box with one less step.
#2598	change - Configuration Guardian: configurable environment name on email notifications	Guardian email notifications can now carry a configurable environment name, making it easy to tell at a glance which deployment a Guardian alert email came from.
#2600	change - Configuration Guardian: email delivery account pre-selected in the notifications setup	The Guardian notifications configuration now pre-selects the first configured email delivery account, consistent with the alert wizard.
#2572	change - UI: auto-fit and resizable columns in data preview tables	Data preview tables (lookup previews, search previews, gap-analysis wizards, CMDB integration) now auto-fit their column widths to the content and support manual column resizing — long column headers and values are no longer truncated into unreadable columns.
#2575	change - FLX converging trackers: richer tracked-entities modal and simulation	The converging tracker tracked-entities modal gains a wider layout with colour-coded availability KPI tiles, and the tracker simulation now exposes the list of covered sub-objects (#2580) so you can verify the tracker's coverage before creating it.
#2592	change - FLX: rich "Last metrics" charts modal	The Flex Object "Last metrics" inspector is replaced by a rich charts modal — the latest tracked metrics are presented as readable visualisations instead of a raw-JSON dump, making it far easier to interpret a Flex Object's current state at a glance.
#2579	change - DSM/DHM: explicit Unlock action for threshold locks	Entities with locked delay/latency thresholds gain an explicit **Unlock** action in the UI (and matching AI Assistant awareness), making it straightforward to return an entity to automatic threshold management.
#2556	change - Virtual Tenants: scoped tenant pre-selected in Clear operational status	Launching "Clear operational status" from a tenant's Operational Status view now pre-selects that tenant instead of defaulting to all tenants, avoiding accidental cross-tenant clears.
#2568	change - Virtual Groups: entity filters reject structured payload fields	The Virtual Group entity filter now rejects structured payload fields (status/SLA JSON messages, dynamic thresholds and similar) which are not meaningful filtering dimensions: the wizard warns as you type and blocks submission, and the API enforces the same rule server-side.
#2534	change - Licensing: legacy free-extended tier decommissioned	The legacy free-extended licensing tier is decommissioned, and NFR (not-for-resale) licenses are now surfaced explicitly in the license information.
#2496	change - Dependency maintenance	Third-party dependencies were refreshed in consolidated maintenance batches as part of the 2.4.8 cycle, and the unused webpack-dev-server development dependency (and its orphaned demo scaffolding) was removed from the UI build toolchain.

Table 49: Bug Fixes

Issue Number	Description	Details
#2517	bug - Virtual Tenants overview could exhaust the browser at very large scale	Opening the Virtual Tenants entities overview (or a Virtual Group) on very large environments loaded full entity records for every row and could kill the browser tab. These views now load slim payloads containing only what the tables render, and fetch the complete record on demand (row expansion, entity overview) — dramatically reducing memory usage with no loss of detail.
#2542	bug - Search commands: late background updates could be silently dropped	Background threads started late in the execution of several TrackMe search commands (summary registrations, KV updates) could be silently killed when the search process terminated, occasionally dropping updates. These operations are now performed synchronously by design, and a permanent safeguard prevents the pattern from being reintroduced.
#2582	bug - FLX: concurrent hybrid trackers could overwrite each other's entity fields	When multiple FLX hybrid trackers updated the same entity concurrently, tracker executions could overwrite entity fields instead of merging them. Field updates are now merged correctly.
#2546	bug - DHM: missing "Status reason" card on entities with quoted status messages	DHM entities whose status message contained quoted content (for example ML outlier messages) showed an empty "Status reason" card — an escaping issue corrupted the message before display. The status reason now renders correctly, self-healing on the next entity view with no migration required.
#2535	bug - AI Assistant: light mode honoured and close button always visible	The AI Assistant panel could render in dark theme while the host page was in light mode, and the panel's close button could become invisible. The panel now follows the application's resolved theme on every page, and the header controls are theme-aware and always visible.
#2550	bug - Hybrid tracker wizards: the tracker name input no longer fights the user	Editing a tracker name in the creation wizards could trigger unwanted resets and repeated random-suffix injections while typing. The unique suffix is now applied once at creation only, and the name input behaves as expected.
#2540	bug - Elastic sources: shared executor refreshes the UI cache after updates	The shared Elastic sources executor did not refresh the UI cache after updating entities, leaving tables stale until the next full refresh cycle. The cache is now refreshed at the end of each shared Elastic cycle.
#2528	bug - Configuration: default values aligned with shipped configuration	Several settings declared default values that did not match the shipped configuration defaults, which could surface misleading values in the configuration interfaces. All defaults are now aligned and verified by an automated consistency check.
#2593	bug - Licensing: a transient KV read error can no longer downgrade a valid license	License validation now distinguishes a genuinely empty license store from a transient KV read error: a momentary KV-store blip could previously be read as "no license" and cause a valid subscription to be replaced with a Foundation trial. Errored reads are now treated as "state temporarily unavailable" and never mint a trial.
#2585	bug - FLX: splk_queues_filling use case set the entity alias to the host instead of the queue	The `splk_queues_filling` Flex Object use case set each tracked entity's alias to the host name rather than the queue name. The alias is now the queue name, so entities are labelled meaningfully.
#2589	bug - FLX: library trackers drop spurious prefixes and emit true description arrays	Corrects two Flex Object library issues where tracker names carried a spurious prefix and entity descriptions were emitted as a single joined string instead of a true array, improving how library-based Flex Objects are named and described.

11.1.19 Version 2.4.7 - build 1783461509 (08/07/2026)

Hint

TrackMe 2.4.7 — AI Routines: your standing instructions to TrackMe, plus a full Configuration Guardian management experience.

- **AI Routines — describe a monitoring task in plain English, and TrackMe watches for it.** Write a sentence like *"tell me if 4 or more high-priority feeds turn red within 15 minutes, correlate what they have in common, and add a note to each entity" — or "every morning at 08:00, send me a plain-English digest of the tenant's health: what is red, what changed since yesterday, and what deserves attention"* — TrackMe compiles it into a structured instruction, runs it on the schedule you choose, and delivers the outcome as an email or a Splunk event, with **real, audited actions** if (and only if) you granted them. Three evaluation strategies put you in control of AI cost: **Agentic** (the LLM reasons on every cycle), **Hybrid** (a cheap SPL trigger gates the LLM — zero tokens when nothing is happening), and **Deterministic** (SPL computes the condition, the LLM formats the outcome). Start from a template or a blank intent, preview the exact workflow before anything is created, and dry-run it with **Evaluate**. Opt-in by construction, with a kill switch, a creator-roles policy, consent pinned to the exact permissions you approved, per-routine AI provider and timeout, full token-cost visibility and a dedicated activity audit dashboard. See *AI Routines*.
- **Configuration Guardian — now a full management experience.** A new Check Catalog lets administrators enable/disable and tune every check, snooze alerts, and review severities with clear escalation ladders; a new degraded-tenant check detects tenants whose operations are impaired; and **email notifications** can be configured on alert lifecycle transitions — so Guardian findings reach you even outside the UI. An Ask-AI panel brings the AI Assistant directly to the Guardian page.
- **AI Advisors — stability.** Advisors now start reliably on OpenAI providers (strict-mode tool schemas), and search-hosted interactive execution gains automatic resume of orphaned runs.
- **Faster and more resilient at scale.** The Virtual Tenants page now loads its cached data in a single batched request (dramatically smaller responses, no more intermittent errors under heavy concurrent search load), and the global Support Diagnostic gains checkpoint/resume resilience with live step-by-step progress — it no longer gets stuck silently on large environments.
- **Fixes & polish.** The AI Assistant now knows bulk actions accept wildcards, refreshed Google Gemini model suggestions, quieter logs on deployments without remote accounts, and a batch of minor hardening fixes from the previous cycle.

- SHA256: 9c869f73ca158a1c8eb3fe14dad4dbce97b4696cdddb54893b8653e1167cea2c

Table 50: New Features

Issue Number	Description	Details
#2460	feature - AI Routines: scheduled natural-language AI tasks that watch, notify and (with consent) act	Describe a monitoring task in plain English and TrackMe turns it into a **standing instruction**: compiled by the AI into a structured, reviewable form, executed on your cron schedule, and delivered as an email or a Splunk event — with real, audited actions when you have explicitly granted them. Each routine selects one of three evaluation strategies (**Agentic**, **Hybrid**, **Deterministic**) to control AI cost directly; Hybrid routines cost zero LLM tokens when nothing is happening. Includes a starter-template gallery, a workflow diagram previewing exactly how the routine will run, an **Evaluate** dry-run with iterative refinement, per-routine AI provider and execution timeout, write permissions scoped by capability family with consent pinned to what you approved, run statistics and per-routine token-cost visibility, and a dedicated activity audit dashboard. Opt-in by construction (inert until the AI Assistant is enabled and a provider is configured) with a system kill switch and a creator-roles policy. Requires Python 3.13 (Splunk 10.2.x+). See *AI Routines*.
#2459	feature - Configuration Guardian: management UI, snooze and per-check tuning	The Configuration Guardian gains a full management surface under the Configuration menu: a searchable **Check Catalog** where administrators can enable or disable each check, tune per-check thresholds, and snooze alerts; a reviewed severity model with critical tiers and visible escalation ladders; and an Ask-AI panel to reason about Guardian findings with the AI Assistant directly from the page.
#2467	feature - Configuration Guardian: degraded-tenant check and email notifications	A new native Guardian check detects tenants whose operations are degraded, and administrators can now configure **email notifications** on Guardian alert lifecycle transitions — so critical configuration findings reach the right people even when nobody is looking at the UI.

Table 51: Enhancements & Changes

Issue Number	Description	Details
#2461	change - AI Advisors: search-hosted execution and orphan auto-resume	Interactive AI Advisor runs now execute search-hosted for reliability, and runs orphaned by an interruption are automatically resumed instead of being lost.
#2464	change - Support Diagnostic: checkpoint/resume resilience and live progress	The global Support Diagnostic is rebuilt around a checkpoint/resume worker with automated retries and a live step-by-step progress view. On large environments it no longer gets stuck silently — progress is always visible, and failures are surfaced instead of hanging.
#2439	change - AI Providers: refreshed Google Gemini model suggestions	The Google Gemini provider suggestions and help text are refreshed for the Gemini 3.x family, including the `gemini-pro-latest` alias.
#2416	change - Dependency maintenance and security updates	Third-party dependencies were refreshed and two security advisories addressed as part of the 2.4.7 cycle. **Routine updates:** `prettier` 3.8.4 → 3.9.4, `@types/node` 26.0.1 → 26.1.0, `@typescript-eslint` 8.62.0 → 8.62.1, `webpack` 5.108.0 → 5.108.3, `date-holidays` 3.30.2 → 3.32.0 (all build/development dependencies). **Security fixes:** `jsonata` deduplicated to 2.2.1 — High severity, CVE-2026-52746 (a transitive dependency of a bundled Splunk UI library); `js-yaml` (3.x) lifted to 3.15.0 — Moderate severity, CVE-2026-53550 (a test-time-only transitive dependency). No functional change to TrackMe behaviour.

Table 52: Bug Fixes

Issue Number	Description	Details
#2462	bug - AI Advisors failed to start on OpenAI providers	On OpenAI providers enforcing strict tool schemas, the AI Advisor agent could fail before starting. Tool schemas are now strict-mode safe, so advisors run reliably on OpenAI.
#2436	bug - Virtual Tenants page could fail intermittently under heavy search load	When many concurrent searches saturated the environment, loading the Virtual Tenants page could intermittently fail with server errors. The page now retrieves its cached data in a single batched request per page — dramatically smaller responses and no sensitivity to concurrent-search saturation.
#2451	bug - Global Support Diagnostic could fail silently	The global diagnostic could fail without surfacing any error, log or visibility of what happened. Failures are now surfaced with clear status, and the new checkpoint/resume worker prevents silent stalls.
#2465	bug - AI Assistant refused wildcard bulk actions	The AI Assistant chat was not aware that bulk actions accept `*` / `?` wildcards in the entity list, and refused whole-group requests. It now uses wildcards where appropriate.
#2406	bug - Noisy warning on deployments without remote accounts	Deployments with no remote accounts configured logged a repeated warning about the remote-accounts configuration on every call. This normal state is no longer reported as a warning.
#2398	bug - Batch of minor hardening fixes from the 2.4.6 review cycle	Eight small fixes deferred from the previous release's final review, including: the Workload overview "# Errors" card no longer shows an alert border when there are zero errors, the Remote Accounts force-rotation flow refreshes the rotation-logs view and restores keyboard focus, more consistent replica and threshold refresh behaviour, and assorted input-validation and display-fallback corrections (#2399, #2400, #2401, #2402, #2403, #2404, #2405).

11.1.20 Version 2.4.6 - build 1782921685 (01/07/2026)

Hint

TrackMe 2.4.6 — Workload execution-error visibility, replica reliability, and estate-wide bulk actions.

- **Workload — see why a scheduled search failed.** TrackMe now automatically captures and stores the actual execution-error messages of monitored scheduled searches (not just counts, and with no AI required), surfaces them in a dedicated Execution-errors view on the entity, and includes them in stateful-alert notifications — so an operator sees the root-cause error directly in the alert.
- **Workload — a redesigned Versioning experience.** The Versioning metadata tab is rebuilt as a structured, filterable version history: each tracked change is shown as an expandable card with *who* changed it, *when*, and *what* changed, and a colour-coded before/after diff for every affected property — the search itself, but also non-search settings such as the schedule and time range. Summary cards (total versions, versions in range, latest version), a time-range selector, a free-text filter, and an *"inspected … ago"* freshness indicator make it easy to review exactly how a scheduled search evolved over time.
- **Full control over your Virtual Tenants landing page.** TrackMe users can now organise the Virtual Tenants overview exactly the way they work — drag tenant cards, Virtual Group categories, and the groups within each category into any order (saved globally for all users), and the Administration menu is grouped into clear labelled sections. Together with the role-scoped Saved Views, this rounds out a rich cycle of Virtual Tenants enhancements that let each team land on a page tailored to their needs.
- **Replica trackers & tenants — a major reliability pass.** Replicated entities are no longer re-evaluated locally (which could corrupt their state and cause them to disagree with the source); they now mirror the source state, their table display fields and status are restored, Tenant Home counters refresh immediately

after a replica run, and the Manage/Execute replica screens are repaired with clear replica-context messaging.

- **Target whole families of entities in one action.** Bulk operations across the REST API (including acknowledgements, logical groups and ML outliers) now accept `*` / `?` wildcards, so you can act on an entire index, sourcetype family, or naming pattern in a single call instead of listing every entity.
- **More helpful alerts & AI.** Stateful-alert notifications now embed Workload error traces, disruption-grace states are labelled correctly, the AI status report stays informational (no un-actionable advisor prompts in one-way emails), and Claude Sonnet 5 is suggested as the default Anthropic model.

- SHA256: b37864f49d8130296ceeef35567a12bd5c562430fe01a0445c159306cd15e280

Table 53: New Features

Issue Number	Description	Details
#2364	feature - Workload: automatic capture & surfacing of scheduled-search execution errors (no AI required)	TrackMe now automatically stores the actual error messages produced by monitored scheduled searches and presents them in a dedicated **Execution-errors** view when you open the entity — no AI Advisor needed. The same error traces are also embedded in stateful-alert notifications, so you can see *why* a scheduled search failed straight from the alert.
#2277	feature - Wildcard targeting for bulk actions across the REST API	Bulk operations now accept `*` and `?` wildcards in the entity list, so an action can target a whole family of entities (an index, a sourcetype pattern, a naming convention) in a single call instead of enumerating each one. Available across bulk entity actions, acknowledgements, logical-group membership and ML outliers, with a safeguard that refuses a bare match-everything pattern.
#2270	feature - Custom display order for Virtual Tenants and Virtual Groups	Administrators can now arrange tenant cards, Virtual Group categories, and the groups within a category in a curated global order (drag to reorder), instead of the fixed alphabetical layout. The Virtual Tenants **Administration** menu is also grouped into labelled sections for easier navigation.
#2273	feature - New Flex Object use case: Cribl Logstream memory usage	A new ready-to-use Flex Object use case monitors Cribl Logstream memory usage, sharing the tracked entity with the existing CPU-usage use case.
#2372	feature - Remote Accounts: manual force bearer-token rotation	The Remote Accounts Overview now offers a manual **force token rotation** action — for all accounts at once or a single account — so administrators can rotate remote bearer tokens on demand rather than waiting for the automatic schedule.

Table 54: Enhancements & Changes

Issue Number	Description	Details
#2379	change - Workload: redesigned Versioning metadata experience	The Workload Versioning metadata tab is rebuilt from a raw data dump into a structured, filterable version history. Each tracked version is an expandable card showing who made the change, when, which properties changed, and a colour-coded before/after (unified) diff for every affected property — the search SPL and non-search settings alike (schedule, earliest/latest time range, wildcard). The full search definition is always shown, even when the change was on a non-search field. The tab adds summary cards (total versions, versions in the selected range, latest version and its author), a time-range selector (24h / 7d / 30d / 90d / 1y / custom, with a graceful fallback to the full history), a free-text filter across definition/user/version id/diff, an *"inspected … ago"* freshness indicator, expand/collapse-all, copy version id, and an "Open in Search" pivot. All client-side over the data TrackMe already stores — no extra searches.
#2308	change - Flex Objects: automatic default-metric backfill for older trackers	Flex Object trackers created from the library before default metrics existed now have their default metric backfilled automatically on upgrade, so opening the entity shows a meaningful metric instead of a raw status. A library audit also added the missing default-metric definitions to the shipped use cases.
#2296	change - Tenant Home: SLA drilldown opens the Audit SLA view inline	Clicking the SLA percentage on Tenant Home now opens the Audit SLA view embedded as a modal, keeping you in context instead of navigating away.
#2289	change - Prominent "Save pending changes" action on unsaved-changes banners	The unsaved-changes banners on the ML Outliers models and thresholds screens now include a clear **Save pending changes** button, so pending edits are easier to commit.
#2387	change - AI: Claude Sonnet 5 suggested as the default Anthropic model	When configuring an Anthropic AI provider, Claude Sonnet 5 is now offered as the suggested default model.
#2265	change - AI: support for gpt-5 and o-series reasoning models	AI provider requests now use the correct token and temperature parameters for OpenAI gpt-5 and o-series reasoning models, so those models work without errors while classic models are unaffected.
#2312	change - Flex Objects: retire the Splunk fields-quality use case	The legacy Splunk fields-quality Flex Object use case has been removed, as its role is now fully covered by the Field Quality Monitoring (FQM) component.
#2264	change - Dependency maintenance batch	A consolidated third-party dependency maintenance batch was applied as part of the 2.4.6 cycle, keeping bundled dependencies current.

Table 55: Bug Fixes

Issue Number	Description	Details
#2340	bug - Replica tenants: replicated entities could get a corrupted state	On replica tenants, replicated entities were being re-evaluated locally, which could corrupt their state and make them disagree with the source (for example a replica showing green while the source was red). Replicated entities now trust and mirror the source state, and the shadow view is refreshed immediately after a replica run.
#2350	bug - Replica entities: blank table fields and wrong red status	Replica entities could show blank display fields in the table, and Outliers / Data-sampling could be shown red incorrectly. Display fields are now enriched while keeping the source state.
#2346	bug - Replica execution did not refresh Tenant Home counters	After a replica execution, the Tenant Home donuts and KPI cards stayed at 0 until the next health-tracker cycle. They now refresh immediately once the replica run completes.
#2338	bug - Replica Virtual Tenants: Manage/Execute replica screens broken	The Manage and Execute replica-tracker screens were broken and gave no replica-specific context. They are repaired and now clearly indicate the replica context.
#2362	bug - Deleting a replica tracker could fail	Deleting a replica tracker could fail with a server error when the request did not include the component. The delete now succeeds without it.
#2392	bug - Replica wizard listed tenants without the required component	The replica creation wizard's source-tenant picker now only lists tenants that actually have the relevant component enabled.
#2375	bug - Workload: non-search versioning changes were lost on re-inspection	Workload versioning changes to non-search properties (schedule, earliest/latest time range, wildcard settings) were dropped from storage when an entity was re-inspected. All versioning differences are now preserved.
#2267	bug - Adaptive delay skipped variable-delay-policy tenants	Adaptive delay never selected entities using a variable delay policy, silently skipping them, and the delay flip-event investigation view could miss variable-delay breaches. Both now handle variable-delay entities correctly.
#2284	bug - Adaptive-delay setting ignored when creating DSM/DHM tenants	The adaptive-delay choice was not honoured during DSM/DHM tenant creation. The setting from the wizard is now applied.
#2293	bug - Disruption-grace state mislabelled in flip events	An entity held in a protective (blue) state by the disruption-queue grace period was mislabelled as logical-group protection in flip events. The state is now labelled correctly.
#2380	bug - Stateful alert failed for entities with no chart-window metrics	The stateful-alert action could fail when an entity had no component metrics in the alert's chart window. This case is now handled and the alert completes.
#2384	bug - AI status report proposed an interactive action inside a one-way email	The AI status report embedded in stateful-alert emails could suggest running an interactive AI Advisor, which is not actionable in an email. The report now stays informational.
#2367	bug - AI Concierge Advisor failed on every tenant	The AI Concierge Advisor failed for each tenant because it looked up the tenant record by the wrong key. It now resolves the tenant correctly.
#2377	bug - Noisy ML warning on empty input	Native ML training/scoring logged a context-free warning on empty input. The message is demoted and enriched, removing the log noise.
#2316	bug - Some changes were slow to appear in the entity table	Configuration changes (blocklist, lagging-class and logical-group) and several **Bulk edit** actions now refresh the entity table immediately — the Status, Score and thresholds update at once instead of waiting for the next tracker cycle (so an action no longer looks like it "did not work"). A related logical-group refresh error for components a tenant does not use is also fixed.
#2276	bug - Pretty-JSON command did not support array fields	The `trackmeprettyjson` command now handles JSON array fields in addition to objects.
#2327	bug - Cribl pipeline percentages could be blank on idle windows	The Cribl Logstream pipeline use case could produce empty sent/dropped percentages when no events flowed in a window. The percentages are now guarded against division by zero.

11.1.21 Version 2.4.5 - build 1782421503 (25/06/2026)

Hint

TrackMe 2.4.5 — expedited hotfix: AI Advisor on Splunk Cloud, and minor fixes

- **AI Advisor now starts reliably on Splunk Cloud (the reason for this release)**: on Splunk Cloud and Splunk hosted LLMs, the AI Advisor fails to start because Splunk's bundled Python pointed at a certificate file that does not exist in the app's runtime. The AI Advisor is now immune to this for every AI provider, while still fully verifying its HTTPS connections. The in-app AI Assistant chat was not affected. **Customers using AI features on Splunk Cloud are encouraged to upgrade.**
- **AI Assistant panel theme**: the AI Assistant side panel no longer follows the operating-system theme instead of the TrackMe theme on some browsers, so it always matches the page.
- **Cleaner upgrades from 2.3.19**: a harmless but misleading migration error that was logged on every cycle after upgrading from 2.3.19 is now resolved, and a few missing default labels are backfilled.
- **Acknowledgement indicator**: enabling or disabling an acknowledgement in bulk now updates the indicator in the Tenant Home table immediately, completing the shadow-records follow-ups from the previous releases.
- **Dependency maintenance**: the cycle's batched dependency updates.

- SHA256: f26f51c4a468338c270ceb2d5e4e64a857c15201211327da04c11a87bf965b16

Table 56: Bug Fixes

Issue Number	Description	Details
#2253	bug - AI Advisor failed to start on some Splunk Cloud search heads	On some Splunk Cloud search heads, Splunk's bundled Python advertises a certificate-bundle path that does not exist in TrackMe's runtime context. The AI Advisor builds its HTTPS clients eagerly and passed that stale path straight to the TLS layer, causing a *file not found* failure before any advisor could run. The AI Advisor path is now made immune to a stale certificate path for **all** AI providers, with full HTTPS verification preserved. The AI Assistant chat was never affected. A small diagnostic command was also added to validate the certificate handling directly in a given environment.
#2256	bug - AI Assistant panel rendered in the wrong theme on some browsers	The AI Assistant side panel could render in the wrong theme (for example a light panel on a dark page) on Edge/Windows, because it followed the operating-system colour scheme instead of the TrackMe theme. The panel now consistently follows the application theme.
#2252	bug - Misleading migration error after upgrading from 2.3.19	Tenants upgraded from 2.3.19 logged a harmless but incorrect schema-migration error on every cycle, and a few default labels were never seeded. The error is resolved and the missing default labels are backfilled automatically on upgrade.
#2248	bug - Acknowledgement indicator could appear stale in the Tenant Home table	Enabling or disabling an acknowledgement in bulk did not update the acknowledgement indicator in the Tenant Home table until the next tracker cycle refreshed it. The indicator now updates immediately, completing the same class of shadow-records fixes delivered in the previous releases.

Table 57: Enhancements & Changes

Issue Number	Description	Details
#2250	change - Dependency maintenance batch	A consolidated third-party dependency maintenance batch was applied as part of the 2.4.5 cycle, keeping bundled frontend dependencies current.

11.1.22 Version 2.4.4 - build 1782379508 (25/06/2026)

Hint

TrackMe 2.4.4 — Slipstream follow-ups, DHM & MHM reliability fixes, and minor corrections

- **Shadow records polish**: a set of follow-up fixes to the 2.4.3 Slipstream read cache. Notes, entity maintenance and entity details now always reflect the latest state immediately, instead of occasionally showing a stale value until the next tracker cycle.
- **Silent metric hosts now raise alerts (MHM)**: a metric host that stopped sending all of its metrics could stay green forever with no alert. TrackMe now re-evaluates fully-silent hosts, so they turn red as expected and SLA and alerting follow.
- **More robust host coverage gap analysis (DHM)**: large reference inventories are now read in full, with no silent truncation, and matched more reliably. Hosts are no longer mis-reported as missing on either side.
- **MHM entity Overview tab restored**: the per-entity Overview tab, broken since 2.3.16, works again.
- **Logical groups protected against accidental member removal**: fixed a case where valid logical-group members could be wrongly purged on certain tenants.
- **Minor fixes**: a half-width Workload chart is corrected, automated AI ML model review now also covers FQM and Workload entities, plus documentation and dependency maintenance.

- SHA256: 520cce3c65fd78ec1f2cc5da84dfb7ae7e065323390ce051c5238a1e506d39b4

Table 58: Enhancements & Changes

Issue Number	Description	Details
#2216	change - Automated AI ML Advisor extended to FQM and Workload	The automated AI ML model review now also covers **Field Quality Monitoring** and **Workload** entities, which already create ML outlier models. Automated review stays opt-in and off by default for these components, and the change also clears a harmless error that was logged when a tenant was deleted or disabled.
#2214	change - Security and dependency maintenance batch	A consolidated security and third-party dependency maintenance batch was applied as part of the 2.4.4 cycle, including security-relevant frontend dependency upgrades, keeping bundled dependencies current.

Table 59: Bug Fixes

Issue Number	Description	Details
#2224	bug - Metric hosts that go fully silent stayed green with no alert (MHM)	A metric host that stopped sending **all** of its metrics dropped out of the live monitoring window and stayed frozen in its last green state — no alert was raised and its SLA stayed at 100%. Partial interruptions were already detected; full silence was not. TrackMe now re-evaluates fully-silent hosts from their stored history, so they turn red as expected, SLA degrades and alerts fire, and they recover automatically when metrics resume.
#2241	bug - Host coverage gap analysis unreliable on large inventories (DHM)	When reconciling a large reference inventory against tracked hosts, very large inventories were silently truncated and some hosts were mis-reported as present in only one of the two sides. The full inventory is now read without truncation, and host matching is more robust, so the comparison is accurate even on very large inventories.
#2222	bug - MHM entity Overview tab was broken	The per-entity Overview tab for Metric Host Monitoring referenced a command removed in 2.3.16 and had been broken since. The tab now renders its metric breakdown correctly again. A build-time safeguard was also added to catch this class of broken command reference in the future.
#2229	bug - Notes could appear stale in entity tables	After adding, editing or deleting a note, the change could remain invisible in the entity table until the next tracker cycle refreshed it. Note changes are now reflected immediately.
#2232	bug - Entity maintenance state could appear stale in entity tables	Placing an entity into, or out of, maintenance could remain invisible in the entity table until the next tracker cycle. The maintenance state now updates immediately.
#2234	bug - Entity SLA could be shown stale in detail views	In single-entity detail views, the SLA indicator could display a stale value because it was read from a cached record instead of being recomputed. Detail views now always show the live SLA.
#2218	bug - Tenant Home entity Overview modal showed incomplete details	The entity Overview modal on Tenant Home could open with some detail fields missing because it rendered the lightweight table row. The modal now enriches the row with the full entity record before displaying it.
#2220	bug - Workload Overview chart rendered at half width	The Workload Overview chart could render at half of its available width inside a modal. The chart now measures its container correctly and uses the full width.
#2244	bug - Logical-group members could be wrongly purged on some tenants	On tenants whose component-enablement flags were stored in a particular format, the logical-group housekeeping task could mis-classify valid members as orphans and remove them. The check now interprets those flags consistently, so members are no longer removed by mistake.

11.1.23 Version 2.4.3 - build 1782214134 (23/06/2026)

Hint

TrackMe 2.4.3 — Slipstream: a major performance & scalability initiative, configurable backups & S3 export, Saved Views, and AI advisor evolutions

- **Slipstream — performance & scalability initiative (the headline of this release)**: TrackMe now makes dramatically fewer REST round-trips on its hot paths, which is the dominant cost on large standalone and Search Head Cluster deployments. Dozens of redundant or sequential calls previously made by the user interface and the schedulers are now consolidated into a single load, resolved in-process, or removed entirely. At the centre of the initiative is the **shadow filesystem cache** — a local read cache that serves hot

KV-store reads without a network round-trip — which is **now enabled by default for every tenant** (new tenants, and every qualifying existing tenant at upgrade). The result is a faster, lighter user interface and materially less pressure on the Splunk KV-store proxy layer, with **no configuration required**.

- **Saved Views**: administrators can curate named, RBAC-scoped subsets of tenants and Virtual Groups, giving each team a focused view of only the tenants relevant to them.
- **AI advisor evolutions**: a dedicated FQM dictionary-generation panel (choose the provider and add scoped instructions without a chat detour), natural-language generation of Flex Object trackers from the AI Assistant, support for the xAI (Grok) reasoning-effort parameter, and protection against AI automation running on tenants owned by *nobody*.
- **Configurable SLA calculation window**: the SLA percentage time window (previously fixed at 90 days) is now configurable, for deployments where searches cannot run over the full default window.
- **Configurable backup directory**: the backup root location is now configurable from the *Backup & Restore* settings (with environment-variable support) and falls back safely to the default location — raising a Configuration Guardian alert — if the configured path is ever unusable.
- **Backup retention managed from the UI**: backup retention is now a setting in the *Backup & Restore* tab instead of being hard-coded in the scheduler search.
- **Optional S3 export of backups**: backup archives can optionally be exported to an S3-compatible object store, with a built-in connectivity test and a fail-open design so a remote issue never blocks local backups.

- SHA256: 045ed3670efb4e74259bdf0153d29ee5a6f453007fc4a20cb362368a06a6cb7c

Table 60: New Features

Issue Number	Description	Details
#2138	feature - Slipstream — REST round-trip reduction & shadow filesystem cache (enabled by default)	A wide-ranging performance and scalability initiative that reduces the number of REST round-trips TrackMe makes on hot paths — the main constraint on large deployments. Many calls the UI and schedulers used to make individually are now consolidated into a single load, resolved in-process, or eliminated, and the **shadow filesystem cache** serves hot KV reads locally. Shadow records and the cache are now **enabled by default for all tenants** (new tenants and, at upgrade, every qualifying existing tenant), with no configuration required. The net effect is a snappier interface and significantly less load on the Splunk KV-store proxy layer.
#2073	feature - Saved Views — curated, RBAC-scoped tenant subsets	Administrators can define named **Saved Views** — curated subsets of tenants and Virtual Groups scoped by role — so each team lands on a focused view of just the tenants relevant to them instead of the full estate.
#2126	feature - Configurable backup directory with safe fallback	The location where TrackMe stores backup archives is now configurable from the *Backup & Restore* settings, with environment-variable support. If the configured directory is ever missing or not writable, TrackMe automatically falls back to the default location and raises a Configuration Guardian alert, so backups never silently stop.
#2124	feature - Backup retention managed from the UI	Backup retention (in days) is now managed as a setting in the *Backup & Restore* tab, rather than being hard-coded in the backup scheduler search.
#2130	feature - Optional S3 export of backup archives	Backup archives can optionally be exported to an S3-compatible object store. The integration includes a connectivity test and is fail-open by design, so an export or connectivity problem never blocks the local backup.
#2192	feature - FQM — dedicated AI dictionary-generation panel	Field Quality Monitoring gains a dedicated panel to generate a CIM-style field dictionary with the help of AI: pick the AI provider and supply scoped instructions directly, without going through the conversational assistant.
#2013	feature - AI Assistant — natural-language generation of Flex Object trackers	The AI Assistant can now turn a plain-language description into a ready-to-create **Flex Object tracker**: it proposes the search, constraint and KPIs, previews the matched entities, and lets the operator apply it through the existing wizard.
#2063	feature - Configurable SLA percentage calculation window	The time window used to compute SLA percentages — previously fixed at the last 90 days — is now configurable, for deployments where a workload rule or RBAC restriction prevents searches from running over the full default window.
#2144	feature - Notes — optional validity period with automatic purge	Entity and tenant notes can now be given an optional validity period, after which they are purged automatically — useful for time-bound operational annotations that should not linger.
#2040	feature - Policies & coverage — richer CMDB lookup matching	Lookup- and CMDB-based matching used by priority/tags/SLA policies and coverage gap analysis is now more expressive: multiple field mappings can be combined with OR/AND logic, the DHM `asset` multivalue field can be matched against (recognising a host by any of its known identifiers), and foreign value mappings support wildcards.
#2067	feature - Coverage gap analysis — multiple host-column matching	Host coverage gap analysis can now match against several host columns at once via OR-combined match blocks, so reference data that spreads identifiers across multiple columns is correctly reconciled.

Table 61: Enhancements & Changes

Issue Number	Description	Details
#2051	change - Policies — faster, non-blocking simulation on large CMDBs	Priority/tags/SLA policy simulation and apply no longer block or time out on large lookups: the simulation now runs asynchronously and the lookup matching is indexed, removing the previous slowdown (and nginx 504s) on large CMDBs. The UI also explains that wildcard match mode is significantly slower than exact matching.
#2080	change - Bulk edit — lock the threshold in the same operation	When bulk-editing DSM/DHM lagging policy or variable delay, the threshold can now be locked as part of the same operation (and the bulk selector defaults to *Lock*), so operators no longer need a second pass to pin their intent.
#2092	change - ML Outliers — edit an existing period of exclusion	An existing ML model period of exclusion can now be edited in place, instead of having to delete and recreate it.
#2070	change - Configuration Guardian — Splunk Cloud SHC member self-heal	For remote Splunk Cloud deployments, Configuration Guardian now detects and self-heals drift in the list of Search Head Cluster members of a remote account, and records an informational notice of the action taken.
#2097	change - Per-entity maintenance — inline indicator	Entities currently under per-entity maintenance now show a small icon next to the entity name, so an operator can see at a glance which entities are in a maintenance window.
#2181	change - Virtual Tenants — per-tenant Clear operational status shortcut	The Virtual Tenants page now offers a per-tenant shortcut to clear operational status, instead of having to open the tenant first.
#2157	change - Flex Objects — graphical scope editor for converging trackers	The converging-tracker edit modal now includes a graphical tenant/component picker and filter editor, making it easier to adjust the scope of a converging tracker without editing raw search syntax.
#2075	change - AI providers — xAI (Grok) reasoning effort & refreshed models	The xAI (Grok) provider now exposes a `reasoning_effort` parameter, and the bundled Grok model list has been refreshed.
#2099	change - Audit — sortable tracker runtime table	The tracker runtime table on the Trackers Performance audit page now has sortable column headers.
#2061	change - Audit — show the Result field in the SLA Flip activity modal	The SLA Flip activity modal now includes the *Result* field, giving more context on each SLA state change.
#2146	change - AI audit dashboards — view full error messages	Long error messages in the AI advisor audit dashboards are no longer truncated with no way to read them: the full text can now be opened in a modal.
#2187	change - Support Diag — include ML outlier models	Entity-scoped support diagnostics now collect the ML outlier models (rules, data and native model), making it easier to troubleshoot outlier behaviour with TrackMe support.
#2190	change - Coverage gap analysis — surface skipped reference rows	Coverage gap analysis now reports which reference rows were skipped because they were corrupted, instead of silently ignoring them.
#2136	change - Health tracker — untracked entities evaluated every cycle	Detection of untracked entities is promoted so it runs on every health-tracker cycle, surfacing newly untracked entities faster.
#2149	change - AI automation — protection on tenants owned by *nobody*	Automated AI advisor runs are now guarded against tenants whose owner is *nobody* (which previously failed with a privileged-execution error), and a persistent banner makes the condition visible so an administrator can reassign ownership.
#2123	change - Consolidated security & dependency maintenance batch	A consolidated security and third-party dependency maintenance batch was applied during the 2.4.3 cycle, keeping bundled frontend and backend dependencies current.

Table 62: Bug Fixes

Issue Number	Description	Details
#2094	bug - AI status report failed on the Splunk-hosted LLM provider	AI-generated status reports in stateful alerts could fail with an HTTP 413 error against the Splunk-hosted LLM provider when the prompt exceeded the provider's 64k-token cap. The prompt is now bounded to stay within the cap, so the report is generated reliably.
#2101	bug - Workload — scheduler KPIs reported in the wrong time window	Workload (WLK) scheduler KPIs were stamped at tracker-run time rather than event time, so delayed or overlapping runs reported their metrics in the wrong window. KPIs are now stamped at event time and land in the correct window.
#2086	bug - ML Outliers — could not define a future period of exclusion	It was not possible to define a period of exclusion in the future, and the rejection message was misleading. Future-dated exclusions are now allowed and the messaging is clear.
#2042	bug - DHM — asset field extracted a spurious value for IPv4 hosts	The DHM `asset` field incorrectly extracted the first octet of an IPv4 address as a separate identifier. It now records the correct identifiers only.
#2163	bug - Flex Objects — Modify modal failed for legacy converging trackers	The converging-tracker *Modify* modal returned a 404 for legacy trackers that predate the stored KV properties. These trackers are now resolved correctly and can be modified.
#2084	bug - Tenant Home — excessive client-side CPU usage	The Tenant Home page consumed excessive browser CPU due to leaked observers and a full-tree re-render every second. The page now refreshes efficiently, with much lower CPU usage on long-lived sessions.
#2198	bug - Virtual Tenants — disruptive auto-refresh of the Entities Overview modal	The automated refresh of the Entities Overview modal was visually disruptive (full-screen spinner, chart flash, tab reset). The refresh is now seamless and preserves the operator's place in the modal.
#2088	bug - Audit dashboards — missing inline search in dropdown filters	Dropdown filters on the audit dashboards were missing the inline search box, making long lists hard to navigate. The search box is restored.
#2090	bug - Virtual Tenants — spurious error when cancelling a loading modal	Cancelling a still-loading modal surfaced a spurious "signal is aborted without reason" error. Cancellations are now handled cleanly without a misleading error.
#2170	bug - Stateful alerts — broken embedded chart images	Embedded PNG charts in stateful-alert notification emails could render as broken images after a charting-library update. The dependency is pinned to a compatible version so charts render correctly again.
#2112	bug - DSM/DHM — delayed-inspector error on partial responses	The delayed-data inspector could raise an internal error when an entity information lookup returned a non-200 response. It now handles partial responses gracefully.
#2059	bug - UI — modal content flashing on close	Several management and creation dialogs briefly flashed their content while closing. The dialogs now close cleanly without the flash.

11.1.24 Version 2.4.2 - build 1781257469 (12/06/2026)

Hint

TrackMe 2.4.2 — ML training hotfix, Splunk-hosted LLM hotfix & minor evolutions

- **ML model training restored**: a regression introduced in 2.3.22 left ML outlier model training fully broken across all components — the training search requested lookup output fields that no longer existed, so no model could be trained. This release fixes the upstream search and, as a safety net, now flags a tenant as

degraded when a training or monitoring execution fails abnormally instead of failing silently. **Customers using ML outlier detection are strongly encouraged to upgrade.**

- **Splunk Cloud-hosted LLM provider fixed for AI Advisors**: AI Advisors could not run against the Splunk-hosted LLM provider (SLIM) — the agent provider bridge mistakenly routed to `api.openai.com`. The `splunk_hosted` provider is now correctly wired into the bridge, and the Azure OpenAI provider contract has been repaired on the same path.
- **Screen-adapted AI Assistant**: the in-app AI Assistant now exposes context and pre-built questions adapted to the specific screen across the Tenant Home and Virtual Tenants administration modals, instead of a generic context.
- **Assorted UI & audit fixes**: a searchable Permanently Deleted Entities modal, a corrected Trackers-performance deep-dive link, and a fix to the empty runtime-statistics chart for direct trackers.

- SHA256: 137a9e47015e6b6f1155e83d011dd820619904dd2f5ea51ab4696ffb53d68e4b

Table 63: Enhancements & Changes

Issue Number	Description	Details
#2008	change - Tenant Home — screen-adapted AI Assistant context & questions	The AI Assistant available across the Tenant Home administration modals now presents **context and pre-built questions adapted to the active screen** (44 AI-wired modals), instead of a generic context. Each modal surfaces suggested questions relevant to what the operator is configuring, making the assistant materially more useful in context.
#2004	change - Virtual Tenants — screen-adapted AI Assistant context & questions	The same screen-adaptation is applied to the **Virtual Tenants** administration screens: the AI Assistant context and its suggested questions are now tailored to the admin screen in use rather than a generic context.
#2010	change - Permanently Deleted Entities modal — search bar	The *Permanently Deleted Entities* management modal (Tenant Home) now includes a **search bar**, so administrators can quickly filter the list of permanently deleted entities instead of scrolling through it.
#1999	change - Consolidated security & dependency maintenance batch	A consolidated security and third-party dependency maintenance batch was applied as part of the 2.4.2 cycle, keeping bundled frontend and backend dependencies current.

Table 64: Bug Fixes

Issue Number	Description	Details
#2018	bug - ML model training fully broken since 2.3.22	A regression introduced in 2.3.22 broke ML outlier **model training across all components**: the `mltrain` upstream search referenced `lookup … OUTPUT` fields that no longer exist, so the training search failed and no model could be produced. The upstream search is fixed, and abnormal `mltrain` / `mlmonitor` executions now **flag the tenant as degraded** so the failure is surfaced instead of passing silently. Customers relying on ML outlier detection should upgrade.
#2023	bug - Bulk *Run ML Train* / *Run ML Monitor* stopped mid-list	When triggering ML training or monitoring in bulk from the UI, the operation could randomly stop part-way through the selection because the background thread was reaped by splunkd. The bulk action now runs as a **durable detached search job**, so it completes the whole list reliably.
#2014	bug - AI Advisors failed with the Splunk-hosted LLM provider (SLIM)	Running any AI Advisor against the **Splunk-hosted LLM provider** failed: the agent provider bridge routed requests to `api.openai.com` with a placeholder API key instead of the Splunk SLIM endpoint. The `splunk_hosted` provider is now correctly wired into the agent provider bridge so AI Advisors work on Splunk Cloud's hosted LLM.
#2019	bug - AI Advisors failed with the Azure OpenAI provider	On the same agent provider bridge, the **Azure OpenAI** provider contract was never applied (missing `api-version` and API-key wiring), so AI Advisor runs against Azure OpenAI failed. The Azure contract is now correctly wired via an httpx hook.
#2007	bug - Trackers-performance deep-dive opened a removed dashboard	The Trackers-performance deep-dive link pointed at a classic dashboard that has been removed, opening a dead view and losing the prefilled context. The link now opens the native React *Trackers Performance* audit page with its context preserved.
#2002	bug - Runtime statistics chart empty for direct trackers	In the Virtual Tenants operational / scheduler status view, the runtime statistics chart was empty for direct trackers because of a tracker/wrapper metric-name mismatch. The chart now resolves the correct metric name and renders the runtime statistics as expected.

11.1.25 Version 2.4.1 - build 1781039179 (09/06/2026)

Hint

TrackMe 2.4.1 — Threshold Lock, Topology graphs, Coverage gap analysis & audit dashboard rewrite

- **Unified Threshold Lock (DSM & DHM)**: a single, safer control replaces the former overlapping *override lagging classes* / *allow adaptive delay* / *lock* options. Locking an entity guarantees the operator's intent — the pinned threshold (static or variable) is never silently overwritten by automation — and is clearly surfaced in the UI with a lock badge. A safety-net reconcile restores any drift and leaves an audit note explaining what was corrected.
- **Automatic Label Assignment**: tenant admins can now define rules that automatically assign tenant labels to entities based on their lifecycle (discovered, enters alert, recovers) or a custom filter expression — with additive or self-reconciling removal — evaluated by the scheduled decision maker, so labelling no longer has to be done by hand.
- **Topology graphs for every component**: a new built-in topology view shows the relationships between nodes (tenant → groups → entities) as interactive link and network graphs, now available for DSM, DHM, MHM, WLK, FQM and Flex Objects.

- **Coverage gap analysis (DSM & DHM)**: a new built-in UI compares TrackMe's knowledge against a CMDB lookup or a live Splunk query in a few clicks, surfacing what is not covered (hosts, indexes, sourcetypes…) in both directions, each gap downloadable as a CSV.
- **DHM asset field**: each host entity now carries a normalised set of all known identifiers (short name, FQDN, alias…), so lookup-based operations recognise a machine however it is named — used by both *Inject Expected Hosts* and the coverage gap analysis.
- **AI Feed Lifecycle Advisor — Data Sampling**: the advisor now understands the DSM Data Sampling feature and can assist the generation of custom sampling rules (regex models) directly from sample events, surfaced inside the existing custom-rule wizard.
- **Adaptive delay**: now surfaces its activity directly on entity nodes through automated summary notes, and drops its last dependency on the Splunk Machine Learning Toolkit native ML commands. TrackMe no longer requires the Splunk AI Toolkit nor the Python for Scientific Computing add-on.
- **Audit dashboards rewrite**: all audit dashboard studios have been fully rewritten as native React user interfaces, with assorted UX improvements.
- **FLX converging trackers**: improved visibility of the underlying member entities directly from the metrics *inspect* screen, plus UI-driven in-place modification of a converging tracker.

- SHA256: 013e04a1e0ec459ac1b606e384b91479d3c23932faa672a0fdfee90bc481f280

Table 65: New Features

Issue Number	Description	Details
#1892	feature - Unified Threshold Lock for DSM & DHM thresholds	A single **Threshold Lock** control replaces the former overlapping *override lagging classes*, *allow adaptive delay* and *lock* options on DSM/DHM entities. Locking an entity guarantees the operator's intent: the pinned threshold — static or variable — is honoured and never silently overwritten by adaptive delay, variable-delay auto-review or lagging-class automation. The lock is clearly surfaced in the UI (a red lock badge on the entity, an explicit *Lock & apply* / *Apply without locking* confirmation when editing a threshold), and an independent safety-net reconcile restores any drift on a locked entity and writes an operator note describing what was corrected. A per-tenant master toggle (`delay_threshold_lock_enabled`) is provided.
#1979	feature - Automatic Label Assignment (rule-driven auto-labels)	Tenant admins can now define rules that **automatically assign tenant labels to entities**, configured from *Manage Labels → Automated* and evaluated by the scheduled decision maker. Each rule fires on an entity lifecycle event — **discovered**, **enters alert**, **recovers** — or when an entity matches a **custom filter** expression, and can be scoped by priority and a filter DSL. Removal is per-rule: **manual** (the label is added and kept) or **auto** (the label is present only while the condition holds and is removed when it clears, touching only the labels that rule owns). The feature is additive and costs nothing for tenants with no rules.
#1871	feature - Topology graphs for every monitoring component	A new built-in **topology** view shows the relationships between nodes for DSM, DHM, MHM, WLK and FQM, reachable from the Tenant Home header (contextual to the active component). Each component uses a hierarchy that fits its data model (e.g. DSM: tenant → data index → entity; DHM: tenant → host → index → sourcetype). The modal offers **link graph** and **network graph** views with per-node status colours, search, All / Down / Up filtering and a member-node cap that keeps every group visible. Data is fetched only when a topology is opened.
#1870	feature - Topology graphs for Flex Objects	Extends the topology view to **Flex Objects** (converging and use-case trackers), with smarter topology entry points and a more accurate Flex hybrid tracker type so the relationships between Flex nodes are represented faithfully.
#1864	feature - DHM Host coverage gap analysis	A new **Host coverage gap analysis** tool reconciles a reference inventory (a CMDB lookup, or a live Splunk search via tstats/raw) against the hosts TrackMe actually tracks, and reports the gaps in both directions — hosts present in the reference but not tracked, hosts tracked but absent from the reference, and hosts covered on both sides. Matching is asset-aware (short name ⇄ FQDN), and each set is downloadable as a CSV. Launched from the Tenant Home DHM view.
#1876	feature - DSM Feeds coverage gap analysis	The DSM counterpart of the Host coverage gap analysis. Reconcile a reference (a lookup, or a live Splunk search) against the DSM feeds TrackMe tracks, on a configurable break-by grain (default `index, sourcetype`, reducible to `index` or extensible with custom fields), and download the *only in reference*, *only in TrackMe* and *in both* sets as CSV. Launched from the DSM one-shot actions menu.
#1853	feature - DHM asset field & asset-aware host recognition	Each DHM host entity now carries an **asset** field — a normalised, deduplicated set of all known identifiers for the endpoint (the tracked object, the bare value, the short hostname extracted from an FQDN, and the alias). This lets TrackMe recognise a machine however it is named: *Inject Expected Hosts* now matches incoming lookup rows against the asset field (short ⇄ FQDN aware, on by default), avoiding duplicate host entities, and the same normalisation powers asset-aware lookup comparisons across the product.

Table 66: Enhancements & Changes

Issue Number	Description	Details
#1904	change - Audit dashboards rewritten as native React UIs	All audit dashboard studios have been fully rewritten as native React user interfaces, with assorted UX improvements: Tenants Operational Status (#1904), Trackers Performance (#1906), KVstore collections (#1908), SVC usage (#1910), Data Sampling (#1900), and the adaptive delay threshold audit dashboards unified into a single UI (#1898).
#1965	change - SLA compliance reporting rebuilt as a single React-native page	The SLA compliance reporting overview and its flip drilldown — previously two separate Dashboard Studio views — are now a single native React page. Clicking an object opens its flip activity in an **in-place modal** (count, over-time chart and transitions table) instead of a new browser tab, keeping context. The existing entry points (Virtual Tenants *Open SLA overview* and the Tenant Home *SLA PCT* KPI) are preserved.
#1963	change - AI Assistant available in the ML Outliers tenant-scope modal	The *Manage: ML Outliers scope* modal now exposes the in-header AI Assistant trigger, consistent with the other Tenant Home admin modals, with context-adapted suggested questions to help configure the priority filter, the filter-expression DSL and the Volume KPI override.
#1873	change - Adaptive delay no longer depends on the Splunk Machine Learning Toolkit	Adaptive delay now uses TrackMe's native density function in place of the residual MLTK `fit DensityFunction` command, removing the last dependency on the Splunk Machine Learning Toolkit's native ML commands. As a result, TrackMe no longer requires the Splunk AI Toolkit nor the Python for Scientific Computing add-on to be installed.
#1877	change - Adaptive delay surfaces its activity through automated summary notes	When adaptive delay updates an entity's threshold, it now automatically writes a Markdown **summary note** on the entity describing the change, so its activity is visible directly on the entity node. Extended to the variable-delay slot path (#1890).
#1953	change - AI wizard modes — additional-instructions textarea on the consent card	The AI wizard consent card now offers an optional pre-launch *additional instructions* textarea, letting the operator pass extra guidance to the advisor before the run starts.

Table 67: Bug Fixes

Issue Number	Description	Details
#1884	bug - Adaptive delay never ran for variable-delay entities	Adaptive delay was unintentionally disabled for variable-delay entities (`allow_adaptive_delay` force-set to false), so those entities never benefited from adaptive thresholds. The mutual exclusion between the two features is now correctly reconciled and adaptive delay runs as expected.
#1880	bug - Adaptive delay produced no threshold on the native engine	Following the move from MLTK to the native density engine, the computed upper bound was always 0, so adaptive delay silently produced no usable threshold. The native computation now returns the correct value.
#1888	bug - AI Assistant read several tenant settings from the wrong source	The AI Assistant's describe layer read a number of tenant settings (CMDB, delay policy, impact score, monitoring, default priority) from a stale source, so they always appeared at their default values. The values are now read from the authoritative record and reported correctly.
#1886	bug - Health tracker re-asserted shadow enablement every cycle	The health tracker re-applied the shadow-enablement setting on every cycle because it read the value from the wrong source. It now reads the authoritative record and only acts when the setting actually changes.
#1896	bug - Tenant Knowledge Objects — *Open in manager* hid saved searches & alerts	The *Open in manager* action could hide saved searches and alerts with an empty owner (rewritten to the current user). Ownership is now preserved so the objects are listed as expected.
#1894	bug - Tenant Knowledge Objects — *transforms* filter returned no rows	The *transforms* kind in the Tenant Knowledge Objects filter never returned any rows. The filter now resolves correctly and lists the matching objects.
#1951	bug - FQM wizard — AI dictionary generation failed when invoked too early	Generating the data dictionary with AI in the FQM wizard could fail with an HTTP 400 error when invoked before the prerequisites were ready. The action now waits for the required context and succeeds.
#1841	bug - FLX converging tracker wizard — duplicate label	The FLX converging tracker wizard displayed a duplicate *Up/Down entities:* label. The redundant label has been removed.
#1939	bug - Inconsistent component naming in backend logs	Backend logs referred to components inconsistently (short form vs the `splk-` long form). Component naming is now consistent across the log output.

11.1.26 Version 2.4.0 - build 1780402698 (02/06/2026)

Hint

TrackMe 2.4.0 — AI Agents (TrackMe Agentic, GA)

- **AI Advisors (GA)**: six AI agents inspect TrackMe's monitored entities, propose grounded recommendations, and — in *act* mode, with consent — apply them through TrackMe's REST API. ML Advisor (outlier models), Feed Lifecycle (DSM/DHM thresholds), FLX Threshold, FQM (field quality + data dictionary), Component Health (WLK/MHM), and the catalog-driven Concierge.
- **Inspect by default**: every advisor runs read-only first and emits typed recommendations; *act* mode applies changes and verifies each write. Write tools are absent from the agent in inspect mode.
- **AI Assistant ↔ Advisor bridge**: the chat proposes an advisor as a one-click consent card, launches it inline with the session's resolved entity/tenant context, and renders the structured result — the chat never calls a write tool directly.
- **AI Concierge**: a catalog-driven generalist grounded in the live REST API; zero static write tools — every action is a consent-card click, with per-action typed confirmation for destructive operations.

- **AI Agents automation**: per-tenant scheduled advisor runs, scoped by priority / freshness filters, with a decommission guard disabled by default.
- **Audit**: two dashboards — *Review AI Advisor activity* and *Review Scheduled AI Advisor activity* — plus an `[AI Agent]` prefix on every entity write.
- **AI Provider accounts**: configure one or more providers — OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral, Ollama, Splunk Hosted (SLIM API), xAI — with optional Anthropic prompt caching.
- **Compatibility**: fully opt-in and fully compatible with all supported Splunk releases. The AI Advisors require Splunk 10.2.x and later (Python 3.13.x); the AI Assistant and AI Concierge run on Python 3.9.x and later. TrackMe blocks an advisor on an incompatible release, and the Assistant will not propose one.
- **Smart Status retired**: superseded by the AI Advisors; schema migration `2401` removes the legacy Smart Status collection on upgrade.
- **Non-AI fixes**: restored shared-library logging traces in the REST request path, KV de-duplication and pagination correctness, an `object_id` alias on write endpoints, per-entity maintenance mode, browser-local time display for variable-delay/threshold slots, and a SplunkUI dependency pin.
- See the new *Artificial Intelligence* documentation section.

- SHA256: 4b203a4e681ae63cd0bd62bfc781bc6f3a427d69c4eb7fdc7d25eb9f350c8bfd

Table 68: New Features

Issue Number	Description	Details
#1760	feature - AI Advisors (TrackMe Agentic) — six AI agents with inspect / act modes	Introduces six LLM-powered advisors built on a shared Splunk Agent SDK runtime (`splunklib.ai`). **ML Advisor** reviews native outlier-detection models (anomaly windows, period exclusions, retrains, false-positive clears); **Feed Lifecycle** tunes DSM/DHM delay & lag thresholds, adaptive/variable delay, monitoring state, priority and impact-score weights; **FLX Threshold** calibrates FLX dynamic thresholds; **FQM Advisor** manages field-quality thresholds and the per-tracker data dictionary (with wizard-time generation); **Component Health** triages WLK and MHM. Each advisor runs **inspect** (read-only, structured recommendations) by default and, where enabled, **act** (applies allowlisted write tools through TrackMe's REST API and re-reads to confirm). Every write is RBAC-checked at the REST boundary and tagged `[AI Agent]` in the entity audit trail. See *The AI Advisors*.
#1760	feature - AI Assistant ↔ Advisor bridge and AI Concierge	The AI Assistant proposes an advisor as a typed action contract rendered as an inline **consent card** (*Run now* / *Inspect-only first* / *Cancel*), launches it with the chat session's resolved entity/tenant context (never LLM-constructed identifiers), shows a live tool-call feed and renders the structured result — the chat never calls a write tool directly. The **AI Concierge** is a catalog-driven generalist grounded in the live TrackMe REST API with **zero static write tools**: it proposes structured action contracts that fire only on the user's Confirm click, with per-action typed confirmation for destructive operations. See *The AI Concierge*.
#1760	feature - AI Agents automation, audit dashboards and AI Provider accounts	**Manage AI Agents automation** configures per-tenant scheduled advisor runs (enable each advisor independently, pin a provider, set priority / SLA / tag filters and custom instructions, set operational caps; decommission disabled by default). **Manage AI Provider accounts** supports OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral, Ollama, Splunk Hosted (SLIM API) and xAI, with optional Anthropic prompt caching. Two audit dashboards — **Review AI Advisor activity** (interactive + automated) and **Review Scheduled AI Advisor activity** (scheduled-only, with an hour-of-day cadence histogram and a per-tenant × advisor last-run matrix) — record every run to the summary index. See *AI Agents automation*, *Auditing AI activity* and *Configuring AI Provider accounts*.
#1789	feature - Per-entity maintenance mode (force BLUE for a window)	New per-entity maintenance window that forces a single entity into `BLUE` for a configured duration, distinct from the tenant-wide and global maintenance modes — lets you suppress one noisy entity without silencing the whole tenant's alerting. Available from the entity *Actions* menu and the Tenant Home overview.
#1785	feature - Variable-delay / FLX variable-threshold slots shown in browser-local time	Variable-delay slot hours and FLX variable-threshold slot hours now display in the browser's local timezone (with a UTC-offset annotation) instead of raw UTC. Storage is unchanged — UTC remains canonical in KV; the conversion is render-side only.

Table 69: Enhancements & Changes

Issue Number	Description	Details
#1760	change - Smart Status retired — replaced by the AI Advisors (schema migration `2401`)	Smart Status is fully removed and superseded by the AI Advisors, which produce richer, more actionable recommendations. Schema migration `2401` drops the legacy Smart Status KV collection on upgrade; the `trackme_smart_status` alert action and its handlers/commands are removed. Saved alerts that referenced the Smart Status alert action silently no-op after upgrade — review and replace them with AI Advisor invocations (via the chat bridge or the scheduled batch).
#1784	change - REST — accept `object_id` as an alias for `keys_list` on write endpoints	Write REST endpoints now accept `object_id` as an alias for `keys_list` (the value is identical — the SHA256 `_key`), removing a frequent small-model failure mode where AI assistants supplied the right value under the wrong field name. Applied across 18 handlers / 78 call sites; the polymorphic shape (string / CSV / list) is preserved and existing `keys_list` callers are unchanged. An INFO line records when the alias path is taken so the rate can be measured.
#1760	change - Splunk Agent SDK integration, Anthropic prompt caching and agent reliability hardening	Pins `splunk-sdk>=3.0.0` (the `splunklib.ai` runtime) with the langchain provider bindings and an MCP tool inventory. Per-advisor Anthropic **prompt caching** (`ai_prompt_caching_enabled`, default on) reduces input cost on the cached system-prompt prefix after the first call. Reliability hardening adds per-call HTTP timeouts, transient-error retry with backoff (inspect-only, so non-idempotent writes are never replayed), a silence watchdog, and a structured-output fallback for providers with stricter schema handling.

Table 70: Bug Fixes

Issue Number	Description	Details
#1816	bug - Logging — restore shared-library traces in the REST request path	After the 2.3.24 logging-hygiene change removed a root-logger redirect from every REST handler, library-level log calls made during a REST request silently vanished (e.g. tenant creation logged only two events, without the per-object creation traces). A new `contextvars`-based effective logger binds the handler's named, non-propagating logger for the duration of dispatch without mutating shared global state, restoring the per-object, `tenant_id`-tagged traces; background threads (custom commands, schema migrations) are unchanged. A CI guard scans every `trackme_libs*.py` for bare `logging` calls to prevent regression.
#1815	bug - KV — consistent de-duplication identifier in full-collection read helpers	Four full-collection KV read helpers tested membership on `_key` but de-duplicated by `object` / `app`, so the returned list and dict could disagree on cardinality when two distinct `_key` records shared the same `object` / `app`. Each call site now uses the correct identifier (`object` / `app` for the read helpers; `_key` for stateful-incident cleanup, which must keep all distinct records). A correctness cleanup with new unit coverage.
#1806	bug - KV — byte-cap pagination short-page guard re-applied	Re-applies the byte-cap pagination short-page guard on `version_2400` after earlier native-pagination work on the branch was reverted. Without it, Python-side KV reads of collections that exceed the per-page byte cap could silently drop records when a page returned short.
#1748	bug - SplunkUI — Outliers time-range picker crash (`isDev is not a function`)	The TenantHome / Outliers time-range picker crashed with `TypeError: (0, m.isDev) is not a function` due to a yarn workspace hoisting mismatch on the AI branch. Pins `@splunk/ui-utils` to `^1.14.0` in the SplunkUI workspace resolutions so the picker loads correctly.

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

Hint

Impact-Based Alerting (IBA), a new scoring-driven alerting model now powering TrackMe.

- IBA replaces rigid, hard-coded alerting rules with a flexible and configurable impact scoring system, inspired by security-grade risk-based approaches and adapted to data reliability and observability.
- With IBA, TrackMe now: - Aggregates anomaly impact over time instead of reacting to isolated signals - Allows fine-grained tuning of how different anomalies contribute to entity health - Provides full trans-

parency with detailed score breakdowns and traceable score events - Significantly reduces alert noise while preserving operational confidence

- The transition is seamless for existing users, requires no manual intervention, and is fully configurable through the UI.

11.1.27 Version 2.3.24 - build 1779386252 (21/05/2026)

Hint

TrackMe 2.3.24 — Stabilization: logging hygiene, restored DHM per-host blocklists, FLX self-heal & Outliers continuity

- **Logging hygiene — schema-upgrade visibility restored**: a long-standing root-logger hijack (since 2.3.22) silently redirected the Health Tracker's schema-upgrade logs into the wrong log file. Health Tracker activity is back where operators expect it, and six previously-unmapped custom-command log files now show up under their proper `trackme:custom_commands:*` sourcetypes.
- **DHM per-host blocklists — restored in React UI**: the per-host index / sourcetype blocklist editor lost during the 2.3 React migration is back, with wildcard matching, suggestion-driven Multiselects, and shortcut buttons from the DHM Inspect modal.
- **FLX entities — self-heal for pre-2.3.12 upgrades**: entities upgraded from <2.3.12 carried stale top-level keys that broke the `splk_hosts_tracking` and `cribl_edge_fleet_metrics` templates. Auto-heals on the next tracker cycle — no migration needed.
- **FLX Outliers continuity on inactive entities**: the inactive tracker now emits zero-value KPI metrics for inactive entities so the Outliers chart keeps a continuous timeline through inactivity windows.
- **FLX thresholds UX**: shared Configuration Guide across the FLX / FQM / WLK threshold modals. The FLX add form gains a Variable schedule switch so a variable threshold can be created in one step.
- **DSM merged-mode (``:@all``) remediation now lands**: the sourcetype-cap safeguard no longer silently drops new `<index>:@all` aggregates, so the documented *blocklist + re-cover with one aggregate* remediation path works again.
- **Performance — initial threshold seeding**: FLX / FQM / WLK trackers' first-run seeding is now batched into a single round-trip — typical 10-100× speed-up on tenants with hundreds of entities.
- **AI Provider — discover models on Splunk Hosted**: the *Discover models* button in *Manage AI Providers* now correctly populates the model selector for the Splunk Hosted (SLIM API) provider.
- **Security & dependencies**: bumps for `ws` and `webpack-dev-server` (security advisories) plus a bundled Dependabot update.

- SHA256: 032852076bcfbda5f7157af439cf04551f5c649b382c5a7a90d65892eb52b3ce

Table 71: New Features

Issue Number	Description	Details
#1692 / #1723 / #1726	feature - DHM per-host index / sourcetype blocklists — restored in React UI with wildcards, Multiselect and immediate apply	Restores the DHM-only per-host blocklist editor that was lost during the 2.3 React migration. The editor is reachable from the entity *Actions → Modify* menu, from the bulk-edit *Per-host Blocklists* category, and from two new shortcut buttons (*Manage blocklists*, *Reset*) on the DHM Inspect modal footer (power+ users only). The matcher now supports wildcards (`fnmatch`) so patterns like `bad_*`, `*-temp` or `nginx:*:error` fan out across whole index / sourcetype families — plain values continue to match literally, so existing blocklists keep working with no migration. The Multiselect inputs are seeded from the host's actual indexes and sourcetypes for one-click selection of known values; free-form patterns can still be typed for wildcards. **Blocklist saves now apply immediately**: matching combos are removed from the stored per-host JSON dict at save time (and the derived `data_index` / `data_sourcetype` are re-derived from the remaining combos), so the *Inspect* view reflects the operator's intent right after save instead of lagging by up to one tracker cycle. Aggregates (state, score, lag) reconverge naturally on the next cycle.
#1677	feature - FLX thresholds — Configuration Guide and inline variable-schedule mode	The FLX, FQM and WLK threshold modals now share a richer **Configuration Guide** with three sub-sections — *How thresholds work*, *Field reference* and (FLX only) *Variable thresholds* — plus an inline impact-score scale showing the green / orange / red bands. The FLX add form gains a **Variable schedule** switch: when on, the primary button relabels to *Configure schedule…* and opens the time-slot editor pre-filled with the form's value / operator / score, so a variable threshold can be created in a single round-trip rather than creating a static threshold and then editing it. The add form is locked while the schedule editor is open so user input between clicks cannot be silently lost.
#1681	feature - FLX — Outliers continuity on inactive entities via zero-value KPI emission	When an FLX use case's upstream SPL stops returning rows for a known entity, TrackMe correctly flips it to red after `max_sec_inactive` but used to stop writing KPI metric points entirely. The native DensityFunction Outliers engine reads via `mstats` and cannot distinguish *no data* from *zero*, so the Outliers chart silently lost coverage during inactivity windows. The inactive tracker now extends each inactivity-cycle write to emit a `0` for every numeric KPI on the entity's last-seen metrics dict, keeping the chart continuous. Default-on for new trackers and inherited by existing trackers without a KV migration; toggleable in step 4 of the FLX use-case wizard.

Table 72: Enhancements & Changes

Issue Number	Description	Details
#1688	performance - Batch-save initial threshold / drilldown / default-metric seeding on FLX / FQM / WLK tracker first execution	The first-execution seeding paths on FLX, FQM and WLK trackers (initial threshold records, drilldown records, default-metric records) used per-record query+insert loops, producing O(N) round-trips when seeding tenants with hundreds of entities. The five affected code paths now use a single broad preload plus a single `batch_update_worker` call (chunked at 500, threaded, with retries). Typical speed-up: 10–100× on the tracker first-execution hot path. Single-entity UI edit endpoints are unchanged. The default-metrics update path uses **insert-first** ordering: new records are written before existing rows are deleted, so a KV-store failure on the insert side preserves the existing default-metric set instead of leaving the tracker empty.
#1699 / #1700 / #1709	change - Security & dependency bumps	Two security-relevant Dependabot bumps and one bundled update. `ws` 8.20.0 → 8.20.1 closes an uninitialized-memory disclosure in `websocket.close()` when called with a `TypedArray` as the `reason` argument. `webpack-dev-server` 5.2.3 → 5.2.4 sets the `Cross-Origin-Resource-Policy` header to prevent source-code theft over HTTP (dev-dependency only — does not ship in the app artefact). A bundled Dependabot update covers `@splunk/themes`, `@splunk/react-time-range`, `@splunk/dashboard-utils`, `requests`, plus a few dev-only and CI references.

Table 73: Bug Fixes

Issue Number	Description	Details
#1711 / #1714 / #1717	bug - Logging hygiene — Health Tracker schema-upgrade activity silenced from 2.3.22 onward	Closes a long-standing root-logger-hijack chain that silently redirected the Health Tracker's schema-upgrade slow-path logs into the wrong log file from 2.3.22 onward — the migration itself ran correctly, but operator-facing activity logs disappeared (~hundreds of `task=schema_upgrade` events per tenant per cycle dropped to two). Every REST handler had a module-load block that overwrote whatever root logger the calling custom command had configured; that block is removed, and a residual transitive-import path is closed by extracting three pure helpers from a custom-command module into a dedicated library. Six previously-unmapped custom-command log files (`trackmeapiautodocs`, `trackmesplkfqminactiveinspector`, `trackmesplkvariabledelay`, `trackmesplkvariabledelayreview`, `trackmefit`, `trackmeapply`) now have `trackme:custom_commands:*` sourcetype mappings in `props.conf`, so they show up where dashboards and SPL expect them. CI regression guards prevent the whole class from coming back.
#1687 / #1698	bug - FLX entities — self-heal for pre-2.3.12 upgrades and safer template placeholder	FLX entities upgraded from a pre-2.3.12 install carried stale top-level fields (`object_description`, `status_description`, `status_description_short`, `metrics`) alongside the tracker-keyed wrapper introduced in 2.3.12. The two shipped templates (`splk_hosts_tracking` and `cribl_edge_fleet_metrics`) only knew about the legacy flat layout and surfaced inconsistent values on those entities. The merge path now strips the legacy flat keys defensively on every cycle, so every pre-upgrade entity self-heals on its next tracker run — no schema migration required. The two shipped templates are made tracker-keyed-aware via a `coalesce` compatibility shim, and the internal substitution token used by these templates is renamed to a distinctive `__flx_tracker_placeholder__` so it can no longer collide with user-authored SPL that happens to contain the literal substring `mytracker`.
#1675	bug - Manage AI Providers — *Discover models* leaves the selector empty on Splunk Hosted	Clicking *Discover models* on a Splunk Hosted (SLIM API) provider succeeded against the backend — the underlying REST call returned 200 with a non-empty model list — but the model selector dropdown stayed empty and the *N model(s) discovered* caption never appeared. The frontend mapper was looking for `id` / `name` fields, while the SLIM API returns `{ model_name, model_id }`. The mapper now reads `model_name` first (with `model_id` / `id` / `name` as ordered fallbacks), so the selector populates correctly and the success caption shows the discovered count.
#1690	bug - DSM — sourcetype-cap safeguard silently dropped `:@all` merged-mode aggregates	The DSM sourcetype-explosion safeguard silently dropped new merged-mode `<index>:@all` entities once the cap was hit for the underlying index, which blocked the documented remediation path (blocklist the over-cap individual sourcetypes and re-cover with one `:@all` aggregate). The safeguard's counter now skips `data_sourcetype == "@all"` records both when building per-index counts and at the per-record cap check, mirroring the existing `data_index == "lookups"` exemption. Detection of the cap condition itself is unchanged — only the silent-drop enforcement learns to let the merged aggregate through.

11.1.28 Version 2.3.23 - build 1779189227 (19/05/2026)

Hint

TrackMe 2.3.23 — Lookups monitoring, Backup & Restore v3 hardening, SLA correctness and DHM per-host extras

- **Lookups monitoring (DSM)**: new `lookups` search mode in the Hybrid Trackers wizard. CSV files and KVstore collections become first-class DSM entities, with delay-based alerting and volume tracking. Supersedes the previous Elastic-source approach. Requires the dedicated `TA-trackme-lookupmonitor` add-on. See *Lookup monitoring*.
- **Backup & Restore v3 — hardening at scale**: consolidated fixes after SHC field testing of 2.3.22. Stuck SHC-local restores, mis-classified imported archives, parallel schema-upgrade backup storms, disruptive UI delete on a per-archive basis, and host-identifier drift are all fixed; orphan sidecars and run manifests are cleaned up by the retention sweep; the download selector becomes a cascading **run → archive** picker.
- **SLA correctness — orange no longer counts as breach**: SLA compliance and `percent_sla` are now computed against time **not in critical (red)** state. Warning (orange) is treated as compliant. The same red-only semantics apply to the Tenant Home *"in alert"* KPI tiles, so tile counts, click-filtered tables and *Open in Search* now agree on a single definition.
- **DHM per-host extras**: new `breakby_extra_fields` option to extend the per-host combo grain beyond `(index, sourcetype)` with extra metadata fields (e.g. `source`). Useful when a host produces multiple distinct event streams.
- **Variable / adaptive delay — honour existing slots**: auto-compute and adaptive delay now refresh thresholds **inside** an entity's existing slot layout instead of regenerating it. Adaptive delay can also heal variable-delay entities automatically.
- **Two new FLX use cases (Splunk infrastructure)**: `splk_bucket_health_per_index` (per-index bucket health via `dbinspect`) and `splk_parsing_issues_per_component` (Splunk indexing-time parsing failures from the three pipeline components). Both ship with default dynamic thresholds and one-click drilldowns.
- **UX polish**: AI Assistant empty state links directly to the in-app provider configuration; Blocklist views gain inline search and bulk actions; lag-monitoring modals show a banner when adaptive delay last refreshed the threshold; the AI Assistant icon now appears on the *Events format recognition* child modals.
- **Various fixes & hygiene**: FQM wizard *Benchmark sampling mode* honours the sampling ratio on LOCAL tenants; *Ops → Parsing issues* and *Data Parsing Quality* panels work again on newer Splunk versions; two high-volume INFO log lines demoted to DEBUG; forward-compat regex cleanup across the licensing handler, the OOTB regex library and the URL validator.

- SHA256: 0a81dde6580e1fe631f57dbedce32aa08c8cc1b1061831be34444b2851189187

Table 74: New Features

Issue Number	Description	Details
#1670	feature - Lookups monitoring as a first-class DSM entity type	A new `lookups` search mode in the DSM Hybrid Trackers wizard discovers Splunk lookups — CSV files (including `.csv.gz`) and KVstore collections — matching an app-namespace glob and a name regex, and turns each one into a standard DSM entity. For CSV files the tracker reads the real filesystem mtime (not the unreliable knowledge-object mtime exposed by Splunk REST) and counts rows efficiently. For KVstore collections it resolves the modification time from an operator-supplied, preference-ordered candidate-fields list (since splunkd does not auto-maintain a `_time` field). The primary alerting signal is the delay since last modification (e.g. `data_max_delay_allowed=86400` ⇒ *must be updated at least once every 24 hours*); volume is tracked via the row / document count. Supersedes the previous Elastic-source workaround. Requires the dedicated TA-trackme-lookupmonitor add-on on every search head where lookups are monitored. See *Lookup monitoring*.
#1584	feature - DHM `breakby_extra_fields` — extend per-host combo grain	DHM hybrid trackers gain an optional `breakby_extra_fields` list that appends additional per-host metadata dimensions (e.g. `source`) to the existing `(index, sourcetype)` combo grain. Useful when a single host produces multiple distinct event streams that need to be monitored independently — operators get one entity per `(host, index, sourcetype, source)` combo, each with its own data-quality and latency story, instead of one combined entity whose latency reflects the loudest stream. Per-host combo cardinality is capped at 100 (least-recently-seen evicted first) so a misbehaving extras field cannot explode the entity count. Trackers created without extras are byte-identical to 2.3.22, so existing trackers are unaffected.
#1610 / #1612	feature - Variable delay & adaptive delay — honour existing slot layout	The **Variable delay threshold** modal auto-compute previously regenerated the slot layout from scratch every time, replacing hand-crafted layouts (business-hours / nights / weekends, tenant-customised templates, …) with anonymous `auto_slot_N` slots. A new strategy selector exposes **Honour existing slots** (new default when slots exist) and **Generate new slots** (the original behaviour). In *Honour* mode, each slot's name, day coverage and hour coverage is preserved; only the per-slot `max_delay_allowed` and the default fallback are refreshed from the historical percentile query. The adaptive-delay backend learns the same semantics: variable-delay entities are no longer excluded from automatic threshold healing, so a variable-delay entity flipping red on `delay_threshold_breached` is now healed automatically instead of requiring manual intervention.
#1572 / #1666	feature - Two new FLX prebuilt use cases — Splunk bucket health & parsing issues	Two new ready-to-use Splunk Flex Objects use cases under the `splunk_infrastructure` category, both shipping with default dynamic thresholds and one-click drilldowns to the offending logs. **(1)** `splk_bucket_health_per_index` monitors per-index bucket health via `| dbinspect` and surfaces prematurely-rolled (small) buckets and quarantined buckets — the canonical symptoms of timestamp-parsing failures where events end up with incorrect `_time` values and Splunk auto-quarantines the resulting buckets. One entity per Splunk index; alert fires when more than 1 % of an index's buckets are quarantined. **(2)** `splk_parsing_issues_per_component` tracks Splunk indexing-time parsing failures across the three pipeline components — `LineBreakingProcessor`, `AggregatorMiningProcessor`, `DateParserVerbose` — by ingesting their WARN / ERROR events from `splunkd.log`

Table 75: Enhancements & Changes

Issue Number	Description	Details
#1581	enhancement - AI Assistant — broader provider examples and direct link to in-app configuration	On a fresh install with no AI provider configured, the AI Assistant empty state used to mention only one or two example providers and offered no actionable path to set one up. The empty state now lists a broader, more representative set of provider examples (Anthropic, OpenAI, Azure OpenAI, Splunk Hosted, …) and includes a direct link to the in-app *Manage AI Providers* page, so a fresh install is no longer a dead end.
#1586	enhancement - Blocklist views — inline search bar with bulk-action support on the filtered subset	Blocklist management views gain an inline search bar so operators can narrow down on deployments with hundreds of entries. Bulk actions (e.g. remove from blocklist) now operate on the filtered subset, so an operator can search for a hostname pattern and remove all matching entries in one round-trip rather than scrolling and clicking individually.
#1660	enhancement - Lag monitoring & Variable delay modals — banner when the threshold was last refreshed by adaptive delay	The Lag monitoring policy modal (DSM, DHM) and the Variable delay threshold modal show a new info banner whenever the entity's delay threshold was last refreshed by the adaptive-delay backend. Operators no longer have to dive into the audit index to tell whether the current threshold was set manually or bumped overnight by adaptive delay. The banner clears immediately when the operator manually overrides the threshold, so it stays honest. Suppressed in bulk-mode modals.
#1552 / #1603	change - Regex hygiene — forward-compatibility cleanup	Forward-compatibility cleanup of regex patterns across the licensing handler, the out-of-the-box data-sampling regex library (37 patterns audited, 25 updated), and the `trackme` search-command URL validator. Removes backslash-escapes on non-metacharacters (`\<`, `\>`, `\/`, `\,`, `\-`, `\:`, `\"`) that work as silent no-ops today but are scheduled to become `re.error` in a future Python release. No behavioural change in the current Python version — every cleaned-up pattern is match-equivalent to its previous form.
#1619 / #1652	change - Logging hygiene — demote two high-volume INFO traces to DEBUG	Two hot-path log lines that produced large amounts of low-value INFO traffic are now logged at DEBUG, so operators running at the default `loglevel = INFO` see less noise. `trackmeload()` per-tenant JSON dumps (fired on every Virtual Tenants / Tenant Home render and every AI Assistant *describe* context build) and `get_full_kv_collection()` entry-point traces (called from REST handlers, scheduled searches, trackers, alert actions and the backup pipeline) are now silent at INFO. Message strings are preserved, so existing operator filters work again at DEBUG.

Table 76: Bug Fixes

Issue Number	Description	Details
#1564	bug - Backup & Restore v3 — consolidated hardening for SHC and at-scale deployments	Consolidated fixes for the v3 backup & restore subsystem after SHC field testing of 2.3.22. **(1)** SHC-local async restores no longer leave restore jobs stuck in a *running* state — the job now reliably reaches a terminal status regardless of how Splunk recycles its REST handler subprocesses. **(2)** Archives exported / imported between TrackMe instances are no longer silently classified as legacy on the target; the 3.0.0 metadata is preserved through registration and imported runs group correctly in the UI. **(3)** The schema-upgrade safety backup is now serialised through a per-cluster lock, so a release upgrade fires exactly one safety backup instead of N parallel ones racing each other. **(4)** The Backup & Restore UI no longer blanks the archive table on every per-archive delete and is SHC-aware, so every peer sees every archive regardless of which peer produced it. **(5)** A host with FQDN ≠ short hostname no longer appears under two different identifiers in the archive list. **(+)** The download selector becomes a cascading **run → archive** picker with an inline filter; the retention sweep now cleans up orphan sidecars and run manifests alongside the archive itself.
#1547	bug - SLA compliance — warning (orange) was treated as breach time	SLA compliance and `percent_sla` were previously computed as time in green / blue versus the rest, which meant warning (orange) reduced SLA compliance the same way as critical (red). SLA is now computed as **time not in critical (red)** divided by total monitored time, so any non-red state — orange, blue, green, unknown — counts as compliant. Applied consistently across the SLA macros, the SLA dashboard, and the SLA flip drilldown, so UI, saved searches and API consumers all agree.
#1659	bug - Tenant Home — *"in alert"* KPI tiles, click-filtered table and *Open in Search* now agree on red-only semantics	The three Tenant Home *"in alert"* KPI tiles on every component tab (ANY / HIGH / CRITICAL PRIORITY) had a visible inconsistency between the tile count and the click-filtered table: HIGH and CRITICAL tile counts were red-only on the backend but the click-filtered tables included orange, while the ANY PRIORITY tile counted red + orange on both sides. All three tiles, their click-filtered tables, the *Open in Search* SPL and the toast wording are now red-only and agree on a single count. The donut chart *"by state and priority"* view keeps orange as its own slice (it is a state breakdown, not an alert filter). Functionally paired with the SLA orange fix above.
#1661	bug - FQM hybrid-tracker wizard — *Benchmark sampling mode* ignored the selected sampling ratio on LOCAL tenants	In the FQM hybrid-tracker wizard step 7 *Collect job strategy*, the *Benchmark sampling mode* button was dispatching the same un-sampled SPL as *Head mode* on LOCAL-account tenants, ignoring the configured sampling ratio. Both panels returned identical event counts and the *Open in Search* link showed *"No Event Sampling"* — actively misleading operators sizing the collect strategy on high-volume datasets. The wizard now passes the sampling ratio as a job-dispatch parameter on the benchmark and forwards it to *Open in Search*, so both surfaces honour the configured ratio. The tracker created in step 9 was always unaffected.
#1663	bug - Events format recognition — AI Assistant icon missing from the custom-rules list and create-rule wizard	The AI Assistant icon was missing from two child modals of the *Data sampling & events format recognition* feature — the custom regex rules list and the 7-step create-rule wizard — so operators lost access to the assistant mid-flow. The icon now appears on both child modals when `enable_ai_assistant` is on, matching the icon coverage rolled out across every other Tenant Home modal in 2.3.22. The contextual *Context:* badge inside the panel also reflects the active screen instead of a generic label.
#1668	bug - Ops Parsing issues & Data Parsing Quality	Two TrackMe macros — `trackme_rex_dateparserverbose` and `trackme_data_quality_parse` — used a regex

11.1.29 Version 2.3.22 - build 1778523370 (11/05/2026)

Hint

TrackMe 2.3.22 — Backup & Restore v3, AI Provider UI, unified AI Assistant UX & Audit AI Assistant

- **Backup & Restore v3**: new multi-archive format — one independent archive per tenant plus one global, per-archive isolation, SHC-aware, with a redesigned restore UI.
- **AI Provider management UI**: dedicated in-product page replaces the UCC tab, with provider-aware forms and on-demand connectivity tests.
- **AI Assistant — unified UX**: compact side-drawer everywhere, horizontal resize, chat history & provider selection persist per page, icon added to all remaining modals and Virtual Tenants admin screens.
- **Audit AI Assistant**: new admin dashboard with per-tenant KPIs, breakdowns and recent-runs / errors / skip-reasons drilldowns; activity also indexed for long-term reporting.
- **In-page tenant configuration editor**: grouped, searchable editor for per-tenant settings, reachable from the 3-dot menu — no more UCC tab round-trip.
- **Per-tenant username allowlist**: server-enforced visibility list to keep under-development tenants hidden from the wider org.
- **Hide-tenant safety net**: confirmation modal with explicit recovery instructions.
- **DHM 'merged' break-by mode** and **DSM/DHM 'none' tstats span**: two tracker performance options for large estates.
- **Bulk edit — Disruption Queue**: apply the same grace period to many entities in one round-trip.
- **Inline search everywhere**: long-list dropdowns and multiselects across React views now support type-to-filter; lookup policy modals default to Wildcard match.
- **Critical bug fixes**: AI status report in stateful alerts, tenant SV cache freshness, phantom KV records in bulk-edit, SOAR failover credential corruption, Anthropic Opus 4+ connectivity, several FQM wizard inputs, Outliers simulation time range.
- **Audit coverage hardening**: `update_comment` and audit events now consistent across licensing, policy-apply, configuration, maintenance and component-power endpoints.
- **Logging hygiene**: noisy INFO perf logs demoted, cross-handler log leaks fixed, `SyntaxWarning` lines in `splunkd.log` cleaned up.
- **Security & dependencies**: patched `@babel/plugin-transform-modules-systemjs` and `fast-uri` (both high) plus a Dependabot bundle.

- SHA256: ae138f42b2f222a41e3cd46b78a2499baf38305d71af230ec2cd4ff3f144613f

Table 77: New Features

Issue Number	Description	Details
#1531	feature - Backup & Restore v3 — multi-archive format, SHC support, redesigned UI	The Backup & Restore subsystem has been redesigned around a new `archive_schema_version=3.0.0` format that decomposes each backup run into **N+1 independent archives** — one self-contained `.tar.zst` per enabled tenant plus one global archive — together with a run-level `manifest.json` listing every archive with its SHA-256, size, and record counts. Per-archive isolation means a failure in one archive no longer blocks the others: each archive is independently created, validated, and restored. A new run-aware restore UI exposes three restore modes — **single archive** (one tenant or just the global side), **whole run** (every archive in the run, in dependency order), and **legacy flat** (for pre-3.0.0 archives still on disk). Backup runs are SHC-aware: each peer delegates its share of work and tracks per-peer outcomes in the run manifest. On restore, missing `vtenant_account` records are auto-recreated from the archive's own embedded tenant JSON, so a deleted tenant can be rehydrated end-to-end from its archive alone. A new Configuration Guardian check (`backup_run_incomplete`) surfaces per-tenant DR degradation — for example, when one tenant archive failed to produce while the rest of the run succeeded — and self-heals on the next complete run. The new docs section *Operations → Backup & Restore* explains the v3 layout, restore modes, SHC behaviour, and Guardian semantics in detail.
#1272	feature - Dedicated TrackMe AI Provider management UI	AI LLM provider accounts (Anthropic, OpenAI, Azure OpenAI, Splunk Hosted, …) are now managed from a new in-product page — **Manage AI Providers** — reachable from the TrackMe navigation. The page replaces the Splunk UCC `Configuration → AI Provider` tab as the recommended entry point. Providers are listed in a table with name, type, model, enabled flag, and on-demand connectivity test status. The create / edit form is provider-aware: only the fields relevant to the selected provider type are shown (no more Azure-only fields surfacing when the user just wants to configure Anthropic), help text is targeted, and validation is inline. The existing UCC tab remains in place as a power-user fallback — both surfaces share the same backend (`/servicesNS/nobody/trackme/trackme_ai_provider`), so existing configurations are picked up unchanged.
#1285	feature - Audit AI Assistant — admin dashboard for AI Assistant activity	A new admin-grade React page — **Review AI Assistant activity** (nav: *Artificial Intelligence → Review AI Assistant activity*) — surfaces, per tenant and over a configurable time range, total runs, success / error / skipped rates, total tokens, average duration, and active-tenants count. Breakdown panels split activity by kind (interactive chat vs. stateful-alert status report), by status, by context (which page launched the chat), and by provider · model. Drill-in tables list recent runs (with full per-event detail — kind, context, status, tenant, object, provider, model, user, duration, tokens), recent errors only, and status-report skip-reasons (with `last_seen` so admins can spot repeating configuration gaps). Every panel exposes a 3-dot *Open in Search* deep-link to the underlying SPL. AI Assistant activity is also written to the audit index as `sourcetype=trackme:audit:ai_assistant`, so it remains available for long-term reporting and external SIEM forwarding.
#1305	feature - Non-disruptive AI Assistant everywhere (compact side-drawer mode)	Until now, the AI Assistant opened in two different modes depending on the entry point: a disruptive full-screen view when triggered from a page-level menu (Tenant Home, Virtual Tenants, Maintenance Mode, …) and the more recent non-disruptive side-drawer when triggered from wizards. The compact side-drawer is now the **only** mode — every entry point

Table 78: Enhancements & Changes

Issue Number	Description	Details
#1526	enhancement - "Hide this tenant" — confirmation modal with explicit undo guidance	Clicking *Personal preferences → Hide this tenant* on a Virtual Tenant card (or in the Tenant Overview modal's actions menu) used to hide the tenant immediately with no confirmation and no explanation, which led to accidental hides being hard to recover from. A small confirmation modal now appears, states clearly that this is a per-user cookie preference (not visible to other users), and explains the exact recovery path: enable *Settings → Virtual Tenants visibility → True*, then click *Personal preferences → Unhide this tenant* on the tenant's kebab menu. The unhide flow is unchanged.
#1294	enhancement - AI Assistant side-drawer — horizontal resize	The non-disruptive AI Assistant side-drawer used to render at a fixed `460 px` width, which felt cramped for long-form answers or wide content. Users can now drag the **left edge** of the panel to widen or narrow it, with sensible bounds (minimum ~360 px so the input and Send button still fit, maximum ~80 % of the viewport so it never fully obscures the underlying screen). Hovering the edge shows an `ew-resize` cursor with a subtle handle indicator. Resize is session-only — no need to persist across reloads.
#1311 / #1322	enhancement - AI Assistant icon in modal headers — no more gap, no more tooltip flash	Two small but visible UX issues in the AI Assistant icon when injected into Splunk `Modal` headers (Tenant Home management modals, Virtual Tenants admin modals) are fixed. The visible gap between the AI icon and the × close button is gone (the icon now sits as a direct flex sibling of the close button instead of being wrapped in its own full-width container), and the *Open AI Assistant* tooltip no longer flashes when a modal opens (focus-triggered opens are now ignored — hover and keyboard activation still work normally).
#1299	enhancement - Inline search on index dropdowns in Virtual Tenants creation wizards	The four index dropdowns rendered in the *Indexes & RBAC* step of the Virtual Tenants creation wizard (Summary, Audit, Metric, Notable) — also reused by the DSM, DHM, and MHM feeds wizards — now expose an inline search box, so users on Splunk deployments with many indexes can type-to-filter instead of scrolling. Built on the existing Splunk UI `filter` prop, no new behaviour to learn.
#1377	enhancement - Inline search on bulk edit Category and Action dropdowns	The Tenant Home **Bulk edit entities** modal's *Category* (~16 options including the new *Disruption Queue*) and *Action* (dynamic per category) selects now support type-to-filter. Existing icons, placeholders, and selection behaviour are unchanged.
#1380	enhancement - Inline search across long-list dropdowns and multiselects	Follow-up sweep that adds the Splunk UI `filter` prop to 18 additional `Multiselect` and `Select` instances across the codebase — including DSM tags, FQM metrics / entities, FLX default metric / metrics / entities, hosts pickers (ops parsing, ops queues), virtual-tenant pickers (Clear Tenant Status, Maintenance Mode, Maintenance KDB), admin / power / user role pickers (Tenant RBAC), and DSM / DHM / MHM index pattern pickers in the wizard. Every site listed is an API-driven or KV-driven dynamic list where the option count can grow large.
#1457	enhancement - Default lookup match mode to "Wildcard" in policy creation modals	In the three Tenant Home lookup-based policy wizards (Priority, SLA, Tags), the *Match mode* dropdown in *Step 2 — Field mappings* now defaults to **Wildcard (supports *, ?)** instead of **Exact (case-insensitive)**. In practice, lookup-based policies mostly map CMDB-style fields (index, hostname, sourcetype) where wildcard matching is what most users actually pick. Editing an existing policy is unaffected — the previously saved match mode is loaded as before.
#1492	enhancement - Bulk edit — clarify when permanent deletion is required	Users frequently confuse *temporary* and *permanent* deletion in the Tenant Home **Bulk edit entities** modal (category *Enablement and Deletion*). A small inline informational banner now appears when the user selects *Delete Permanently*, explaining

Table 79: Bug Fixes

Issue Number	Description	Details
#1284	bug - AI status report silently fails in stateful alert action	When the `trackme_stateful_alert` modular alert action tried to generate the AI-driven status report for an entity, the entity context could not be built and the alert helper logged *"Entity context could not be built, skipping"*. The email was then sent without the AI section, so the feature was effectively dead in stateful alerts. Two causes are fixed: (1) the alert-helper's request-info adapter was missing the `server_rest_uri` and `system_authtoken` attributes that `build_entity_context()` needs, raising silently-swallowed `AttributeError` exceptions; (2) the alert helper was looking up entities by raw `object` string, which is fragile across components — the lookup now uses `object_id`, matching the canonical key used everywhere else. AI status reports are now produced end-to-end in stateful alerts.
#1350	bug - Score-mutation endpoints did not refresh tenant component summary cache	Three score-mutation REST endpoints — *Set false positive*, *Manual influence scoring* (both on entity records), and *Set false positive* on outlier scores — wrote the score event and score cache entry but did not refresh the per-component `extended_stats` cache that drives the Single Value cards on the Tenant Home page. As a result, the entity table updated immediately (it goes through the live decision maker) but the Single Value cards above it lagged behind until either the next scheduled tracker fired or the 15-minute staleness fallback kicked in. All three endpoints now trigger a tenant-component summary refresh on success, so the SVs and the entity table stay in sync.
#1370	bug - Inject expected entities admin handler did not refresh tenant component summary cache	Same family as #1350. The `post_inject_execute` endpoint that injects expected entities (DSM/DHM) into a tenant's catalog directly changes the cached `extended_stats` blob driving the Tenant Home Single Value cards, but did not call the cache-refresh helper. It does now (gated on `entities_created > 0` to avoid no-op refreshes). The companion schedule-management endpoints did not need the change because they only manipulate schedules — when the schedule later runs, it calls back into `inject_execute` which now refreshes correctly.
#1436	bug - Bulk-edit endpoints could silently insert phantom KV records	The shared bulk-edit helper `generic_batch_update` would silently create a phantom KV record for any value in the caller's `keys_list` that did not match an existing record `_key`. The phantom row contained only `_key`, the caller's update fields, and `mtime` — no `object` field — which then corrupted the entity collection. Any downstream consumer that reads `entity_record["object"]` (priority / SLA / tags policy trackers, the decision maker, scheduled trackers) crashed with `KeyError: 'object'`, returning HTTP 500 and degrading the affected trackers. The create branch is removed (no current caller of the helper legitimately needs it) and a defensive guard refuses to create entity rows without an `object` field, so this class of corruption cannot reoccur.
#1454	bug - SOAR Automation Broker failover could corrupt asset credentials	The `soar_automation_broker_manage` endpoint (Automation Broker failover) re-POSTs each affected asset's full configuration with a swapped broker ID, after first stripping salted secrets and a static list of known credential field names (`apikey`, `api_key`, `password`, `auth_token`, `client_secret`). When a SOAR app declared its credential field with a name outside that static list — for example `personal_access_token`, `access_token`, `refresh_token`, `bearer_token`, `token`, `secret`, `secret_key`, `private_key`, `passphrase` — and the GET asset response returned the masked secret in a representation that did not start with the `salt:` prefix, the masked placeholder was POSTed

11.1.30 Version 2.3.21 - build 1777411258 (28/04/2026)

Hint

TrackMe 2.3.21 — Hotfix for Tenant Home lag policy modal regression

- **Hotfix — DSM/DHM Lag monitoring policy modal no longer clobbers in-progress edits**: A regression introduced in 2.3.20 made per-entity lag policy overrides effectively unreachable from the UI on Tenant Home. Sliders for delay/latency impact-score weights and text fields for maximal latency / maximal delay reverted to the persisted/inherited values within seconds of editing, so whatever value happened to be reasserted at the moment **Apply lagging policy** was clicked was what got persisted. The latent defect dates back to the modals' creation in November 2025 but only surfaced once the 2.3.20 adaptive auto-refresh feature started re-rendering the parent on a tight cadence while the modal was open. This hotfix is strongly recommended for every customer running 2.3.20.
- **Configuration Guardian — bearer-token expiry now reads the JWT itself**: The `remote_account_token_expiring_soon` check no longer derives token expiry from TrackMe-side scheduling signals (KV `mtime` and `token_rotation_frequency`). It now decodes the bearer token's JWT `exp` claim directly, eliminating false-positive `critical` "Remote account token expired" alerts on accounts whose underlying token still has weeks of validity.
- **Entity tables — label tooltips now surface label descriptions**: Hovering a label chip in the Virtual Tenants and Tenant Home entity tables now shows the configured `label_description` alongside the label name, so the meaning of a label (e.g. *"ML is generating seasonal patterns"*, *"Issue acknowledged, no immediate action needed"*) is visible at glance without opening the Manage Labels modal.

- SHA256: 1d77fc233828d5fa4fe05e9d045fdc03be6184f14edc220a50d36d50bc22f9fb

Table 80: Enhancements & Changes

Issue Number	Description	Details
#1246	enhancement - Show label description on hover in entity tables	Labels rendered in the Labels column of the entity tables on Virtual Tenants and Tenant Home (FLX, DSM, DHM, MHM, FQM, WLK) used to show only the label name in their hover tooltip. Each label can carry a meaningful `label_description` configured via the Manage Labels modal, but that description was not exposed at glance time. Hovering a label chip now surfaces both the label name and its description in the contextual tooltip when a description is set; if none is set, the tooltip falls back to the label name only (current behaviour). The `labels_objects` array materialised on entity records by the decision maker (`dynamic_labels_lookup`) and by the post-assignment shadow patch in `trackme_rest_handler_labels_power.py` now propagates `label_description` alongside the existing `label_id` / `label_name` / `label_color` fields.

Table 81: Bug Fixes

Issue Number	Description	Details
#1250	bug - DSM/DHM Lag monitoring policy modal: form fields revert during edit (regression in 2.3.20)	Fixed a regression in 2.3.20 where the **Lag monitoring policy** modal (DSM and DHM) silently reverted any user input — sliders for delay/latency impact-score weights and text fields for maximal latency / maximal delay — within seconds of editing, making per-entity overrides effectively unreachable from the UI. The "Apply lagging policy" button could still be clicked, but whatever value happened to be reasserted at that moment was what got persisted. Root cause: a `React.useEffect(..., [open, initialValues])` in `EntityModifyLagDsmModal.tsx` and `EntityModifyLagDhmModal.tsx` re-initialised form state every time `initialValues` changed by reference, and the parent (`TenantOverviewTable`) builds `initialValues={{…}}` as an inline object literal — a new reference on every parent render. The 2.3.20 adaptive auto-refresh feature (#1223) re-rendered the parent on a tight cadence while the modal was open, which made the latent defect visible. The dependency array is now `[open]` so initialisation runs only on open→true transitions, with the intentional omission of `initialValues` documented inline. The latent defect was present since the modals were created in November 2025 (so dormant in 2.3.18 / 2.3.19) but only surfaced in 2.3.20.
#1244	bug - Configuration Guardian: bearer-token expiry check fired false-positive `critical` alerts	Fixed the Configuration Guardian `remote_account_token_expiring_soon` check firing `critical` *"Remote account token expired"* alerts against remote accounts whose underlying bearer token was in fact still valid for weeks. The check used to derive `remaining` from two TrackMe-side scheduling signals — `now - kv_trackme_remote_account_token_expiration.mtime` and `token_rotation_frequency_days`. Neither input represents the token's actual expiry: `mtime` is when TrackMe last *rotated* the token (our cadence), and `token_rotation_frequency` is how often we *attempt* rotation, not the JWT lifetime (Splunk-issued tokens default to ≈ 30 days regardless). When a rotation hiccup, a stale duplicate KV record, or a `frequency=7` cadence on a 30-day JWT pushed `now - mtime` close to or past `frequency × 86400`, the check fired `critical` even though the token still had weeks of validity and remote searches kept succeeding. `check_remote_account_token_expiry` in `trackme_libs_guardian.py` now base64url-decodes the JWT payload of the configured bearer token and reads the authoritative `exp` (and `iat`, `aud`) claims directly. Proportional warning / critical thresholds are now derived from the actual token lifetime `exp - iat` (with `rotation_frequency_days × 86400` retained as a fallback when `iat` is missing). Non-JWT tokens (e.g. external IdP credentials) skip cleanly with a documented `unable_to_decode_jwt` outcome and clear any prior expiry alert — actual auth failures remain covered end-to-end by `remote_account_connectivity_degraded`. Alert metadata exposes `token_exp_epoch`, `token_exp_iso`, `token_iat_epoch`, `token_audience`, and a hashed token id (no raw token in audit/UI surfaces). The KV `mtime` remains useful for the dashboard's `time_since_last_rotation` column — it just stops being load-bearing for expiry detection.

11.1.31 Version 2.3.20 - build 1777282739 (27/04/2026)

Hint

TrackMe 2.3.20 — Configuration Guardian P0 wave, AI Assistant everywhere, adaptive Tenant Home & stateful alert performance

- **Configuration Guardian — six P0 checks, severity tiers, audit trail & AI Assistant integration**: A new framework surfaces silent misconfigurations directly in the Virtual Tenants UI. This release ships the full P0 wave of six checks covering the most common "silently degrades TrackMe" conditions: insufficient tenant-owner capabilities, declared tenant indexes that don't exist, remote-account tokens approaching expiry, the per-tenant health tracker not executing, remote-account connectivity failures, AI provider unreachable, and stale backup archives. Alerts come with two severity tiers (warning and critical), structured remediation guidance, and a full audit trail in the `trackme_audit` index. The AI Assistant on Virtual Tenants and Tenant Home is now Guardian-aware: it sees every active alert with parsed metadata and recommended actions, and is steered by a built-in playbook to act as an active guide rather than a passive reporter.
- **AI Assistant — non-disruptive side panel everywhere**: Building on the hybrid tracker wizard integration, the compact AI Assistant side panel is now available across every Tenant Home configuration screen — policies (priority, tags, SLA, metric, monitoring time), lagging classes, blocklists, logical groups, CMDB integration, data sampling, default and variable delay, replica trackers, FLX utilities, outliers, FQM dictionary, ops queues, parsing issues, and more. Users get ambient AI assistance without ever leaving the screen they are working on.
- **Tenant Home — adaptive auto-refresh and preserved navigation state**: The Tenant Home page used to auto-refresh on a hardcoded 5-minute interval, which was either too frequent for small tenants or actively harmful for large ones (a refresh would fire while the user was still reading the previous result). The cadence now scales automatically with the observed table load duration, between a 5-minute floor and a 30-minute ceiling, with a visible *"Next refresh in MM:SS"* countdown. Manual refresh resets the countdown so consecutive refreshes don't double-load. As a separate quality-of-life fix, the group-by dropdown, quick filter dropdown, and group expand/collapse state now survive tab switches within the same tenant page session, matching how the search box already worked.
- **AI providers — enable/disable without deletion**: Administrators can now disable a configured AI provider (Configuration → AI Provider) instead of having to delete it. Disabled providers disappear from the AI Assistant chat selector, are skipped by stateful alert AI status reports, and are excluded from the Guardian `ai_provider_unreachable` probe (the prior alert is also self-cleared). The admin connectivity test endpoint still works against disabled providers so admins can troubleshoot before re-enabling.
- **Stateful alert performance — significant speedups on high-throughput deployments**: Several performance improvements to the stateful alert helper and `trackmestateful` command, with substantial runtime gains on production deployments: `tenant_idx_resolution` is now cached per tenant within a single invocation (saves ~3s per event); chart storage uses `batch_save` instead of one REST call per chart (was ~14-16s per event with ~10 charts, now near-instant); object-state validation can run in-process instead of via a per-entity REST call to `load_component_data` (opt-in, safe fallback); and the `trackmestateful` SPL pre-resolves the active-incident key list in Python instead of running a per-cycle subsearch.
- **Splunk Hosted LLM — auto-resolve model_name to model_id**: The `splunk_hosted` provider transparently accepts both the human-friendly `model_name` (e.g. `gpt-oss-20b`) and the `model_id` expected by the SLIM API. Customers unfamiliar with Splunk's internal model ID convention no longer hit opaque 404s. The mapping is cached.
- **Backup & Restore — gateway-timeout-proof Create backup**: The **Create backup** action now dispatches a Splunk search job instead of keeping a long-lived browser REST connection. Large backup runs on production deployments no longer hit reverse-proxy or load-balancer 502/504s even when the backup itself completes successfully backend-side. The UI also gains copy-to-clipboard buttons for every SPL and cURL snippet in the Documentation & examples modal.
- **Scheduler skip detection — alert-aware**: Scheduler completeness and "skipping searches" detection now captures alert firings in addition to regular tracker searches. The Scheduler Review modal routes its row

actions appropriately per row type.

- **UI polish & consistency**: Slider for *Min green %* in the converging tracker wizard, translated Splunk query in the CMDB lookup modal, near-full-width AI Assistant responses, other group members listed directly in the Logical Groups association panel, accurate **HIGH PRIORITY ENTITIES IN ALERT** header counter (no longer counts disabled tenants/components), and Virtual Tenant wizard index/role dropdowns no longer truncated at 30 entries.
- **Support Diagnostic — robustness for large environments**: The *Generate diag* feature shipped in 2.3.19 is hardened for busy customer environments where a *Global* run could time out before completing. The backend timeout is raised to 60 minutes, the worker emits a heartbeat so long steps don't get misclassified as dead, and the UI poll matches the new ceiling. When something does fail mid-collection, a new *Partial failures* card lists the failed steps and the user still receives a partial archive instead of losing the entire run. The page also now correctly follows the user's Splunk light/dark theme.
- **Various bug fixes**: Notes and labels audit events are now correctly surfaced in the entity **Audit changes** tab; disabled tenant overview no longer loops requests; the Ops Status modal's *Run: Clear Tenant Status* button works again; SLA tracker scheduling follows SLA policies (copy-paste bug); clipboard fallback surfaces real copy failures; variable delay auto-compute works again; MHM lagging class overrides are applied (and the *Manage: lagging classes* menu entry is renamed back); leftover stanza keys no longer pollute splunkd.log on Splunk 9.x.

- SHA256: 933c725146bf6e94900dc1c875fac560472dbfb479f839741a78d660c47d4b27

Table 82: New Features

Issue Number	Description	Details
#1147	feature - Configuration Guardian framework with full P0 wave of checks	Configuration Guardian is a new framework that surfaces silent misconfigurations directly in the Virtual Tenants UI. When a problem is detected, administrators see a toast with a clear title, explanation, and remediation steps — yellow for warnings, red for critical — plus a **Dismiss alert** action to clear it while fixing the root cause. If the condition persists, the alert reappears on the next check cycle. Every state transition (created, updated, cleared, dismissed) is recorded in the `trackme_audit` index for traceability. This release ships the full P0 wave covering six "silently degrades TrackMe" conditions: • **Insufficient tenant_owner capabilities** — the service account behind a Virtual Tenant must hold the Splunk capabilities required by scheduled tracking; without them, tracking operations silently drop writes or skip updates. • **Assigned index does not exist** — every index referenced in a tenant's account configuration is verified to exist on the search head. • **Health tracker not executing** — the per-tenant health tracker must have run within the last 30 minutes; raised as `critical` because most other tracking depends on it. • **Remote-account token expiring soon** — based on the configured rotation frequency; severity escalates from `warning` (approaching expiry) to `critical` (past or imminent expiry). • **Remote-account connectivity degraded** — daily probe of every configured remote account; escalates from `warning` to `critical` after 24 hours of continued failure. • **AI provider unreachable** — daily probe of every enabled AI provider; skipped entirely on installs without AI features. • **Backup archive too old** — compares the most recent backup mtime against the configured backup cadence (warning) or 7 days (critical, fixed). The AI Assistant on Virtual Tenants and Tenant Home is fully Guardian-aware: it sees every active alert with parsed metadata and recommended actions, knows what each check means and how to remediate it, and is steered by a built-in playbook to act as an active guide rather than a passive reporter. A dedicated read endpoint (`GET /trackme/v2/configuration/admin/guardian_alerts`) returns every active alert with parsed metadata, recommended actions, and counts by severity / check type — designed for AI-agent integration.
#1170	feature - AI Assistant side panel extended to all Tenant Home configuration screens	Following the introduction of the non-disruptive AI Assistant side panel in the hybrid tracker wizards (#1166), the same compact panel is now available across every configuration / management screen reachable from Tenant Home. This covers all the policy editors (priority, tags, SLA, metric policies, monitoring time), lagging classes, blocklists, logical groups, impact score, default and variable delay templates, CMDB integration, data sampling, replica tracker creation and management, FLX utilities / drilldown searches / default metrics, outliers, FQM dictionary, ops queues, parsing issues, and permanently deleted entities. Wherever you are configuring TrackMe, an AI icon in the screen header opens the assistant on the right while the configuration screen stays visible and interactive on the left. The panel is visible only when the AI Assistant is enabled system-wide.
#1166	feature - Non-disruptive AI Assistant panel in hybrid	The TrackMe AI Assistant can now be opened directly from any of the six hybrid tracker creation wizards (DSM, DHM, MHM,

Table 83: Enhancements & Changes

Issue Number	Description	Details
#1227	enhancement - Tenant Home: manual refresh resets the adaptive auto-refresh countdown	Follow-up to the adaptive auto-refresh enhancement (#1223). When auto-refresh is enabled, clicking the manual refresh button now resets the countdown to the next automatic refresh — so consecutive manual + automatic refreshes don't double-load the page. If the manual refresh fails or auto-refresh is disabled, the countdown is left untouched.
#1223	enhancement - Tenant Home: adaptive auto-refresh based on observed table load duration	The Tenant Home page used to auto-refresh on a hardcoded 5-minute interval. On large-scale tenants where the main entities table can take 60-120s to load, that meant the next refresh would fire while the user was still reading the previous result; on small tenants it was unnecessarily frequent. The cadence now scales automatically with the most recent observed table load duration on the active tab, between a 5-minute floor and a 30-minute ceiling. A small *"Next refresh in MM:SS · cadence Nm"* label appears under *Last updated* with a tooltip explaining the source. A concurrency guard prevents stacked loads if a refresh is still running when the timer fires (or if the user spam-clicks the manual refresh button). Switching tabs resets the observed duration so a slow DSM reading doesn't penalise a fast FQM tab. The existing auto-refresh on/off toggle (cookie) still controls the whole feature.
#1213	enhancement - Tenant Home: preserve dropdown selections and group expand/collapse across tab navigation	On the Tenant Home page, navigating between tabs (e.g. *Data source monitoring* → *Tracking Alerts* → back) preserved the search input but reset the **group-by** dropdown, the **quick filter** dropdown, and the per-group expand/collapse state. The behaviour was asymmetric and surprising — users naturally expected all the table controls to behave like the search box. All three pieces of state now survive tab navigation within the same tenant page session, exactly like the search input. State is reset only on a full page refresh or when navigating to a different tenant.
#1193	enhancement - AI provider: allow administrators to disable a configured provider	The AI provider configuration screen (Configuration → AI Provider) now exposes an `ai_enabled` toggle so administrators can disable a provider without deleting its configuration. Disabled providers are filtered out of the AI Assistant chat selector, are skipped by stateful alert AI status reports (with a clear log line), and are skipped by the Configuration Guardian `ai_provider_unreachable` check (any pre-existing alert is also self-cleared, so disabling a provider stops surfacing it as broken). The admin connectivity test endpoint deliberately still works against disabled providers — admins can troubleshoot connectivity before re-enabling. Existing provider records are treated as enabled by default, so the upgrade is fully backwards-compatible.
#1168	enhancement - Logical Groups modal shows other members of the current group	When you open the Logical Groups modal on an entity that belongs to a group, the association panel now lists the other members of that group directly — no more scrolling down and visually matching the group name in the overview table. If the entity is the only member, a clear *"This entity is the only member of this group."* message is shown instead. Small but useful quality-of-life improvement for operators investigating grouped entities.
#1164	enhancement - Converging tracker wizard: use Slider for *Min green %*	In the Flex Objects converging tracker creation wizard, *Min green %* is now a slider with a live value readout instead of a plain numeric input. This matches the style already used elsewhere for 0-100 integer fields (like Impact score in the Outliers model modal) for a more consistent and touch-friendly experience. The default value stays at 100.
#1152	enhancement - Scheduler skip detection: alert firings now captured end-to-end	Scheduler completeness metrics, skipping-searches detection, and the Scheduler Review modal used to only cover regular tracker saved searches — alert firings were silently ignored be-

Table 84: Performance

Issue Number	Description	Details
#1210	performance - Stateful alerts: in-process object-state validation (opt-in)	In the stateful alert helper, `validate_object_state_via_rest` used to issue one HTTP call to `load_component_data` per alerting entity to re-validate the entity's state before opening, updating, or closing an incident. On high-throughput deployments where many entities can fire concurrently, this drove significant splunkd pressure and long alert-helper runtimes. An opt-in in-process evaluation path is now available — same scoring logic, just no HTTP / JSON / pagination overhead. Default behaviour is unchanged (the REST path remains the fallback), and any failure of the new path transparently falls back to the REST path so the existing fail-open semantics are preserved. Operators can opt in via a kill switch when they are ready to soak it.
#1206	performance - `trackmestateful`: pre-resolve active-incident key list in Python	The search that feeds the stateful alert pipeline used to pull every `trackme:state` heartbeat in the 10-minute window across every monitored entity, then drop ~99.9% of them in pre-filter. On high-frequency deployments this drove search runtimes well into the hundreds of seconds (max observed >8000s) with significant search-slot queueing. The active-incident key list is now pre-resolved in Python before the SPL is built and injected as a literal predicate, drastically cutting the volume of returned events while preserving the bootstrap path for entities that don't yet have a stateful record. The `trackme_notable` arm is also now scoped to its sourcetype, preventing cross-app eventtype expansion in some Splunk environments.
#1179	performance - Stateful alerts: cache `tenant_idx_resolution` per tenant	In `process_event()` of the stateful alert helper, `trackme_idx_for_tenant()` was called unconditionally inside the event loop — once per event — even though the result only varies per tenant and doesn't change within a single alert action invocation. Production observation: ~3.2-3.4s average per event, p95 up to ~5.3s, max ~13.2s. The result is now cached per tenant within an invocation (matching the pattern already used for ack collections and vtenants), saving roughly ~3s per event on every event after the first one for a given tenant. For a mono-tenant alert with 50 events this saves ~2.5 minutes per invocation.
#1175	performance - Stateful alerts: batch chart storage via `batch_save`	In the stateful alert helper, charts attached to incidents were stored in the chart KV collection one at a time — one HTTP roundtrip per chart inside a loop. With ~10 charts per event this cost ~14-16 seconds on average (p95 ~22s) per event in production. Charts are now persisted in a single `batch_save` call, matching the pattern already used elsewhere in the codebase, bringing this step down to near-instant.

Table 85: Bug Fixes

Issue Number	Description	Details
#1230	bug - Support Diagnostic: timeouts on large environments, partial-success archive, light-theme rendering	On large customer environments (many active tenants, busy `_internal`), running a *Global* support diagnostic could fail with a *"diag creation timed out"* message before completing — the original async-job timeout (15 min on both backend and UI) was simply too short, and a single long step could go minutes without bumping `last_activity`, occasionally letting the cleanup pass mistakenly classify a still-alive worker as orphaned. Several fixes raise the ceiling and make the feature resilient on busy instances: backend job timeout is now 60 min, the worker emits a heartbeat every 30 s while it runs, and the UI poll matches the new ceiling. When something does fail mid-collection, a *Partial failures* card now lists the failed steps with their error messages and the user gets a partial archive (with a clear *info* notification) instead of a thrown-away run. As a side fix, the Support Diagnostic page now correctly follows the user's Splunk theme — light-theme users no longer see a barely-readable dark page.
#1198	bug - Splunk < 10.x: silence `Invalid key: python.required` splunkd warnings	On Splunk < 10.x, splunkd logged `Invalid key` warnings at startup for every TrackMe stanza carrying `python.required` — a Splunk 10.x addition that older Splunk versions don't recognise and reject loudly. The setting is harmless on older Splunk (it is simply ignored), but the noise made log triage harder. `python.required` is now declared in TrackMe's `.conf.spec` files so Splunk treats it as a known key. Behaviour on both Splunk 9.x and 10.x is unchanged.
#1196	bug - Remove leftover `splk_general_cim_cmdb_s` stanza key	A stanza key from the decommissioned CIM monitoring component was left behind in `trackme_settings.conf` and produced an `Invalid key` warning on every conf load. The leftover line is now removed.
#1190	bug - Virtual Tenants header: *HIGH PRIORITY ENTITIES IN ALERT* counter no longer counts disabled tenants/components	On the Virtual Tenants page, the **HIGH PRIORITY ENTITIES IN ALERT** counter at the top of the screen could show a value that did not match the sum of the per-tenant card badges — typically because the header summed in stale priority counts from disabled tenants or disabled components. The header aggregation now skips tenants whose status is `disabled` and only counts components where the per-tenant `enabled` flag is set, matching exactly what the per-tenant cards do.
#1182	bug - Virtual Tenant wizard: index & role dropdowns no longer truncated at 30 entries	In the *Create a new Virtual Tenant* wizard (Step 4 — *Indexes & RBAC*), the Summary / Audit / Metric / Notable index dropdowns and the Admin / Power / Readonly role pickers silently omitted entries when the search head had more than 30 indexes or 30 roles — Splunk's REST defaults to `count=30` rather than unlimited. Both frontend calls now pass `count=0` so all available indexes and roles are returned. Users on search heads with large index catalogues or many custom roles can again select entries beyond the first 30 alphabetically.
#1172	bug - MHM: lagging class overrides applied + screen renamed back to *Manage: lagging classes*	Two related MHM regressions are fixed. First, MHM per-metric-category lagging class overrides were silently never applied — the loader hit a benign exception that fell back to defaults on every cycle, so customer overrides had no effect since the feature was introduced. The loader now reads the collection correctly and overrides take effect again. Second, the Tenant Home kebab menu entry for MHM had been renamed to *Manage: metric policies*, which made the screen undiscoverable for users looking for *"lagging classes"* (and inconsistent with the DSM/DHM equivalents). The entry is renamed back to *Manage: lagging classes* so terminology is uniform across DSM, DHM and MHM.
#1145	bug - Virtual Tenants: disabled tenant overview loops	Opening the Virtual Tenants *Entities Overview* modal on a disabled tenant caused the UI to fire repeated backend requests that

11.1.32 Version 2.3.19 - build 1776258970 (15/04/2026)

Hint

TrackMe 2.3.19 — Entity Labels, CMDB Integration, Variable Delay, Splunk Cloud AI & Support Diagnostics

- **Entity Labels — lifecycle visibility at a glance**: Introducing a lightweight, color-coded labeling system for entities. Labels give teams instant visibility into the lifecycle stage and operational context of every tracked entity — directly in the entity tables, in stateful and notable alert events, and through Virtual Groups that can aggregate entities by label across tenants.
- **CMDB integration improvements**: Stateful and notable alert actions now automatically enrich events with CMDB data at alert time. A new simplified configuration screen replaces the complex UCC-based setup, and CMDB icons now appear correctly in Virtual Groups tables.
- **Entity Notes enhancements**: Notes are now visible as a first-class column in entity tables with a new clone-to-entities action for bulk operations.
- **Variable delay management improvements**: Timezone notice banners in all variable delay and threshold editors, plus fully customizable per-tenant slot templates.
- **Policy tracker scheduling**: SLA, Tags, and Priority policy trackers now default to a 12-hour cadence instead of every 15 minutes, cutting scheduled-search load for these jobs while policy content remains applied; cron remains adjustable per tenant if you need a faster refresh.
- **Support — generate diagnostics**: A self-service **Support - generate diags** experience (nav under API & tooling and Audit & troubleshoot) lets admins produce a timestamped `.tgz` for TrackMe support — entity-scoped or global, async job + polling + secure download, optional tenant anonymisation with a separate mapping for support, and RBAC via `trackmepoweroperations`.
- **AI Assistant — realtime entity context**: Entity-level AI chat now loads the same decision-maker view as `/trackme/v2/describe/entity` (via `load_component_data`), so labels, scores, smart status, and other joined fields match the UI instead of a raw KV snapshot.
- **Splunk Cloud — splunk_hosted LLM (SLIM)**: The Splunk-hosted AI provider now sends the `request_id` header required by the SLIM gateway on Splunk Cloud, resolving HTTP 400 `Request ID not present in header` failures so model discovery, in-app AI Assistant chat, and AI status content in stateful alerts work again for customers using `splunk_hosted`.
- **Various bug fixes and improvements**: SLA ranking corrections, REST handler hardening, Cribl use case updates, multiple Virtual Groups fixes, quieter routine logging from the `trackmestateful` command, KV index transforms for unquoted keys, consistent status messaging when impact score is raised manually without anomalies, and ML outliers simulation with auto-correction disabled when training volume is below the native fit minimum.

- SHA256: f444af4de9043a196ddbd54a65e24cab2cfc70796a7e55b5e36077db9754a3be

Table 86: New Features

Issue Number	Description	Details
#1024	feature - GitHub-style entity labels for lifecycle visibility	TrackMe now supports entity labels — lightweight, color-coded tags that give teams instant visibility into the lifecycle stage and operational context of every tracked entity. Labels are fully managed per tenant and can be freely defined to match your operational vocabulary (e.g. `review-in-progress`, `decommissioning`, `new-onboarding`, `known-noisy`). Once assigned, labels appear directly in the entity tables as colored badges. Labels are also included in stateful and notable alert events (both as a flat `labels` multi-value field and as structured `labels_objects` JSON), enabling downstream routing, filtering, and enrichment in Splunk searches or SOAR playbooks. Virtual Groups can leverage labels to create focused cross-tenant views — for example, aggregating all entities labeled `incident-active` across production tenants.
#1048	feature - Entity notes: table column, shared modal, clone-to-entities	Notes have been elevated from a hidden detail to a first-class column in the entity tables. A new Notes column shows the note count per entity, and clicking it opens a shared notes modal that works consistently across entity overview, table rows, and bulk operations. A new clone-to-entities action lets operators push a note to multiple entities at once — useful for documenting a shared incident, a maintenance window, or an operational decision that affects a group of data sources or hosts.
#1035	feature - CMDB integration in stateful and notable alert actions	Stateful and notable alert actions now automatically perform a CMDB lookup at alert time, enriching the event with ownership, criticality, contact information, and any other fields defined in your CMDB lookup. Alert recipients and downstream automation receive full operational context without additional manual configuration.
#1064	feature - Simplified CMDB integration configuration screen	A new "Manage: CMDB integration" modal is accessible from the tenant-home Features menu for all six component types (DSM, DHM, MHM, FLX, FQM, WLK). It replaces the previous UCC-buried configuration with a guided experience: enable/disable toggle, local or remote account selector (via a new `cmdb_account` field), guided lookup builder with content preview and auto-generated SPL, and a full SPL editor with Edit/Preview toggle.
#1037	feature - Virtual Groups category sub-grouping	Virtual Groups can now be organized into categories for better visual structure in the Virtual Tenants dashboard. Groups sharing the same `group_category` are rendered together, making large deployments with many cross-tenant views much easier to navigate. Combined with the new labels integration, Virtual Groups become a powerful tool for creating dynamic, cross-tenant operational views — for example, a group that tracks all entities labeled `incident-active` across your production tenants.
#1009	feature - Inject expected sources/hosts via one-shot action wizard	A new Inject Expected wizard lets operators declare data sources (DSM) or hosts (DHM) that should exist in a tenant, triggering immediate entity creation rather than waiting for the next discovery cycle. This is particularly useful for onboarding new data feeds where you want proactive alerting from day one, re-creating previously deleted entities, or pre-populating a tenant before a migration. The wizard supports optional recurring operations for use cases where periodic re-injection is needed.
#1054	feature - Variable delay / threshold editors: Splunk server time disclosure banner	Variable delay and variable threshold slots are evaluated by the decision maker using the Splunk server's local clock, which may differ from the browser timezone of the admin configuring them. A live Timezone notice banner now appears in every variable delay and threshold editor, showing both the server clock and the browser clock in real time with the signed offset, so administrators always know which time frame their slot def-

Table 87: Enhancements & Changes

Issue Number	Description	Details
#1041	enhancement - Include labels field in stateful and notable alert events	The `labels` (flat list) and `labels_objects` (JSON) fields are now included in stateful alert events and notable events, enabling downstream routing, filtering, and enrichment based on entity labels.
#1043	enhancement - Tenant-scoped error logging	All tenant-related error responses and log messages now consistently include the `tenant_id`, making multi-tenant troubleshooting considerably easier when diagnosing issues across tenants.
#1069	enhancement - SLA, Tags, and Priority policy trackers: default schedule 15 minutes → 12 hours	For each component (DSM, DHM, MHM, FLX, FQM, WLK), the per-tenant scheduled searches `trackme_{component}_tags_tracker_tenant_{tenant_id}`, `trackme_{component}_priority_tracker_tenant_{tenant_id}`, and `trackme_{component}_sla_tracker_tenant_{tenant_id}` previously ran every 15 minutes (`*/15 * * * *`). New defaults use a 12-hour cadence (`17 */12 * * *`), reflecting that priority, tag, and SLA assignments change infrequently. New tenants receive the updated cron in tracker definitions; existing tenants are migrated via `trackme_schema_upgrade_2319`. This reduces execution count for these trackers by roughly 96% with no change to how policies are evaluated when they run. Administrators who need more frequent application can still edit the cron schedule on the relevant saved searches.
#1071	enhancement - Stateful alert helper: structured performance timing in logs	The modular alert helper `modalert_trackme_stateful_alert_helper.py` now records `run_time` (seconds) in structured `log_info` lines for major initialization steps (request info, Splunk service connect, maintenance and bank-holiday checks, email account and alert-definition retrieval) and for each per-entity phase (tenant index resolution, acknowledgment collection, KV lookups, state validation, metrics catalog, chart search and render, chart KV persistence, AI status report). The pattern mirrors the health tracker so operators can search Splunk for slow steps without ad hoc instrumentation.
#1073	enhancement - `trackmestateful` command: routine per-entity logs moved from INFO to DEBUG	The `trackmestateful` custom command previously logged per-entity `ack_check`, `context_gathering`, `pre_filter`, and routine `filtering` decisions at INFO for every entity, which could produce hundreds of thousands of lines per day per tenant at scale. Those messages now use DEBUG; initialization summaries, completion, errors, and configuration-related lines stay at INFO, cutting noise in `trackme_stateful.log` while preserving detail when debug logging is enabled.

Table 88: Bug Fixes

Issue Number	Description	Details
#1062	bug - SLA classes ranking is inverted	The default SLA classes had inverted ranking — platinum was ranked lowest (1) while gold was ranked highest (3), with thresholds also assigned backwards. New installations now receive correct defaults: platinum = rank 3 with the strictest threshold (4h), gold = rank 2 (24h), silver = rank 1 with the most lenient threshold (48h). SLA class dropdowns are now sorted by rank descending (most important first). Existing installations with customized SLA classes are unaffected.
#1055	bug - REST handlers describe-flag parser truthiness bug	The inline `describe=true` flag parser across 77 REST handler files treated any non-empty string — including the literal `"false"` — as a describe request due to Python's string truthiness. All handlers now use a shared `trackme_parse_describe_flag()` helper that only returns True for boolean `true` or case-insensitive string `"true"`. Normal UI and API flows were not affected (they send either nothing or `{"describe": true}`), but SPL callers explicitly passing `{"describe": "false"}` would have received the describe payload instead of executing the real handler.
#1050	bug - Cribl Logstream pipeline FLX use case broken after dimension rename	The OOTB `cribl_logstream_pipeline` FLX use case broke after Cribl renamed its `pipeline` dimension to `id`. The use case definition has been updated to the new field name.
#1039	bug - Virtual Groups REST handlers missing explicit port	Virtual Groups REST handlers were omitting the explicit port in `client.connect()` calls, causing IPv6 resolution failures on certain platforms. All handlers now pass the explicit port.
#1066	bug - Missing CMDB icon in Virtual Groups entities table	The CMDB icon was missing from the actions column in Virtual Groups entity tables, even when CMDB was enabled on the underlying tenants. Fixed by supporting per-tenant CMDB lookup resolution in the Virtual Groups overview, so each entity's CMDB availability is determined by its source tenant's configuration.
#1082	bug - Manual impact score: empty `status_message` and `anomaly_reason` when no anomalies	Raising impact score via **manually influence scoring** could move an entity to orange or red while leaving `status_message` and `anomaly_reason` empty because no real anomalies were present. Updates across DSM, DHM, MHM, FLX, FQM, and WLK in the decision maker now populate human-readable status text, appropriate anomaly reason (including `score_breached` for `manual_score` where applicable), and `score_definition.components` for UI and downstream consumers (stateful alerts, smart status, notable events) consistent with the manual-score path.
#1075	bug - KV indexed transforms: unquoted `tenant_id` and `object` values not captured	The indexed-field transforms `trackme_indexed_kv_tenant_id` and `trackme_indexed_kv_object` only matched double-quoted values, so keys such as `tenant_id=my_tenant` were missed. `transforms.conf` regexes now treat quotes as optional so both quoted and unquoted forms index correctly.
#1033	bug - REST API autodocs missing endpoints	`trackmeapiautodocs.py` was missing registrations for `virtual_groups`, `labels`, `component/admin`, and `inject_expected` handlers. The REST API Reference page now lists all endpoints correctly.
#1022	bug - Virtual Group modals use group_id instead of entity tenant_id	Virtual Group overview modals were using the Virtual Group's `group_id` as the tenant identifier when opening ACK, CMDB, and Score modals. Fixed to use the underlying entity's real `tenant_id`.
#1017	bug - DHM entity overview pie charts fail on empty data	The "Stats per index" and "Stats per sourcetype" pie charts in the DHM entity overview crashed when the entity had no index or sourcetype data. Now renders gracefully with an empty state.
#1015	bug - Report creation retry logic does not detect HTTP 409 Conflict	The retry logic in `trackme_create_report` did not recognize HTTP 409 Conflict responses and exhausted its full retry budget. 409s are now detected as terminal and short-circuit the

11.1.33 Version 2.3.18 - build 1774997466 (01/04/2026)

Hint

TrackMe 2.3.18 — Virtual Groups, Policy Enhancements & Performance Optimizations

- **Virtual Groups — cross-tenant aggregation views**: This release introduces Virtual Groups, a powerful new feature for the Virtual Tenants UI. Virtual Groups are read-only cards that aggregate entities from multiple tenants and components into a single unified view, displayed alongside regular tenant cards. Combined with the recently introduced priority and tags policies — both lookup-based and search-based — and a new lightweight entity filter DSL, Virtual Groups enable a high level of flexibility in representing and monitoring key entities across tenants. A full creation wizard, RBAC-controlled visibility, and simulation mode make it easy to define, preview, and manage groups.
- **Key performance optimizations**: The FLX converging tracker has been re-engineered with direct REST calls and parallel execution, delivering up to 6-10x faster execution and eliminating excessive skip rates. The WLK metadata tracker received connection caching and batch KV loading optimizations for similar ~6-10x performance gains at large scale.
- **Various bug fixes and improvements**: Multiple fixes across DHM policies, table rendering, input sanitization, and policy simulation improve reliability and usability across the platform.

- SHA256: 2af279cb984a4d89e7cc2361060ae2929cbbe7d2118fb1006a706c64bef5f72b

Table 89: New Features

Issue Number	Description	Details
#895	feature - Virtual Groups — cross-tenant aggregation views	Virtual Groups are read-only aggregation cards displayed in the Virtual Tenants grid alongside real tenant cards. They allow users to select multiple tenants and components, then view all associated entities grouped by component family in a unified overview — providing cross-tenant visibility without creating new monitoring infrastructure. A 7-step creation wizard guides through group identity, tenant/component selection, priority filtering, additional entity filters via a lightweight DSL (supporting field=value matching with glob wildcards, AND/OR logic, and parentheses), RBAC role assignment, simulation preview, and final review. Virtual Group cards display entity count badges and priority indicators, and clicking a card opens a full overview modal with per-component donut charts and a complete entity table with all standard table features.
#891	feature - Monitoring Time Policy in bulk edit menu	Monitoring Time Policy is now available as a bulk action category in the bulk edit menu for all six component types (DSM, DHM, MHM, FLX, FQM, WLK). Users can select multiple entities and apply a built-in policy (e.g. "Business days, all hours") or configure custom day/hour combinations — applied to all selected entities in a single operation. The existing `EntityModifyMonitoringTimeModal` component has been extended with a bulk mode that joins all selected entities for the backend endpoint.
#932	feature - Cribl LogStream pack traffic monitoring use case	A new pre-built Flex Objects use case `cribl_logstream_pack_traffic` is available for customers using Cribl LogStream packs instead of routes. This use case monitors pack in/out event traffic via mstats (`cribl.logstream.pack.out_events`, `cribl.logstream.pack.in_events`), complementing the existing `cribl_logstream_route_traffic` use case for route-based deployments.

Table 90: Enhancements & Changes

Issue Number	Description	Details
#918	enhancement - FLX converging tracker optimization via direct REST calls	The FLX converging tracker (`trackmesplkflxconverging`) has been re-engineered to eliminate the costly SPL dispatch chain. Entity data is now fetched via direct REST calls to `/services/trackme/v2/component/load_component_data` instead of sequential SPL search jobs, with parallel execution using up to 4 concurrent workers for multi-tenant trackers. A new Python filter engine replaces SPL `| search` for root constraint evaluation, matching the Virtual Groups filter DSL syntax. These changes deliver up to 6-10x faster execution and effectively eliminate the ~48% skip rate observed on multi-tenant converging trackers.
#916	enhancement - WLK metadata tracker optimization with connection caching and batch KV loading	The WLK metadata tracker (`trackmesplkwlkgetreportsdefstream`) has been optimized for large-scale deployments where the per-record connection overhead caused excessive runtimes (26+ minutes average) and ~50% skip rates. Account credentials and service connections are now cached per `(account, user, app)` tuple, and the versioning KV Store collection is loaded upfront in batch for O(1) lookups instead of per-record queries. Expected improvement is ~6-10x faster execution, bringing runtimes in line with other hybrid trackers.
#898	enhancement - Policy simulation spinner animation and execution time display	Policy simulation buttons in Tags, Priority, and SLA policy modals now display a `WaitSpinner` and updated label text while the simulation is running, providing clear visual feedback during long-running simulations on large collections. Execution time is displayed upon completion.

Table 91: Bug Fixes

Issue Number	Description	Details
#923	bug - Table cells render `[object Object]` for user-added fields	Fixed entity table cells rendering the literal string `[object Object]` when users add custom fields containing objects or JSON data via column settings. The `renderCell` fallback in both Virtual Tenants and Tenant Home tables now detects object fields and renders them with an inspect badge and JSON inspector button, consistent with the existing handling of known structured fields (e.g. `metrics`).
#920	bug - Sanitize non-printable control characters from `root_constraint`	Fixed an issue where non-printable control characters (e.g. `\x08` backspace) present in `root_constraint` input — typically from copy-pasting SPL from PDFs or rich text editors — caused Splunk REST API `HTTP 400 Bad Request` errors during hybrid tracker, FLX, WLK, and replica tracker creation or update. All affected REST handlers now strip control characters (0x00-0x08, 0x0B, 0x0C, 0x0E-0x1F, 0x7F) from `root_constraint` immediately after reading from the request payload, preserving valid whitespace.
#910	bug - DHM policies: missing fields and broken multi-value matching	Fixed two inconsistencies in the DHM policy system affecting priority, tags, and SLA policies. First, the `data_index` and `data_sourcetype` fields were missing from the DHM entity field dropdown in the policy wizard, preventing their use in lookup-based or search-based policies. Second, multi-value field matching was broken for DHM entities because these fields are stored as comma-separated lists — exact, wildcard, and regex matching all compared against the full concatenated string instead of individual values. A new `MULTI_VALUE_FIELDS` constant and updated matching logic now correctly split and evaluate each value independently.
#906	bug - DHM: prevent `where` SPL error when no indexes/sourcetypes known	Fixed the "Stats per index" and "Stats per sourcetype" pie charts in the DHM entity overview failing with a malformed `where` expression error when the entity has no known `data_index` or `data_sourcetype` values. The `| format` command produced an empty string in this case, causing a parse error. A safety `| append` block now guarantees the subsearch always produces at least one row, ensuring the `where` clause receives a valid expression that gracefully filters out results instead of erroring.
#896	bug - Tags policies simulation parameter mismatch	Fixed the "Run policy simulation" button in the Tags Policies editor failing with `HTTP 400` when the policy type is search. The search mode payload incorrectly used shortened parameter names (`field_mappings`, `tags_field`, `tags_separator`, `match_mode`) instead of the full `tags_policy_lookup_` prefixed names expected by the backend. Applying the policy was not affected — only the simulation step was broken.
#912	bug - Incorrect `collection_name` in tracker health log	Fixed a misleading error log in the health tracker's `inspect_collection:handle_sync_entities` task that referenced the wrong variable for the collection name. The log reported `collection_name` (resolving to `kv_trackme_virtual_tenants`) instead of `inspect_collection_name` (the actual collection being operated on), causing confusion during troubleshooting.
#888	bug - Cribl LogStream use case normalization	Fixed the `cribl_logstream_total_traffic_inputs` Flex use case containing a copy-paste error from the outputs use case. The rex normalization block incorrectly referenced the `output` field instead of the `input` field, causing the input normalization to be a no-op since the `output` field does not exist in the input monitoring context.
#926	bug - Health tracker mlmodels-management task accesses collections for disabled tenants	Fixed the general health tracker's `mlmodels-management` task failing with `HTTP 400` errors when iterating tenants that are disabled or configured as replicas. Disabled tenants have their KV Store collections disabled, causing collection access to

11.1.34 Version 2.3.17 - build 1774424737 (25/03/2026)

- SHA256: 3a6a82c329f447d06a9a7869a7de7249fecaf5e581ed9e544ee8c9c054272600

Table 92: New Features

Issue Number	Description	Details
#874	feature - Replica tracker Execute/Manage/Create modals in Tenant Home	Full replica tracker modal support is now available from the Tenant Home Actions menu for replica tenants. The Execute modal allows selecting and running replica trackers with real-time execution monitoring via an events panel. The Manage modal provides a table view with search and filtering, multi-select deletion with confirmation, and direct links to orchestrator and replicator logs. The Create wizard offers a 4-step guided flow — select source tenant, define constraint, simulate results, and confirm creation — with an option to immediately run the new tracker on success.
#862	feature - Add 'Open in Search' action for Scheduler Review and Ops Health Statuses tables	A new "Open in Search" menu item is available in the gear/cog dropdown of the Scheduler Review & Investigate and Operational Health Statuses modals. The action opens the underlying SPL query directly in the Splunk Search view with proper time range parameters (earliest/latest), allowing administrators to inspect and troubleshoot scheduler and operational health data in their native search context.

Table 93: Enhancements & Changes

Issue Number	Description	Details
#881	enhancement - Health tracker performance optimization and shadow copy management	The health tracker received significant performance improvements. A tier-based task frequency model (Tier 1–4) reclassifies housekeeping tasks by criticality — from every-cycle (Tier 1) to daily (Tier 4) — drastically reducing per-cycle compute overhead on large deployments. License checks now use a global shared KV cache with a 24-hour TTL (immediately invalidated on any license mutation), eliminating redundant per-tenant REST calls. Saved search loading is batched into a single call per cycle instead of per-report REST roundtrips. The untracked entities safety net is moved from Tier 1 to Tier 2 with a short-circuit pre-check that skips the expensive SPL pipeline when no stale entities exist. Additionally, a new `shadow_enabled` per-tenant master switch allows administrators to fully disable shadow copy operations; shadow is now disabled by default for new tenants and automatically enabled by the health tracker when any component's entity count exceeds the configurable `shadow_entity_threshold` (default: 1000). A schema migration (2317) handles one-time auto-detection of shadow enablement status on upgrade.

Table 94: Bug Fixes

Issue Number	Description	Details
#849	bug - Failure to load a single tenant in Virtual Tenants affects all other tenants	Fixed the Virtual Tenants page failing to load entirely when a single tenant had a missing `vtenant_account` configuration (e.g. lost due to a Splunk failure). Faulty tenants are now skipped with a toast warning, and auto-recovery via `maintain_vtenant_account` is attempted after the main processing loop completes, ensuring all healthy tenants load normally.
#871	bug - Replica tenant Actions menu shows hybrid/elastic tracker options instead of replica tracker options	Fixed the Tenant Home Actions menu incorrectly displaying hybrid tracker, elastic source, and feature options for replica tenants. Replica tenants now correctly show replica tracker options (Execute, Manage, Create) with proper admin permission checks.
#860	bug - Virtual Tenant creation fails with IndexError in slow or busy environments	Fixed Virtual Tenant creation failing with an `IndexError` when the KV Store write is not immediately readable due to eventual consistency in slow or busy Splunk environments. The tenant creation process now includes retry logic with exponential backoff (up to 5 attempts) and falls back to the already-known tenant record if all retries are exhausted.

11.1.35 Version 2.3.16 - build 1774013895 (20/03/2026)

Hint

TrackMe 2.3.16 — High-Scale Performance, Cost Optimization & Platform Resilience

- **Scaling TrackMe to new heights**: This release is dedicated to pushing TrackMe's scalability boundaries with deep performance optimizations across all monitoring components. Key concepts — including hybrid trackers and the health tracker — have been re-factored to drastically reduce processing and compute costs, especially at high scale. All components (DSM, DHM, MHM, FLX) received extensive enhancements validated against scenarios with up to 100,000 entities in a single tenant.
- **Shadow copy KVstore collections — a new scalability paradigm**: TrackMe introduces a new concept of pre-computed, enriched entity snapshots stored in dedicated KVstore collections. The UI and backend automatically leverage shadow copies for near-instant table rendering, replacing expensive per-entity REST calls with efficient `| inputlookup` operations. With progressive loading and configurable page sizes, the UI now scales smoothly to multi-hundreds of thousands of entities.
- **DHM & hybrid tracker refactoring for extreme scale**: The Data Host Monitoring pipeline has been completely refactored with asynchronous data loading and optimized JSON parsing. DHM trackers are now capable of handling several tens of thousands of hosts per tracker, making enterprise-scale host monitoring practical and cost-effective.
- **Health tracker deep optimization**: The health tracker now implements task frequency tiering, batch KVstore operations, and parallel processing — significantly reducing its compute footprint and enabling faster health assessment cycles across large deployments.
- **DSM sourcetype explosion safeguard**: A new protective mechanism prevents runaway sourcetype discovery from overwhelming TrackMe and the Splunk environment. Administrators can configure a cap on sourcetypes per index, ensuring unexpected data patterns do not degrade performance or generate unmanageable entity volumes.
- **Blocklist, entity management & UI enhancements**: The blocklist interface has been expanded with custom categories, wildcard UX improvements, match type display, and a new simulation mode. New pagination options and a clickable degraded badge further improve the day-to-day operational experience.

- SHA256: ef9e84f24c6d7316b5baa1b5d75ff44cbdf46f7f860c1886262a5d4a869fccc7

Table 95: New Features

Issue Number	Description	Details
#782	feature - Shadow copy KVstore collections for instant UI loading at scale	TrackMe introduces shadow copy KVstore collections — precomputed, enriched entity snapshots that are automatically maintained by hybrid trackers and the health tracker. The UI loads entity tables via efficient `| inputlookup` operations instead of per-entity REST calls, delivering near-instant rendering even with hundreds of thousands of entities. Shadow copies are transparently managed and require no manual intervention.
#793	feature - Progressive shadow loading with configurable page size	Shadow collections support progressive loading with configurable batch sizes ranging from 5,000 to 50,000 records (default 25,000). Administrators can tune the page size via Configuration > User Interfaces to balance memory usage and rendering speed based on their environment.
#805	feature - DSM sourcetype explosion safeguard	A new configurable safeguard caps the number of sourcetypes tracked per index in Data Source Monitoring. When the cap is reached, TrackMe alerts administrators and prevents further sourcetype discovery for the affected index. This protects the environment from scenarios where unexpected data patterns cause an explosion of sourcetypes, leading to excessive entity creation and processing overhead.
#779	feature - DHM pipeline refactoring with async loading and optimized parsing	The Data Host Monitoring pipeline has been completely refactored with asynchronous component data loading and optimized JSON fast parsing. These changes enable DHM trackers to efficiently handle tens of thousands of hosts per tracker, making enterprise-scale host monitoring practical and cost-effective.
#745	feature - Health tracker deep optimization with tiered frequency, batch operations, and parallel processing	The health tracker has been deeply optimized with task frequency tiering, batch KVstore operations, and parallel processing. These improvements significantly reduce compute costs and accelerate health assessment cycles, especially in deployments with large numbers of tenants and entities.
#783	feature - Unified MHM pipeline and deprecated command cleanup	The Metric Host Monitoring pipeline has been streamlined with a unified processing approach and removal of deprecated commands, improving performance and maintainability.
#781	feature - FLX preloaded mode for high-cardinality scaling	Flex Objects tracking now extends preloaded mode to `trackmepersistentfields`, enabling efficient handling of high-cardinality datasets without excessive search-time processing.
#802	feature - Comprehensive blocklist management improvements with simulation mode	The blocklist management interface has been significantly enhanced with expanded fields, custom categories, improved wildcard UX, match type display, and a new simulation mode. The simulation feature allows administrators to preview which entities would be blocked before applying changes, reducing the risk of unintended entity suppression.
#755	feature - Cribl Edge fleet monitoring use cases	Two new pre-built Flex Objects use cases are available under a new Cribl Edge category for monitoring fleet node health. The first use case queries the Cribl Edge API via the TA-trackme-cribl add-on for rich fleet visibility including node health, input/output status, and heartbeat tracking. The second uses Cribl Edge internal metrics indexed in Splunk for node availability monitoring with built-in disruption queue grace periods.
#752	feature - Remote Splunk deployment support for priority, SLA, and tags policies	Priority, SLA, and Tags policies now support remote Splunk deployments. A new Splunk deployment selector in the policy creation and editing modals allows lookup-based and search-based policies to target remote Splunk accounts with automatic connectivity validation. The schema migration automatically adds the new `account` field to existing policy KVstore transforms upon upgrade.
#809	feature - Immediate score visibility via KV store cache	False positive and manual score influence changes are now immediately reflected in the UI via a dedicated KV store score cache, eliminating the delay previously experienced while waiting for the next tracker cycle to update entity status.
#817	feature - Degraded badge	The degraded status badge displayed on Virtual Tenant cards

Table 96: Enhancements & Changes

Issue Number	Description	Details
#794	enhancement - Shadow record patching after score and priority changes	Shadow records are now immediately updated after false positive, manual score influence, outlier reset, and priority changes, ensuring the UI reflects the latest entity state without waiting for the next full shadow refresh cycle.
#795	enhancement - Shadow write requester logging for improved observability	Shadow write log entries now include the requester identity (hybrid tracker report name, health tracker, or REST handler), improving observability and troubleshooting of shadow collection updates.
#792	change - Shadow write restricted to admin-only endpoint	Shadow collection writes are now restricted to a dedicated admin-only endpoint, preventing frontend-triggered shadow writes and ensuring shadow updates only occur through authorised backend processes.
#812	enhancement - Improved bulk deletion performance with backend batching	Bulk entity deletion now uses batched KVstore operations on the backend combined with a fire-and-forget frontend pattern, significantly reducing the time and resource consumption of large-scale entity removal operations.
#797	enhancement - Bulk delete actions with fire-and-forget pattern	Bulk delete operations now use a fire-and-forget pattern where the frontend triggers the operation and continues without blocking, while the backend processes deletions asynchronously with batched audit event posting and shadow cleanup.
#815	enhancement - Bulk support for variable delay disable endpoint	The variable delay disable endpoint now supports bulk operations, reducing the number of API calls required when managing variable delay settings across multiple entities.
#770	enhancement - Progressive backoff and batched search for delay inspection	The delayed entity inspector now implements progressive backoff (escalating 1x–4x multiplier for entities with consecutive empty inspections) and batched entity searches that combine multiple entities into a single tstats query, significantly reducing search load on the Splunk infrastructure while maintaining timely delay detection.
#777	enhancement - Proxy restricted index searches through REST API	Searches against restricted indexes (`_internal`, `_audit`) for scheduler status, completeness metrics, and alert trigger data are now proxied through the REST API, removing the need for direct user access to these indexes.
#771	enhancement - Elastic sources shared edit capabilities	Elastic Sources now support shared editing with a dedicated modal, improved explanation of shared versus dedicated reports, and a fix for the dedicated report link. These changes make it easier for teams to collaboratively manage elastic source configurations.
#758	enhancement - FLX Object description in entity header	Flex Objects entity headers now display the Object description field with truncation support, providing better context about each entity directly in the entity overview.
#767	change - Eliminate temporary saved search from FLX/WLK benchmark tests	The FLX and WLK burn test (benchmark) process has been refactored to avoid creating temporary saved searches during tracker creation, reducing search artefacts and cleanup overhead.
#784	enhancement - Extended pagination options with 100 and 1,000 entities per page	New pagination options for 100 and 1,000 entities per page are now available, with 100 set as the new default. These options complement the shadow copy architecture to deliver smooth navigation across large entity volumes.
#796	enhancement - Cribl Edge fleet API monitoring improvements	The Cribl Edge fleet monitoring use case has been further refined with improved API integration and enhanced data handling for fleet-level observability.
#774	enhancement - SPL Search Editor in elastic sources shared modal	The elastic sources shared edit modal now uses the SPL Search Editor component for an improved search editing experience with syntax highlighting and validation.
#847	enhancement - Improved readability of FQM wizard default thresholds and simulation table	The Data Quality Monitoring (FQM) wizard has been significantly improved for readability and usability. The simulation results table has been streamlined from 16 to 12 columns with human-readable headers, and a new per-row inspect button opens a detailed modal with quality summary, metadata

Table 97: Bug Fixes

Issue Number	Description	Details
#760	bug - FLX decision maker fails to promote orange entities to red when score threshold is met	Fixed a bug where the FLX decision maker's `status_not_met` re-check after score increment was dead code, preventing entities in orange status from being correctly promoted to red when their impact score exceeded the threshold.
#748	bug - Broken first chart in stateful alert email notifications on Splunk Cloud	Fixed chart rendering in stateful alert email notifications on Splunk Cloud deployments. The email structure now uses RFC 2387-compliant multipart formatting to ensure inline charts render correctly across mail clients.
#757	bug - Misleading HTTP 500 error returned for expired sessions	Fixed the REST API returning HTTP 500 errors instead of the correct HTTP 401 status when a user session has expired. Authentication-specific exceptions are now properly detected, logged at warning level instead of error (as this is a transient condition), and returned with a clear user-facing message.
#753	bug - Health tracker status output missing on some cycles	Fixed the health tracker not emitting its runtime status metric on every cycle, which could cause gaps in tracker health monitoring and observability.
#747	bug - Health tracker early return in tenant index settings check	Fixed an early return statement in the health tracker's `check_tenants_indexes_settings` function that could cause the function to exit prematurely, skipping necessary configuration checks.
#766	bug - FLX wizard not pre-populating configuration fields	Fixed the FLX tracker creation wizard not pre-populating the `drilldown_search`, `max_sec_inactive`, and `disruption_min_time_sec` fields from hybrid tracker results, requiring unnecessary manual entry.
#763	bug - MessageBar notifications ignoring light theme preference	Fixed MessageBar notification components not respecting the user's light theme preference, causing visual inconsistency in light-themed environments.
#775	bug - SearchJob.create() calls blocked by Splunk Workload Management	Fixed `SearchJob.create()` calls without an explicit time range being blocked by Splunk Workload Management (WLM) policies. All search job creation calls now include proper time range parameters.
#799	bug - WLK default thresholds using duplicate skipped percentage field	Fixed WLK default thresholds incorrectly using the `skipped_pct_last_24h` field instead of `skipped_pct_last_4h` for the 4-hour skipped percentage threshold, which could lead to incorrect anomaly detection for short-term patterns.
#820	bug - WLK default thresholds fixup applied via schema migration	Added a schema migration step to automatically correct WLK default threshold configurations affected by the duplicate field issue, ensuring existing deployments are fixed upon upgrade.
#807	bug - Shadow records not written for small entity counts	Fixed shadow records not being written when the entity count was below the shadow threshold, ensuring consistent shadow collection behaviour regardless of entity volume.
#811	bug - Shadow collection not cleared when no records to write	Fixed the shadow collection not being properly cleared when a tracker cycle produces no records to write, preventing stale shadow data from persisting.
#808	bug - Shadow refresh executor using incorrect thread type	Fixed the shadow refresh executor using daemon threads, which could be terminated prematurely during shutdown before writes complete. Shadow refresh threads are now non-daemon to ensure data integrity.
#828	bug - Virtual Tenant creation triggers health tracker failure and orphan account cleanup	Fixed health tracker execution failures during Virtual Tenant creation caused by the saved search not being ready immediately after creation. The immediate execution has been removed — the health tracker now naturally runs within its scheduled 5-minute cycle, allowing all knowledge objects to be fully initialised. Additionally, a new orphan account cleanup task automatically detects and removes stale vtenant configuration stanzas that no longer have a matching KVstore record.
#842	bug - Variable delay threshold — cannot clear duration input	Fixed an issue where duration input fields in the variable delay configuration could not be fully cleared using Backspace. The

11.1.36 Version 2.3.15 - build 1773055841 (09/03/2026)

Hint

TrackMe 2.3.15 — Native Outliers Engine, Workload Refinements & Enhanced Policies

- **Native ML density function engine**: TrackMe now implements its own outlier detection engine, fully independent from the Splunk AI Toolkit. Models are stored and managed in KVstore collections instead of file-based storage, delivering significant improvements in performance, flexibility, and operational simplicity — particularly in Search Head Cluster (SHC) environments where file-based models caused excessive replication activity between cluster members. This change also addresses regressions introduced by recent Splunk AI Toolkit updates. **Note:** Upon upgrading, TrackMe automatically resets the model training dates. Existing outlier models will not be available until they are re-trained, either automatically by the scheduled training process or manually through the UI.
- **Workload (WLK) threshold management overhaul**: WLK components received a major refinement with per-anomaly threshold configuration, providing granular control over how individual anomaly types contribute to entity health. Several regressions affecting anomaly detection accuracy have also been resolved.
- **Search-based (SPL) policies**: Priority, Tags, and SLA policies now support a new **search-based policy type**, allowing administrators to define policies using Splunk SPL queries. This enables seamless integration with source tenants and external data, making it easy to maintain and synchronise governance rules across environments.
- **React UI refinements**: Toast notifications have been deprecated in favour of Splunk UI Toolkit's MessageBar component, delivering a more consistent and polished notification experience throughout the application.

- SHA256: 627f7ebb79559c91b6eae398b8b9d7094a39f43f4e3b8536279aea7b7355d5e2

Table 98: New Features

Issue Number	Description	Details
#667	feature - Native TrackMe ML density function engine with KVstore model management	TrackMe now provides its own built-in outlier detection engine, removing the dependency on the Splunk AI Toolkit. Outlier models are stored and managed in KVstore collections, replacing the file-based approach used by the toolkit. This delivers better performance, easier management, and eliminates replication overhead in Search Head Cluster environments.
#678	feature - Search-based (SPL) policy type for priority, tags, and SLA policies	A new search-based policy type is available for priority, tags, and SLA policies. Administrators can define policies using Splunk SPL queries, enabling integration with source tenants and external data sources for centralised governance management across environments.
#685	feature - WLK flexible threshold configuration with per-anomaly management	Workload (WLK) components now support flexible, per-anomaly threshold configuration, giving administrators granular control over how each anomaly type is evaluated and scored. Out-of-monitoring-time removal and skipped percentage fixes are also included.
#702	feature - Continuous impact score metrics generation	Impact score metrics are now generated continuously, providing a complete time-series view of entity health scoring for improved observability and trend analysis.
#707	feature - Impact score over time chart in Score Definition modal	A new chart in the Score Definition modal visualises impact score trends over time, making it easier to understand how entity health evolves and to identify patterns.
#664	feature - Outliers training cooldown after score events	After an outlier score event is generated, a cooldown period now prevents immediate model retraining, preserving the model state so that administrators can investigate the anomaly before the model adapts.

Table 99: Enhancements & Changes

Issue Number	Description	Details
#731	enhancement - Migrate toast notifications to MessageBar	Toast notifications throughout the application have been replaced with Splunk UI Toolkit's MessageBar component, providing a more consistent and integrated notification experience.
#713	enhancement - Improved entity name visibility in High Priority Entities Overview	Entity names in the High Priority Entities Overview modal are now displayed with improved visibility, making it easier to identify entities at a glance.
#711	enhancement - Pre-populate secret fields when editing UCC configuration accounts	When editing existing UCC configuration accounts, secret fields are now pre-populated, reducing friction during account management and configuration updates.
#699	enhancement - Icons in bulk edit category dropdown	The bulk edit category dropdown now includes descriptive icons for each category, improving usability and visual navigation.
#690	enhancement - Prompt to enable lagging class override when modifying entity-level thresholds	When modifying entity-level thresholds, the UI now prompts users to enable the lagging class override if applicable, ensuring threshold changes take effect as expected.
#672	change - Default delay policy to variable with Business hours template for DSM/DHM	New DSM and DHM tenants now default to a variable delay policy with a built-in Business Hours template, providing more realistic monitoring thresholds out of the box.
#674	change - Lagging classes default to Priority level with Variable delay mode	Lagging class defaults have been updated to use Priority-level matching with Variable delay mode, aligning with the enhanced threshold management introduced in recent releases.
#693	change - Remove Windows OS-specific library dependencies	Removed Windows-specific library dependencies from the application package, reducing package size and simplifying deployment.
#732	change - Workload table Anomaly Reason column uses filtering instead of sorting	The Anomaly Reason column in WLK workload tables now uses filtering instead of sorting, making it easier to focus on specific anomaly types.
#735	change - Backup & Restore minor layout enhancements	Minor layout improvements to the Backup & Restore interface for better readability and consistency.

Table 100: Bug Fixes

Issue Number	Description	Details
#744	bug - Monitoring time policy not honoured — entities turn red outside monitoring window	Fixed entities turning red instead of remaining capped at orange when outside their configured monitoring time window. The impact-based scoring system was overriding the monitoring time cap by re-promoting entities to red when their score exceeded the threshold. All monitoring components (DSM, DHM, MHM, FLX, WLK) now correctly enforce monitoring time policies after scoring. Additionally, MHM now supports monitoring time policies for the first time.
#741	bug - Misleading "Splunk Service Unavailable" error during Virtual Tenant creation on reverse proxy timeout	Fixed the UI incorrectly displaying a "Splunk Service Unavailable" error banner when a reverse proxy timeout (504) occurs during Virtual Tenant creation. The backend operation completes successfully in the background, and the wizard already shows the correct "Background operation" notification — but the global error handler was also triggering a misleading error. The global handler now correctly excludes tenant creation API calls.
#680	bug - WLK decision maker fails to detect several anomaly types	Fixed a regression introduced in v2.3.6 where the WLK decision maker failed to detect errors, skipping, orphan, and delayed anomalies, potentially causing missed alerts for affected entities.
#669	bug - Duplicate and missing status messages for outlier-driven anomalies	Fixed duplicate or missing status messages being displayed for outlier-driven anomalies, ensuring accurate and consistent anomaly reporting.
#657	bug - Hybrid tracker records re-created in loop after saved searches are deleted	Fixed an issue where hybrid tracker records were continuously re-created after their corresponding saved searches had been deleted, causing unnecessary processing.
#665	bug - Schema version upgrade fails when upgrading from pre-2.0.9 directly to 2.3.14+	Fixed a schema migration failure that could occur when upgrading from very old TrackMe versions (pre-2.0.9) directly to 2.3.14 or later.
#700	bug - Outliers model stores numeric values as strings	Fixed outlier model dictionaries storing numeric values as strings instead of proper numeric types, which could affect model accuracy and calculations.
#652	bug - Missing keys in conf spec files cause btool check warnings	Fixed missing key definitions in configuration spec files that caused invalid key warnings during Splunk btool checks.
#709	bug - Date columns sort by string instead of epoch time in entity overview tables	Fixed date columns in entity overview tables sorting alphabetically by string value instead of chronologically by epoch time.
#704	bug - Scoring metrics contain null taskName dimension	Fixed scoring metrics containing null taskName dimensions caused by a Python 3.12 LogRecord attribute change.
#687	bug - Regex-based policy modal width inconsistent with other policy modes	Fixed the regex-based policy modal having inconsistent width compared to lookup and search policy modes.
#692	bug - ESLint errors in React views	Fixed ESLint errors in React view components to ensure code quality and consistency.
#724	bug - Pin packaging dependency for Python 3.7 compatibility	Pinned the packaging library version to maintain compatibility with Python 3.7 environments (Splunk 9.2.x).
#729	bug - AI status report missing in SHC environments	Fixed the AI status report being unavailable in Search Head Cluster environments due to the trackme_ai_provider configuration not being replicated across cluster members.

11.1.37 Version 2.3.14 - build 1772407097 (02/03/2026)

Hint

TrackMe 2.3.14 — Policies Management & Lookup-Driven Workflows

- **Major policies management enhancement**: This release introduces a significant step forward in how TrackMe manages entity governance at scale. Priority, SLA, and Tags policies now all support **lookup-based matching** alongside traditional regex-based rules, enabling seamless integration with external CMDB, asset inventories, and any structured data source available as a Splunk lookup (CSV or KVstore). Through a guided wizard, users can map lookup fields to entity attributes, configure value translations, and simulate policy results before applying — providing full visibility and control over policy outcomes. Regex policies are also enhanced with the ability to match against any entity field, not just the entity name. See *Managing priority via policy*, *Tags*, and *Using SLA alerting to build a 2-tier monitoring system*.
- Several bug fixes address outlier scoring accuracy, lagging class matching with variable delay policies, and AI Assistant navigation.

- SHA256: ce78970b699219a985268e4e195f186ca9b02764c508deefbc5312929429641a

Table 101: New Features

Issue Number	Description	Details
#641	feature - Lookup-based priority policies for CMDB-driven priority management	Priority policies now support **lookup-based matching**, enabling administrators to leverage existing Splunk lookups (CSV or KVstore) — such as CMDB or asset inventory data — to automatically assign priority levels to entities based on field matching. A guided wizard walks users through lookup selection, field mapping, priority assignment, and optional value translations (e.g. mapping external tier labels to TrackMe priority levels). Both exact and wildcard match modes are supported. Regex policies are also enhanced with **configurable match fields**, allowing patterns to match against any entity field (alias, index, sourcetype, etc.) instead of only the entity name. Both policy types coexist with highest-priority-wins conflict resolution, and a built-in **simulation mode** previews matched entities before applying. See *Managing priority via policy*.
#644	feature - Lookup-based policies extended to SLA and Tags	Following the same approach introduced for priority policies, **SLA policies** and **Tags policies** now also support lookup-based matching. SLA policies can assign SLA classes (gold, silver, platinum, etc.) from a Splunk lookup with configurable field mappings and value translations, using highest-SLA-rank-wins conflict resolution. Tags policies can assign tags from a lookup field with configurable separators, merging contributions from all matching rows and across all matching policies (regex and lookup combined). Both support exact and wildcard match modes, and include a simulation mode to preview matched entities and tag distribution before applying. See *Tags* and *Using SLA alerting to build a 2-tier monitoring system*.

Table 102: Enhancements & Changes

Issue Number	Description	Details
#639	enhancement - Bulk edit: show priority management info banner on category selection	The bulk edit priority management info banner — which indicates entities managed by policy or externally — now appears immediately when selecting the Priority Management category, rather than waiting for a specific action to be chosen. This aligns with the pattern established for lagging class banners and provides earlier visibility into policy-governed entities.
#629	enhancement - Shared validation utility for variable delay and lagging class time slots	Consolidated the duplicated time slot validation logic between variable delay and lagging class handlers into a shared utility, improving maintainability and ensuring consistent validation behaviour across both features.

Table 103: Bug Fixes

Issue Number	Description	Details
#646	bug - Outliers impact scoring generated when model confidence is low	Fixed outlier detection generating impact score events even when the ML model confidence is low (insufficient historical data). This could result in inflated and misleading outlier impact scores in the UI while the confidence banner clearly indicated insufficient data for reliable predictions. Scoring events are now correctly suppressed until the model has accumulated enough historical metrics.
#636	bug - AI Assistant "Execute in Search" opens in wrong app namespace	Fixed the AI Assistant's "Execute in Search" button opening SPL queries in the `search` app namespace instead of the `trackme` app namespace, which caused TrackMe custom commands and macros to be unavailable in the search context.
#650	bug - Missing "Critical" and "Pending" priorities in tenant creation wizard	Fixed the tenant creation wizard's default priority dropdown only offering Low, Medium, and High options. The Critical and Pending priority levels — which are valid backend values — were missing from all six creation wizards (FLX, FQM, WLK, DHM, DSM, MHM). The dropdown now includes all five priority levels.
#633	bug - Lagging class matching broken when tenant uses variable delay policy	Fixed lagging class matching being completely bypassed for tenants configured with a variable delay policy. All entities in such tenants showed no lagging class match regardless of configuration. The automatic coupling between variable delay and the lagging class override flag has been removed — lagging classes now correctly evaluate and apply alongside entity-level variable delay settings, and the override flag is purely user-controlled.

11.1.38 Version 2.3.13 - build 1772196565 (27/02/2026)

Hint

TrackMe 2.3.13 — Lagging Classes Redesign & Reliability Improvements

- **Lagging Classes Redesign for DSM and DHM**: Major redesign of the lagging classes framework introducing per-component collections (DSM and DHM), variable delay threshold support within lagging classes, and a modernised management UI. Lagging classes can now define either a static delay override or a full variable delay schedule with named time slots — bringing the same time-aware threshold flexibility introduced in 2.3.12 at the entity level, directly into lagging class policies. The matching engine has been

hardened with full-match semantics for regex patterns and priority-based evaluation. See *Lagging classes*.

- Several bug fixes address Splunk URL generation, schema migration resilience, startup modal behaviour, and KVStore record integrity.

- SHA256: 20c8af75cd9be91a74f8754bea7d7b3f5282660a42234170ab60d75a57d72c80

Table 104: Enhancements & Changes

Issue Number	Description	Details
#607	enhancement - Warn user when outliers confidence configuration changed since last training	When the minimum days for confidence setting is changed after a model has been trained, the UI now displays a warning advising the user to retrain the model to reflect the updated confidence parameters.
#615	enhancement - AI Assistant: show response time for AI answers	Added a response time indicator below each AI Assistant answer (e.g. "responded in 2.3s"), providing helpful feedback about AI response performance.
#617	enhancement - AI Assistant: improved latency investigation searches	Simplified the latency investigation searches provided by the AI Assistant for DSM and DHM entities, making them more readable and easier to customise.

Table 105: Bug Fixes

Issue Number	Description	Details
#608	bug - Splunk URL generation regression with multi-segment path prefixes	Fixed a regression where UI-generated URLs could produce broken links when TrackMe is deployed behind a multi-segment path prefix, causing missing locale or prefix path segments.
#613	bug - Regex mode uses prefix matching unlike other modes	Fixed an inconsistency in lagging class pattern matching where regex mode performed prefix matching instead of full-string matching. A regex pattern like `syslog` would unexpectedly match `syslog_secure`. Regex mode now enforces full-string matching consistent with exact and wildcard modes.
#623	bug - Startup modal dialog opening logic	Fixed startup modal dialogs appearing when not appropriate. The licensing update modal no longer appears for environments with an active paid license, and modal sequencing has been improved to prevent multiple dialogs from opening simultaneously.
#625	bug - DSM/DHM records written to KVStore without tenant_id field	Fixed a bug where DSM and DHM entities could be persisted without a `tenant_id` field, causing the entity to appear outside its tenant group in the Tenant Home UI. All entities are now properly associated with their tenant during record creation.

11.1.39 Version 2.3.12 - build 1772017884 (25/02/2026)

Hint

TrackMe 2.3.12 — Variable Thresholds at Scale & AI Assistant Maturity

- **AI Assistant from A to Z**: The AI Assistant reaches full maturity with refinements and enhanced capabilities. New anonymisation options protect sensitive data (entity names, index names) when sending context to AI providers. Stateful alert actions now integrate the AI Status Report directly in email notifications, delivering AI-generated entity summaries tailored to the incident lifecycle (opened, updated, closed). See *AI Assistant*.

- **Variable Delay Thresholds for DSM and DHM**: A long-requested feature that gives TrackMe full flexibility to dynamically adapt how data sources and hosts delays are evaluated at scale. Define time-aware thresholds that change based on business days, hours, weekends, and quiet periods — tighter thresholds during active periods, relaxed thresholds during nights and weekends. Benefits all customers from Foundation to Enterprise and Unlimited. See *Variable delay*.
- **Variable Thresholds in Flex Objects**: Similarly to feeds tracking, Flex Objects now support variable thresholds — Enterprise-scale flexibility for any KPI according to business rules. Define time slots with different threshold values per metric, enabling context-aware monitoring that adapts to your operational calendar.

- SHA256: 12f90414b657782c1a49268eca54e47a05ed515f48a3f2a8ac2e02e39f628604

Table 106: New Features

Issue Number	Description	Details
#1566	feature - Variable delay thresholds for DSM and DHM	Introduced variable delay thresholds for splk-dsm and splk-dhm components, enabling time-aware delay monitoring. The delay threshold for an entity can now change dynamically based on the day of the week and hour of the day. Define named time slots (e.g. business hours, weekends) each with its own threshold value — tighter during active periods, relaxed during nights and weekends. Supports auto-computation from historical metrics, auto-review when data patterns shift, template presets (business hours, weekday/weekend, three-tier), tenant-level defaults, per-entity configuration, and bulk operations. Mutually exclusive with adaptive delay. See *Variable delay*.
#1573	feature - Variable thresholds in Flex Objects	Introduced variable thresholds for Flex Objects (splk-flx), providing Enterprise-scale flexibility for threshold-based monitoring. Define time slots with different threshold values per KPI according to business rules — adapt monitoring sensitivity based on day of week, hour of day, weekends, and operational calendars. Enables context-aware Flex Object monitoring that aligns with your organisation's activity patterns.
#1570	feature - AI Status Report in stateful alert email notifications	Stateful alerts can now include an AI-generated entity status report in email notifications. When enabled, each email notification embeds a concise, actionable summary generated by the configured AI provider, tailored to the incident lifecycle stage (opened: what went wrong and where to investigate; updated: whether the situation has changed; closed: confirmation of recovery). Uses the same AI provider configuration — no additional setup required. Fail-open behaviour ensures emails are never blocked by AI unavailability. Additionally, a new `email_component_filter` option allows administrators to specify which components (dsm, dhm, mhm, fqm, flx, wlk) are eligible for email notifications, displayed as a multi-select dropdown in the wizard when the tenant has more than one component enabled. See *AI Assistant*.
#1565 / #1575	feature - AI Assistant anonymisation capabilities	Added configurable anonymisation options for the AI Assistant to protect sensitive data when sending entity context to AI providers. **Anonymize entity names for AI**: When enabled, entity names (object and alias fields) are anonymized using SHA256 hashing; the AI is guided to use object_id references instead. **Anonymize index names for AI**: When enabled, Splunk index names are anonymized in DSM, DHM, and MHM entity context. Both options are configurable in **Configuration > General**. See *AI Assistant*.
#1568	feature - Add Splunk Hosted LLM (SLIM API) as AI provider	Added `splunk_hosted` AI provider type for Splunk Cloud customers to use Splunk's hosted generative AI models via the SLIM API. The SLIM API endpoint is auto-discovered from SCS tenant info — no admin-provided base URL is needed. Authentication uses dynamic SCS tokens — no API key is required. A new model discovery endpoint (`GET /services/trackme/v2/ai/admin/models`) lists available SLIM API models. Includes SSE streaming support with non-SSE fallback for robust compatibility. See *AI Assistant*.
#1571	feature - AI Assistant chat UX enhancements	Enhanced the AI Assistant chat experience across all AI Assistant panels. Adds a resizable input area with a drag handle, fenced code block detection and rendering with SPL syntax highlighting, a **Copy to clipboard** button on all code blocks, and an **Execute in Search** button on SPL code blocks that opens the query directly in Splunk Search. User messages now render as plain text for clarity, and suggested question buttons use a modern ghost/outline style. See *AI Assistant*.

Table 107: Enhancements & Changes

Issue Number	Description	Details
#1577	enhancement - Describe API endpoints use realtime decision maker view instead of KVstore records	Changed the AI describe endpoints to query the realtime decision maker view for live entity state rather than relying on KVstore records. This ensures that AI responses reflect the current entity conditions as computed by the decision maker, instead of potentially stale KVstore data. Applies to all describe endpoints across DSM, DHM, MHM, FLX, and FQM components.
#1567	enhancement - Preserve search pattern across tab switches in Tenant Home UI	The search bar pattern in Tenant Home is now preserved on a per-component basis when switching between tabs (DSM, DHM, MHM, FQM, FLX, WLK). Patterns persist until the user performs a full page refresh or manually clears the search input, eliminating the need to re-enter the filter each time a tab is switched.
#1564	enhancement - Virtual Tenants UI: add tenant multiselect filter in Scheduler and Operation Status views	Added a tenant multiselect dropdown filter to the Scheduler Review and Operation Status pages. Users can now filter tracker jobs and operational data by one or more selected tenants. The filter enforces mutual exclusivity between the "All Tenants" option and individual tenant selections, and defaults to showing all tenants when no filter is applied.
#1572	enhancement - Preserve cross-package user preferences in tenant-home and trackme-ai-assistant	Fixed a critical issue where toggling dark/light mode from the Tenant Home or AI Assistant pages destructively overwrote Virtual Tenants user preferences. All UI pages now correctly preserve existing preferences (column visibility, widths, ordering, hidden tenants, auto-refresh settings) when updating the colour scheme, instead of replacing them.
#1578	change - Increase default Outliers minimal number of days for confidence from 7 to 30 days	Increased the default value of `splk_outliers_min_days_history` from **7 days** to **30 days** in `trackme_settings.conf`. The previous 7-day minimum was too short to establish reliable confidence levels for outlier detection. A 30-day baseline provides a more robust foundation for the outlier detection confidence calculation.
#1580	enhancement - Improve readability and SPL syntax highlighting in hybrid tracker wizard search input	Improved readability of SPL search input zones across all hybrid tracker wizard creation flows, drilldown search modals, and stateful alert configuration. All SPL input fields now feature an **Edit / SPL Preview toggle**: Edit mode provides a larger monospace font for improved readability, while SPL Preview mode renders the query with full SPL syntax highlighting. Supports both light and dark mode themes. Applied consistently across all DSM, DHM, MHM, FQM, FLX, WLK, and Elastic Sources wizards, as well as drilldown search and alert configuration modals.
#1579	change - Reduce right image size in FLX converging panel info	Fixed the diagram image on the right side of the FLX converging wizard panel info being displayed at the same size as the screenshot on the left, making it look oversized and degraded. The diagram is now properly constrained and scaled to maintain visual clarity alongside the screenshot.

Table 108: Bug Fixes

Issue Number	Description	Details
#1576	bug - High priority entities count discrepancy between header badge and modal list	Fixed a mismatch where the header badge count for high priority entities did not match the number of entities shown in the High Priority Entities Overview modal. When disabling a component, stale priority counts were left behind in the tenant summary, inflating the header badge total beyond the actual entities displayed in the modal. The modal now correctly filters to enabled entities only, and disabling a component properly cleans up all associated summary and statistics data.
#1574	bug - Some backends are not honouring system-level splunkd timeout in backend REST calls	Fixed several backend handlers that had a hardcoded 120-second timeout in their REST calls instead of respecting the system-level `splunkd_timeout` setting from `trackme_settings.conf`. Affected areas include the FLX converging custom command (customer-reported), the AI Assistant chat endpoint, and the AI configuration endpoint. All now honour the administrator-configured timeout value.
#1569	bug - Splunk root URI auto-detection may include /app suffix causing double /app in drilldown links	Fixed the Splunk root URI auto-detection logic which incorrectly included `/app` in the detected base URL, causing drilldown links in stateful and notable alert email notifications to contain a double `/app/app/` in the path. The locale detection was also improved to support all Splunk locales instead of a hardcoded list. Additional defence-in-depth stripping was added on the backend side for both stateful and notable alert handlers.
#1563	bug - Stateful alert: premature incident closure when entity is red due to manual score influence (empty anomaly_reason)	Fixed premature incident closure in the stateful alert helper when an entity remains in a red state due to manual score influence but has an empty `anomaly_reason` field. The alert helper incorrectly interpreted an empty `anomaly_reason` as the entity having recovered, triggering a `closed` lifecycle event and sending a recovery email notification. The fix ensures that the entity's actual `priority` status is evaluated alongside `anomaly_reason` to correctly determine whether the entity has genuinely recovered before closing the incident.

11.1.40 Version 2.3.11 - build 1771520654 (19/02/2026)

Hint

TrackMe 2.3.11 — AI Assistant Everywhere & Reliability Improvements

- This release extends the **AI Assistant to all TrackMe UI pages**, making context-aware AI guidance available across the entire product — from entity investigation to REST API reference, backup management, maintenance windows, bank holidays, and license management. Each page automatically provides its own structured context to the AI, enabling precise, data-driven answers without manual explanation. See *AI Assistant*.
- **Python 3.13 compatibility**: Fixed Data Sampling regex failures on Splunk environments running Python 3.13, where global inline flags are now enforced at position 0.
- **User preferences integrity**: Fixed a critical bug where toggling dark/light mode from secondary UI packages silently destroyed all user preferences (column visibility, widths, ordering, hidden tenants) stored by the Tenant Home and Virtual Tenants packages.
- **Codebase cleanup**: Fully decommissioned the deprecated identity cards feature and removed unused KV-store collections, eliminating dead endpoints and reducing codebase complexity.

- Several additional fixes address restore reliability, Logs Inspector performance, component metrics generation, Flex Converging error handling, and logging verbosity.

- SHA256: bf58a5de087059f771b0decb23c1e7ecba4ea50a4eb6c795c4b44d66c33b3ead

Table 109: New Features

Issue Number	Description	Details
#1551	feature - AI Assistant extended to all TrackMe UI pages	Extended the TrackMe AI Assistant from entity-level investigation to a product-wide feature available across all major TrackMe UI pages. The AI Assistant button is now available in the top-right header of: **REST API Reference** (helps navigate the API catalogue with curl and SPL `| trackme` usage examples), **Backup & Restore** (guidance on backup status, creation, restore procedures, and best practices), **Global Maintenance Mode** (maintenance window management, scheduling, and alerting impact), **Maintenance Knowledge Database** (per-entity maintenance record management for planned and unplanned windows), **Bank Holidays Management** (holiday calendar configuration, country-based imports, recurring calendars, and SLA impact), and **License Management** (license status, edition differences, feature availability, and renewal workflows). Each page automatically collects its own structured context (live system state) and passes it to the LLM via a dedicated system prompt, enabling the AI to provide specific, data-driven answers. Suggested questions are tailored per page to help users get started quickly. No additional configuration is required — the existing AI Assistant settings (enable/disable toggle and AI provider configuration) apply globally to all entry points. These complement the existing AI Assistant availability in entity investigation panels (DSM, DHM, MHM, FLX, FQM, WLK), Virtual Tenants management, and Tenant Home. See *AI Assistant* for full documentation.

Table 110: Enhancements & Changes

Issue Number	Description	Details
#1553	enhancement - Virtual Tenants: add clear button to tenant search bar	Added a clear (×) button inside the tenant search bar input field in the Virtual Tenants UI. When a search string is entered to filter tenants by id, alias, or description, a small cross icon now appears on the right side of the input. Clicking it clears the search field and resets the filtered tenant list in a single click, instead of requiring users to manually select and delete the text.
#1555	enhancement - Flex Converging: graceful handling when tenant/component no longer exists	Improved the `trackmesplkflxconverging` custom command to validate tenant/component combinations before attempting to access their entities. Previously, when the command was called for a tenant or component that no longer exists, it generated confusing error-level exceptions. The command now calls the `/trackme/v2/vtenants/trackmeload` endpoint at startup to retrieve the full list of active tenants and their enabled components, then validates each `tenant_id:component` pair. Invalid combinations are logged at WARNING level and gracefully skipped, while processing continues for any remaining valid pairs.
#1558	change - Decommission deprecated identity cards REST endpoints and macros	Fully decommissioned the deprecated identity cards feature, which was deprecated in version 2.0.87 and replaced by the notes feature. Removed all REST endpoint handler files (`trackme_rest_handler_splk_identity_cards_power.py`, `trackme_rest_handler_splk_identity_cards_user.py`), associated macros (`trackme_get_identity_card(2)`, `trackme_identity_card_default_url`, `trackme_identity_card_default_note`), REST map and web.conf stanzas, handler map entries in `trackmeapiautodocs.py`, and legacy references in `trackme_libs_describe.py`. This cleanup removes dead endpoints, eliminates invalid context pollution from the AI-powered describe endpoint, and reduces codebase complexity.
#1556	change - Decommission deprecated KVstore collections `kv_trackme_user_pref` and `kv_trackme_user_uipref`	Removed the deprecated and unused KVstore collections `kv_trackme_user_pref` and `kv_trackme_user_uipref`, which were deprecated in TrackMe 2.2.4 and are no longer used by any component. Collection definitions were removed from `collections.conf` and references cleaned up in the KVstore audit dashboard (`trackMe_kvstore.xml`).

Table 111: Bug Fixes

Issue Number	Description	Details
#1550	bug - Data Sampling: PatternError with Python 3.13 due to global regex flags not at the start of the expression	Fixed a bug where Data Sampling execution failed with a `PatternError` on Splunk environments running Python 3.13. Python 3.13 enforces that global inline regex flags (e.g. `(?i)`, `(?s)`, `(?m)`) must appear at position 0 of the regex pattern (previously a `DeprecationWarning` since Python 3.11, now a hard error). Two OOTB regex patterns in `trackmedatasampling_ootb_regex.py` used `^(?i)...` which places the flag at position 1 (after `^`): `raw_start_by_current_time_then_timestamp` and `raw_start_by_timestamp %A %d %B %Y %H:%M:%S.`. The fix moves `(?i)` before `^` in OOTB patterns, and adds a `sanitize_regex_global_flags()` helper function in `trackmesamplingexecutor.py` that automatically relocates standalone global inline flags to the start of any regex pattern before calling `re.search()`, ensuring backward compatibility with user-defined custom sampling models.
#1552	bug - Restore: saved searches and alerts fail when `earliest_time` / `latest_time` are None in backup	Fixed a bug where restore operations failed with `HTTP 400 Bad Request -- Cannot parse time argument 'dispatch.earliest_time': 'None'` when the backed-up `earliest_time` or `latest_time` properties were `None`. During backup, `savedsearch_object.content.get("dispatch.earliest_time")` returns Python `None` when the field is not explicitly set, which is serialised to JSON as `null`. During restore, these `None` values were unconditionally included in the properties dictionary, causing the Splunk SDK to convert them to the string `"None"` which the Splunk REST API rejects. The fix applies the same conditional guard already used for `cron_schedule` to `dispatch.earliest_time` and `dispatch.latest_time` in both the saved search and alert restore code paths: values are only included if they are not `None`, `"None"`, or `"null"`.
#1557	bug - User preferences cookie overwrite destroys cross-package preferences	Fixed a critical bug where toggling dark/light mode from any of the 8 secondary React UI packages (Backup & Restore, License, Maintenance KDB, Maintenance Mode, Bank Holidays, REST API Reference, Logs Inspector, Remote Accounts Overview) silently destroyed all user preferences stored by the Tenant Home and Virtual Tenants packages. All 10 React UI packages share the same `trackme_user_preferences` browser cookie, but the 8 secondary packages performed a destructive overwrite — replacing the entire cookie with only `{colorScheme: "dark"}` instead of performing a read-modify-write to preserve existing preferences. This caused loss of column visibility, column widths, column ordering, hidden tenants, auto-refresh settings, and other user-configured layouts. The fix changes all 8 packages to read existing preferences, merge the new `colorScheme` value, then write back the merged object. Cookie expiry was also aligned from 365 days to 3650 days (10 years) to match the Tenant Home and Virtual Tenants packages.
#1559	bug - Components Register: runtime metrics not generated on first tracker execution	Fixed a bug where runtime metrics (`trackme.components_register.runtime`) were not generated on the first execution of tracker jobs. In `trackme_register_tenant_object_summary` and `trackme_register_tenant_object_summary_gen_persistent` (`trackme_libs.py`), the call to `trackme_components_register_gen_metrics` only occurred when updating an existing report entry. When the report key did not yet exist (first execution) or the dictionary was empty, metrics generation was skipped entirely. While this was barely noticeable for frequently-running DSM/DHM

11.1.41 Version 2.3.10 - build 1771340984 (17/02/2026)

- SHA256: 31e2cf8c25447ef7f4e2694beadeac67ba89fb30580d4cabdf6bffe2ac2daaf8

Table 112: Enhancements & Changes

Issue Number	Description	Details
#1542	enhancement - Allow configuring the splunkd timeout for local REST API calls and SDK connections	Introduced a configurable `splunkd_timeout` setting under the `[trackme_general]` stanza in `trackme_settings.conf`, with a default value of 300 seconds. This addresses a regression introduced post-v2.3.4 where several critical timeout values for local splunkd REST API calls and SDK connections were silently reduced from 600 seconds to 60 seconds during the backoff/retry refactoring in v2.3.7, causing `HTTPSConnectionPool Read timed out (read timeout=60)` errors under high load conditions. Administrators can now adjust this value through the TrackMe Configuration UI (**Configuration > General**, Performance behavior section) within a valid range of 30–600 seconds. The setting applies to all local splunkd REST API calls (`requests.get`/`requests.post`) and SDK connections (`client.connect`) except external SOAR/Phantom API calls which remain at their own dedicated timeout. A new `get_splunkd_timeout()` helper function and `SPLUNKD_TIMEOUT_DEFAULT` constant were added to `trackme_libs.py` for consistent usage across all affected modules.

Table 113: Bug Fixes

Issue Number	Description	Details
#1541	bug - Outliers: Missing audit events for bulk rules update and per-entity ML model operations	Fixed multiple issues with missing or incorrect audit events in Outliers-related operations. **Bulk rules update**: the `post_outliers_bulk_rules_update` endpoint was missing audit event generation entirely; audit events (success and failure) are now properly created for each modified entity. Additionally, the `update_comment` field from the request payload was not being excluded from the update fields, potentially causing it to be incorrectly applied as an outlier rule field. **Per-entity ML model operations**: all per-entity ML model endpoints (`add`, `update`, `delete`, `reset`, `manage period exclusion`) had the `object_category` parameter hardcoded to `"splk-dsm"` instead of using the dynamic component value (`splk-{component}`), causing audit events for non-DSM entities (e.g., flx, fqm, dhm, wlk) to be stored with the wrong object category and become invisible in the entity's audit trail. **Bulk period exclusion**: the `post_outliers_bulk_period_exclusion` endpoint was calling `trackme_audit_event` with only 7 positional arguments instead of the required 11, causing the audit event to fail silently. All issues have been resolved.
#1543	bug - Schema upgrade fails with [Errno 97] Address family not supported by protocol on IPv6-disabled systems	Fixed a bug where the schema upgrade task failed with `[Errno 97] Address family not supported by protocol` on systems where `localhost` resolves to the IPv6 loopback address (`::1`) but IPv6 networking is disabled. All 60 `client.connect()` calls in `trackme_libs_schema.py` did not specify the `host` parameter, causing the Splunk Python SDK to default to `"localhost"` which could resolve to `::1`. The fix passes `host=reqinfo.get("server_rest_host", "127.0.0.1")` to all SDK connection calls, ensuring the connection uses the actual splunkd host address with a safe IPv4 fallback. Additionally, in `trackmetrackerhealth.py`, the `schema_version` variable is now initialised to `None` before the try block to prevent a `NameError` from masking the original connection error when `trackme_schema_get_version()` raises an exception. This bug completely blocked KV store schema migrations on affected systems.
#1545	bug - Flex Objects wizard: Missing impact score slider in Outliers Metrics (ML) definition	Fixed a bug where the Flex Objects creation wizard was missing an impact score slider in the Outliers Metrics (ML) section. While the Default Thresholds section already included an impact score slider per threshold rule, the Outliers section did not have this equivalent control, preventing users from setting a custom impact score for each metric's outlier detection. The fix adds a `score` field to the `FlxOutlierMetricParams` type and an impact score slider row (0–100) in the outlier metric configuration card, pre-populated with the tenant's default outlier impact score from the Manage Impact Score Configuration. The score value is now properly serialised into the `outliers_metrics` eval statement.

11.1.42 Version 2.3.9 - build 1771243524 (16/02/2026)

Hint

TrackMe 2.3.9 — Introducing the In-App AI Assistant & Reliability Improvements

- This release marks a landmark evolution for TrackMe with the introduction of the **In-App AI Assistant**, bringing AI-powered entity investigation and contextual guidance directly into the Splunk interface.
- **AI Assistant**: A fully integrated, context-aware AI chat assistant that analyses entity health, interprets ML Outliers results, suggests investigation steps, and provides threshold tuning advice — all within the entity investigation panel. Entirely opt-in, no data is sent to any external service until an administrator explicitly configures a provider. Provider-agnostic, it supports OpenAI, Azure OpenAI, Anthropic Claude, Google Gemini, Mistral AI, Ollama, and any OpenAI-compatible endpoint — giving organisations full control, including the option to use self-hosted open-source LLMs for complete data sovereignty. See *AI Assistant*.
- **Python 3.13 compatibility**: Critical fixes for Splunk 10.2+ / 11.x environments, addressing SVG-to-PNG chart conversion in Stateful Alerting and SOAR integration module loading.
- **Data integrity**: Fixed a persistent field handler bug where custom per-entity impact score weights and other user-configured values could be silently overwritten during tracker conflict updates.
- **UI refinements**: Improved Outliers detection disabled-state visibility, threshold simulation accuracy, React chart stability, and refreshed all Splunk UI React dependency packages to their latest versions with measurable bundle size reductions.
- Several additional enhancements address global license usage monitoring use cases and default groupby consistency.

Important

Introducing the TrackMe AI Assistant — AI-Powered Entity Investigation

The TrackMe AI Assistant is a **context-aware, provider-agnostic AI chat interface** embedded directly into the entity investigation workflow. This is a pivotal capability that positions TrackMe at the forefront of the Splunk ecosystem, bridging the gap between data observability and intelligent, AI-driven operations.

Why it matters:

- **Instant contextual analysis**: When a user opens the AI Assistant from any entity, TrackMe automatically collects the full structured description of the entity — health state, scoring breakdown, metrics, thresholds, ML Outliers results, investigation SPL searches, and more — and injects it into the LLM context. The AI provides specific, data-driven answers without the user having to describe the situation manually.
- **Actionable investigation guidance**: The assistant suggests concrete investigation steps, explains anomaly scoring, interprets ML model outputs, and provides entity-type-specific threshold tuning advice — turning complex observability data into clear next actions.
- **Opt-in only**: No AI functionality is active and no data leaves your environment until an administrator explicitly configures a provider. The feature is completely inert without a configured provider.
- **Full provider flexibility**: Administrators can configure one or multiple LLM providers simultaneously — cloud-based (OpenAI, Azure OpenAI, Anthropic, Google Gemini, Mistral AI) or fully self-hosted (Ollama) for environments with strict data governance requirements. Users can switch between providers from the chat panel to compare results or optimise costs.
- **Data sovereignty**: Organisations with strict data privacy or regulatory requirements can connect TrackMe to their own LLM infrastructure — self-hosted open-source models via Ollama, private corporate deployments via any OpenAI-compatible endpoint, or region-specific cloud deployments. The choice is entirely yours.
- **Enterprise-grade security**: RBAC-enforced entity context reads, encrypted API key storage, system-level credential isolation, and configurable concurrency limits ensure the AI Assistant operates within your

organisation's security boundaries.

- **Streaming experience**: Real-time token streaming with Markdown rendering, syntax-highlighted SPL code blocks, and conversation context within sessions provide a modern, responsive chat experience.

For full configuration and usage details, see the *AI Assistant* documentation.

- SHA256: 3e3d9ba1b2c759697e21d4aa6001d285dd5264d9e1da7c8ba2e5b5b4fcc80504

Table 114: New Features

Issue Number	Description	Details
#1537	feature - In-App AI Assistant for entity investigation	Introduced the TrackMe AI Assistant, an integrated, context-aware AI chat assistant accessible from any entity investigation panel across all entity types (DSM, DHM, MHM, FLX, FQM, WLK). The assistant automatically collects the full structured entity description (health state, scoring breakdown, metrics, thresholds, ML Outliers results, investigation SPL searches, and threshold tuning guidance) and passes it to the configured LLM provider as context. Supports two entry points: the **AI Assistant** button in the entity panel header and the **Ask AI** option in the entity three-dot actions menu. Features include five suggested starter questions, free-form conversational input with session context, real-time token streaming, Markdown rendering with syntax-highlighted SPL code blocks, multi-provider switching, and automatic job cancellation on panel close. The assistant is provider-agnostic with support for OpenAI, Azure OpenAI, Anthropic Claude, Google Gemini, Mistral AI, Ollama (self-hosted), and any OpenAI-compatible API endpoint. Security features include user-level RBAC enforcement for entity context reads, system-level credential isolation, encrypted API key storage via Splunk `passwords.conf`, and a configurable concurrency semaphore to prevent resource exhaustion. Configuration is managed through **Configuration > General** (enable/disable, max concurrent chats) and **Configuration > AI Provider** (provider-specific settings including base URL, model, max tokens, temperature, request timeout, custom prompt). See *AI Assistant* for full documentation.

Table 115: Enhancements & Changes

Issue Number	Description	Details
#1532	enhancement - Add Global License Usage Monitoring Use Cases	Added new FlexLog use cases for monitoring global Splunk License usage at the environment level. `splk_license_usage_global_splunkcloud` monitors the total Splunk License usage for Splunk Cloud environments, generating rolling metrics (60m, 4h, 24h) for the overall license volume in MB, ideal for ML-based outlier detection. `splk_license_usage_global_splunkenterprise` monitors total usage for Splunk Enterprise environments, generating rolling metrics for both volume in MB and pool percentage consumption. These global use cases complement the existing per-index use cases by providing a single `@splunk-global` object that tracks the entire environment's license consumption without splitting by index or pool, making it straightforward to detect overall volume anomalies at the environment level. Additionally, the group definition for the existing per-index use cases (`splk_license_usage_per_index_splunkcloud` and `splk_license_usage_per_index_splunkenterprise`) has been changed from `volume-indexes` to `volume-outliers`.
#1535	enhancement - Default GroupBy Behavior Consistency - Handle `none` Keyword Properly	Improved the default groupby behavior in the Tenant Home UI so that setting the keyword `none` in the Virtual Tenant account GroupBy settings now produces the same behavior as when users manually select "No grouping" from the UI dropdown. Previously, the `none` keyword was not properly interpreted, leading to inconsistent behavior between the admin-configured default and the user-selected "No grouping" option. The `none` keyword is now correctly handled as equivalent to "No grouping" across all components.
#1540	change - Splunk UI - Refresh All React Dependency Packages to Latest Versions	Refreshed all Splunk UI React dependency packages to their latest compatible versions for release 2.3.9. Notable updates include `@splunk/dashboard-*` (29.1.0 to 29.4.0), `@splunk/react-ui` (5.4.0 to 5.8.0), `@splunk/react-icons` (5.6.0 to 5.8.0), `@splunk/react-page` (8.1.0 to 8.2.1), `@splunk/themes` (1.4.1 to 1.5.0), `@splunk/visualizations` (28.3.0 to 28.4.0), `@babel/core` (7.28.4 to 7.29.0), `@typescript-eslint/*` (8.45.0 to 8.55.0), and `webpack` (5.102.0 to 5.105.2). Bundle sizes decreased across the board, with TenantHome.js going from 9.39 MiB to 8.65 MiB, VirtualTenants.js from 9.03 MiB to 8.27 MiB, and other bundles seeing similar reductions.

Table 116: Bug Fixes

Issue Number	Description	Details
#1529	bug - Stateful Alerting - SVG to PNG Chart Conversion Fails Under Python 3.13 (Splunk 10.2+)	Fixed an issue where stateful alert email chart images failed to render as PNG when Splunk runs Python 3.13 (introduced in Splunk 10.2+ via `python.required = 3.9,3.13`). The `rl-renderPM` package (v4.0.3) used by `reportlab` for PNG rendering ships pre-compiled C extension wheels only up to Python 3.12, and the alternative backends (`rlPyCairo` via `py-cairo`/`cairocffi`) also fail because they require system-level `libcairo2`. The fix builds `rl-renderPM` from source for Python 3.13 during CI and proactively verifies the renderPM backend at module import time to cleanly activate the SVG fallback without per-chart error tracebacks.
#1530	bug - Email Delivery Configuration Fails Validation When Email Security Is Set to None	Fixed a bug where setting the **Email security** field to `none` when adding or editing an Email Delivery configuration triggered a validation error (`ssl, tls, or none`), preventing the configuration from being saved. The root cause was a redundant regex validator on the `email_security` singleSelect field in `globalConfig.json`. Since the dropdown already constrains allowed values to `ssl`, `tls`, and `none`, the regex validator was removed.
#1531	bug - dsm/dhm - Custom Per-Entity Impact Score Weights Not Persisting Reliably	Fixed a critical bug in the persistent field handler (`trackmepersistentfields`) where custom impact score weights configured per-entity via the lag monitoring policy modal were intermittently lost, reverting to inherited tenant-level defaults. The root cause was in the conflict-update logic: when a user save overlapped with a tracker run, the tracker correctly read fresh KV store values into a `summary_record` but never applied them — the stale `record` from the search pipeline was what got saved via `batch_save`, silently overwriting user changes. The fix ensures conflict-update preservation writes directly to `record`, and persistent fields present in the KV store but missing from the search result are explicitly carried forward. Affects all components using `trackmepersistentfields` (DSM, DHM, MHM, FLX, FQM, WLK).
#1533	bug - dsm/dhm - Threshold Simulation Returns Incomplete Search Results	Fixed an issue where the DSM/DHM threshold simulation screen returned incomplete search results when simulating latency and delay thresholds against entity historical data. The React simulation modal (`EntityLagSimulationModal`) was not specifying `count=0` when subscribing to search results via `@splunk/search-job`, causing the SDK to default to only the first 100 results. This led to truncated chart data (simulation timechart with `bins=1000` showing only 100 points), truncated events tables, and potentially incorrect breach counts. All search job subscriptions now consistently pass `count: 0`. Additionally, several other search subscriptions in overview tabs (`OverviewTabWlk`, `OverviewTabFlx`, `MetricsTabFqm`) were fixed preventively.
#1534	bug - Outliers Anomaly Detection UI Does Not Indicate When Outliers Detection Is Disabled at the Entity Level	Fixed a UX inconsistency where navigating to the "Outliers anomaly detection" tab for an entity with disabled outliers detection showed no indication of the disabled state — charts were empty and the "Manage outliers detection" button remained accessible but non-functional. The fix adds a clear informational message when outliers detection is disabled, offers a direct option to re-enable it, prevents loading of outliers stats/charts when detection is disabled, and hides the "Manage outliers detection" button and related configuration controls.
#1536	bug - Intermittent React Crash on Chart Rendering When Navigating Back From Fullscreen Slider	Fixed an intermittent React crash (`TypeError: Cannot read properties of null (reading 'isOutlier')` / `Minified React error #185`) that occurred when clicking the "Back to tenant home" button in

11.1.43 Version 2.3.8 - build 1770827560 (11/02/2026)

Hint

TrackMe 2.3.8 — Performance, UI Modernization & New Capabilities

- This release delivers significant performance optimizations, new UI capabilities, AI-ready REST API endpoints, and important bug fixes.
- **Performance**: Major improvements eliminate redundant REST loopback calls and subsearch macro overhead, significantly reducing latency and search slot consumption across all UI operations.
- **New React UIs**: The Logs Inspector and Splunk Remote Accounts Status Overview dashboards have been fully migrated from XML to native React, providing richer visualizations and consistent theming.
- **AI Agent Integration**: New REST API endpoints under `/trackme/v2/describe` provide comprehensive, structured entity descriptions designed for AI agent consumption, enabling automated investigation and triage.
- **Flex Objects**: A new graphical configuration panel for tracker options simplifies hybrid tracker creation, and a critical fix resolves stuck `status_not_met` anomalies.
- **Bulk Actions & UX**: Improved bulk action feedback, new search capabilities in hybrid tracker management, and enhanced column sorting across Virtual Tenants UI.
- Several additional bug fixes address stateful alerting, global notes consistency, Bank Holidays Admin, and maintenance routines for disabled tenants.

- SHA256: 8d44f23482f8707755c830b553e5e1c9e43285669b33c5e5b0eb50f8cb9bbb3b

Table 117: New Features

Issue Number	Description	Details
#1520	feature - Flex Objects (splkflx) - Graphical Tracker Options UI for Hybrid Tracker Creation	Added a graphical configuration panel in the Flex Objects hybrid tracker creation wizard (Step 4 - "Test and Review") that allows users to visually define tracker options instead of manually writing `| eval` statements in their search query. The panel provides UI controls for: default metric (select dropdown populated from detected metrics), default thresholds (dynamic rows with metric, operator, value, condition), max sec inactive, disruption min time, drilldown search with earliest/latest, and outliers metrics configuration. Metric-dependent dropdowns are auto-populated from simulation results, and existing eval statements from use case library searches are pre-populated in the UI. A "Re-simulate with options" button lets users validate their configuration before tracker creation. Fully backward-compatible: if users do not interact with the options panel, behavior is identical to before.
#1521	feature - Replace Logs Inspector Dashboard with Native React UI	Replaced the existing XML-based logs inspector dashboard (`trackMe_commands_log_inspector.xml`) with a native React UI built on Splunk UI components. The new interface provides richer visualizations (pie charts for log level distribution, stacked column charts over time), better filtering and navigation (time range picker, log source selector, log level filter, keyword search), consistent theming with the rest of TrackMe React pages (dark/light mode via cookie preferences), "Open in Search" actions for deeper investigation, and support for all 3 TrackMe log sources: Custom Commands, REST API, and Modular Alerts.
#1522	feature - Entity Description REST API Endpoints for AI Agent Integration	Introduced new REST API endpoints under `/trackme/v2/describe` that provide comprehensive, structured descriptions of TrackMe entities designed for AI agent consumption. `POST /trackme/v2/describe/entity` returns a full, AI-consumable description of any entity across all 6 types (DSM, DHM, MHM, FLX, FQM, WLK) in a single call, including identity, entity info, health (state, anomaly reasons, score breakdown), configuration, metrics summary, and dynamically built SPL investigation searches. `POST /trackme/v2/describe/entities_summary` returns a lightweight listing of entities for discovery and triage with state counts, optional filters, and per-entity state information. This is the foundational step toward AI-powered entity investigation and triage capabilities.
#1524	feature - Migrate Splunk Remote Accounts Status Overview to React Native UI	Replaced the existing XML dashboard "TrackMe - Splunk Remote Accounts Status Overview" with a native React UI built on the Splunk UI toolkit. The new React page preserves all existing functionality with tab-based navigation: Overview Statuses (connected/failing accounts, donut chart, detailed tables), Token Rotation Statuses (color-coded status table), and Token Rotation Logs (events viewer for past 30 days). Includes status filter dropdown, Show Detailed Config toggle, dedicated refresh button, and full dark/light mode support consistent with TrackMe UI preferences.

Table 118: Enhancements & Changes

Issue Number	Description	Details
#1501	enhancement - Search Functionality for Hybrid Tracker Management	Added a powerful search capability to all hybrid tracker management screens, making it easier to find and manage trackers. All hybrid tracker management modals now include a search bar that allows you to filter trackers in real-time. Simply type in the search box to instantly filter the tracker list.
#1502	enhancement - TrackMe Health Tracker - SLA Breaches Events Generation Optimization	Optimized the SLA breaches events generation task to improve performance for tenants who have disabled SLA breaches events generation. The `gen_sla_breaches_events` task now skips execution entirely when SLA breaches events generation is disabled at the tenant level, reducing unnecessary processing overhead.
#1503	enhancement - Bulk Actions - Bulk Period Exclusion for Outliers Detection	Added a powerful bulk action that allows you to add exclusion periods to multiple entities and their ML models simultaneously, streamlining the management of Machine Learning Outlier Detection exclusion periods. A new bulk action "Add a period of exclusions" is now available in the **Outliers Detection Actions** category. This action allows you to select multiple entities and apply the same exclusion period to all of them and their corresponding ML models in a single operation.
#1504	enhancement - StateFul Alerting - Optional Email Security (SSL/TLS) for Email Delivery Accounts	Made email security (SSL/TLS) optional for email delivery accounts, giving you more flexibility when configuring email delivery gateways. The email security field in email delivery account configuration is now optional, allowing you to configure email delivery without SSL or TLS encryption when needed.
#1506	enhancement - Bulk Actions - Improved User Experience for Bulk Actions Impact Scoring	Improved the user experience for bulk actions that process entities in the background, providing immediate feedback so you can continue working without waiting. When you trigger bulk actions for "Set as false positive" or "Manually influence the score", you'll now receive immediate feedback that the action is running in the background. The system shows an immediate notification (toast message), closes the modal immediately so you can continue working, processes entities sequentially in the background without blocking your workflow, and notifies you with a summary when all processing is finished. This enhancement applies to the following bulk actions in the **Impact Score** category: "Set as false positive", "Manually influence the score", and "Set outliers as false positive" (Outliers Detection Actions).
#1507	enhancement - Virtual Tenants UI - Column Sorting Enhancement	Added column sorting functionality for numerical columns in the Virtual Tenants UI, making it easier to analyze and compare data in the Scheduler Review and Operational Health Status views. Numerical columns in the Virtual Tenants UI tables now support sorting with visual indicators (up/down arrows), matching the sorting functionality available in the Entity Overview table.
#1508	enhancement - Central KVcollection Mode Configuration	Added support for configurable modes for querying KVstore collections, allowing administrators to optimize performance based on their environment and requirements. A new configuration parameter `central_kvcollection_mode` has been added to the General settings, allowing you to choose between two modes: **search_mode** (default) - uses Splunk search commands to query KVstore collections, executes one search query per operation, stable and well-tested approach, recommended for most environments; **python_mode** - uses the Python SDK to directly access KVstore collections, no Splunk searches executed, may provide better performance for very large collections, uses direct KVstore API access.
#1510	enhancement - Tenant-Level Global Note Override	Virtual Tenants can now override the system-wide global note and link with tenant-specific values, allowing different tenants to display different global documentation while maintaining the flexibility to fall back to system-wide defaults. Each

Table 119: Bug Fixes

Issue Number	Description	Details
#1505	bug - StateFul Alerting - Fix for log_warn AttributeError in Stateful Alert Helper	Fixed a critical bug where the `trackmestateful` alert action would fail with an AttributeError: object has no attribute 'log_warn'. This error occurred whenever the alert action attempted to log warning messages, causing alert actions to fail unexpectedly. The code was using an incorrect method name (`log_warn`) that doesn't exist on the Splunk modular alert helper object. The correct method name is `log_warning`. Fixed all 36 instances of the incorrect method call throughout the stateful alert helper file, replacing `helper.log_warn()` with `helper.log_warning()`. Alert actions now correctly log warnings and continue processing normally.
#1509	bug - Global Notes Now Available Across All Components	Fixed an issue where global notes configured in the TrackMe Configuration page were only appearing in the DSM (Data Sources Monitoring) component, despite the configuration UI indicating they should apply to "all entities". Global notes and links are now correctly displayed across all TrackMe components: DSM (Data Sources Monitoring), DHM (Data Hosts Monitoring), FQM (Fields Quality Monitoring), FLX (Flex Objects Tracking), WLK (Workload Monitoring), and MHM (Metrics Hosts Monitoring).
#1512	bug - General Health Manager - Stateful Records Maintenance Routines Skip Disabled Tenants	Fixed an issue where the general health manager maintenance routines for stateful records did not check if a tenant is disabled before attempting to operate on their KVstore collections, causing errors for disabled tenants. The root cause was that maintenance loops iterate over all tenants including disabled ones. The following tasks in `trackmegeneralhealthmanager.py` now correctly skip disabled tenants: `virtual_tenants:stateful_alerts_records_expiration`, `virtual_tenants:stateful_alerts_duplicate_opened_incidents_c` and `virtual_tenants:stateful_charts_records_expiration`.
#1513	bug - Flex Objects (splk-flx) - status_not_met Can Remain Stuck in KVstore Preventing Entity Recovery	Fixed an issue where the `status_not_met` anomaly reason in Flex Objects tracking could become permanently stuck in the KVstore, causing entities to remain in an alerting state (red) even when the upstream search returns `status = 1` (green), all thresholds are satisfied, and the UI on-the-fly calculation shows green. The root causes included stale `original_upstream_status` determination from previous runs, a safeguard that blocked green state recovery, and lack of explicit clearing of `status_not_met`. The fix ensures that `status_not_met` is explicitly cleared when the upstream status is good, prioritizes fresh status from search results over stale KVstore data, and improves the safeguard logic. Note: The threshold-based approach is now preferred over the legacy `status` field.
#1514	bug - Bank Holidays Admin - Helper Function Naming Collision Causes REST API Errors	Fixed a bug where a helper function in the Bank Holidays Admin REST handler had a naming collision with the REST routing pattern, causing API errors. The REST framework routes requests by constructing function names from `{http_method}_{path}`, and the internal helper function `delete_maintenance_kdb_record` was being incorrectly matched as a REST endpoint. The fix renames the helper function with an underscore prefix (`_delete_maintenance_kdb_record`), following Python convention for private methods and preventing the REST router from matching it as an endpoint.
#1519	bug - Hide Impact Score Sliders When Related Features Are Disabled at Tenant Level	Fixed an issue where the Manage Impact Score Configuration modal in the Tenant Home UI showed all impact score sliders regardless of whether the related features were enabled at the tenant level. The **Outliers - Default Impact Score** slider is now hidden when Machine Learning is disabled for the tenant, and

11.1.44 Version 2.3.7 - build 1769984652 (01/02/2026)

Hint

Hotfix Release - Logical Group Health Calculation Fix

- This hotfix release addresses a critical issue with logical group health percentage calculations and entity protection.
- Fixed an issue where logical groups were not correctly calculating health percentages and protecting entities when the group remained healthy.
- This fix ensures accurate health percentage calculations for logical groups and correct entity state display (blue when protected by healthy group).

- SHA256: 8df79010be52a457a8db49d38b130c17f1761669ce3b698e026134b3e2a47b65

Table 120: Fixed Issues

Issue Number	Description	Details
#1495	bug - Logical Groups - Logical groups are not correctly calculating health percentages and protecting entities when the group remained healthy	Fixed an issue where logical groups with multiple members sometimes showed incorrect health percentages, causing false alerts even when the group was actually healthy. Improved handling of logical group member lists to ensure accurate percentage calculations regardless of how the data is stored. Logical group protection is now correctly applied after all state evaluations, ensuring entities are properly protected when the group remains healthy. Entities that should be protected by a healthy logical group (shown as blue) were incorrectly remaining red, especially when entities became unhealthy due to impact scores. This fix affects all entities that are members of logical groups, ensuring accurate health percentage calculations, correct entity state display, and accurate logical group status messages.
#1496	enhancement - improves the reliability and resilience of the TrackMe Health Tracker by implementing automatic retry logic for transient connection errors when communicating with the Splunk REST API	Improved the reliability and resilience of the TrackMe Health Tracker by implementing automatic retry logic for transient connection errors when communicating with the Splunk REST API. The Health Tracker now automatically retries failed connections to Splunkd with exponential backoff, reducing false failure reports caused by temporary network issues or Splunkd service interruptions. Key improvements include faster error detection (reduced timeout from 10 minutes to 1 minute), better error handling and logging, and improved connection stability.
#1497	bug - Virtual Tenants UI Bug Fix for "Open in search all skipping events"	Fixed an issue where the "Open in search all skipping events" button in the Virtual Tenants Scheduler Review modal was using an incorrect search query. The button now correctly searches for TrackMe scheduler events with proper status normalization. The search query now correctly targets the *scheduler* sourcetype instead of *splunkd*, includes proper filtering for TrackMe app events, and normalizes status values for consistent reporting (completed, skipped, deferred).
#1498	enhancement - Table Sort Arrow Indicators	Enhanced the visual feedback for sortable columns in the Virtual Tenants entity overview and Tenants UI entity tables. Sort arrows now clearly indicate the current sort direction: ↑ (Up arrow) for ascending order, ↓ (Down arrow) for descending order, and ↑↓ (Double arrow) for sortable but not currently active columns. This makes it easier to understand which column is currently sorted and in which direction, improving the overall user experience when working with entity tables.
#1499	enhancement - SOAR Integration Error Handling Improvements	Significant improvements to error handling and reliability for the SOAR (Splunk SOAR) integration in TrackMe. The integration now includes automatic retry logic with exponential backoff for failed requests, making it more resilient to transient network issues, API timeouts, and temporary SOAR service unavailability. All SOAR API requests now have timeout protection (60 seconds by default), preventing requests from hanging indefinitely. When automation brokers are temporarily offline, the system now skips affected assets with a warning instead of failing the entire operation, with automatic retry on the next run when brokers become available. Error messages now include more context (endpoint, server, request details) making it easier to diagnose issues. The integration now validates SOAR API responses before processing them, preventing errors from malformed or unexpected response formats.

11.1.45 Version 2.3.6 - build 1769529938 (27/01/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

Hint

Impact-Based Alerting (IBA), a new scoring-driven alerting model now powering TrackMe.

- IBA replaces rigid, hard-coded alerting rules with a flexible and configurable impact scoring system, inspired by security-grade risk-based approaches and adapted to data reliability and observability.
- With IBA, TrackMe now: - Aggregates anomaly impact over time instead of reacting to isolated signals - Allows fine-grained tuning of how different anomalies contribute to entity health - Provides full transparency with detailed score breakdowns and traceable score events - Significantly reduces alert noise while preserving operational confidence
- The transition is seamless for existing users, requires no manual intervention, and is fully configurable through the UI.

- SHA256: 56a4143e40392b4767f9405837f0e1ab4ee53ce2cede21551ef56e5321757f1d

Table 121: Fixed Issues

Issue Number	Description	Details
#1464	bug - Tenant Home UI - Impact Score Configuration Update Fix when the tenant has deprecated parameters	Fixed an issue where Impact Score Configuration updates failed with the error "Argument 'data_sampling_set_state' is not supported by this handler" for tenants with deprecated configuration fields. The update endpoint now automatically filters out deprecated fields before updating tenant configurations.
#1465	bug - Impact Score Weight Zero Now Properly Handles Entity Status	Fixed an issue where entities with impact score weights configured to *0* incorrectly remained red instead of being green. The decision maker process now consistently checks score components when *total_score == 0*, correctly setting entity status to green when all components have a score of *0*.
#1466	bug - Bulk threshold updates for Flex Objects (FLX) and Flex Query Manager (FQM) entities were failing	Fixed bulk threshold updates for FLX and FQM entities that were failing with "The component is required" error. The API payload structure now correctly includes the component identifier and properly formats threshold fields according to backend requirements.
#1467	bug - Flex Objects/Fields Quality (splk-flx/fqm) - thresholds created through the FLX and FQM threshold management screens were not functioning correctly	Fixed an issue where thresholds created through the FLX and FQM threshold management screens were not functioning correctly due to incorrect metric name formatting in API calls.
#1470	bug/enhancement - Tags Bulk Edit Improvements	Fixed 404 errors for bulk editing tags for FLX, FQM, WLK, DHM, and MHM components that were incorrectly calling the DSM-specific endpoint. All components now use their correct endpoints and provide clearer, more flexible tag management options.
#1471	bug - Stateful Alert Email Account Retrieval Bug	Fixed an issue where stateful alerts configured with "ingest_only" delivery mode (or other non-email modes) would fail during initialization when attempting to retrieve email account settings. The alert initialization now only retrieves email account settings when email delivery is actually enabled, and continues with other delivery types if email account retrieval fails.
#1472	bug - Flex Objects - Converging Tracker Status Recovery Issue	Fixed a bug where converging FLX trackers would not recover from red (alerting) state back to green state, even when the percentage of availability (*pct_availability*) reached 100% and all underlying entities were healthy.
#1482	bug - Remove misleading warning when a brand-new entity is first discovered before its KVstore entry exists	Fixed misleading warning when a brand-new entity is first discovered before its KVstore entry exists. When a record has no KVstore entry and *record_is_new* is true, the system now logs a debug message and skips the rejected-record check, avoiding unnecessary warnings for new entities.
#1483	bug - Fix Schedule Configuration Display for Stateful Alerts	Fixed an issue where the Schedule Configuration section (containing Cron Schedule, Suppress fields, Suppress period, and Day time filtering) was incorrectly displayed in the "Various" step when creating stateful alerts. Stateful alerts now correctly hide this section as they use a fixed, non-configurable schedule (**/5 * * * **).
#1486	bug - Virtual Tenants UI - Resolved intermittent flash when opening the Virtual Tenant creation wizard	Resolved an intermittent flash when opening the Virtual Tenant creation wizard by preventing immediate close on the initial click.
#1488	bug - Fields Quality (fqm) - Quality Tracker Creation Fails for Remote Targets	Fixed an issue where creating a new quality tracker using sampling mode with a remote target would fail. The "Benchmark sampling mode" button was incorrectly disabled for remote targets, and tracker creation would fail with an API error stating that *collect_limiter* must be greater than 0. The application incorrectly assumed that sampling mode was not supported for remote targets, even though sampling mode is fully supported.
#1489	bug - Standardized Splunk URL generation in Virtual	Fixed "Open in Search" links in the FQM wizard and standardized Splunk URL generation in Virtual Tenants and Ten-

Table 122: Enhancements, changes and new features

Issue Number	Description	Details
#1468	enhancement - Enhanced Bulk Edit Actions for Outliers Detection Rules	Expanded bulk edit functionality for Outliers Detection Rules, allowing you to update all configuration settings in bulk operations. This enhancement brings full parity between bulk edit and individual model settings, making it much easier to manage outlier detection rules across multiple entities.
#1469	enhancement - Update Maintenance Period Feature	Added ability to update (extend or reduce) the maintenance period directly from the Maintenance Mode UI without needing to disable and re-enable maintenance mode. The feature automatically updates both the maintenance mode status and the associated knowledge database record.
#1474	enhancement/feature - Entity Actions and Search Enhancements	Improved entity actions menu with clearer labeling: the "Open" action has been renamed to "Open entity" with a new icon to better indicate it opens the entity detail view. For DSM and DHM components, a new "Open in search" action is now available directly in the actions menu, allowing users to quickly open the entity's search query in Splunk Search.
#1475	enhancement - Time Range Picker Persistence	The time range picker in entity modal views now remembers your selection as you navigate between different entities. Your chosen time range for each tab is preserved throughout your session until you reload the page, eliminating the need to manually adjust the time range for each entity you view.
#1476	enhancement - FLX Entity Overview - "ALL" Option Enhancement	Added a convenient "ALL" option to the entity selection dropdown in the FLX component's entity overview modal. This allows you to quickly view metrics for all available entities without manually selecting each one individually. Selecting this option displays charts and statistics for all entities that match your selected metrics and pre-filter criteria.
#1477	enhancement - Non-Disruptive Refresh for Overview Tabs	Improved user experience when refreshing data in the Overview entity modal for FLX and FQM components. Components now remain visible during refresh operations, eliminating the disruptive full-page spinner that previously appeared. Only individual loading indicators update during refresh, providing a smoother, less disruptive experience.
#1478	enhancement - Flex Objects - Multiple Default Metrics Support for Flex Objects	TrackMe now supports defining multiple default metrics for a single Flex Object tracker. You can select multiple default metrics for each tracker using a multi-select dropdown, allowing you to preset multiple metrics in the metric selector when viewing entities managed by a tracker. The default metrics management interface displays one row per tracker with all associated metrics shown in a multi-select component, making it easier to manage configurations for multiple trackers.
#1479	enhancement - Flex Objects (flx) - Threshold Management UI Improvements	Enhanced the threshold management interface for FLX and FQM components to make it easier to understand and configure thresholds. These improvements address user feedback about UI complexity and provide clearer, more intuitive threshold configuration.
#1480	enhancement - Configurable Auto-Refresh Interval for Action Progress Events	Introduced a configurable auto-refresh interval for action progress event logs. Administrators can now adjust the refresh rate (5-300 seconds) based on their system's performance characteristics. The default value is 15 seconds (less aggressive than the previous hardcoded 5 seconds). Polling behavior remains identical to the previous implementation - only the interval value is now configurable.
#1481	enhancement - Benchmark Race Condition Fix	Fixed benchmark (burn test) execution failures in large Search Head Cluster (SHC) environments. After creating temporary reports for burn tests, the system now waits longer before executing them to account for knowledge object propagation delays in large/complex SHC environments, preventing "Unable to find saved search" errors.
#1484	enhancement - TrackMe	Optimized the *unclosed_stateful_incidents* health tracker task to

11.1.46 Version 2.3.5 - build 1768473196 (15/01/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

Hint

The highlight of this release is the introduction of Impact-Based Alerting (IBA), a new scoring-driven alerting model now powering TrackMe.

- IBA replaces rigid, hard-coded alerting rules with a flexible and configurable impact scoring system, inspired by security-grade risk-based approaches and adapted to data reliability and observability.
- With IBA, TrackMe now: - Aggregates anomaly impact over time instead of reacting to isolated signals - Allows fine-grained tuning of how different anomalies contribute to entity health - Provides full transparency with detailed score breakdowns and traceable score events - Significantly reduces alert noise while preserving operational confidence
- The transition is seamless for existing users, requires no manual intervention, and is fully configurable through the UI.

- SHA256: d2484bb1cdb13245b3a07097c899274bba1f09688c55b92771c9245b37b5b0d0

Table 123: Fixed Issues

Issue Number	Description	Details
#1406	bug - ML Outliers Detection - DSM and DHM Components KPI Metrics Dropdown	Fixed an issue where the KPI metric dropdown was empty when adding Machine Learning models for outliers detection in the Data Source Monitoring (DSM) component.
#1410	bug - Flex Objects/Fields Quality (flx/fqm) - HTTP 500: records_list must be a list of records when trying to update a threshold record	Trying to update existing threshold records through the UI fails with HTTP error 500.
#1415	bug - Fix typos, mispells or non ideal English sentences in alert action labels	Fixed various typos or spelling issues in alert actions.
#1416	bug - Fix: Error Panel Not Clearing After Service Recovery	When the Splunk service becomes unavailable or returns an error response, the application correctly detects and displays an error panel. However, once the service recovers and requests start succeeding again, the error panel remains visible even though the issue has been resolved.
#1417	bug - Fix Data Sampling default in DSM tenant creation wizard	The option “Enable Data Sampling” should be configurable in the tenant advanced options at the stage of the creation wizard.
#1425	bug - Fix: FLX Component Cartouche Text Overflow Issue	Fixed a UI bug where long group names in the FLX component cartouche could overflow and overlap with adjacent items. The fix ensures text wraps properly across multiple lines when needed.
#1426	bug - Flex Objects library - syntax error in UC splk_detect_drop_events_count	There is a syntax error in the UC splk_detect_drop_events_count_absolute
#1440	bug - Fix missing threshold lines in Performance Metrics charts for DSM and DHM	Fixed a bug where threshold-based performance metric charts were not displaying threshold lines. The search queries for latency and delay metrics with threshold options now correctly include threshold values from entity records (data_max_lag_allowed and data_max_delay_allowed). Additionally, removed unnecessary timechart parameters that were not needed for these queries.
#1445	bug - Virtual Tenants UI - wrong default tstats root time span values in Virtual Tenants UI and partial support of advanced options	The Virtual Tenants UI hybrid tracker creation wizards for DSM and DHM now properly honor all user selections and use consistent default values matching the Tenant Home UI.
#1446	bug - Data Sources monitoring (dsm) - Fixed incorrect entity naming convention in merged mode for delayed inspector	Fixed a bug where entities with an incorrect naming convention (@all without colon) were being created when the delayed inspector processed offline entities in merged mode. The fix ensures that existing entities using the legacy :all convention continue to use that convention, while new entities correctly use the :@all convention. This maintains full backward compatibility and prevents data integrity issues.
#1454	bug - Improved ML Outliers User Experience - Removed Confusing Error Messages for Empty States	Fixed an issue where an error toast message was incorrectly displayed when accessing the ML Outliers Anomaly Detection section for entities that don’t have any ML models defined yet. This is a normal scenario when ML models haven’t been created for an entity, and the UI now correctly displays the empty state message without showing an error. Actual errors (such as network failures or server errors) will still display error messages as expected, ensuring users are properly notified of real issues.
#1460	bug - Virtual Tenants/Tenants Home - Auto-Refresh Pre-Selection	Fixed auto-refresh setting not showing current value in Settings panel. The auto-refresh setting in the Settings panel (available in both Virtual Tenants and Tenant Home UIs) now correctly displays the current value when the panel is opened. Previously, neither “Enabled” nor “Disabled” was highlighted, even when a preference was set. The setting now properly shows the user’s saved preference (if set), system default value (if no user pref-

Table 124: Enhancements, changes and new features

Issue Number	Description	Details
#1405	feature - TrackMe transition to impact score basis anlysis	TrackMe introduces Impact Scoring, a major milestone that makes alerting smarter, tunable, and far more actionable. Administrators can fine-tune impact per tenant/component/anomaly through a rich UI, reduce false positives with one-click suppression, and keep full transparency with a clear score breakdown and score events traceability.
#1407	change - remove remaining artefacts of icons_state_mode	There are some remaining artefacts of the option icons_state_mode, which is decomissioned since TrackMe 2.3.x.
#1408	enhancement - Updates all Outliers management related logic to work out with the object_id rather than object as a convention	Historically, all TrackMe related commands and logics dealing with Outliers detection management rely on the object value rather than its unique identifier, which can eventually be less robust when facing complex object string structures. These changes prevent any issue at this level.
#1412	enhancement - Flex Objects / Fields Quality (flx/fqm) - Enhanced Threshold Management with Dynamic Metric Selection and Improved Change Tracking	The threshold management screens for FLX and FQM entities have been significantly improved with a better user experience. Users can now select metrics from a searchable dropdown instead of typing them manually, receive clear notifications when they have unsaved changes, and get immediate feedback through toast notifications. The modal has also been widened to provide more space for viewing and editing thresholds. Additionally, critical bugs preventing threshold updates have been fixed.
#1413	enhancement - bulk edit for lagging policies - allow human time patterns similarly to per entity edit	This enhancement allows bulk edit to accept and automatically translate human based time patterns, in the same way we do in the per entity edit screen.
#1414	enhancement - Virtual Tenants / Tenant Home UI - match color code for the right hande side pie charts with their respective label	For UI consistency purposes, we should match the color scheme with their associated label
#1418	change - decomission red on outliers and red on sampling following the scoring implementation	With the implementation of configurable impact scoring, entity status (green, orange, red, blue) is now determined by aggregated impact scores from various anomaly types. Users can control the score attributed to outliers and data sampling anomalies through the impact scoring configuration. Therefore, Red on Outliers and Red on Sampling options are now decomissionned.
#1419	enhancement - ML Train and ML Monitor Now Support Sequential Bulk Processing	The ML Train and ML Monitor API endpoints have been updated to accept lists of entities and process them sequentially instead of concurrently. This change provides better resource management and prevents resource contention when processing multiple entities.
#1421	feature - Enhanced Table Customization	Users can now resize table columns by dragging column borders and reorder columns via drag-and-drop in the “Show/Hide Columns” menu. All preferences are automatically saved per tenant and component for a personalized experience.
#1422	enhancement - Enhanced Tag Grouping in Entity Tables	We’ve improved how entities are grouped by tags in the entity overview tables. Entities with multiple tags now appear in each individual tag group, making it easier to discover and organize entities by their tags.
#1423	enhancement - Flex Objects (flx) - handle the epochtime conversion of the Tracker Runtime into human readable format	The cartouche shows the Tracker Runtime in epochtime format, but should rather be shown in human readable format.
#1427	enhancement - Add “Open in Search” Button for Failed Search Simulations in FLX Hybrid Tracker Wizards	This enhancement adds an “Open in Search” button that appears when search simulations fail in the FLX hybrid tracker creation wizards. This allows users to manually execute the search query in Splunk’s search UI to debug syntax errors, execution failures, or missing mandatory fields.

11.1.47 Version 2.3.4 - build 1766484049 (23/12/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

- SHA256: 44db7515f3ca64a7c5ed63895633e277d2b16eef96490f324b04bc6695b279d4

Table 125: Fixed Issues

Issue Number	Description	Details
#1388	bug - Tenant Home UI - Alert creation dropdown provides invalid options for acknowledgment	The alert creation modal was providing invalid options for the acknowledgment dropdown. Fixed to display only valid options: sticky and unsticky.
#1390	bug - Tenant Home UI - Performance issues in logical groups management screens with high number of groups	When managing a large number of logical groups, the management screens experienced performance issues that could freeze the browser window. This fix introduces automated pagination to address the performance problem.
#1396	bug - Tenant Home - Select all checkbox inconsistency behavior does not actually select all matching entities	The “select all” checkbox in the Tenant Home Entities Overview table only selected records visible in the viewport instead of all records matching the active filter or search. Fixed by updating the selection logic to use all filtered records, ensuring bulk operations work correctly with filtered result sets.

Table 126: Enhancements, changes and new features

Issue Number	Description	Details
#1389	enhancement - Command trackmegetcoll - Ensure tenant_id value cannot be null or empty	The trackmegetcoll command is used in various concepts such as alerts. This update ensures the tenant_id value in a record cannot be null or empty, preventing tenant_id from being rendered as null or empty in case of anomalies.
#1391	enhancement - Expandable Charts for Scheduler and Ops Status Tables	Adds expandable rows to the Scheduler Review and Tenants Operational Health modals, showing two charts per report: Runtime Statistics Over Time (line chart) and Scheduler Execution Over Time (column chart). This enhancement allows quick access to runtime metrics and scheduler views for a particular search without leaving the user interface.
#1392	enhancement - Add refresh icon shortcut in entity full screen view and normalize entity views buttons layout	This enhancement adds a refresh icon in the entity view next to the main selector. It also normalizes button styles and layout for UI consistency purposes, including standardizing the slide panels back button.
#1393	feature - Stateful alerting refactoring with pre-filtering custom command and single alert enforcement	Stateful alert events are now pre-filtered before they reach the alert action backend, improving performance and reducing unnecessary processing. The command automatically retrieves the stateful alert configuration (including "orange as alerting state" setting) to ensure consistent filtering. Additionally, the UI now prevents users from creating multiple stateful alerts per tenant, enforcing a one-to-one relationship between tenants and stateful alerts.
#1394	enhancement - Tenant Home UI - Improve user interface behavior and consistency for automated and manual refreshing	In the Tenant Home UI, auto-refresh now pauses when the user is reviewing entity detailed statistics, preventing disruption once the entity slider view is opened. Additionally, the refresh action now properly refreshes entity cartouches, the primary screen, and the actual charts or views currently being inspected by the user.
#1395	feature - Bank Holidays Management	TrackMe administrators can now configure bank holiday periods that prevent alerts from triggering, similar to maintenance mode. Bank holidays apply globally to all tenants and can be set as one-time periods or recurring holidays that automatically repeat each year. The feature includes: an intuitive web interface for creating, editing, and deleting bank holiday periods with a modern React-based UI; calendar visualization with each holiday period displaying a mini calendar showing the month with highlighted dates; year organization with periods automatically grouped by year (oldest first) for easy navigation; country import for importing holidays for 40+ countries including US, UK, France, Germany, Italy, Spain, Canada, Australia, Japan, Brazil, and many more; recurring holidays that can be marked to automatically generate future occurrences; bulk operations to clear all periods at once with a single action; and automatic maintenance where the system automatically creates future occurrences and cleans up past periods for recurring holidays daily, ensuring bank holidays are always up-to-date without manual intervention.
#1397	enhancement - Hybrid trackers management - Automatically re-create hybrid tracker records if KVstore content was unexpectedly purged	The tenant health tracker now automatically handles the re-creation of hybrid tracker records if the central KVstore contains the hybrid trackers metadata, and these trackers exist in the central store but not in the dedicated KVstore of the tenant.
#1398	enhancement - Tenant Home UI - Automatically detect automated Ack conflicts when creating alerts	This enhancement automatically detects if a tenant (and component if applicable) already has an automated acknowledgment creation action when creating a stateful alert or vice versa when creating a legacy alert for the component. This prevents users from creating both a stateful alert with auto-acknowledgment enabled while an existing legacy technical alert exists, or creating a legacy component alert if there is an existing stateful alert that performs auto-acknowledgment. The system will inform

11.1.48 Version 2.3.3 - build 1765917730 (16/12/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

- SHA256: fb3624e3be08255b79b52ffa5501e6344805535d9a191d5bf8492e3aef9526f0

Table 127: Fixed Issues

Issue Number	Description	Details
#1355	bug - TrackMe Virtual Tenants UI - Remove the back button from RBAC update screen	There should not be a back button in the RBAC update screen.
#1359	bug - TrackMe Tenant Home UI - Wrong endpoint called for dhm when doing bulk edit lagging policy, leading to a 404	When performing bulk edits for lagging policy on the dhm component, the endpoint called is incorrect, which leads to a 404 error.
#1360	bug - Virtual Tenants UI - Long alias or description in the Virtual Tenant account can badly affect the cards layout	Fixed a layout bug in the Virtual Tenants cards where long tenant descriptions (or other long text) caused cards to expand beyond their width and overlap adjacent cards.
#1364	bug - TrackMe Tenant Home UI - Inconsistent results in the enabled entities count (first single view) in some cases	There is an issue in the calculation made by the UI for the first single view showing the number of enabled entities. In some contexts, the calculation is incorrect and leads to an invalid number.
#1366	bug - DB Connect poor quality and capability management can affect the SDK method service.confs["app"] in limited privileges mode	DB Connect is affecting applications calling service.confs["app"] SDK methods in non-privileged modes.
#1368	bug - Schema upgrade - Missing variable initialization leads to backup error message during schema upgrade	A missing initialization for a variable is leading to an exception during the schema upgrade for the backup verification.
#1369	bug - Remove outdated app dependency verification for semicircle_donut which is not required anymore in 2.3.x	The app dependency semicircle_donut is not required anymore for TrackMe 2.3.x.
#1374	bug - Tenant Home UI - Group by view in the table results gets unexpectedly reset to the default when accessing the top menu selector	When accessing the top selector menu while a different group by was selected, the group by is unexpectedly reset to the default.
#1375	bug - Virtual Tenants UI - Redundant API call to trackmeload and missing pre-settings for indexes and RBAC	Redundant API call: trackmeload was called twice — once in VirtualTenantsContentInner and once in VirtualTenantsContent (for theme), causing duplicate requests. Missing pre-settings: Tenant creation wizards didn't use trackmeconf from the trackmeload response to pre-set owner and RBAC defaults (indexes, owner, admin/power/user roles), so they used hardcoded values instead of the configured defaults.
#1382	bug - Open in search FQDN consistency	Some "Open in search" links in Tenant Home and Virtual Tenants UIs used inline make_url calls instead of the makeSplunkUrl helper, so they didn't respect the server FQDN and root URL context. Standardized all "Open in search" links to use makeSplunkUrl for consistent URL handling.

Table 128: Enhancements, changes and new features

Issue Number	Description	Details
#1354	enhancement - TrackMe Home UI - Add tooltip to entity table icons (Actions column)	This minor enhancement adds tooltips for UI consistency purposes to the icons in the Actions column.
#1356	feature - Virtual Tenants/Tenant Home UI - Add a “collapse all” / “expand all” shortcut icon in the table header	This enhancement adds a new icon action in the table header after the “All entities” dropdown, that allows collapsing / expanding all groups easily.
#1357	change - Upgrade all Splunk UI packages to latest releases	Updated Splunk UI packages to latest releases.
#1358	enhancement - Virtual Tenants UI - Missing drilldown in pie charts to match the Tenant Home behaviour	The Virtual Tenants UI and the overview tenant modal contain the same pie charts as the Tenant Home UI does. However, the screen was missing the drilldown chart actions. This enhancement updates the Virtual Tenants UI so it matches the Tenant Home behaviour.
#1361	enhancement - Virtual Tenants/Tenant Home UI - Improve rendering of the dhm Idx/st summary column	This is a minor improvement of the dhm column for Idx/st summary.
#1362	enhancement - Virtual Tenants/Tenant Home UIs - Prevent sort on summary / last metrics columns for the components where it does not make sense	In mhm, the “Summary” column should not be sortable. In flx/fqm, the “Last metrics” column should not be sortable. In wlk, the “Metrics summary (24h)” column should not be sortable.
#1363	enhancement - TrackMe Home UI - Add the kebab selector in single view cards of the main screen and when missing	Some single view cards were missing the kebab selector.
#1365	enhancement - Virtual Tenants/Tenant Home UIs - Minor user interface enhancements	Added refresh button with tooltip to TenantUpdateStatus modal, positioned in the top-right corner. Added refresh button with tooltip to Virtual Tenants main screen header, before the settings button. Added refresh button with tooltip to Tenant Home UI header, before the settings button. Added tooltips to settings and action buttons in both Virtual Tenants and Tenant Home UIs. Replaced remaining emoji/unicode icons with CSS dots.
#1370	change - Deprecating and removing the usage of the Python sparklines module which was only used by the SmartStatus action	We are deprecating and removing the usage of the sparklines module which was used in SmartStatus, and providing very low value.
#1371	change - Python compatibility - Prepare TrackMe for Python 3.13.x compatibility with upcoming Splunk releases	These changes are performed to handle TrackMe compatibility with the upcoming Splunk migration to Python 3.13.
#1372	enhancement - Virtual Tenants UI - Minor enhancements in the table views for update status and operation status	Minor improvements to the rendering of the tables.
#1373	enhancement - Schema upgrade - Safer and more robust identification of past backups in the last 24 hours during schema upgrade	Implemented a more robust identification of backup operations initiated or performed during the past 24 hours when the schema upgrade is triggered. This avoids triggering a backup during the schema upgrade if a backup was requested or completed recently.
#1376	enhancement - Stateful alerts - Adding a safety check REST call to the realtime decision maker to verify the object	Added a REST call to load_component so we can verify the object_state from realtime decision maker, before accepting the alerting status. Reduced the 60 minutes check that refuses creating a new incident if the last non closure event is not newer to

11.1.49 Version 2.3.2 - build 1765452718 (11/12/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

- SHA256: 358637a2c2bc04292df77f1af8fe2dd539214bcad5c9370c72a4bf02a60f5b78

Table 129: Fixed Issues

Issue Number	Description	Details
#1346	bug - TrackMe Tenant Home/Virtual Tenants - The table should show by default only enabled entities	By default, only the enabled (monitored_state="enabled") entities should be shown in the table for both UIs
#1347	bug - Tenant Home UI - Bulk edit priority management is missing the pending priority level	The bulk edit modal should allow handling the priority level "pending"
#1348	bug - Tenant Home UI - Wrong endpoint is used for some actions such as delete temporary and permanenlty for other components than dsm	The delete action from bulk and for some of the components is targeting a wrong endpoint, leading to a 404 error.
#1351	bug - Tenant Home UI - minor inconsistent checkbox layout in some modal screens	Very minor layout issues for the checkboxes in some modal screens.
#1352	bug - Tenant Home UI - Issue opening the data sampling add rule modal from the main screen menu instead of on a per entity basis	The link to "Manage: Data sampling custom rules" doesn't behave properly when opened from the main screen menu.
#1353	bug - Tenant Home - Missing advanced tools for lagging policy simulation and auto definition	These advanced tools have not been migrated from the legacy UI.

Table 130: Enhancements, changes and new features

Issue Number	Description	Details
#1343	enhancement - TrackMe Tenant Home - Adds a per entity selector in the table view providing quick access to entity edit options	This enhancement adds the kebab selector on a per entity basis.
#1344	enhancement - TrackMe Tenant Home/Virtual Tenants - Add the column monitored_state in the table	This enhancement re-adds the monitored state column in the table view
#1345	enhancement - TrackMe Tenant Home/Virtual Tenants - Sets all columns not based on filterable to be sortable	Ensure all columns show the sort header option
#1349	enhancement - Virtual Tenants / Tenant Home UIs - Adds a “Refresh” action in the table engine menu allowing to refresh the table content only	This enhancement adds a “Refresh” action underneath the “Open in search” action within the engine icon next to the table views.
#1350	enhancement - Virtual Tenants/Tenant Home UIs - Adds the “Group: Anomaly reason” in the shortcut menu for both user interfaces	The group by anomaly reason is a valuable option that needs to be re-added.

11.1.50 Version 2.3.1 - build 1765362371 (10/12/2025)

Hint

TrackMe 2.3 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

- SHA256: e2a26c6f06626bfa18f22279e338ea2512d406ddf1c31365190b562a623437c1

Table 131: Fixed Issues

Issue Number	Description	Details
#1340	bug - Tenant Home UI - inconsistent selector menu width in benchmark step from hybrid tracker creation wizard	The width of the kebab selector in the benchmark step for some components is not consistent and expands over the whole div of the container
#1341	bug - Rest API reference UI - usage of the search facility can lead to a critical uncatched exception in the UI	In the Rest API reference UI, a weakness in the code may lead to a critical failure of the user interface with a typescript error "TypeError: Cannot read properties of undefined (reading 'toLowerCase')"
#1342	bug - Tenant Home UI - Missing priority level selection multiselect for the creation of a stateful alert in the alert creation wizard	The alert creation wizard is missing the multiselect input to the selection of the priority levels for emails notifications. This is a regression from the React UI migration.

11.1.51 Version 2.3.0 - build 1765269590 (09/12/2025)

Hint

TrackMe 2.3.0 — Major UI Modernization Release

- We are pleased to introduce our fully redesigned user interfaces.
- This release represents a major step forward in performance, scalability, and user experience.
- With TrackMe 2.3.0, all of user interfaces are now built on top of Splunk UI native React.
- This is a complete and full rewrite of TrackMe main user interfaces, which we are excited to share with you.
- Many various improvements and enhancements were implemented to fully support the new interface, and to ensure a smooth transition from the old Splunk stack JS to a modern and native React UI.

- SHA256: 3ac6e7ef8f236a357f702bd44b91e3a0a78484e24164b136708379493beb3e2f

11.1.52 Version 2.2.4 - build 1764227551 (27/11/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 94120a98ec7f36abc8be4f1d6207136b209914c8d8cede60fcf02cd49ddc4b00

Table 132: Fixed Issues

Issue Number	Description	Details
#1323	bug - Metric hosts tracking - permanently deleted objects not honored	In splk-mhm, permanently deleted objects are not properly taken into account.
#1325	bug - Virtual Tenants UI (React) - pie charts are not refreshed when hitting the refresh all action, nor through auto-refresh	There is a logic issue in the React UI which leads to the pie charts for scheduler activity and tenants activity not updating on their own through auto-refresh or manual refresh.
#1331	bug - Virtual Tenants UI (React) - debug console log left active when expanding entities overview	A console log debug statement has been left active in the React Virtual Tenants UI.
#1332	bug - TrackMe Home UI (Legacy) - drilldown on single view for enabled entities should filter out on enabled entities only (all components)	The drilldown action for the “All enabled entities” should filter entities with monitored_state=”enabled”, but currently filters monitored_state=*.

Table 133: Enhancements, changes and new features

Issue Number	Description	Details
#1318	enhancement - Backup & Restore - Show the backup archive size in the user interface	The backend stores the size in bytes of the backup archive. This enhancement ensures this information is displayed in the summary table within the user interface.
#1319	enhancement - Backup & Restore - Add a per KVstore size summary that can be easily accessed through the UI	This enhancement adds the detail of the per KVstore size, easily accessible in the per backup details through the user interface.
#1320	enhancement - Virtual Tenants UI (React) - Additional safety to prevent tenant cards overlap at certain DPI/Zoom	This enhancement provides an additional safeguard to prevent tenant cards from overlapping at certain DPI/Zoom levels.
#1321	enhancement - TrackMe REST API - Prevents duplications in permanently deleted entities KVstore collections	When performing permanent deletion, the current implementation does not account for already existing records in the KVstore for the same object. This enhancement adds a verification and will bypass the creation of the deleted record if necessary.
#1322	enhancement - TrackMe Health Tracker - Adding a subtask to verify for duplicated records in the permanently deleted records collection and act if necessary	This enhancement adds a subtask in the main task inspect_collection:permanently_deleted_records_inspection which detects and purges any duplicated records based on the object value.
#1326	feature - Virtual Tenants creation and update wizards (React) - Adds a panel showing events creation so we provide better visibility of the operations being processed	This new feature provides an enhanced user experience when creating or updating Virtual Tenants by automatically adding an event view showing the logs associated while the operation progresses.
#1327	change - Decommission of the legacy Virtual Tenants UI	The legacy Virtual Tenants UI based on Splunk JS is now decommissioned and removed from TrackMe.
#1335	feature - TrackMe REST API - provide a merged endpoint to retrieve rules and data from a pure Python logic	This feature provides a new endpoint to get a merged summary of the data and rules for Outliers. It also enhances the endpoints for rules, and provides an additional data endpoint.
#1336	change - Hybrid trackers - For splk-dsm, sets the default time span to 30s rather than 1s	This minor change updates the default time span for splk-dsm trackers from 1s to 30s.
#1338	change - Configured webpack to build Splunk React UI components in production mode	Configured webpack to build Splunk React UI components in production mode per Splunk React UI documentation. This removes development warnings and guidance from production builds published to Splunk Base. Changes include: added webpack DefinePlugin to all webpack configs to inject process.env.NODE_ENV, removed devtool: 'eval-source-map' from production builds, and updated build/build.py to set NODE_ENV=production when building Splunk UI packages.

11.1.53 Version 2.2.3 - build 1762443437 (06/11/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 62a2250a099215174561d46474c4cb47ae48edb98586dc4e3455a9b4a3243366

Table 134: Fixed Issues

Issue Number	Description	Details
#1315	bug - Virtual Tenants UI (React) - regression when expanding an entity in the overview table	A regression was identified when expanding the entity from the Virtual Tenant UI overview.
#1316	bug - Virtual Tenants UI (React) - properly handle the API response if the tenant fails to be created so the response appears in the modal	Ensure to properly handle the response if the tenant fails to be created.
#13824	bug - regression in Backup & Restore UI when attempting to download depending in some contexts	Fixed a regression in the Backup & Restore UI where archive downloads failed due to missing validation of the archive_base64 field in the response. Added error handling for JSON parsing and base64 decoding, plus validation to ensure the response contains valid archive data before processing.

Table 135: Enhancements, changes and new features

Issue Number	Description	Details
#1317	change - trackmetrackerexecutor - Allows forcing the execution of a tracker through the savedsearch command using the argument force_savedsearch_execmode=′	This minor change allows forcing the execution of a tracker using the savedsearch command.

11.1.54 Version 2.2.2 - build 1762387662 (06/11/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 6273f390c19fd3e699a09d760cf9d19b919d9a854b9a4ab2a93493d2ad360177

Table 136: Fixed Issues

Issue Number	Description	Details
#1311	bug - Virtual Tenants UI (React) - the open in search metrics link generates a non-working search due to single quotes escaping	The open in search link for metrics at the tenant level does not work as intended due to single quotes escaping.
#1312	bug - Virtual Tenants UI (React) - Pagination issues in the overview table	There is an issue in the pagination of the table from the new Virtual Tenants user interface, which leads to stopping from generating all entities in the table. This update addresses these issues and implements options in the table header allowing to navigate through the pages, in addition with an automated scrolling if the user scrolls down the entities.

Table 137: Enhancements, changes and new features

Issue Number	Description	Details
#1310	change - AppInspect warning check_idx_binary_compatibilit	Address AppInspect warning.
#1313	enhancement - Virtual Tenants Overview: row virtualization	Implemented lightweight row virtualization with dynamic height measurement for expansion rows; DOM stays tiny and scrolling remains smooth even with 10k+ rows.
#1314	enhancement - Virtual Tenants UI (React) - Improved responsive tenants cards to avoid some issues in specific customer environments	These enhancements improve the responsive aspects of the Virtual Tenants cards, to address some potential UI glitches in customer environments with very high desktop scale.

11.1.55 Version 2.2.1 - build 1762134116 (03/11/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0931aba1c09903a63cceedad05a3452665e4a15b9d2e353b6192347f7872453b

Table 138: Fixed Issues

Issue Number	Description	Details
#1301	bug - Backup & Restore - potential failures in restoring TrackMe at the stage of the virtual accounts creation	Under some conditions, it is possible that TrackMe is unable to complete a restore operation when attempting to re-create the Virtual Tenant accounts, due to several inconsistencies in the restore code.
#1302	bug - TrackMe Health tracker - Virtual tenant account check and repair comes too late in the execution and is ineffective if the account does not exist	The TrackMe health tracker has a step that checks and repairs the virtual account if missing, however this comes too late in the code logic and we fail to continue in the early stage in this case.
#1304	bug - Virtual Tenants UI (React) - missing preview charts for ML Outliers	In the new Virtual Tenant UI, the overview panel should show the ML Outliers mini-charts when ML is enabled for the entity. Due to a silent Python exception, the endpoint does not render the charts definition currently.
#1305	bug - Virtual Tenants UI (React) - information panels should vary depending on the components	When expanding entities, the new Virtual Tenant shows entity key information in a nicely formatted manner. However, the current implementation is stuck to dsm/dhm and should vary depending on the components.
#1307	bug - Virtual Tenants UI (React) - prevents issues refreshing charts when collapsing and re-expanding twice the same entity	There is an issue affecting the mini-charts generation when collapsing and re-expanding the same entity twice.

Table 139: Enhancements, changes and new features

Issue Number	Description	Details
#1303	enhancement - Backup & Restore - Verify that the user context exists on the target during a restore operation, and fallback to nobody otherwise	When performing a restore operation, we would fail to properly perform the restore if the original owner context does not exist on the target. This enhancement ensures that we verify the user context, and if necessary we fallback to the nobody user context.
#1306	enhancement - Virtual Tenants (React) - ensure to show a wait spinner as soon as the user expands a given entity	When an entity is expanded, we should immediately show a wait spinner to inform the user that TrackMe is retrieving the entity information for the mini-charts queries to be executed.
#1308	enhancement - Virtual Tenants UI (React) - handle overview status message for converging Flex trackers to only keep down_entities in extra_attributes to prevent from ending with a massive large payload	This enhancement prevents from ending with a very large payload in the overview panel for converging Flex trackers.

11.1.56 Version 2.2.0 - build 1761787469 (30/10/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 2c2de97ac5befc07d178ab87ea4b692a5b34b29157fa2f68759c2c94b6bd832d

This major release introduces a brand new refactored user interface for the Virtual Tenants management built on top of Splunk UI native React.

Table 140: Fixed Issues

Issue Number	Description	Details
#1300	bug - mcatalog should include the object_id in stateful alerts, minor issue in the charts_resources	The mcatalog generation should include the object_id. The logic for the charts_resources is incorrect for wlk/flx/fqm.
#1297	bug - TrackMe REST API & libs - tracker simulation API response differs if breakby is an explicit none or not set	The API responds unexpectedly if the breakby is set to a "none" string depending on the component.
#1295	bug - duplicated rename in the macro trackme_idx leads to no results being returned (this macro is historical and not used any longer)	This macro has a duplicated rename statement and returns no results because of this. However, this is not used anymore in our codebase and will be considered for removal in a future release.
#1298	bug - TrackMe REST API - multiops endpoints for tenant creation should properly interpret the absence of hybrid_objects in the REST payload	If the payload does not include hybrid_objects, it should be set as an empty list to avoid the API from rejecting the call rather than not having any trackers to be created.

Table 141: Enhancements, changes and new features

Issue Number	Description	Details
#1294	feature - Virtual Tenants next generation user interface - Native React with Splunk UI	This release introduces the first version of the next generation user interface for Virtual Tenants, built in native React on top of Splunk UI.
#1299	feature - TrackMe REST API endpoints - Adds charts_resources object to the API endpoint load_component_data for usage in TrackMe next generation user interface	This feature adds the charts_resources object to the API response, which will be reused in TrackMe next generation user interface.
#1296	enhancement - TrackMe stateful and notable alert actions - Adding a safety check for the status of the maintenance mode to control at the Python level any decision according to the status of the maintenance mode	This enhancement adds an additional Python-level layer of safety which verifies the status of the maintenance mode and allows or prevents decisions according to it.
#1293	enhancement - React UI - standardize error handling across React UIs	Standardized error handling across React UIs: Maintenance Mode, Backup/Restore, License, and Maintenance KDB. Implemented a robust parseErrorResponse that prefers API-provided fields (error, message, response, payload, messages[0].text), falls back to raw text, then status text. Ensured the existing error modal displays these detailed messages consistently instead of generic HTTP phrases.
#1292	enhancement - Maintenance KDB user interface - improve responsive input modal	The input modal for adding records needs to be more responsive and properly adapt to any screen resolution.
#1291	change - Remove dependency from Splunk_SA_CIM application	TrackMe currently relies on the Splunk_SA_CIM for a few minor actions; this change allows removing any dependency regarding the SA CIM application for TrackMe.

11.1.57 Version 2.1.33 - build 1759998519 (09/10/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 88af89b916d798baf032f00f8574765fc671fe9cca57ed15ac594ff6b090a9bc

Table 142: Fixed Issues

Issue Number	Description	Details
#1284	bug - duplicated rename statement in macro trackme_metrics_idx causes no results	There is an invalid duplicated rename in the macro trackme_metrics_idx which causes it to no return any results.
#1282	bug - Fields Quality Monitoring (splk-fqm) - the concept of sample_hard_limit was not fully implemented in release 2.1.32	This safety was meant to be adding an hard limit in the creation of sampling based jobs for splk-fqm, but was not properly implemented at the phase of the actual job creation.
#1281	bug - Backup & Restore - prevent from stoping the restore operation if failing to re-create the Virtual Tenant account	There can be conditions where we fail to restore the Virtual Tenant account, and we should not stop the restore process if this happens. Secondly, we should sliently attempt to delete the Virtual Tenant account first, and have a safer definition of what keys are allowed in the Virtual Tenant account data.
#1280	bug - TrackMe Backup & Restore - Backup and restore operations now handle existing backup files correctly by using timestamped temporary directories, resolving “same file” errors that occurred when backup archives already existed on the target server.	Potential failure of restore requests due to working directory management.

Table 143: Enhancements, changes and new features

Issue Number	Description	Details
#1276	enhancement - License management user interface - migrating to native Splunk UI React	The license management interface is now migrating in native React UI.
#1278	feature - Rest Api Reference user interface - migrating this UI to a Splunk UI React Native user interface	The Rest Api Reference dashboard is migrated to a React native user interface built on top of Splunk UI.
#1283	feature - Maintenance Kdb UI transition to Splunk UI React native	The Maintenance Kdb user interface has been converted to native Splunk UI React.
#1287	feature - TrackMe Backup & Restore - Implement a function to allow deleting a specific target archive backup file	This adds a target delete capability to the delete_backup endpoint, to delete a specific archive file (locally or remotely). This updates the Backup & Restore management user interface to allow calling this action.
#1279	feature - Maintenance Mode user interface migration to Splunk UI native React	The maintenance mode user interface has been migrated to a React native UI on top of Splunk UI.

11.1.58 Version 2.1.32 - build 1759703239 (05/10/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: b970de58c208cf22512c41a5149723b5fde09c998ac390d91d0578117523c2de

Table 144: Fixed Issues

Issue Number	Description	Details
#1227	Feeds tracking - The adaptive delay backend does not preserve the value for future tolerance for the entities due to a bug in the underneath API endpoints logic	When the adaptive delay does request an update of the delay value, the underneath API endpoints for dsm and dhm do not preserve properly the existing value for the future tolerance due to a flaw in its logic and especially for this field.
#1236	Fields Quality Monitoring (splk-fqm) - searches underneath charts in fields overview screen should contain the context constraint to honor properly contexts with complex breakby	In relatively complex scenarios where a single tracker generates more than a single context, the searches underneath charts do not honor properly the context, which leads to incorrect results and longer run time searches.
#1237	Backup & Restore - The backup API export endpoint deletes the original backup archive after its export instead of preserving it on the search head file system	The export endpoint unexpectedly removes the original backup archive from the search head file-system. Once hit, the endpoint should provide the content of the archive without altering the original file.
#1246	TrackMe General Health Tracker - prevent licensing issue	Prevent unexpected unbound reference in case of license endpoint failure.
#1248	Potential failure of the custom command trackme-testremoteaccounts under some circumstances	Some issues were identified in the code of this custom command which can lead to its failure under some circumstances.
#1249	TrackMe Health Tracker - Prevent the task optimize_tenant_scheduled_reports from generating errors for replica tenants	This task improperly attempts to verify reports for replica tenants, leading to the generation of error messages.
#1250	Flex Objects (splk-flx) - duplicated log file handler trackme.flx.metrics leads a failure of the flx parse process when using metrics_list	A duplicated log file handler leads to the failure of the Flex parsing if using metrics_list
#1251	Disruption Queue - Prevents error messages in logs related to _key is missing from the disruption record due to a flaw in the assignment logic	In some conditions, TrackMe might unexpectedly modify the original disruption record removing the _key from the record while attempting its update leading to a non-ending loop of error messages regarding the missing _key in the record.
#1252	TrackMe log file handler - logic weakness can lead to a file handler unbound error	A weakness in the Python logic for TrackMe log file handlers can result in error messages and failures to generate the expected content.
#1253	TrackMe audit logger - prevent audit generation failure with bad character (49) in reply size	This error leads to potential error messages with symptoms of bad character (49) in reply size due to different coding issues and weaknesses.
#1264	TrackMe Backup & Restore - variable mismatch for restoring sample ratio for alerts	A mismatch in the variable calls would impact restoring sample_ratio on alerts.
#1265	TrackMe Backup & Restore - Fields Quality Monitoring has revealed some potential issues with non scheduled reports failing to be restored to a None value in the cron_schedule of the properties	We have identified potential failures in restoring Fields Quality related objects due to an unexpected None value in the savesearch properties, that the function underneath attempted to load as a cron expression.
#1266	Python logging - TrackMe loads various libs from the lib	This issue is limited to the logging performed directly in the functions stored in lib/*.py files and loaded from there. In

Table 145: Enhancements, Changes and New Features

Issue Number	Description	Details
#1227	Add a dedup capability in the show hybrid trackers REST API endpoints	This enhancement adds dedup capabilities to the show hybrid trackers endpoints, to handle any non expected duplication of the records in the KVstore collection.
#1228	Add an additional safety to prevent from generating any duplicated record in the hybrid KVstore collections	This adds an additional safety to prevent any duplicate generation.
#1230	Virtual Tenants - Check and fix tenant_id and component target API endpoint	This feature adds a new endpoint which can be used to verify and create any missing knowledge object of a given Virtual Tenant and a target component.
#1232	TrackMe REST API - Ensure each resource group for Virtual Tenants resources groups use their own handler class	Python code improvement to ensure each REST API endpoint resource group refers to its own super class.
#1233	Feeds tracking (splk-dsm/dhm) - Bulk Edit lagging policy improvement, separate each configuration item in its own bulk edit to avoid updating unwanted parameters for the lagging policy	In the bulk edit function, the current behavior updates all parameters at once, but this might not be desirable in most of the cases as this will update all policy parameters.
#1234	Bulk Edit - better alignment for the tags update action buttons	Tags buttons should be better aligned in the bulk edit screen.
#1235	Fields Quality (splk-fqm) - When using sampling mode, allows defining a custom value in addition with preset values from the sampling dropdown	In Fields Quality, the user can select the sampling mode with a preset value for the events ratio, this enhancement allows choosing a custom value for the ratio straight from the wizard. This issue addressed some minor issues regarding the call of the function in charge of refreshing the summary view of the creation of the tracker, which was not calling the right function for non CIM context tracker creation.
#1238	TrackMe Backup & Restore - Allows downloading a backup archive from the user interface via a simple download button	This new feature allows TrackMe administrators to easily download any available backup archive file using the backup & restore user interface, in a simple click!
#1239	TrackMe Backup & Restore - Automatically support Search Head Cluster context for the export archive API endpoint	This enhancement allows the API export endpoint to automatically attempt to retrieve a requested backup archive from the member actually owning the archive file, providing native support for Search Head Cluster (SHC) contexts.
#1240	TrackMe Backup & Restore - Allows importing (uploading) a TrackMe backup in the user interface	This feature allows importing a backup archive in a simple click!
#1242	TrackMe Backup & Restore - Always skip stateful charts collection from backup operations	TrackMe stateful charts KVstore collections should always be excluded from backup operations.
#1243	TrackMe packaging - prevents the generation of empty .conf files by the Splunk UCC	TrackMe relies on the Splunk UCC framework for the application package generation. However, Splunk UCC generates empty files for expected Splunk config files such as tags.conf, which is unnecessary and can be removed from the final package.
#1245	General Health Tracker - Adding a task to verify and purge KVstore records from the stateful charts collections after 48 hours (by default)	Chart records in the stateful charts record collection are not useful for a long period of time, and should be safely purged automatically by TrackMe.
#1254	TrackMe General Health Tracker - Adding the	This new task in the general health trackers identifies and fixes Virtual Tenants with missing component reports

11.1.59 Version 2.1.31 - build 1758668980 (23/09/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 2fe51a9696419e90d0e4c53c35d74cb1a044444c7663b9d1dcf5a4dd53714950

Table 146: Fixed Issues

Issue Number	Description	Details
#1217	Tenant Health Tracker wrongly re-enabled	Fixes a bug where the general Health Tracker re-enabled tenant-specific health trackers even when the corresponding Virtual Tenant was disabled.
#1219	Incorrect command mapping for FQM inspector	The *trackmesplkfqminactiveinspector* custom command was mistakenly targeting the Flex component instead of Fields Quality Monitoring (splk-fqm).
#1224	Health Tracker task failure on entities_auto_disablement	Improves error resilience of the *inspect_collection:entities_auto_disablement* task and prevents incorrect assignment of virtual accounts during its execution.

Table 147: Enhancements, Changes and New Features

Issue Number	Description	Details
#1218	Drilldown search support for Flex Objects	Introduces drilldown search capability for Flex trackers. Drilldowns can be configured per tracker and support both local and remote accounts. A new UI button opens the drilldown search in Splunk directly.
#1220	Remove unused unit test archive	Removes the deprecated *unit_tests_disabled* directory from release packaging.
#1221	Normalize Python dependency loading	Standardizes all Python backend modules to use the centralized *import_declare_test* loader for OS/Python compatibility. Fixes minor typos in library files.
#1222	Improve HEC error tracking use case in Flex	Enhances the default HEC tracking use case with a default drilldown search and corrects time alignment by using the original *_time*.
#1223	Default metric selection in Flex entity screen	Adds support for configuring a default metric to be preselected when opening the entity modal. Controlled via the *default_metric* field in tracker settings or UI.
#1225	Update OOTB Flex use cases with default metric	Updates built-in Flex use cases to utilize the default metric selection when relevant for improved user experience.

11.1.60 Version 2.1.30 - build 1758113820 (17/09/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.

- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 81a7446c680a27c7da21a889ec03bbdcc1cdc60126e09db7bcf95c71b32a0eb6

Table 148: Fixed Issues

Issue Number	Description	Details
#1216	Hybrid trackers UI shows duplicate entries	Fixes a regression introduced in *#1210*, where the Health Tracker task to patch missing hybrid tracker entries caused visual duplication in the Hybrid Trackers UI, although no actual duplication of reports occurred.

Table 149: Enhancements, Changes and New Features

Issue Number	Description	Details
#1214	Tenant-level override of CMDB lookup settings	Introduces the ability to override global CMDB lookup search definitions at the Virtual Tenant level, allowing component-level customization per tenant. If no override is set, the global default applies.
#1215	New SPL command *trackmeyamlpath* for YAML parsing	Adds a new custom streaming search command that enables parsing of YAML-formatted data directly in SPL pipelines. Useful for dynamic extraction of YAML fields from logs and structured events.

11.1.61 Version 2.1.29 - build 1757965912 (15/09/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 60ffd192607ac2c7fc650e5c051f758d65905c5408d26b7f072b059de3c6974c

Table 150: Fixed Issues

Issue Number	Description	Details
#1212	Flex: status description not overridden when dynamic thresholds alert	When dynamic thresholds are in alert, the *status_description* and *status_description_short* fields are now properly overridden to avoid conflicting messages from the search logic.

Table 151: Enhancements, Changes and New Features

Issue Number	Description	Details
#1207	UI layout optimization for high-resolution displays	Enhances layout scaling and chart height calculations for high-resolution screens, resolving visual compression issues where charts appeared too small relative to the screen space.
#1210	Health Tracker: hybrid tracker KVstore verification	Adds a new Health Tracker task to validate and patch missing hybrid tracker entries in the dedicated KVstore collection, ensuring consistency in tracker metadata.
#1211	Flex HEC error tracking use case enhancements	Updates the Flex HEC parser error tracking use case to use absolute time ranges and include explicit *0* values, improving clarity and data representation in charts.
#1213	Deprecation of *splk-cim* component	Removes the legacy *splk-cim* component from the Virtual Tenant creation UI as part of deprecation efforts.

11.1.62 Version 2.1.28 - build 1757413356 (09/09/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 444c16c01234b7787292c9a81f65907d655e2b999d22783384cf84a185db98ac

Table 152: Fixed Issues

Issue Number	Description	Details
#1194	Splunk Remote Deployments: bearer token validation step	Adds an extra step during token rotation to validate connectivity using the new bearer token before updating a remote account. Addresses rare edge cases where a token appears valid but fails in use.
#1199	Macro not applied in splk-fqm wrapper monitor reports	The macro *trackme_fqm_get_description_extended* was not used in wrapper monitor jobs or simulation searches. This fix updates the schema upgrade to ensure it's consistently applied.
#1200	Optimize tenant scheduled reports task failure	Addresses an edge case where missing required properties (e.g., *earliest*, *latest*, *cron*, *time_window*) prevent search enablement logic from executing properly in the Health Tracker.
#1202	KVstore insert error for splitline workload data	Fixes a rare condition in the Workload component where KVstore insertions for splitlines could fail unexpectedly.
#1201	UI glitch in Run Tracker filter for some components	Resolves visual misplacement issues with the tracker filter box introduced in earlier UI updates.
#1206	Flex thresholds using zero interpreted as boolean	Corrects threshold creation logic where *0* values could mistakenly be treated as boolean *false*, preventing proper threshold creation in use cases.

Table 153: Enhancements, Changes and New Features

Issue Number	Description	Details
#1195	Backup compression upgraded to Zstandard (zstd)	TrackMe backup archives now use *.zst* for faster compression/decompression at scale. Legacy *.tgz* format remains supported for backwards compatibility.
#119	Purge leftover uncompressed backup directories	Adds a pre-check during backup operations to clean up any previously uncompressed backup directories that may have been left behind.
#1197	Improved error handling in Backup & Restore	Backup operations now continue despite certain errors, ensuring partial data loss doesn't cause total backup failure.
#1198	SLA breach event frequency changed from 24h to 7d	The default value for *sla_breaches_events_frequency* has been increased from 24 hours to 7 days to reduce alert fatigue and align with typical SLA reporting cycles.
#1203	Workload versioning: configurable SHA256 parameter selection	Adds support for configuring which search parameters (*search*, *earliest*, *latest*, etc.) contribute to *version_id* hashing. Wildcards are supported. Changes are logged with *diff_<parameter>* tracking.
#1204	Workload versioning defaults improved	The default list of fields used for version hashing now includes *cron*, *enablement*, and *schedule_enablement* to capture more relevant changes.
#1205	New Flex use case: Detect HEC parser errors	Introduces a new Flex Object tracker designed to monitor for Splunk HEC parsing errors in indexed events, aiding troubleshooting and ingestion pipeline health.
#1208	Fields Quality Monitoring: default to head mode	Changes the default data collection mode to *head* rather than *sampling*, as it is simpler to manage in most environments despite the latter being more powerful.
#1209	Workload ML Outliers disabled by default	Machine Learning Outlier detection is now disabled by default during Virtual Tenant creation for Workload, as it provides limited value in this context.

11.1.63 Version 2.1.27 - build 1756708723 (01/09/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: c7718c937bdc4f270e6e482e5b705962bea30009256a6aede264792e942ca2dd

Table 154: Fixed Issues

Issue Number	Description	Details
#1186	Schema upgrade blocked by stale tracker references	If trackers were manually deleted outside of TrackMe, schema upgrades (e.g. from 2.1.11) may fail due to stale references in TrackMe's central collections. This update ensures these orphaned references are handled gracefully.
#1187	dispatch.sample_ratio not properly backed up	TrackMe's backup process did not include the *dispatch.sample_ratio* property, which is required to restore certain Fields Quality Monitoring reports using Splunk sampling. The property is now correctly backed up and restored.
#1192	Simulation view fails for non-CIM contexts	A regression affecting single-panel views in simulation mode for non-CIM contexts in Fields Quality Monitoring (splk-fqm) has been fixed. The issue was caused by incorrect default metadata definitions.

Table 155: Enhancements, Changes and New Features

Issue Number	Description	Details
#1188	Exclude stateful chart records from backup restore	Records from *stateful charts* collections are now excluded from restore operations to avoid unnecessary error messages. These are temporary records and do not require restoration.
#1189	Improved Virtual Tenant status handling in UI	Introduces a new *PENDING* status to better reflect the readiness of a tenant. Also improves the logic for clearing tenant statuses to avoid delays caused by automatic health tracker runs.
#1190	Filter input for tracker selection in Home UI	Adds a filter box to the tracker selection dropdown in the Run Tracker screen, improving usability especially during Fields Monitoring workflows.
#1191	Disable SmartStatus alert action by default	SmartStatus alert actions are now disabled by default for legacy alerting definitions, avoiding unintended alert behavior.
#1192	Support for phase 2-only Fields Quality collect jobs	Enables users to create collect jobs in Fields Quality Monitoring that skip phase 1 and only perform phase 2. This is useful when reusing previously collected quality events to produce alternate aggregations or views without redundant collection.

11.1.64 Version 2.1.26 - build 1756360749 (28/08/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0c5080e7cf19f45176cd916fe0f62601c75ee8c54b047f86bf34ece2d8edef7f

Table 156: Fixed Issues

Issue Number	Description	Details
#1175	Invalid search syntax in health tracker task *virtual_tenants:stateful_alerts_dup*	The scheduled task responsible for identifying and closing duplicate opened incidents contained invalid SPL syntax, preventing execution.
#1178	splk-fqm performance issues in distributed environments	Simulation and execution searches in Quality Monitoring (splk-fqm) were experiencing abnormally long runtimes in distributed Splunk environments. This was caused by improper streaming command behavior. The issue was resolved by explicitly forcing result sorting to ensure processing happens on the search head.
#1180	splk-fqm custom break-by fields not honored	Fields Quality Monitoring (splk-fqm) failed to respect user-defined custom break-by field configurations across simulations, global entity tracking, and internal logic. This update ensures consistent support throughout the system.
#1184	backup and restore sample ratior not restored	Fields Quality Monitoring (splk-fqm) introduced the usage of sample ratio, which is not currently supported by the restore functionality. This update ensures that the sample ratio is properly restored when restoring a backup.

Table 157: Enhancements, Changes and New Features

Issue Number	Description	Details
#1176	Safety check for *data_lag_alert_kpis.allow_adap* in decision logic	Introduces validation for this configuration value, preventing misbehavior due to invalid macro definitions.
#1177	Deduplication of stateful records at alert execution	Adds logic to automatically detect and close duplicate stateful alert records that may occur due to race conditions.
#1179	Null byte protection in Quality Monitoring	Implements a search-time filter to prevent Python SDK errors caused by null bytes in Splunk events.
#1181	Backoff retry for REST API config retrieval	Adds automatic retry logic for REST calls that fetch configuration data with privilege escalation, improving reliability under heavy load.
#1182	New REST API endpoint to update Virtual Tenant account parameters	Adds a granular administrative endpoint to update individual tenant configuration fields without requiring access to the full Configuration UI.
#1183	Improved schema upgrade backup handling	Enhances backup initiation logging and timeout handling during schema upgrades in large environments to prevent false error reporting and reduce retry attempts.

11.1.65 Version 2.1.25 - build 1756195621 (26/08/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: a526985fea753a44513214a98ae049115c65544fbd8497e7731a1f125b52f55c

Table 158: Fixed Issues

Issue Number	Description	Details
#1146	Non-ASCII characters in stateful alert email object names	When the monitored object contains non-ASCII characters (e.g. accented letters), TrackMe ensures safe storage in KVstore using a hexadecimal representation. However, this encoding was leaking into email notifications, showing unreadable object names. This fix now properly decodes the object name when generating email notifications so that the human-readable name is restored.

Table 159: Enhancements, Changes and New Features

Issue Number	Description	Details
#1169	Upgrade to latest Splunk UCC framework	This release updates TrackMe to leverage the latest stable version of the Splunk Universal Configuration Console (UCC) framework. Previous breaking changes in UCC had prevented immediate adoption. This upgrade restores compatibility and enables new UCC capabilities moving forward.
#1144	License expiration popup warning	TrackMe now proactively warns users when the installed license (Enterprise, Unlimited, or Trial Edition) is about to expire. If fewer than 15 days remain, a modal popup is automatically displayed to alert administrators.
#1170	Improve naming of merged index-level entities in feeds tracking (splk-dsm)	This change improves the naming convention for merged entities that operate at the index level in feeds tracking. By using the `:@all` suffix instead of `:all`, the merged entities are prioritized visually in grouped views. The change is backward compatible and does not alter previously discovered entities.
#1171	Prevent non-ASCII object duplication in search-time extractions	In Splunk, indexed fields containing non-ASCII characters can sometimes produce multiple values at search time (decoded and raw). This enhancement modifies the search-time field extraction logic to ensure only the decoded, user-friendly value is shown, improving consistency and usability.
#1172	Chart auto-type detection for count metrics in stateful alerts	For embedded charts in stateful alerting emails (splk-fqm, splk-flx, splk-wlk), if the metric name contains the word “count”, TrackMe will automatically switch the chart type to a bar chart for better visual representation of discrete quantities.

11.1.66 Version 2.1.24 - build 1755187604 (14/08/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0937d0f5e94f94d31515ffae5c635971534b2a4ea22e11ab74cca219af7ce9ef

Table 160: Fixed Issues

Issue Number	Description	Details
#1168	Handle reserved fields in FQM dictionaries	Fields Quality Monitoring (splk-fqm) previously did not correctly handle reserved fields such as *event_id*, *metadata*, and *summary* when they were included in a data dictionary. These fields are internally used by TrackMe and cannot be used as-is for auditing. This fix ensures any such reserved field is dynamically renamed with a *orig_* prefix (e.g. *orig_event_id*) during processing, preventing conflicts and unexpected behavior.

Table 161: Enhancements, Changes and New Features

Issue Number	Description	Details
#1166	Full support for non-CIM contexts in FQM	This feature completes the support for non-CIM contexts in Fields Quality Monitoring (splk-fqm). While the backend was already capable of handling non-CIM data, this enhancement adds full UI integration and logic in TrackMe to allow users to manage and monitor fields quality across any type of data, not just CIM-compliant events.

11.1.67 Version 2.1.23 - build 1754290192 (04/08/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: ee76ec6fadb1617bffa6c075a882ee7dca12c345b6ea5a58cd4f5c548290ba0a

Table 162: Fixed Issues

Issue Number	Description	Details
#1160	FQM global pie chart does not show failures	In the Fields Quality Monitoring (splk-fqm) global entity view, the right-hand pie chart fails to display failed results due to a typo in the field name used to calculate the chart.
#1165	FQM tenant status icon shows incorrect red cross	In the Virtual Tenant UI for splk-fqm, the mouseover tooltip incorrectly shows a red cross indicating failure for the overall tenant status, despite the actual status being healthy.

Table 163: Enhancements, Changes and New Features

Issue Number	Description	Details
#1158	Improved Test Regex filter in FQM dictionaries UI	Enhances the "Test Regex" button in the dictionaries management screen by dynamically pre-setting the source filter according to the dictionary's context, simplifying testing workflows.
#1159	Drilldown support for pie charts in FQM entity views	Adds support for interactive pie chart drilldowns in both global and per-field entity views in Fields Quality Monitoring (splk-fqm).
#1161	New shortcut to get non-matching events for FQM	Introduces a shortcut button to directly access sampled events that do not match the dictionary regex, streamlining troubleshooting of data quality issues.
#1162	Create empty dictionaries and UI improvements	Adds the ability to create a brand new empty dictionary from the UI, along with various enhancements to improve the usability of the dictionary management interface.
#1163	Bulk thresholds editing for FQM and Flex	Introduces a bulk edit feature for threshold management, applicable to both Fields Quality Monitoring (splk-fqm) and Flex Objects (splk-flx), to speed up large-scale configuration changes.
#1164	Enhanced status messages and per-category summary in FQM	Adds dynamic, detailed status messages that reflect the outcome of each field inspection. Also introduces a per-category summary in the field quality results with absolute and percentage metrics, providing clearer insight into compliance across all checks.

11.1.68 Version 2.1.22 - build 1753866030 (30/07/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 091bf77bfae127dcc2b926c29681a3e707d9e8a62e7ac9a4b013a500f6934ab1

Table 164: Major New Features

Issue Number	Description	Details
#1133	New Component: Fields Quality (splk-fqm)	Introduces a brand new component for CIM and non-CIM fields quality monitoring in Splunk and TrackMe. Based on the work described in the white paper (https://docs.trackme-solutions.com/latest/white_paper_fields_quality.html), it provides a scalable and mature workflow for continuous fields quality monitoring with a major focus on CIM contexts. Create jobs in a couple of clicks and get multi-dimensional entities instantly.

Table 165: Fixed Issues

Issue Number	Description	Details
#1149	Flex Object CMDB icon broken	The CMDB link icon stopped working for Flex Objects due to a regression introduced when UI-driven thresholds were added for Flex.
#1152	Stateful alerts email account listing fails in Splunk Web	The Splunk REST query used to populate the email account dropdown in Splunk Web (after alert creation) was invalid, causing the dropdown to fail.
#1153	SmartStatus compatibility issue with Splunk 9.1/9.2	Updated sparkline library caused SmartStatus to fail in Splunk 9.1/9.2 due to Python 3.7.x incompatibility.

Table 166: Enhancements, Changes and New Features

Issue Number	Description	Details
#1142	New streaming command: trackmeexpandtokens	Adds a streaming command designed to expand tokens in a streaming manner, primarily for stateful alerting active commands.
#1143	Stateful alerting improved token replacement	Enhances active commands token replacement logic by leveraging the newly introduced *trackmeexpandtokens* command.
#1148	REST API empty string fallback	Ensures TrackMe-level options render as empty strings if null to prevent issues with the latest Splunk UCC release.
#1150	Stateful alert creation wizard validation	Prevents users from creating stateful alerts unless mandatory inputs (e.g., email recipients) are provided, improving usability and reducing API errors.
#1154	Consistent import_declare_test usage	All TrackMe REST API endpoints now use *import_declare_test* for consistent Python library loading and improved control.
#1155	Flex Objects subgroup support	Adds support for defining subgroup values in Flex trackers, enabling multi-dimensional grouping in Tabulator views.
#1156	Stateful alerting latest priority retrieval	Enhances stateful alerting to always retrieve and consider the latest priority value from the tenant KVstore collection before upstream results.

11.1.69 Version 2.1.21 - build 1752477750 (14/07/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0d10f8440707aa1939d20cbc95995f58d753b5412c4a7de47854f3d38a454077

Table 167: Fixed Issues

Issue Number	Description	Details
#1127	Flex Object (splk-flx) mvfind matches substrings	Fixes a logic error in the mvfind function used in SPL for Flex Object CIM compliance. The previous implementation matched substrings, which caused inconsistent summary status values. Now matches only exact strings.
#1134	Flex Object simulation fails if object field is missing	Ensures the object field is present in Flex simulations. If the field is missing, the parser sets status to 1 instead of raising a Python exception.
#1140	Stateful alerting fails on Python 3.7 (Splunk 9.2.x)	Resolves Python compatibility issues in stateful alerting caused by OS-dependent libraries missing support for Python 3.7 in Splunk 9.2.x.

Table 168: Enhancements and New Features

Issue Number	Description	Details
#1125	New “pending” priority level	Adds a new “pending” priority level to support qualification workflows, enabling entities to be flagged for review before being included in alerting.
#1128	Alert macro usage update	Ensures alerts use the updated macro *trackme_apply_maintenance_mode_v2* for improved behavior.
#1129	Maintenance macro matching safety	Prevents incorrect tenant ID matches due to substring logic in macros, improving reliability of maintenance targeting.
#1130	Maintenance timezone safety	Improves handling of timezone conversion when the user’s profile does not explicitly define a timezone and operates outside UTC.
#1131	Schema upgrade robustness	Ensures upgrades can continue safely even if conflicting objects exist from prior rollbacks or restores.
#1132	Fields quality - Render regex per field	Improves backend and UI to track and display the regex expression applied per field in field quality analysis.
#1135	Flex Object thresholds - Compare against metrics	Allows threshold logic to reference another metric in the same record rather than being limited to static numbers.
#1136	Stateful alert - Custom chart time range	Adds support for customizing the time range used to generate embedded charts in stateful alert emails.
#1137	Stateful alert - Cooldown between alerts	Prevents creation or updates of incidents within 60 minutes of the last opened alert to avoid premature or duplicate detections.
#1138	General Health Tracker - Cleanup duplicates	Adds logic to automatically detect and remove duplicated open incidents from the stateful KVstore collection.
#1141	Fields Quality - Accept null or empty fields	Allows the field dictionary to accept empty or null field values and override regex failures using *accept_unknown* or *accept_empty*.

11.1.70 Version 2.1.20 - build 1751374209 (01/07/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: f054ff962fac174aba5ecd6efdaf080247f8cdc9225bd3b0a95582b0027146dd

Table 169: Fixed Issues

Issue Number	Description	Details
#1124	Elastic Source tracker disabled by mistake	A hotfix regression affected Elastic Sources where the health tracker optimization maintenance task incorrectly disabled the shared elastic source tracker even when entities still required management. This release restores correct tracker enablement logic.

11.1.71 Version 2.1.19 - build 1751273959 (30/06/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: fd492b3d6fd7a1271ac6069baf0c9f5106d6199fdb25ce4095b993ee23a01a1e

Table 170: Fixed Issues

Issue Number	Description	Details
#1093	Non existing custom command reference	A non-existing custom command (*trackmesendemail*) was referenced in *commands.conf*, potentially leading to Splunk warnings or appinspect failures. Removed for compliance and clarity.
#1098	Configuration screen rejects advanced group by	Regex validation in Virtual Tenant configuration prevented *eval* or non-CSV group by options, limiting Tabulator flexibility. Validation logic is now adapted.
#1099	Regex validation and default value issues	Fixes logic errors in regex validators and corrects invalid default values to ensure robust configuration parsing.
#1100	SmartStatus future tolerance extraction bug	Host and source extractions failed due to incorrect retrieval of future_tolerance values. Now resolved for reliable extractions in DSM/DHM.
#1101	UI freezes on large search source code	Show Trackers view loaded entire SPL source, which could freeze the browser with large searches. Optimized for performance stability.
#1104	Email formatting inconsistencies	Remaining formatting issues in some email clients, notably Outlook, resolved by simplifying HTML div structures.
#1105	Embedded chart compatibility in emails	Replaces SVG with PNG in alert emails to ensure full compatibility with all clients (Outlook, Gmail, etc.), along with supporting code enhancements.
#1111	Outliers chart UI glitch	Deleting and re-creating ML models broke the main outlier chart until full UI reload. Fixed to restore seamless chart rendering.
#1112	Flex status metric mismatch	Under some threshold breach conditions, the status metric did not reflect the decision maker logic. Now moved generation to *trackmedecisionmaker* for consistency.
#1113	Incorrect component registration in Disruption	Invalid component sent to registration function caused harmless log errors. Now corrected to send the proper component.
#1114	Ack tracker anomaly reason comparison bug	Under specific conditions, Ack could release prematurely due to incorrect comparison logic. Normalization now ensures accuracy.
#1116	Role concurrency exception handling in SDK search	*run_splunk_search* did not handle role concurrency limits gracefully. Now improved to detect and retry accordingly.
#1123	Event field wrongly indexed as metadata	*trackme_events_ingest_evals* and *trackme_audit_events_ingest_evals* transforms caused *event* fields to be indexed as metadata, creating warnings due to length. This is now prevented.

Table 171: Enhancements and New Features

Issue Number	Description	Details
#1102	Incremental schema upgrades	The schema upgrade process now updates the version incrementally after each function completes, improving resilience against upgrade interruptions.
#1103	Prevent concurrent backups post-upgrade	Health trackers now detect in-progress backups from logs rather than KVstore, preventing simultaneous backups during upgrades.
#1106	More embedded charts in alerting	Adds incident, flipping, and state charts as embedded visuals in StateFul alerting emails, enhancing diagnostic clarity.
#1107	Persistent charts across alert phases	Ensures ML Outliers and related charts are included consistently in opened, updated, and closure phases for continuity.
#1109	Outliers exclusion periods apply to all models	Period exclusions can now be applied to all ML models for an entity simultaneously via UI bulk actions.
#1110	Background ML bulk actions	*mlmonitor* and *mltrain* bulk operations now execute asynchronously (fire-and-forget), avoiding UI timeout issues.
#1116	Status metrics for inactive Flex entities	Flex objects now generate status metrics via inactive trackers with associated handler events for accurate lifecycle tracking.
#1117	Automated utility scheduling	TrackMe utilities (e.g., mltrain/mlmonitor) schedules are automatically enabled/disabled based on operational need, reducing workload.
#1118	Health tracker self-monitoring	Adds maintenance task to verify Virtual Tenant Health Tracker is active, ensuring core functionality is never unintentionally disabled.
#1119	Hide cron schedule in Alerts UI	The cron schedule field is now hidden from alert creation UI to prevent misconfiguration by untrained users.
#1120	Simplified StateFul alert creation	Provides granular priority-based actions for notifications and commands, streamlining the StateFul alert setup workflow.
#1121	Improved search backoff strategy	*run_splunk_search* now uses progressive backoff for retries, improving reliability on systems with concurrency limits.
#1122	Fields Quality major enhancements	Introduces three new SPL utilities (*trackmefieldsqualityextract*, *trackmefieldsqualitygensummary*, *trackmefieldsqualitygendict*) and a full rewrite of the Flex Object OOTB use case for CIM compliance and continuous monitoring.

11.1.72 Version 2.1.18 - build 1748377090 (27/05/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 6aab45bc762ea66ca4f06f961efdc2244a52c31c511dff5b70fbba7090602c05

Table 172: Fixed Issues

Issue Number	Description	Details
#1034	Tabulator group state not preserved	Resolves improper group state reset on manual or auto refresh in TrackMe UI Tabulator.
#1041	DHM metrics incorrect for multi-sourcetypes	Fixes macro logic to account for all sourcetypes on volume calculations in *splk-dhm*.
#1043	*email_send_update_if_ack_acti* not respected	Ensures updates are emailed if entity is acknowledged and flag is enabled.
#1045	Unbalanced quotes in *splk-mhm*	Fixes tracker regression from 2.1.16 affecting hybrid creation.
#1049	Email chart uses *min* instead of *max*	Status now reflects highest state instead of lowest.
#1050	Outlook formatting issues in emails	Enhances HTML for Outlook client compatibility.
#1052	Virtual Tenant creation fails with limited permissions	Applies proper privilege elevation in REST headers.
#1055	StateFul alerts not restored	Includes StateFul alerts in backup and restore processes.
#1056	Scoped restore affects unrelated KV collections	Filters collections by tenant during scoped restore.
#1057	Restore fails on Splunk email actions	Excludes native Splunk email settings to prevent restore failures.
#1059	Drilldown uses API v1 instead of v2	Corrects dashboard links to target REST API v2.
#1064	Outliers default latest time not applied	Updates *trackme-settings.conf* to use *-1d* as default.
#1065	ML charts fail with past *latest_time*	Fixes chart simulation to support historic latest values.
#1066	No UI refresh after Flex logical group edit	Automatically refreshes status views post group changes.
#1069	Missing *event_id* for inactive objects	Ensures summary events include *event_id* for red-state entities.
#1070	Full-down service shows orange	Adjusts status logic to show red (status=2) if all members are down.
#1074	State event error in *splk-cim*	Resolves outlier-based Python error in CIM component.
#1076	Ack expires on empty anomaly list	Prevents misinterpretation of empty lists as state changes.
#1079	Tracker fails if *tags_manual* is native list	Adds type safety to ensure *tags_manual* is always a string.
#1086	Blue state considered down	Treats *object_state=blue* as up for Flex convergence logic.
#1089	Missing *mvexpand* for email accounts	Ensures all delivery options are listed in UI popup.
#1090	Tag audit records include full entity	Fixes audit trail to log tag list only, not entire object.

Table 173: Enhancements and New Features

Issue Number	Description	Details
#1032	Load tenants if one is corrupted	Backend resilience added to avoid full failure on partial tenant corruption.
#1033	Skip disabled entities in convergence	Default behavior now excludes *monitored_state="disabled"* from convergence.
#1035	Right-click popup improvements	Enhances readability of contextual entity view.
#1036	Add *converging_status* to extras	Adds *up/down* visibility in extra_attributes.
#1037	Auto-Ack config in wizard	Enables toggling Auto-Ack at creation time for StateFul alerts.
#1038	Prevent new alert if Ack is active	Updates only if Ack exists, avoids duplicate threads.
#1039	Render gaps in alert charts	Gaps now rendered instead of misleading zeroes.
#1040	Allow closure within 5 mins	Prevents suppression of incident closure events.
#1042	Accept threshold in seconds or units	Enhances threshold entry with unit suffix support (h/d/w).
#1044	Fetch tracker definition live	UI now pulls real-time definition to avoid outdated data confusion.
#1047	New Studio dashboard template	Adds OOTB Dashboard Studio template for service health.
#1048	Support JSON dict parsing	Allows *trackmefieldsquality* to extract structured fields.
#1051	Minimum percentage up for green	Convergence logic now accepts minimum healthy percentage threshold.
#1053	Alert status in email header	Visually tags alerts as critical/informational with color code.
#1054	Retry on restore failure	Adds fallback retry if dependency isn't restored yet.
#1058	Filter API by keys	Accepts comma-separated object keys for component data.
#1060	UI/URL filter by keyid	Adds keyid param support for safer drilldown with special chars.
#1061	Use keyid in alert drilldown links	Avoids URL errors for non-ASCII object names.
#1063	Configurable thresholds per entity	Flex tracker and UI support for per-entity metric thresholds.
#1067	Pre-checks before Ack call	Ensures entity is alerting before REST-based Ack.
#1068	Larger chart height in Flex UI	Defaults to 450px for improved visual clarity.
#1072	List down entities first	Prioritizes down entities in status descriptions.
#1073	Prism.js for JSON formatting	Syntax highlights JSON in entity popups.
#1075	Skip disabled entity in alerting	Ignores processing for disabled entities in StateFul alerting.
#1077	Priority selector in flip UI	Adds dropdown in status flipper for setting entity priority.
#1078	Disruption Queue feature	Introduces time-based disruption tracking before turning entities red.
#1080	Improve audit search consistency	Enhances uniformity of audit tab searches across components.
#1081	Update OOTB use cases	Leverages new thresholds and disruption logic in bundled Flex trackers.
#1082	Push expected data from CMDB	Adds *trackmepushdatasource* to create entities from referentials.
#1083	Improve tenant wizard preview	Enhances UX with embedded table preview in creation flow.
#1084	Disable SLA events if frequency=0	Skips SLA breaches generation for disabled or configured tenants.
#1085	Suppress entity list in all-green	Avoids listing all entities if they're all up.
#1088	Improve corrupted tenant error UI	Adds friendly remediation steps for tenant load failures.

continues on next page

Table 173 – continued from previous page

Issue Number	Description	Details
#1091	Check for orphan/outdated incidents	New health tracker task verifies alert integrity.
#1092	Remove deprecated health task	*synchronize_trackers_attr* task removed as obsolete.

11.1.73 Version 2.1.17 - build 1746041932 (30/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: bbc6cb524f5e4d93d70f5c9f003408f8a8b6d9a0231919ba3e966abafc1ab5a3

Table 174: Fixed Issues

Issue Number	Issue Description	Issue Details
#1031	Appinspect - prohibited characters in sourcetypes	A newly introduced Appinspect check `check_props_conf_has_no_prohibited_characters_in_sourc` leads to Appinspect failure. This update resolves the issue.

Table 175: Enhancements and New Features

Issue Number	Enhancement Description	Enhancement Details
#1027	New streaming command `trackmefieldsquality` for fields quality auditing	Adds a new TrackMe custom command designed to support fields quality auditing in Splunk environments.
#1028	SmartStatus - improve search for data in the future	Extends the future/past time range in the UC logic to capture events where timestamps are more than 4 hours ahead, preventing search failures.
#1029	Upgrade Tabulator JS library	Upgrades the embedded Tabulator JavaScript library to version 6.3.1.
#1030	Health Tracker - new task `check_tenants_indexes_setti`	Introduces a new health task to automatically verify and fix missing index declarations in tenant configurations, which could otherwise lead to errors during event generation.

11.1.74 Version 2.1.16 - build 1745791632 (27/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 9f31ae61532da1b343e6ff1dcc5734c33e3dce61972809c5ce2dd4bd54715f9c

Table 176: Fixed Issues

Issue Number	Issue Description	Issue Details
#1024	TrackMe sub-logging regression	TrackMe 2.1.15 caused regressions in logging subsystems, resulting in missed ingestion of audit events, handler events, and others. This release fixes all associated logging issues.
#1025	TrackMe Home UI - SLA theme color issue	In the SLA tab, if the SLA is breached, the message should display in red theme. This was not happening properly in some use cases and is now corrected.

Table 177: Enhancements and New Features

Issue Number	Enhancement Description	Enhancement Details
#1018	StateFul Alerting - Execute commands based on incident state	Introduces active commands triggered when incidents are opened, updated, or closed, enabling state-aware integration with third-party systems (e.g., ServiceNow). See documentation.
#1020	SOAR Monitoring - Page size control and recent window logic	Enhances the `soar_playbook_status` action to allow page size control and reduce the concurrent playbooks search window from 1 hour to 5 minutes for better scalability.
#1021	Audit & Troubleshoot - Splunk Remote Accounts dashboard	Provides a new dashboard to review the status of Splunk Remote Deployment accounts and monitor token rotation status via the new `trackmetestremoteaccounts` custom command.
#1022	Flex Object Library - Search Head health check UC update	Updates the Search Head health check use case to immediately trigger red status if the response is anything other than success.
#1023	StateFul Alerting - Include trackme:state events for early trigger	Incorporates `trackme:state` events into alerting logic to allow faster incident triggering without waiting for new event movements. Schema upgrade will update existing alerts automatically.
#1026	StateFul Alerts - Control email updates when Acknowledged	Adds a control option at the alert level to determine whether updated email notifications should be sent when the entity has an active Acknowledgement, improving alerting behavior.

11.1.75 Version 2.1.15 - build 1745332184 (22/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 6a3df5cfd71bfdd1f841c558f5db19c486c7a627716c8993cfb8dda1420b60bf

Table 178: Fixed Issues

Issue Number	Issue Description	Issue Details
#1019	Trackers and degraded mode - remote accounts regression	A regression caused remote account-based trackers to remain in degraded mode even after the issue was resolved, preventing proper recovery after outages.

Table 179: Enhancements and New Features

Issue Number	Enhancement Description	Enhancement Details
#1016	TrackMe REST API logging - prevent external collision	Additional improvements were made to TrackMe's REST API logging to ensure no logging collision occurs with other external applications in shared environments.
#1017	StateFul Alerting - Alert wizard UI update	Enhances the alert creation wizard to dynamically display or hide "Emails only" options based on the selected alert mode, improving clarity and user experience.

11.1.76 Version 2.1.14 - build 1744885691 (17/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0087a889a3c25510a9a6a71c86f5b0c7f696d064ac3c17a24ca9ff6f25efeb89

Table 180: Fixed Issues

Issue Number	Issue Description	Issue Details
#1013	TrackMe logging - REST API potential logging conflicts	In certain environments, TrackMe's REST API logging lacked explicit handler definitions. This could result in logging conflicts where TrackMe logs contain unrelated external logs, or TrackMe logs are leaked into external logs.
#1015	Hosts Tracking (splk-dhm) - regression in host-level latency/delay	A regression introduced in version 2.1.11 prevented proper persistence and handling of host-level latency and delay thresholds in *splk-dhm* monitoring logic.

11.1.77 Version 2.1.13 - build 1744663474 (14/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 60a9959a9779c3f62420eec025368a156a46b78692209b57957714d99193da53

Table 181: Fixed Issues

Issue Number	Issue Description	Issue Details
#1009	StateFul Alerting - ingest only does not generate stateful events	Due to a bug in the alert action, ingest-only mode does not function as intended and fails to generate stateful events.
#1010	StateFul Alerting - Alias field is missing	The *alias* field is not included in the stateful events, which may impact downstream logic or integration relying on this metadata.
#1011	StateFul Alerting - local MTA triggers configuration exception	Systems using a local MTA (mail transfer agent) for email delivery encountered an exception due to missing *trackme_emails* configuration. This update ensures the fallback to localhost config is handled properly.
#1012	StateFul Alerting - configurable options for local MTA	Added new system-level options in the General tab to support sender address and other configurations when using the localhost MTA for email delivery.

11.1.78 Version 2.1.12 - build 1744327446 (11/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 16eb3f5bc532b09860006c3faf7741939a8ae1d3680095e749e4d3d15368b269

Table 182: Fixed Issues

Issue Number	Issue Description	Issue Details
#1008	StateFul Alerting - Splunk 9.1/9.2 Python 3.7 compatibility	StateFul Alerting introduced in 2.1.11 fails in Splunk 9.1.x and 9.2.x due to import errors caused by Python 3.7. This fix ensures the StateFul feature no longer crashes and gracefully disables embedded charts if Pygal cannot be loaded. Note: Emails charts are not available in Splunk 9.1/9.2 environments.

11.1.79 Version 2.1.11 - build 1744200878 (09/04/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.

- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 0122ea5cd0ed73a17562f24a12cb3c8808a75501d7021d70b07c37bf8970494c

Table 183: Fixed Issues

Issue Number	Issue Description	Issue Details
#969	Minor typos in config/help descriptions	Fix various typos or grammatical issues in TrackMe.
#974	Audit Data Sampling dashboard token bug	Hard coded value in a main search for the tenant_id token replacement.
#978	Schema upgrade fails if alerts were manually deleted	Customers experienced issues upgrading to 2.1.10 if alerts were manually deleted.
#979	Restore alert actions not re-enabled	Issue in re-enabling alert actions due to improper "actions" formatting.
#981	SLA regex input issue	UCC config doesn't allow editing SLA classes due to regex validation bug.
#982	UI filters in Virtual Tenant view	KO filtering breaks navigation when no results exist.
#986	SVC metrics now in _cmc_summary	Splunk Cloud index change; update to TrackMe logic accordingly.
#988	Ack expiry bug for future_over_tolerance	Incorrectly treated as anomaly reason change, prematurely expiring Acks.
#991	0d auto-disablement disables all entities	Logic flaw disables all entities when 0d is defined.
#993	Transforms backup fails if admin renamed	Hardcoded admin username in REST call macro.
#990	Flex group rename escaping issue	Problem handling double quotes in remote contexts.
#998	Incorrect export for timeline viz	Should be "none" instead of "app" in default.meta.
#1001	Disabled ranges ignored in delayed entity inspector	Setting range to 0 doesn't disable it properly in backend.
#1002	Default entity priority dropdown ignored	Affects Flex/Workload/CIM Virtual Tenant creation.
#1003	UC large lookup file inconsistent timeout	max_sec_inactive is 1h while schedule is every 12h.
#1004	ML Outliers only renders one model	Bug prevents rendering of multiple models per entity.
#1005	ML tab missing in splk-cim	Outliers tab missing in UI even if ML is enabled.
#1007	Adaptive delay causes config loss	Auto-updates override non-delay settings like max latency.

Table 184: Enhancements and New Features

Issue Number	Feature Description	Feature Details
#963	Health Tracker alert consistency check	New task to check for manually removed alerts and purge stale entries.
#964	Schema Upgrade resilience	Prevents failure if alerts were removed outside of TrackMe.
#962	Handlers event notifications system	Track handler actions and health status in a new tab per entity.
#961	Exclude KVstores from backups	Backup exclusion list for KVstore collections.
#965	Exclude KVstore/KO from restores	Restore blocklist for collections and knowledge objects.
#966	Logging deprecation fix	Address deprecated log.setLevel usage.
#967	Virtual Tenant creation UI refresh	CSS and label improvements.
#968	Modal UI enhancements	Ensure all modals have a header close button.
#970	SOAR failure use case optimizations	Performance improvements for high-scale environments.
#971	Prevent browser grammar on Flex inputs	Disable grammar checking on code inputs.
#972	SOAR Broker status enhancements	Status red triggered if not active.
#975	Tracker health consistency tasks	Checks existence of trackers and KO references.
#976	Constraint inputs UI improvement	Use textarea for better experience.
#983	Decision Maker message clarity	Better status messages to reduce confusion.
#985	Alert modal UI refresh	Simplified and improved per-alert modal design.
#987	Host thresholds precedence	Host-level thresholds now override sourcetypes.
#980	Stateful alerting and email threading	Full-featured incident lifecycle with HTML email and embedded charts.
#989	Sticky Acks default	Sticky Ack now default based on user preference.
#992	SOAR concurrent playbooks tracking	REST API and Flex tracker to monitor playbooks live.
#994	Info/SPL/success UI consistency	Prevent UI messages from rendering off-screen.
#995	SLA breach event throttling	Configurable generation frequency for SLA breach events.
#996	Virtual Tenant auto-repair	Automatically detects and fixes upgrade failures.
#997	Update Splunk Remote Account URLs	New admin API endpoint to update *splunk_url*.
#999	Improved rendering for search/inspector menus	Better UI handling of embedded Splunk actions.
#1000	Remote Search performance metrics	Query each SH member, return response time and job count.
#1006	Reduce frequency of delayed entity inspector	Less frequent checks to reduce resource costs.

11.1.80 Version 2.1.10 - build 1741820646 (12/03/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 1503eff8cd5b8864a30ea9bc04c6518c2ac96d14508f192256cfc3be76fb0930

Table 185: Fixed Issues

Issue Number	Issue Description	Issue Details
#909	UI Virtual Tenant - Remove Deprecated Color Option	The *color_theme_red_default* option, replaced by *color_theme_alert_default* in 2.1.9, was still available and has now been removed.
#910	TrackMe Outliers Detection - ML Model Creation Failure	Fixed an issue where new ML models could not be created if no previous models existed for an entity.
#911	TrackMe Outliers Detection - *auto_correct* Not Defined	Resolved an issue where *auto_correct* was not being set when adding a new model for Flex Object trackers via the UI.
#915	TrackMe Health Tracker - Missing Tenant Account Handling	Fixed an issue where the health tracker would not recreate missing Virtual Tenant accounts due to a missing *force_create_missing* parameter.
#916	TrackMe REST API - Incorrect *tenant_default_priority* Check	Fixed an issue where *tenant_default_priority* was incorrectly validated as *default_priority*, causing API failures.
#918	Hybrid Trackers - Inconsistent *time_window* Scheduling	Addressed inconsistencies in how TrackMe schedules *time_window* for hybrid trackers, ensuring uniform behavior.
#919	Virtual Tenant UI - Drag-and-Drop Issue	Fixed a UI issue where dragging Virtual Tenant boxes did not work correctly when multiple rows were present.
#920	CIM Tracking - Exception Handling for ML Detection	Fixed a Python assignment error that could occur when an ML anomaly was detected in CIM tracking.
#922	Data Sampling UI - *relative_time_window_seconds* Not Honored	Fixed an issue where user-defined *relative_time_window_seconds* values were not being correctly applied in the UI.
#923	Data Sampling Backend - *relative_time_window_seconds* Not Applied	Fixed an issue where the backend (*trackmesamplingexecutor* command) ignored user-defined *relative_time_window_seconds*.
#924	Remote Accounts - Token Rotation Failure	Fixed a token rotation failure due to a hardcoded service account name in the REST API.
#933	CIM Compliance - Incorrect Outlier Field Count	Fixed an issue where green and red field counts were not displayed correctly in the UI when ML Outliers were detected.
#934	ML Outliers - Training Failure When Deleting Old Models	Resolved a failure in ML Outliers training where old models were not being properly deleted before retraining.
#935	ML Outliers - UI Chart Rendering Issue	Fixed an issue where ML Outliers charts would not load unless the page was manually refreshed.
#940	Batch Priority Updates - Missing Action Field	Ensured that *generic_batch_update* always includes an *action* field in responses for improved debugging.
#941	CIM Compliance - Duplicate JSON Download Requests	Fixed an issue where downloading a CIM JSON configuration triggered duplicate requests.
#946	Virtual Tenants UI - Status Preview for Flex Objects	Improved Flex Object preview to display *status_description_short* and allow right-click actions in the UI.
#949	Acknowledgment Tracker - Expiration Failure for Removed Components	Fixed an issue where expired acknowledgments could not be processed for components that had been removed.
#951	Virtual Tenant Component Deletion - Associated Alerts Not Removed	Ensured that when deleting a component, any associated alerts are also removed automatically.
#956	Python Backend - Improper Logging Restoration	Fixed an issue where TrackMe's Python metric generation libraries failed to restore proper logging in some conditions.
#957	jQuery Scan Failure in Splunk URA App	Addressed a jQuery scan failure in Splunk URA caused by the deprecated Splunk-built timeline app.

Table 186: Enhancements, New Features, and Changes

Issue Number	Issue Description	Issue Details
#912	Virtual Tenants and Home UI - Enhanced Look and Feel	Improved TrackMe UI with modernized visuals and better handling of modal screens, especially for low-resolution contexts.
#913	Flex Objects - Group Renaming Capabilities	Added ability to rename Flex Object groups, automatically updating associated trackers, migrating entities, and preserving historical metrics.
#917	REST API - Remove Unnecessary Logging for Global Index Settings	Prevented unnecessary error messages when a Virtual Tenant uses "global" index settings, reducing redundant log entries.
#921	Data Sampling Dashboard Enhancements	Improved out-of-the-box Data Sampling review dashboard for better usability and insights.
#925	Outliers Detection - Default *kpi_span* Control	Introduced a system-wide option to define the default *kpi_span* value for newly created ML models.
#926	*trackmegenjsonmetrics* Command - Enhancements	Improved command to support automatic recognition of numerical values and prefix-based modifications for better metric handling.
#927	Flex Objects UI - Pre-Filtering Entities in Metric Views	Added a pre-filter text input to help users refine selections when adding entities to metric charts.
#928	Flex Object - Improved Splunk Cluster Global Status Use Case	Enhanced the cluster monitoring use case to consider additional health factors such as replication, site factors, and active bucket fixing.
#929	Flex Object - Filter Valid OOTB Use Cases	Ensured only valid out-of-the-box use cases are displayed, filtering out outdated or removed configurations.
#930	SOAR with Flex Objects - Refactored Playbook and Adhoc Error Detection	Redesigned detection for SOAR playbook and adhoc failures, adding REST lookups to extract associated metadata.
#931	TrackMe Theming - Match Skipping/Degraded Colors	Adjusted color scheme to ensure consistency with TrackMe's global theming.
#932	ML Outliers Detection - Bulk Rule Updates	Introduced a REST API endpoint for bulk updates to ML outlier detection rules, integrated into the UI.
#936	Flex Objects - Split License Usage Volume by Deployment Type	Separated license usage tracking for Splunk Cloud and Splunk Enterprise due to differences in pool concepts.
#937	Priority Management - Identify Policy/External Control	Added *priority_reason* field to clarify if an entity's priority is controlled by policies or external sources.
#938	SOAR Integration - Prevent ES8 Conflict	Renamed *soar.py* to avoid conflicts with embedded libraries in Splunk Enterprise Security 8.
#939	Enhanced Priority Management - Policy and External Overrides	Improved UI messaging and per-entity override capabilities when policies or external systems manage priorities.
#942	Virtual Tenant - Control Machine Learning Activation	Introduced *mloutliers_allowlist* parameter to selectively enable ML outlier detection per component.
#943	Virtual Tenants - Improved Creation UI	Enhanced tenant creation UI with clearer labels and real-time feedback.
#945	Flex Objects - Converging Multi-Dimensional KPI	Introduced a KPI *pct_availability* to aggregate multiple Flex entities into a single service availability metric.
#947	License Verification - Prevent Incorrect Actions on KVstore Failure	Ensured TrackMe does not take license-related actions if a KVstore exception prevents verification.
#950	TrackMe Acknowledgements - Preselect *ack_type*	Defaulted *ack_type* dropdown to match the existing acknowledgment record for an entity.
#952	UI Enhancements - Improved Legend Styling	Adjusted legend design for better readability.
#954	Flex Objects - Deduplication for Metric Ingestion	Added deduplication capabilities to prevent duplicate metrics by tracking the last seen epoch in KVstore.
#955	Flex Objects - Refactored Splunk Cloud SVC Consumption Use Cases	Improved Splunk Cloud SVC tracking use cases, adding a dedicated use case for service consumers.
#958	Feeds Tracking - Adaptive Delay Enhancement	Incorporated SLA percentage into Adaptive Delay logic to avoid threshold adjustments during outages.
#959	Flex Objects Library - Universal Forwarders Version Compliance	Added a new use case to track the percentage of compliance for Splunk Universal Forwarder versions, lever-

11.1.81 Version 2.1.9 - build 1738934449 (07/02/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: 85f83d14bb98010e9fe1f2f56989b4daf085a118367f587eae9522bf7bd03ac8

Table 187: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#895	Flex Object - Ensure *status_description_short* Always Has Value	Fixed an issue where *status_description_short* could be empty under rare conditions, particularly when *max_sec_inactive* is 0 and the entity has not been handled by the inactive entities tracker.
issue#897	CIM Compliance Tracking - Flipping Event Generation Errors	Fixed a Python error causing failures in generating flipping events under specific conditions.
issue#905	SOAR Tracking - Automation Brokers HA Issues with Large Asset Volumes	Resolved REST API pagination issues that impacted SOAR Automation Brokers tracking and high availability features when a large number of assets were configured.

Table 188: Enhancements, New Features, and Changes

Issue Number	Issue Description	Issue Details
issue#894	Flex Object/Workload - Allow Disabling Inactive Entity Purging	Updated *trackmesplkflxinactiveinspector* and *trackmesplkwlkinactiveinspector* commands to allow *0* as a valid argument, disabling entity purging.
issue#896	Trackers - Inherit Earliest/Latest and Control Alert Behavior	Removed the explicit need for *earliest* and *latest* arguments, allowing them to inherit from metadata variables unless specified. Introduced an argument to control whether an empty result set should impact Virtual Tenant operational status.
issue#898	Data Source Monitoring - Real-time Entity Status Refresh	Enabled real-time updates to entity status for Data Sampling (*splk-dsm*).
issue#899	Health Tracker - Improved Logging and Performance	Enhanced logging for *trackme:health* events, including ACL details and additional metadata. Improved performance using *requests.session* to reduce REST overhead.
issue#900	Hybrid Tracker Wizard - Clarify Optional Burn Benchmark Tasks	Explicitly marked burn benchmark tasks as optional in the Hybrid Tracker creation wizard to reduce user confusion.
issue#901	Flex Objects - Multi-Entity Selection for Chart Generation	Added support for selecting multiple entities within the same Virtual Tenant when generating charts for specific metrics.
issue#902	Flex Object Trackers - UI Enhancements with Single Stats Visualization	Added single stat visualizations (Perc95, Max, Avg) to the Flex entity modal views for improved usability.
issue#904	Cribl Monitoring - Enhanced Status Descriptions	Added *status_description_short* field to all Cribl monitoring use cases for improved user experience.
issue#906	Virtual Tenant UI - Improved Alert Visualization	Updated the Virtual Tenant UI with an enhanced color scheme for better visualization of high and critical priority entities in alert states.
issue#907	Home UI - Modernized Alert Priorities Display	Improved the TrackMe Home UI with a modernized look and feel for managing single views associated with alert priorities and alert counts.

11.1.82 Version 2.1.8 - build 1738021014 (27/01/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.0.x and earlier.
- The last release compatible with Splunk 9.0.x is TrackMe 2.0.99

- SHA256: fb323b1bc0c529ffcb1342dbd6c0146d8f5b20e8fcbbbf7c06719801ef759f46

Table 189: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#870	Notable Events - Drilldown Link Ignores Custom Web Root Endpoint	Fixed an issue where drilldown links in notable events did not preserve additional URL path parts when using a custom Splunk Web root endpoint.
issue#872	Home UI - Refresh Button Failure with Custom Splunk Web Root	Fixed an issue where the refresh button failed due to an invalid REST endpoint definition when using a custom Splunk Web root endpoint.
issue#873	Audit System - Missing Tenant ID in Audit Events	Resolved missing *tenant_id* values in some audit events, ensuring they are properly displayed in the entity audit tab.
issue#874	AppInspect Failure - Inline Comment Parsing Issue	Addressed an issue where the AppInspect check *check_collections_conf_for_specified_name_field_type* incorrectly failed due to inline comments in configurations.
issue#875	Notable Alerts - Missing Tenant ID Filter	Fixed an issue where notable alerts did not include *tenant_id* as a search filter, preventing cross-notable events in multi-tenant environments.
issue#876	Prevent splunkd Startup Error from Timeline Viz Integration	Addressed splunkd startup error messages caused by missing README spec instructions in the timeline visualization integration.
issue#881	REST API - Missing Double Quotes in SPL Import/Export Examples	Fixed missing double quotes in resource examples for SPL import/export.
issue#882	Tags Policies - Updating Policy ID Causes 404 Error	Resolved an issue where updating the policy ID from the UI led to a 404 error due to incorrect KVstore record retrieval logic.
issue#884	SLA & Priority Policies - Policy ID Update Causes 404	Fixed a similar issue affecting SLA and Priority policy updates in the UI.
issue#886	Adaptive Delay Tracker - Default Search Time Issue	Fixed an issue where the recent activity search in the Adaptive Delay Tracker unexpectedly ran over all time due to a missing earliest time argument.
issue#889	Workload (splk-wlk) - SmartStatus searches for errors tracking	The generated search for errors (function smartstatus_investigations_uc_wlk_execution_errors) should not use indexed earliest and latest.
issue#890	TrackMe Dashboards - Errors loading TrackMe logo	Fixed an issue where TrackMe dashboards failed to load the TrackMe logo due to incorrect path resolution.
issue#892	Week days monitoring - (all components except splk-dsm) - Week days monitoring by the day selection issues	Different UI and API issues have been identified for all components, except splk-dsm.

Table 190: Enhancements, Changes, and New Features

Issue Number	Issue Description	Issue Details
issue#871	Splunk SOAR - High Availability & Monitoring Enhancements	Introduced a pool definition capability for automation brokers, improving flexibility over active/passive configurations. Decommissioned the monitor use case in favor of a unified Flex Object use case.
issue#854	Drilldown Links - Preset Time Range Support	Added support for influencing preset time ranges in entity views via drilldown links. Allows relative time formats such as *-24h* or *24h*.
issue#877	Notable Events - Per Alert Default Earliest Time	Introduced a *drilldown_earliest* option for defining a per-alert default earliest time in notable event drilldowns.
issue#878	Timechart Span Definition Enhancements	Introduced a new macro for automated span value definition in timechart calls, improving reliability over UI-based span determination.
issue#880	Outliers Detection - Default Latest Time Adjustment	Changed the default *latest* value for ML training to *-1d* instead of *now* to prevent abnormal periods from affecting training.
issue#879	Outliers Detection - Handling Feed Interruptions	Updated outliers detection for *splk-dsm/dhm* to generate *0* value metrics, ensuring feed interruptions are properly accounted for in detection processes.
issue#883	Flex Objects - Inactive State Management Improvements	Improved inactive state detection, real-time status updates, and enhanced anomaly reasoning in *splk-flx*.
issue#885	Splunk Remote Account Management - Bearer Token Auto-Rotation	Introduced automatic bearer token rotation for Splunk remote accounts. Configurable per account with a default rotation interval of 7 days.
issue#887	Flex Object use cases library - Splunk introspection CPU & Memory Monitoring	Enhancement to these use cases with the definition of the status short description field.
issue#888	Flex Object use cases library - DataModel Acceleration Monitoring	Enhancement to the use case the definition of the status short description field.
issue#891	TrackMe dashboards - Refresh for Dashboard Studio based dashboards	Syntax and source code refresh for out of date dashboard studio based dashboards.

11.1.83 Version 2.1.7 - build 1735907247 (03/01/2025)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: 3fc504463e4270fd8142ad0bff0c32c46bf33fb329fa13b9d9a6e04f35cefa90

Table 191: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#826	Dependency App Issues - Timeline App EOL by Splunk	The timeline visualization app has reached EOL as of December 24, 2024, and is no longer available on Splunk Base. TrackMe now incorporates these visualization components natively, removing the external dependency.

Table 192: Enhancements, Changes, and New Features

Issue Number	Issue Description	Issue Details
issue#865	Flex Object Library - Update for Cluster Management Use Cases	Updated Flex Object library use cases to replace legacy references to the Splunk master convention with modern terminology for cluster management.
issue#861	Bulk Edit Tags	Introduced a bulk edit UI capability for managing manual tags efficiently.
issue#866	Flex Object Library - Deployment Server Clients Tracking Refactor	Refactored the Splunk deployment server clients tracking use case to utilize the *_ds** indexes introduced in Splunk 9.3.x for better tracking capabilities.
issue#868	Flex Object Trackers Library Enhancements	Enhanced the Flex Object trackers library with additional attributes and improved status descriptions, including updates to: - *splk_splunk_enterprise_cluster_status* - *splk_splunk_enterprise_cluster_peers_status*
issue#867	Flex Object Trackers - Short Status Description Field	Added a *status_description_short* field for displaying concise status descriptions in the main Tabulator table while maintaining a detailed status description for additional insights. The logic now automatically handles this new field if not present in existing configurations.

11.1.84 Version 2.1.6 - build 1734273789 (15/12/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: b9236c33c34fcdfb948629de3007ce874380e20f9f63029a658fcffba7930715

Table 193: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#858	Knowledge Objects and Shortcuts Issues with Custom Splunk Web Root Endpoint	Resolved issues accessing knowledge objects and built-in shortcuts when using a custom Splunk Web root endpoint. Fixed drilldown link generation for TrackMe technical alerts and addressed JS errors in license registration and maintenance knowledge database notifications.
issue#858	Maintenance Mode Enablement Fails via UI	Fixed a regression in TrackMe 2.1.5 causing a 404 error when enabling maintenance mode through the UI.
issue#857	Backup & Restore Issues with Disabled Virtual Tenants	Resolved exceptions during backup operations when disabled Virtual Tenants exist. The restore process now forces purging of disabled tenants to prevent loading issues.
issue#858	Remote Search - Default HTTPS Port Handling	Improved handling of default HTTPS/443 port in remote Splunk REST API requests to avoid failures when the port is not explicitly provided in the URL.

11.1.85 Version 2.1.5 - build 1734004063 (12/12/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: 2d3ccbe77a6929fc982f645378521fa56fff97a0d865b11642918f7c8c252eb4

Table 194: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#834	System Level Preferences Misconfiguration	Home UI system level preferences were not properly applied, with Virtual Tenant UI preferences taking precedence. Fixed to honor distinct system level preferences for Home and Virtual Tenant UIs.
issue#837	Virtual Tenants - Scheduled Search Count Issue	Show Knowledge Objects screen incorrectly displayed a count of 0 for scheduled searches due to misparsed *is_scheduled* field.
issue#838	Get Knowledge Objects - JSON Parsing Error	The macro *get_tenants_reports* did not escape double quotes in macro definitions, leading to invalid JSON structures.
issue#839	Data Sampling Migration Issue	Schema migration from TrackMe versions < 2.0.36 directly to 2.1.x failed for the Data Sampling Tracker. A workaround migration path is recommended until resolved in version 2.1.5.
issue#840	Schema Upgrade Errors	Various errors during schema upgrade tasks (2084, 2064, 2099) resolved for migrations from TrackMe 2.0.x to 2.1.x.
issue#845	Alerts Creation - JSON Key Duplication	Resolved an issue where duplicate keys were generated in JSON payload during alert creation.
issue#843	Data Sampling - Typo in Remediation Messages	Fixed typos in Data Sampling remediation messages.
issue#849	FIPS Transition Issues	Minor issues with SHA256 transition from MD5 in version_id metrics ingestion and flipping events identification for splk-wlk and splk-cim resolved.
issue#848	Flex Objects - Typo in Lookup Files Monitoring Use Case	Corrected typo in JSON file name and use case name for large lookup files monitoring.
issue#852	Modular Alerts Logs - Timezone Issue	Modular alerts logs now properly handle non-UTC timezone settings.
issue#853	Drilldown Error in Multi-Component Tenants	Resolved drilldown activation error for components not first in the tab list.
issue#855	Splunk Web Custom Root Endpoint Issue	Addressed compatibility issues with custom root endpoint configurations in *web.conf*.
issue#856	Workload Metadata Tracker - Minimal Privileges Issue	Improved handling of minimal privilege accounts in metadata tracker operations for splk-wlk.

Table 195: Enhancements, Changes, and New Features

Issue Number	Issue Description	Issue Details
issue#832	Splunk Python SDK Upgrade	Upgraded to the latest Splunk Python SDK version 2.1.0.
issue#833	Splunk UCC Validator Warning Messages	Ensured all UCC configurations include input validators to address warning messages introduced in UCC 5.52.0.
issue#835	Virtual Tenants Wizard - Custom Fields Enhancement	Added missing hover action for custom field configurations in the Virtual Tenants wizard.
issue#841	Splunk UCC Upgrade to 5.53.0	Upgraded Splunk UCC to version 5.53.0.
issue#842	Configuration Management Enhancement	Improved readability of TrackMe configuration screens by grouping configuration items using UCC 5.53.0 features.
issue#836	Backup & Restore Feature	Introduced capabilities to backup and restore knowledge objects and KVstore content for Virtual Tenants and components.
issue#847	Virtual Tenant API Enhancements	Improved input verification for tenant index settings to prevent misconfigurations.
issue#850	Flex Objects Tracker Name Simplification	Removed `my_` prefix from Flex Object tracker names.
issue#851	Flex Objects Tracker Name Sanitization	Enhanced sanitization for requested tracker names during simulation and creation.

11.1.86 Version 2.1.4 - build 1731085887 (08/10/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: 46b4db465c4dafc67fdaffc534d629ad894a6ac47a67210d2eda7d508a427e9f

Table 196: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#822	Inline Bulk Edit - Adaptive Delay Persistence Issue	The Adaptive delay setting, modified via inline bulk edit in the Tabulator, does not persist. This occurs due to the UI's restricted persistent fields, missing Adaptive delay, which should rely on TrackMe's Python library for centralized handling.
issue#823	Virtual Tenants - System Level Fallback Misconfiguration	The fallback configuration for *splk_feeds_auto_disablement_period* fails if Virtual Tenant lacks the expected definition. Issue impacts auto disablement period settings for inactive entities.
issue#825	Flex Objects - Error in Inactive Entities Tracker	Flex inactive entity tracking may fail if *max_sec_inactive* is undefined, caused by a Python code error when handling this missing value.
issue#830	Remote Search - Timeout Config Error	Remote search failures occur due to timeout values received as strings instead of integers. This fix ensures timeout values are processed correctly as integers.
issue#831	Data Sampling - Remote Entities issues	Data sampling issues with remote entities. This fix addresses various issues with Data Sampling v2 when entities are remote entities.

Table 197: Enhancements, Changes, and New Features

Issue Number	Issue Description	Issue Details
issue#818	Tabulator - JS Upgrade to v6.3.0	Upgraded the Tabulator JS component to version 6.3.0 for improved functionality and performance.
issue#819	Splunk UCC Upgrade to 5.52.0	Updated Splunk UCC from version 5.48.2 to 5.52.0.
issue#820	Virtual Tenants UI - Usability Improvements	Added quick access buttons for Scheduler and Ops Status, improved modal screens, and a more consistent layout with closing icons where needed.
issue#821	Flex/Workload Blocklist Extension	Blocklist features extended to splk-flx/wlk components, adding flexibility to block specific patterns with a schema upgrade to initialize allowlist collections.
issue#824	Virtual Tenant Account Management Enhancement	Implements a more secure approach for verifying and updating Virtual Tenant accounts, with auto-check and repair features through the Health Tracker.
issue#827	Flex Objects - Enhanced Cribl Logstream CPU Usage Detection	Improved Cribl Logstream CPU consumption use case for fewer false positives and clearer alerts.
issue#828	Flex Objects - New Use Case for Dynamic Sourcetypes	Introduces a new use case to detect and track dynamic sourcetypes in Splunk, optimizing log rotation handling.
issue#829	Data Sources Tracking - Hybrid Tracker Enhancements	Allows inclusion/exclusion of sourcetypes during hybrid tracker creation for splk-dsm, adding control over custom break-by fields.

11.1.87 Version 2.1.3 - build 1728629753 (11/10/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: f33353510b450588df38976b465d87bf95f7614d9223c4a3348b08d48b95af1b

Table 198: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#817	TrackMe Home UI - Regression with TrackMe 2.1.2	High priority regression due to a missed token validation for all components except splk-dsm.

11.1.88 Version 2.1.2 - build 1728540776 (10/10/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

- SHA256: 361ea0ee5bd07584f96ebd8960af8f1ff6ac82d5f2b68b08ea45cae6f880e848

Table 199: Fixed Issues

Issue Number	Issue Description	Issue Details
issue#780	TrackMe Home UI - Workload / Flex / Metric hosts (splk-wlk/flx/mhm)	The entity screen incorrectly shows "host state" rather than "entity state" next to the state icon.
issue#781	Acknowledgement - Auto expiration of Acknowledgements based on condition changes	The Ack that auto raised can be expired by the Ack auto-management. If the anomaly reason is none, it will incorrectly expire the Ack.
issue#777	TrackMe Hybrid trackers - Missing Hybrid tracker from central collection	A new task ensures that the hybrid tracker KV collection remains consistent and fixes records if needed.
issue#782	Blocklists features - Unexpected additional dot in regex-based blocklists	Adding a regex blocklist with a wildcard results in an extra dot due to incorrect handling of non-regex blocklists.
issue#786	Data Source monitoring - Data sampling engine issues with future data	Earliest time can be after the latest, causing sampling searches to fail due to future data indexing.
issue#788	Home UI - Donut chart doesn't show green state entities	A bug prevents green state entities from being represented in the top-right donut chart.
issue#789	Elastic Sources - Background task for entity count refresh not called	The Shared Elastic tracker does not call the method to refresh entity count, leading to incorrect data.
issue#790	Elastic Sources - mstats-based searches don't generate expected dcount host metrics	An SPL field name prevents expected host distinct count generation.
issue#786	TrackMe Health Tracker - Incorrect warning for tags tracker	Health Tracker generates incorrect warnings due to hard-coded DSM component definition in tags tracker.
issue#802	API Documentation & Reference - Incorrect API reference examples for tags policies management	Examples in the documentation are missing the component argument.
issue#804	Persistence issue for entities with backslashes	Backslashes in entity names cause issues with persistence of settings like priority and lagging thresholds.
issue#808	Data Sampling & Events format recognition - Confusing regex simulation message	When regex does not match any events, the result message is confusing and should clearly indicate 0% match condition.

Table 200: Enhancements, Changes, and New Features

Issue Number	Issue Description	Issue Details
issue#783	Blocklist management for splk-feeds - Various improvements	Enhanced management screen, added comment storage, and background entity count updates for blocklists.
issue#784	TrackMe Virtual Tenants UI licence information & Schema Upgrade	Added clickable TrackMe version display, schema version info, and shortcut to manage licence and upgrades.
issue#785	Virtual Tenants UI - Notify messages for quick action buttons	Ensured all buttons in the Virtual Tenants screen show notify messages when hovered.
issue#787	Virtual Tenants UI - Embedded Entities Overview Tabulator	Introduced a single-pane Entities Overview in Virtual Tenants for easier navigation.
issue#791	TrackMe Home UI - Quick access to reports from Elastic Dedicated screen	Added quick Splunk Web access for reports from Elastic Sources management UI.
issue#793	Data Sources monitoring - Tenant level control of Data Sampling	Added an option to enable/disable Data Sampling for Virtual Tenants, hiding UI functions accordingly.
issue#792	TrackMe Home UI - Hiding adaptive_delay feature when disabled	Automatically hides adaptive_delay-related UI elements when the feature is disabled.
issue#795	Elastic Sources - Support for mpreview-based searches	Added mpreview-based searches as a replacement for mtstats, providing true metrics count reporting.
issue#796	Elastic Sources wizard - Presets for earliest/latest based on search type	Automatically presets recommended earliest/latest times based on the type of search.
issue#797	TrackMe Health Tracker - Auto fix duplicated entities	Automatically detects and fixes duplicated entities in all components.
issue#798	Cribl Logstream monitoring - CPU usage metrics	Added CPU usage metrics, including time spent in green/red states, to TrackMe metrics indexes.
issue#799	Virtual Tenants UI - UX enhancement for Tenants Ops view	Improved user experience with status selectors, quick access to reports, and logs in the Tenants Ops view.
issue#800	Virtual Tenants & Home UI - Tabulator sort header improvement	Removed the sort header for fields where sorting is not meaningful in Tabulator.
issue#801	Virtual Tenants UI - Scheduler overview enhancement	Replaced Splunk table with a tabulator view for the scheduler overview with quick actions.
issue#803	Splunk Remote Search - Configurable timeouts for remote accounts	Added per-account configurable timeouts for connection and search in Splunk Remote Search.
issue#805	ML Outliers - Minor log improvements	Enhanced the quality of logs generated by ML outlier detection.
issue#807	Hybrid Trackers - Cron schedule validation	Added cron schedule validity checks using croniter library for all scheduled logic.
issue#809	Data Sampling & Events format recognition - Python code improvements	Improved Python code quality and safer behavior for the Data Sampling engine.
issue#810	Data Sampling & Events format recognition - Entity settings overview	Added a dynamic entity settings overview in JSON format within the Data Sampling UI screen.
issue#811	Bulk edits & Audit logging - New audit format for bulk edits	Refactored bulk edit function to track changes per field, improving audit logging.
issue#812	TrackMe Audit subsystem - Mass audit REST call improvements	Switched to mass audit REST calls for better performance and flexibility in the Audit subsystem.
issue#813	TrackMe events - Consistent event_id convention	Standardized event_id across all TrackMe-generated events using sha256 hash.
issue#814	TrackMe Home UI - Allows selecting visible tabs and their order at the level of the Virtual Tenant account	This feature adds a new parameter in the Virtual Tenant account, to control the order and visibility of the main tabs in the Home UI.
issue#815	TrackMe Home UI - A new component replaces the usage of the input list for the top tab links in Home, for more flexibility and control	This adds more flexibility and control, and notably ensures Virtual Tenant level parameters to be initialized before calling components loading.
issue#816	Virtual Tenant - Allows overriding the system general auto disablement settings for splk-feeds at the level of the Virtual Tenant account	This features adds a Virtual Tenant level option to override the system general setting defining the behaviour for disabling inactive entities from a certain amount of days.

11.1.89 Version 2.1.1 - build 1726614488 (18/09/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

Introducing TrackMe's data sampling events format recognition v2!

- TrackMe 2.1.0 welcomes the introduction of a brand new version of the events format and recognition for data sampling. (Data quality inspection, PII tracking, and more)
- This is major change and improvements in the way TrackMe handles data sampling events, and will allow for slightly more flexibility and control over the data sampling process.
- For more information about the engine v2 and its capabilities, see the admin guide: *Data sampling*

- SHA256: eb750290ecf39e926fe7e6528de8fbfdac8f078f9a6922cca7b0167a69cc4f0d

Fixed issues:

- **trackme-limited/trackme-report-issues#769 - bug - TrackMe Data Sampling UI - links to pre-built KPI metrics search generate an incorrect earliest time**
  - In the data sampling, several quick access buttons allow generating automated KPIs metric searches in a new blank tab.
  - However, due to a Javascript bug, an incorrect pattern is unexpectedly added to the earliest time.
- **trackme-limited/trackme-report-issues#770 - bug - Virtual Tenant UI & REST API - Splunk Knowledge Objects explorer and associated endpoint generated an invalid JSON, preventing the UI formatting to work as expected**
  - The Virtual Tenant UI screen for knowledge object access should generate a JSON pretty printed of the properties field.
  - However, the macro called by the endpoint generates an invalid JSON, and the REST API endpoint should better handle the parsing too.
- **trackme-limited/trackme-report-issues#774 - bug - Flex Objects & Workload - trackmesplkflxinactiveinspector/trackmesplkwlkinactiveinspector custom commands are designed to handle inactive entities purge, however the process is not currently working as intended**
  - These commands are designed notably to purge entities which have been inactive for too long, after the configured period in days passed as an argument to the commands.
  - However, this particular action does not work as intended currently, and purge of entities is not currently effective.
  - This fix addresses these issues and also slightly improves the code quality and logging of these commands.
- **trackme-limited/trackme-report-issues#779 - bug - SLA tracking - When entities status change, the SLA status will temporary be inconsistent and will show an incorrect time until trackers have reflected the real time change in the KVstore**
  - When a given entity status changes from one to another, the SLA status is temporarily inconsistent due to the discrepancy between the real time status provided by the Decision Maker and the fact that the KVstore object_state is yet to be updated.
  - With this fix, the SLA status takes into account both values, and will show a specific SLA refresh pending message, and the SLA status and timer will wait for the KVstore to be updated accordingly to avoid any inconsistency.

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#771 enhancement - Virtual Tenant UI - Tenants Splunk Knowledge Objects explorer screen - Add quick actions button for the Tabulator**
  - This enhancement adds various quick action buttons to filter out the Tabulator content against saved-searches only, macros only, and so forth
- **trackme-limited/trackme-report-issues#772 - enhancement - Notify bar lookup & feel - Responsive design and modern look & feel for the notification bar in TrackMe**
  - This enhancement is a major refresh of the responsive notification bar plugin in TrackMe.
  - The provides a responsive look & feel notification bar to TrackMe and a much improved and more modern appearance.
- **trackme-limited/trackme-report-issues#773 - enhancement - TrackMe Home UI - Add quick Splunk Web access to reports & macros from the Hybrid Tracker**
  - This enhancement adds quick access in Splunk Web for reports & macros through the Hybrid trackers management UI for all components
- **trackme-limited/trackme-report-issues#775 - change - Python - Addressing the deprecation of calling log.setLevel with logging.getLevelName when defining the current logging level in the various Python backends**
  - This change addresses a Python deprecation of calling the method logging.getLevelName within log.setLevel
- **trackme-limited/trackme-report-issues#778 - enhancement - In Virtual Tenants & Home UI, show the number of enabled entities rather then total number of entities**
  - This enhancement updates the number of entities primarily shown in the Virtual Tenants UI, as well as the Home UI and the left single view, so that we show the total number of enabled entities, instead of the total number entities active + inactive.
  - This provides more valuable information as well as better clarity in TrackMe.

11.1.90 Version 2.1.0 - build 1726088942 (11/09/2024)

Hint

For Splunk 9.1.1 and later

- From TrackMe 2.1.x, there is no more compatibility with Splunk 9.1.0 and earlier.
- The last release compatible with Splunk 9.0.x up to 9.1.0 is TrackMe 2.0.99

Introducing TrackMe's data sampling events format recognition v2!

- TrackMe 2.1.0 welcomes the introduction of a brand new version of the events format and recognition for data sampling. (Data quality inspection, PII tracking, and more)
- This is major change and improvements in the way TrackMe handles data sampling events, and will allow for slightly more flexibility and control over the data sampling process.
- For more information about the engine v2 and its capabilities, see the admin guide: *Data sampling*

- SHA256: 2a79af2a363bf1d10beb2f51f8c3f1334298d046015d4a23d1197c14ae5572c9

Fixed issues:

- **trackme-limited/trackme-report-issues#757 - bug - TrackMe schema migration from 2.0.97 and prior to 2.0.98 and latest should address the tags extension to splk-mhm**

  - In TrackMe 2.0.98, the tags features were normalised and extended to all components.
  - However, splk-mhm was not included in the list of eligible components leading to various issues in this components.
  - This change addresses the issue automatically, if the splk-mhm component was enabled, the tags extension will be managed during the schema upgrade.

- **trackme-limited/trackme-report-issues#761 - bug - TrackMe Health Tracker - subcontext="entities_auto_disablement" is attempting to perform a REST call per entity instead of a mass disablement operation, leading the tracker to eventually take an abnormal amount of time to be executed while failing to disable entities, and possibly generate skipping searches**
  - The Tenant health tracker runs a subcontext task called entities_auto_disablement, which is designed to automatically disable the monitoring state of entities that have not generated any data according to the system wide setting splk_general_feeds_auto_disablement_period.
  - However, a bug affecting this task incorrectly attempts to run a REST call per entity, instead of a mass REST call.
  - In some circumstances, this leads to an abnormal amount of run time for tracker and can cause skipping searches for the tracker
- **trackme-limited/trackme-report-issues#763 - bug - Typo in UI - Create Hybrid trackers**
  - This fixes a typo in the Home UI and the Virtual Tenant UI, and for the Hybrid tracker creation wizards
- **trackme-limited/trackme-report-issues#765 - bug - trackmesplkgetflipping can still be affected by a non expected missing object_category value in the KVstore record**
  - In some conditions, exceptions can still be encountered during the call of the streaming custom command trackmesplkgetflipping, leading to failures in properly handling the search logic. (especially in splk-dhm)
  - This update adds the object_category as an argument to the streaming custom command instead of getting this value from the records, the schema upgrade will process the update of all hybrid trackers wrapper search automatically so that the argument is called.
- **trackme-limited/trackme-report-issues#766 - bug - Transition to SHA256 based logic for FIPS compatibility mode in TrackMe 2.0.99 needs to be retro-applied on any tracker created by TrackMe**
  - Since TrackMe 2.0.99, we use sha256 instead of md5 to calculate the expected hash for TrackMe entities objects.
  - In some specific use cases, such as the Flex object for host tracking or some contexts in Workload, this change needs to be reflected on the search logic itself.
  - The schema upgrade will verify all trackers and automatically update any tracker if needed.
- **trackme-limited/trackme-report-issues#767 - bug - Virtual Tenants UI - When creating hybrid trackers during the Virtual Tenant wizard, tracker names should be a random combination rather than only the account name**
  - During the Virtual tenant creation, Hybrid trackers can be created per tenant/component, the hybrid tracker names should be a combination of "tracker-<random ID>" instead of just the account name.
- **trackme-limited/trackme-report-issues#768 - bug - TrackMe Backup REST API endpoints - avoids raising a file does not exist exception in some rase cases**
  - In some specific circumstances, the POST backup API endpoint can raise an exception if the expected file does not exist, this update simply avoids this condition.

Enhancement, changes and new features:

- **trackme-limited/trackme-report-issues#756 - feature - Data Source tracking - Introducing the Data Sampling events and format recognition engine v2**
  - TrackMe 2.1.0 welcomes the introduction of the engine v2 for the data sampling and events format recognition.

  - This is a completely rewritten engine in full Python, providing flexible and powerful capabilities for events quality inspection in TrackMe.
  - The new engine provides flexible options at the system wide level, which can be customised on a per entity basis, such as controlling the min time between sampling iterations, the number of events sampled per iteration (which is now 10K), the truncation of events when storing samples for investigation purposes, initial thresholds for min match inclusive percentage, and more!)
  - With the the engine v2, parsing of recognitions models is made against the whole event and is no longer limited due to truncation, events are no longer stored in the KVstore then processed, but processed then a sample of sampled events is stored in the KVstore per model matched and for review purposes.
  - The new engine introduces a brand new concept of major / minor models matching, allowing to tackle minor quality issues without generating non meaningful alerts, TrackMe admin can control the minimal threshold of acceptable percentage of events matching the main model.
  - Tracking Personally Identifiable Information (PII) can handle as many models as required (exclusive match)
  - The interfaces were rewritten so the data sampling feature can be controlled and reviewed more efficiently and with more capabilities.
  - KPIs generation from Data Sampling: The engine now generates KPIs in TrackMe's metrics models, so you can review over time the events matching percentage per model, the amount of events parsed and matched, as well as other KPIs such as the run time of the sampling operation per entity.
  - Many additional improvements were made in the data sampling engine v2!

- **trackme-limited/trackme-report-issues#758 - enhancement - TrackMe Schema upgrade - normalise the schema version to always use a 4 digits based logic, handling the patch version number**
  - This update ensures that TrackMe uses a consistent 4 digits based logic for the schema_version number, and handles notably the question of a new minor release and its associated patch number (ex: 2.1.0 versus 2.0.99)
- **trackme-limited/trackme-report-issues#759 - feature - TrackMe Notable events - Add a unique identifier in each TrackMe notable event**
  - Some customers may make use of a unique identifier in TrackMe notable events, especially to ensure notables have been consumed accordingly.
- **trackme-limited/trackme-report-issues#760 - feature - TrackMe Home UI & Tabulator - Add a Download button which allows downloading visible and filtered entities as a CSV file**
  - This new feature adds a Download button above the Tabulator table which allows quickly exporting visible and filtered entities as a CSV file for quick review out or data manipulation out of TrackMe
- **trackme-limited/trackme-report-issues#762 - enhancement - TrackMe Health Tracker - Logging and code improvements to allow easily monitoring the run_time taken by each task processed by the tracker**
  - This enhancement slightly improves the logging of the run_time taken by each task executed by the Health Tracker using a concept of task_instance_id associated with a task_name
  - A sample SPL:

```
` index=_internal sourcetype=trackme:custom_commands:trackmetrackerhealth instance_id=* task_instance_id=* task=* run_time=* tenant_id=* | table _time tenant_id instance_id task task_instance_id run_time _raw | sort 0 - _time `
```

- **trackme-limited/trackme-report-issues#764 - enhancement - TrackMe Schema Upgrade - before starting upgrade procedures, execute TrackMe's builtin backup**
  - With this enhancement, when TrackMe detects that migration procedures must be initiated, it will first query a TrackMe backup to be executed.

11.1.91 Version 2.0.99 - build 1723722498 (15/08/2024)

- SHA256: 6d7174da69a584a5dfdef160f2cb07410630db5b5bcf397aeb1956f83b037cd2

Additional notes about this release

- To address FIPS compatibility requirements, we have migrated from md5 to sha256 various TrackMe internal search logics.
- There are near no impacts to existing installations, however for customers using TrackMe Workload, you will notice that all monitored objects (Scheduled discovered) will generate a Metadata event, which normally happens only when a change in the search is detected.
- This behaviour is due to the change from md5 sum calculations to sha256 calculations for FIPS compatibility purposes, and can be safety ignored and acknowledged as part of the upgrade to TrackMe 2.0.99

Fixed issues:

- **trackme-limited/trackme-report-issues#738 - bug - TrackMe pagination - When using pagination mode = local, the pagination size is not submitted by the Tabulator and should default to size = 0 since the pagination is performed by the Tabulator rather than the server, which leads to missing records in high scale collections**
  - The default pagination mode is local rather than remote, however when using pagination = local, the default size should be 0 as the Tabulator will not submit this as an argument to the REST call to TrackMe.
  - This leads currently to missing records in the UI for high scale collections.
- **trackme-limited/trackme-report-issues#740 - bug - TrackMe Home UI - When opening an entity and if the tenant_id field of the Kvstore has an empty content unexpectedly, the UI fails to load the entity overview modal screen**
  - If in the tenant_id/component KVstore collection, the tenant_id field does not have a content, the UI fails to open the entity overview
- **trackme-limited/trackme-report-issues#741 - bug - trackmehealthtracker - logging reports the details of untracked entities for splk-dhm rather than the number of them**
  - for splk-dhm, the trackmehealthtracker should not report the detailed content of untracked entities, but how many of them were found instead.
- **trackme-limited/trackme-report-issues#743 - bug - Data Sampling for splk-dsm - Managing the Data sampling feature (enable/disable/run/reset) would fail for an entity which has not been processed at least once by the data sampling engine**
  - In the current release, managing the status of the data sampling feature can only happen if the data sampling has processed the entity at least once, and would fail otherwise.
  - This fix ensures that we properly manage the feature depending on the user request, no matter if the entity was processed already or not.
- **trackme-limited/trackme-report-issues#746 - bug - CIM compliance tracking (splk-cim) - When creating a new entity by cloning and if the CIM constraint contains one or more double quotes, the creation fails**
  - Creating a new entity by cloning for splk-cim fails if the CIM constraint contains double quotes
- **trackme-limited/trackme-report-issues#753 - bug - Outliers Anomaly detection - Workload (splk-wlk) - TrackMe should not attempt to train models for entities parts of applications that have been disabled in the Workload component**
  - In TrackMe's Workload component, entities can be enabled/disabled at the app level.
  - When an application is disabled, TrackMe should not attempt to consider training ML models.

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#736 - feature - FIPS compatibility for TrackMe**
  - Splunk 9.3.x introduced several fixes which made Splunk really FIPS compatible, which disabled some crypto algo such as md5 which is used by TrackMe.
  - These developments allow TrackMe to be fully FIPS compatible, and FIPS validated from TrackMe Limited.
- **trackme-limited/trackme-report-issues#742 - enhancement - Data Sampling for splk-dsm - Improve the message and status returned when data sampling is disabled for a given entity to avoid any confusion**
  - When data sampling is disabled, the message shown in the UI, and returned underneath by the API endpoint, should be clearer to avoid any confusion.
- **trackme-limited/trackme-report-issues#737 - bug - Data Source tracking (splk-dsm) - Grouping issues with reduced pagination and entities for the same indexes that are split over multiple pages**
  - If using a reduced pagination size, entities relying on the same indexes can be split over multiple pages.
  - This fix may not entirely prevent this as this depends on the pagination size, but the addition of the index in the initial sort should limit the risk of this happening.
- **trackme-limited/trackme-report-issues#744 - enhancement - Health Tracker - Addition of tenant_id value verification in the record inspection steps**
  - This added step verifies that records of the main KVstore collection have a valid value for the field tenant_id, for consistency purposes.
- **trackme-limited/trackme-report-issues#745 - change - Virtual Tenant UI default settings - reducing the flex box default size from 374px to 350px**
  - This change updates the default flex box size in of the Virtual Tenants UI
- **trackme-limited/trackme-report-issues#747 - change - Data Hosts tracking (splk-dhm) - Change the default delay value from 3600 to 86400 seconds for Hosts Tracking**
  - In most use cases, it makes sense to increase the default delay value for Hosts tracking compared to Data source tracking.
  - Hosts tracking is a very different activity and most often, we need less restrictions when it comes to tracking the last time hosts have sent data as a default.
- **trackme-limited/trackme-report-issues#748 - bug - TrackMe Home UI - Flex Object creation screen can hide the bottom action buttons if the screen resolution is very low**
  - When creating a new Flex Object tracker, a very low screen resolution can prevent access to the bottom action buttons, unless zooming out from the Web Browser.
- **trackme-limited/trackme-report-issues#739 - feature - Flip events - Add a calculated disruption_time value in seconds within the flip results message when the entity switches from green to red**
  - When a given entity state changes from red to green, this changes adds a new calculated field disruption_time in seconds to ease further calculations of availability for users.
  - When the state are matching other conditions, the field has a 0 seconds value.
- **trackme-limited/trackme-report-issues#749 - enhancement - Data Sampling for Data Sources tracking - improve the message in the UI when sampling has not been processed yet**
  - When Data Sources tracking is pending and has not been processed yet, the message shown is simply N/A, and would deserve to be better explained.
- **trackme-limited/trackme-report-issues#750 - enhancement - TrackMe Health Tracker - Add a step to verify consistency regarding permanently deleted records**
  - When entities are permanently deleted, TrackMe stores records referencing these entities, so we will not discover these entities again.

  - In some circumstances such as restoring the KVstore collection, there could be a discrepancy due to the fact that entities are existing in the main KVstore collection, while in the same time listed as permanently deleted.
  - This additional verification ensures that if such a case happens, TrackMe would automatically purge associated records in the main KVstore.

- **trackme-limited/trackme-report-issues#751 - feature - Data Sources/Data Hosts tracking (splk-dsm/splk-dhm) - Add avg/max choices and additional choices with threshold in performance metrics tab**
  - This feature adds further more choices in the performing metrics screen, notably it adds choices between avg/max calculations at the time chart level, as well as options to include the current threshold for latency/delay.
- **trackme-limited/trackme-report-issues#752 - enhancement - Flex Objects library - Improve grouping for Splunk Cloud SVCs tracking use cases**
  - TrackMe has currently two builtin use cases for SVCs tracking in Splunk Cloud, the grouping should be improved so we dissociate global SVC tracking and per app SVC tracking.
- **trackme-limited/trackme-report-issues#754 - enhancement - Workload (splk-wlk)- When managing applications enablement, the tenant component summary should be refreshed**
  - In the workload component, TrackMe administrators can enable or disable entities at the application level.
  - When doing so, we should refresh the component summary as soon as possible, without waiting for the main tracker to perform it.

11.1.92 Version 2.0.98 - build 1722591315 (02/08/2024)

- SHA256: 371c327a8f492c07e57b60c8f8e505ecb8e9ba0aacca5d864ebfb31084612d26

Fixed issues:

- **trackme-limited/trackme-report-issues#700 - bug - SmartStatus alert action - Python exception in some circumstances when accessing the anomaly_reason**
  - The SmartStatus alert action can raise an exception while trying to investigate the anomaly_reason field.
- **trackme-limited/trackme-report-issues#702 - bug - Bulk edit - Critical priority button should be available in Bulk edit entities**
  - Critical priority was added in TrackMe 2.0.95, but in Bulk Edit we didn't add the associated button to mass update for the new priority.
- **trackme-limited/trackme-report-issues#705 - bug - Virtual Tenants UI - copy to clipboard TrackMe spl for Virtual Tenant creation can failed when executed due to boolean in JSON not properly handled**
  - When creating a new Virtual Tenant via the UI, one can at the end of the execution copy to clipboard the TrackMe SPL command that can be used to achieve the same creation in CLI.
  - However, an issue appears with the enablement of the component that remains in boolean and is not correctly handled in the SPL statement.
- **trackme-limited/trackme-report-issues#708 - bug - TrackMe Home UI Tabulator - regression due to trackme-limited/trackme-report-issues#697 for the management of encoded backslashes prevents the Alias to be inline editable**
  - A regression is affecting the Alias editable capability within the Tabulator due to the management of the encoded backslashes in issue#697.
  - This fix ensures the Alias is editable again within the Tabulator while still handling encoded backslashes.
- **trackme-limited/trackme-report-issues#710 - bug - Adaptive Delay (command trackmesplkadaptivedelay) - In some conditions, the backend tries to split a string into a list while already a list, raising a Python exception**

  - In the command trackmesplkadaptivedelay, we turn the anomaly_reason into a Python list from pipe separated, in some circumstances the field is already a list and the backend should check for the type of the object before applying the split.

- **trackme-limited/trackme-report-issues#719 - bug - TrackMe Notables and multi value fields in properties - mv fields should be properly handled in the properties, and stored as list within the JSON event**
  - When TrackMe generates a TrackMe notable event, the properties field contains all fields stored in the KVstore record for that entity.
  - Currently, multivalue fields are not correctly handled, and end in a pseudo multi value string structure instead.

- **trackme-limited/trackme-report-issues#725 - bug - Data Source tracking (splk-dsm) - Missing call to trackme_default_allow_adaptive_delay in the abstract macro called by the health tracker results in the field allow_adaptive to be empty in some conditions**
  - When the Health Tracker inspects offline entities (entities not actively generating data within the trackers scope), it shall call the macro that defines the default allow adaptive value.

- **trackme-limited/trackme-report-issues#727 - bug - SmartStatus - The use case search for future tolerance and the extraction of samples in the future is not consistent**
  - When SmartStatus is called and run the UC for data in the future (future over tolerance), one of the searches extracts a sample of events in the future.
  - The current search syntax is not ideally consistent and should be fixed for more meaningful results.

- **trackme-limited/trackme-report-issues#729 - bug - Feeds tracking (splk-dsm/dhm/mhm) - Auto-disablement of entities handled by the system wide configuration setting does not work as expected**
  - For feeds tracking, a system wide option was meant to allow automatically disabling the monitoring state of feeds tracking entities if the entity has not actively sent data since a certain period of time. (45 days by default)
  - However, the features is not properly working and does not influence the monitored state.
  - This change updates the process and transfer this work to the Virtual Tenant health tracker instead.
  - It fixes the action and enhances the worklfow by calling instead the associated API endpoint (rather than modifying silently the monitored_state), which also allows audting the change properly.
  - The period is also changed by default to 60 days, and the option is moved from General to splk-general for more consistency.

- **trackme-limited/trackme-report-issues#731 - bug - Machine Learning Outliers - Avoid generating an error with the command trackmesplkoutliersgetrules when dealing with Flex tracking that do not handle ML models**
  - Prevents generating an error message from this custom command and for Flex trackers that do not handle ML models.

- **trackme-limited/trackme-report-issues#732 - bug - TrackMe Home UI - Missing open in search for Notable events in the entitiy screen for all components**
  - When the mouse focus is on the Notable table in the entities screen, there should be an open in search option underneath the table.

- **trackme-limited/trackme-report-issues#734 - bug - TrackMe Home UI for splk-flx/splk-wlk - Machine Learning Outliers - In Adding model, the KPI dropdown selector does not populate properly**
  - When adding a new ML model for splk-flx/splk-wlk, the dropdown selector for the KPI selection does not populate due to a token generation issue.

- **trackme-limited/trackme-report-issues#735 - bug - TrackMe Home UI - regression in the tracking alert screen when clicking on a given alert to see the different charts, leading to none of the charts to be visible**

  - In the Home UI and the Tracking alerting tabs, a regression due due to a previous change (defining the default timerange via the Virtual Tenant account) leads to non of the charts to be visible when opening the activity of a given alert.

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#699 - change - Splunk UCC 5.8 decommissioned placeholder in entity from globalConfig.json**
  - Splunk UCC 5.8 removed the placeholder option for entity, this change ensures compatibility for the current and future releases of Splunk UCC.
- **trackme-limited/trackme-report-issues#701 - change - Upgrade of moment.js to last version 2.30.1 - Appinspect warning**
  - Appinspect warning has raised a message regarding the moment.js lib that needs to be upgraded.
- **trackme-limited/trackme-report-issues#703 - feature - Bulk Edit - Allow handling all options for lagging policies via bulk edit for eligible components**
  - In Bulk edit, this evolution adds a section providing full control for lagging monitoring policies for splk-dsm/splk-dhm.
- **trackme-limited/trackme-report-issues#707 - feature - TrackMe Home UI - Add buttons to expand all / collapse all grouped items in the Tabulator JS**
  - This feature adds “Expand all” and “Collapse all” buttons to allow expanding or collapsing items per group in the Tabulator JS table for the TrackMe Home UI.
- **trackme-limited/trackme-report-issues#709 - feature - Flipping events - Log the previous anomaly_reason when generating flipping events**
  - When TrackMe detects a change in the status of an entity, it generates a flipping event.
  - With this evolution, TrackMe will also log the previous anomaly_reason in addition to the new anomaly_reason.
- **trackme-limited/trackme-report-issues#711 - feature - All components - Tabulator in Home UI and entities grouping - Allows controlling entities grouping by configuration**
  - This new feature allows controlling the Tabulator group at the level of the Virtual Tenant account.
  - With this control, you can define a list of custom fields of your choice for multi-level grouping, or you can use expressions to compose a custom field for the Tabulator grouping.
  - This evolution provides a number of quick actions buttons in the Home UI, so you can change the grouping temporarily, for instance such as grouping by anomaly reason or priority, and allows calls back the default grouping.
- **trackme-limited/trackme-report-issues#712 - change - Address Appinspect warning about splunk_resource_usage**
  - Appinspect reports a warning about: default/props.conf contains a [splunk_resource_usage] stanza.
  - This change addresses this warning which is due to a field alias for the purposes of the Workload component (splk-wlk).
- **trackme-limited/trackme-report-issues#713 - enhancement - Outliers Anomaly detection - Before attempting to render an Outlier model, verify the true existing of the model to avoid failing the search if the model is not yet ready**
  - This enhancement allows TrackMe to verify the true existence and readiness of a Machine Outliers model before attempting to process with the render search, avoiding to generate a failing search in the system.
  - In some circumstances, TrackMe may spawn rendering searches while the model is not yet ready, it has not been trained yet or the KPI underneath does not generate metric points, which results in the generation of a failing search from Splunk perspective.
  - This evolution prevents this situation by performing a true verification of the model readiness.

- **trackme-limited/trackme-report-issues#714 - change - Virtual Tenant creation - When creating a new Virtual Tenant and splk-dhm is the only enabled component, automatically disable ML Outliers at the Virtual Tenant account so it can be qualified for further enablement**
  - This change automatically disables ML Outliers detection feature at the Virtual Tenant account level, and when creating a new Virtual Tenant where splk-dhm is the only enabled tenant, so it can be decided later on to enable it or not.
  - The purpose of this change is to reduce system pressure for users that do not qualify enough TrackMe configuration, leading to very large ML models volume to handle.
- **trackme-limited/trackme-report-issues#715 - change - Data Hosts/Metric Hosts tracking (splk-dsm/splk-mhm) - Add a safety regarding the presence of object_category before calling the command trackmesplkgetflipping**
  - This change adds a safety feature at the Python level to ensure the presence of a valid value for the field object_category in the tracker process execution, and before it calls the streaming command trackmesplkgetflipping.
  - The objective is to avoid an unexpected condition that could lead the search to fail.
- **trackme-limited/trackme-report-issues#717 - feature - Extend and normalize the tags feature to all TrackMe components**
  - This extends the concept of tags, handled at the entity level and by policies, for all TrackMe components equally. (this was first released for splk-dsm)
  - Decommissioning the historical enrichment tag for splk-dhm/splk-mhm which were made redundant since we introduced the CMDB integration, and for consistency purposes.
  - After the upgrade, the schema upgrade will upgrade necessary objects and create new objects for newly eligible components, there are no interventions required.
- **trackme-limited/trackme-report-issues#718 - enhancement - TrackMe REST API - Improve the behavior when forcing the deletion of a Virtual Tenant via the del_tenant endpoint**
  - When calling the del_tenant API endpoint in force mode, we should systematically try to clean any report that could be associated with the Virtual Tenant as per the upstream request.
  - Avoid systematically returning the status of failure, in force mode we will always try to delete knowledge objects that may not exist.
- **trackme-limited/trackme-report-issues#720 - change - TrackMe Home UI - When creating a technical component alert, the Ack mode should be a dropdown instead of a text box selector**
  - When creating a new alert for the component, the Ack mode selector is provided as a text input rather than a more adapted dropdown selector as only two options are possible.
- **trackme-limited/trackme-report-issues#722 - enhancement - Tabulator in Home UI - Add control against the allow adaptive column in the Tabulator, Add missing column for more consistency in splk-dsm/splk-dhm**
  - This enhancement adds the allow adaptive thresholding column to the Tabulator for splk-dsm/splk-dhm for consistency purposes.
  - It also adds some missing columns regarding lagging policies features for these components, and makes column size and titles more consistent.
- **trackme-limited/trackme-report-issues#723 - change - TrackMe persistent backend - Address some inconsistency in the list of the persistent fields per component**
  - TrackMe uses a library Python file that defines the list of fields which should be considered as persistent.
  - This is used to ensure that we detect a modification of these fields while a concurrent update logic (tracker) can be running, so these changes are not lost.
  - This change addresses some inconsistency in these lists.

- **trackme-limited/trackme-report-issues#721 - enhancement - Role Based Access Control (RBAC) - Ensure vtenant main collections and vtenant summary main collections are made readable to roles added to Virtual Tenants**
  - TrackMe's built-in mains KVstore collections and transforms are by default readable to a few specific roles, admin/sc_admin and TrackMe built-in roles.
  - When handling RBAC to allow access to foreign roles, we should also check and grant read access to these collections for RBAC to work as expected without further intervention from Splunk admins.
- **trackme-limited/trackme-report-issues#724 - enhancement - Maintenance mode - Access as a non-admin should ideally show an informational message rather than a blocked error**
  - When accessing the maintenance mode dashboard as a non-TrackMe admin, we should ideally show an information message, instead of having the dashboard blocked with an insufficient permission issue.
- **trackme-limited/trackme-report-issues#726 - enhancement - Virtual Tenants creation - Safer verification and management of requested Virtual Tenant identifier**
  - When creating a new Virtual Tenant, there are some conventions that TrackMe will apply, such as forcing lowercase, using hyphens as the separator.
  - In some conditions, the current verification can be bypassed leading to issues during the Virtual Tenant creation. This enhancement ensures a safer and more consistent approach.
- **trackme-limited/trackme-report-issues#728 - feature - Allows defining a Virtual Tenant wide indexed constraint for splk-dsm/splk-dhm that automatically influences associated generated searches created by TrackMe, such as UI search action button or SmartStatus**
  - This feature allows defining at the Virtual Tenant level a custom indexed constraint, which is then automatically used while defining automated searches such as the search button in the Home UI, or searches created by the SmartStatus.
  - This can be useful in a scenario where each Virtual Tenant is associated with a custom indexed constraint, such as referring to a splunk_server_group or any other required indexed string.
- **trackme-limited/trackme-report-issues#730 - change - Moving the default future tolerance system wide option from General to splk-general for consistency purposes**
  - The system wide option Future indexing tolerance is moved from General to splk-general for more consistency in the options.
- **trackme-limited/trackme-report-issues#733 - enhancement - Flex Object library - Add dcount host to the drop detect use cases (splk_detect_drop_events_count_absolute/splk_detect_drop_events_count_rolling)**
  - Improvement to the Flex Obect library use cases related to drop events detection, adding the dictinct count host KPI.

11.1.93 Version 2.0.97 - build 1720562684 (09/07/2024)

- SHA256: 44c4a80e9584f546583f4cc43661a45c499f05b50f26e31159fa419225ced26e

Fixed Issues:

- **trackme-limited/trackme-report-issues#669 - bug - Flex Object (splk-flx) - When triggering due to Anomaly Outliers and when not actively managed by a tracker, a Flex Object entity will not return to green state if the Outliers conditions is fixed**
  - If a Flex entity turns red due to Outliers condition, and if that same entity is not actively managed by a tracker (for instance if the tracker is time conditioned for some reasons it's not active in the tracker time window), TrackMe will not update the status of the entity properly.
  - This is due to the fact that we should take into account the flag field status in addition with the object_state especially for Flex objects

- **trackme-limited/trackme-report-issues#670 - bug - Outliers Anomaly detection - Singles in the simulation screen do not honour the simulation screen time range selector and use the front page UI time range instead**
  - When performing Outliers simulation, there are different single views designed to show the key behaviours and statistics.
  - However, the searches driving the calculations underneath do not honour the simulation specific time range selector, and instead obey to the time range selector of the main entity screen.
- **trackme-limited/trackme-report-issues#672 - bug - Outliers Anomaly detection - Disabling ML detection on a per entity basis is not properly honoured by TrackMe**
  - ML Outliers Anomaly detection can be disabled on a per entity basis.
  - There is a regression in the current release of TrackMe and the Decision Maker component which prevents this setting from being properly honoured.
  - This fix addresses the issue and also provides an enhanced behaviour making this immediately reflected.
- **trackme-limited/trackme-report-issues#674 - bug - TrackMe State events - the sourcetype trackme:state should by default expect object_state and not state as the field name containing the object_state (default configuration for allow list in trackme_settings.conf)**
  - When trackers are executed, TrackMe generates state events in trackme:state
  - The fields behaviours are dicted by the TrackMe configuration, however the allow list currently expects the field "state" where we should expect "object_state" as per TrackMe's convention
- **trackme-limited/trackme-report-issues#675 - bug - TrackMe Flip events - For consistency purposes, TrackMe should also include the anomaly_reason field in the event generation**
  - When TrackMe entities experience a status change, a corresponding flipping event is generated with the sourcetype trackme:flip
  - The field anomaly_reason is a key field in TrackMe's convention, it is part of the flipping message but should also be included on its own for consistency purposes with other TrackMe concepts.
- **trackme-limited/trackme-report-issues#676 - bug - Data Source tracking - Tags manual - Creating manual tags fail if there are no tags policies created for the tenant**
  - If there are no tags policies in the Virtual Tenant, attempting to create manual tags for a given entity fails due to a Python raise condition, leading to the an error message instead.
- **trackme-limited/trackme-report-issues#685 - bug - Bulk Edit - Select all tick box in the Tabulator does not honour table filters and leads to all visible entities to be selected**
  - When performing bulk edit entities in TrackMe, there is an option "tick all" which allows selecting all entities.
  - However, there has been a regression, and TrackMe does not apply current filters from the Tabulator table, leading to all entities to be selected by the tick all checkbox, rather than only resulting entities.
- **trackme-limited/trackme-report-issues#690 - bug - Virtual Tenants UI - Avoid switching between operation and degraded on the Virtual Tenant flex box when there is an actual tracker in failure**
  - In TrackMe's Virtual Tenant UI, if an issue is affecting a tracker, in some circumstances the degraded cross information on the Virtual Tenant box will switch from degraded to operation, then at back to degraded.
  - This is due to a Python flaw in the logic of the process_exec_summary function in lib/trackme_libs_load.py.
- **trackme-limited/trackme-report-issues#691 - bug - Virtual Tenants UI - screen TrackMe Tenants Operational health statuses can have the Tabulator table hidding buttons with large number of tenants**
  - In the Virtual Tenants UI and when there are a large number of tenants, the bottom buttons of the screen TrackMe Tenants Operational health statuses may be hidden by the table.

- **trackme-limited/trackme-report-issues#693 - bug - Metric Hosts tracking (splk-mhm) - When creating a new hybrid tracker on a remote target, the break by statement is invalid leading to no results for the tracker**
  - This bug affects the creation of a new hybrid tracker for splk-mhm when the target is a remote target.
  - The break by statement generated is invalid and missing the host call in the statement.
- **trackme-limited/trackme-report-issues#696 - bug - Hybrid Tracker (all components) - Ensure to safety truncate the submitted tracker name to 40 chars at the API level and prevents from any risk of failure due to Splunk 100 chars limit**
  - Handles conditions where the submitted tracker ID could lead to a report name requested by TrackMe's API that goes beyond the 100 max chars accepted by Splunk

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#668 - change - Tabulator - Upgrade to Tabulator 6.2.1**
  - Upgrade of Tabulator JS to release 6.2.1
- **trackme-limited/trackme-report-issues#619 - feature - Maintenance Mode - Support for enabling Maintenance Mode with selection of applicable tenants, support for Maintenance Knowledge DataBase #619**
  - This introduces support for selective Maintenance Mode on a per tenant basis, you can enable the Maintenance Mode for all tenants (default) or a list of applicable Virtual Tenants.
  - Support is also added to the Maintenance Knowledge DataBase in TrackMe, as well as automatically influencing SLA calculations depending on if the maintenance period for applicable for the entity tenant.
  - After the upgrade to 2.0.97, TrackMe's schema upgrade will automatically update TrackMe alerts to call the new macro trackme_apply_maintenance_mode
- **trackme-limited/trackme-report-issues#650 - change - Overview eventcount timechart calculations and Performance metrics tab timechart calculations for spl-dsm/splk-dhm, for consistency purposes, use a sum calculation per metrics at the timechart level instead of an avg, as it happens for the latest_eventcount_5m**
  - In the overview chart tab, when looking at increased timeranges, we should rather use a sum calculation for eventcount metrics for consistency purposes.
  - In the Performance Metrics tab, and for splk-dsm/splk-dhm. TrackMe uses a "sum(latest_eventcount_5m) as latest_eventcount_5m" where others metrics are calculted using an avg.
  - None of these are technically false and just different reading, but for users going trough a basic approach of comparing true eventcount, this can be confusing.
- **trackme-limited/trackme-report-issues#671 - change - Outliers Anomaly detection - Allow full control on the period_calculation definition**
  - This update allows complete control for the definition of the period_calculation.
  - The time quantifier period expression can be submitted without pre-defined periods, for default models generation and on per model basis.
- **trackme-limited/trackme-report-issues#677 - change - Hybrid Trackers creation screen - Automatically pre-fill the tracker name with a randomly generated ID**
  - When creating hybrid trackers, a text input is expecting a name to be choosen for this tracker.
  - To improve the global user experience, automatically prefill this input with a randomly generated identifier.
- **trackme-limited/trackme-report-issues#678 - enhancement - Flex Objects (splk-flx) and TrackMe KPI generation - Support for time definition in metrics generation at ingest time**
  - This enhancements provides support in TrackMe to generate metrics with an upstream value for the metric time stamp.

  - This allows supporting use cases where the time in the Tracker logic is not equal to when the tracker is executed, but rather part of the SPL statement.

- **trackme-limited/trackme-report-issues#679 - feature - Flex Object use cases library - New use cases splk_detect_daily_variations_volume_global / splk_detect_daily_variations_volume_index**
  - These two new use cases are designed to leverage the Splunk license indexing logs to track the daily absolute amount of data indexed globally on the license pool, and on a per index basis.
  - These KPIs then are used to train Outliers Models with the goal of detecting abnormal decrease/increase of indexing volume per entity.
- **trackme-limited/trackme-report-issues#680 - enhancement - Outliers Anomaly detection - Add %w as an option for the time_factor (time factor influenced per week day)**
  - This enhancement adds support for a time factor option per week day (%w) for the configuration of Machine Learning models via TrackMe's UI
- **trackme-limited/trackme-report-issues#681 - feature - TrackMe Home UI and Virtual Tenant account preferences - Add time range selections up to 1y and allows defining the default time range at the level of the Virtual Tenant account preferences**
  - Adds new period for 6 months (180d) and 1 year (365d) in the time ranger selector of TrackMe's main UI.
  - Allows defining on a per Virtual Tenant account the default time range to be selected when accessing entities.
- **trackme-limited/trackme-report-issues#682 - enhancement - Common Information Model compliance (splk-cim) - Improvement of the preview search functions and direct links to open searches in a new window**
  - Add the from datamodel search
  - Add direct links button for from / datamodel / tstats searches which dynamically open the search into a new window with local/remote account support
- **trackme-limited/trackme-report-issues#684 - feature - Acknowledgment management - Expire Ack on anomaly reasons changes so TrackMe can raise a new alert when conditions for alerting have changed**
  - This new feature allows TrackMe Ack to be influenced by the change of anomalies affecting entities.
  - Conditioned by system level configurable options (See Configure / General / Expire Ack on anomaly reason change behaviour, Expire Ack on anomaly reason change min time since, Expire Ack on anomaly reason only for auto ack), this new feature completes and enhances the Ack capabilities in TrackMe.
  - If an entity that turned red due to an Outliers detection for instance, and later on is also affected by an additional condition such as a lag breach, the Ack will be automatically expired so that a new alert can be raised transparently by TrackMe.
- **trackme-limited/trackme-report-issues#687 - enhancement - Data Hosts tracking (splk-dhm) - The Performance metrics tab in entity overview should include the Delay Metrics and also include dynamic explanations as with splk-dsm**
  - In Overview entity then Performance Metrics tab, we should for splk-dhm provide access to the Delay metrics, as well as explanations regarding these metrics calculation, similarly to splk-dsm
- **trackme-limited/trackme-report-issues#688 - enhancement - TrackMe Tracker executor backend - Improved detection of silently failing trackers**
  - When TrackMe executes trackers, this execution goes through a quality and review backend with the custom command `trackmetrackerexecutor`
  - Especially, this process tracks for execution failures, generates run time metrics for TrackMe's trackers and feeds the Tenant operation statuses.
  - In some circumstances, some types of execution failres can happen silently and the current version of the backend does not notice it, this fix slightly enhances and is capable of detecting any conditions leading to the failure of the tracker.

- **trackme-limited/trackme-report-issues#689 - enhancement - TrackMe Health Tracker - Add an additional safety check to identify and purge unexpected foreign records that would have been added by mistake to a main data KVstore collection**
  - Each tenant has a TrackMe Health Tracker which performs various maintenance routines, in this issue we add an additional action to check for the presence of unexpected foreign records in the main KVstore collection, and purge these records automatically if any.
  - Foreign records could have been added by mistake when manipulating KVstore collections, and would lead to be blocking many logics in TrackMe.
- **trackme-limited/trackme-report-issues#692 - feature - TrackMe Virtual Tenants - New API endpoint to clear the Virtual Tenants Operation Status actionable through the Virtual Tenants UI**
  - This new API endpoint allows TrackMe Admins to clear the Virtual Tenants Operation Status and optionally request the imediate refresh through the execution of the tenant's health tracker.
  - In the Virtual Tenants UI, this feature can be requested via the screen TrackMe Tenants Operational health statuses for all tenants, or a selection of tenants with the option to execute or not the health tracker.
  - Clearing the Virtual Tenant Operation status can be useful when dealing with a degraded Virtual Tenant which status is blocked due to some issues.
- **trackme-limited/trackme-report-issues#694 - feature - SOAR Monitoring - Adding Flex Object UC to track SOAR/Splunk forwarding integration (splk_soar_forwarding_splunk)**
  - This additional Flex Object use case for SOAR focusses on tracking the SOAR/Splunk forwarding integration
- **trackme-limited/trackme-report-issues#695 - feature - Flex Object library - New Flex Use Case splk_splunk_infra_log_level_variations which deals with Splunk logs and their logging level and use Machine Learning to detect abnormal behaviours of your Splunk instances and deployments**
  - This new Flex Object use case tracks Splunk internal log events and their associated logging level to detect suspscious trends, which are symptomatics of Splunk behaving improperly and facing or about to face serious issues.
  - To achieve this, we leverage TrackMe's Flex component and our Machine Learning implementation, we then track trends notably of errors in Splunk logs to alert when an abnormal amount of errors is detected.
- **trackme-limited/trackme-report-issues#697 - bug - All components - Entities containing backslashes generate all sorts of issues in TrackMe, this condition can notably be encountered in Workload (splk-wlk) with very bad report naming**
  - Entities ending up with backslashes can generate various issues in TrackMe, especially in advanced features such as ML Outliers or Metadata tracking for splk-wlk.
  - This issue addresses this problematic by encoding backslahes at the discovery Python phases, and decode transparently for users as needed.
- **trackme-limited/trackme-report-issues#698 - enhancement - Workload (splk-wlk) - Management of dupplicated entities at the phase of the health tracker execution**
  - In the Workload component (splk-wlk), the Health Tracker verifies for duplicated entities, and deletes automatically one of the duplicated randomly.
  - However, it can happen in some conditions that we will keep continously deleting the wrong entity which then keeps being re-created.
  - For more consistency, this fix will allow TrackMe to purge both concerned entities, so only the right one gets re-created accordingly.

11.1.94 Version 2.0.96 - build 1718623969 (17/06/2024)

Major UI filtering performance improvements

- This release introduces major performance improvements in the TrackMe main UI, especially when performing entity filtering, which is now nearly instantaneous, regardless of the collection size.
- These improvements are made possible by the switch to client-side (local) pagination and filtering in Tabulator, which can now also be controlled through general and per-tenant parameters since this release.

- SHA256: 02e835c27b2c681a7ad89f0c9e100a4a6317e824bb9fb3d21286e1108ceabee0

Fixed issues:

- **trackme-limited/trackme-report-issues#657 - bug - Outliers Anomaly Detection - TrackMe does not honour the method_calculation defined at the model level when performing training and rendering of the model #657**
  - On per model basis, a method calculation can be applied at the level of the mstats search, which will be associated with the KPI span to influence the Outliers root calculation.
  - However, currently TrackMe does not honour properly the method calculation due to a bug in TrackMe's Outliers Python library.
- **trackme-limited/trackme-report-issues#659 - bug - Outliers Anomaly Detection - Default system parameters should not require a value for static LowerBound/UpperBound #659**
  - In Configuration / splk-outliers-detection, saving parameters should not require a value for LowerBound/UpperBound
- **trackme-limited/trackme-report-issues#665 -bug - TrackMe Home User Interface - Outliers Anomaly Detection appearance remaining issues after performing training via the Manage Outliers UI**
  - Folllowing fixes from trackme-limited/trackme-report-issues#638, there are remaing issues and conditions leading to the Outliers MLTK chart to fail appearing properly after a training is made through the Manage Outliers UI screen.
  - This is due to the fact that refreshing the search underneath the MTLK Outliers charts while the chart is not visible yet leads to this issue.

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#658 - enhancement - Outliers Anomaly Detect - Add additional options for the Outliers kpi_span #658**
  - Complete selectable options for the kpi_span per model with additional values up to 24h
- **trackme-limited/trackme-report-issues#660 - feature request - Filter functions - Add in the "By Acknowledgment state" new options to filter on Acknowledged entities per priority**
  - In TrackMe's UI, one can use filter functions to prefilter on multiple conditions at once.
  - This feature requests is to add filter functions for Acknowledged entities based on priority filters
- **trackme-limited/trackme-report-issues#661 - enhancement - TrackMe Home UI performance - client side pagination and filtering in Tabulator for largely improved performances especially when filtering**
  - By implementing client side (local) pagination and filtering, this release introduces major performance gains in TrackMe main UI, especially when performing entities filtering based on any available simple or complex conditions.
  - The pagination mode and pagination size can now also be controled at the level of the Virtual Tenant account, with the base general configuration that can be customised when creating tenants, and once the tenant has been created through the Virtual tenant account
  - These enhancements bring major performance improvements to TrackMe, slightly improving the end user experience.

- **trackme-limited/trackme-report-issues#662 - enhancement - Virtual Tenants UI - Make the results from focus searches more readable and valuable**
  - In the Virtual Tenants UI, when putting the focus on a given tenant / status by priority, a search runs and provides an high level overview of underneath entities.
  - The purpose of this issue is to simpify the approach to get more readable and valuable results, from this release the search will generate a simpler list of concerned entities ordered by their flip status. (ordered by the last time these entities have had a status changed)
- **trackme-limited/trackme-report-issues#663 - enhancement - Adaptive Thresholding - Allows to control the review period through the argument review_period_no_days at the level of the adaptive tracker**
  - In this issue, we introduce a new argument to the Adaptive delay custom command which controls the period of time used to identify entities to be reviewed over time following a change made the Adaptive Treshold backend.
  - The argument review_period_no_days accepts 3 period options: 7, 15 or 30 days for the period of review.
  - After the upgrade, TrackMe will automatically update existing and active trackers through the schema upgrade.
- **trackme-limited/trackme-report-issues#664 - change - Adaptive Thresholding - Change of the default period for review from 7 days to 30 days following the introduction of the new option review_period_no_days**
  - Associated with the new argument review_period_no_days, the default is now set to 30 days to improve the behaviour over time of the Adaptive Tresholding backend, and ensure we review for long period enoughs entities that have been modified by the backend.

11.1.95 Version 2.0.95 - build 1718137396 (11/06/2024)

New priority level with critical priority

- This release introduces an additional priority level "critical" for TrackMe entities.
- This will provide more flexibility and consistency for customers to leverage various CMDB and logics, and alert with different types of actions depending on the importance of associated entities.
- You may need to review your current alerts, and include the new priority level in your alerting logic.

- SHA256: 10f1318c0895f7cd4d648f1a9e48795858ebc5991c0c27447ace816058a9c84a

Fixed issues:

- **trackme-limited/trackme-report-issues#638 - bug - TrackMe Home User Interface - Outliers Anomaly Detection chart may not show up properly in some circumstances, as after Models modifications or attempting to handle a non valid model #638**
  - The Outliers Anomaly Detection tab triggers when actioned by the user, and will display the models statistics and the Outliers chart.
  - In some circumstances, such as after a modification of a model or after attempting to display an entity with no models, the chart fails to display properly and will not display until the UI is fully refreshed.
  - This is caused by attempting to enable / disable the containing HTML div at the CSS level which does not behave well with the MLTK viz chart.
- **trackme-limited/trackme-report-issues#641 - bug - Metrics hosts monitoring (splk-mhm) - Get component loads to fail entities due to regression since 2.0.93**
  - Entities fail to be loaded properly for splk-mhm due to the load component librairies evolutions
  - The component incorreclty attempts to load Outliers KVstore collections, which is not applicable to splk-mhm resulting in failure to load entities when opening the UI

- **trackme-limited/trackme-report-issues#642 - bug - Data Source monitoring (splk-dsm) - Data Sampling status and enablement should be immediately reflected by the DecisionMaker**
  - When modifying the Data sampling feature enablement, this should be immediately and properly reflected in the DecisionMaker results as well as the TrackMe UI screen.
- **trackme-limited/trackme-report-issues#645 - bug - REST API - Update endpoints calling the method generic_batch_update will not take into account replacements with empty values, which impacts reset actions such as in splk-dhm/mhm**
  - When calling REST API endpoints for update purposes, TrackMe implements a batch update method to update KVstore records as fast as possible.
  - This Python method is called generic_batch_update and currently ignores replacement of values by an actual empty value.
  - However, doing so causes a regression for some specific endpoints such as the reset endpoint for splk-dhm/mhm.
  - This fix updates the function to call a Python native object method instead to update the records transparently.
- **trackme-limited/trackme-report-issues#652 - bug - TrackMe logs rotation should ideally be taken into account for Splunk ingest purposes #652**
  - When TrackMe logs are rotated, our props.conf should take into account incremented log.* files

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#636 - enhancement - Splunk SOAR monitoring of Automation Brokers - enhancement of the REST API endpoint access to retrieve the status of the automation brokers to avoid hitting some scenarios where the normal REST API endpoint misses some statuses errors of the brokers #636**
  - The Flex Object tracker use case for SOAR Automation Brokers monitoring allow retieveing and acting when the SOAR Automation Brokers are not in active state
  - In some edge use cases, the SOAR automation_brokers REST API misses an offline status of the Automation Broker wrongly if the API endpoints is not accessed with some additional arguments in the parameter of the REST API call
- **trackme-limited/trackme-report-issues#637 - enhancement - Acknowledgements - Safer code to handle unexpected records with no object_category #637**
  - if Ack records are unexpectly created without a valid object_category, the Ack tracker would not handle this issue properly, and would attempt and fail to retrieve the corresponding record in the data collection.
  - This would lead the tracker to fail expiring non corrupted Ack records.
  - This evolution ensures that any corruputed record would be purged accordingly, and will avoid the tracker from failing to manage other valid Ack records
- **trackme-limited/trackme-report-issues#639 - feature - Bulk edit entities - For Data Sources monitoring (splk-dsm), allows to manage Data Sampling via bulk edit**
  - Manage Data sampling actions via bulk edit: enable / disable / run / reset
- **trackme-limited/trackme-report-issues#640 - enhancement - Bulk edit - Ensures scroll bar would appear if the screen resolution is too low**
  - If the screen resolution is too low, ensure to load the vertical scrolling bar to avoid truncating the bulk edit screen
- **trackme-limited/trackme-report-issues#643 - enhancement - SOAR Automation Broker active management - Add a safety layer for ignoring typical fields containing secrets in the JSON post response when updating assets, in addtion with existing automated salted fields exclusion #643**
  - When updating SOAR Assets, and when not using a Vault for password management, we must not include secrets when performing the POST call.

  - TrackMe already automatically excludes fields which have been salted by SOAR, however as an additional safety and to be retro-compatible with older Assets defined, we also exclude typical fields: apikey,api_key,password,auth_token,client_secret
  - This can be controled at the level of the POST call using the option: assets_update_forbidden_fields

- **trackme-limited/trackme-report-issues#646 - enhancement - Data Host/Metric host tracking (splk-dsm/mhm) - Behaviour improvements for the reset actions**
  - The reset actions can be used to reset the current knowledge for a given entity when it comes to indexes, sourcetypes for splk-dhm or metric categories for splk-mhm.
  - The current behaviour can be improved to better cleanup the associated fields with a more consistent approach.
  - This enhancement also avoids removing the visibility for the entity that was reset until knoweldge is built again.
- **trackme-limited/trackme-report-issues#647 - enhancement - TrackMe Notable events - automatically parse the anomaly_reason and turn into a list so Splunk can extract it as an mvfield**
  - The anomaly_reason a primordial field in TrackMe which is used by the DecisionMaker to insert all conditions encountered for a given entity, at the lowest level it is a native Python list.
  - However, when generating TrackMe notable events, the field is turned into a pipe separated string.
  - To allow automated mv structure extraction in Splunk, the field should rather be turned back into a list in the JSON structure.
- **trackme-limited/trackme-report-issues#648 - enhancement - SmartStatus - smartstatus_investigations_uc_dsm_latency and smartstatus_investigations_uc_dhm_latency should rather leverage tstats based search to slightly reduce associated costs**
  - When the SmartStatus is executed and when the entity is red for latency reasons, we currently generate a raw search for a full accuracy regarding the latency calculation.
  - However, these searches can be slightly expensive at high scale for a relative value, in this issue we migrate the generated searches to a tstats based search instead.
- **trackme-limited/trackme-report-issues#649 - enhancement - SmartStatus alert action - Protect Splunk workload and prevent SmartStatus alert action from being executed more than once per 24 hours per entity**
  - In some circumstances such as if a TrackMe alert was badly setup without leveraging TrackMe's Ack concepts, or increasing the suppression period, the SmartStatus alert action could be triggered and executed more than wanted, which in turn could affect Splunk workload and generate more activity than required.
  - In this issue, we introduce a concept that keeps track of the last seen execution per entity, and we will automatically skip the SmartStatus action if the action has been executed in the past 24 hours already.
- **trackme-limited/trackme-report-issues#651 - enhancement - Add sum and min as calculation methods when missing as selectable options in Outliers configuration and other dropdown in TrackMe's UI**
  - In Outliers calculation methods configuration (default configuration and per model fine tuning), the sum and min options should be available.
  - In selectable options parts of drildown selectors such as in Flex Objects, these methods should be available.
- **trackme-limited/trackme-report-issues#653 - change - Anomaly Outliers detection - Add -15d in selector for period of calculation, change -360d to -365d for consistency when requesting 1 year for the period, reflect the same periods in the configuration screen for default assignment**
  - Add -15d in selectable options
  - Swtich -360d to -365d for consistency regarding a year of relative period of time for the calculation period
  - Reflect the same options in the configuration screen for default perod assignment for consistency

- **trackme-limited/trackme-report-issues#654 - feature - Entities priority management - Add a new priority with critical priority to provide more flexibility in TrackMe entities management**
  - This release introduces a new priority level "critical" for TrackMe entities.
  - This will provide more flexibility and consistency for customers to leverage various CMDB and logics, and alert with different types of actions depending on the importance of associated entities
- **trackme-limited/trackme-report-issues#655 - feature - Priority management - Migrate priority management from macro based to a per Virtual Tenant account option for more flexibility**
  - The priority management is being migrated from the macro trackme_default_priority to an easily configurable option per Virtual Tenant account.
  - Users can now define the default priority at the time of the creation of the Virtual Tenant, or any time in the Configure / Virtual Accounts configuration screen.
  - This provides a more flexible and more consistent approach to the priority management in TrackMe.
- **trackme-limited/trackme-report-issues#618 - Feature Request - Alert configuration "trigger on outliers" and "trigger on sampling" behaviour would lead to miss other anomaly reasons**
  - When creating a TrackMe alert, one can select to trigger or not against Outliers, and Sampling. (note: Sampling is splk-dsm only)
  - However, the current logic can be much improved to also handle use cases where we have a mutli-detection, and we have more than anomaly in addition with Outliers/Sampling.
  - With this evolution, TrackMe will parse automatically the anomaly_reason as part of the trackmegetcoll output, adds a new field for the count of anomaly_reason, and finally updates the method when creating a new alert.
- **trackme-limited/trackme-report-issues#656 - change - CIM Compliance trackers (splk-cim) - Licensing restriction increase to 32 trackers for Enterprise Edition customers**
  - We are increasing the max number of CIM Compliance trackers for Enterprise Edition customers from 16 to 32.

11.1.96 Version 2.0.94 - build 1716849152 (27/05/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 465a36c35cc9a161218a9b9c7ff14204d8fd896d72878c943277fc9b5664d4ff

Fixed issues:

- **trackme-limited/trackme-report-issues#626 - bug - SOAR Automation Broker high availability management - update of the broker will reset unexpectly any secrets of the assets for users not using a Password Vault #626**
  - When performing an active update of the automation broker via the Flex Object use case, we perform an update of the Asset configuration via the SOAR API to swtich the broker from A to B.
  - For users not using a Password Vault, SOAR handles any credential such as an API token, the token is salted in the data.
  - When performing the REST POST call to the API, we should remove any field in the JSON structure which starts with a "salt:" to avoid resetting this secret unexpectly, or the asset connectivity is lost.
  - This only applies to internal SOAR secret management, in the sense that SOAR customers using a Password Vault are not affected by this issue.
- **trackme-limited/trackme-report-issues#628 - bug - error when clicking on refresh entities in TrackMe UI when looking at a given entity: search_kv_collection() got an unexpected keyword argument #628**

  - Issue happens when clicking on refresh when looking at a given entity
  - This is a regression introduced in TrackMe 2.0.92

- **trackme-limited/trackme-report-issues#629 - bug - Adaptive Tresholding - Avoid attempting to take into account during the review a feed that was previously updated but stop indexing to Splunk in the past 7 days #629**
  - When the adaptive threshold backend updates an entity, this entity automatically enters the review phase to ensure we take into account updated behaviours, such as an outage that was resolved in the meantime.
  - However, if an entity that was previously updated stop indexing data to Splunk, we should not take it into account anymore if it didn't index any event for the past 7 days to avoid raising an exception while accessing the adaptive_delay result.
- **trackme-limited/trackme-report-issues#631 - bug - Workload (splk-wlk) - Regression in TrackMe 2.0.93 due to missing fields in lookup transforms leading to status not met instead of advanced status distinction #631**
  - In TrackMe 2.0.93 and to address some CPU & Memory pressure, we have swtiched the base logic to access KVstores to a search based approach.
  - This impacted the Workload component due to missing fields in the Lookup transforms, which cannot be access in a search unless part of the transform, this had lead to a status not met instead of the detailed statuses.
  - Once upgraded, the TrackMe health tracker schema upgrade routine will update the lookup transforms accordingly with no action required.
- **trackme-limited/trackme-report-issues#633 - bug - Outliers detection potential regression in TrackMe 2.0.93 leading to isOutlier not reported at DecisionMaker time**
  - Due to search based approach when accessing KVstore records in TrackMe 2.0.93, in some circumstances it is possible that detected Outliers do not get reported while loading entities.
  - This issue introduces a robust and consistent approach at the Python level to lookup Outliers, similarly to other phases in the TrackMe Decision Maker.

Enhancements, changes and new features:

- **trackme-limited/trackme-report-issues#627 - enhancement - trackmehealthtracker - Optimize run time, costs and behaviour of the inspect_collection phases #627**
  - The schedule job trackmehealthtracker is responsible for various maintenance routines, one of these is called "inspect_collection"
  - This maintenance routine verifies the consistency of TrackMe entities statuses (object_state) between the KVstore record view and the realtime view from TrackMe's DecisionMaker processes
  - The TrackMe DecisionMaker process is a suite of many Python functions depending on the time of component which apply conditions for monitoring, such as delay/latency rules, logical group mapping, Outliers detection and so forth
  - In this issue, we optimized this specific process with a faster and lighter search based approach to load the KVstore raw collection, load the TrackMe DecisionMaker view (using the trackmedecisionmaker streaming custom command) then performing the comparison
  - The objective of this update is therefore to reduce the costs of this step, reduce the global runtime of the trackmehealthtracker job and avoids generating skipping search
- **trackme-limited/trackme-report-issues#630 - enhancement - Adaptive Threshold - Avoid attempting to inspect entities that have not actively generated data in Splunk for a minimum period equivalent to the max_delay_sec argument given to the backend #630**
  - For optimization and costs reduction purposes, TrackMe's adaptive threshold backend should not attempt to inspect entities which are not actively sending data to Splunk.
  - The current behaviour implies that we may continue to attempt to inspect entities that are monitored actively but without any recent ingest activity (past 7 days)

  - To improve consistency while reducing TrackMe's workload, the Adpative Threshold backend should ensure to take into account entities in the initial inspection phase only if the current delay is < max_auto_delay_sec (default to 7 days)

- **trackme-limited/trackme-report-issues#632 - enhancement - Decomission ML models orphans cleanup from the trackmehealthtracker as it is also handled via the general health tracker #632**
  - In TrackMe 2.0.84 was introduced the general health tracker which is executed once per day amongst all Virtual Tenants.
  - Especially, this job handles all cleaning related to Machine Learning, such as detecting and purging Orphans models. (models which entities have been purged, or the tenant was purged)
  - Previously, this activty was handled by the tenant level health tracker, this is not required any longer and we can save from this activity to optimize and reduce TrackMe's workload.

11.1.97 Version 2.0.93 - build 1716457845 (23/05/2024)

Hint

High CPU and Memory pressure regression from TrackMe 2.0.92: this release addresses several important issues leading to extra CPU and memory pressure introduced with TrackMe 2.0.92

- SHA256: 9d6d5cd975f6f7fcbb1966b212206f77a414938b5f5b0446a0bded64233f550c

Fixed issues:

- trackme-limited/trackme-report-issues#620 - bug - Option sla_default_threshold in sla is not used on purpose and should have been removed from the configuration UI #620
- trackme-limited/trackme-report-issues#621 - bug - Virtual Tenants UI - If using legacy TrackMe load mode, this should also apply to automated refresh #621
- trackme-limited/trackme-report-issues#622 - bug - Maintenance mode management UI - Web browser over consumption over time due to resources leak with Javascript autorefresh #622
- trackme-limited/trackme-report-issues#623 - bug/enhancement - Performance and footprint reduction at high scale (More than 10k/100k collections) - changes introduced in TrackMe 2.0.92 can lead to extra CPU and memory consumption #623
- trackme-limited/trackme-report-issues#624 - bug - CIM Compliance tracking (splk-cim) - object_category should be in the collections for consistency purposes regarding all other components, its lack currently impact notables and acknowledgement #624
- trackme-limited/trackme-report-issues#625 - bug - CIM Compliance tracking - When creating a notable or SLA alert, components alert actions are wrongly added to the alert #625

11.1.98 Version 2.0.92 - build 1715771041 (15/05/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 800d2adbf600d31124558b3dff4104bc3bb41e405365c284077ddc1500e38864

Fixed issues:

- trackme-limited/trackme-report-issues#617 - bug - Regression with the usage of numpy which impacts schedule logic where we limit their run time - due to an Appinspect restriction and numpy storing libs in a hidden directory which was removed automatically by our automation, this leads to the custom command to fail at exec time #617

11.1.99 Version 2.0.91 - build 1715725834 (14/05/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 0d41329cd50ed1531b4548ff0e7136b293dec45f25eec57993b315ad5aa0e6ee

Fixed issues:

- trackme-limited/trackme-report-issues#601 - bug - Logical Group REST API - avoid raising an exception when groups members, or green / red members are unexpectly null #601
- trackme-limited/trackme-report-issues#603 - bug - Prevents an exception in the REST API endpoint post_component_summary_update which is responsible for caching the tenant and component statistics #603
- trackme-limited/trackme-report-issues#604 - bug - Missing searchbnf providing usage syntax for the custom command trackmesplkpriority #604
- trackme-limited/trackme-report-issues#605 - bug - Priority Policies apply in TrackMe UI - incorrect variable leads to slient failure while applying policies for other components than splk-dsm #605
- trackme-limited/trackme-report-issues#608 - bug - Logical Groups - Unexpected non list structured in object_group_members / object_group_members_green / object_group_members_red can lead to Python exceptions and to the related entities not be available in the UI or from trackmegetcoll #608
- trackme-limited/trackme-report-issues#609 - bug - Data Hosts tracking (splk-dhm) - At high scale collection (more than 10k hosts), the current pagination count per count leads to incomplete rendering of entities #609

Enhancement, changes and new features:

- trackme-limited/trackme-report-issues#597 - enhancement - Adaptive Threshold tracker for Data Sources / Data Host tracking - The recent activty instrospection should take into account a change of the allow_adaptive field in case it has changed after the entity entered the cycle of adaptive review #597
- trackme-limited/trackme-report-issues#598 - feature request - Implement a per entity SLA timer and threshold concept, this would be used in a 2 tiers alerting system when a specifc alert would be sent when the SLA of entity is breached after having spent too long in a red state #598
- trackme-limited/trackme-report-issues#606 - change - Virtual Tenants UI - entities summary while double clicking on a given tenant should specify “enabled entities” rather than simply “entities” to avoid any confusion #606
- trackme-limited/trackme-report-issues#610 - change - Adaptive Treshold tracker - At the creation phase, the Adaptive Treshold tracker should be executed every 20 minutes to avoid risks of generating skipping searches at high scale #610
- trackme-limited/trackme-report-issues#611 - enhancement - Improving TrackMe logic to avoid generating skipping searches in various TrackMe scheduled logics #611
- trackme-limited/trackme-report-issues#612 - feature - TrackMe Alerting Architecture - Allows creating TrackMe Notables from TrackMe UI, Add builtin documentations and design good practices #612
- trackme-limited/trackme-report-issues#613 - change - REST API - bulk edit endpoints update to verify if json_data is submitted as a string, and if so loads it as a dict #613
- trackme-limited/trackme-report-issues#614 - enhancement - Persistent fields - centralization of per component persistent fields in collection_dict.py for more consistent and safer code #614
- trackme-limited/trackme-report-issues#615 - feature - Flex Object Library - Add a new use case to track the daily volume of data ingested per day and per index, and leverage Machine Learning for the Outlers detection #615
- trackme-limited/trackme-report-issues#616 - feature - Bulk Edit performance - Massive improvement in bulk edit performance in TrackMe, bulkd edit now runs in a fraction of seconds no matter the volume of the collection #616

11.1.100 Version 2.0.90 - build 1714432454 (30/04/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: a0d0360ae77c807bc991fa960580675860a4e75828e6b55f08bf5cd70d97e1b2

Fixed issues:

- trackme-limited/trackme-report-issues#584 - bug - New ctime field is not persistent in some components (dsm, wlk) #584
- trackme-limited/trackme-report-issues#593 - bug - Data Hosts tracking / Metric Hosts tracking - error message trackmeextractsplkmhm/trackmeextractsplkdhm when the command is executed in no metric generation mode #593
- trackme-limited/trackme-report-issues#595 - bug - Data Source / Data host tracking (splk-dsm/splk-dhm) Persistence of fields issue when the Adaptive tracker runs due to some Python level issues with batch update related code in the specific circumstances of sending a partial update #595

Enhanccements, changes and new features:

- trackme-limited/trackme-report-issues#585 - feature - priority management - provide a component wide feature for priority dynamic managements using regex based policies #585
- trackme-limited/trackme-report-issues#587 - enhancement - Virtual Tenants - Load Tenants high level statistics available when double clicking on the tenant flex box from cachedstats for consistency and better performance at high scale #587
- trackme-limited/trackme-report-issues#588 - enhancement - Virtual Tenants UI - Add a configuration choice for the trackmeload mode (REST versus legacy search driven) to address some limited compatibility issues reported by FEDRAMP Classic Splunk Cloud #588
- trackme-limited/trackme-report-issues#589 - feature - Machine Learning engine - Add capabilities to define static static_lower_threshold / static_upper_threshold per model #589
- trackme-limited/trackme-report-issues#590 - change - Data Hosts tracking (splk-dhm) - presets tstats root span to 1m by default #590
- trackme-limited/trackme-report-issues#591 - feature - Virtual Tenants creation UI - Allow in the first steps to define tenants level settings (ML Outliers features and other main Tenants level optons) #591
- trackme-limited/trackme-report-issues#592 - feature - Virtual Tenants - Allows to control the enablement of TrackMe Machine Learning Outliers Anomaly detection at the level of the Virtual Tenant #592
- trackme-limited/trackme-report-issues#596 - enhancement - Machine Learning - Avoids the error “The ML search is not yet available for rendering” when the ML model is not yet ready for rendering #596

11.1.101 Version 2.0.89 - build 1713898383 (23/04/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: d7a561e975b0ddfa4c1ee423c7e7eef7f59f339c6d1ee415112ca89cb1a2ec47

Fixed issues:

- trackme-limited/trackme-report-issues#562 - bug - REST API - Maintenance mode disable endpoint should return a native JSON response rather than a JSON dumped response #562
- trackme-limited/trackme-report-issues#563 - bug - REST API - fix various documentation errors in TrackMe’s REST API endpoints #563

- trackme-limited/trackme-report-issues#566 - bug - Machine Learning - perc_min_lowerbound_deviation in repeated twice in dsm Outliers table management, min_value_for_lowerbound_breached/min_value_for_upperbound_breached are missing from dhm tables #566
- trackme-limited/trackme-report-issues#569 - bug - DecisionMaker - Prevents against various possibilities of Python exceptions in the TrackMe Decision Maker libraries and calls which can lead to Error processing record #569
- trackme-limited/trackme-report-issues#570 - bug - Logical Groups - Ensure to limit match=1 for logical grouping enrichment at search time before reaching the DecisionMaker #570
- trackme-limited/trackme-report-issues#571 - bug - Backup and Restore - Builtin TrackMe KVstore backup fails when there are disabled tenants #571
- trackme-limited/trackme-report-issues#576 - bug - CIM (splk-cim) - SLA metrics are not generated if the trackme_metric index has been customised #576
- trackme-limited/trackme-report-issues#579 - bug - Machine Learning - ML Model addition UI in some components would not render a result when simulating the addition of the model as the command should call the lightsimulation mode rather than the simulation mode since TrackMe 2.0.88 #579
- trackme-limited/trackme-report-issues#580 - bug - Machine Learning - custom command trackmesplkoutlierssetrules generates errors when dealing with Flex Object trackers with no Outliers definition #580

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#557 - feature - Flex Object library use cases - Add new UCs for detecting abnormal drop in Splunk feeds events count using Flex #557
- trackme-limited/trackme-report-issues#558 - feature - Machine Learning Outliers - Allows up to 1 year in the time range selection for the Outliers calculation by step of 30 days #558
- trackme-limited/trackme-report-issues#559 - feature - Machine Learning Outliers - Add max in calculation methods available #559
- trackme-limited/trackme-report-issues#560 - feature - Machine Learning Outliers - Flex Object - Support all settings to be defined per Flex Object tracker rule, update built in documentation #560
- trackme-limited/trackme-report-issues#564 - enhancements - REST API - When deleting entities, permanently or temporary, the API should also clean up records for Outliers and Sampling, if any. #564
- trackme-limited/trackme-report-issues#565 - feature - New immutable KVstore field called ctime in TrackMe main KVstore component collections to keep track of entities origin creation time #565
- trackme-limited/trackme-report-issues#567 - enhancement - Virtual Tenants UI - When defining custom indexes as default indexes, the new Virtual Tenant creation UI should preset indexes with corresponding default indexes #567
- trackme-limited/trackme-report-issues#568 - enhancement - Workload (splk-wlk) - SmartStatus searches code improvements, ensure to include host=* splunk_server=* in SmartStatus Workload searches, more consistent searches matching the trackers, code improvements #568
- trackme-limited/trackme-report-issues#572 - feature - Data Host tracking (splk-dhm) - Add the capability to exclude (blocklist) a list of indexes and/or sourcetypes per host #572
- trackme-limited/trackme-report-issues#573 - feature - Machine Learning Outliers - Allow pre-defining at the system level extra parameters for the MLTK fit command, which can also be defined on a per model basis #573
- trackme-limited/trackme-report-issues#575 - enhancement - User Interface Home - ensure the main entity modification screens use scroll bar if the screen resolution is too low #575
- trackme-limited/trackme-report-issues#577 - feature - Machine Learning Outliers - allow using a custom MLTK algorithm #577
- trackme-limited/trackme-report-issues#581 - enhancement - Add an additional numerical verification in the Python function trackme_components_register_gen_metrics to prevents from any risks of generating malformed metrics leading to Splunk notification #581

11.1.102 Version 2.0.88 - build 1712331711 (05/04/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 70b5d340687c3e45d3702b1c4ce84e8cb6edb7a866fba75915c4de3cdafff8db

Fixed issues:

- trackme-limited/trackme-report-issues#550 - feature - Home interface drilldown & notable drilldown link - Allows submitting an object or alias URL param which filters out and opens automatically the entity overview, also add a drilldown_link to TrackMe Notables #550
- trackme-limited/trackme-report-issues#552 - bug - Virtual Tenant UI - count discrepency in summarized stats due to the monitoring enablement not being taken into account #552
- trackme-limited/trackme-report-issues#553 - bug - Python shared functions - get_kv_collection function used in some backends can lead to the generation of error messages with document ID conflict #553
- trackme-limited/trackme-report-issues#553 - bug - Python shared functions - get_kv_collection function used in some backends can lead to the generation of error messages with document ID conflict #553
- trackme-limited/trackme-report-issues#554 - bug - Data Source tracking - trackmesplktags does not implement batch_save leading to potentially increased run time #554
- trackme-limited/trackme-report-issues#555 - bug - TrackMe UI - Entities filtering functions do not properly take into account the show Enabled True/False dropdown #555
- trackme-limited/trackme-report-issues#556 - bug - Flex Object UC - SOAR Services monitoring - non reachable SOAR shoudl lead to services being red immediately #556

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#550 - feature - Home interface drilldown & notable drilldown link - Allows submitting an object or alias URL param which filters out and opens automatically the entity overview, also add a drilldown_link to TrackMe Notables #550

11.1.103 Version 2.0.87 - build 1711995624 (01/04/2024)

- SHA256: 40e3bd2e52eed4c5e27e62b6e6386d13264284f65ac91a8cf61ebc6db8e9914b

High performance for high scale collections in TrackMe with pagination, server side filtering, KVstore batch_find & Tabulator theming

- This release introduces massive performance improvements in TrackMe, allowing notably high scale collections to be managed with ease.
- `REST API Pagination` - With TrackMe REST pagination capabilities and Tabulator capabilities, TrackMe can handle any number of entities in a collection without any performance degradation, allowing to deal with large collections of more than 100K entities.
- `Server side REST filtering` - TrackMe and the Tabulator now perform server side REST level filtering, this slightly optimises response time while filtering for entities with simple or complex filters even when working with very large collections.
- `Server side stats caching` - TrackMe now caches tenants and components statistics at the server level, allowing it to retrieve the stats in a fraction of the time it used to take.
- `Python native implementation for the Decision Maker and filter handling` - From this release, TrackMe handles entirely the Decision Maker phases and filtering handling in Python, without involving any Splunk searches, allowing to largely optimise the performance of these operations.

- `Background Python threading` - TrackMe also uses background side Python threading methods to maintain cached statistics, allowing to largely optimise performance run time of these operations and slightly reducing the usage of search slots in TrackMe.
- `KVstore batch_find and batch_update implementation` - This release also implements KVstore batch_find and batch_update for all user side interactions, allowing all entities update actions such as bulk edits or per entity/feature edit (priority update, etc) to take a fraction of the time it used to take in previous releases, no matters the number of entities in the collection.
- `Massive UI side performance improvements` - All these changes are reflected in TrackMe's UI by major reduction of load time, major reduction of the response time during entity updates, and globally slightly enhanced response times in TrackMe.
- `Tabulator theming` - This release also introduces new capabilities to update at the system and user level the look and feel of the Tabulator, allowing users to choose between 5 different themes, at the system and user level. (Dark Site, Dark, Light Site, Light, Light Modern)

Fixed issues:

- trackme-limited/trackme-report-issues#525 - bug - Data Hosts / Metric Hosts tracking (splk-dhm/splk-mhm) - Allow list KV transforms definitions are lacking the is_rex field, this will be corrected automatically with TrackMe's schema upgrade #525
- trackme-limited/trackme-report-issues#539 - bug - Data Source tracking (splk-dsm) - Allow Adaptive Delay field persistence is not honoured by hybrid trackers #539

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#505 - feature - Filter for acknowledgement comment content in main dashboard #505
- trackme-limited/trackme-report-issues#520 - feature - Implement systematic pagination mechanisms at the TrackMe's REST API level for high scale collections performance, implement server REST side filtering for high performance #520
- trackme-limited/trackme-report-issues#522 - change - Tabulator JS - Upgrade to version 6.1 #522
- trackme-limited/trackme-report-issues#523 - enhancement - Docs references feature for splk-dsm - Allows robust system wide default parameters, decomission related knowledge objects #523
- trackme-limited/trackme-report-issues#524 - feature - REST API TrackMe - Support for params GET based endpoints #524
- trackme-limited/trackme-report-issues#526 - enhancement - Blocklists for Feeds tracking (splk-dsm/dhm/mhm) - Allows the alias in addition with the object to choosen as the field to apply the blocklists against, code improvements #526
- trackme-limited/trackme-report-issues#534 - change - Decomission of the DataGen concepts replaced with more meaningful blocklist concepts for Feeds tracking #534
- trackme-limited/trackme-report-issues#535 - change - Splunk Python SDK 2.0.0 - deprecation explicit lib is required #535
- trackme-limited/trackme-report-issues#536 - enhancement - Dependencies verification - Add the Splunk Scientific package in dependencies verifications #536
- trackme-limited/trackme-report-issues#540 - enhancement - Data Sources tracking (splk-dsm) - Manual tags refreshed UI, new management endpoints and enhanced workflow #540
- trackme-limited/trackme-report-issues#541 - enhancement - REST API endpoints performance optimization - Implement KVstore batch_find and optimize all actions for much faster performances in REST API calls #541
- trackme-limited/trackme-report-issues#542 - enhancement - Tags policies tracker for Data Sources tracking (splk-dsm) - Immediately apply tags against the data collection in a batch_save manner for optimial performances and behaviour #542

- trackme-limited/trackme-report-issues#543 - feature - TrackMe's Vtenant UI and Home Tenants themes for Tabulator - Allow to define at the system and user level between 5 Tabulator theme (Dark Site, Dark, Light Site, Light, Light Modern) #543
- trackme-limited/trackme-report-issues#545 - change - Machine Learning models management - Ensures privately owned TrackMe ML models from the splunks-system-user are excluded from the Knowledge Bundle replication #545
- trackme-limited/trackme-report-issues#546 - change - Python and Splunk SDK 2.0.x - remove outdated or non necessary imports #546
- trackme-limited/trackme-report-issues#547 - change - trackmetenantstatus custom command - log in warning rather than error when there is not yet activity registered for a newly created tenant #547
- trackme-limited/trackme-report-issues#548 - enhancement - Maintenance mode & Maintenance Knowledge Database - Better handle user local time and show the local time information properly #548

11.1.104 Version 2.0.86 - build 1710525022 (15/03/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 30cc9a93c821b1d772b55ff8ed89aa4ba40de9394409b4792d9f8890c7d9d512

Fixed issues:

- trackme-limited/trackme-report-issues#529 - bug - Data Hosts tracking (splk-dhm) - Bulk edit for Ack enablement does not honour Ack expiration and type dropdowns (only affects this component) #529
- trackme-limited/trackme-report-issues#530 - bug - Data Sources tracking (splk-dsm) - Tags policies update through the UI breaks the policies structure #530
- trackme-limited/trackme-report-issues#531 - bug - Python function for central searching in Splunk - preview must be set to false or results may appear to be duplicated #531
- trackme-limited/trackme-report-issues#532 - bug - TrackMe performance counters for Trackers report inaccurate measures (trackmetrackerexecutor) #532

11.1.105 Version 2.0.85 - build 1710194416 (11/03/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: f79ca52d8eed0b4d8db4fad80373c4ea079aeae1ddb8cfd1bbf61cb1b5de0744

Fixed issues:

- trackme-limited/trackme-report-issues#527 - bug - splunkremovesearch - The local account should not be accounted against the license restriction (legacy Free Community edition, now Foundation Edition trial, 1 remote account should be granted) #527
- trackme-limited/trackme-report-issues#528 - bug - Data Sources tracking (splk-dsm) - TrackMe REST API will not accept global_dcount_host as the min_dcount_field value #528
- trackme-limited/trackme-report-issues#521 - bug - Trackers and Licensing - If the user calls a tracker with "_tracker" part of its name, other reports (abstract, wrapper) are wrongly accounted against the license #521

11.1.106 Version 2.0.84 - build 1709505402 (03/03/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Schema upgrade for TrackMe version 2.0.84

- This release includes a TrackMe schema upgrade which will automatically clean Outliers orphan records and orphan ML models.
- The schema upgrade is executed within the next 5 minutes after the upgrade, through the tenant's health tracker jobs.
- If there is a large amount of orphan models to be cleaned up, this can temporarily increase generate skipping searches for the health tracker as its execution would eventually take much longer than usual.
- After this, the health tracker will resume its normal execution and skipping searches for it will disappear.
- This process is fully automated, and there are no intervention required.

Schema upgrade issues for jump from old releases of TrackMe v2.0.x

- Different issues where addressed in this release to properly support migrating from very old versions of TrackMe v2.0.x. to this release.
- You can therefore safety migrate from any earlier version of TrackMe v2.0.x without expected issues.

- SHA256: 425ce2d470ec072f17289eedccbb94ce87115c5a063e92af4998a39ff4ed27da

Fixed issues:

- trackme-limited/trackme-report-issues#474 - bug - Workload (splk-wlk) - diff_search and other related deleted modification fields are not preserved in the KVstore record in other iterations of the metadata job (but preserved as indexed events, however). #474
- trackme-limited/trackme-report-issues#476 - bug - Alert action - The label is incorrect on the type of Ack for the TrackMe auto Ack action. #476
- trackme-limited/trackme-report-issues#482 - bug - Flex Library - The lastchanceindex object name should not include the current prefix. #482
- trackme-limited/trackme-report-issues#483 - bug - Flex Library - Cribl Logstream destination pressure UC should take into account yellow state metrics (value: 1) as well as green/red metrics. #483
- trackme-limited/trackme-report-issues#485 - bug - Hybrid Trackers - Creation via REST API endpoints should mirror UI default False options for break by host/splunk_server. #485
- trackme-limited/trackme-report-issues#486 - bug - Virtual Tenant UI - Overview duplicates entities in red state. #486
- trackme-limited/trackme-report-issues#489 - bug - Machine Learning models update screen - Depending on the component, the list of metrics is incorrect or incomplete, for Flex Objects, a free text update capability is required. #489
- trackme-limited/trackme-report-issues#492 - bug - Adaptive Thresholds for Data Sources (splk-dsm) - Error in the formula for review over time logic when defining the average of the 3 KPIs over 30d/7d/24h. #492
- trackme-limited/trackme-report-issues#494 - bug - Adaptive threshold (splk-dsm/splk-dhm) - The Adaptive threshold does not parse the pipe-delimited nature of anomaly_reason properly, thus it ignores entities affected by delay breached in addition to any other anomaly. #494

- trackme-limited/trackme-report-issues#497 - bug - Tenants Knowledge Objects permissions issue with Schema Upgrade - Read and Write permissions were inverted in the Schema upgrade in recent versions using standardized libs to manipulate KOs, this leads to created objects during the schema upgrade to eventually define inconsistent permissions. This update fixes it and also automatically fixes any existing tenant. #497
- trackme-limited/trackme-report-issues#500 - bug - Reject/remove special or unprintable characters when automatically adding newly discovered sources to TrackMe. #500
- trackme-limited/trackme-report-issues#501 - bug - Workload (splk-wlm) - Discrepancy and remaining issues when searches contain non-unicode or foreign characters. #501
- trackme-limited/trackme-report-issues#502 - bug - Data Host tracking (splk-dhm) - In some conditions, all sourcetypes red should be overridden by global host level thresholds (host shows red, should show green). #502
- trackme-limited/trackme-report-issues#504 - bug - Add quotes for object token in the dashboard "Adaptive delay threshold audit." #504
- trackme-limited/trackme-report-issues#508 - bug - Data Sources (splk-dsm) - Permanent entity deletion via the dedicated button through the modification screen performs a temporary deletion instead (but bulk permanent deletion works as expected). #508
- trackme-limited/trackme-report-issues#511 - bug - Virtual Tenants creation can fail during the upgrade process from an old enough version of TrackMe V2. #511
- trackme-limited/trackme-report-issues#518 - bug - REST API documentation - A few REST API endpoints incorrectly set the root uri (admin/write) for the resource_spl_example value #518

Enhancements, changes, and new features:

- trackme-limited/trackme-report-issues#472 - enhancement - Virtual Tenants - Major performance improvements in the loading time of the UI by avoiding a slot search to get TrackMe tenants in pure Python. #472
- trackme-limited/trackme-report-issues#475 - enhancement - Python backend search framework - A consistent and centralized approach to programmatic Pythonic searching in Splunk. #475
- trackme-limited/trackme-report-issues#477 - enhancement - Flex Library - Performance runtime improvements for the use case splk_license_usage_per_index. #477
- trackme-limited/trackme-report-issues#478 - bug - Flex Library - Wrong outlier metric name in OOTB use case cribl_logstream_pipeline. #478
- trackme-limited/trackme-report-issues#480 - enhancement - Flex Library - Queues filling use case set max_inactive_sec to 0, which is now allowed by splk-flx. #480
- trackme-limited/trackme-report-issues#481 - change - Alert naming default - Remove "custom on" from the alert default name in the input alert name. #481
- trackme-limited/trackme-report-issues#488 - feature request - Data Source tracking (splk-dsm) - Generate and ingest a global dcount host metrics that is not driven by the ingest and is closer to a simple dcount host. #488
- trackme-limited/trackme-report-issues#491 - feature - Flex Objects (splk-flx) - New use cases for Splunk Search Head Clusters (SHC) infrastructure monitoring. #491
- trackme-limited/trackme-report-issues#493 - feature request - Filter option for acknowledged entities. #493
- trackme-limited/trackme-report-issues#495 - enhancement - Adaptive Threshold for Feeds tracking (splk-dsm/splk-dhm) - Use max_auto_delay_sec in case the calculated threshold is higher than max_auto_delay_sec. #495
- trackme-limited/trackme-report-issues#496 - enhancement - PersistentFields command (KVstore batch update process) - For splk-dsm/splk-dhm, reject a KVstore record update request if the current KVstore value for data_last_time_seen is bigger than the upstream value from the tracker run. #496
- trackme-limited/trackme-report-issues#498 - feature - Data Sources tracking (splk-dsm) - Tags management - Major improvements to the tags policies for splk-dsm: Allow multi-match tags policies, new dedicated Python backend replacing the previous SPL native logic, enhanced UI elements for tags, enhancements tags policies management UI. #498

- trackme-limited/trackme-report-issues#499 - feature - Flex Objects / Workload (splk-flx/splk-wlk) - Allows more flexibility for charting type and mode selection in Flex Objects and Workload. #499
- trackme-limited/trackme-report-issues#506 - Feature - Entities in blue state show as alert in dashboard. #506
- trackme-limited/trackme-report-issues#509 - change - Virtual Tenants wizard - Disable splk-dhm/splk-dhm components by default unless requested. #509
- trackme-limited/trackme-report-issues#512 - feature - Outliers engine - New automated training feature, this allows automatically performing an ML model train operation when the backend attempts to render an out-of-date ML model to avoid false positives. #512
- trackme-limited/trackme-report-issues#514 - Bulk Acknowledgement unified for all components (Allows bulk Ack with expiration selection similarly to splk-dsm). #514
- trackme-limited/trackme-report-issues#515 - change - Tags for Data Sources (splk-dsm) - Include tags as part of minimal events indexed with trackme:state events by default #515
- trackme-limited/trackme-report-issues#516 - feature - Bulk actions - Provide various bulk actions capabilities for Outliers management (reset Outliers status, enable/disable Outliers detection, run mltrain / mlmonitor) #516
- trackme-limited/trackme-report-issues#517 - change - Logging - Outliers error message "The ML search is not yet available for rendering" should be rendered as warning rather than errors #517

11.1.107 Version 2.0.83 - build 1706721363 (31/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 7348149f074e719719bce7cad50c1861ee1c46646b03b1bd1294726c07924e92

Fixed issues:

- trackme-limited/trackme-report-issues#451 - bug - Hybrid Trackers / Flex Object trackers - latest_time is not used during tracker creation #451
- trackme-limited/trackme-report-issues#456 - bug - Logical Group - object is red even though logical group has sufficient green members #456
- trackme-limited/trackme-report-issues#459 - bug - Decision Maker - If both out of monitoring days and monitoring hours are True, a dplicated message is generated in status_message and status_message_json #459
- trackme-limited/trackme-report-issues#461 - bug - User Interface - In some conditions, the status message screen may not allow access to the footer management buttons due to the timeline component #461
- trackme-limited/trackme-report-issues#462 - bug - Data Hosts tracking (splk-dhm) - Outliers status should be looked up before the Decision Maker is called for the anomaly_reason and status_message to be reflected in the KVstore (which however has no impact on the detection) #462
- trackme-limited/trackme-report-issues#465 - bug - Data Hosts tracking (splk-dhm) - outliers_readiness is not preserved while running DHM trackers, leading the ML screen to display ML not ready message altrhough ML is actually ready #465

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#457 - feature - Virtual Tenant - Introducing a tenant alias concept, this allow assigning an alias per tenant which can be updated via the Configure UI, this value is now used in the Virtual Tenant UI rather than the tenant_id which is immutable #457
- trackme-limited/trackme-report-issues#458 - feature - Logical Groups - Extend Logical Groups to Flex Object (splk-flx) #458

- trackme-limited/trackme-report-issues#460 - enhancement - Logical Groups - Major rewrite of the backend management for Logical Groups which is now full taken in charge by the Decision Maker, we also automatically detect and purge orphans logical group members (via the health tracker), major improvements and immediate change reflection via the Decision Maker #460
- trackme-limited/trackme-report-issues#463 - enhancement - SmartStatus - Extend SmartStatus to Flex Object, various improvements to the SmartStatus backend for automatic search retry, improved search management and search use cases for all components, more consistent approach with normalized ML UC #463
- trackme-limited/trackme-report-issues#464 - enhancement - Virtual Tenants UI - show/hide spinner while loading tenant's knowledge objects until API call is over #464
- trackme-limited/trackme-report-issues#466 - change - Virtual Tenants UI - Disable by default the splk-mhm when creating a new feeds tenant, unless instructed otherwise in the wizard #466
- trackme-limited/trackme-report-issues#467 - enhancements - Flex Objects (splk-flx) - Improving inline documentation and added max_sec_inactive as well as time_factor in ML models generation #467
- trackme-limited/trackme-report-issues#468 - enhancement - Flex Object library (splk-flx) - Improving the Splunk DMA builtin use case #468
- trackme-limited/trackme-report-issues#469 - enhancement - Flex Object (splk-flx) - Allowing a max_sec_inactive = 0 to disable automated red trigger based on detected inactivity #469
- trackme-limited/trackme-report-issues#470 - feature - Logical Groups - Add new management screen allowing to add / update / delete Logical Groups with easier access and management #470
- trackme-limited/trackme-report-issues#471 - feature - Health Tracker - Implement a new context called inspect_collection which ensures that object statuses in KVstore collections are always consistent with the Decision Maker, this also addresses some specific use case where there could be an inconsistent object_state in the KVstore collection #471

11.1.108 Version 2.0.82 - build 1705991568 (23/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: c7b039911bf8f9506096b5b1b03f98edf9a53d6ea0d4b7f22edfc68e80b66935

*Fixed issues:**

- trackme-limited/trackme-report-issues#452 - bug - Adaptive delay audit dashboard - remaining typo and dead link in the navigation menu #452
- trackme-limited/trackme-report-issues#453 - bug - Maintenance mode & Maintenance Knowledge DataBase - Prevents failure to load the Knowledge DataBase UI when the maintenance mode was enabled through a REST call #453
- trackme-limited/trackme-report-issues#454 - bug - Maintenance mode & Maintenance Knowledge DataBase - Retro-compatbility for older version of Firefox due to issues with the datetime-local input selector #454

Enhancement, changes and new features:

- trackme-limited/trackme-report-issues#455 - enhancement - Splunk Remote Search - Improve logging and error handling when testing / configuration / using Splunk Remote Search in TrackMe #455

11.1.109 Version 2.0.81 - build 1705906378 (22/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 3c33e18c7fb3920523eebaf795dfb02c9f80220292353ed6fc99f8d44c5b452d

Fixed issues:

- trackme-limited/trackme-report-issues#447 - bug - Typo in the new adjustements dashboard for Adaptive audit #447

Enhancement, changes and new features:

- trackme-limited/trackme-report-issues#448 - enhancement - Adaptive delay adjustment audit dashboard user experience improvements #448
- trackme-limited/trackme-report-issues#449 - enhancement - Acknowledgment management REST API endpoints - code and behaviour enhancements, allows listing all Ack, better management and new API endpoint for the UI purposes #449
- trackme-limited/trackme-report-issues#450 - enhancement - UI Acknowledgement - Enhanced Ack management screen relying on direct REST integration for faster and richer user experience #450

11.1.110 Version 2.0.80 - build 1705650542 (19/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 06bba3369ad7fa358e026bcbec3bc7e604b20cc47e648308b63ad1944d9fc0b3

Fixed issues:

- trackme-limited/trackme-report-issues#446 - change - Splunk Base failure to properly initiate Appinspect vetting request #446

11.1.111 Version 2.0.79 - build 1705620290 (18/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: bf29cb3fe4d65e1decbb958dfe2b32c625a3950ed58d2e38fadb4dc3bb9b2cd5

Fixed issues:

- trackme-limited/trackme-report-issues#439 - bug - Logging system - missing log_level search time extraction for alert actions logs #439
- trackme-limited/trackme-report-issues#440 - bug - Bulk edit Acknowledgment - The Ack period selected is interpreted in seconds instead of days when doing Ack through Bulk editing #440
- trackme-limited/trackme-report-issues#441 - bug - Acknowledgement backend logging - Avoid improperly generating the message “no object state information could be retrieved” #441
- trackme-limited/trackme-report-issues#442 - bug/enhancement - Decision Maker for Data Hosts tracking (splk-dhm) - logic adjustementfor entity level thresholds management #442

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#437 - Feature request - Allow to define if automated acknowledgements should be sticky or unsticky within TrackMe’s builtin alert action #437

- trackme-limited/trackme-report-issues#438 - enhancement - Flex Object Library - Last Chance Index use case improvements #438
- trackme-limited/trackme-report-issues#443 - feature request - Data Source monitoring (splk-dsm) - Overview chart series selection improvements to allow more choices and alertnatively hide the delay and/or latency series #443
- trackme-limited/trackme-report-issues#444 - feature - Adaptive Threshold - Adding a new Audit dashboard focusing on reviewing the adjustments made by TrackMe #444
- trackme-limited/trackme-report-issues#445 - enhancement - Logging backend - Retrieve report and macros details and log them before attempting to delete knowledge objects when requested to do so #445

11.1.112 Version 2.0.78 - build 1705310134 (14/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 71ae1311bc9fc6bd87b01f60da8b80c727094766d9c92fc9f0b8a8769eac7bd6

Fixed issues:

- trackme-limited/trackme-report-issues#429 - bug - Adaptive Delay backend - prevent UnboundLocalError errors when mstats returned no results in some conditions #429
- trackme-limited/trackme-report-issues#430 - bug - trackmepersistentfields (TrackMe persistent fields) - prevent exception message="could not convert string to float: " if tracker_runtime is unexpectly empty #430
- trackme-limited/trackme-report-issues#431 - bug - Cribl Logstream Flex Object use cases for inputs and outputs health check should take into account green/yellow/red returns from Cribl #431
- trackme-limited/trackme-report-issues#432 - bug - Data Hosts/Metric Hosts (splk-dsm/splk-mhm) - Avoid error "gen_metrics" failed with exception 'NoneType' object has no attribute 'get' #432
- trackme-limited/trackme-report-issues#435 - bug - Adaptive Delay (Data Sources / Data Hosts tracking - splk-dsm/splk-dhm) - TrackMe does not honour properly allow_adaptive_delay #435
- trackme-limited/trackme-report-issues#436 - enhancement - Adaptive Delay (splk-dsm/splk-dhm) - Improved logic and logging for the management of ML based adaptive delay tresholding #436

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#433 - enhancement - Flex Object Library - Splunk Queues filling use case review and improvements #433
- trackme-limited/trackme-report-issues#434 - feature - Flex Object Library - New use case for Splunk Search Heads key activity tracking #434

11.1.113 Version 2.0.77 - build 1704838956 (09/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: fe1d3723cd2a091781a71992b13255884275170ee8d23b5b22f9b2ca6e375706

Fixed issues:

- trackme-limited/trackme-report-issues#425 - bug - Workload / Flex Objects - muliselect dropdown should automatically refresh when the time range is changed #425

- trackme-limited/trackme-report-issues#428 - bug - Decision Maker - regression with custom wdays / hours ranges parameters not properly taken into account #428

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#392 - enhancement - Future data detection - Take into account a negative latency as a likely data in the future use case and turn entity orange as expected when future detection is operated against _time #392
- trackme-limited/trackme-report-issues#423 - enhancement - Status message improvements with a new native JSON structure and enhanced viz mode #423
- trackme-limited/trackme-report-issues#424 - enhancement - CIM compliance - extend week days & hours ranges concepts to CIM compliance tracking #424
- trackme-limited/trackme-report-issues#426 - enhancement - Cribl Logstream - Flex Object library use cases improvements, enhanced syntax and improved logic, better use ML Outliers rather than basic thresholds for some of the use cases, globally improved use cases #426
- trackme-limited/trackme-report-issues#427 - enhancement - Flex Object library - review use case splk_splunk_cloud_svc_usage_by_app and base threshold on ML Outliers #427

11.1.114 Version 2.0.76 - build 1704492296 (05/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: d6c01dc0e902605422c7375cc920172e01595d3f44689fb5f8cc8e03d0dc117f

Fixed issues:

- trackme-limited/trackme-report-issues#422 - bug - Decision Maker - regression when red on outliers or red on sampling is turned off on the tenant but an an actual outliers or sampling alert is active #422

11.1.115 Version 2.0.75 - build 1704475839 (05/01/2024)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 5436020d80f4cb2c7cc55ead436e3c3d4ccd0102fe6797fa87233f9903de573f

Fixed issues:

- trackme-limited/trackme-report-issues#416 - bug - Timezone offset management - properly handle time information management honoring users & system timezone offsets #416
- trackme-limited/trackme-report-issues#420 - bug - trackmesplkoutlierstrain - this command should not call directly the component register when raising an exception (leading to unexpected error logging) #420

Enhancement, changes and new features:

- trackme-limited/trackme-report-issues#410 - enhancement - Workload (splk-wlk) - Improved and safer scheduler and introspection tracking logic to avoid missing execution traces and false positive execution delayed alerts #410
- trackme-limited/trackme-report-issues#411 - enhancement - Outliers Adaptive Thresholding (splk-dsm/splk-dhm) - adjustments of the logic for enhanced behaviour #411
- trackme-limited/trackme-report-issues#398 - Feature Request: Acknowledgement overlay in Tabulator tables (right click context popover) #398

- trackme-limited/trackme-report-issues#414 - feature - Add row click popover context for Outliers and Data Sampling #414
- trackme-limited/trackme-report-issues#415 - feature - Introducing TrackMe decision maker backend, this new concepts replaces SPL based complex evaluations to define the status of TrackMe entities depending on the context and components, for a safer and more robust decision making #415
- trackme-limited/trackme-report-issues#417 - feature - Allows enabling/disabling at the tenant level the adaptive delay threshold feature (via a Virtual Tenant account switch) #417
- trackme-limited/trackme-report-issues#418 - enhancement - Flex Object - Complete popover context menu (Outliers status, status message and anomaly_reason) #418
- trackme-limited/trackme-report-issues#419 - change - Data Sources tracking (splk-dsm) - Do not include the remote account information in the definition of the alias #419
- trackme-limited/trackme-report-issues#421 - enhancement - Workload (splk-wlk) - Improved logic for detection and purge of any duplicated entities in Workload #421

11.1.116 Version 2.0.74 - build 1703259037 (22/12/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: d2a7e5c5447741cc166256589174e31eb01d9e658fcedd54f24264f9c5f92f15

Fixed issues:

- trackme-limited/trackme-report-issues¢12 - bug - Workload - Regression issue with outliers definition when performing the schema migration, leading to invalid eval and interrupting the Workload detection - #412

11.1.117 Version 2.0.73 - build 1703095950 (20/12/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: ddb21e231b5ba7d4f0fc31306ce538a9466b1801e3f3dfe67fcdccba633663f2

Fixed issues:

- trackme-limited/trackme-report-issues#408 - bug - Virtual Tenants UI - regression on the listing of reports in TrackMe Tenants Operational health statuses #408
- trackme-limited/trackme-report-issues#409 - bug - Virtual Tenants UI - Tenants Operational health statuses can show empty last_exec under some conditions #409

11.1.118 Version 2.0.72 - build 1703080417 (20/12/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 99c071e498886209e5a231f9443f96741e5ed7026fbb9aa1ded365774c6a174a

Fixed issues:

- trackme-limited/trackme-report-issues#379 - bug - Data Source tracking (splk-dsm) - regression in the simulate thresholds screen due to the migration to restricted summary state events in TrackMe 2.0.68 #379

- trackme-limited/trackme-report-issues#380 - bug - Configuration UI - title wording is not consistent for thresholds default configuration management #380
- trackme-limited/trackme-report-issues#381 - bug - Workload (splk-wlk) - Outliers are set with lower breached enabled unexpectly with elapsed KPI, shema version upgrade will address this issue automatically #381
- trackme-limited/trackme-report-issues#387 - fix - Avoid permissions issues for the Health tracker shema upgrade when handling TrackMe's knowledge upgrade #387
- trackme-limited/trackme-report-issues#394 - bug - Workload/Flex (splk-wlk/splk-flx) - Metric dropdown populating search use static -24h earliest time range #394
- trackme-limited/trackme-report-issues#395 - bug - Outliers - Permissions issues for Power users in different advanced Outliers related actions such as resetting or force training models #395
- trackme-limited/trackme-report-issues#399 - bug - Flipping status detection - Non unicode chars can lead to continuous discovery #399
- trackme-limited/trackme-report-issues#401 - bug - Elastic processing backend - error message local variable 'count_processed' referenced before assignment when no entities to be processed #401
- trackme-limited/trackme-report-issues#403 - bug - User Interface - Auto-refresh should be disabled automatically when performing bulk edition & inline edition #403

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#372 - change - Allow assigning an Ack to a blue state entity #372
- trackme-limited/trackme-report-issues#373 - enhancement - Least privileges & permissions - Some ingest related activities (health tracker & Notables) require the edit_tcp capability, which can be avoided by controled TrackMe capabilities #373
- trackme-limited/trackme-report-issues#374 - feature - Manage permanently deleted entities through a builtin UI screen from components #374
- trackme-limited/trackme-report-issues#376 - enhancement - Flex Objects - Add a group filter option in the Tabulator #376
- trackme-limited/trackme-report-issues#382 - enhancement - Workload (splk-wlk) - Take into account status delegated_remote_error as parts of scheduler excution failures, existing trackers will be updated automatically by the schema upgrade #382
- trackme-limited/trackme-report-issues#383 - change - Workload (splk-wlk) - Increase the SmartStatus earliest time from -24h to -7d for the execution error search #383
- trackme-limited/trackme-report-issues#384 - feature - Adaptive delay - Introducing the Adaptive delay feature to allow managing automatically delay threshold value for Data Sources and Hosts tracking (splk-dsm/splk-dhm) #384
- trackme-limited/trackme-report-issues#385 - enhancement - Outliers - Add more context information in the isOutlierReason field when an Outlier is triggered #385
- trackme-limited/trackme-report-issues#386 - feature - Machine Learning Outliers - Introducing the confidence concept to reduce false positive and identify low confidence models and entities #386
- trackme-limited/trackme-report-issues#388 - feature request - Overview Table: Column for human readable thresholds #388
- trackme-limited/trackme-report-issues#389 - change - User Interfaces - Increase 90% width modal screens to 96% of the screen as a basis for enhanced user experience #389
- trackme-limited/trackme-report-issues#390 - enhancement - Flex Objects / Workload / CIM compliance (splk-flx/splk-wlk/splk-cim) - Include the Outliers column in the Tabulator view #390
- trackme-limited/trackme-report-issues#391 - feature - Maintenance Knowledge DataBase - Intoducing a concept of a maintenance knowledge database, which can be used in association with the maintenance mode or independently to influence the SLA calculations by injecting knowledge of planned or unplanned operations that have lead to an impact on TrackMe entities #391

- trackme-limited/trackme-report-issues#393 - feature - Add Ack duration and Ack type as customizable options for bulk edit actions #393
- trackme-limited/trackme-report-issues#396 - feature - Introducing a new command "trackmesplkoutliersget-data" to get easier access to Outliers results #396
- trackme-limited/trackme-report-issues#397 - change - Virtual Tenants - code improvements for the managment of boolean options when creating tenants #397
- trackme-limited/trackme-report-issues#400 - feature - Outliers - Allowing to set the time_factor to none which enables TrackMe to apply a simpler LowerBound/UpperBound with no seasonability variations #400
- trackme-limited/trackme-report-issues#402 - change - Workload (splk-wlk) - define the Outliers by default based on time factor with no seasonability for elapsed based metrics for enhanced results #402
- trackme-limited/trackme-report-issues#404 - feature - Workload (splk-wlk) - Automatically process a diff of the 3 main search Metadata (search, earliest, latest) and attempt to identify the user who performed the change and the time of the change when detecting a saved search version change #404
- trackme-limited/trackme-report-issues#406 - enhancement - Virtual Tenant - Health Status reporting - enhanced Tabulator view #406
- trackme-limited/trackme-report-issues#407 - change - Tenants & knowledge objects creation ownership - switch the default owner from admin to nobody #407

11.1.119 Version 2.0.71 - build 1700472127 (20/11/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 174868c183c78036487881576355a9bc7de228e6295811caf5ac8d3428af8fc8

Fixed issues:

- trackme-limited/trackme-report-issues#364 - bug - Typo in distinct count #364
- trackme-limited/trackme-report-issues#370 - bug - Replica tenants - Do not attempt to perform the replica tracker for a disabled tenant #370

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#365 - enhancement - Workload (splk-wlk) - Handle use cases when Splunk incorrectly logs scheduler activity with no user context, introducing a new dynamic get owner retrieval component, scheduler trackers are updated automation during the schema upgrade #365
- trackme-limited/trackme-report-issues#366 - enhancement - Review of timeout policies in TrackMe, ensures all service definition and Python request define a timeout #366
- trackme-limited/trackme-report-issues#367 - enhancement - Flex Objects library - Improvement of the splk_kvstore_size use case for Flex #367
- trackme-limited/trackme-report-issues#368 - enhancement - Feeds tracking - Improving the status message for latency & delay alerts (including durations, incude both thresholds, round to 3 decimals) #368
- trackme-limited/trackme-report-issues#369 - feature - Data Sources tracking (splk-dsm) - Allow choosing between any of the dcount metrics to define minimal distinct count host thresholds rather than the default mandatory choice (latest_dcount_host_5m) #369

11.1.120 Version 2.0.70 - build 1700087843 (15/11/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 4690b0653623f7a96b9347a493a24806c3120bf098fd95fcd2a75d939b369f24

Fixed issues:

- trackme-limited/trackme-report-issues#362 - bug - healthtracker - errors generating the expected audit events in trackme_audit for the health tracker itself due to a regression #362
- trackme-limited/trackme-report-issues#363 - bug - last_exec is reported as null in the component register audit events #363

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#360 - enhancement - When missing the right permissions and capabilities, show a clearly understandable message for admins to take actions #360
- trackme-limited/trackme-report-issues#361 - feature - Workload component (splk-wlk) - Introducing the over-group feature, allowing to override the per application grouping and allowing to colocate multiple Search tiers in the same tenant #361

11.1.121 Version 2.0.69 - build 1699886135 (13/11/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: dd9ca1df32eb23008db8d128f7dee9665562224e93aa2fe5e384af64ffc3808e

Fixed issues:

- trackme-limited/trackme-report-issues#352 - bug - Shared Elastic - minor logging errors #352

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#353 - enhancement - Elastic Dedicated - improve the manage screen rendering #353
- trackme-limited/trackme-report-issues#354 - feature - Migrate component register tracker run time to TrackMe's metric store for faster queries, and better retention than from the _internal only #354
- trackme-limited/trackme-report-issues#355 - feature - Bootstrap icons / Emoji ascii compatibility mode - provide a configurable option for both Vtenants UI / Home UI to switch between Emoji ascii based statuses icons and Bootstrap based icons, this addresses compatibility issues for some customers on Wndows not supporting Emoji ascii fonts #355
- trackme-limited/trackme-report-issues#356 - enhancement - Flex Objects library - Enhancement search for the DMA use case #356
- trackme-limited/trackme-report-issues#357 - feature - Flex Object library - New use case for Splunk large lookup files detection #357
- trackme-limited/trackme-report-issues#359 - change - Increase minimal time betweem two ML training per entity from 24 hours to 7 days for TrackMe footprint reduction #359

11.1.122 Version 2.0.68 - build 1699407909 (08/11/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 939013c951a1884369efeb934f12cad919c1d41a22411de8fd1c09a1f3a25ee7

Note

SLA to metrics migration

- This new release introduces the migration for SLA metrics to metrics based indexes instead of the previous SLA calculations based on the state events
- This allows slightly reducing the size and volume of state events, reducing storage and licensing costs for TrackMe, as well as performing much faster queries and allowing much longer retentions
- If you wish to backfill the existing SLA knowledge after you have migrated to TrackMe 2.0.68, run the following Splunk search to backfill SLA metrics using `mcollect`
- We made the choice not to automate the SLA migration such that you can decide to do it or not, and control its execution process

*Use this search after the migration to TrackMe 2.0.68 to backfill SLA metrics (this search can takes a while, think about modifying indexes if necessary, reduce the timerange if you do not care about all metrics, and send this to the background for the best control of its excution)**

```
index=trackme_summary sourcetype="trackme:state" object_category=* object=* key=*␣
↪tenant_id=* current_state=* earliest=-90d
| fields _time, tenant_id, object_category, object, alias, current_state,␣
↪monitored_state, priority, key
| bucket _time span=1m
| stats latest(current_state) as object_state, latest(alias) as alias,␣
↪latest(monitored_state) as monitored_state, latest(priority) as priority by _
↪time, tenant_id, object_category, object, key

``` convert string status to numerical ```
| eval object_state=case(
    object_state = "green", 1,
    object_state = "red", 2,
    object_state = "orange", 3,
    object_state = "blue", 4,
    1=1, 5
    )

``` rename to the metric_name target, key is objct_id in the new metrics schema ```
| rename object_state as trackme.sla.object_state, key as object_id

``` use mcollect to backfill metrics ```
| mcollect index=trackme_metrics split=t tenant_id, object_category, object,␣
↪object_id, alias, monitored_state, priority
```

Note

Introducing the TrackMe stats events minimal mode

- This new release introduces a major reduction of the TrackMe state events (sourcetype=trackme:state) in terms of volume and size, as well as a consistent schema
```

- This change was made possible in association with the SLA to metrics migration
- You can control in the Configuration screen the mode of generation, minimal (default) or full (as prior to 2.0.68), as well as the list of fields to allow (minimal mode) or block (full mode)
- These options are available in the *General Configuration* tab (Minimal state events, allowlist fields (minimal), In full, block list fields)
- There are no actions required to benefit from this change, unless you had some custom reporting or alerting based on the state events, in which case you should review your use cases and adapt them to the new schema

Fixed issues:

- trackme-limited/trackme-report-issues#339 - bug - Virtual Tenant UI regression on dynamic theme system level preferences application (flex cards should turn red properly) #339
- trackme-limited/trackme-report-issues#342 - bug - Health Tracker (inactive entities tracking) - handle if tracker_runtime is null #342
- trackme-limited/trackme-report-issues#343 - bug - Health Tracker (inactive entities tracking) - offline abstract macros should not exclude permanently deleted entities #343
- trackme-limited/trackme-report-issues#344 - bug - command trackmepersistentfields - logic assignement error in persistent fields definition #344
- trackme-limited/trackme-report-issues#346 - bug - Elastic Sources - Addressing various issues in this release (eventcount not parsed with from lookups, results duplicated in simulation, code weakness) #346
- trackme-limited/trackme-report-issues#348 - bug - Virtual Tenants - Issues with underscores in tenant identifiers when created through the REST API #348
- trackme-limited/trackme-report-issues#350 - bug - Virtual Tenant - Enabling a previously tenant that has splk-dhm/wlk will report a failure on enabling some macros #350
- trackme-limited/trackme-report-issues#351 - bug - Data Sources tracking (splk-dsm) - regression on honoring not including the host in the tstats root break by fields #351

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#337 - change - Tabulator update to version 5.5.2 #337
- trackme-limited/trackme-report-issues#338 - feature - Flex Objects - Introducing the Splunk practices use cases for the Flex Objects component #338
- trackme-limited/trackme-report-issues#340 - feature / enhancements - Introducing major improvements for the Elastic Sources Shared backend with parallel muti-processing, automated job max runtime definition, ordering of execution and improved logging #340
- trackme-limited/trackme-report-issues#341 - feature/enhancement - SLA metrics - For enhanced performances and better management, SLA calculations are moving to true metrics #341
- trackme-limited/trackme-report-issues#345 - enhancement - Logging - standardize run_time logging to 3 decimals for all TrackMe backends #345
- trackme-limited/trackme-report-issues#349 - feature - State events minimal mode - Major reduction in the state events volume and size to reduce the impact on storage and license (migrates splk-dhm/mhm to full metrics, introducing the state event minimal configuration to ingest minimal state events) #349

11.1.123 Version 2.0.67 - build 1698669312 (30/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 4c6c90fcad4bf91dbdc17c434d19e4c00de5f18dab7860c18d4f72b9c059fb66

Fixed issues:

- trackme-limited/trackme-report-issues#329 - bug - Persistentfields - Python exception if the mtime or tracker_runtime is not in the expected format #329
- trackme-limited/trackme-report-issues#330 - bug - Workload (splk-wlk) - Non ASCII characters in knowledge objects names such as foreign accents are not properly handled #330
- trackme-limited/trackme-report-issues#331 - bug - Maintenance mode - Failure when attempting to enable the maintenance mode #331
- trackme-limited/trackme-report-issues#332 - bug - Missing arguments in searchbnf.conf for the Data Sampling tracker executor #332
- trackme-limited/trackme-report-issues#333 - bug - Flex Objects / CIM compliance - missing filehandler rotation in Python lib leads to the log file not being rotated #333
- trackme-limited/trackme-report-issues#336 - bug - Flex Objects - properly handle some problematic escaped rex sequences when running remote searches #336
- trackme-limited/trackme-report-issues#304 - bug - Virtual Tenant UI - Dropdown text search is not working (affects initial creation and RBAC update modal screens) #304

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#334 - feature - Adding the new command trackmesplkoutliersexpand to expand ML outliers results for further processing #334
- trackme-limited/trackme-report-issues#335 - feature - Adding a new expending streaming command for Flex Objects (trackmesplkflxexpandextra), its purpose is to expand the extra_attributes for new use cases management in the Flex Object library #335

11.1.124 Version 2.0.66 - build 1698184235 (24/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 2593a02f2a8f2a475a6e0318bddd48d94b31fc014a8441cfef10c1168dc495f6

Fixed issues:

- trackme-limited/trackme-report-issues#328 - bug - Data Sources tracking (splk-dsm) - The overview single average latency and percentile 95 incorrectly show the same metric (regression from 2.0.65) #328

11.1.125 Version 2.0.65 - build 1698103284 (24/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: e14d7b9e4e198cf79680c2ea6dd598ab3b2b58450077127bc3dbba4f4bedd728

Fixed issues:

- trackme-limited/trackme-report-issues#324 - bug - Data Hosts tracking (splk-dhm) - regression on alias value definition at discovery #324
- trackme-limited/trackme-report-issues#325 - bug - Ack - wrong audit message #325
- trackme-limited/trackme-report-issues#326 - bug - Flex Objects library - error in default cron schedule for lastchanceindex use case #326
- trackme-limited/trackme-report-issues#327 - bug - Data Sources tracking (splk-dsm) - If adding host in the custom break by field, the hybrid tracker incorrectly defines entities #327

11.1.126 Version 2.0.64 - build 1698044829 (23/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 2a0981f700bf2d3c759bb37839578e35876dd5aa7947b17aaf0f15b30d3b816e

Fixed issues:

- trackme-limited/trackme-report-issues#317 - bug - Data Sources/Data Hosts tracking (splk-dsm/splk-dhm) - fix discrepency between banner delay and single form delay as well as the Tabulator delay (ensures last delay is refreshed against now) #317
- trackme-limited/trackme-report-issues#318 - bug - Data Hosts tracking (splk-dhm) - Issue in the offline abstract macro called by the health tracker (execution fails due to missing pipe when called) #318
- trackme-limited/trackme-report-issues#320 - bug - Data Hosts tracking (splk-dhm) - Alias is not correctly persisted when the entity goes out of the trackers range #320

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#319 - change - Data Sources/Hosts tracking (splk-dsm/splk-dhm) - decomission the delayed entities tracker which features are now better handled by the health tracker #319
- trackme-limited/trackme-report-issues#321 - enhancement - Data Sources/Hosts tracking (splk-dsm/splk-dhm) - maintain the generation of the delay metric (lag_event_sec) when entities are out of the range of trackers #321
- trackme-limited/trackme-report-issues#322 - enhancement - Data Sources / Data Hosts tracking (splk-dsm/splk-dhm) - Extend the auto-lagging screen to include both ingest latency and delay concepts #322
- trackme-limited/trackme-report-issues#323 - enhancement - Data Sources/Hosts tracking - show the delay metric (lag_event_sec) in the overview timechart #323

11.1.127 Version 2.0.63 - build 1697650503 (18/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 1c506fe8b6535228631f8e5c72a817bb00e0a6fac7da886912e30c9932fb2ce6

Fixed issues:

- trackme-limited/trackme-report-issues#310 - bug - ML Outliers - Avoid generating an error message when attemping to load the period of exclusion if not a list (add safety) #310
- trackme-limited/trackme-report-issues#313 - bug - Workload (splk-wml) - TrackMe should not attempt to perform replacement for app stanza criterias any more if target is remote as these are now explicit in the creation process #313

- trackme-limited/trackme-report-issues#314 - bug - Ingest - Since the migration to INGEST_EVAL in 2.0.60, some expected key indexed fields in trackme:state and others are not indexed any longer #314
- trackme-limited/trackme-report-issues#315 - bug - SmartStatus - ingested alert actions are lacking the tenant_id and object_category fields, breaking the indexed key consistency scheme in TrackMe #315
- trackme-limited/trackme-report-issues#316 - bug - Fix splunkd WARN message "with request data but no Content-Type: header; not parsing POST arguments" #316

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#311 - feature - Allow defining the default sharing level (app or global) when TrackMe creates or manages Splunk Knowledge Objects #311
- trackme-limited/trackme-report-issues#312 - change - INGEST_EVAL - Add a safety fail back condition for ingest evals defining the index target #312

11.1.128 Version 2.0.62 - build 1697551318 (17/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: b2f8e6fb03716ce1d9950ca39be0d40c6ded740e7a64035fcf34ef2a3cc9ea24

Fixed issues:

- trackme-limited/trackme-report-issues#303 - TrackMe bug report - Hybrid Tracker cron no applied in the report schedule #303
- trackme-limited/trackme-report-issues#307 - bug - ML Outliers - Auto Correct should not allow lowerBound and upperBound to be equals #307

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#306 - change - Dark theme compatibility - Enable dark theme compatibility in app.conf #306
- trackme-limited/trackme-report-issues#305 - change - ML Outliers - Disable by default the generation of the latency based model for Feeds which is not a great candidate in most of the use cases #305
- trackme-limited/trackme-report-issues#308 - enhancement - ML Outliers - inherit earliest and latest from the time range picker rather than explicitely for the ML rendering commands #308
- trackme-limited/trackme-report-issues#309 - feature - ML Outliers - Capability to add or delete a period of time for exclusions in the ML models training #309

11.1.129 Version 2.0.61 - build 1697150459 (12/10/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Note

Inheritance support for RBAC

- This release introduces support for roles inheritance for RBAC in TrackMe
- Virtual Tenants are Splunk Remote Accounts can be accessed, managed and administrated by inheriting roles according to your configuration

- SHA256: ad69875eba15dd7680add23d5fba72131916ea04ec862d04df3479fd9e56bf21

Fixed issues:

- trackme-limited/trackme-report-issues#294 - bug - Workload / Flex Objects - When more than a single Outliers model is in anomaly, the status_message comes back null as the macro did not expect the multivalue nature of these fields #294
- trackme-limited/trackme-report-issues#300 - bug - SLIM Packing for Splunk Cloud Classic - spec files are not instructing the partitioning properly #300
- trackme-limited/trackme-report-issues#301 - bug - Data Sources tracking (splk-dsm) - UI token manipulation related issues leads to a null search eating the user disk quota under some circumstances #301

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#290 - enhancement - Flex Objects (splk-flx) - improvement of the use case splk_splunk_enterprise_cluster_peers_status (calculate buckets inbalance deviation and alert) #290
- trackme-limited/trackme-report-issues#291 - enhancement - Flex Objects (splk-flx) - improvement of the use case splk_splunk_enterprise_cluster_status #291
- trackme-limited/trackme-report-issues#292 - enhancement - Flex Objects (splk-flx) - New use case for rolling tracking of license usage per index and pool #292
- trackme-limited/trackme-report-issues#293 - bug/enhancement - Machine Learning Outliers detection - Auto correct logic defects leads to avoid generating true positive outliers #293
- trackme-limited/trackme-report-issues#295 - enhancement - Flex Object - Cribl integration UC improvements for health inputs and outputs to remove false positive #295
- trackme-limited/trackme-report-issues#296 - enhancement - Flex Objects use cases library - UC splk_queues_filling improvement - avoid generating alerts when the queues are inactive
- trackme-limited/trackme-report-issues#297 - change - Remove owner=admin as the default in default.meta to avoid Enterprise customers with no admin users to be impacted by the default behavior of TrackMe #297
- trackme-limited/trackme-report-issues#298 - enhancement - Roles Based Access Control (RBAC) - Support inheritance globally in TrackMe #298

11.1.130 Version 2.0.60 - build 1695681952 (25/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 859bd778ac65750a5e4eb05cc3c11a884ddbdedd9fffcb1e33fafd54909dd71b

Fixed issues:

- trackme-limited/trackme-report-issues#289 - bug - SLIM partitioning causes ingest issues in Splunk Cloud Classic experience, requires explicit stanza placement in spec files #289

11.1.131 Version 2.0.59 - build 1695559981 (24/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 67d7a8466af72c68705cfeeca6504589ad732bc01c0961f8597f1e1236059d44

Fixed issues:

- trackme-limited/trackme-report-issues#283 - bug - trackmetrackerhealth (Health Tracker) - Hybrid tracker macro update in the KVstore should only happen if the currently known definition differs from system #283
- trackme-limited/trackme-report-issues#284 - bug - TrackMe alert actions (notable, SmartStatus, Ack) - failures to run actions in the context of a strict least privilege service account owning the tenant #284

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#285 - change - Health Tracker - Improve logging for inactive entities tracking for splk-dsm/splk-dhm #285
- trackme-limited/trackme-report-issues#286 - change - entity_info API endpoints - always return the object and key value in the response to recycle values as needed and ease further processing #286
- trackme-limited/trackme-report-issues#287 - change - Reduce the timerange considered by the delayed entity trackers to 24h by default, after this time inactive entities are taken into account by the health tracker #287
- trackme-limited/trackme-report-issues#288 - enhancement - Data Sources and Hosts tracking (splk-dsm/splk-dhm) - Ensures that the delayed entities tracker updates last entity Metadata information even if the target search did not return any results #288
- trackme-limited/trackme-report-issues#261 - enhancement - Provide cURL examples for each REST API endpoints in the REST API auto-documentation #261

11.1.132 Version 2.0.58 - build 1694716015 (14/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 90119b248d9a1a820a335254a3d994ab4b45a7839f2468c7d087d3604208a91a

Fixed issues:

- trackme-limited/trackme-report-issues#281 - bug - splunkremotesearch - Non meaningful Python exception when calling a non existing account #281

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#282 - enhancement - Workload (splk-wlk) - Workload Virtual Tenant creation wizard improvements #282

11.1.133 Version 2.0.57 - build 1694635429 (13/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 5febf5ab3f93abf7ce8b0218f192374bdd5e3094d6150cc98f1e1a9b6126470a

Fixed issues:

- trackme-limited/trackme-report-issues#275 - bug - Data Hosts tracking (splk-dhm) - error when deleting entity on a per entity basis (list index out of range) #275
- trackme-limited/trackme-report-issues#277 - bug - Data Hosts tracking (splk-dhm) - error when trying to update monitoring hours of a given entity due to wrong REST API endpoint path #277

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#276 - feature - Introducing the CMDB integrator feature - Allows queriying an external third data source for contextual information in TrackMe tenants #276 - See: https://docs.trackme-solutions.com/admin_guide_cmdb_integration.html

- trackme-limited/trackme-report-issues#279 - change - RBAC - Optimisation for role membership verification #279
- trackme-limited/trackme-report-issues#280 - enhancement - Workload (splk-wlk) - Virtual Tenant creation wizard improvements, split the search filters to be specific in the UI for Scheduler / Introspection / Splunk Cloud SVC #280

11.1.134 Version 2.0.56 - build 1694411312 (11/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: d7d5ed282cda25375216de5e47eb770c6b8bc34d5d1c89354d7e123923374879

Fixed issues:

- trackme-limited/trackme-report-issues#264 - bug - typo in RBAC ownership view #264
- trackme-limited/trackme-report-issues#266 - bug - Workload (splk-wlk) - When creating the main tracker, the SVC usage should be part of the avg_svc_usage is trackmegenjsonmetricsmissing from the calls in #266
- trackme-limited/trackme-report-issues#268 - bug/change - INGEST_EVAL migration for all summary events and metric generation workflow, this migration is performed to overcome a Splunk Cloud Classic DMC deployment bug when deploying applications using transforms to override the DEST_KEY - While this issue is Splunk Cloud responsability, this is not going to be fixed in any acceptable timeline, TrackMe therefore turns to a different approach which is not affected by this #268
- trackme-limited/trackme-report-issues#271 - bug - Audit events - When using custom indexes per tenant, audit events remain generated in the default TrackMe configured index rather than the tenant specific index #271
- trackme-limited/trackme-report-issues#273 - bug - Benchmark Burn Test tends to time out for long run queries in Splunk Cloud due to time out reach in Splunk Cloud Web reverse proxy #273

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#265 - feature - TrackMe SVC usage audit dashboard for Splunk Cloud customers #265
- trackme-limited/trackme-report-issues#267 - change - Workload - Switch the default stats mode for the dropdown to max rather than latest to ensure visibility in most use cases #267
- trackme-limited/trackme-report-issues#269 - feature - Flex Object library (splk-flx) - New use case to track SVC consumption in Splunk Cloud by application #269
- trackme-limited/trackme-report-issues#270 - change - Flex Objects (splk-flx) - Licensing restriction increase to 32 trackers for Enterprise Edition customers #270
- trackme-limited/trackme-report-issues#272 - change - Ack behaviour default system wide configuration when returning to green - enables purging Ack by default when returning to non green if non sticky #272
- trackme-limited/trackme-report-issues#274 - enhancement - Feeds tracking (splk-feeds) - synchronize macros knowledge hybrid trackers attributes when the macros are updated in Splunk #274

11.1.135 Version 2.0.55 - build 1693924977 (05/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: b22a72485ba6b09d0b01bb0b19c4faf265aafd3e30a41f076fdc4eba75322b2d

Fixed issues:

- trackme-limited/trackme-report-issues#263 - bug - Virtual Tenants UI for Feeds tracking - indexes discovery feature does not work as expected due to Javascript regression when configured at the Virtual Creation phase #263

11.1.136 Version 2.0.54 - build 1693744485 (03/09/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 1a745c8ae615620d3c526e94742908897a0aa1e85dfa8454b1fb48d84a5b808e

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#42 - feature - Data Sources tracking (splk-dsm) - Tags for Data Source monitoring - Remove tags linked to a tag policy when the tag policy is removed #42
- trackme-limited/trackme-report-issues#259 - bug/enhancement - Virtual Tenants UI optimizations with a new unified endpoint for a faster and safer user experience, this also addresses issues observed in Splunk Cloud classic only #259
- trackme-limited/trackme-report-issues#260 - change - Update moment.js to version 2.29.4
- trackme-limited/trackme-report-issues#262 - enhancement - Virtual Tenants UI - Alphabetically sort tenants in the UI if no positions are preset for the user profile #262

Fixed issues:

- trackme-limited/trackme-report-issues#256 - bug - Data Hosts / Metrics Hosts (splk-dhm/splk-mhm) - Cannot filter on tags within the Tabulator #256
- trackme-limited/trackme-report-issues#257 - bug - Data Hosts tracking (splk-dhm) - Max global latency & delay per entity should match the highest relevant value between all sourcetypes related to it #257
- trackme-limited/trackme-report-issues#258 - bug - logging issues when checking permissions for trackmeload/trackmetenantstatus (not logging the right user name) #258

11.1.137 Version 2.0.53 - build 1692273340 (17/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 42654231000a4bae75d40d4d9317babd93b8cc5e080e8d2367ebc5d45365333f

Enhancement, changes and new features:

- trackme-limited/trackme-report-issues#251 - feature - Data Hosts / Metric hosts preset the alias equal to the raw object without the key(s) addition #251
- trackme-limited/trackme-report-issues#252 - feature - Flex Objects - New use cases for CPU and Memory infrastructure tracking via Splunk introspection #252
- trackme-limited/trackme-report-issues#253 - feature - Data Hosts and Metric Hosts tracking - enhancement for tags enrichment purposes #253

11.1.138 Version 2.0.52 - build 1692002557 (14/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 11dc12c922f8005257c1d8bc5eccf0e8d0f3b848b0881a6eabe42ea56944850f

FIxed issues:

- trackme-limited/trackme-report-issues#247 - bug - Replica tenants - logic issues when having more than a single replica tracker with the same component leading to the incorrect purge of replica records #247
- trackme-limited/trackme-report-issues#248 - bug - Replica tenants - The Flex object inactive entities tracker should not be created for Replica tenants #248

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#249 - feature - Allow pre-defining default owner and defaults admin/power/roles in TrackMe general configuration for the Virtual Tenants user interfaces #249

11.1.139 Version 2.0.51 - build 1691618697 (09/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 90b21e5cffa2ec91e968def2b857d083f46eb6c0fecfe5cc4f423d3d87168617

Fixed issues:

- trackme-limited/trackme-report-issues#245 - bug - All components - In large scale scenarios with more than 50k entities on a per tenant/component basis, the Tabulator is limited to 50k entities due to the underneath oneshot SDK search #245
- trackme-limited/trackme-report-issues#246 - bug - Data Sources/Data Hosts tracking (splk-dsm/splk-dhm) - In some rare conditions, a null search can be generated and run unexpectly impacting user quota #246

11.1.140 Version 2.0.50 - build 1691356328 (06/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 0147f78edb580e0a67229ee7eb42699e211d1b5791e844e6eb280d52fcf66043

Fixed issues:

- trackme-limited/trackme-report-issues#242 - bug - SOAR integration custom command trackmesplksoar - issues rendering a POST response rendered as a list #242
- trackme-limited/trackme-report-issues#243 - bug - SOAR integration - pagination issues in some circumstances restricts the number of entities returned #243

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#244 - feature - SOAR integration - Manage Automation Brokers High Availability with TrackMe, update SOAR Assets automatically when an Automation Broker is inactive to an active counter part - High Availability for SOAR Automation Brokers via TrackMe #244

11.1.141 Version 2.0.49 - build 1691080561 (03/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 6bd3ea567f0465a4f9e388c04cd95cb839b17594f3adb84619b01d00311de1b2

Fixed issues:

- trackme-limited/trackme-report-issues#236 - bug - SLA dashboard - Dropdowns populating search is using static 24 hours range rather than timerange picker from the dashboard #236
- trackme-limited/trackme-report-issues#240 - bug - Flex Objects (splk-flx) - UC Splunk Cloud SVC usage - ensure to generate metrics of SVC usage if the licensed SVCs is null #240

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#237 - enhancement - Flex Objects (splk-flx) - Allows the priority to be defined at the phase of the Flex Tracker execution #237
- trackme-limited/trackme-report-issues#238 - change - Workload (splk-wlk) - Increase the last_seen filter to last 90m for the metadata retrieval #238
- trackme-limited/trackme-report-issues#239 - enhancement - Flex Objects (splk-flx) - Include pool_quota_gb metrics in the license pool usage tracking #239
- trackme-limited/trackme-report-issues#241 - enhancement - Flex Objects (splk-flx) - Simplification and better code for the Deployment Server tracking use case #241

11.1.142 Version 2.0.48 - build 1690973605 (02/08/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: cc03ecd66725692e332ad6604ce5c3baddd4f3336883ffb65ff7aaad7ee67a42

Fixed Issues:

- trackme-limited/trackme-report-issues#219 - bug - Feeds Tracking (splk-dsm) - The delayed entities trackers re-generates non merged entities in a hybrid context of merged / non merged and does not track merged entities properly #219
- trackme-limited/trackme-report-issues#221 - bug - Virtual Tenants UI - Addresses some issues with theming and user preferences, more consistent management of preferences
- trackme-limited/trackme-report-issues#222 - bug - Workload (splk-wlk) - error in trackmesplkwlkgetreports-defstream for metadata retrieval when using remote target multiple load balanced search head targets #222
- trackme-limited/trackme-report-issues#224 - bug - Workload (splk-wlk) - simulation fails for Splunk Cloud SVC when running through the UI due to incorrect quote #224
- trackme-limited/trackme-report-issues#225 - bug - Workload (splk-wlk) - Back button not working from create hybrid trackers #225
- trackme-limited/trackme-report-issues#230 - bug - incorrect report names for the mltrain reports when adding to the report state register component #230
- trackme-limited/trackme-report-issues#231 - bug - Workload (splk-wlk) - Under some circumstances an entity generating execution errors could lead to incorrect definition of the user and looping with multivalue fields gnerating bad objects #231

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#223 - enhancement - Outliers engine - When requesting reset ML, the endpoint performs a search, if the max concurrency is searched on the Search Head this can lead to an unexpected failure, ensures we attempt automated retry if it is the case before failing permanently if necessary #223
- trackme-limited/trackme-report-issues#226 - feature - Flex Object (splk-flx) - new use case for tracking KV-store collections size #226
- trackme-limited/trackme-report-issues#227 -enhancement - Allows a service account owner to be using the minimal level of permissions and capabilites to own and run properly TrackMe objects #227
- trackme-limited/trackme-report-issues#228 - enhancement - Python code sanitization, auto-formatting and unit testings for automated bug identification #228
- trackme-limited/trackme-report-issues#229 - enhancement - Fix any hard coded reference to localhost for the communication with splunkd using best practice Python splunkd uri inherited URI #229
- trackme-limited/trackme-report-issues#232 - enhancement - Data Sources/Data Hosts tracking (spl-dsm/splk-dhm) - Health tracker maintains untracked entities which are out of the scope of any tracker to update and maintain state consistency #232
- trackme-limited/trackme-report-issues#233 - feature - Flex Object (splk-flx) - Use Case for Splunk Enterprise license pool usage tracking #233
- trackme-limited/trackme-report-issues#234 - enhancement - Splunk SOAR integration - Allows a least privilege approach for SOAR interactions #234
- trackme-limited/trackme-report-issues#235 - change - Feeds Tracking - delayed entities tracker switch to False for break by splunk_server and host which is the default now in TrackMe #235

11.1.143 Version 2.0.47 - build 1690295356 (25/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: bcdf0903d3fe531786764ff009911ade7a1a3ca779193733ea3771806d6ef0e3

fixed issues:

- trackme-limited/trackme-report-issues#220 - bug - regression in trackmeapiautodocs introduced in 2.0.46 when Splunk App for SOAR is not installed on the Search Tier #220

11.1.144 Version 2.0.46 - build 1690266086 (25/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: c62b857fc20638a97e3b17fd03e9cb5f6fb0d76c5027c8d95ba5cb661bc88fb0

fixed issues:

- trackme-limited/trackme-report-issues#210 - bug - Flex Objects (splk-flx) - When a given entity turns red due to inactivity, a summary state event should also be generated to properly influence the SLA percentage calculation #210
- trackme-limited/trackme-report-issues#213 - bug - Virtual Tenants - endpoint post_vtenants_accounts should not return an exception when there are no tenants yet #213
- trackme-limited/trackme-report-issues#215 - bug - Workload (splk-wlk) - status_message can come back null in some circumstances #215

- trackme-limited/trackme-report-issues#216 - bug - Virtual Tenants - deleting a component should clean up the vtenant summary record #216

Enhancements, changes & new features:

- trackme-limited/trackme-report-issues#211 - feature - Flex Objects - Splunk SOAR native integration (UCs for SOAR monitoring) #211
- trackme-limited/trackme-report-issues#214 - feature - Flex Object (splk-flx) - lastchanceindex use case for Splunk data_collection #214
- trackme-limited/trackme-report-issues#217 - change - Data Hosts tracking - automatically restrict the indexes to the main and internal indexes for splk-dhm if indexes is left unconfigured at the tenant creation phase with Hybrid tracker creation enabled (click next disease) #217

11.1.145 Version 2.0.45 - build 1689676533 (18/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 2b394e1617836c6e5757cac1ad9c2896d5d1340e008d23d403c47ba52c23f78d

Fixed issues:

- trackme-limited/trackme-report-issues#201 - bug - Flex UC splk_splunk_enterprise_cluster_status - wrong term Down rather than Stopped #201
- trackme-limited/trackme-report-issues#206 - bug - Flipping REST API issue (hitting Splunk CIM) #206
- trackme-limited/trackme-report-issues#207 - bug - CIM Tracking - regression in ML Outliers model generation #207
- trackme-limited/trackme-report-issues#208 - bug - CIM Tracking - deletion of entities in bulk fails since 2.0.40 #208
- trackme-limited/trackme-report-issues#209 - bug - CIM Tracking - failure to generate the initial discovered flipping event #209

Enhancements and new features:

- trackme-limited/trackme-report-issues#202 - feature - Flex Objects - Cribl Logstream use cases for deep monitoring of Cribl Logstream in TrackMe #202
- trackme-limited/trackme-report-issues#203 - enhancement - Flex Objects - allow multiselect metrics in entity overview #203
- trackme-limited/trackme-report-issues#204 - enhancement - Flex Object - preset the alias of the entity as the short value of the object (without the group) and allows defining custom values for the alias at the entity discovery phase of the tracker #204
- trackme-limited/trackme-report-issues#205 - enhancement - Flex Objects (splk-flx) - Manage inactive entities #205

11.1.146 Version 2.0.44 - build 1689362642 (14/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 7602e39ffcdfa299100fb33e0b25363a11ae25da6a5d3ec5051a8bad3bbb235c

Enhancement and new features:

- trackme-limited/trackme-report-issues#191 - feature - Flex Objects tracking - Introducing the Flex Objects use case library and major component features improvements #191

11.1.147 Version 2.0.43 - build 1689342033 (14/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Workload upgrade:

- review the release special instructions if you are using the workload component

- SHA256: 2af481f61b93eaa3c5811856e29871742c50ea176f59446ef39948cac5075cdf

Fix issues

- trackme-limited/trackme-report-issues#195 - bug - Workload (splk-wlk) - In some circumstances the Splunk scheduler logs can lack app and user context leading to the creation of new entities in case of execution errors detected #195
- trackme-limited/trackme-report-issues#198 - bug - Data Sources (splk-dsm) - enable/disable entities in bulk fails due to regression (object not defined) #198
- trackme-limited/trackme-report-issues#199 - bug - Outliers - regression due to the ds_account field decommissioning leading to failures in generating Outliers rules for new entities #199
- trackme-limited/trackme-report-issues#200 - bug - Remove the characters length restrictions in the Vtenant configuration in UCC #200

Enhancements and new features:

- trackme-limited/trackme-report-issues#197 - enhancement - All components - Execution of TracKers via the UI and when permited via RBAC should be executed as the system user to avoid user related context to impact results consistency #197

Special intructions or notes for this release:

- To benefit from the fix of issue #195 related to the Workload, the scheduler tracker should be deleted and re-created for each Workload tenant
- This can be achieved via the UI, or via REST API

11.1.148 Version 2.0.42 - build 1688984590 (10/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 7d4cf2359d629d9f56dd121ab03e981efe0fb1eb2bf98225f1cce6fcb7a882db

fixed issues:

- trackme-limited/trackme-report-issues#190 - bug - Workload - the main tracker does not include the count_ess_notable metrics in the metrics summary popup #190
- trackme-limited/trackme-report-issues#192 - bug - Data Sources (splk-dsm) - Clear state & run sampling resets the entity for DSM #192

- trackme-limited/trackme-report-issues#193 - bug - The number of currently existing trackers should show up in the management UI for Flex Objects and Workloads #193
- trackme-limited/trackme-report-issues#194 - bug - Data Hosts Tracking (splk-dhm) - summary level sourcetype state does not honour properly the latency/delay independently as expected #194

11.1.149 Version 2.0.41 - build 1688538958 (05/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: 9ee5384747ee3d022a3a3d8aaf0ae3794dffb9a501de0ce9e9c4a4002ac593a4

Fixed issues:

- trackme-limited/trackme-report-issues#189 - bug - splk-dsm (Data Source) bulk edit regression for enable/disable monitoring via bulk edit due to change #182 #189

11.1.150 Version 2.0.40 - build 1688457335 (04/07/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: a163d0b1b0892edecfd09784b39b6ae0ba13aad275b54355d86c92ccb1fa950e

Fixed issues:

- trackme-limited/trackme-report-issues#182 - bug - All components - handle entities changes via their unique identifier rather than the object (handles bad entities with unexpected special characters) #182
- trackme-limited/trackme-report-issues#183 - bug - Performance issues at large scale of entities for Flex / Workload trackers #183
- trackme-limited/trackme-report-issues#186 - bug - splunkremotesearch - splunk-system-user and admin users should be RBAC granted for all configured accounts #186
- trackme-limited/trackme-report-issues#187 - bug - Virtual Tenants UI - count=0 is missing from some rest searches, leading to avoid returning all results from the upstream search (ex: user account selection) #187

Enhancements, changes and new features:

- trackme-limited/trackme-report-issues#184 - change - Flex Object - allows automated width for the Status description in the Tabulator #184
- trackme-limited/trackme-report-issues#185 - feature - SmartStatus for Workload entities, allows the SmartStatus to handle Workload UCs as well as capturing Splunk internal events with a least privileges approach (no need for users to be able to access to the _internal index to review internal scheduler errors through the SmartStatus control) #185
- trackme-limited/trackme-report-issues#188 - enhancement - REST API logical groups - allows updating min percent if an existing group via REST without having to have to provide the list of current members #188

11.1.151 Version 2.0.39 - build 1687757627 (26/06/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA256: d855a2c6467e7a1d97abfb783a91883a2205b0b59102bef0471aa74aacf49303

Fixed issues:

- trackme-limited/trackme-report-issues#176 - bug - User Interface - Using DSM "Show disabled entities" filter clears the "Filter field or function" dropdown #176
- trackme-limited/trackme-report-issues#177 - bug - Data Hosts Tracking (splk-dhm) - truncation in trackme:state for entities with a very large amount of related sourcetypes #177

Enhancements and new features:

- trackme-limited/trackme-report-issues#178 - enhancement - Do not allow deleting or cloning Virtual tenants accounts in the Configuration UCC UI #178
- trackme-limited/trackme-report-issues#179 - enhancement - Check the Splunk Remote account connectivity and authentication at the creation / edit step in the Configuration UI (UCC framework) #179
- trackme-limited/trackme-report-issues#181 - change - Data sources/Data hosts (splk-dsm/spl-dhm) - sets break by splunk_server/host by default to False #181

11.1.152 Version 2.0.38 - build 1687154702 (19/06/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Metrics expansion mode and Workload upgrade:

- review the release special instructions for more information about the metrix expansion mode change in this release
- review the release special instructions if you are using the workload component

- SHA256: 90a6d51fc68b5e78b2b5a523d834fabbc2eea18cbcefb78e34f3f1ac793de04b

Fixed issues:

- trackme-limited/trackme-report-issues#151 - bug - Workload - the app filter provided as an example in the tracker search constraint can lead to the non detection of some use cases of execution errors #151
- trackme-limited/trackme-report-issues#152 - bug - failure to populate tenants dropdowns in SLA and Data Sampling Dashboard studio dashboards due to earlier changes in trackmeload output #152
- trackme-limited/trackme-report-issues#153 - bug - Workload - trackmesplkwlkgetreportsdefstream should call select url function to properly handle multiple Splunk endpoints for a remote account #153
- trackme-limited/trackme-report-issues#154 - bug - error in endpoint /splk_dsm/ds_get_dsm_sampling_obfuscation_mode due to obfuscation Virtual tenant account change #154
- trackme-limited/trackme-report-issues#155 - bug - Logical group auto group command - flow logic when adding single member groups #155
- trackme-limited/trackme-report-issues#158 - bug - Data Hosts (splk-dhm) - logic flow in trackme_dhm_tracker_abstract macro does not preserve per host max latency/delay and does therefore leads to no honouring these settings #158

- trackme-limited/trackme-report-issues#150 - bug - Elastic Sources - metrics generation fails for raw/from based Elastic Sources definition (shared and dedicated) #150
- trackme-limited/trackme-report-issues#159 - bug - Common Information Model tracking (splk-cim) - button horizontal alignment issue in TrackMe UI #159
- trackme-limited/trackme-report-issues#163 - bug - Vtenant UI - Prevents the running spinner to be removed (due to auto-refresh) before then end of the operation when executing long run operations such as tenants creation #163
- trackme-limited/trackme-report-issues#164 - enhancement - avoids running trackers during the Virtual Tenant creation phase to reduce time required for its creation (multiops endpoints) #164
- trackme-limited/trackme-report-issues#165 - bug - HTML duplicated ids, issues in label definition, various UI related issues #165
- trackme-limited/trackme-report-issues#166 - bug - Workload (splk-wlk) - indentation issues when creating Workload trackers, failures in the tracker creation UI to check remote connectivity #166
- trackme-limited/trackme-report-issues#167 - bug - Acknowledgments - typo when creating Ack manually leads to unstricky rather than unsticky status for Ack, prevent their proper expiration #167
- trackme-limited/trackme-report-issues#168 - bug - Workload (splk-wlk) - Orphan tracker enhancements from Issue#117 were lost during the transition to least privileges #168
- trackme-limited/trackme-report-issues#171 - bug - missing props definition for the command trackmeprettyjson #171

New features and enhancements:

- trackme-limited/trackme-report-issues#156 - enhancement - Logical Groups - round the percentage of current group status commitment, allows filtering on Blue entities for splk-dsm/dhm/mhm #156 enhancement - User Interface minimal mode and context popup approach to improve readibility for all eligible components #157
- trackme-limited/trackme-report-issues#160 - enhancement - Health Tracker - automatically detect when a TrackMe object no longer exists and cleanup the register knowledge #160
- trackme-limited/trackme-report-issues#161 - bug - mlmonitor reports are not registered with the right name in the component register #161
- trackme-limited/trackme-report-issues#162 - enhancement - Workload - Adding the notable type tracker to allow tracking the number of Enterprise Security notable events per correlation search #162
- trackme-limited/trackme-report-issues#169 - enhancement - Flex Objects (splk-flx) - The tracker wizard should allow trackers not returning any entities to be created, as lookling only bad conditions can be a use case #169
- trackme-limited/trackme-report-issues#170 - enhancement - splunkremotesearch - handle Splunk automated extractions when fields resuting from remote events are not consistents #170
- trackme-limited/trackme-report-issues#172 - enhancement - Workload (splk-wlk) - provides a deeper visibility with a 3 periods metrics approach of scheduled activity #172
- trackme-limited/trackme-report-issues#173 - enhancement - Tabulator component upgrade 5.5 #173
- trackme-limited/trackme-report-issues#174 - enhancement - Bulk edit - when clicking on all entities selector, ensures selected entities honour current filters including header filters and add the count number of entities to be impacted in the bulk edit screen #174
- trackme-limited/trackme-report-issues#175 - enhancements - Logs inspector dashboard - fixes and improvements for the log inspector dashboard #175

Special instructions for this release:

Default metrics expanded mode

- This new release introduces a change in the visibility of eligible components (splk-wlk/splk-cim/splk-flx/splk-dhm/splk-mhm) regarding the default expansion of the metrics column and/or JSON formatted context columns

- From 2.0.38, the column is not expanded any longer, a user would see a "right click for popup" message instead, right clicking will provide the expected information in a more context menu, providing better global readibility when dealing with many entities
- At anytime in the UI, one can switch to the expanded mode by selecting the "full" visibility in the mode selector dropdown in TrackMe
- Also, TrackMe administrators can update the default visbility mode when the tenant is loaded by editing the Vtenant preferences (Configuration / Virtual Tenant account) and defining the default mode for **UI prefs - expand metrics**

Workload (splk-wlk)

Workload notable tracking:

- If you are using Splunk Enterprise Security, you way want to track the notable activity which is a new type of Workload tracker added to this release
- The notable track will monitor the number of notable events generated per ES correlation search, and add a new metric "count_ess_notable" which can be used for context and investigations, or Outliers detection eventually.
- To add the new notable tracker, run the following command: (replace mytenant with the tenant name, define account according to your context)

```
| trackme mode=post url="/services/trackme/v2/splk_wlk/admin/wlk_tracker_create"␣
↪body="{'tenant_id': 'mytenant', 'account': 'local', 'tracker_type': 'notable'}"
```

- Also, you need to add the "count_ess_notable" metric in the main tracker, you can either edit manually the wrapper main report or follow the next instructions to re-create a brand new main tracker
- TrackMe schema version update will not perform this for you as you filter preferences (app filters for instance in the root constraints) would be lost and because this can run on a remote target, this cannot be added to a local macro for persistence)

Workload behaviour enhancements:

If you are using the Workload component, you may want to perform the following actions to benefit from some specific updates:

step 1: - Go in the tenant, click on "Manage: Workload Trackers" - Locate the main tracker, and click on Delete

step 2: - Go in a search, run the following command (replace mytenant by the tenant_id, the account is not relevant for main tracker and should always be local):

```
| trackme mode=post url="/services/trackme/v2/splk_wlk/admin/wlk_tracker_create"␣
↪body="{'tenant_id': 'mytenant', 'account': 'local', 'tracker_type': 'main'}"
```

step 3: - Search the following macro: "trackme_wlk_set_status_tenant_<tenant_id>" - Update its content to: (replace the occurences of <tenant_id> with the name of your tenant)

```
lookup local=t trackme_wlk_orphan_status_tenant_<tenant_id> object OUTPUT orphan,␣
↪mtime as orphan_last_check | eval␣
↪orphan_last_check=case(isnotnull(orphan_last_check), strftime(orphan_last_check,
↪"%c"))
| lookup local=t trackme_wlk_versioning_tenant_<tenant_id> object OUTPUT␣
↪cron_exec_sequence_sec
``` init a status 1```
| eval status=1
``` If there are execution errors detected, status=2, we use periods data from 60m␣
↪to 4h to 24h, the JSON metrics will not contain the metric if it equals to 0 ```
``` Therefore, if a given search generating errors if fixed and has frequent␣
↪executions, it likely will turn green in the next 60m from the deployment of the␣
↪fix ```
| eval status=case(
```

(continues on next page)
```

(continued from previous page)

```
count_errors_last_60m=0, status,
count_errors_last_4h=0, status,
count_errors_last_24h=0, status,
count_errors_last_60m>0 OR count_errors_last_4h>0 OR count_errors_last_24h>0, 2,
1=1, status
)
``` If there are skipping searches, define two levels of alerting, less than 5% is␣
↪3 (orange), more is 2 (red) ```
``` we base the calculation over the 24 period (suffix last_24h) - this can be␣
↪customised up to your preferences if you wish to used the additional periods ```
| eval status=case(
isnum(skipped_pct_last_24h) AND skipped_pct_last_24h>0 AND skipped_pct_last_24h<5,␣
↪3, isnum(skipped_pct_last_24h) AND skipped_pct_last_60m>0 AND␣
↪skipped_pct_last_24h>=5, 2,
1=1, status
)
``` If we detected the search as an orphan search (not period related) ```
| eval status=if(orphan=1, 2, status)
``` Calculate the delta in sequence between now and the last execution compared␣
↪against the requested cron schedule sequence, add 1h of grace time, detect if␣
↪the execution has been delayed ```
| eval status=if(cron_exec_sequence_sec>0 AND ( now()-last_seen >␣
↪(cron_exec_sequence_sec + 3600) ), 2, status)
``` Set a brief status description, a more granular description will be provided␣
↪with the anomaly_reason and status_message fields ```
| eval status_description=case(status=1, "normal", status=2, "degraded", status=3,
↪"warning", 1=1, "unknown")
```

11.1.153 Version 2.0.37 - build 1686088225 (06/06/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Roles Based Access Control enhancements:

- From version 2.0.34, TrackMe implements a new strict **least privilege Role Bbased Access Control**
- A new role **trackme_power** is now builtin in TrackMe and designed to allow performing updates to entities of a granted tenant
- Access to TrackMe is driven by builtin capabilities provided by TrackMe builtin roles (trackme_user, trackme_power, trackme_admin)
- The least privilege approach implemented since this release allows granular access to TrackMe without requiring problematic capabilities which have security implications (list_settings, list_storage_passwords)
- TrackMe user interfaces automatically adapt its content and provided options depending on the profile of the current user, a normal user will for instance not see write or admin related actions
- TrackMe REST API endpoints are now classified in 3 groups, user level endpoints, write level endpoints and admin level endpoints
- The TrackMe `splunkremotesearch` also supports Roles Based Access Control, a user calling a given account must be a member of any of the listed roles in the account configuration to be granted access to
```

this account

- For retro-compability purposes, TrackMe will allow access to an existing Remote account that has no RBAC roles setup yet to typical admin users in addition with TrackMe builtin roles (admin, sc_admin, trackme_user, trackme_power, trackme_admin)
- When TrackMe is **upgraded**, the migration of existing tenant is automatically performed by the **schema version management**, *Upgrading TrackMe*
- For more **information**, see: *Roles & access control*

- SHA256: 5a0b110099a769abea3af34cb61f4725c686d0554fcf89a1e63ce98486a7cc23
- trackme-limited/trackme-report-issues#147 - bug - splk-dsm (Data Source) - regression when call run sampling on a particular entity due to obfuscation change in v2.0.36 #147
- trackme-limited/trackme-report-issues#148 - bug - splk-dhm (Data Hosts) - the title of the modal screen incorrectly mentiones splk-mhm #148
- trackme-limited/trackme-report-issues#145 - enhancement: Higher width for the status column (which can truncated under Ack circumstances) #145
- trackme-limited/trackme-report-issues#149 - bug - Workload / Flex (splk-wlk/splk-flx) - Truncate long description to avoid impacting the view screen #149

11.1.154 Version 2.0.36 - build 1685947587 (05/06/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Roles Based Access Control enhancements:

- From version 2.0.34, TrackMe implements a new strict **least privilege Role Bbased Access Control**
- A new role **trackme_power** is now builtin in TrackMe and designed to allow performing updates to entities of a granted tenant
- Access to TrackMe is driven by builtin capabilities provided by TrackMe builtin roles (trackme_user, trackme_power, trackme_admin)
- The least privilege approach implemented since this release allows granular access to TrackMe without requiring problematic capabilities which have security implications (list_settings, list_storage_passwords)
- TrackMe user interfaces automatically adapt its content and provided options depending on the profile of the current user, a normal user will for instance not see write or admin related actions
- TrackMe REST API endpoints are now classified in 3 groups, user level endpoints, write level endpoints and admin level endpoints
- The TrackMe `splunkremotesearch` also supports Roles Based Access Control, a user calling a given account must be a member of any of the listed roles in the account configuration to be granted access to this account
- For retro-compability purposes, TrackMe will allow access to an existing Remote account that has no RBAC roles setup yet to typical admin users in addition with TrackMe builtin roles (admin, sc_admin, trackme_user, trackme_power, trackme_admin)
- When TrackMe is **upgraded**, the migration of existing tenant is automatically performed by the **schema version management**, *Upgrading TrackMe*

- For more **information**, see: *Roles & access control*

- SHA256: f0c47447023dca0daf9cb5e5e434dc077a0e8c71bfc75233d73717268eef33a3
- trackme-limited/trackme-report-issues#135 - bug - Data Sampling - Creating an mstats based Elastic Source breaks the Data Sampling query execution #135
- trackme-limited/trackme-report-issues#136 - bug - Outliers engine - When reseting Outliers models, TrackMe should also reset the data outliers records for a more consistent approach #136
- trackme-limited/trackme-report-issues#137 - bug - Acknowledgement - Updating Ack fails due to Python regression introduced in 2.0.34 #137
- trackme-limited/trackme-report-issues#138 - enhancement - Add a new command utility trackmeautogroup to allow auto management of logical group association from an upstream SPL logic #138
- trackme-limited/trackme-report-issues#139 - bug - SmartStatus - incorrect timechart search in UC delay causes no results to be found #139
- trackme-limited/trackme-report-issues#140 - enhancement - SmartStatus - rely on latest known event rather than latest - - trackme-limited/trackme-report-issues#141 - known ingest when defining the earliest for UC delay/latency for better results when looking at an offline entity #140
- trackme-limited/trackme-report-issues#141 - enhancement - vtenants accounts integration scheme for more flexible tenant level configuration management #141
- trackme-limited/trackme-report-issues#142 - enhancement - Improvements and minor fixes for user interfaces behaviours when user is a power user (capability: trackmepoweroperations) #142
- trackme-limited/trackme-report-issues#143 - bug - splk-dhm (Data Host Tracking) - TrackMe does not honor properly the per sourcetype policy due to evaluation of the state at the table loading time which avoids taking into account the status per sourcetype #143
- trackme-limited/trackme-report-issues#144 - feature - Introducing the TrackMe Configuration Manager (TCM) to provides CI/CD capabilities for TrackMe #144

Additional notes: - In version 2.0.36, the data sampling obfuscation macro is deprecated and decommissioned automatically, it is replaced by a much more flexible approach relying on the tenant account setting - To enable the obfuscation mode for a given tenant post-migration, go in Configuration / vtenant preferences and edit the tenant to enable the obfuscation mode

11.1.155 Version 2.0.35 - build 1684913150 (24/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Roles Based Access Control enhancements:

- From version 2.0.34, TrackMe implements a new strict **least privilege Role Bbased Access Control**
- A new role **trackme_power** is now builtin in TrackMe and designed to allow performing updates to entities of a granted tenant
- Access to TrackMe is driven by builtin capabilities provided by TrackMe builtin roles (trackme_user, trackme_power, trackme_admin)
- The least privilege approach implemented since this release allows granular access to TrackMe without requiring problematic capabilities which have security implications (list_settings, list_storage_passwords)

- TrackMe user interfaces automatically adapt its content and provided options depending on the profile of the current user, a normal user will for instance not see write or admin related actions
- TrackMe REST API endpoints are now classified in 3 groups, user level endpoints, write level endpoints and admin level endpoints
- The TrackMe `splunkremotesearch` also supports Roles Based Access Control, a user calling a given account must be a member of any of the listed roles in the account configuration to be granted access to this account
- For retro-compability purposes, TrackMe will allow access to an existing Remote account that has no RBAC roles setup yet to typical admin users in addition with TrackMe builtin roles (admin, sc_admin, trackme_user, trackme_power, trackme_admin)
- When TrackMe is **upgraded**, the migration of existing tenant is automatically performed by the **schema version management**, *Upgrading TrackMe*
- For more **information**, see: *Roles & access control*

- SHA-256: 0fbba6699287c2ac6fdcbeb28d4d6ccfa3d889b351b26f1e5010bd2ba74f8fef
- trackme-limited/trackme-report-issues#133 - bug - SmartStatus - regression introduced by version 2.0.34 causes SmartStatus function failure #133
- trackme-limited/trackme-report-issues#134 - bug - bad entities containing double quotes lead trackmesplk-outlierstrainhelper and trackmesamplingexecutor to continuously fail running searches for these entities with bad request #134

11.1.156 Version 2.0.34 - build 1684860645 (23/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Hint

Roles Based Access Control enhancements:

- In this release, TrackMe implements a new strict **least privilege Role Bbased Access Control**
- A new role **trackme_power** is now builtin in TrackMe and designed to allow performing updates to entities of a granted tenant
- Access to TrackMe is driven by builtin capabilities provided by TrackMe builtin roles (trackme_user, trackme_power, trackme_admin)
- The least privilege approach implemented since this release allows granular access to TrackMe without requiring problematic capabilities which have security implications (list_settings, list_storage_passwords)
- TrackMe user interfaces automatically adapt its content and provided options depending on the profile of the current user, a normal user will for instance not see write or admin related actions
- TrackMe REST API endpoints are now classified in 3 groups, user level endpoints, write level endpoints and admin level endpoints
- The TrackMe `splunkremotesearch` also supports Roles Based Access Control, a user calling a given account must be a member of any of the listed roles in the account configuration to be granted access to this account
- For retro-compability purposes, TrackMe will allow access to an existing Remote account that has no RBAC roles setup yet to typical admin users in addition with TrackMe builtin roles (admin, sc_admin, trackme_user, trackme_power, trackme_admin)

- When TrackMe is **upgraded**, the migration of existing tenant is automatically performed by the **schema version management**, *Upgrading TrackMe*
- For more **information**, see: *Roles & access control*

- SHA-256: ce0d5a73b314c8dc246737149962dc5bd2038f89b313429f13485e3e99e2cd35
- trackme-limited/trackme-report-issues#106 - enhancement - Least privilege implementation - TrackMe implementation of a least privileges approach to provide with minimal capabilities requirement and a best practice security implementation #106
- trackme-limited/trackme-report-issues#119 - enhancement - All components - Performance optimisations #119
- trackme-limited/trackme-report-issues#120 - bug - Compliance Tracking (splk-cim) - UI affected by a previous change (regression from #116) #120
- trackme-limited/trackme-report-issues#121 - enhancement - UI behaviours - Call spinner in a more consistent manner when actions are being performed #121
- trackme-limited/trackme-report-issues#122 - bug - Flex Object (splk-flx) - Convention for status in the docs explanation is wrong #122
- trackme-limited/trackme-report-issues#101 - enhancement - Data Source/Host (splk-dsm/dhm) - Allows managing data in the future detection on a per entity basis #101
- trackme-limited/trackme-report-issues#124 - enhancement - major performance improvements for trackmesplkoutlierssetrules #124
- trackme-limited/trackme-report-issues#125 - enhancement/bug - major performance improvements for Trackers execution (trackmepersistentfields) #125
- trackme-limited/trackme-report-issues#126 - enhancement - major performance enhancements for bulk edit operations in TrackMe #126
- trackme-limited/trackme-report-issues#127 - bug - Remove component does not remove some knowledge objects #127
- trackme-limited/trackme-report-issues#128 - enhancement - Workload - Allow the component to be added to / deleted from an existing Virtual Tenant #128
- trackme-limited/trackme-report-issues#129 - enhancement - splunkremotesearch - Roles Based Access Control support #129
- trackme-limited/trackme-report-issues#130 - enhancement - trackmeapiautodocs - Remove redundant resource_spl_example/resource_desc from endpoint usage output #130
- trackme-limited/trackme-report-issues#131 - bug - Data sampling & events format recognition - escaped double quotes are incorrectly escaped again leading the sampling generation to fail #131
- trackme-limited/trackme-report-issues#132 - bug - Data sampling & events format recognition - Reset loses the preset number of records, sets the number of records would fail if the entity has not been processed yet #132

11.1.157 Version 2.0.33 - build 1683898726 (12/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: b9e8494d654bc60d1f0e12afe220d10c10f87aab1dd2fd20e517511040f9f9c8
- trackme-limited/trackme-report-issues#115 - bug - splk-dsm - tags - tags policies not applied as expected due a native multivalue format when taken into account by TrackMe's REST API #115

- trackme-limited/trackme-report-issues#116 - enhancements - Acknowledgments UI behaviours consistency #116
- trackme-limited/trackme-report-issues#117 - enhancement - Workload (splk-wlk) - The Orphan check and maintain search takes too long #117
- trackme-limited/trackme-report-issues#118 - bug - Data Host Monitoring (splk-dhm) - max delay and max latency are not honoured properly #118

11.1.158 Version 2.0.32 - build 1683797653 (11/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: b570f9e6a668cfd895832cb2812e540e8a8e263606b49ae9014900d8e0683137
- bug - Workload (splk-wlk) - false positive issues with anomaly_reason=execution_delayed under some specific conditions #113
- bug - Workload (splk-wlk) - introspection metrics generation - introduce a bucket _time span=1m to properly aggregate metrics for pct_cpu/memory, sum the scan eventcount #114

11.1.159 Version 2.0.31 - build 1683730441 (10/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: 32d31b6b3c8eade39c27af09dbe2e5d8497a7cecbc5b374f1ba939555ae59069
- bug - ucc-framework issue with urllib3 v2.0.x - latest version of urllib3 require fresher openssl version which builtin Splunk versions do not meet causing issues in alert actions #112

11.1.160 Version 2.0.30 - build 1683715542 (10/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: 4652676182e6271bef61bc368db1fcdc3c216a26d022d4eb54dd6f28e8ec9168
- bug - all components - Tracking Alerts UI always created splk-dsm Alert #110
- bug - all components - SLA single should turn red if the entity has never been green since it was discovered #111

11.1.161 Version 2.0.29 - build 1683576225 (08/05/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: 60e8e0665f3d924d3f7b636fc372fb8f1c6d4ca9274681913ea795706ac804cb
- bug - Workload (splk-wlk) - issues in Metadata collection when using a remote account with more than one member in the account definition #107

- bug - Flex Object - demo search for deployment servers should filter for the group when doing the inputlookup back #108
- bug - Workload (splk-wlk) - mltrain should be scheduled once per hour, mlmonitor should be scheduled every 20 minutes to prevent skipping searches #109

11.1.162 Version 2.0.28 - build 1682667017 (28/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: 198ddc37df076de98e42a530bf66aa903eff8ae87c4c7d2e601b0c6316611c5d
- bug - splk-wlk (Workload) - If running in remote, introspection and Splunk Cloud SVC queries cannot rely on app fieldaliases #105

11.1.163 Version 2.0.27 - build 1682578920 (27/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: b226ad96a069f070b5293bfe50fab101503e56c2bdf2c2d2027ed2d06bb8bf50
- bug - splk-wlk - Missing field alias for svc-consumer causes SVC consumption not to render expected SVC metrics #104

11.1.164 Version 2.0.26 - build 1682503730 (26/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: fe68d95983066a1f8a2fcf2a4a60271ad1ce91d457c56f76f228a68418059baa
- feature - Introducing the new Splunk Workload component for TrackMe, to monitor your Splunk scheduling activity and take the control back #102
- bug - splk-cim - avoids append=t in the very first pipe which causes issues in Splunk Cloud #103

11.1.165 Version 2.0.25 - build 1682069909 (21/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

Note: Hybrid Trackers need to be re-created to benefit from the latest_eventcount_5m

- SHA-256: d992c12d1bb9998bc39be0171c3721d4c3f30ecef2ee0be1bfc1ab93dac29897
- bug/enhancement - latest_eventcount_5m from TrackMe metrics should perform an aggregation to properly represent the 5m sum of eventcounts #94

11.1.166 Version 2.0.23 - build 1681985039 (20/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: e03e25136a8803cea926721d959a2312cdbcdec70f810279de3ffdf9c3cf5043
- bug - splk-feeds - Hybrid tracker creation, if breaking by host in splk-dsm, the dcount host leads to wrongly interpreting the host value, issues with burn test in raw mode #99
- bug - Outliers detection - incorrect message statement when upperBound is breached #100

11.1.167 Version 2.0.22 - build 1681860827 (19/04/2023)

Hint

Splunk 8.1.x and later, Linux, Python3 support only

- SHA-256: 08ae4facab3c6c141f0967998562bd1440fe1e1d6fe8ee8c85cef47a0191b81a
- bug - ack tracker regression issue introduced in release 2.0.21 #97
- bug - alerts creation - incorrect statement when including orange status for entities #95
- enhancement - splunkremotesearch - accepts a list of multiple Splunk REST endpoints and address targets randomly with HA and DR #93
- bug/enhancement - avoid disabling access to the acnknowledgement if it is still active althrough the entity is back in green state #96

11.1.168 Version 2.0.21 - build 1681766136 (17/04/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 3b15dff23199adb46b8305cda8172062e25ddc24d3610e8da3a90345e4d08077
- bug - regression in trackmecollect for splk-dhm. the field splk_dhm_st_summary is required by the UI for processing #92

11.1.169 Version 2.0.20 - build 1681751403 (17/04/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 59f122da1acc5728f8192365adf4a8b4f83bbd5e740d87f05d62678bdfaea020
- change - disable drilldown in API ref table #78
- change - Add skipping search shortcut access in Virtual Tenant (skipping donut screen) #79
- bug - mistmatch between custom command log files and associated props stanza #80
- bug/enhancement - improve detection of latency at ingest and its sensittivity using TrackMe metrics #81

- bug - trackmepersistentfields backend would raise an exception and block the remaining updates if an unexpected error occurs in the update process #82
- enhancement - avoids TrackMe custom command to be distributed amongst indexes while it's unecessary #83
- bug/enhancement - reduce the foot print of TrackMe state events stored in the summary indexes, prevents unecessary large fields (metrics summary, etc) #84
- enhancement - Preparation for the Implementation of least privileges approach in TrackMe and advanced capabilities management #85
- enhancements - Python backend enhancements #86
- enhancement - Add or Delete components for a TrackMe Virtual Tenant after it was created #87
- bug - "Show burn test search" creates a persistent macro #88
- bug/enhancenent - splk-feeds - Maintain delayed entities running out of the scope of TrackMe trackers #89
- enhancement - massive performance gains in events generating Python backends #90
- enhancement - trackmesplkoutlierstrainhelper should implement a max run time sec mechanism to avoid generating skipping search #91

11.1.170 Version 2.0.19 - build 1680519959 (03/04/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 7f418e954415f4bdd74e8ce685eca7dab1b160ea6706dc6a0170b8fca65b571a
- bug - splk-dsm - data_first_time_seen should be part of persistent fields in the macro trackme_dsm_lookup_persistent_fields #75
- enhancement - trackmepersistentfields command - in some circumstances, there can be an unexpected duplication of entities, this enhancement ensures that this cannot happen #76

11.1.171 Version 2.0.18 - build 1680475914 (02/04/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 71dd7ac5314ea3826c19a323844834bad95f3f98de317edb1ea05313761667e3
- bug/enhancement - TrackMe metrics generation and vizualisation issues when suffering from latency or low frequency entities #72
- bug - Virtual Tenant UI graphical issue when testing remote connectivity #73
- bug/enhancement - Improve latency detection by taking into account TrackMe metrics at Hybrid Tracker execution time #74
- enhancement - improve consistency of wording for lagging / latency / delay concepts #10

11.1.172 Version 2.0.17 - build 1680257518 (31/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: c4c68dc01cf1998db95566c15dc89228478848d969a583eaa617b142ac276547
- bug - splk-dsm/splk-flx status flipping will incorrectly continue to see new entities being discovered due to regression in 2.0.15 #71

11.1.173 Version 2.0.16 - build 1680138733 (30/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: e07a3f909033b93089541f27b1834ef327910f9f6c50ff11eade33b7e24fbb5c
- bug - splk-dsm - bad syntax in screen auto lagging def #68
- bug - splk-dsm/splk-dhm - avoid continuing to generate TrackMe metrics for an entity which data flow is interrupted, restrict the metrics scope to the 5 last minutes against the last event of the entity #69
- enhancement - Some high scale SHC environments with a large number of entities, especially in Splunk Cloud, were reported to encounter out of sync issues due to ML models update activity, this release reduce the frequency of the ML train activity to avoid this #70

Notes:

- Regarding fix #69, Hybrid Trackers need to be re-created, or manually updated:

trackme_dsm_hybrid_abstract_<id>

the break by change may change depending on your context, the fix relies on restricting the the spantime to avoid generating new metrics while the flow is interrupted

```
| eval spantime=_time | eventstats max(data_last_time_seen) as data_last_time_seen␣
↪by index,sourcetype | eval spantime=if(spantime>=(now()-300), spantime, null())
```

11.1.174 Version 2.0.15 - build 1679995508 (28/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: affba63ecf9fc7a8b718d5c45894dc64f920ec6d36f1e9794ca7d76f3ca54272
- bug / enhancements - introducing the custom command trackmepersistentfields to protect KVstore collection records from conflicting updates and replace the call to outputlookup Splunk command with more control #55
- bug - Vtenant creation endpoint should set the current schema_version immediately at the creation phase #56
- enhancement - Allow splunkremotesearch command to inherit earliest and latest from the environment (time range picker) #57
- bug/enhancement - avoid skipping searches for ML train/monitor and data sampling by reducing the default cron to every 20 when creating a new tenant #58
- enhancement - Limit the tenant name identifier to 15 characters max to avoid allowing users from reaching any Splunk limitations, reduce the random digits for trackers to 5 #59

- bug/enhancement - splk-dsm and splk-flx, at large scale with large number of concurrent Hybrid Trackers, concurrent loading of whole collections lead to impacts on other entities #60
- enhancement - Store the root constraint in a macro when creating the Hybrid Trackers for splk-feeds, for easier design, update and management #61
- bug - inherit trackmer_user role in trackme_admin to avoid any non explicit read access #62
- bug - If using Federated search in the instance running TrackMe, makeresults duplicates results unexpectly #63
- enhancement - splk-feeds Hybrid Tracker creation improvements, new builtin options to control performance denominators, review Burn test search before execution #64
- bug - Outliers management issues and enhancements #65
- change - Licensing management evolutions #66
- bug - log rotation is lacking for the various trackme logs #67

11.1.175 Version 2.0.14 - build 1679295918 (20/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 5cc6306228293260ee82801bbf198a65ca13aedc6bf68bc0bda983b6ba6cae8c
- bug - conflict the same object exists already error when attempting to create a lagging class for the same conditions if one exists already for another category #45
- feature - splk-flx - Allow to control grouping of entities #46
- bug - splk-cim/splk-flx - metric ingestion issues when objects have space characters #47
- bug - negative value metrics will be ignored in splk-flx #48
- bug - indexes preset by default in tenant creation dropdown regression from 2.0.13 - showing first result index rather than preset index #49
- bug/enhancement - detect and degrade a Virtual Tenant using remote splunk account that was removed later on, or if all remote accounts were removed post configuration #50
- bug - Virtual Tenant UI - copy spl button may generate trackme SPL commands that cannot be parsed properly #52
- feature - Provide a burn test performance benchmark feature while creating Hybrid Trackers to investigate the run time performance ahead of the tracker creation #53

11.1.176 Version 2.0.13 - build 1678259747 (08/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: cc4d34f9f54e4fce2dd4299cc4bb549974ec7395a63b6eb4159ee46f2a7b02e5
- bug/enhancement - reduce volume of logs in trackme_splk_outliers_train_helper.log #41
- bug - lagging classes does not accept splk-dsm / splk-dhm pattern, failures to apply lagging classes against object!=all, various issues affecting lagging classes for splk-dhm #40
- bug - timezone issue in REST API and custom command logging events when the user running the command is in a non UTC timezone #43

11.1.177 Version 2.0.12 - build 1678171647 (07/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 001d57ab9960024fde3eabf9439e1643ee118b99626b7f46e1d7ad3797c65378
- enhancement - avoids any enabled scheduled report by default including app level management utilities (Ack tracker, backup scheduler, maintenance mode tracker) #33
- bug - merged mode for splk-dsm not behaving as expected #34
- bug - Virtual Tenants UI regression when deleting the last tenant (should refresh and show up Welcome modal screen) #35
- enhancement - reduce the default earliest to -4h instead of -7d when creating Hybrid trackers to limit design requirements for first time users #36
- enhancement - improve consistency of wording for lagging / latency / delay concepts #10
- bug - missing perc95_latency_5m and stdev_latency_5m metrics for splk-dhm #38
- enhancement - Improve global TrackMe experience for splk-feeds with Overview based on TrackMe metrics primarly rather than direct Splunk query (Allows faster query and scalability, enhance RBAC consistency) #37

11.1.178 Version 2.0.11 - build 1677767350 (01/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 16f797f4140bbff976c9d7ff7fb093f5ac519f1b699ff7010aa097e8474c4e8e
- bug - Entity remains in red state due to Data sampling detection altrhough the feature has been disabled #28

11.1.179 Version 2.0.10 - build 1677707255 (01/03/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 423dc06178dd7360ccbffa3741dd7e41ae4ad63eb8cdb9bb703f86828729a3d2
- bug - custom indexes not properly used when creating Virtual Tenants from the user interfaces for splk-dsm/dhm/mhm #30
- bug - regression from 2.0.9 preventing access to RBAC update from the Virtual Tenant UI #31

11.1.180 Version 2.0.9 - build 1677588126 (28/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: edd8c6d22bc6fb80c9b7c08ee46b58d05ea2970f41678c89d6cfbf8f88f3d5d4

- bug: Virtual Tenants UI fails to load properly if a Virtual Tenant is disabled and was created with value for its description #21
- bug: Virtual Tenant creation error handling issues can lead to undetected failures within the Virtual Tenant user interface #22
- bug/enhancement: Virtual Tenants objects creation - avoid and enhance detection and re-attempt if splunkd API is not ready yet to server the newly created object #23
- bug/enhancement: disable auto-refresh in Virtual Tenants UI during long run operations to avoid loosing the spinner #24
- enhancement: splk-feeds - bulk edit management for Logical groups (splk-dsm/dhm/mhm) #25
- feature: introducing the concept of TrackMe schema versioning to allow future automated updates to the Virtual Tenants & Knowledge Objects schema #27
- feature: Sticky Acknowledgements #9
- bug/enhancement: Single forms and Donut drilldown do not lead to actions (all components) #16
- feature: license model update to allow an intermediate pricing plan with the Enterprise Edition #29

11.1.181 Version 2.0.8 - build 1677163367 (23/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 80d0437c355c1ab71930bbf68f6ae0739817994c087712888f65d86d074678b2
- bug/enhancement: splk-dsm Data sampling - Tabulator occasionally loads before the modal screen, optimize and avoid multiple REST calls #11
- bug/enhancement - splk-flx - simplify the regular expression used in the deploymet server example #12
- bug - splk-flx - copy to clipboard button not working for deployment server example from first level modal screen #13
- Enhancement - improving naming convention consistancy in status and anomaly_reason #20
- Feature request - logical grouping to be made available for splk-dsm component #18
- bug - splk-dhm/splk-mhm entity view host Metadata filter do not apply when hybrid tracker was created manually in a tenant (opposed to created during the Virtual Tenant creation phase) #19

11.1.182 Version 2.0.7 - build 1676377640 (14/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 13bc28f5693f9e6f7391ac2f61ddd598818d372c396d4f0d53bc6f5faf4fa865
- bug: splk-dsm - dictinct count host issue inconsistency when setting up a dcount_host treshold #1
- bug: splk-dsm - Elastic source syntax issue with from datamodel sources - error in identification of remote from searches #5
- feature: splk-dsm - Feature request - Simulation of thresholds before applying #3
- enhancement: Put a clear RBAC related message in when creating Virtual Tenants regarding membership explicit management
- enhancement: TrackMe Alert Suppression/Throttling Enhancements #6

- bug/enhancement: bug Tabulator loading modal - all components - In some circumstances, the screen can load before the REST endpoint call return the Tabulator data #7
- enhancement: Feature - Disable Ack when an entity goes back to green #8 - You can now enable the option "Remove Ack behaviour" in configuration if you wish to have Ack being disabled automatically when a previously non green entity comes back to green, rather than relying only on the Ack expiration - As well, there has been enhancements on the Ack tracker backend for better reporting and auditing of its activity (generate an audit event per entity)

Notes: - Hybrid/Elastic Trackers need to be re-created to benefit from the new distinct count hosts metrics for splk-dsm (Feeds tracking for Data Sources)

11.1.183 Version 2.0.6 - build 1675851310 (08/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: a5bf6e9580ca9924d20ea00c029a4cd61f6bffa700a493a2a8e251934d030bdb
- issue with splk-dhm timecharts in Splunk remote deployments when data gaps occur #9
- issue with splk-dhm compact mode which should show the sourcetype in addition with the index in the JSON summary #11
- wrong label in lagging classes applies to dropdown for splk-dsm/splk-dhm #12

11.1.184 Version 2.0.5 - build 1675711433 (06/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: ab77d89634b3debc5d2ddd881243310bbb18b959254efc53dcf6a83a873c5427
- Fix - Some REST endpoints are unexpectedly limiting their output to the first 100 records #7

11.1.185 Version 2.0.4 - build 1675617150 (05/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- Optimization - function dataset_update_cache should sleep before retrying in case of max concurrent searches run Optimization - function dataset_update_cache should sleep before retrying in case of max concurrent searches run #4
- Optimization - avoid logging check license return in non debug mode Optimization - avoid logging check license return in non debug mode #3
- Optimization - reduce internal logs from datagen custom command Optimization - reduce internal logs from datagen custom command #6

11.1.186 Version 2.0.3 - build 1675586140 (05/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: 661069bc7dfe803c9e6c10021cb693c85e616dce13b54c708f38ddc760848df4
- Data sampling engine - syntax error leads custom rule in simulation mode to fail rendering the expected results #1

11.1.187 Version 2.0.2 - build 1675379421 (02/02/2023)

Hint

Splunk 8.2.x/9.x and Python3 support only

- SHA-256: b5edf46f5bf6a293b318d33b0e4b07c982019dae427d4ad7b7b1b6881fb74145
- This the first official release for TrackMe V2

11.2 Release notes: trackme-broker

These are the release notes of the TrackMe Broker add-on, which runs on a Heavy Forwarder on your side. For TrackMe itself, see *Release notes: trackme*. For what the add-on does and how to set it up, see *TrackMe Broker*.

TrackMe Broker is published on Splunkbase. Each release is also on TrackMe Solutions Downloads, next to its checksum, `release-sha256.txt`.

Note

TrackMe Broker needs TrackMe 2.4.24 or later on the stack and, on the forwarder, Splunk Enterprise 9.4.x or later with the TrackMe Broker add-on 1.1.1 or later. It is compatible with Splunk FIPS mode. See *TrackMe Broker compatibility*.

11.2.1 Version 1.1.3 - build 1794843047 (16/11/2026)

Hint

TrackMe Broker 1.1.3 — the add-on's page has its browser tab icon, and its interface libraries are up to date.

Package: trackme_broker-1.1.3.tgz

SHA-256: a982c552a89c0d3251f62758667525035e1a32943041494ba30fb436fb9668aa

- The add-on's page now shows the TrackMe icon in the browser tab.
- The add-on's page is built with the latest Splunk UI libraries.
- Nothing changes in pairing, tokens or calls: a forwarder upgraded from 1.1.2 keeps its connection and its targets.

11.2.2 Version 1.1.2 - build 1793980453 (06/11/2026)

Hint

TrackMe Broker 1.1.2 — a healthy broker no longer writes errors to splunkd.log.

Package: trackme_broker-1.1.2.tgz

SHA-256: 08e657d05912c069facb373c7f0a5ac41264eea8c48220443659051db87c8819

- About once a minute, the add-on wrote a Python traceback to stderr about a credential that is normally absent, and splunkd recorded each of its lines as an `ExecProcessor` error. Nothing was failing: pairing, tokens and calls were not affected, and the lines held no secret. Monitoring that forwarder could report them as input errors on the broker's own input.

11.2.3 Version 1.1.1 - build 1793891238 (05/11/2026)

Hint

TrackMe Broker 1.1.1 — initial public release.

Package: trackme_broker-1.1.1.tgz

SHA-256: 6b125d484012ba2ae3d3a6cbec8e664fad0cbceed9fbcc65e08aba005967cc10

CHAPTER

TWELVE

THIRD PARTY INTEGRATIONS:

12.1 Palo Alto XSOAR Integration

About Palo Alto XSOAR Integration

- TrackMe provides a native integration with Palo Alto Cortex XSOAR, allowing to easily track and monitor Palo Alto XSOAR alerts and incidents.
- This integration provides REST API integration capabilities with Palo Alto Cortex XSOAR for comprehensive incident management and monitoring.

12.1.1 Overview

TrackMe XSOAR Integration (TA-trackme-xsoar) is a Splunk Technology Add-on that provides seamless integration between TrackMe and Palo Alto Cortex XSOAR. This integration enables:

- **REST API Integration**: Create one or more XSOAR accounts in TrackMe to interact on-demand with Cortex XSOAR
- **Authentication**: Based on XSOAR key and key ID as per the XSOAR API
- **Incident Management**: Create or update incidents in a streaming manner
- **Resilient Backend**: Automatically stores and reattempts failing REST API calls
- **Comprehensive Logging**: Consistent and easy access to different logs

12.1.2 Key Features

Core Functionality:

- **XSOAR Command**: Use the generating command *xsoar* to run GET/POST/DELETE REST calls to the XSOAR API
- **Streaming Incidents**: Use the streaming command *xsoarstreamincident* to create or update incidents in a streaming manner
- **Resilient Store**: Automatically stores and reattempts failing REST API calls for incident management
- **Incident Store**: Stores incidents and key information in KVstore for tracking and management

Authentication & Security:

- XSOAR key and key ID based authentication
- FIPS compatible
- Compatible with both standalone instances and Search Head Cluster (SHC) deployments

12.1.3 Installation

Installation Requirements:

- No dependencies required
- Developed on top of the Splunk UCC framework
- Compatible with all supported versions of Splunk
- Compatible with both standalone instances and Search Head Cluster (SHC) deployments
- FIPS compatible

Installation Steps:

1. Download the application from the official download page
2. Install as usual in Splunk (For Splunk Cloud, install as a private application)
3. Restart Splunk after the initial installation
4. Configure XSOAR accounts in the application

Download:

- **Official Download Page**: *https://downloads.trackme-solutions.com/TA-trackme-xsoar/*

12.1.4 Usage

XSOAR Command Usage:

The *xsoar* command provides REST API access to XSOAR:

```
| xsoar account="xsoar" method="GET" endpoint="/incident"
```

Streaming Incident Command:

The *xsoarstreamincident* command creates or updates incidents:

```
| eval incident_name="TrackMe entity alert", incident_details="Entity status␣
↪changed", incident_severity=2
| xsoarstreamincident account="xsoar" incident_details="$incident_details$"␣
↪incident_name="$incident_name$" incident_severity=2 incident_type="unclassified"
```

Active Commands for TrackMe Integration:

For opened incidents:

```
| eval incident_name="The TrackMe entity " . alias . " has received an incident",␣
↪incident_details="The TrackMe entity " . alias . " has received an incident: " .␣
↪object_state . ", detection_time: " . detection_time . ", messages: " .␣
↪mvjoin(messages, "|")
| xsoarstreamincident account="xsoar" incident_details="$incident_details$"␣
↪incident_name="$incident_name$" incident_severity=2 incident_type="unclassified"␣
↪external_incident_id="$incident_id$"
```

For closed incidents:

```
| eval incident_name="The TrackMe entity " . alias . " has received an incident␣
↪closure", incident_details="The TrackMe entity " . alias . " has received an␣
↪incident closure update and is now in a non-alerting state: " . object_state . ",
↪ detection_time: " . detection_time . ", messages: " . mvjoin(messages, "|"),␣
↪incident_reason="The TrackMe entity has returned to a non alerting state"
| xsoarstreamincident account="xsoar" incident_details="$incident_details$"␣
↪incident_name="$incident_name$" incident_severity=2 incident_type="unclassified"␣
```

(continues on next page)

(continued from previous page)

```
↪incident_reason="$incident_reason$" incident_closeNotes="$incident_details$"␣
↪external_incident_id="$incident_id$"
```

12.1.5 Command Parameters

xsoarstreamincident Command Parameters:

The following arguments are available for the *xsoarstreamincident* command:

- **external_incident_id**: The external incident ID for lookup and update
- **incident_closeNotes**: Notes for closing the incident
- **incident_closeReason**: Reason for closing the incident
- **incident_closed**: The date the incident was closed
- **incident_createInvestigation**: Whether to create an investigation for the incident
- **incident_customFields**: Custom fields for the incident, as a JSON string
- **incident_details**: The details of the incident
- **incident_labels**: Labels for the incident, as a JSON string
- **incident_modified**: The date the incident was last modified
- **incident_name**: The name of the incident
- **incident_playbookId**: The ID of the playbook to run for the incident
- **incident_rawJSON**: The raw JSON of the incident
- **incident_reason**: The reason for the incident
- **incident_severity**: The severity of the incident (number)
- **incident_sla**: The SLA for the incident (number)
- **incident_status**: The status of the incident (number)
- **incident_type**: The type of the incident

12.1.6 Data Storage

Incident Store:

The application stores incidents and their key information in the following KVstore collection:

- **KVstore name**: kv_xsoar_incidents_store
- **KVstore transforms name**: xsoar_incidents_store

Access the KVstore content:

```
| inputlookup xsoar_incidents_store | eval keyid=_key
| eval _time=mtime
| sort - 0 _time
```

Resilient Store:

The application stores failing calls in a Splunk KVstore for automated reattempt purposes:

- **KVstore name**: kv_xsoar_resilient_store
- **KVstore transforms name**: xsoar_resilient_store

Access the resilient store content:

```
| inputlookup xsoar_resilient_store | eval keyid=_key
| eval _time=ctime
| sort - 0 _time
```

Resilient Store Configuration:

- Enable or disable resilient store functionalities (default is enabled)
- Define maximum number of attempts for a given REST call
- Automated reattempts using the command *xsoarresilient*
- Orchestrated by the saved search: xsoarresilient (scheduled by default)

12.1.7 Troubleshooting

Log Access:

The application provides comprehensive logging for troubleshooting:

Application internal REST API: .. code-block:

```
index=_internal sourcetype=xsoar:rest_api
```

Command xsoar: .. code-block:

```
index=_internal sourcetype=xsoar:custom_commands:xsoar
```

Command xsoarstreamincident: .. code-block:

```
index=_internal sourcetype=xsoar:custom_commands:xsoarstreamincident
```

Command xsoarresilient: .. code-block:

```
index=_internal sourcetype=xsoar:custom_commands:xsoarresilient
```

Audit & Troubleshoot Menu:

The application provides an "Audit && Troubleshoot" menu with shortcuts to internal log events. Use the extracted field *log_level* to filter on certain types of log events such as *error*.

Resilient Store Behaviors:

- When a reattempt is needed and processed successfully, the information is logged and the KVstore record is permanently deleted
- When a transaction ID record reaches the maximum number of attempts, the information is logged and the KVstore record is permanently deleted

Resilient Store Limitations:

- Updates or closures require first the response from XSOAR to know the incident id
- Updates or closures cannot be re-attempted if the initial incident opening call has not been successful
- The resilient store can re-attempt an update of an incident only if the incident could be opened successfully first

12.1.8 Documentation References

Official Documentation:

- **Cortex SOAR REST API Documentation**: *https://docs-cortex.paloaltonetworks.com/r/Cortex-XSOAR-8-API*
- **Create or Update Incident API**: *https://docs-cortex.paloaltonetworks.com/r/Cortex-XSOAR-8-API/Create-or-update-an-incident*

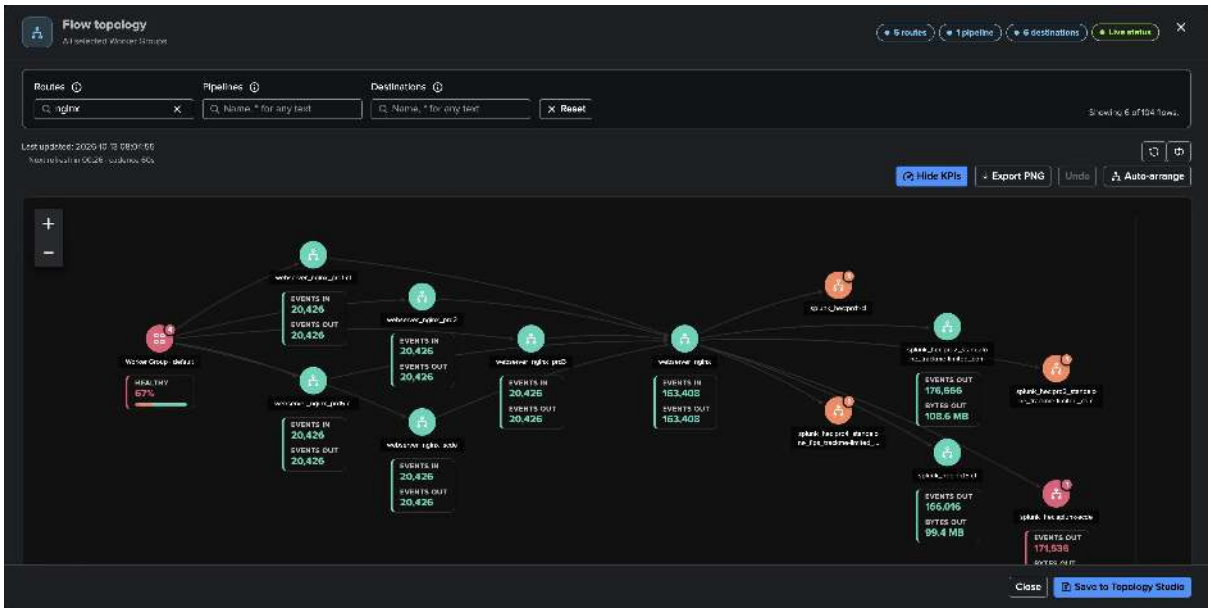
Resources:

- **Download Page**: *https://downloads.trackme-solutions.com/TA-trackme-xsoar/*

Related TrackMe Documentation:

- *Stateful alerting* - For information about TrackMe alerting capabilities

12.2 Cribl Stream & Cribl Edge



Cribl sits in the middle of the data path: when a Cribl Leader, a Worker, an Edge Node, a Source or a Destination has a problem, the data behind it slows down, backs up or stops. TrackMe monitors Cribl in complementary ways. Pick the one that matches what you want to watch:

What you want to watch	What to use
A Cribl Stream deployment — its Leader, Worker Groups, Workers, Sources, Destinations, Pipelines and Routes	The **Cribl Stream tenant**: *Cribl Stream*
A Cribl Edge fleet — its Fleets, Edge Nodes, Sources and Destinations	The **Cribl Edge tenant**: *Cribl Edge*
The data sources and hosts whose logs pass through Cribl	DSM and DHM with Cribl signals: *Monitor data sources and hosts with Cribl signals*
Custom Cribl monitoring on your own SPL	Flex Objects and the Cribl API add-on: *Cribl API add-on and Flex use cases (annex)*

The Cribl Stream and Cribl Edge tenants are generally available in TrackMe 2.4.19. Both start from a **Cribl connection** (*Cribl connections*), declared once in TrackMe and shared by every tenant that reads the same Cribl deployment.

12.2.1 How the Cribl tenants work

Dedicated tenant types. A Cribl Stream tenant and a Cribl Edge tenant are dedicated tenant types, not entity components: they have no entities, no decision maker and no health score, and none of the DSM, DHM, MHM, FLX, FQM, WLK or VOL components. Each has its own workspace (Tenant Home), its own collection jobs, a catalogue of built-in **policies** and the **findings** those policies raise.

Read-only, nothing installed on Cribl. TrackMe reads what already exists and never changes anything in Cribl:

- the **Cribl Leader API**, for the configuration (Worker Groups or Fleets, Sources, Destinations and, on Stream, Pipelines, Routes and Packs), the Workers or Edge Nodes and the deployment state. Every call is a read; the

only requests sent with POST are the sign-in and, on Edge, the query of an Edge Node's own system metrics, which changes nothing;

- for Cribl Stream, the **Cribl internal metrics** that Cribl already sends to a Splunk metric index. They are computed in place with `mstats`: nothing is copied into a TrackMe index, and the metrics cause no additional licence ingestion.

No agent, Pack or configuration is installed on Cribl, and no tracker has to be written.

Gentle with the Leader, at any size. Every TrackMe caller that reads one Leader — every tenant, every job, the connection tests — draws from one shared **Leader request budget** (1 200 requests per hour by default, a setting of the connection). A run that would exceed it waits for the next one and resumes where it stopped instead of bursting; unchanged Worker Groups are confirmed without a read; Cribl Edge reads the node list in bulk, never one call per node. When the budget cannot keep up, Collection health says so and names the remedy.

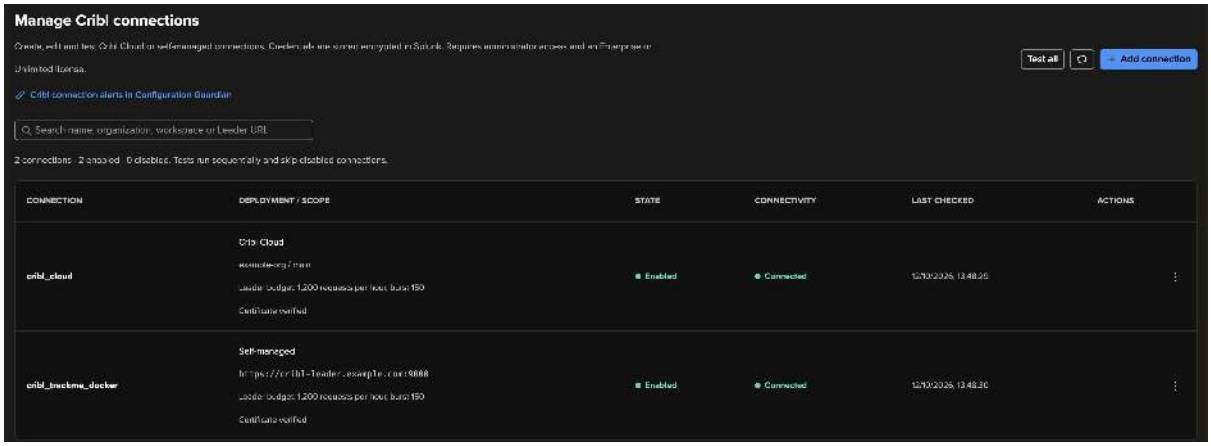
Findings, the shared model. Both tenants raise findings — one per object, policy and measurement — with Finding groups, the TrackMe severity scale (Informational, Low, Medium, High, Critical), triage, email notifications, assignment and findings actions. The model is the same in every dedicated tenant type and is described once in *Findings, Finding groups and findings actions*.

Cribl Stream and Cribl Edge at a glance

	Cribl Stream tenant	Cribl Edge tenant
Monitors	The Leader, Worker Groups, Workers and Worker processes, Sources, Destinations, Pipelines and Routes	Fleets (and Subfleets), Edge Nodes, the Sources and Destinations of each Fleet, the Leader's availability
Reads	The Cribl internal metrics in a Splunk metric index (required) and the Leader API	The Leader API; optionally a System Metrics index for the CPU and memory of the Edge Nodes
Built-in policies	16	11
Traffic ML models, Expected traffic	Yes	No
AI	AI Assistant, Concierge actions and the Cribl Stream AI Agent	AI Assistant and the Cribl Edge AI Agent
Collection jobs	Inventory, live metrics, traffic model training	Inventory, live
Size of one tenant	Up to about 20 000 monitored objects	Any fleet size
Shared by both	*Cribl connections*, the Leader request budget, findings and Finding groups, notifications, assignment, findings actions, Collection health, Topology Studio	

12.2.2 Cribl connections

A **Cribl connection** tells TrackMe how to reach one Cribl Leader: a Cribl.Cloud workspace or a self-managed Leader. It is the prerequisite of every Cribl Stream and Cribl Edge tenant. Connections live in **Configuration → Manage Cribl connections**; the credential is stored encrypted in Splunk and never shown again, and in a Search Head Cluster the connections replicate to every member.

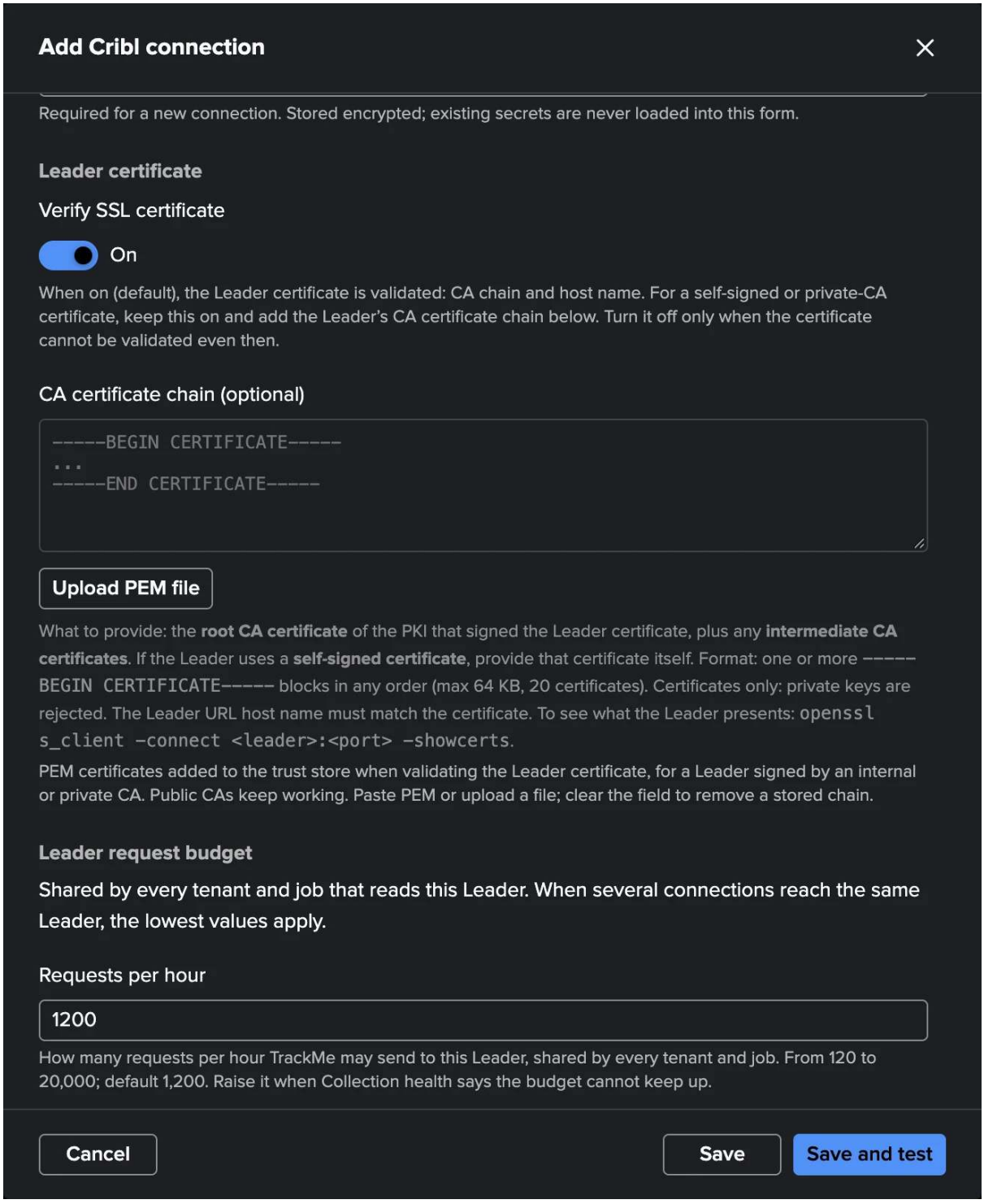


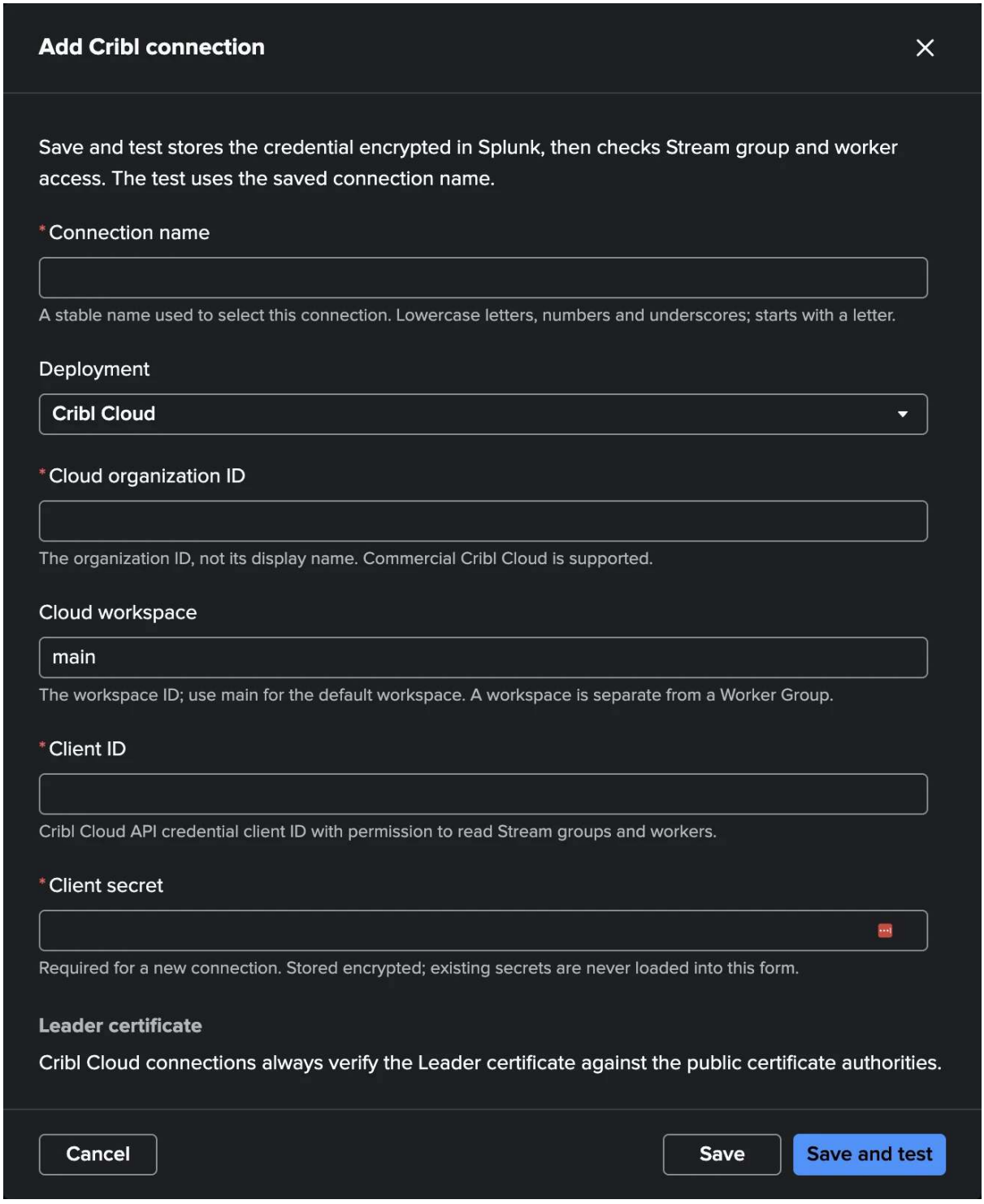
Adding a connection

Click **Add connection**, give the connection a name (lowercase letters, digits and underscores, starting with a letter; it cannot change later) and choose the deployment:

	Cribl.Cloud	Self-managed Cribl
Where	**Cloud organization ID** (the ID, not the display name) and **Cloud workspace** (`main` for the default workspace)	**Leader URL**: an HTTPS origin only, for example `https://cribl.example.com:9000` (no path, credentials or query)
Credential	An API credential: **Client ID** and **Client secret** (OAuth client credentials), created in Cribl.Cloud (*below*)	A Cribl **Username** and **Password**: a service account that signs in with a user name and password, a Cribl local user when the Leader uses SSO (*below*)
How TrackMe signs in	Exchanges the client ID and secret at `login.cribl.cloud` for a bearer token, valid 24 hours	Signs in at `/api/v1/auth/login` on the Leader for a bearer token, valid for the Leader's **Auth token TTL** (1 hour by default)
TrackMe reaches	`login.cribl.cloud` for the sign-in, then the workspace URL `https://<workspace>-<organization>.cribl.cloud`	The Leader URL
Leader certificate	Always verified against the public certificate authorities	Verified by default; a custom CA chain can be added (below)

The Splunk search heads that run the tenants' collection must reach these addresses over HTTPS. HTTP proxies and client certificates (mutual TLS) are not supported. A self-managed Leader they cannot reach, such as one on your own network seen from Splunk Cloud, can go *through a TrackMe Broker*.


Add Cribl connection
Required for a new connection. Stored encrypted; existing secrets are never loaded into this form.
Leader certificate
Verify SSL certificate
On
When on (default), the Leader certificate is validated: CA chain and host name. For a self-signed or private-CA certificate, keep this on and add the Leader's CA certificate chain below. Turn it off only when the certificate cannot be validated even then.
CA certificate chain (optional)
-----BEGIN CERTIFICATE-----
...
-----END CERTIFICATE-----
Upload PEM file
What to provide: the root CA certificate of the PKI that signed the Leader certificate, plus any intermediate CA certificates. If the Leader uses a self-signed certificate, provide that certificate itself. Format: one or more -----BEGIN CERTIFICATE----- blocks in any order (max 64 KB, 20 certificates). Certificates only: private keys are rejected. The Leader URL host name must match the certificate. To see what the Leader presents: openssl s_client -connect <leader>:<port> -showcerts.
PEM certificates added to the trust store when validating the Leader certificate, for a Leader signed by an internal or private CA. Public CAs keep working. Paste PEM or upload a file; clear the field to remove a stored chain.
Leader request budget
Shared by every tenant and job that reads this Leader. When several connections reach the same Leader, the lowest values apply.
Requests per hour
1200
How many requests per hour TrackMe may send to this Leader, shared by every tenant and job. From 120 to 20,000; default 1,200. Raise it when Collection health says the budget cannot keep up.
Cancel
Save
Save and test



In both modes every API request carries a **bearer token**, never the credential itself. TrackMe obtains the token, keeps it while it is valid and signs in again on its own when it expires: nothing needs renewing by hand.

A self-managed Leader the stack cannot reach

From Splunk Cloud, a self-managed Leader on your own network cannot be reached directly. A *TrackMe Broker* solves this: an add-on on a Heavy Forwarder on your side carries TrackMe's calls to the Leader. Every connection starts on your side, so you open nothing inbound.

1. *Set up a broker*, and on its forwarder declare the Leader as a target, for example `https://cribl.example.com:9000`. A Cribl target covers the whole origin.
2. On the connection, choose **Self-managed Cribl** and set **Connectivity** to **Through broker** and that broker's

name.

3. Select **Save and test**.

Connectivity appears only for a self-managed Leader, because a Cribl.Cloud connection stays direct. Through a broker, the broker's target decides how the Leader certificate is verified, so a private CA is declared on that target and not on the connection. The list of connections shows **Broker** and its name under **Connection type**.

A broker outage raises no Leader finding: the Leader reads *unknown*, then *reachable* again by itself. The broker carries TrackMe's calls to the Leader API. Cribl's internal metrics still reach your Splunk metric index by their own route. See *TrackMe Broker — manage and monitor* to run the broker.

Cribl.Cloud: the API credential

1. In Cribl.Cloud, as an organization **Owner** or **Admin**, open **Organization → API Credentials** and select **Add Credential**. Name it after its use, for example `trackme`.
2. Under **Organization Permissions**, choose the **User** role, then under **Workspace Access** give it the workspace and the product the tenants monitor (Cribl Stream, Cribl Edge or both), with the most restricted permission that can read what TrackMe needs (*What the Cribl account needs*). The **Admin** and **Owner** roles give administrator access to every workspace of the organization: avoid them. On Cribl.Cloud plans without the Organization Permissions selector, every API credential has the Admin role.
3. Copy the **Client ID** and the **Client secret** into the TrackMe connection. The **Cloud organization ID** and the **Cloud workspace** are the two parts of your workspace URL, `https://<workspace>-<organization>.cribl.cloud`.

The client secret never leaves Splunk except for the sign-in at `login.cribl.cloud`. When the credential is rotated or disabled in Cribl.Cloud, update the connection.

Self-managed Cribl: the user account

A self-managed (customer-managed) Cribl deployment has **no API credential and no API token** to create: Cribl reserves the client ID and secret to Cribl.Cloud. Its API accepts one way in, a sign-in with a user name and password at `/api/v1/auth/login`, which returns a short-lived bearer token. This is **not** HTTP Basic authentication:

- TrackMe sends the user name and password **only to that sign-in endpoint**, over HTTPS (with the Leader certificate checks described below). Every other request carries the bearer token.
- The token lives for the Leader's **Auth token TTL** (*Settings → Global → General Settings → API Server Settings → Advanced*, 1 hour by default). TrackMe signs in again when it expires or when the Leader refuses it. There is no need to raise the TTL for TrackMe.
- The password is stored encrypted in Splunk and is never shown again.

The account TrackMe uses:

- **Signs in with a user name and password.** With Local, Splunk or LDAP authentication on the Leader, that is an account that can sign in on the Leader's login page with a user name and password.
- **Is a Cribl local user when the Leader uses SSO.** Single sign-on accounts (OpenID Connect or SAML) cannot sign in through the API. Create a local user for TrackMe and make sure **Allow login as Local User** is on (*Settings → Global → Access Management → Authentication*); otherwise the connection test fails on the sign-in.
- **Is a dedicated service account**, for example `trackme`, rather than a person's account, so that a password change or a departure does not stop the collection.
- **Reads, never writes**: give it the most restricted role that can read what TrackMe needs (*What the Cribl account needs*).

When the password changes, or expires under your Cribl password policy, update the connection: until then its tests fail and the Configuration Guardian reports `cribl_connection_health`.

Leader certificate (TLS)

HTTPS is mandatory in every mode. For a self-managed Leader:

- **Verify SSL certificate** (on by default) validates the Leader certificate: its CA chain and its host name. Keep it on.
- **CA certificate chain (optional)**: when the Leader certificate is signed by a private CA, or is self-signed, paste or upload (**Upload PEM file**) the root CA certificate and any intermediate CA certificates — or the self-signed certificate itself. The chain is **added** to the public certificate authorities, so publicly trusted Leaders keep working. Certificates only: a private key is refused (64 KB and 20 certificates at most). The Leader URL host name must match the certificate.
- Turning **Verify SSL certificate** off keeps the traffic encrypted but no longer checks the Leader's identity. Use it only as a last resort, when the certificate cannot be validated even with its CA chain. It is refused for Cribl.Cloud.

The table shows the posture of each connection: *Certificate verified*, *Certificate verified · custom CA* or *Certificate not verified*.

Warning

With **Verify SSL certificate** off, traffic stays encrypted, but TrackMe does not check the Leader's identity: a host impersonating the Leader would receive the connection's credentials. To keep verification on with a private CA, add the Leader's CA chain. TrackMe raises no alert for this choice.

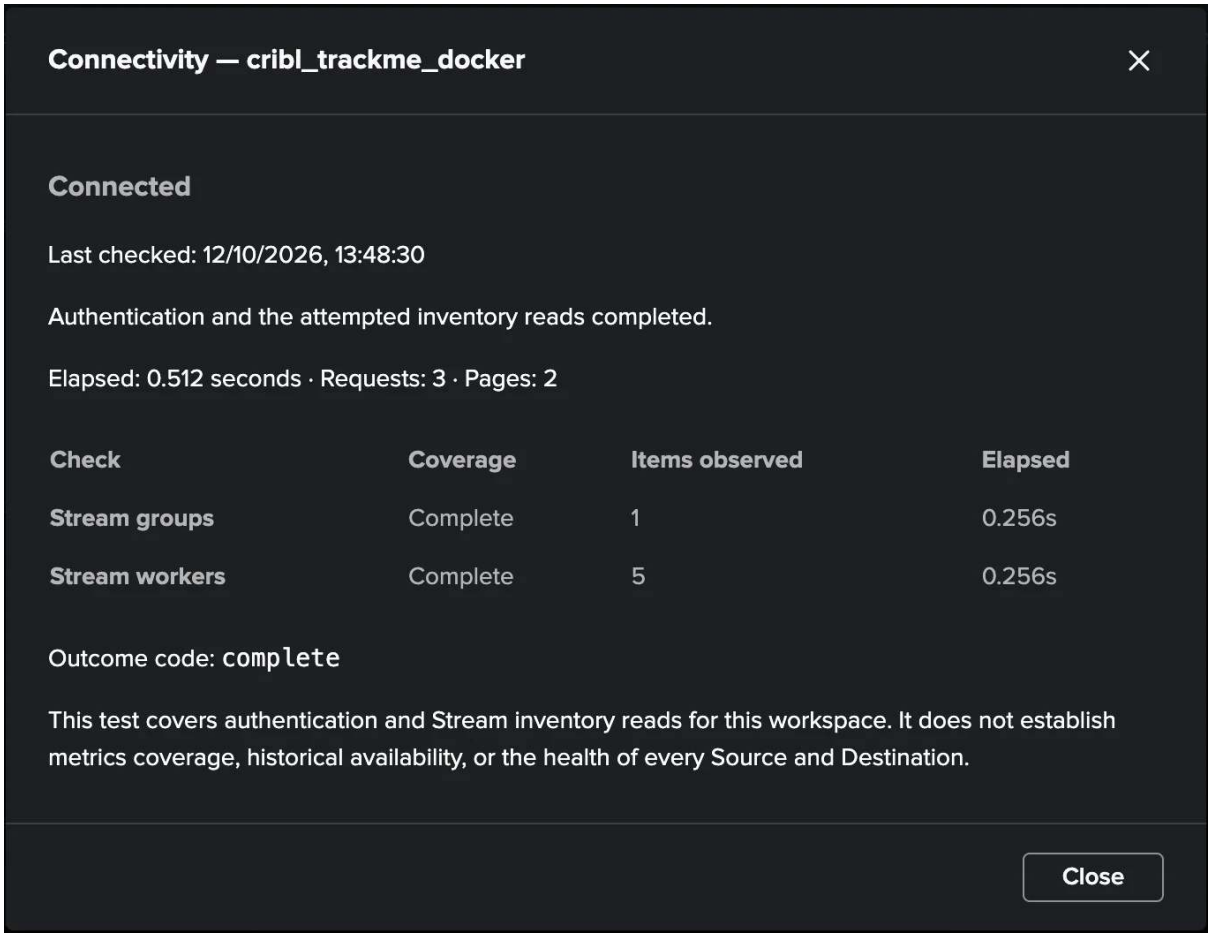
Leader request budget

Setting	Default	Range	Meaning
Requests per hour	1 200	120 – 20 000	How many requests per hour TrackMe may send to this Leader, shared by every tenant and job that reads it.
Burst	150	10 – 2 000	The most requests TrackMe may send at once after a quiet period.

When several connections reach the same Leader (a Cribl Stream and a Cribl Edge tenant on the same Cribl.Cloud workspace, for example), they share one budget and the **lowest** values apply. Raise the budget when a tenant's Collection health reports that the inventory or the Worker sampling cannot keep up.

Testing connections

- **Save and test** stores the connection, then signs in to Cribl and reads the Worker Groups and the Workers. **Save** alone stores it without a test.
- **Test connectivity** (row menu) tests one saved connection; **Test all** tests every enabled connection one after the other (the first 20 when there are more).
- The **Connectivity** column shows the last result (*Connected*, *Partial discovery*, *Test failed*, *Not tested*, or *Test again* when the connection changed since). **View result** explains a failure and what to do next.
- A connection can be **disabled**: it is kept, but cannot be tested or used.



Who can manage connections

Managing connections (create, edit, test, delete) is a **Splunk administrator** task: the connection configuration and its encrypted credential are restricted to the `admin` and `sc_admin` roles, and TrackMe also checks its own administrator capability (`trackmeadminoperations`), which those roles hold.

Nobody else needs access to the connections. A tenant's users never read the credential: when they act in the workspace, TrackMe checks their tenant role first, then uses the connection on their behalf.

What the Cribl account needs

Read access to the Cribl API of the product the tenant monitors:

- the Worker Groups or Fleets and the Workers or Edge Nodes (`/api/v1/products/stream/…` or `/api/v1/products/edge/…`);
- the configuration of each Worker Group or Fleet (Sources and Destinations, plus Pipelines, Routes and Packs on Stream);
- the Leader's system information;
- the Leader-proxied reads of a Worker or Edge Node: its system information and the status of its Sources and Destinations, and on Edge its own system metrics.

TrackMe never asks for write access and never changes anything in Cribl. TrackMe does not prescribe a specific Cribl role: give the account the most restricted role of your Cribl deployment that can read these. A `forbidden` test result means the account cannot read part of them.

Licence

Cribl connections and the Cribl Stream and Cribl Edge tenants need a valid **Enterprise**, **Unlimited**, trial or **Developer** licence. They are not available with the Foundation edition or the Foundation trial. Without a valid licence the connections can still be listed, but not created, changed or tested.

Configuration Guardian

A system check watches the connections. It is shown to administrators only; the connections page links to it (*Cribl connection alerts in Configuration Guardian*):

Check	What it reports
`cribl_connection_health`	A connection whose daily connectivity test fails, is incomplete or has no recent result. A warning at first, critical when the failure persists. Cleared by the next complete test.

Each tenant has its own collection check as well (`cribl_collection_health`), described in its pages. See *Configuration Guardian*.

12.2.3 Choosing your tenants

- **One connection per tenant.** A tenant reads one Cribl connection, chosen at creation; changing it later starts a new source. A connection can serve several tenants.
- **Stream or Edge, never both.** A tenant is a Cribl Stream tenant or a Cribl Edge tenant. A Cribl.Cloud workspace that runs both products takes one tenant of each, on the same connection.
- **Several deployments, several tenants.** One tenant per Cribl Stream deployment or Cribl Edge Leader you monitor.
- **Size.** A Cribl Stream tenant monitors up to about **20 000 objects** (Sources, Destinations, Pipelines, Routes and Worker processes): above that, split the deployment across tenants by Worker Group. A Cribl Edge tenant follows a fleet of **any size**.

Tenants of the same Leader share its request budget: Collection health tells you when it needs raising.

12.2.4 Permissions

Daily operation of a Cribl tenant needs **no Splunk administrator role**. TrackMe decides on its own capabilities and the tenant's roles, as for every tenant (*Roles & access control*):

Who	What they can do in a Cribl tenant
Tenant user role (`trackmeuseroperation`	Read every tab of the workspace; ask for the live status of an object.
Tenant power role (`trackmepoweroperatio`	Triage findings and Finding groups (investigate, assign, snooze, dismiss, resolve, change the severity, group by hand), write notes, turn monitoring of an object on or off, *Stop monitoring this object*; ask the AI Agent and let it act; on Cribl Stream, *Not an anomaly*.
Tenant admin role (`trackmeadminoperatio` "a TrackMe administrator of this tenant"	Everything above, plus policies, the source settings, *Collect now* and *Run now*, notifications, assignment and findings actions settings, the AI Agent settings, *Forget this object*, and on Cribl Stream the traffic ML models and Expected traffic.
Splunk administrator (`admin`, `sc_admin`)	Every tenant, and the Cribl connections.

Each tier needs the TrackMe capability **and** one of the tenant's matching roles: a TrackMe administrator of one tenant has no right on another. The tenant **owner** (the account its reports run as) is `nobody` or an account that holds the capabilities the creation wizard lists.

12.2.5 Where to go next

Cribl Stream Create a Cribl Stream tenant and work in its workspace: findings, monitoring, traffic ML models, infrastructure, deployment, policies.

Cribl Stream Cribl Stream reference Collection jobs, policy catalogue and defaults, settings, capacity and troubleshooting.

Cribl Stream reference Cribl Edge Create a Cribl Edge tenant and follow your Fleets and Edge Nodes.

Cribl Edge Cribl Edge reference The Edge policy catalogue, collection, settings and troubleshooting.

Cribl Edge reference Findings Findings, Finding groups, triage, notifications, assignment and findings actions.

Findings, Finding groups and findings actions Cribl signals for DSM and DHM Monitor the data sources and hosts whose logs pass through Cribl.

Monitor data sources and hosts with Cribl signals Flex Objects and the Cribl API add-on Custom Cribl monitoring built on your own SPL.

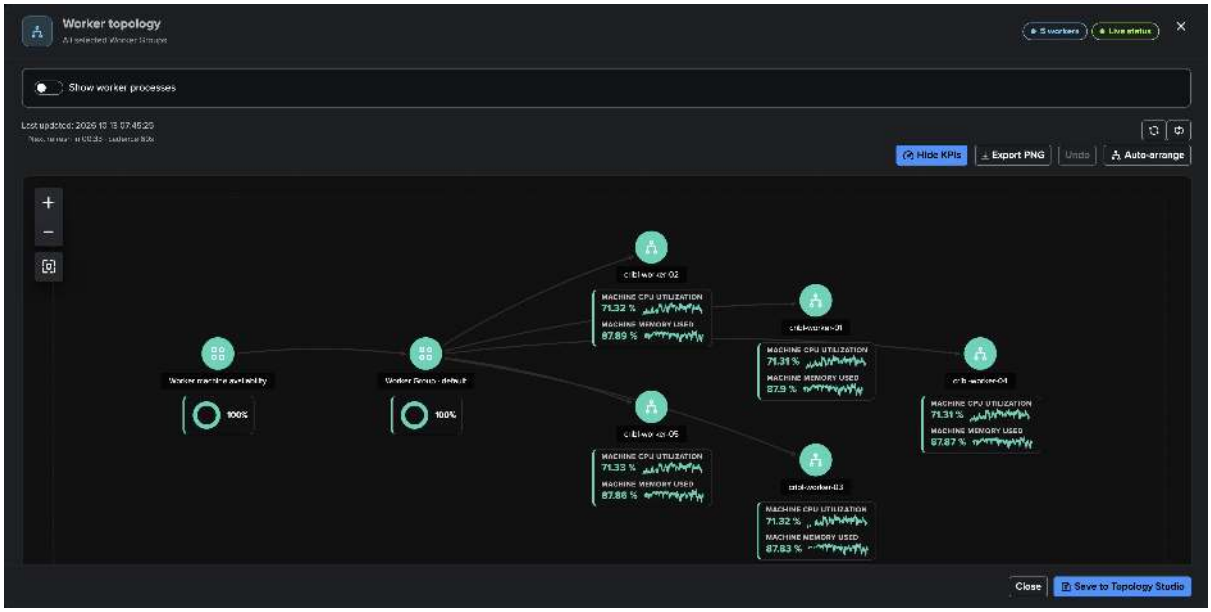
Cribl API add-on and Flex use cases (annex) Hands-on: Cribl Stream A guided tour of a Cribl Stream tenant, step by step.

Cribl Stream Hands-on: Cribl Edge A guided tour of a Cribl Edge tenant, from the Fleets to one node up close.

Cribl Edge

Cribl Stream

Cribl Stream sits in the middle of your data path: every Source, Route, Pipeline and Destination it runs is a place where data can slow down, back up, be dropped or stop. A **Cribl Stream tenant** watches one Cribl Stream deployment end to end and tells you, in a Cribl operator's words, what needs attention: a Destination under backpressure, a Source whose traffic fell outside what it usually sends at this hour, a Pipeline raising errors, a Worker that stopped answering, runs another version than the Leader or an old configuration, a Leader whose disk is filling up.



The view above is the **Worker topology** of a self-managed lab deployment: the Worker Group, its five Workers with their machine KPIs and their live status, drawn from what TrackMe already knows, refreshed in place and ready to be saved to Topology Studio. It is one of the views this page walks you through.

A Cribl Stream tenant reads two things you already have, and changes nothing in Cribl:

- the **Cribl internal metrics** that your Workers send to a Splunk **metric index** (read where they are: nothing is copied, no licence is consumed by the metrics);

- the **Cribl Leader API**, read-only and within a shared request budget, for the configuration (Worker Groups, Sources, Destinations, Pipelines, Routes, Packs), the Workers and the Leader itself.

On top of that, a catalogue of 16 built-in **policies** raises **findings**, **traffic ML models** learn what normal traffic looks like for every object and every hour, and a purpose-built **workspace** gives you the investigation and triage workflow.

A Cribl Stream tenant is a **dedicated tenant type**: it has no entities, no decision maker and no health score, and none of the DSM, DHM, MHM, FLX, FQM, WLK or VOL components. Its objects are Cribl objects and what it raises are findings. It shares its findings model — findings, Finding groups, triage, notifications, assignment, findings actions, AI Agent — with the other dedicated tenant types: the concepts are explained once in *Findings, Finding groups and findings actions*, and this page documents what is specific to Cribl Stream.

Tip

Looking for the catalogue of policies, every setting and its default, the schedules, the REST endpoints or the troubleshooting table? They are in *Cribl Stream reference*. Monitoring Cribl Edge Fleets is a different tenant type: see *Cribl Edge*.

What it monitors

One tenant monitors **one Cribl connection** — a Cribl.Cloud workspace or a self-managed Leader — and the **Worker Groups** you choose in it.

Cribl object	What TrackMe watches
Leader	Whether its API can be reached and used (network, certificate, credentials, service), and the CPU, memory and disk of each Leader node.
Worker Groups	The scope of the tenant: every other object belongs to a Worker Group. The metrics of a Worker Group that sends none are reported in Collection health.
Workers	Connected or not and the age of their heartbeat, their Cribl version against the version of the running Leader, their configuration against their Worker Group's, and the CPU, memory, load and disk of the machine.
Worker processes	CPU and resident memory of every worker process, per Worker.
Sources	Events and bytes in, and the health Cribl reports for each Source.
Destinations	Events and bytes out, dropped events, the health Cribl reports, **backpressure** and the **blocked** state.
Pipelines	Events in and out, dropped events and **errors**.
Routes	Events in and out, tied to the configured Route with its Pipeline and Destination.
Packs	Listed with their Routes, Pipelines, Sources and Destinations; the Routes of a Pack are measured and tied to their Pack.

Every Source, Destination, Pipeline and Route gets its own **traffic ML model**.

Note

What is not monitored. TrackMe does not read the events flowing through Cribl, does not monitor Cribl Edge Fleets from a Stream tenant (that is a *Cribl Edge* tenant), and does not measure filesystems other than the one Cribl reports for each Worker and each Leader node (separate queue, log or shared-storage mounts are not measured). It never changes anything in Cribl: every action you take in TrackMe — dismissing a finding, turning monitoring off for an object — stays in TrackMe.

Before you start

You need four things.

1. A licence that includes Cribl Stream. A Cribl Stream tenant needs the **Enterprise** or **Unlimited** edition (the Unlimited trial and the Developer licence included). It is not available on the Foundation edition or the Foundation

trial: the tenant creation card reads *Restricted component* there.

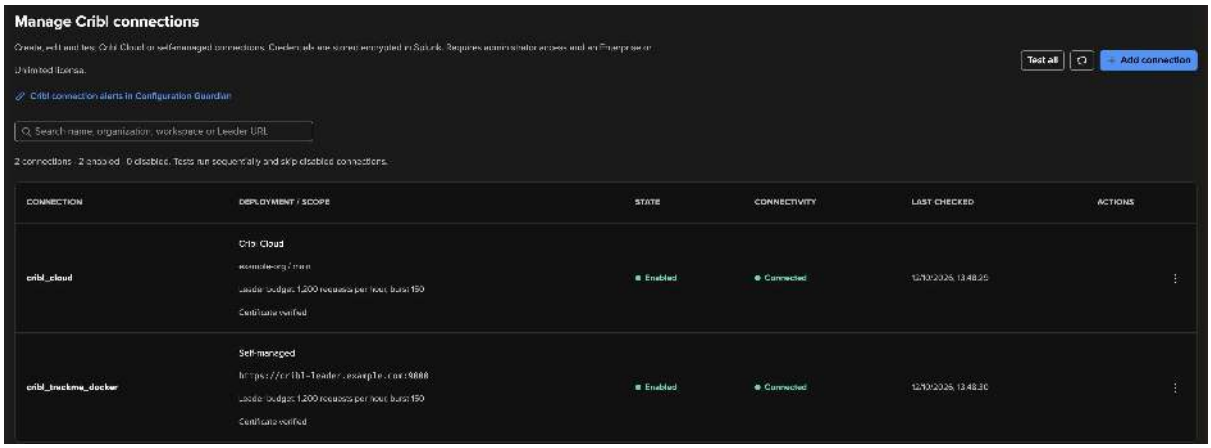
2. A Cribl connection. Declare it once in *Configuration → Cribl connections*: a Cribl.Cloud organization and workspace with an API client ID and secret, or a self-managed Leader URL with a user name and password, plus its TLS settings and the **Leader request budget**. The account only needs **read** access to Cribl Stream's configuration and Workers: TrackMe never writes to Cribl. See *Cribl connections*. A self-managed Leader that this instance cannot reach, such as from Splunk Cloud, can go *through a TrackMe Broker*.

3. Cribl's internal metrics in a Splunk metric index. Cribl Stream produces internal metrics (`cribl.logstream.*` by default) about every Source, Destination, Pipeline, Route and worker process. Send them to a Splunk **metric index** searchable from the TrackMe search head: in Cribl, enable the built-in **Cribl Internal** Source for its metrics and route them to a Splunk Destination that writes into a metric index, on every Worker Group the tenant will monitor. If your deployment renames them, note the **metric prefix**: the wizard asks for it (`cribl.logstream.` by default). TrackMe reads the metrics in place.

Tip

You do not need to know the index name in advance: the wizard looks at your metric indexes, finds the ones where **this connection's Workers** report, and proposes them with a *Use this index* button.

4. The right roles. Every role of the tenant reads the workspace; the tenant's power and admin roles triage findings; *a TrackMe administrator of this tenant* — an account holding the TrackMe administrator capability and one of the tenant's admin roles — edits policies and settings and runs jobs. Splunk administrators can do everything on every tenant. The full table is in *Cribl Stream reference*.



Create the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Cribl Stream** card. The wizard has four steps.

Step 1 — Basics

The **tenant identifier** (lowercase letters, digits and hyphens, up to 20 characters; it cannot change later), an optional **alias** shown on the tenant card — for example *Cribl Stream - Cloud* — and an optional description.

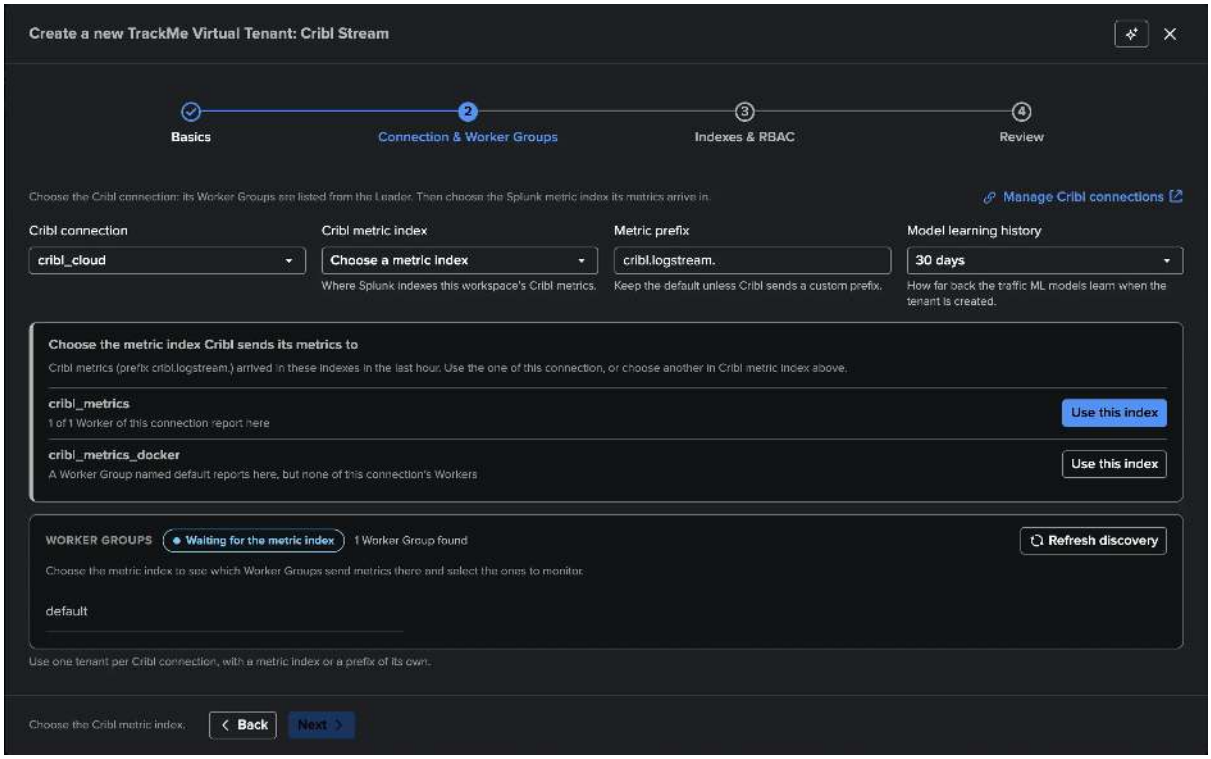
Step 2 — Connection & Worker Groups

This is where TrackMe meets your Cribl deployment:

1. **Cribl connection** — choose the connection. Its **Worker Groups** are listed from the Leader at once. If the Leader refuses the credentials or asks TrackMe to slow down, the wizard says what to do next and offers **Refresh discovery**.

2. **Cribl metric index** — under *Indexes with Cribl metrics*, TrackMe lists the metric indexes where this connection's Workers report (up to five), each with **Use this index**. It warns you when an index also carries Worker Groups of the same name from another deployment, so two deployments never mix.
3. **Metric prefix** — `cribl.logstream.` unless your Cribl deployment renames its metrics.
4. **Model learning history** — how far back the traffic ML models learn when the tenant is created: *From today on*, 7, 14, **30** (default), 60 or 90 days. With history already in the metric index, the models can detect on the first day; with *From today on*, they need about a week before they open findings.
5. **Worker Groups** — all are selected; keep the ones this tenant should monitor (1 to 100).

A disabled **Next** always says why (a connection to choose, discovery still running, no Worker Group selected).



Tip

Use **one tenant per Cribl connection**. A very large deployment (more than about 20 000 monitored objects) is split across several tenants by Worker Group.

Step 3 — Indexes & RBAC

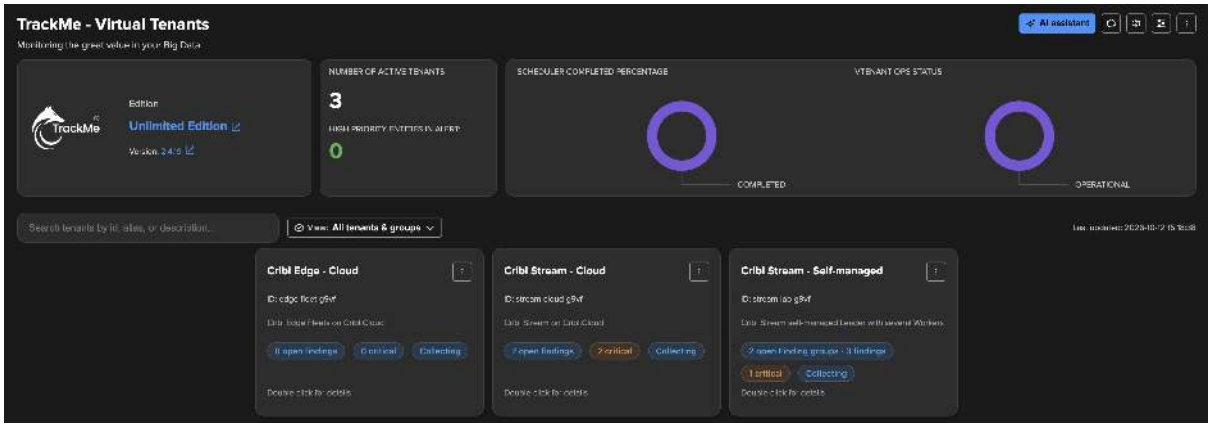
The TrackMe indexes of the tenant (summary, audit, metric, notable — the notable index receives the finding events), the **tenant owner** (`nobody` or an account holding the capabilities the wizard lists) and the admin, power and user roles of the tenant.

Step 4 — Review

Check the summary and create. The first collection starts at once: the live metrics, the inventory of the Leader and the model training all run within minutes, so the workspace fills without waiting for the schedules.

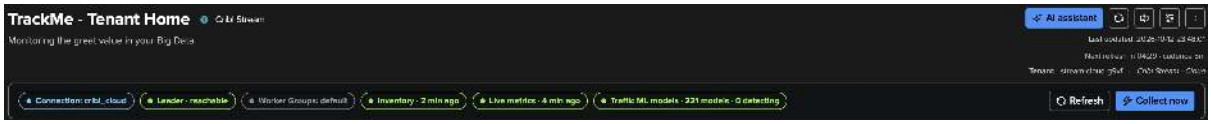
The workspace

A Cribl Stream tenant appears on the Virtual Tenants page with the Cribl Stream logo, its unresolved and critical findings and its collection status. Selecting it opens its **workspace**.



The status strip. At the top of the workspace, a strip tells you at a glance that the tenant is working:

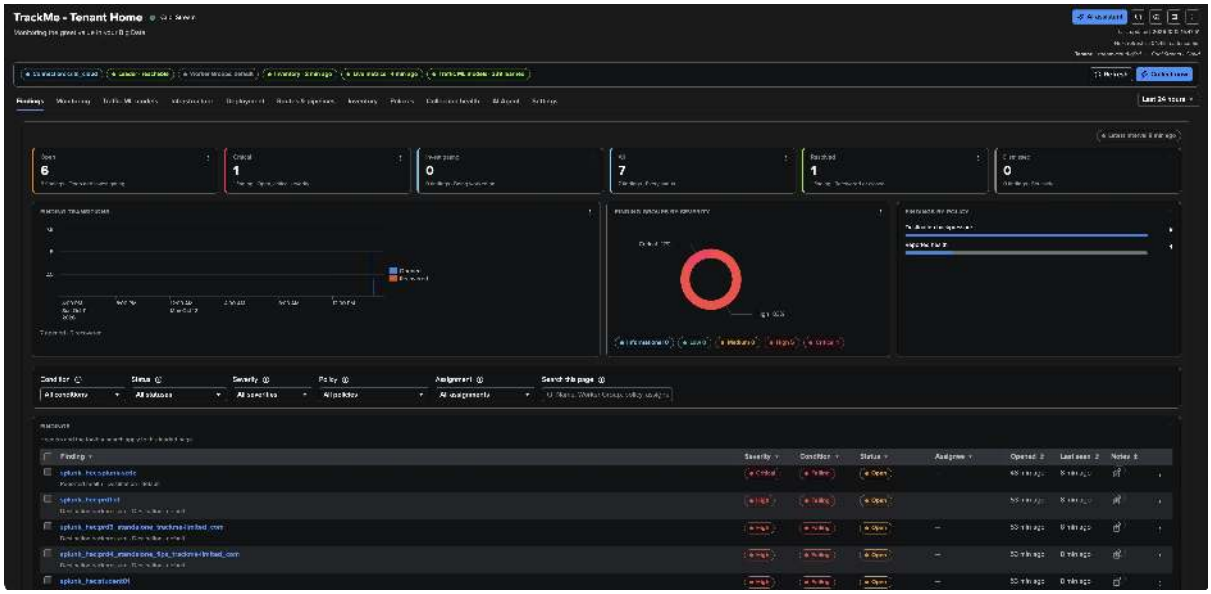
- **Connection** — the Cribl connection this tenant monitors;
- **Leader** — *reachable*, *unavailable*, *rate limited* or *unknown*, from the last Leader API check (hover for the reason and when it was checked);
- **Worker Groups** — the Worker Groups monitored;
- **Inventory**, **Live metrics** and **Traffic ML models** — the last run of each job, and how many models are learned;
- **Refresh**, and for a TrackMe administrator of the tenant **Collect now** (read the newest metrics and evaluate the policies now).



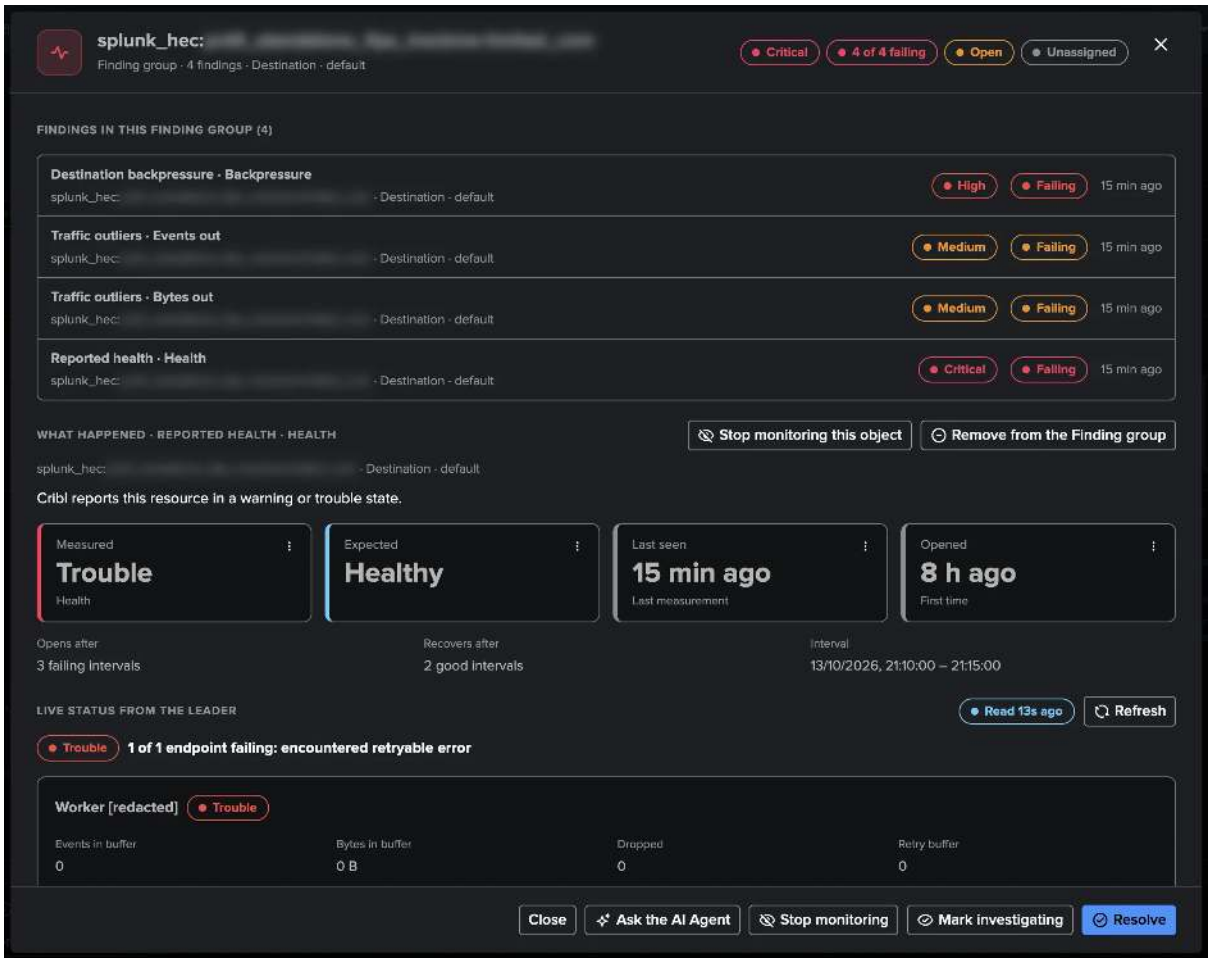
Eleven tabs follow, in this order: **Findings**, **Monitoring**, **Traffic ML models**, **Infrastructure**, **Deployment**, **Routes & pipelines**, **Inventory**, **Policies**, **Collection health**, **AI Agent** and **Settings**. Every table is sorted, filterable from its column headers and paged on the server, so it stays fast at any size. Every figure has a ? **menu** with *View chart data*, *Open in Search* (a plain, editable search on your metric index or on the tenant's data that reproduces the figure) and *Refresh*.

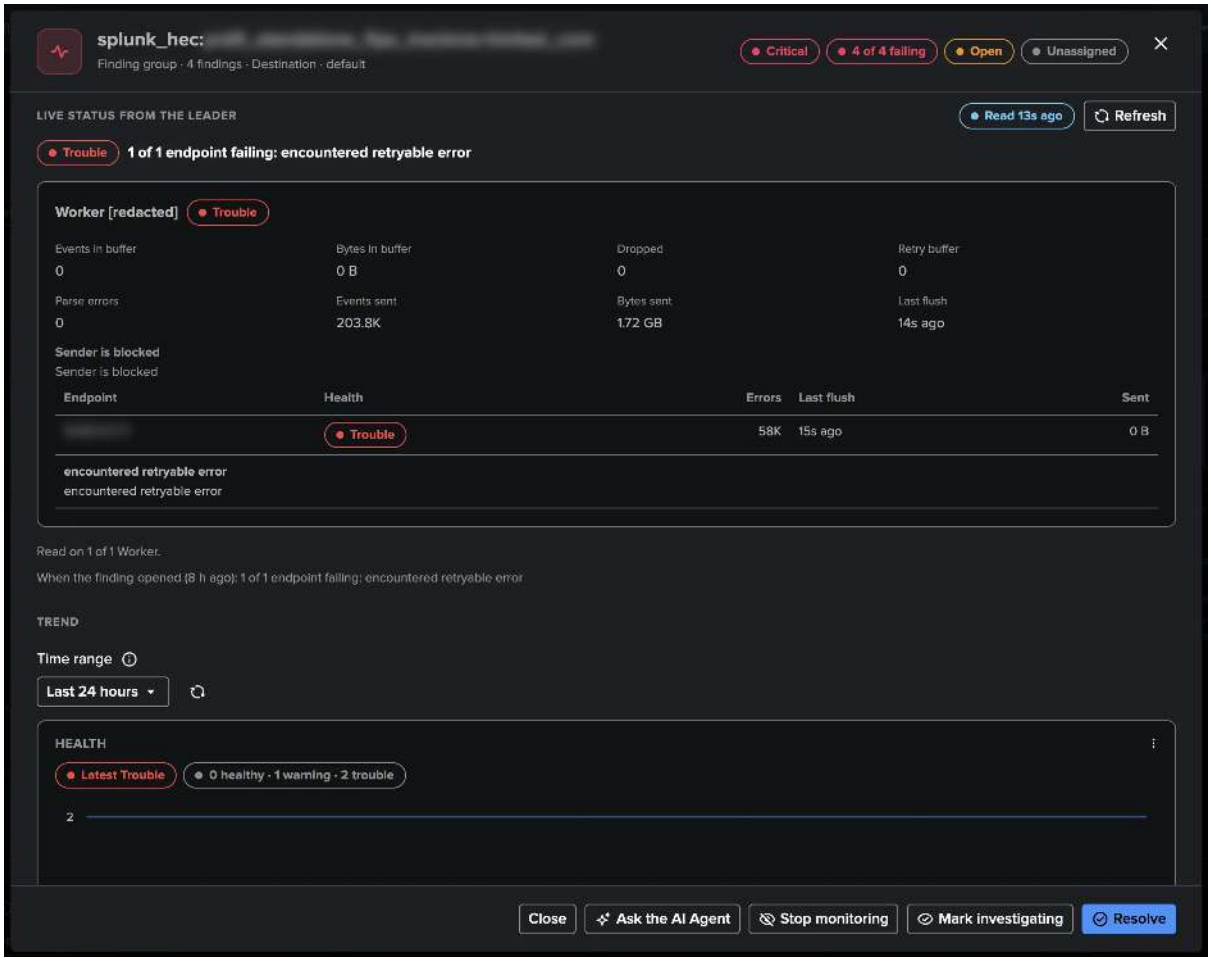
Findings

Your investigation backlog, and the tab the workspace opens on. Tiles count what needs attention, charts show the finding transitions over the selected time range and the open findings by severity and by policy, and the table lists one row per **Finding group** — usually one object with all its findings — with its condition, status, severity, policy, assignee and last measurement. Filters cover the condition, status, severity, policy and assignment, and a search box finds an object, a Worker Group or an assignee.



Selecting a row opens the **investigation drawer**: what happened in plain words, the chart of the measurement with its expected range (or the policy's threshold and recovery lines for a Worker or the Leader), the live status the Leader reports for a Source or Destination, the object's state in the Inventory, the notes and the full history. A Finding group lists its findings first: select one to see what happened and its measurement, and the triage below applies to the whole group.



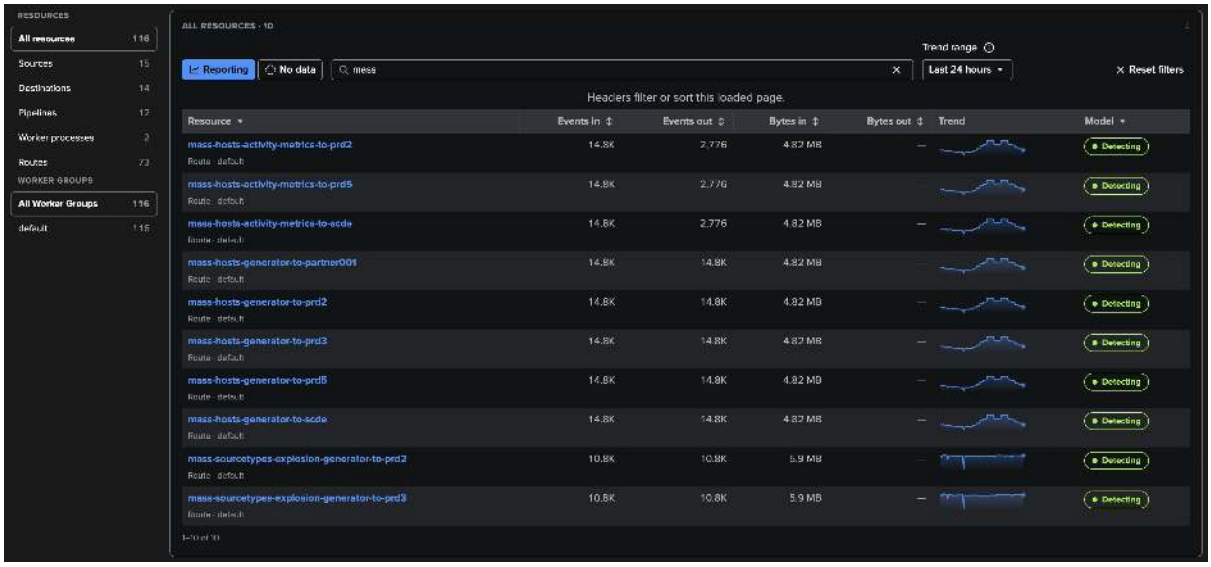


Monitoring

The newest five-minute interval of every object. Tiles summarise the deployment (events and bytes in and out, dropped events, Pipeline errors, process CPU and memory), **Traffic over time** charts the flow over the selected range, and the resource tables list each Source, Destination, Pipeline, Route and worker process with its latest values, its health, its traffic ML model and a trend. A row opens the **resource details**: its charts with the expected range, the measurements to show, the way into its traffic ML model and, for a Source or Destination, its live status from the Leader.

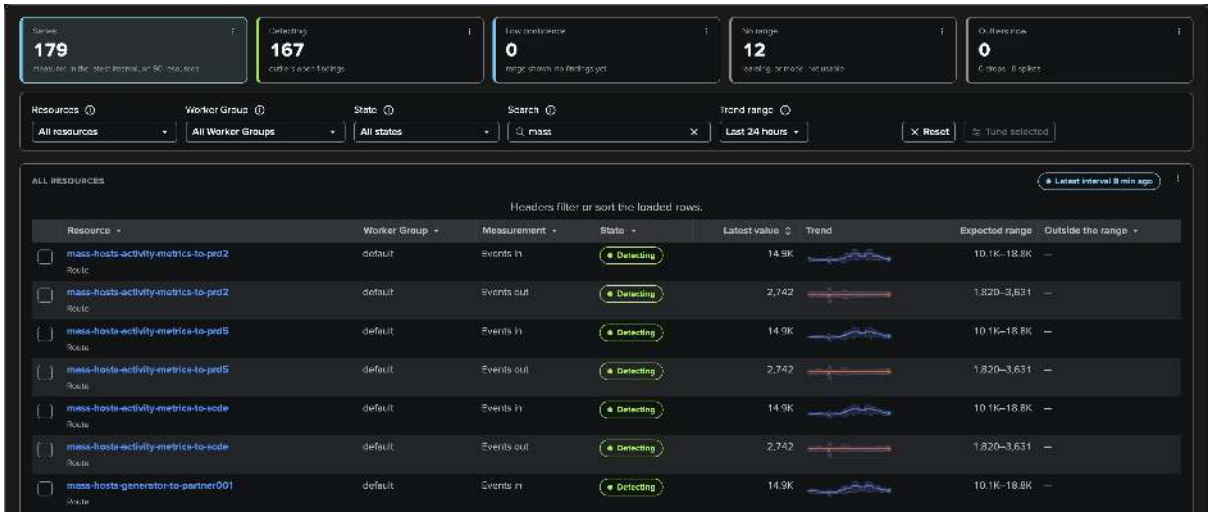
A **Reporting / No data** switch shows either the objects measured in the latest interval or the objects that **stopped reporting** (see *Objects that stop reporting*).





Traffic ML models

One row per model — an object and one of its traffic measurements — the models outside their expected range first. Each row shows the model's state, the latest value, a trend with the expected range as a band and *Drop* or *Spike* when the value is outside it. A row opens the model: what it learned, its drops and spikes, its expected range, the history learned and its seasonality, with **Tune model** and **Relearn** for a TrackMe administrator of the tenant. See *Traffic ML models and Expected traffic*.

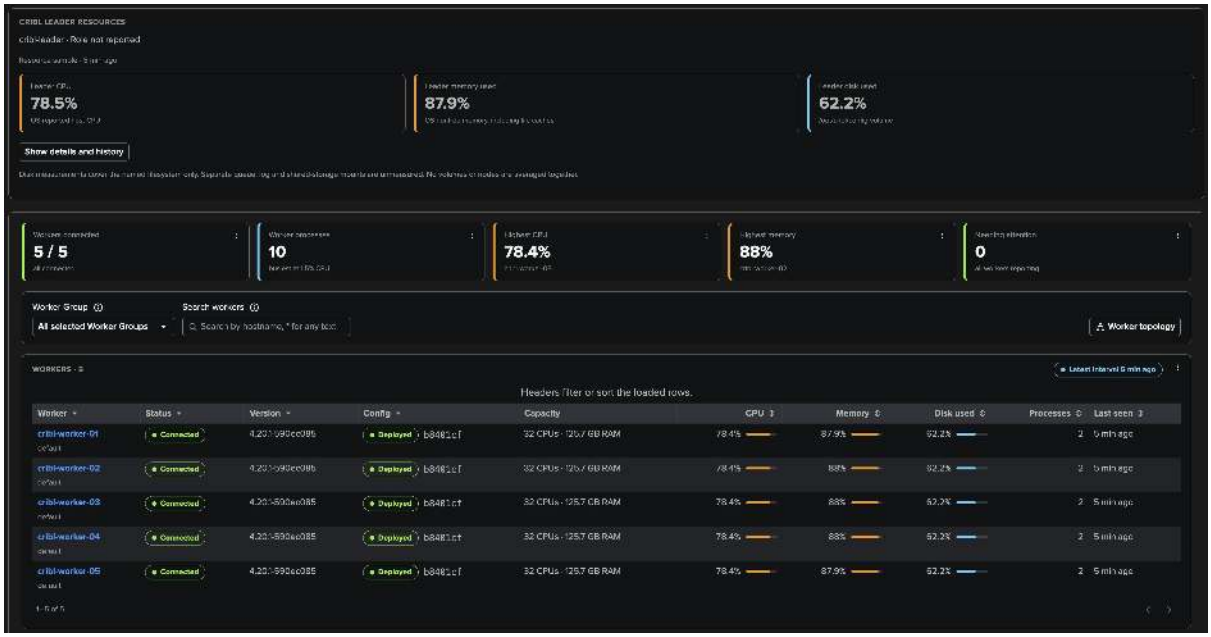


Infrastructure

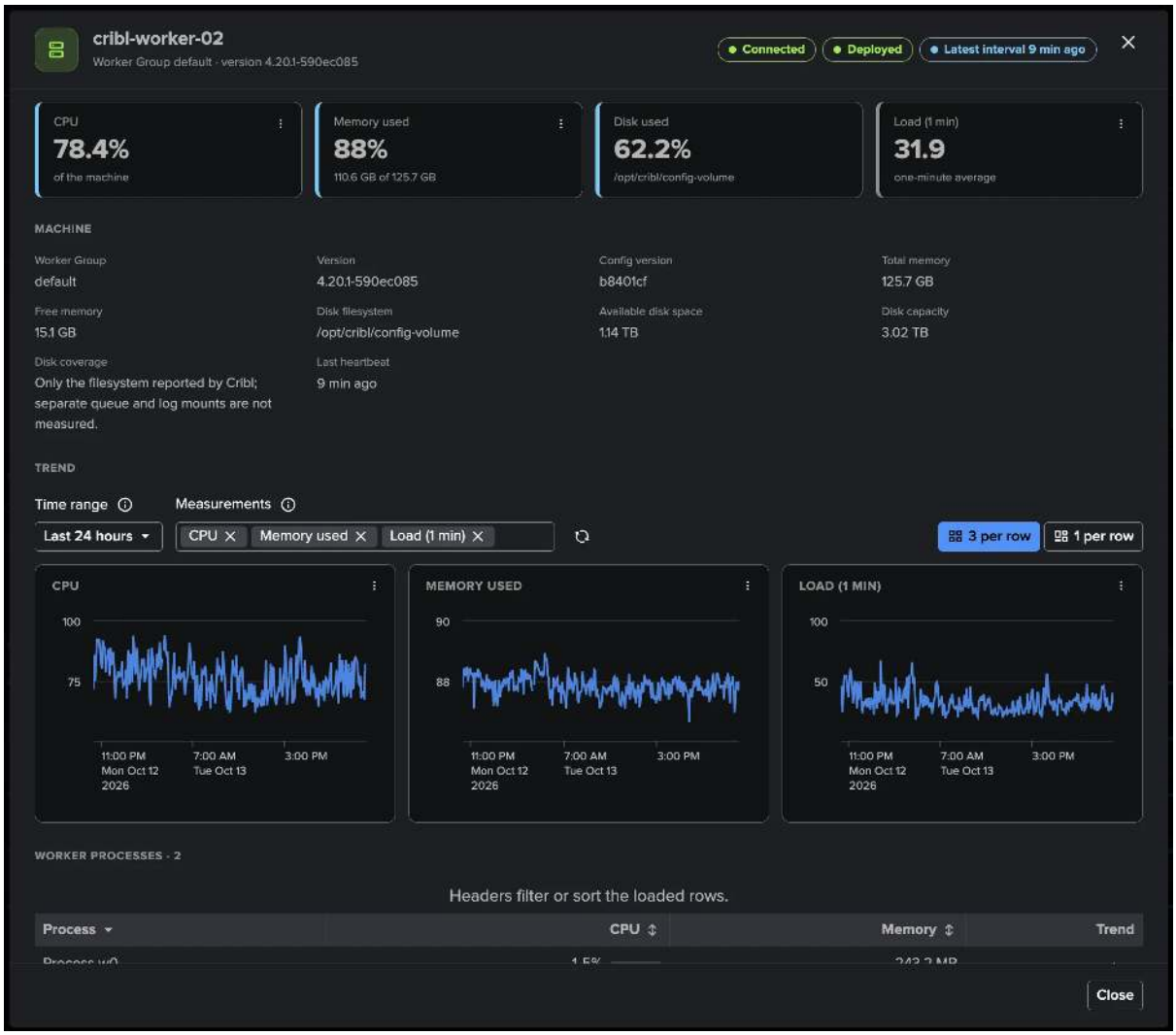
The machines behind Cribl Stream:

- **Cribl Leader resources** — each Leader node with its role, version, CPU, memory, load and the disk of its named filesystem, with their history;
- **Workers** — every Worker the Leader lists: connected or not, last heartbeat, CPU, memory, load and disk, with a trend and the machine's capacity; a Worker opens its machine charts and its processes;
- **Worker processes** — CPU and resident memory per worker process, and the memory used by all worker processes.

The **Worker topology** and **Process topology** buttons draw these machines as a live map (see *Worker topology*).

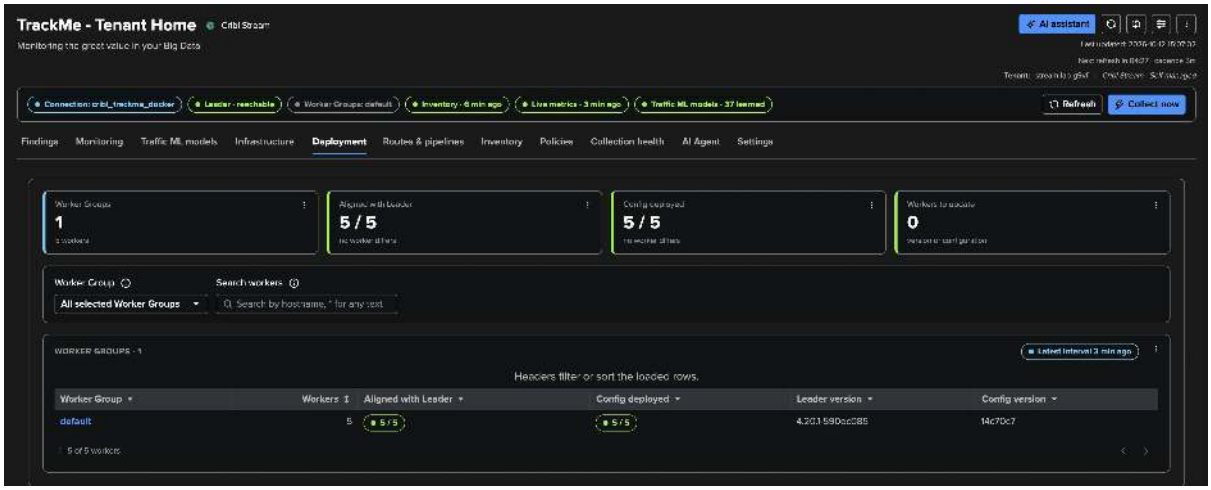


A Worker opens its own view: its CPU, memory, disk and load, the machine (Worker Group, version, configuration version, memory and disk capacity, last heartbeat) and its trend over the selected range, with the measurements to chart and one or three charts per row, then its worker processes.

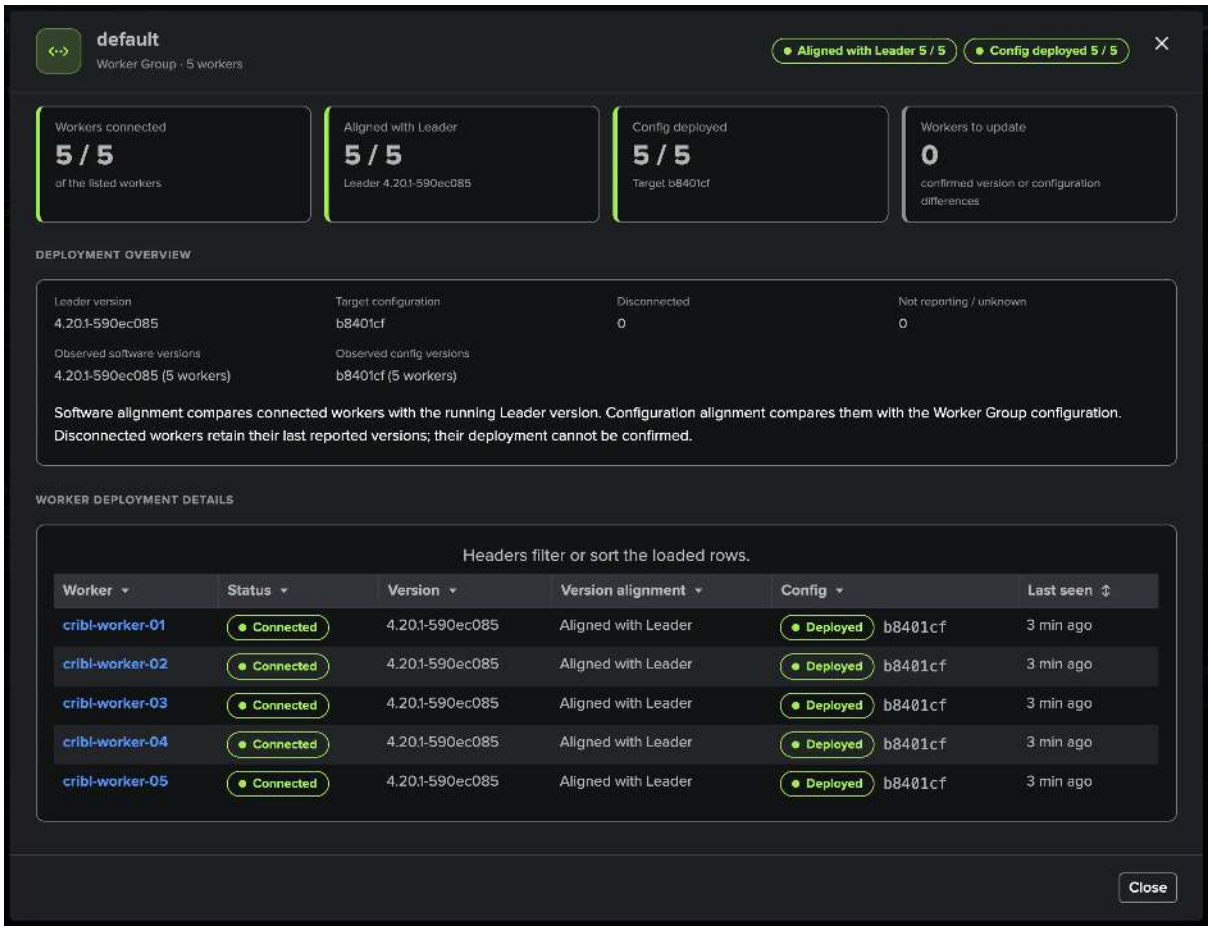


Deployment

Per Worker Group, how far your Workers are from where they should be: the **Cribl version** of each Worker against the version of the running Leader, and the **configuration** each Worker runs against its Worker Group's configuration, with the Workers to update.

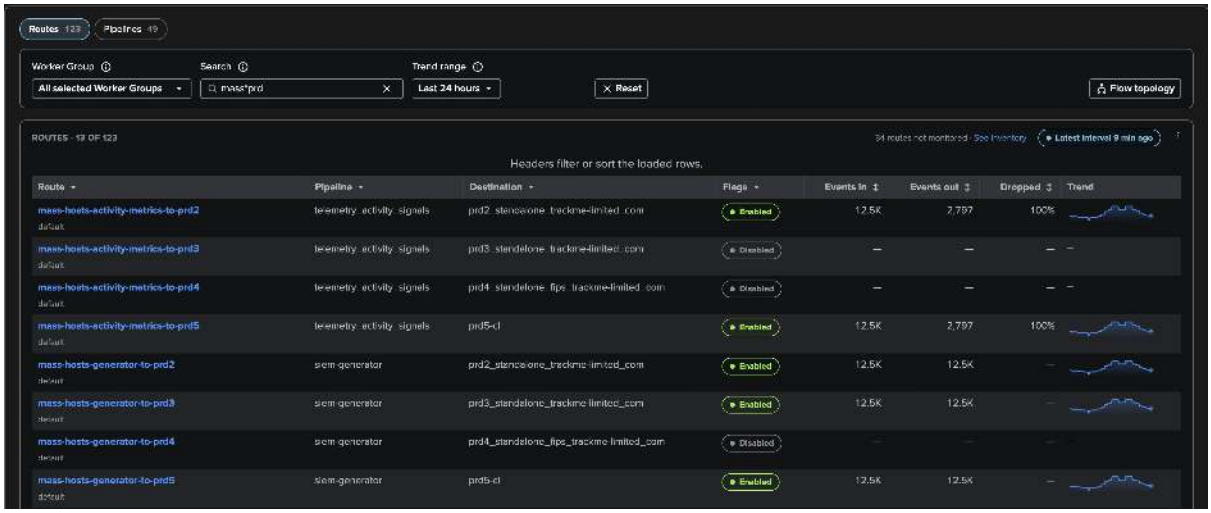


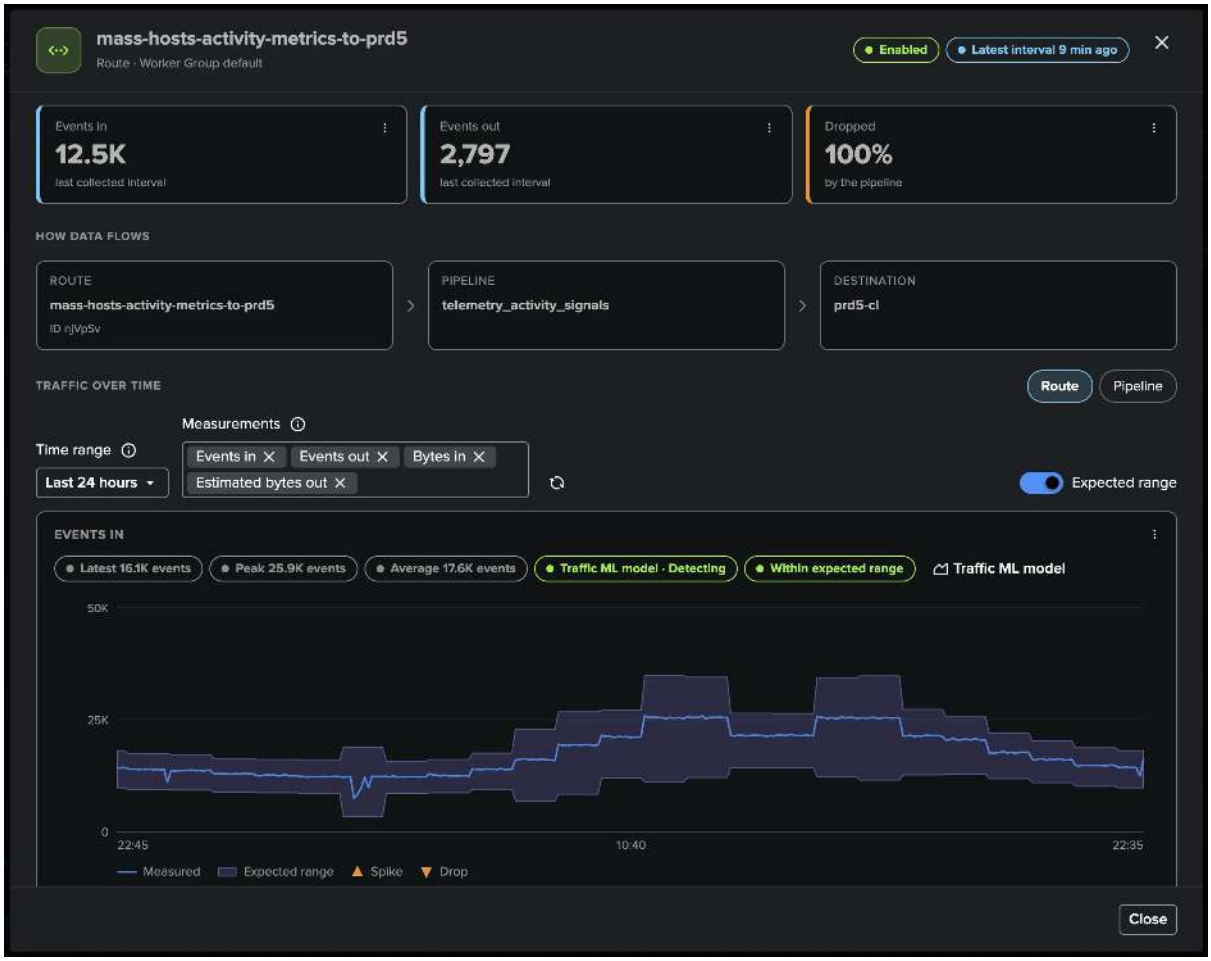
A Worker Group opens its deployment view: the Workers connected, aligned with the Leader's version and running the deployed configuration, the Workers to update, the versions and configurations observed, and each Worker's version alignment and configuration.



Routes & pipelines

How data flows. The **Routes** table lists each Route with its Pipeline and Destination, whether it is final, and its measured events in and out, dropped events and trend; the **Pipelines** table lists each Pipeline with its functions, its traffic, errors and drops. Filter by Worker Group or search by name (* stands for any text). A Route or Pipeline opens **Traffic over time** with its **configured processing path** beside it.

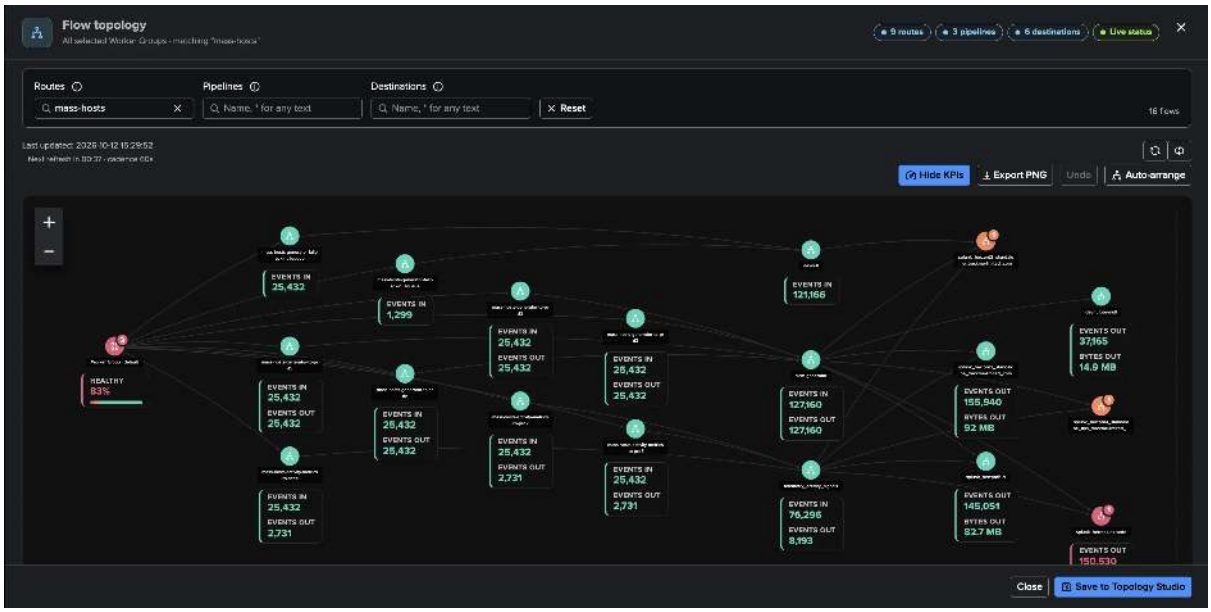




Flow topology: your data paths, drawn for you. Select **Flow topology** on this tab and TrackMe draws the Routes the table selects as flows, left to right: **Worker Group → Route → Pipeline → Destination**. You configure nothing: TrackMe reads each Route's Pipeline and Destination from the Leader configuration and joins them with the traffic it measures, so every Route is connected to where its data goes, and every node carries its live KPIs — events in and out, bytes out, the health Cribl reports — and its findings. A Destination that several Routes feed shows once, with

all its incoming flows, so you see at a glance which Routes depend on a Destination under backpressure. Nothing is invented: a link is drawn only when both of its ends are measured. While the topology is open, **Live status** refreshes the states and KPIs every minute.

The topology opens with the search of the table and has its own **Routes**, **Pipelines** and **Destinations** filters (* stands for any text) to narrow a large deployment to one group of flows. **Export PNG** saves the picture; **Save to Topology Studio** keeps it as a view that stays live, with the same colours as the workspace.

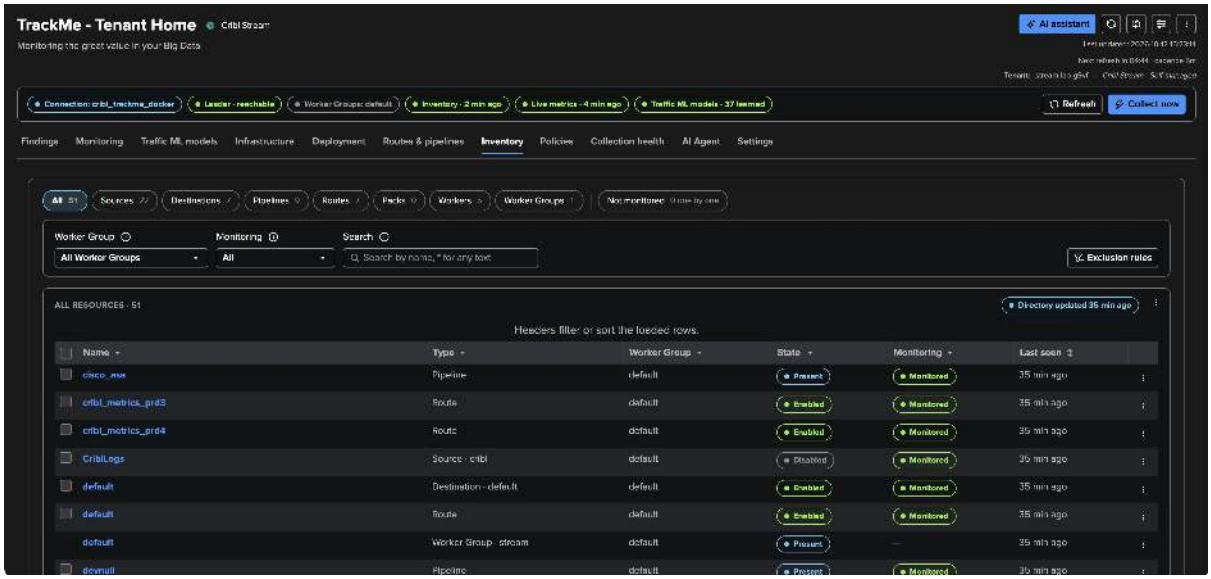


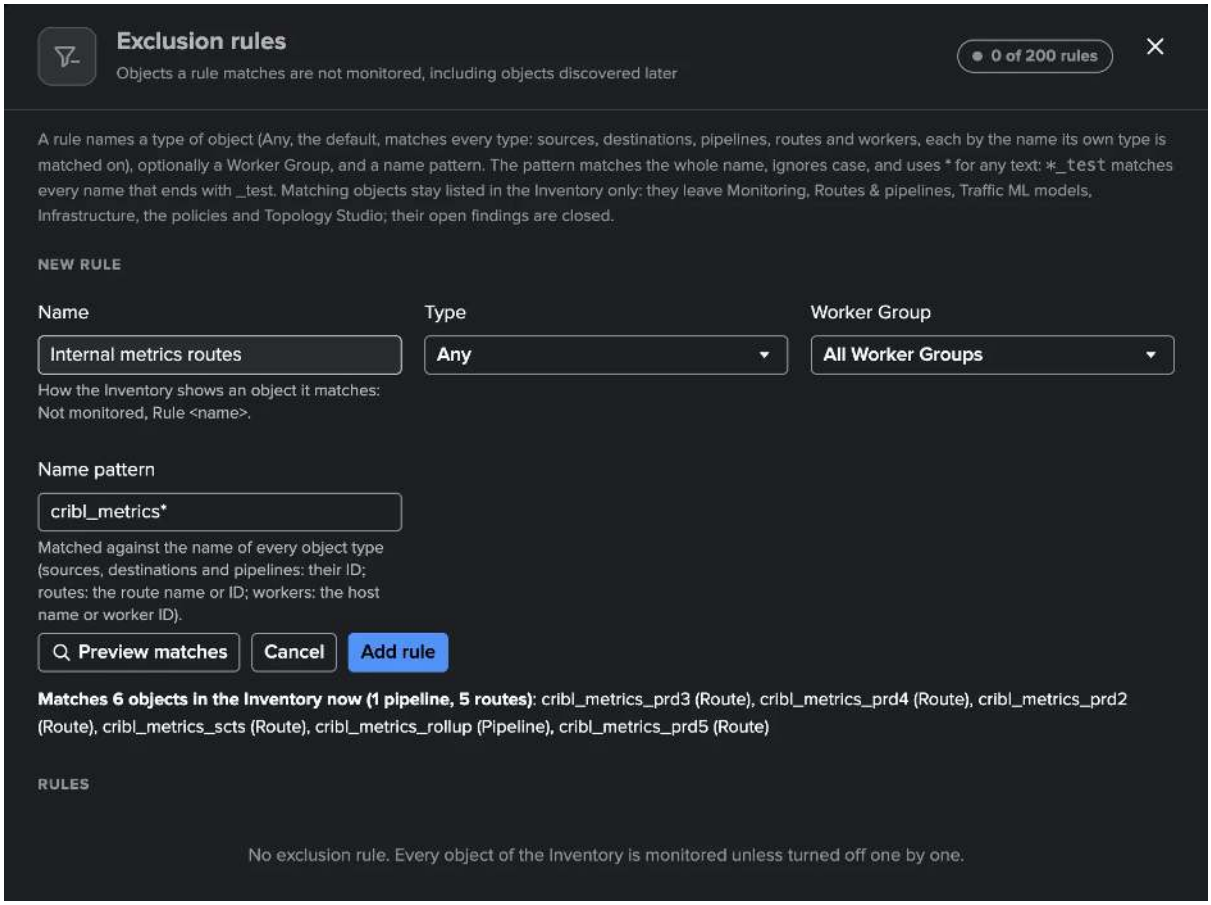
Tip

The flow topology covers up to 200 Routes at a time. On a larger deployment, choose a Worker Group or search first, or use the filters of the topology.

Inventory

Every object the Leader reports, per type (*All*, Sources, Destinations, Pipelines, Routes, Packs, Workers, Worker Groups, *Not monitored*) and Worker Group, with its configuration, when it was first and last seen, whether it is still present in Cribl, and its **monitoring state**. This is where you turn monitoring off and back on for objects you do not care about (a test Route, a retired Destination, a lab Worker), one by one or with **Exclusion rules**, and where a TrackMe administrator of the tenant can **Forget this object** for an object removed from Cribl.

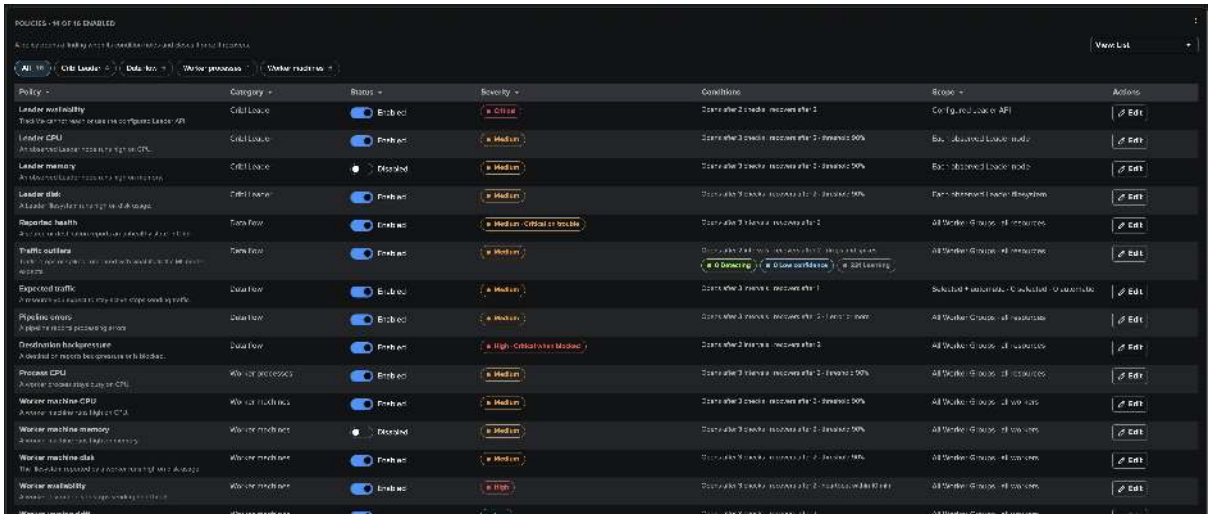




An object turned off appears only in the Inventory (*Disabled by <user> on <date>* or *Disabled by rule <name>*): it leaves the other tabs, the policies, the model training and Topology Studio, and its open findings close.

Policies

The 16 built-in policies in four groups — **Cribl Leader**, **Data flow**, **Worker processes** and **Worker machines** — shown as cards, a compact list or a table. Each policy has its enablement, its severity, how many consecutive checks open and recover a finding, its thresholds, and its scope (Worker Groups and selected objects), with a **preview** of what it would select. The editor has three tabs: **General**, **Actions** (drilldowns, links and webhooks offered on its findings) and **Automations** (rules that run an action automatically). A saved change applies from the next collection interval.



The full catalogue is in *Policy catalogue*.

Custom policies

When no built-in policy covers a rule of your own, write it as an SPL search on the tenant's Splunk data: **Policies → Custom policies → Create policy**. Only the tenant's TrackMe administrators see and edit custom policies, because they hold SPL; other roles see a one-line note.

Each result row names a Cribl object, and each object named opens a finding of the policy. That finding behaves like a built-in one:

- it joins the Finding group of its object;
- it is triaged, noted and assigned the same way;
- it is notified, offered findings actions and reviewed by the AI Agent.



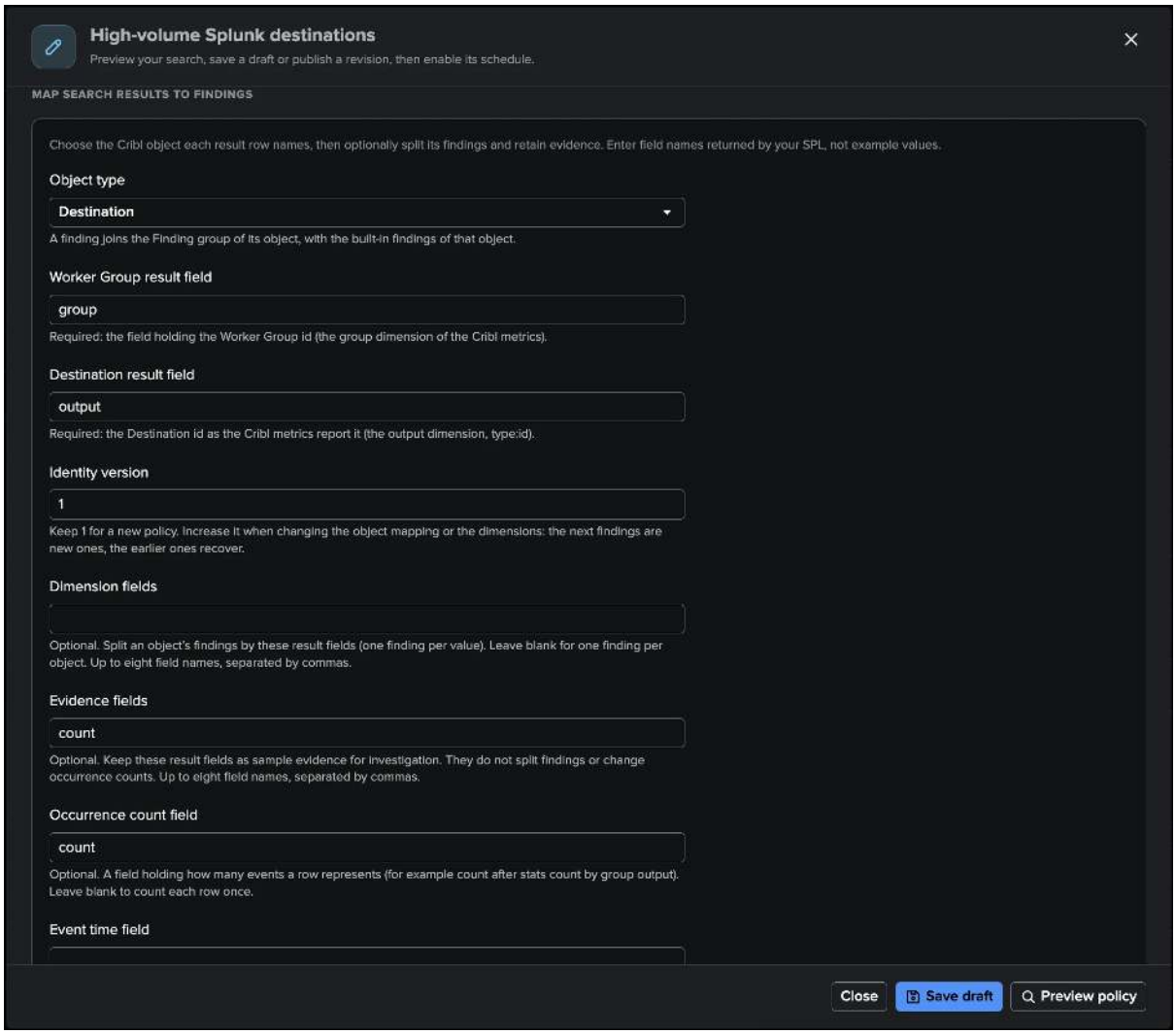
The editor has six parts:

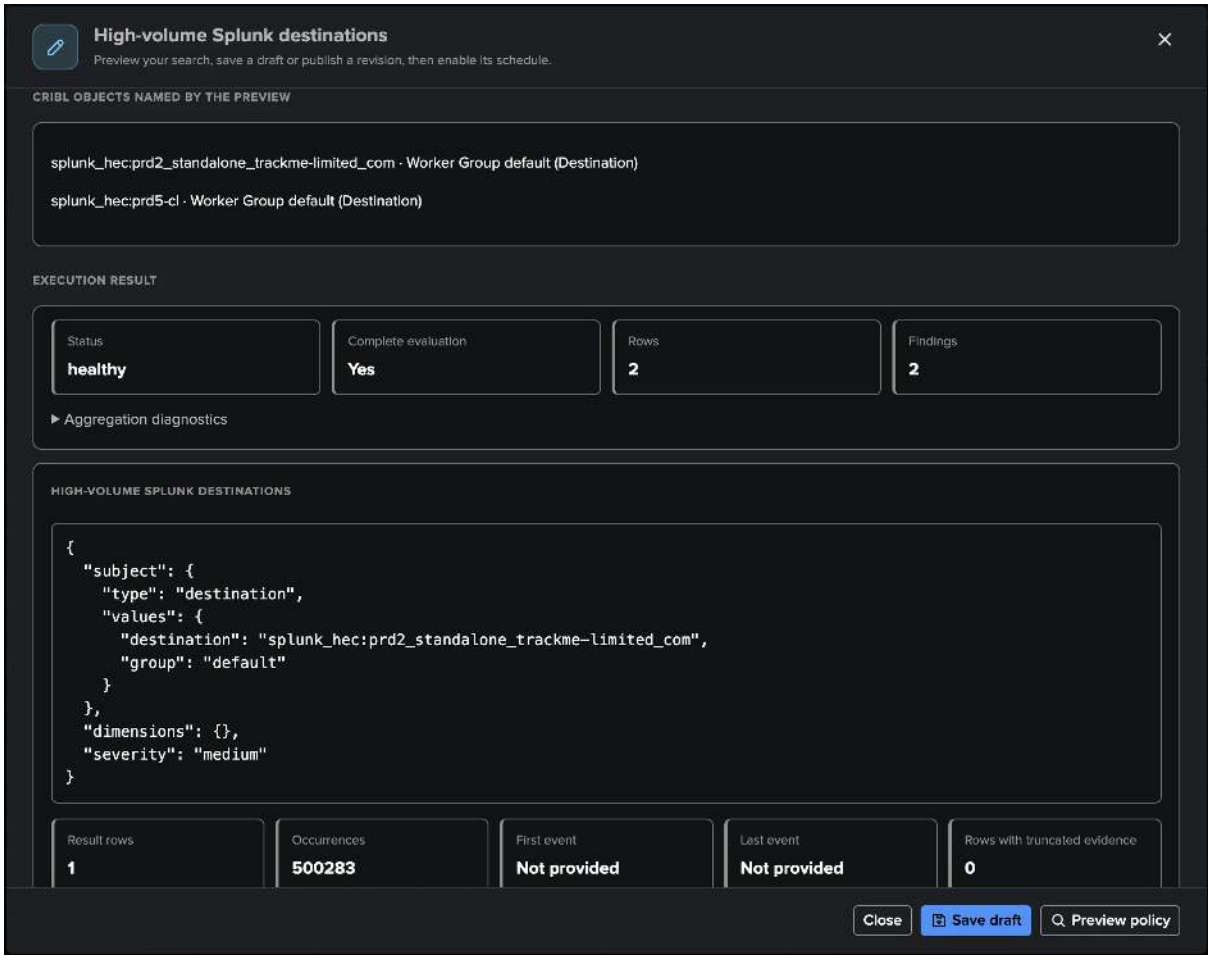
1. **Definition**: title, description, default severity and remediation.
2. **Search and schedule**: the app context, the SPL, a relative window ending now (at most 31 days) and a five-field cron. The search reads Splunk data only, for example the Cribl metrics in the tenant's metric index; it never calls the Cribl Leader.
3. **Findings**: how many runs that name an object open its finding, how many complete runs that no longer name it recover it, and **During bank holidays** (**Keep opening findings** by default, or **Hold new findings**).
4. **Map search results to findings**: the object type and the result fields that name it.

Object type	Result fields
Source, Destination, Pipeline, Route	The Worker Group and the object (for a Source or a Destination, the `input` or `output` dimension of the Cribl metrics, `type:id`)
Worker Group	The Worker Group
Worker machine	The Worker, by its Worker id or its host name
Worker process	The Worker Group, the host and the process (`cribl_wp`)

Optional fields split an object's findings by dimension (one finding per value), keep sample evidence, and give an occurrence count or an event time.

5. **Preview policy**: runs the draft as you and shows up to 20 candidate findings, each with the Cribl object it resolves to. A row that names no object of the inventory is reported as unresolved.
6. **Lifecycle**:
    - **Save draft** keeps your changes without touching what runs;
    - **Publish** creates an immutable revision;
    - **Enable** schedules the published revision as its own report, which runs as the tenant owner;
    - **Version history** loads, compares, restores or deletes a version;
    - **Disable**, **Archive** and **Delete permanently** stop the policy and close its open findings; the findings, their evidence and notes remain.





For example, this search opens a finding on every Destination that dropped events in the last 15 minutes. Map it as a **Destination**, with `group` as the Worker Group field, `output` as the Destination field and `count` as the occurrence count:

```
| mstats sum(cribl.logstream.total.dropped_events) as count
  where index=cribl_metrics by group output
| where count > 0
```

How a run decides.

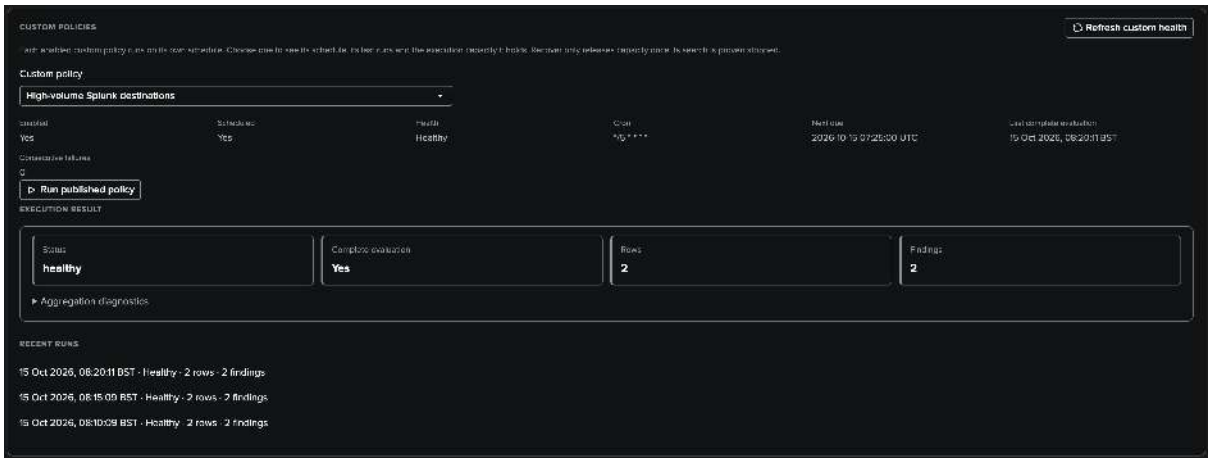
- An object named by a run fails. An object that a complete run no longer names recovers after the number of runs set in **Findings**.
- A run that has a row it cannot resolve to an object recovers nothing: that row may be the one still failing.
- Objects turned off in the Inventory, or with **Stop monitoring this object**, are neither opened nor kept, as for built-in policies.

Where custom findings appear.

- On the Findings tab, with the policy's title. The **Policy** filter lists each custom policy, and the *Findings by policy* chart counts them together as **Custom policies**.
- In the investigation drawer, under **Custom policy evidence**: result rows, occurrences, dimensions and bounded samples.
- In the notification settings, whose policy filter lists each custom policy. Findings action rules filter on built-in policies only.

Health. *Collection health → Custom policies* shows each policy's schedule, health, last runs and the execution capacity it holds. From there, **Run published policy** runs the policy now, exactly as its schedule does, and **Verify**

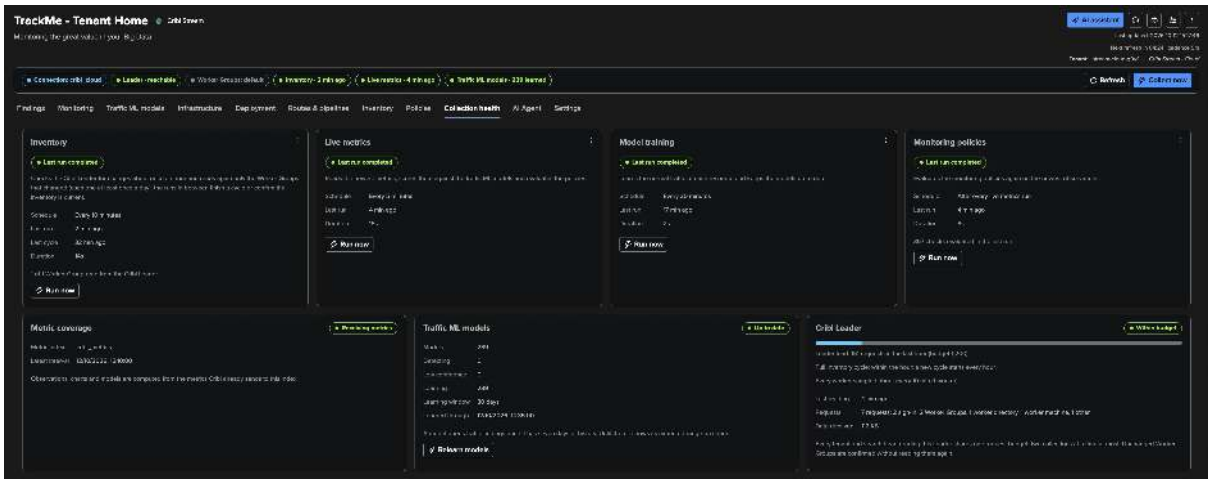
and recover releases a run whose stop could not be proven. Problems also raise the Configuration Guardian check `cribl_collection_health`.



A tenant holds up to 100 custom policies, 25 of them enabled. Every limit and rule is in *Custom policies*.

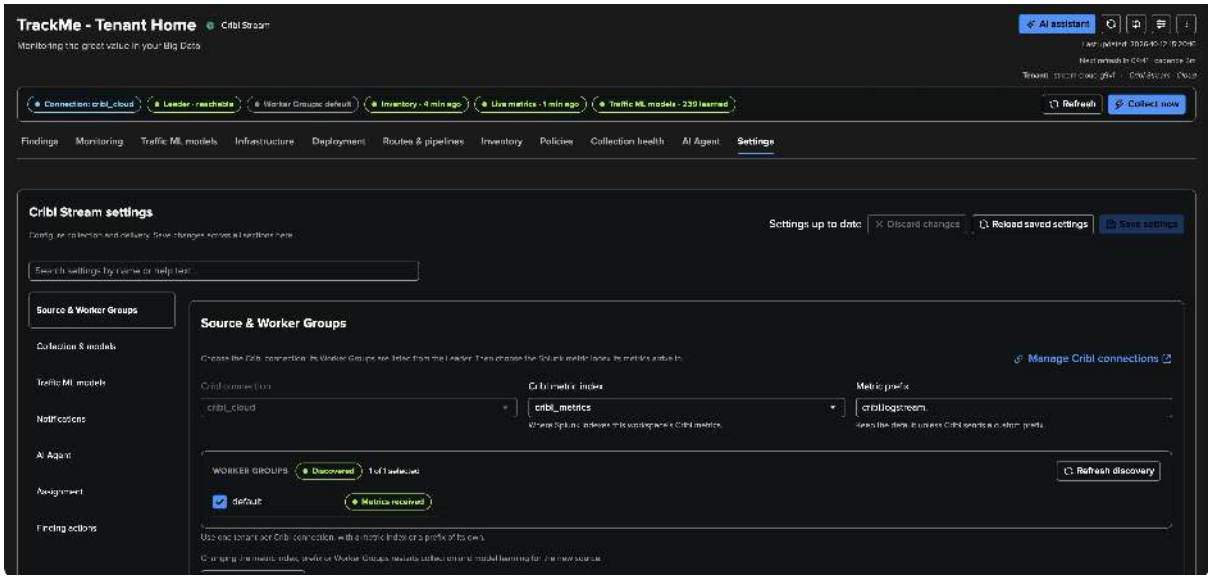
Collection health

Is TrackMe collecting what it should? One card per job — **Inventory**, **Live metrics**, **Model training**, **Monitoring policies** — with its last run, its outcome in words, *View last run*, *Open last run in search* and **Run now**; the **metric coverage** with any Worker Group that sends no metric; the **traffic ML models** with their learning progress and **Relearn models**; and the **Cribl Leader** card with the requests of the last hour against the budget, any slow-down the Leader asked for, the inventory cycle progress and **Read every Worker Group again**.



AI Agent and Settings

AI Agent shows the Cribl Stream AI Agent: whether it can run, its totals and its decision log (see *The Cribl Stream AI Agent*). **Settings** gathers every tenant setting in one searchable page with one *Save*: **Source & Worker Groups**, **Collection & models**, **Traffic ML models**, **Notifications**, **AI Agent**, **Assignment** and **Finding actions**. Changing the metric index, the prefix or the Worker Groups starts a new source: collection and model learning restart and the findings of the previous source close. Changing the schedule or the learning history keeps everything.



Findings in practice

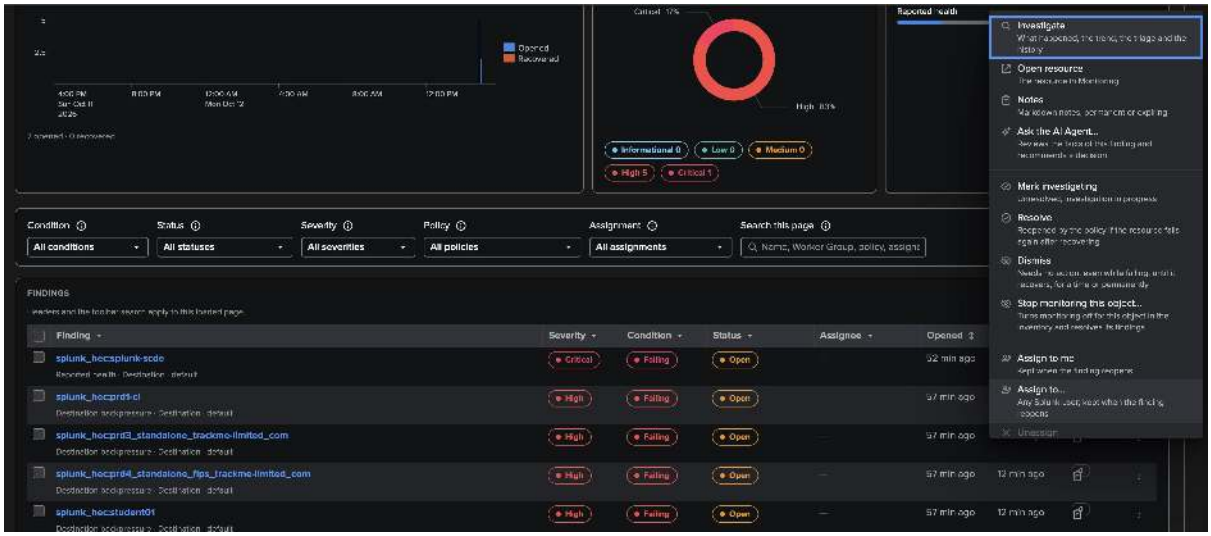
A **finding** is one policy applied to one measurement of one object: *Destination backpressure* on the Destination `splunk_hec`, *Traffic outliers · Events in* on a Source, *Worker availability* on a Worker. It **opens** once its condition held for the policy's consecutive checks and **recovers** after the recovery checks; a measured recovery resolves an open or investigating finding by itself, and a new failure reopens it.

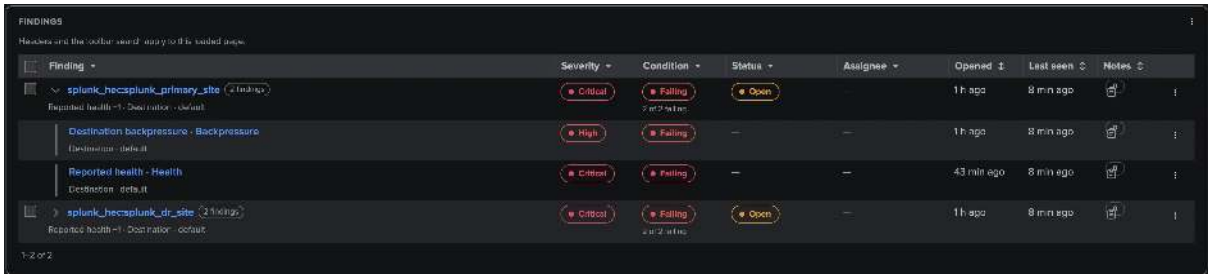
Each finding shows two things side by side: its **condition** — what the newest evaluation measures (*Failing*, *Recovered*, *Awaiting data*) — and its **status**, your decision (*Open*, *Investigating*, *Resolved*, *Dismissed*). Its **severity** is on the TrackMe scale, *Informational*, *Low*, *Medium*, *High*, *Critical*: each policy sets it (a Source Cribl reports *unhealthy* or a fully blocked Destination is critical) and you can override it.

Finding groups. One object can fail several policies at once — a Worker whose machine CPU, memory and processes all run hot. TrackMe groups the findings of one object **automatically** into a **Finding group**, so you see one row and make one decision: status, assignee, snooze, dismissal and severity override live on the group, and its severity is the highest of its failing members. You can also merge groups, add or remove findings, or ungroup by hand.

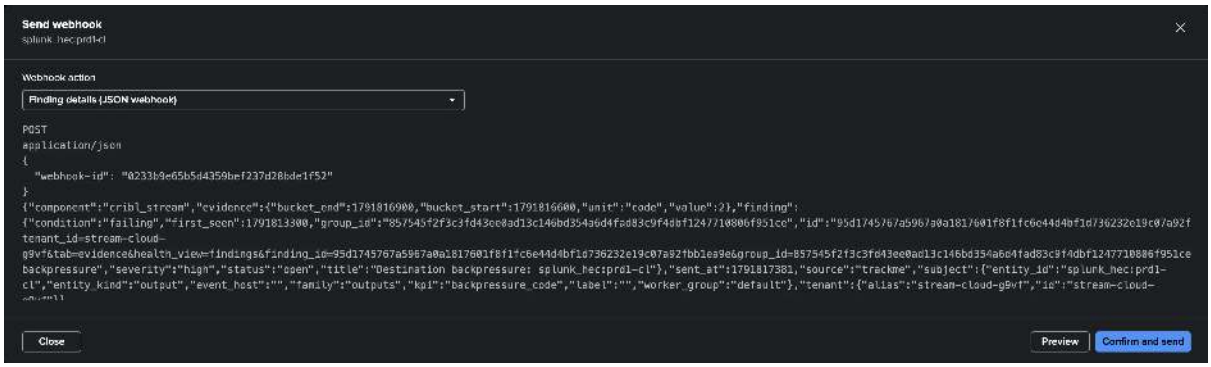
Triage. From the row menu, the drawer or a bulk selection: *Mark investigating*, *Resolve*, *Dismiss* (until it recovers, for 1, 7, 30 or 90 days, or permanently), *Reopen*, *Assign* or *Assign to me*, *Snooze* (stops notifications and colours the object blue in Topology Studio without changing the status), *Set severity*, **Notes**, and on a Traffic outliers finding **Not an anomaly**. Two Cribl-specific actions:

- **Stop monitoring this object** turns monitoring off for the object in one step and resolves its open findings — for a Source you retired on purpose.
- **Forget this object** (a TrackMe administrator of the tenant) removes an object that stopped reporting or was removed from Cribl at once, instead of waiting for its retention.





Assignment and findings actions. Assignment groups and rules route new findings to the right team automatically (*Settings → Assignment*). Findings actions add drilldown searches, links and **webhooks** to your findings, run by hand or by automation rules, through system-wide **Webhook accounts** — for example the `webhook_demo` account posting a new critical finding to a chat channel.



The concepts — Finding groups, triage, assignment, findings actions and webhooks — are shared with the other findings-based tenants and explained once:

- *Findings, Finding groups and findings actions* — findings-based monitoring;
- *Finding groups* — Finding groups;
- *Assignment and automatic routing* — assignment groups and rules;
- *Findings actions* — drilldowns, links, webhooks and automation rules;
- *Webhook accounts* — Webhook accounts.

Traffic ML models and Expected traffic

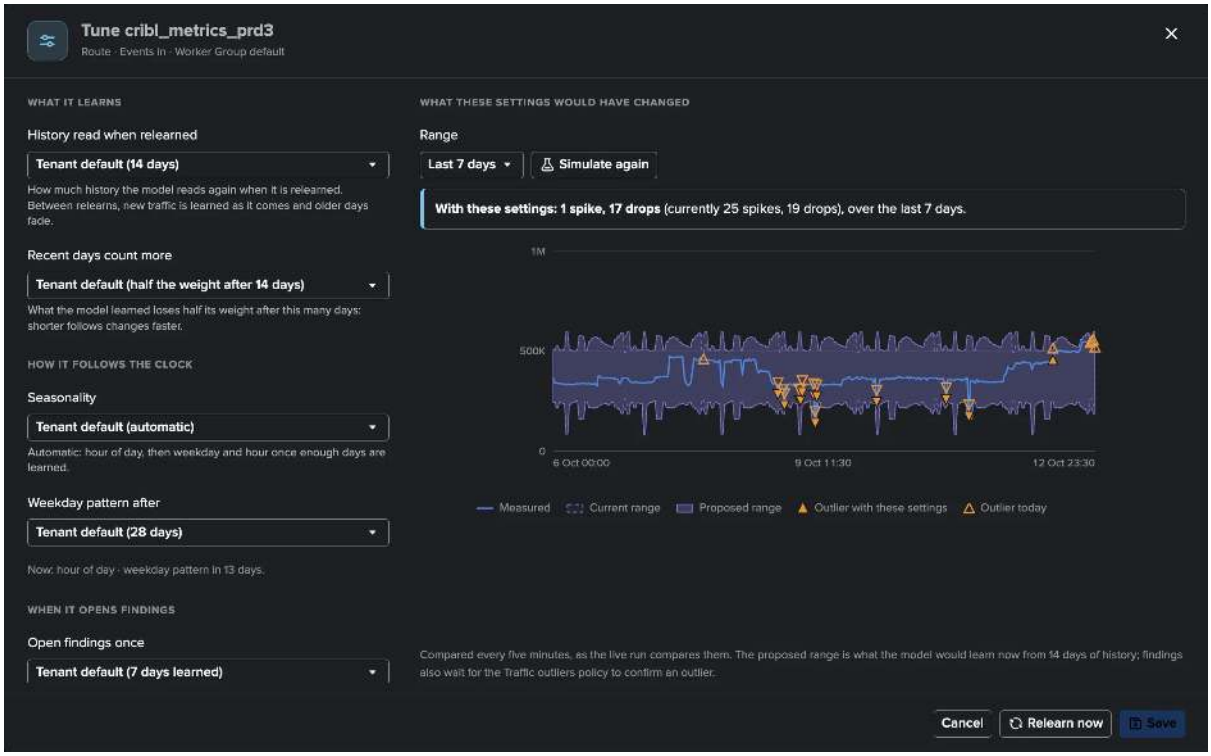
How a model learns. Every Source (events and bytes in), Destination (events and bytes out), Pipeline and Route (events in and out) gets its own model. A model learns the expected range of its traffic **per hour of the day**, then **per weekday and hour** once it has four weeks of history, with recent days counting more. It learns from the history already in your metric index (the *Model learning history* of the wizard), then keeps learning a few times a day from the newest intervals only: there is no periodic full retrain.

A model is in one of these states:

State	Meaning
Learning	Less than a day of history, or hours not covered yet: no expected range.
Low confidence	From a day of history: the range is drawn on the charts, but it opens no finding.
Detecting	From seven days of history: the **Traffic outliers** policy opens a finding when the traffic stays outside the range.
No data	The object stopped reporting (see *Objects that stop reporting*).

Tuning. A TrackMe administrator of the tenant sets the tenant defaults in *Settings* → *Traffic ML models* (sensitivity, seasonality, when the weekday pattern starts, how much more recent days count, when findings open, the smallest deviation that counts) and can give a model — or a selection of models — **its own settings** with **Tune model**: the same settings plus the training days, the direction (drops, spikes or both), a pause, and **excluded periods** the model must not learn from. The tuning screen runs a **live simulation**: the current band and the proposed one over the last 24 hours, 7 or 30 days, with the drops and spikes each would have raised, before anything is saved.





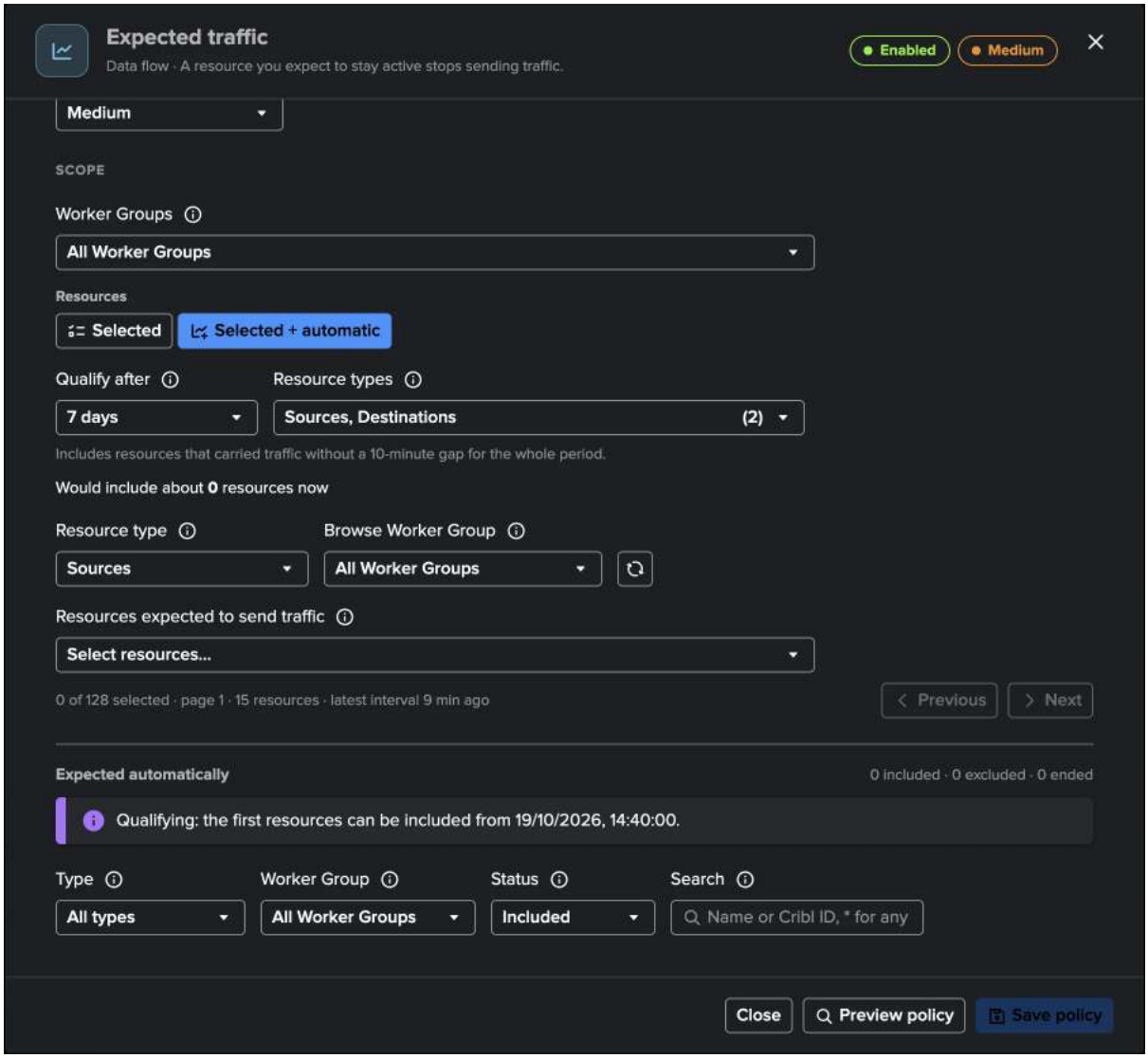
Not an anomaly. When a Traffic outliers finding was expected — a maintenance window, a load test, a planned migration — a power or admin role of the tenant marks it **Not an anomaly**: the episode is excluded from what the model learns, the model relearns without it at the next training run, and the finding is dismissed with your reason as its note.

Bank holidays. A Cribl Stream tenant uses the *bank holiday calendars* like every tenant: the system default calendars, unless it selects its own or none (*Bank Holidays → Tenants & settings*, or *Configure tenant → Bank Holidays*). Two of the four holiday effects apply to it, each on by default:

- **Exclude from ML training** — the traffic models do not learn the hours of the tenant's holidays, so a quiet holiday does not become normal traffic. This applies from the next training run on; a model that learned a holiday before keeps it until it fades or is relearned (nothing is relearned automatically). The Traffic ML models card and the tuning simulation say how many holidays were left out.
- **Suppress new alerts** — while a holiday is active, the policies **set to hold** open no new finding. Each policy chooses it in its editor, section *Bank holidays*: **Hold new findings** or **Keep opening findings**. By default **Traffic outliers** holds (traffic is expected to change on a holiday) and every other policy — Expected traffic, Reported health, backpressure, Pipeline errors, the Worker policies — keeps opening findings, so an infrastructure failure on a holiday still opens its finding. The Leader policies have no setting. The policies list says *held on bank holidays* in the conditions of a policy that holds. A finding already open is still evaluated and can recover and close; a resolved or dismissed finding is not reopened by a held condition; a calendar that cannot be read holds nothing. The policies run and its details report the holiday, its name and how many failing checks were held.

SLA and monitoring windows have no effect on a Cribl Stream tenant. Cribl Edge tenants take the *Suppress new alerts* effect, with their own defaults (see *Cribl Edge*).

Expected traffic. Some objects must never go quiet — a Source fed by a critical application, a Destination to your SIEM. The **Expected traffic** policy opens a finding when an expected object carries no traffic. In **Selected + automatic** mode (the default of a new tenant), TrackMe expects, on top of the objects you select, **every object that carried traffic steadily** — with no gap of more than ten minutes — for the whole qualifying window (2 or 7 days, 7 by default) among the resource types you choose (Sources and Destinations by default; Pipelines and Routes optional). It stops expecting an object after 7 days of silence, when it is removed from Cribl or when it leaves the scope. In the policy, the list of objects expected automatically lets you **Exclude** one, **Include again**, or **Keep as selected**.



Objects that stop reporting, and objects removed from Cribl

Objects that stop reporting

TrackMe remembers every object it ever measured. When an object stops sending metrics, it is not silently forgotten: from its second missing interval it appears under **Monitoring → No data** (and *Traffic ML models → No data*) with the time it last reported and last carried traffic, and it keeps its name in findings, events and emails.

A silence can matter: a Source that stops is often exactly what you want to know. The **Traffic outliers** policy counts the silence of an object as **zero traffic**, but only when TrackMe is sure the silence is real — the Inventory is current and still lists the object, its monitoring is on, its Worker Group is still reporting, it carried traffic within the last 24 hours, its model is Detecting, and most of its Worker Group did not stop at the same time. A silence can open or keep a finding; it never recovers one. When many objects stop together, Collection health says so instead of raising a finding per object.



Objects removed from Cribl

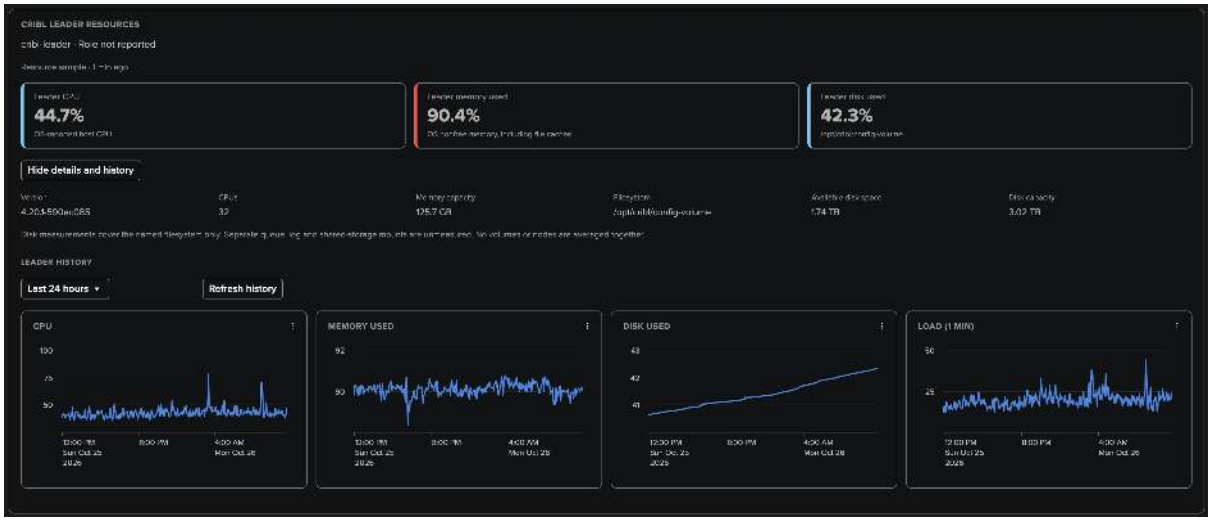
When the Inventory reports an object missing in two consecutive complete readings, TrackMe treats it as **removed from Cribl**: its findings close at once, its silence is never counted, it leaves *No data* and it is purged after 7 days. If more than half of a Worker Group's objects of one type (at least five) disappear at once — a Leader glitch more often than a real removal — TrackMe holds them and notes it in Collection health.

Forget this object (a TrackMe administrator of the tenant, up to 100 objects at a time, audited) purges a silent or removed object at once, from the *No data* list, the Inventory or a finding.

Leader availability and Leader resources

Leader availability. Collecting metrics from Splunk can succeed while the Cribl Leader is unreachable, so TrackMe checks the Leader API on its own at every live run: a sign-in and one small read. The status strip shows **Leader · unavailable** after a failed check, with the reason (network, certificate not trusted, host name mismatch, credentials refused, service unavailable, unexpected answer). Two failed checks in a row open a **Critical** *Leader availability* finding; two successful checks recover it. A Leader asking TrackMe to slow down (*rate limited*) is neither a failure nor a recovery.

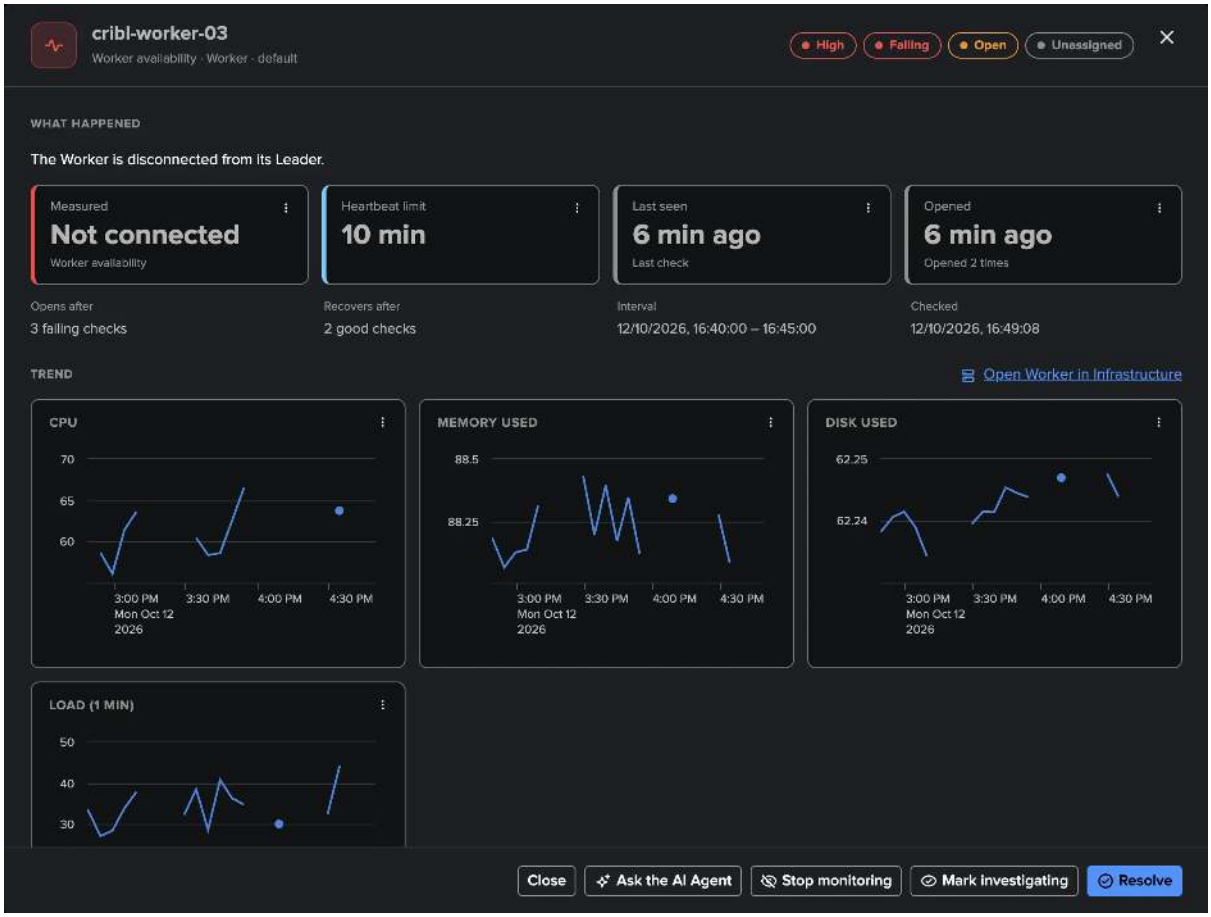
Leader resources. The same live run reads each Leader node's CPU, memory and disk — on a high-availability Leader, each node separately — and keeps their history. The **Leader CPU** and **Leader disk** policies open a finding above 90 % (recovering below 80 %); **Leader memory** is available but off by default, because the memory the operating system reports as used includes its file cache.



Workers, drift and machine resources

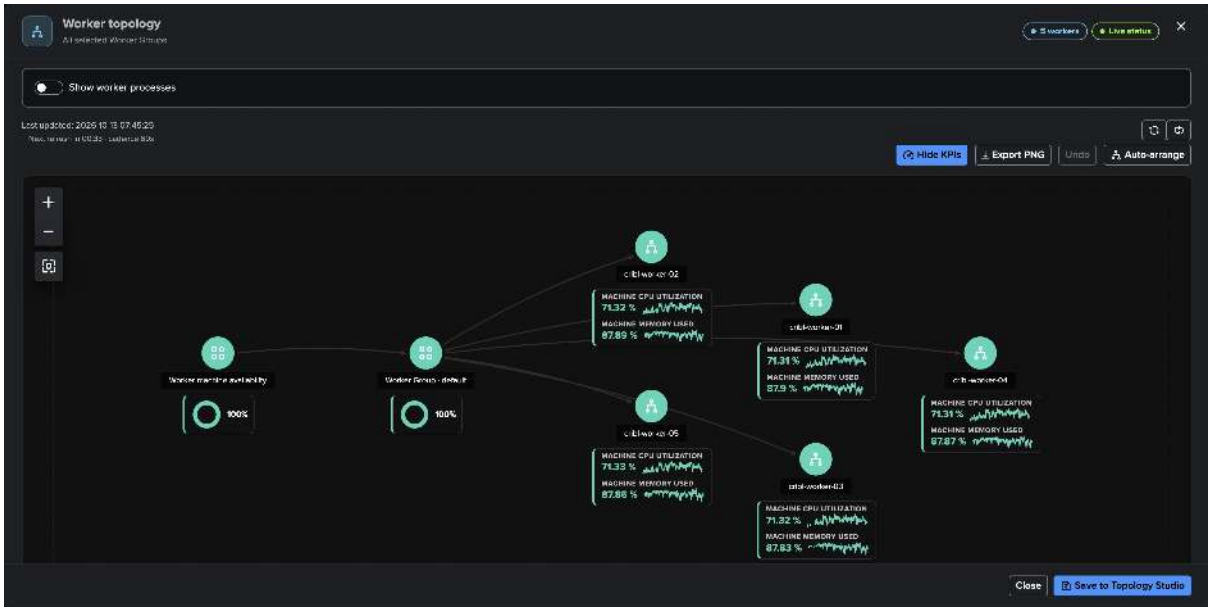
From the Leader, TrackMe knows every Worker and how it should look:

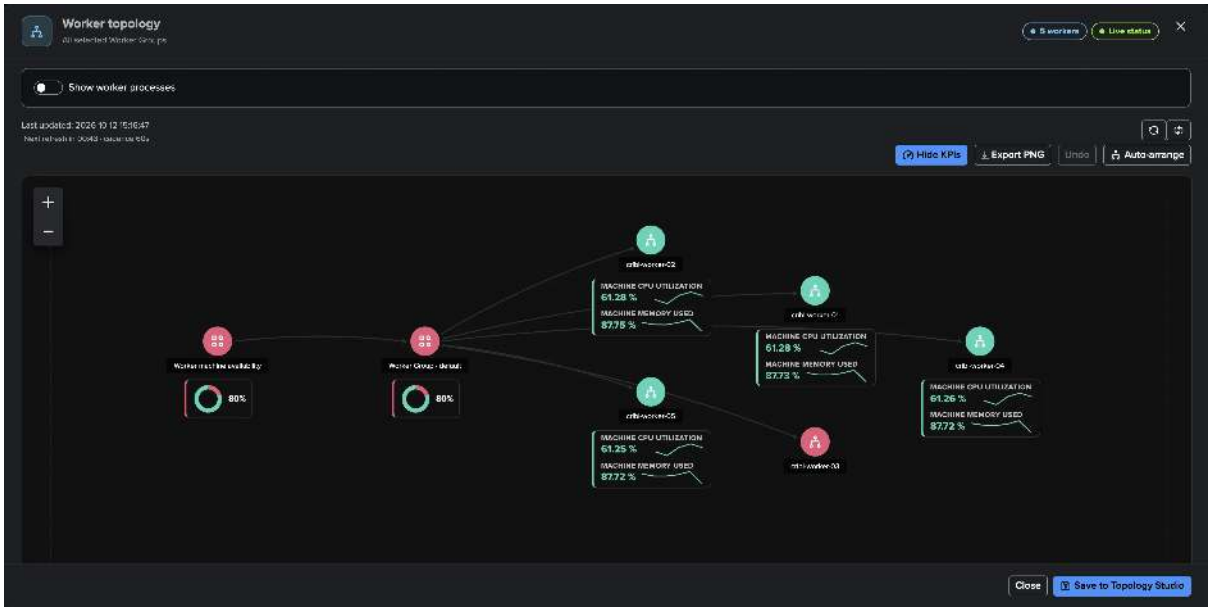
- **Worker availability** (High) — the Leader lists the Worker as disconnected, or its heartbeat is older than 10 minutes.
- **Worker version drift** (Low) — a connected Worker runs another Cribl version than the running Leader, typically during an upgrade that did not finish.
- **Worker configuration drift** (Low) — a Worker does not run its Worker Group's current configuration: a deploy that did not reach it.
- **Worker machine CPU** and **Worker machine disk** (Medium, above 90 %); **Worker machine memory** is available but off by default, for the same reason as the Leader's.
- **Process CPU** (Medium) — a worker process above 90 % CPU, on average or at its peak.



Worker topology

Infrastructure → Worker topology draws the Worker Group and its Workers as a live map, each Worker with its KPI cards (CPU, machine memory used, disk) and its live status: a Worker with a finding turns orange or red, a healthy one green. **Show worker processes** adds each Worker's processes under it. **Live status** refreshes the states and KPIs every minute; **Export PNG** saves the picture and **Save to Topology Studio** keeps it as a view that stays live. **Process topology** draws the worker processes alone.

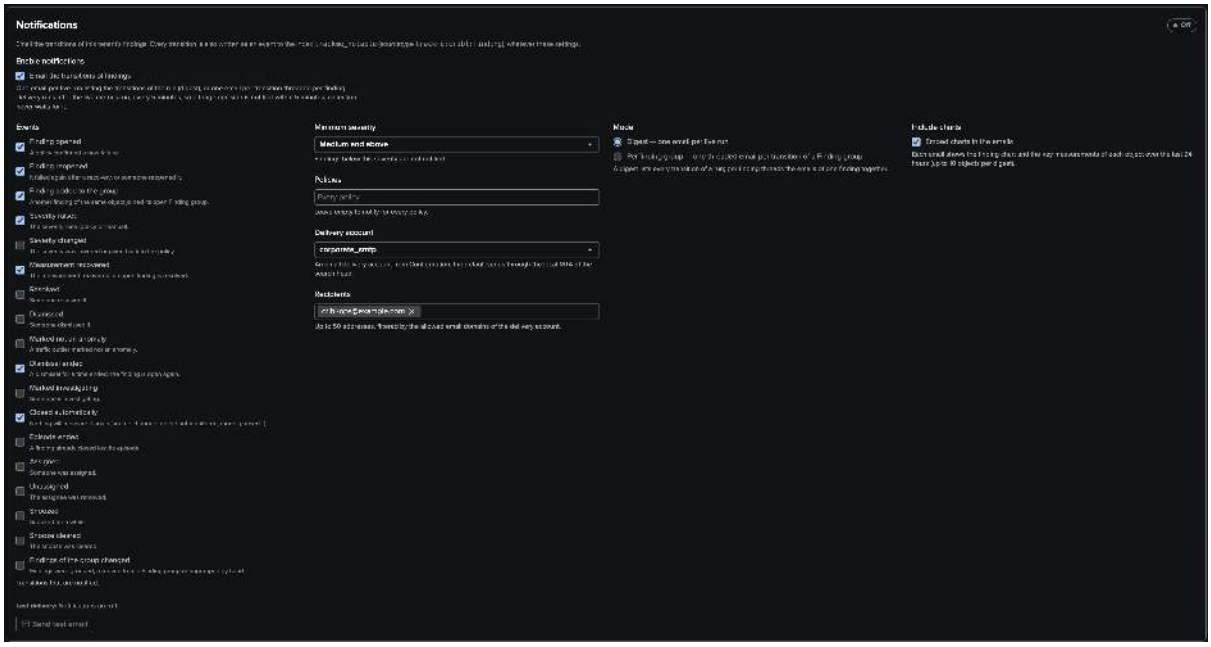




Notifications

Every transition of a finding or a Finding group is written as an event in the tenant's **notable index** (`sourcetype=trackme:cribl:finding` and `trackme:cribl:finding_group`), whatever the notification settings, for your Splunk alerts, Enterprise Security or SOAR.

Email notifications are off by default. In *Settings → Notifications*, a TrackMe administrator of the tenant chooses the events (opened, reopened, added to the group, escalated, recovered, dismissal ended and closed by default), the minimum severity (*Medium and above* by default), the policies, the email delivery account and up to 50 recipients, and whether to send **one digest per run** or **one threaded email per Finding group**, with the charts of the objects inline. **Send test email** checks the saved settings. Emails leave after each live run and link straight to the finding.



The Cribl Stream AI Agent

The **Cribl Stream AI Agent** reviews findings the way an operator would, from facts TrackMe checks rather than from guesses: when the object was first seen, whether it is disabled in Cribl or gone from the Inventory, which Routes reference it, its traffic and its traffic ML model. It assesses each finding (true positive, likely false positive, object not

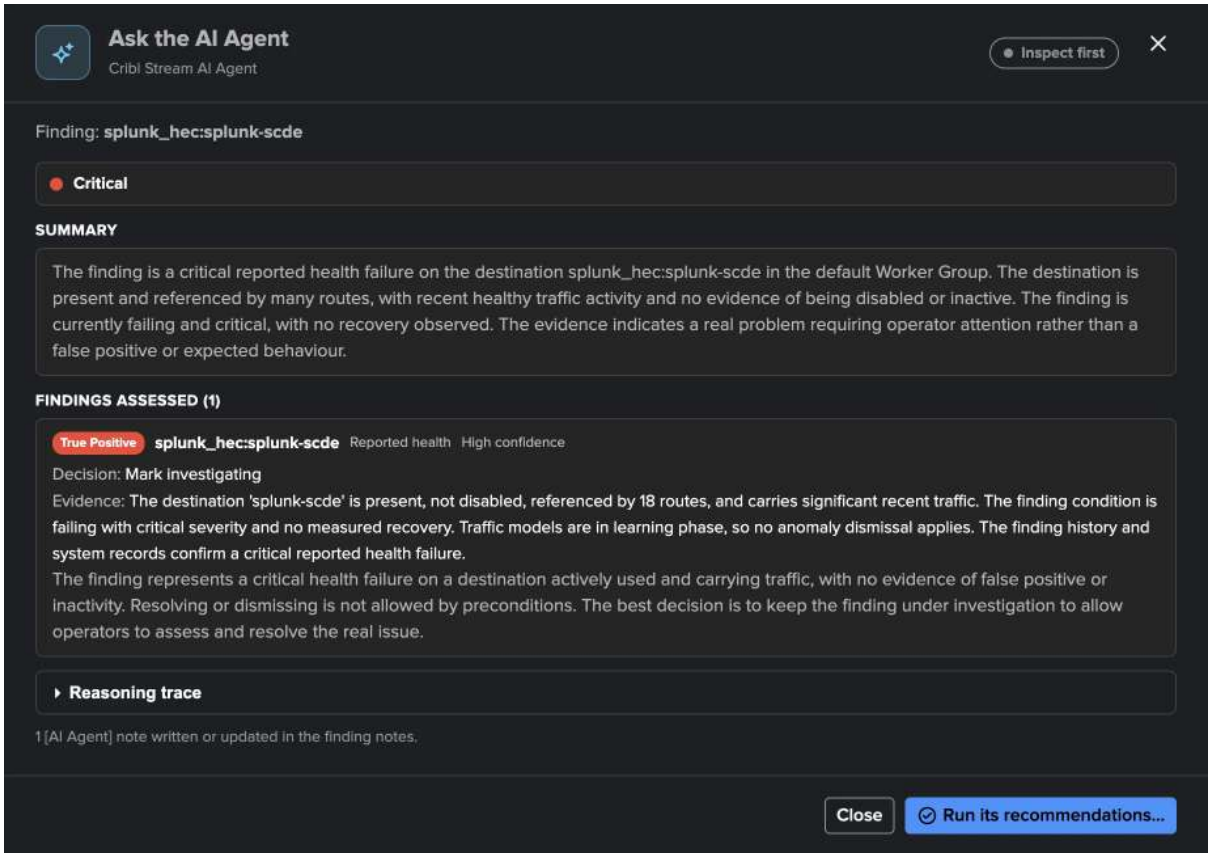
in use, expected behaviour, needs investigation), recommends a decision, and keeps one current **[AI Agent]** note on the finding.

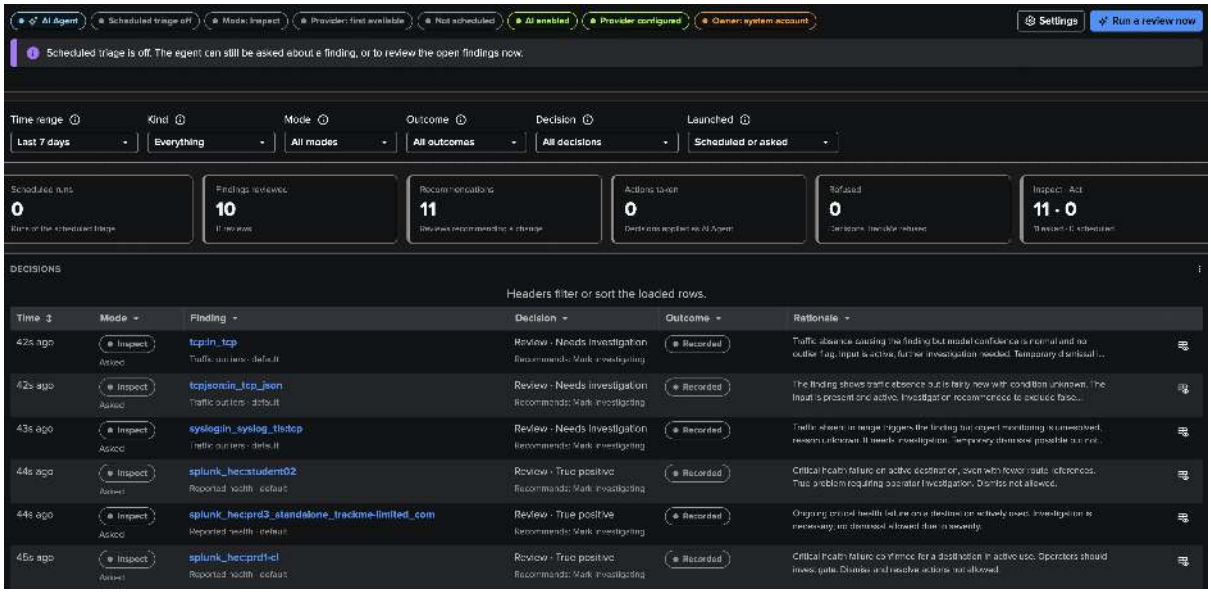
- **Inspect** only recommends: nothing changes but the note.
- **Act** applies the recommendation, but only what the tenant allows: by default *Mark investigating*, *Dismiss for a time* (7 days at most) and *Not an anomaly*. *Resolve* and *Stop monitoring* are off by default and need a proof (for example the object is disabled in Cribl or gone from the Inventory). The agent never dismisses permanently, assigns, changes a severity, a policy or a model, and never forgets an object (it may recommend it to an administrator).

Ask the AI Agent from a finding, a Finding group, the row menu or the AI Agent tab: it inspects first, and you confirm before it acts. Asking and applying need a power or admin role of the tenant.

Scheduled triage is off by default. A TrackMe administrator of the tenant turns it on in *Settings → AI Agent*: hourly, up to 10 findings per run, a finding reviewed again after a day unless a new episode started. It needs the AI Assistant enabled, an AI provider and a tenant owner that is a Splunk administrator account. The agent is also listed in **Manage AI Agents automation** (*AI Agents automation*).

Custom instructions (*Settings → AI Agent*) are plain English the agent follows on every run, asked or scheduled: for example, that the Destinations named `test_*` are test outputs whose backpressure is expected. At most 4000 characters; they never widen what the agent may do. **Evaluate…** tries an unsaved draft in inspect mode, on one finding or a short review, optionally side by side with a run without instructions, and writes nothing.



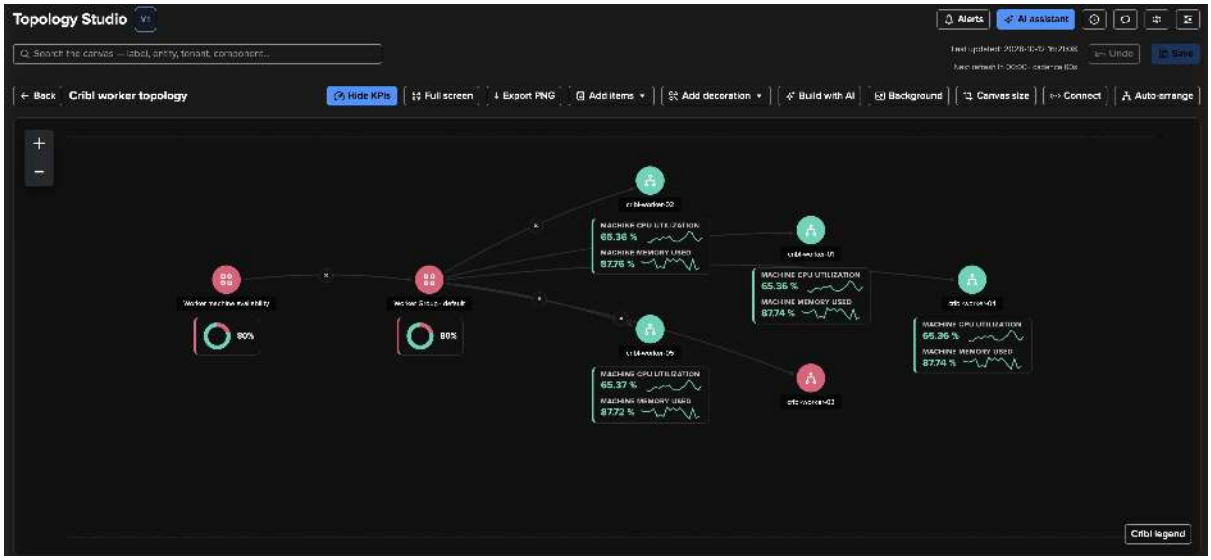


The concepts shared with the other findings AI Agents are explained in *Findings AI Agents*, and the AI side of the agent in *AI Agents for findings*.

Topology Studio

Cribl objects are first-class citizens of *Topology Studio*:

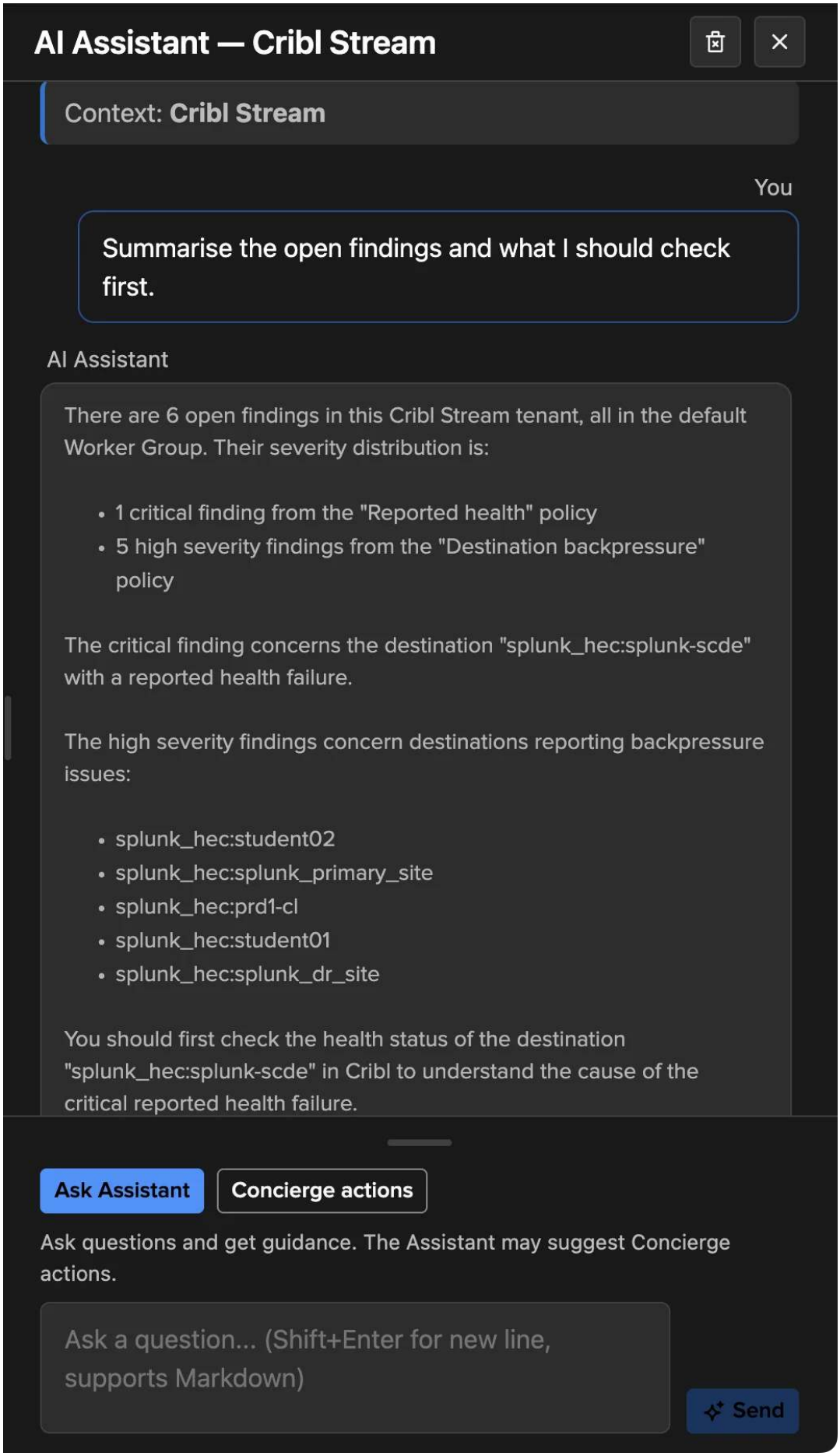
- **From the workspace —** the Flow topology of *Routes & pipelines* and the Worker and Process topologies of *Infrastructure* are saved as live views.
- **In the Studio —** *Add entities* picks Sources, Destinations, Pipelines, Routes, Workers and worker processes of a Cribl Stream tenant; **dynamic Cribl groups** follow a selection (a type, Worker Groups, a name pattern, optionally only the objects with a finding) as objects come and go (*Example — Cribl Logstream with a priority filter*); five templates build a block in one step: *Flow of a Worker Group*, *Worker fleet*, *Destinations wall*, *Problems wall* and *Deployment overview*.
- **Live state —** a Cribl node is red when one of its findings is critical, orange otherwise, blue when its findings are only snoozed, green when measured without a finding, and grey when not measured or not monitored. Nodes carry their KPIs and 24-hour sparklines.
- **Alerts and AI —** *Topology Alerts* work on Cribl nodes, and *Build with AI* and the Topology Studio assistant name your Cribl objects.

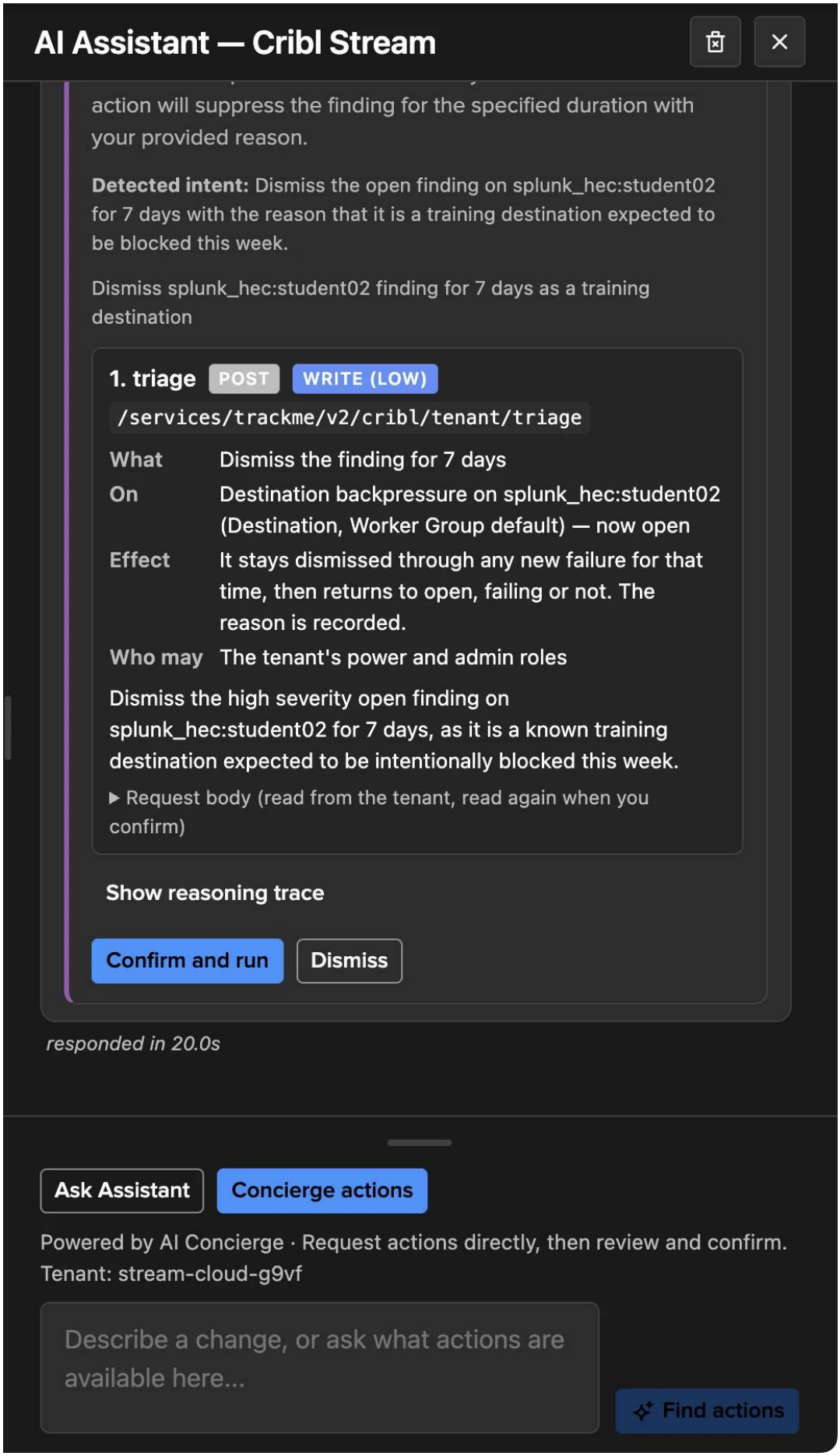


AI Assistant and Concierge

The **AI Assistant** of a Cribl Stream tenant (*AI Assistant — Cribl Stream* on the workspace) is grounded in the tenant: its findings, traffic ML models, collection health, monitoring state and Leader load. Ask it why a finding is open, what to check in Cribl, or when a model will be ready.

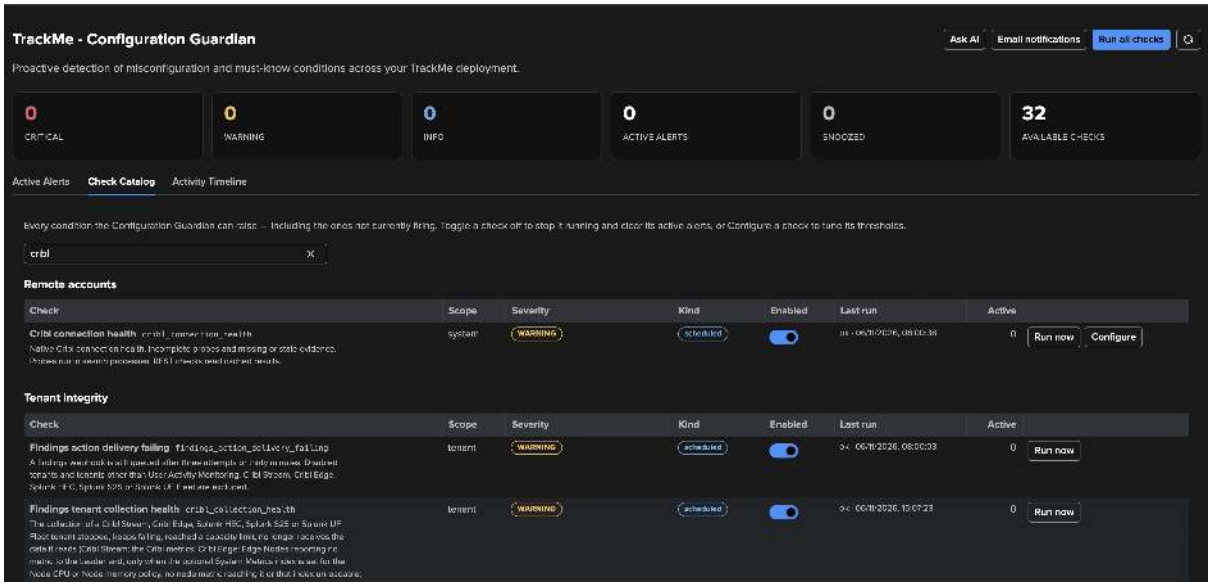
When you ask for a change it can make, it proposes an *AI Concierge* consent card that you confirm before anything runs: a decision on a finding or a Finding group, *Not an anomaly*, monitoring on or off for named objects, *Stop monitoring*, running a collection job, or relearning one model or all of them. Each action keeps the role it requires; policies, exclusion rules, tuning and settings stay in the workspace.


AI Assistant — Cribl Stream
Context: Cribl Stream
You
Summarise the open findings and what I should check first.
AI Assistant
There are 6 open findings in this Cribl Stream tenant, all in the default Worker Group. Their severity distribution is:
1 critical finding from the "Reported health" policy
5 high severity findings from the "Destination backpressure" policy
The critical finding concerns the destination "splunk_hec:splunk-scde" with a reported health failure.
The high severity findings concern destinations reporting backpressure issues:
splunk_hec:student02
splunk_hec:splunk_primary_site
splunk_hec:prd1-cl
splunk_hec:student01
splunk_hec:splunk_dr_site
You should first check the health status of the destination "splunk_hec:splunk-scde" in Cribl to understand the cause of the critical reported health failure.
Ask Assistant
Concierge actions
Ask questions and get guidance. The Assistant may suggest Concierge actions.
Ask a question... (Shift+Enter for new line, supports Markdown)
Send


AI Assistant — Cribl Stream
action will suppress the finding for the specified duration with your provided reason.
Detected intent: Dismiss the open finding on splunk_hec:student02 for 7 days with the reason that it is a training destination expected to be blocked this week.
Dismiss splunk_hec:student02 finding for 7 days as a training destination
1. triage
POST
WRITE (LOW)
/services/trackme/v2/cribl/tenant/triage
What
Dismiss the finding for 7 days
On
Destination backpressure on splunk_hec:student02 (Destination, Worker Group default) — now open
Effect
It stays dismissed through any new failure for that time, then returns to open, failing or not. The reason is recorded.
Who may
The tenant's power and admin roles
Dismiss the high severity open finding on splunk_hec:student02 for 7 days, as it is a known training destination expected to be intentionally blocked this week.
▶ Request body (read from the tenant, read again when you confirm)
Show reasoning trace
Confirm and run
Dismiss
responded in 20.0s
Ask Assistant
Concierge actions
Powered by AI Concierge · Request actions directly, then review and confirm.
Tenant: stream-cloud-g9vf
Describe a change, or ask what actions are available here...
Find actions

Collection health and the Configuration Guardian

Collection health keeps you from trusting a silent tenant. Besides the tab, the **Configuration Guardian** check `cribl_collection_health` raises a warning when a job is not running or keeps failing, when metrics stop arriving or a Worker Group sends none, when the inventory cycle or the worker sampling slows down under the Leader budget, when many objects stop together, or when notification emails fail. Each problem carries its next step. A disabled tenant or a tenant on manual collection reports nothing. The Guardian also checks your Cribl connections (`cribl_connection_health`). Every problem code is listed in *Cribl Stream reference*.



See also

- *Cribl Stream reference* — the policy catalogue, schedules, settings, events, RBAC, Guardian checks, REST endpoints, capacity and troubleshooting.
- *Cribl Stream* — the Cribl Stream hands-on walkthrough.
- *Cribl connections* — Cribl connections.
- *Findings, Finding groups and findings actions* — findings-based monitoring, shared by every dedicated tenant type.
- *Cribl Stream & Cribl Edge* — the Cribl API add-on and Flex Objects use cases.

Next: the reference

Continue with *Cribl Stream reference* for every policy, setting, schedule and limit.

Cribl Stream reference

This page is the reference companion of *Cribl Stream*: the catalogue of built-in policies, the jobs and their schedules, the Leader request budget, the traffic ML models, the rules for silent and removed objects, the severity scale, every tenant setting, the indexed finding events, the permissions, the Configuration Guardian checks, the REST endpoints, the capacity and the troubleshooting table. The concepts shared with the other findings-based tenants (Finding groups, triage, assignment, findings actions, AI Agent) are in *Findings, Finding groups and findings actions*.

Policy catalogue

A Cribl Stream tenant evaluates **16 built-in policies**, shown in four groups on the **Policies** tab. Every policy has the same common settings:

- **Enabled** — on or off.

- **Severity** — the severity of the findings it opens (see *Severity scale*).
- **Open after / Recover after** — how many consecutive failing checks open a finding and how many healthy checks recover it (1 to 12). Metric policies count five-minute intervals; Worker and Leader policies count checks at least five minutes apart, so repeated manual collections never shorten a confirmation.
- **Worker Groups** — the Worker Groups it applies to (empty = all).
- **Selected objects** — the objects it applies to (up to 128; empty = all).
- **Bank holidays** — *Hold new findings* or *Keep opening findings* while a bank holiday of the tenant's calendars is active and the tenant suppresses new alerts on holidays. Default: **Traffic outliers** holds, every other policy keeps opening findings; the four Leader policies have no such setting. A finding already open keeps being evaluated.

Threshold policies are **bad at or above the threshold**, **good at or below the recovery threshold**, and hold their current condition in between. Missing data is never turned into a zero, except by the *Traffic outliers* policy for silent objects (see *Silent and removed objects*). A saved policy applies from the first collection interval after the save.

Cribl Leader

Policy	What it detects	Default	Severity
Leader availability	The configured Leader API cannot be reached or used.	On	Critical
Leader CPU	Operating-system CPU of each Leader node.	On	Medium
Leader memory	Memory in use on each Leader node.	Off	Medium
Leader disk	Used space of each filesystem a Leader node reports.	On	Medium

Data flow

Policy	What it detects	Default	Severity
Reported health	A Source or Destination Cribl reports degraded or unhealthy.	On	Medium → Critical
Traffic outliers	Traffic outside the range its traffic ML model expects.	On	Medium
Expected traffic	A selected or automatically expected object carries no traffic.	On	Medium
Pipeline errors	Pipeline errors at or above the threshold.	On	Medium
Destination backpressure	A Destination reports backpressure or is blocked.	On	High → Critical

Worker processes

Policy	What it detects	Default	Severity
Process CPU	CPU of each worker process.	On	Medium

Worker machines

Policy	What it detects	Default	Severity
Worker machine CPU	Operating-system CPU of each Worker.	On	Medium
Worker machine memory	Memory in use on each Worker.	Off	Medium
Worker machine disk	Used space of the filesystem Cribl reports for each Worker.	On	Medium
Worker availability	A Worker is disconnected or its heartbeat is too old.	On	High
Worker version drift	A connected Worker does not run the running Leader's version.	On	Low
Worker configuration drift	A Worker does not run its Worker Group's configuration.	On	Low

Default settings

What each policy applies to, its checks to open and to recover, and its own settings with their defaults.

Policy	Applies to	Open / recover	Settings and notes (defaults)
Leader availability	The Leader API	2 / 2	Fails on the network, the TLS certificate or host name, authentication, an unavailable service or a malformed answer. A *rate limited* answer is unknown, never a failure.
Leader CPU	Leader node	3 / 2	Threshold 90 %, recovery 80 %.
Leader memory	Leader node	3 / 2	Threshold 90 %, recovery 80 %. File cache included (see the note below).
Leader disk	Leader node filesystem	3 / 2	Threshold 90 %, recovery 80 %, per named filesystem.
Reported health	Source, Destination	3 / 2	Critical when Cribl reports it unhealthy, the policy severity when degraded. Only objects that carried traffic in the last 24 hours, are selected, or are already failing (*Include resources without traffic*: off).
Traffic outliers	Source, Destination, Pipeline, Route (per measurement)	2 / 2	Detecting models only. Direction: drops and spikes (or drops only, spikes only; also per model). The silence of an object counts as zero traffic, under guards.
Expected traffic	Source, Destination, Pipeline, Route	3 / 1	Opens nothing until objects are selected or expected. A new tenant uses *Selected* + *automatic*: objects qualify after 7 days; Sources and Destinations.
Pipeline errors	Pipeline	3 / 2	Threshold 1 error per interval.
Destination back-pressure	Destination	2 / 2	Critical when fully blocked. At most one finding per Destination and condition.
Process CPU	Worker process	3 / 2	Threshold 90 %, recovery 80 %; statistic *Average* (or *Peak*).
Worker machine CPU	Worker	3 / 2	Threshold 90 %, recovery 80 %.
Worker machine memory	Worker	3 / 2	Threshold 90 %, recovery 80 %. Cache included (see the note below).
Worker machine disk	Worker filesystem	3 / 2	Threshold 90 %, recovery 80 %.
Worker availability	Worker	3 / 2	Heartbeat limit 600 s (60 to 3,600).
Worker version drift	Worker	3 / 2	Compared with the version of the running Leader.
Worker configuration drift	Worker	3 / 2	Compared with the Worker Group's configuration version.

Note

Why memory policies are off. The memory the operating system reports as used includes its file cache, which fills a healthy Linux host. *Leader memory* and *Worker machine memory* are available but not enabled by default; enable them knowingly, with a threshold that suits your hosts.

Each policy's editor also has an **Actions** tab (drilldowns, links and webhooks offered on its findings) and an **Automations** tab (rules that run an action automatically): see *Findings actions*.

Severity scale

Cribl Stream findings use the TrackMe scale **Informational, Low, Medium, High, Critical**.

Rule	Effect
Policy severity	Each policy has its default severity (catalogue above), which a TrackMe administrator of the tenant can change.
Raised by the condition	*Reported health*: a Source or Destination Cribl reports **unhealthy** is Critical. *Destination backpressure*: a fully **blocked** Destination is Critical. Such a raise is an *escalated* transition.
Operator override	*Set severity* on a finding or a Finding group overrides the policy's; *Use policy severity* restores it.
Finding group	The highest severity of its failing members, unless overridden on the group.
Settled findings	A recovered, ended or resolved finding is shown **Informational**; the severity its episode reached stays recorded (`peak_severity`).
Earlier builds	A severity stored as *warning* reads **Medium**.

Notifications filter on the minimum severity (*Medium and above* by default).

Custom policies

A custom policy is an SPL search on the tenant's Splunk data whose result rows each name a Cribl object. How to write one: *Custom policies*.

Object types and result fields.

Object type	Result fields	Resolution
Source	Worker Group, Source	The Source as the Cribl metrics name it (the `input` dimension, `type:id`).
Destination	Worker Group, Destination	The Destination as the Cribl metrics name it (the `output` dimension).
Pipeline	Worker Group, Pipeline	The Pipeline id.
Route	Worker Group, Route	The Route id or name. A custom Route finding joins the Finding group of its Worker Group and Route value.
Worker Group	Worker Group	The Worker Group id.
Worker machine	Worker	The Worker id or its host name, matched without regard to case in the current Worker directory. Not listed, or a host name shared by several Workers: unresolved.
Worker process	Worker Group, host, process	The host and the worker process (`cribl_wp`).

A Worker Group id is letters, digits, `_`, `.`, `:` and `-` (up to 128 characters). **Dimension fields** (up to eight) give an object one finding per value; changing the object mapping or the dimensions needs a new **identity version**, which starts new findings while the earlier ones recover.

Runs.

- Each enabled policy has its own scheduled report, `trackme_cribl_custom_<policy id>_tenant_<tenant>`. It runs as the tenant owner, one run at a time, on the policy's cron, while the tenant is enabled and its collection is scheduled. **Run published policy** dispatches the same report.
- Custom policy searches run in parallel: two per tenant and four per search head, shared with UAM custom policies. A run that finds no free slot is deferred to its next schedule.
- A custom finding is current until its next run is due: the longest gap of its cron plus 15 minutes, at most 32 days.
- An object that a complete run no longer names recovers after the runs set in **Findings**. A run with an unresolved or ambiguous row recovers nothing.
- Objects turned off in the Inventory or with **Stop monitoring this object** are neither opened nor kept (`not_monitored` in the run receipt).
- **During bank holidays: Hold new findings** opens no new finding while a holiday of the tenant's calendars is active at the end of the search window. It takes effect when published.

Lifecycle.

- **Save draft** keeps up to 30 drafts; **Publish** creates an immutable revision (up to 100 kept). The published revision in use cannot be deleted.
- **Disable**, **Archive** and **Delete permanently** close the policy's open findings (resolved, reason *not evaluated*). A deletion removes the definition, its report and its run history, and keeps the closed findings with their evidence and notes. If the findings could not be closed, the deletion is refused and can be retried.
- An archived policy cannot be enabled again.
- A tenant deletion removes its custom policies and their reports. It is refused while a custom policy search still holds execution capacity.
- Backup and restore: definitions are archived, run state is not. A restored policy comes back disabled: publish it, then enable it.

Limits.

Limit	Value
Custom policies per tenant / enabled	100 / 25
Search window	Relative, ending now, at most 31 days
Run: duration / result rows	300 seconds / 10 000
Preview: duration / result rows / candidates shown	60 seconds / 1 000 / 20
Findings per run	1 000
Evidence per finding: total / samples / per sample	6 144 bytes / 3 / 2 048 bytes
Findings a policy keeps current (open, confirming or closed while still failing)	1 000
Unresolved custom findings per tenant	5 000; beyond, new failures are refused while recorded ones keep being evaluated

Jobs and schedules

Each tenant is served by scheduled reports, each with its own lock and run receipt. Schedules are **offset per tenant** (from the tenant identifier), so several tenants reading the same Leader never start in the same minute.

Job	Schedule	What it does
Live metrics	every 5 minutes	Checks the Leader API, reads the newest settled five-minute interval of every metric family, scores traffic against the models, publishes the snapshot every tab reads, refreshes the Workers and the Leader resources, evaluates every policy, then delivers the queued notifications.
Inventory	every 10 minutes; a new cycle about hourly	Reads the configuration and the Workers from the Leader. A new cycle starts once the previous one is 50 minutes old; a Worker Group whose configuration version did not change is confirmed without a read (and re-read at least once a day); at most 10 Worker Groups are read per run, the changed ones first.
Traffic ML models	twice an hour	Learns the models when a pass is due (every 6 hours, or while bootstrapping); returns at once otherwise. After repeated failures it waits 1, 2, 4 then 6 hours.
Monitoring policies	on demand only	Evaluates the policies again on the newest snapshot. The scheduled evaluation is the one that ends every live run.
AI Agent	hourly, when enabled	Scheduled triage by the Cribl Stream AI Agent (off by default; the report stays unscheduled until it is enabled).

Manual collection. *Settings → Collection & models → Scheduled collection* is on by default; off, the reports are kept but unscheduled and Collection health reads *Manual collection*.

Run now. Collect now (status strip) runs the live job; **Run now** on each Collection health card runs that job; **Read every Worker Group again** (Inventory card) starts a cycle that reads every Worker Group regardless of its configuration version, for edits saved on the Leader but not yet deployed. Cooldowns per tenant, whoever asks: **inventory 120 s**, **live 60 s**.

Changing the source. Changing the connection, the metric index, the metric prefix or the Worker Groups starts a **new source**: collection and model learning restart, and the findings of the previous source close. Changing the schedule or the learning history keeps everything.

Leader request budget

TrackMe is gentle with the Cribl Leader at any size. Every caller that reads one Leader — every tenant, every job, the connection test, the wizard discovery — draws from **one shared budget** for that Leader:

- **Requests per hour** — 1 200 by default (120 to 20 000), a setting of the Cribl connection.
- **Burst** — 150 by default (10 to 2 000), also on the connection.
- When several connections reach the same Leader, the **lowest** values apply.
- At most two runs read one Leader at a time, at most 8 requests per second together.

A run that would exceed the budget is **deferred** and resumes exactly where it stopped (inventory page, Worker directory page, machine-sample position): protection, not failure. When the Leader answers *slow down* (HTTP 429 with a wait), every later caller waits for it; after three failing runs in a row the Leader is left alone for 60 seconds, doubling up to 30 minutes. The Collection health **Cribl Leader** card shows the requests of the last hour against the budget, the last run's requests and data received, and any requested wait.

Only the live and inventory jobs, the connection test, the wizard discovery and the live status of a Source or Destination read the Leader. Every tab, Topology Studio, the policies and the model training read what TrackMe already stored.

Traffic ML models

Aspect	Behaviour
What gets a model	One model per object and measurement, all worker processes combined: Sources (events in, bytes in), Destinations (events out, bytes out), Pipelines and Routes (events in, events out). Up to **20 000 models** per tenant; the busiest series get one first.
How it learns	A normal distribution per hour of the day, then per weekday and hour once 28 days of history are learned. Recent days count more (half-life 14 days by default).
Learning history	At creation, the models learn from the history already in the metric index, up to the *Model learning history* (30 days by default, 0 to 90), bounded by the index's retention.
Retraining	No periodic full retrain: a pass every 6 hours reads only the intervals since the last one and merges them in. Each run has a bounded time budget and continues at the next run where it stopped.
Relearn	**Relearn** one model (or a selection) or **Relearn models** (all) re-reads the history. TrackMe administrators of the tenant.
States	**Learning** (less than a day, or hours not covered), **Low confidence** (range shown, opens no finding, until 7 days), **Detecting** (opens findings), **No data** (the object stopped reporting).
Expected range	Expected value ± z deviations: *Normal* 3.0, *Low* 4.0, *High* 2.5 (custom 2.0 to 6.0); deviations smaller than 10 % of the expected value or 1 event or byte per five minutes are ignored.
Per-model settings	Up to 500 models with their own settings (Tune model): training days (7 to 90), direction, pause (closes its Traffic outliers findings), up to 20 excluded periods.
Not an anomaly	Excludes the episode (from an hour before it opened to its recovery, 7 days at most) from what the model learns, relearns it at the next training run and dismisses the finding. Power and admin roles of the tenant; for non-administrators the excluded periods of a model total at most 7 days and never leave less history than the model needs to open findings.
Bank holidays	With the tenant's *Exclude from ML training* effect on (the default), the hours of its bank holidays are not learned, like an excluded period, by training, the history bootstrap, a relearn and the tuning simulation. From the next training run on; no automatic relearn. With *Suppress new alerts* on, the policies set to hold (Traffic outliers by default, see *Policy catalogue*) open no new finding while a holiday is active (reason *Bank holiday*); open findings keep being evaluated. See *Bank holidays*.

Expected traffic: automatic inclusion

Rule	Value
Modes	**Selected** (only the objects you select) or **Selected + automatic** (default of a new tenant; tenants created on earlier builds keep *Selected*).
Qualify after	An object is expected automatically once it carried traffic with no gap longer than 10 minutes for the whole window: **7 days** (default) or 2 days.
Resource types	Sources and Destinations by default; Pipelines and Routes optional.
Stops expecting	After **7 days of silence**, when the object is removed from Cribl, or when it leaves the scope.
Per object	**Exclude**, **Include again**, **Keep as selected** (adds it to the selected objects).

Silent and removed objects

Rule	Value
Shown under *No data*	From the second missing five-minute interval.
Silence counted as zero traffic	By *Traffic outliers* only, and only when the Inventory is current and lists the object, monitoring is on, its family is collected, its Worker Group reports, it carried traffic within 24 hours, its model is Detecting (and watches drops), and no mass silence is in progress. A silence can open or keep a finding, never recover one.
Many objects stopping together	Most objects of a type in one Worker Group stopping in the same interval is a mass silence: not counted as zero, reported in Collection health (*objects stopped together*).
Retention of a silent object	Removed 30 days after its last measurement (7 days for a worker process), never while a policy selects it or an unresolved finding names it.
Removed from Cribl	Missing from two consecutive complete Inventory readings: its findings close at once (*removed from Cribl*), its silence is never counted, it leaves *No data* and is purged after **7 days**.
Mass removal	More than half of a Worker Group's objects of one type (at least 5) missing at once are held, not removed, and noted in Collection health.
Forget this object	Purges a silent or removed object at once. TrackMe administrators of the tenant, up to 100 objects per request, audited.

Findings lifecycle

Aspect	Value
Condition	*Failing*, *Recovered*, *Awaiting data* (unknown), *Superseded*.
Status	*Open*, *Investigating*, *Resolved*, *Dismissed*. A measured recovery resolves an open or investigating finding; a new failure reopens it.
Dismiss	Until it recovers (default), for 1, 7, 30 or 90 days (then back to open), or permanently.
Snooze	Up to 30 days; not a status. Stops notifications and colours the object blue in Topology Studio.
Automatic closures	Source or connection changed; object not observed or not evaluated for 24 hours; traffic no longer reported; Worker no longer listed; monitoring turned off; model paused; removed from Cribl.
Finding groups	The findings of one object form one group automatically (up to 50 members); merge, add, remove and ungroup by hand. See *Finding groups*.
Retention	Closed findings and Finding groups are kept 90 days (20 000 of each at most). At most 5 000 unresolved subjects per tenant.

Settings reference

Settings are saved by a TrackMe administrator of the tenant, except the policies (the tenant's admin roles) and the monitoring state (the tenant's power and admin roles).

Section	Setting	Default	Meaning
Source & Worker Groups	Cribl connection	—	Fixed once saved: monitoring another deployment is another tenant.
	Cribl metric index	—	The metric index that receives Cribl's internal metrics.
	Metric prefix	`cribl.logstream.`	Letters, digits and dots, ending with a dot.

continues on next page

Table 1 – continued from previous page

Section	Setting	Default	Meaning
	Worker Groups	all discovered	1 to 100.
	Repair workspace	—	Recreates the tenant's stores and reports; keeps findings and settings.
Collection & models	Scheduled collection	On	Off = manual collection.
	Model learning history	30 days	0 to 90 days; applies the next time the models are relearned.
Traffic ML models	Sensitivity	Normal (3.0)	Low 4.0, High 2.5, or custom 2.0 to 6.0 deviations.
	Seasonality	Automatic	Hour of day, weekday and hour, or none.
	Weekday pattern after	28 days	History needed before the weekday × hour pattern is used.
	Recent days count more	14 days (half-life)	How fast older days fade.
	Open findings once	7 days of history	When a model moves from Low confidence to Detecting.
	Ignore deviations smaller than	10 % and 1 event or byte per 5 minutes	Smallest deviation that counts.
Notifications	Enable notifications	Off	Email delivery of finding transitions.
	Events	opened, reopened, added to the group, escalated, recovered, dismissal ended, closed	Other transitions can be added.
	Minimum severity	Medium and above	Low and above, Medium and above, High and above, Critical only.
	Policies	all	Limit the emails to some policies.
	Delivery account	the first configured account	A TrackMe email delivery account, or *Default (localhost MTA)*.
	Recipients	—	Up to 50, filtered by the account's allowed domains.
	Mode	Digest	One digest per live run, or one threaded email per Finding group.
	Include charts	On	The finding's chart and key measurements of 24 hours, inline.
AI Agent	Scheduled triage	Off	Hourly review of open findings.
	Mode	Inspect (recommend only)	Or *Act (apply decisions)*.
	AI provider	First available provider	The provider of the scheduled runs.
	Review interval (days)	1	0 to 365; a reviewed finding is skipped this long unless a new episode starts.
	Findings per run	10	1 to 50.
	Time budget per run (seconds)	3 000	60 to 86 400.
	Custom instructions	empty	Plain English the agent follows on every run, asked or scheduled (at most 4000 characters); **Evaluate…** tries an unsaved draft in inspect mode and writes nothing.

continues on next page

Table 1 – continued from previous page

Section	Setting	Default	Meaning
	Allowed actions	Mark investigating, Dismiss for a time, Not an anomaly	*Resolve* and *Stop monitoring* are off by default.
	Longest dismissal (days)	7	1 to 90.
	Proven inactivity (days)	7	1 to 365; how long an object must be inactive before the agent may stop monitoring it.
Assignment	Assignment groups, rules	none	See *Assignment and automatic routing*.
Finding actions	Actions, automation rules	none	See *Findings actions*.
Policies tab	Every policy, Expected traffic mode	see *Policy catalogue*	Tenant admin roles.
Inventory tab	Exclusion rules, objects turned off	none	Up to 200 rules and 20 000 objects turned off; tenant power and admin roles.
Cribl connection	Verify SSL certificate, CA certificate chain, requests per hour, burst	On, —, 1 200, 150	Configuration → Cribl connections (*Cribl connections*).

The *Source & Worker Groups* section also has two settings available through the REST API only: the bucket span (300 s) and the settle delay (120 s).

Finding events

Every transition is written as one JSON event in the tenant's **notable index**, whatever the notification settings:

- `sourcetype=trackme:cribl:finding` — one event per finding transition (`event_kind=finding_transition`);
- `sourcetype=trackme:cribl:finding_group` — one event per Finding group transition (`event_kind=finding_group_transition`).

Main fields: `tenant_id` and `object_category` (`cribl_stream`), `finding_id`, `group_id`, `transition` and `transition_label` (*Finding opened*, *Severity raised*, *Measurement recovered*, *Closed automatically*, …), `transition_id` (deduplicate on it), `policy_id` and `policy_name`, `severity`, `display_severity`, `peak_severity`, `condition`, `status`, `reason`, `family`, `worker_group`, `entity_name` (the object), `kpi`, `value`, `expected_lower`, `expected_upper`, `assignee`, `actor`.

```
index=trackme_notable sourcetype=trackme:cribl:finding tenant_id="<tenant>"
    transition IN ("opened", "reopened", "escalated") severity="critical"
```

```
index=trackme_notable sourcetype=trackme:cribl:finding_group tenant_id="<tenant>"
| stats latest(_time) as last_at latest(status) as status latest(severity) as␣
↪severity
        latest(object) as object latest(worker_group) as worker_group by group_id
| where status IN ("open", "investigating")
```

Use them for Splunk alerts, Enterprise Security correlation searches or SOAR playbooks.

Permissions

Permissions combine a TrackMe capability with the tenant's roles. Members of `admin` or `sc_admin` can do everything on every tenant. A TrackMe administrator of one tenant has no right on another.

Tier	Who	Actions
Read	`trackmeuseroperations` and any role of the tenant	Every tab, every chart, the investigation drawer, the AI Assistant, drilldown and link actions.
Triage	`trackmepoweroperations` and the tenant's power or admin role	Status, dismiss, snooze, severity, assignment, notes, Finding groups, Not an anomaly, monitoring on / off and exclusion rules, Stop monitoring this object, Ask the AI Agent and apply its recommendations, send a webhook action.
Policies	`trackmeadminoperations` and the tenant's admin role	Edit policies, the Expected traffic mode and its automatic list. Custom policies: read, write, preview, publish, enable, run and recover.
Manage (a TrackMe administrator of this tenant)	`trackmeadminoperations` and the tenant's admin role	Source settings, Collect now and Run now, model tuning and relearn, Repair workspace, Forget this object, notifications, AI Agent settings, assignment and finding actions.

Tenant owner. The tenant's reports run as its owner: `nobody`, a Splunk administrator, or an account holding `trackmeuseroperations`, `trackmepoweroperations`, `trackmeadminoperations`, `schedule_search`, `list_settings`, `list_storage_passwords` and `admin_all_objects`. The **scheduled** AI Agent needs an owner that is a Splunk administrator account (not `nobody`).

Cribl connections are managed in *Configuration → Cribl connections* (*Cribl connections*).

Configuration Guardian checks

`cribl_collection_health` (tenant scope, warning; evaluated at every tenant health cycle) is raised with one or more of these problems. A disabled tenant, a tenant without a configured source, or a tenant on manual collection reports nothing.

Problem code	Meaning
`live_never_ran` / `live_not_running`	No live metrics run since the source was configured, or none finished in the last 20 minutes.
`live_failing`	Live runs happen but publish nothing, or their traffic scoring fails, for 30 minutes.
`metrics_not_arriving`	No Cribl metric reached the metric index for 30 minutes.
`worker_groups_without_me`	Selected Worker Groups send no metric.
`inventory_never_ran` / `inventory_not_running`	No inventory run since the source was configured, or none finished for 3 hours.
`inventory_failing`	Inventory runs do not complete for 90 minutes.
`inventory_not_published`	The inventory was last published more than a day ago (runs keep waiting for the Leader or do not finish).
`inventory_cycle_slow`	A full inventory cycle takes more than a day under the Leader request budget.
`worker_sampling_slow`	Worker machine samples do not keep up under the Leader request budget.
`worker_directory_stale`	The Worker directory is not refreshed (30 minutes): Worker availability and versions are out of date.
`presence_failing`	The directory of objects that stopped reporting is not kept up to date.
`objects_stopped_together`	Many objects stopped reporting in the same interval (for 30 minutes).
`policies_failing`	Policy evaluation does not complete for 30 minutes.
`models_failing`	Model training does not complete for 90 minutes.
`capacity_reached`	A capacity limit stopped collection, scoring or training, or some traffic series have no model (more series than the tenant keeps models for).
`notification_delivery_failing`	Finding emails fail for 30 minutes.
`custom_schedule_drift`	The scheduled report of an enabled custom policy is missing or differs from its published revision. The tenant health tracker reinstalls it on its next cycle; do not disable the policy to repair it (that closes its findings).
`custom_policy_failing`	A custom policy failed, or returned incomplete results, three runs in a row (a run deferred for capacity does not count).
`custom_missed_schedule`	A custom policy missed three scheduled runs.
`custom_recovery_required`	A custom policy search could not be proven stopped and holds execution capacity: use **Verify and recover** in *Collection health → Custom policies* once it is done.

Notes shown in Collection health that never raise the check: candidates capped, objects removed together, Expected traffic capped or out of time.

System check on Cribl connections:

- `cribl_connection_health` — the daily connection probe fails: warning, critical after 24 hours; visible to administrators.

Also relevant: `insufficient_tenant_owner_capabilities` (the tenant owner lacks a required capability), `ai_automation_owner_is_system_user` (scheduled AI Agent with a `nobody` owner), `findings_action_delivery_failing` (webhooks or other findings actions failing) and `topology_view_stale_references`. See *Configuration Guardian*.

REST endpoints

All endpoints are under `/services/trackme/v2/` and described in the REST API Reference (`describe=true`). Reads run with the caller's session; changes are authorized on the caller's session first, then applied and audited.

Group	Operations (`cribl/tenant/<operation>`)
Workspace and reads	`workspace`, `evidence`, `inventory`, `workers`, `summary`, `health_summary`, `findings`, `finding`, `finding_groups`, `finding_group`, `traffic_models`, `traffic_model`, `tuning`, `presence`, `policies`, `expected_auto`, `monitoring`, `object_status`, `agent_settings`, `agent_findings`, `agent_evidence`, `agent_decisions`
Triage	`triage`, `group_triage`, `group_members`, `not_anomaly`, `monitoring` (POST), `monitoring_rules`, `monitoring_preview`, `finding_stop_monitoring`, `agent_decide`, `agent_review`
Policies	`policies` (POST), `policy_preview`, `expected_auto_preview`, `expected_auto_override`; custom policies `custom_policies`, `custom_policy`, `custom_policy_health` (reads), `custom_policy_write` (create, save, restore, publish, enable, disable, archive, delete, delete a version) and `custom_policy_execution` (preview or run, status, cancel, recover)
Manage	`configure`, `collect`, `collect_result`, `models`, `repair`, `tuning` (POST), `forget_objects`, `notifications` (GET and POST), `notification_test`, `agent_settings` (POST)
Other	`cribl/admin/connection_health` (administrators); the AI Agent jobs `ai_findings_agent/agent`, `agent_status`, `agent_cancel`; findings actions `findings_actions/*`; the Cribl connections through the configuration endpoint `trackme_cribl_connection`.

Capacity

Limit	Value
Monitored objects per tenant (Sources, Destinations, Pipelines, Routes, worker processes in the snapshot)	**About 20 000**; split a larger deployment across tenants by Worker Group.
Worker Groups per tenant	100
Traffic ML models	20 000
Models with their own settings	500
Unresolved findings (subjects)	5 000; beyond, new failures are refused while every recorded one keeps being evaluated.
Closed findings and Finding groups kept	20 000 each, 90 days
Selected objects per policy	128
Exclusion rules / objects turned off one by one	200 / 20 000
Notification recipients	50
Custom policies per tenant / enabled	100 / 25 (see *Custom policies*)
Forget this object	100 objects per request

Models need about a week of history before they open findings, unless the metric index already holds that history at creation.

Logs

Every line starts with `tenant_id="<tenant>"`.

Log	Sourcetype	Contains
`trackme_cribl_collect.log`	`trackme:custom_commands:`	One summary line per run; WARNING when a run did not complete; *Cribl collector failure* for an internal error.
`trackme_cribl_tune.log`	`trackme:custom_commands:`	Failures of the tuning simulation.
`trackme_cribl_notify.log`	`trackme:custom_commands:`	One line per test email.
`trackme_cribl_custom.log`	`trackme:custom_commands:`	One line per custom policy run or preview: status, rows, findings, committed, unresolved, diagnostics (never the SPL or the result values).
`trackme_rest_api_cribl.log`	`trackme:rest_api`	Degraded paths of the REST API.

```
index=_internal sourcetype=trackme:custom_commands:trackmecriblcollect␣
↪tenant_id="<tenant>"
```

```
index=_internal sourcetype=trackme:custom_commands:trackmecriblcollect␣
↪tenant_id="<tenant>" job="live"
| stats count, latest(_time) as last, values(reason) as reasons by complete
```

Tokens, passwords, email addresses, request bodies and search strings are never logged.

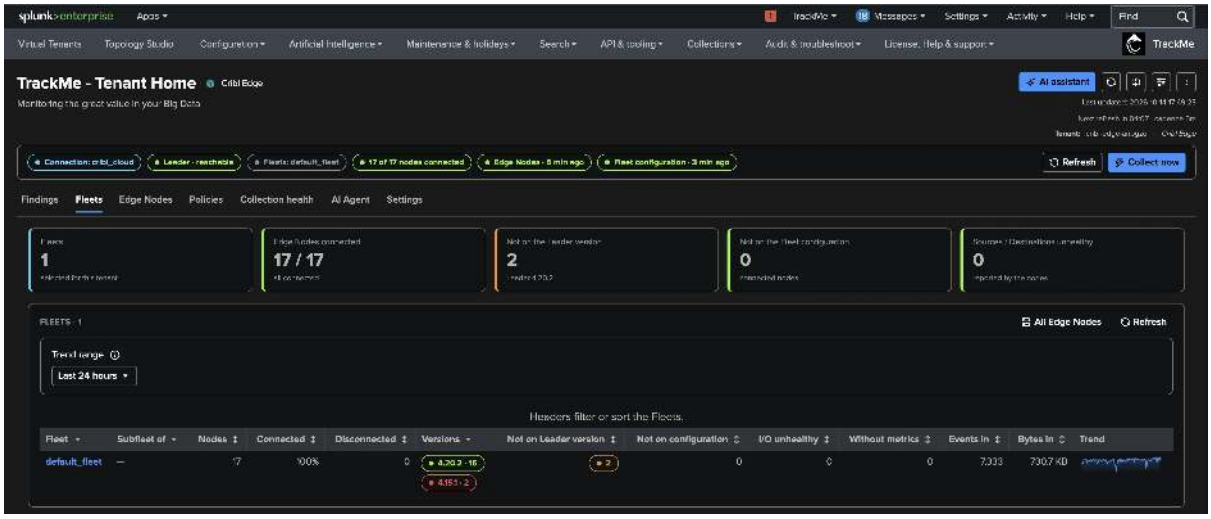
Troubleshooting

Symptom	Likely cause	Fix
The wizard proposes no metric index	Cribl's internal metrics do not reach a Splunk metric index, or they use another prefix.	Route Cribl's internal metrics to a Splunk metric index; set the metric prefix your deployment uses.
Metrics not arriving in Collection health	The metrics stopped reaching the index, or the index or prefix is wrong.	Check the Cribl Splunk Destination and the metric index; compare the prefix in Settings.
Worker Groups without metrics	A selected Worker Group does not send its internal metrics.	Enable the internal metrics on that Worker Group, or deselect it.
Leader · unavailable, a Critical *Leader availability* finding	Network, certificate, credentials or service problem on the Leader API (the tooltip names it).	Certificate not trusted or host name mismatch: add the CA chain to the connection. Credentials refused: update them. Then test the connection.
Leader · rate limited, deferred runs	The Leader asked TrackMe to slow down, or the request budget is spent.	Wait; lower the connection's requests per hour if the Leader is shared; check the Cribl Leader card in Collection health.
Inventory cycle slow / *Worker sampling slow*	The deployment needs more requests than the Leader budget allows.	Raise the connection's requests per hour if the Leader can take it, or split the deployment across tenants.
Traffic outliers open nothing	Models are still *Learning* or *Low confidence*.	Wait until they are *Detecting* (7 days of history), or recreate with a longer *Model learning history*.
Too many Traffic outliers findings	The models are too sensitive for this traffic.	Use *Not an anomaly* on expected episodes; lower the sensitivity or raise *Ignore deviations smaller than*; limit the direction to drops.
Worker memory findings on healthy hosts	Memory used includes the file cache.	Keep *Worker machine memory* off, or raise its threshold.
A deleted object still shows under *No data*	The Inventory has not seen it missing twice yet, or it was turned off.	Wait for the next inventory cycle, or **Forget this object**.
The scheduled AI Agent does not run	AI Assistant disabled, no provider, or a `nobody` tenant owner.	Enable the AI Assistant, configure a provider, and give the tenant a Splunk administrator owner.
No email	Notifications off, events or severity filtered out, or delivery failing.	Check *Settings → Notifications*, then **Send test email**; see `notification_delivery_failing`.

See also

- *Cribl Stream* — the Cribl Stream user guide.
- *Cribl connections* — Cribl connections, TLS and the Leader budget.
- *Findings, Finding groups and findings actions* — findings-based monitoring.
- *Cribl Edge reference* — the Cribl Edge reference.

Cribl Edge



The **Cribl Edge tenant** follows a Cribl Edge deployment from TrackMe: the **Fleets** of one Cribl Edge Leader, the **Edge Nodes** enrolled in them, and the **Sources** and **Destinations** those nodes run. It answers the questions an Edge operator asks every day: *are my nodes connected? did my last deploy reach every node? is every node on the same Edge version? which Source or Destination is failing, and on how many nodes? which host is running out of disk?* The answers are **findings**, raised by built-in policies, with the same triage, Finding groups, notifications, assignment and findings actions as the Cribl Stream tenant, and a findings AI Agent of its own, the Cribl Edge AI Agent.

A Cribl Edge tenant is a **dedicated tenant type**: one tenant follows the Fleets of one Cribl Edge Leader, on Cribl.Cloud or self-managed, through a *Cribl connection*. A tenant is a Cribl Edge tenant or a Cribl Stream tenant, never both; a Cribl.Cloud workspace that runs both products takes one tenant of each, on the same connection. TrackMe only reads: nothing is installed or changed on Cribl.

The Cribl Edge tenant is generally available in TrackMe 2.4.19.

Every node, up close. Open any Edge Node from the workspace and its **System activity** shows what Cribl's own System Activity page shows for that host — CPU by mode, memory, network, disk I/O and load over the last 15 minutes to 12 hours — read live from the node through the Leader, with nothing to forward and nothing to configure. The fleet-wide view stays cheap; the detail of one node is one click away.



What it monitors

Object	What TrackMe watches
Fleets (and Subfleets)	**Connectivity**: the share of the Fleet's active nodes that are disconnected. **Version drift**: connected nodes not on the Leader's Edge version (or the Fleet's most common one) once a rollout grace has passed. **Configuration drift**: connected nodes not running the Fleet's deployed configuration. **Coverage**: a Fleet with fewer connected nodes than expected, or emptied. **Node identity**: one host name reported by several node ids (cloned images), or a node id that keeps changing host name.
Sources and Destinations of each Fleet	The **health** each node reports for the Sources and Destinations it runs (healthy, warning, trouble) and their traffic, rolled up per Fleet: "this Destination is in trouble on 4 of 17 nodes".
Edge Nodes	**Connection** (Leader state and heartbeat) for the nodes that must never drop, **disk** used on every node, and **CPU** and **memory** when the Fleets forward their Edge System Metrics to Splunk (a System Metrics index, optional).
The Cribl Leader	Whether TrackMe can reach and use the Leader API at every run (the same Leader availability policy as Cribl Stream).

Built for any fleet size

An Edge fleet can hold a handful of servers or tens of thousands of endpoints. The Cribl Edge tenant is designed so that its cost does not grow with one call per node:

- **The node directory is read in bulk.** At every live run, TrackMe reads the Leader's list of Edge Nodes page by page and keeps it as one published **node directory**. There is **no API call per node**, and no rotation: every node is current within one live interval (5 minutes).
- **No node limit.** Nothing caps the number of nodes a tenant follows. Fleet figures, policies, the Edge Nodes tab and Topology Studio all read the directory in pages, so memory and work stay bounded whatever the fleet size.
- **Fleet-level judgement first.** Disconnections, version and configuration drift and Source / Destination health are judged per Fleet, as a share of its nodes. One laptop that goes to sleep does not open a finding; a fifth of a Fleet dropping does.
- **Per-node detail on demand.** A node's CPU, memory, network and disk activity, and the live status of one of

its Sources, are read from the Leader only when someone opens that node in the workspace.

No machine learning for Edge

The Cribl Edge tenant has **no traffic ML models**, no model training job and no Expected traffic policy. Edge nodes are often endpoints whose activity is sparse and irregular, and there can be very many of them: a model per node and object would cost far more than it would detect. Edge relies instead on explicit, explainable rules (shares of a Fleet, graces, thresholds) and on the health Cribl itself reports for each Source and Destination.

Before you start

A licence. A valid **Enterprise**, **Unlimited**, trial or **Developer** licence. The Cribl Edge tenant is not available with the Foundation edition or the Foundation trial.

A Cribl connection to the Edge Leader, created by a Splunk administrator in **Configuration → Manage Cribl connections**: a Cribl.Cloud workspace (organisation, workspace, client id and secret) or a self-managed Leader (URL, credentials, and for TLS **Verify SSL certificate** with an optional CA chain). The Cribl account behind it needs **read access** to the Cribl Edge API (Fleets, Edge Nodes, the configuration of each Fleet, and the Leader-proxied reads of a node). See *Cribl connections*. A self-managed Leader that this instance cannot reach, such as from Splunk Cloud, can go *through a TrackMe Broker*.

A tenant owner: `nobody`, or an account (typically a service account) holding the capabilities the creation wizard lists, so that the tenant's scheduled reports run. The scheduled triage of the Cribl Edge AI Agent needs more: an owner that is a Splunk administrator account, not `nobody` (see *The Cribl Edge AI Agent*).

Optionally, a System Metrics index. Cribl Edge reports no host CPU or memory with the node heartbeat, and TrackMe never reads nodes one by one. To monitor the CPU and memory of your Edge Nodes continuously, the Fleets forward their **Edge System Metrics** to a Splunk **metrics** index, and the tenant reads it:

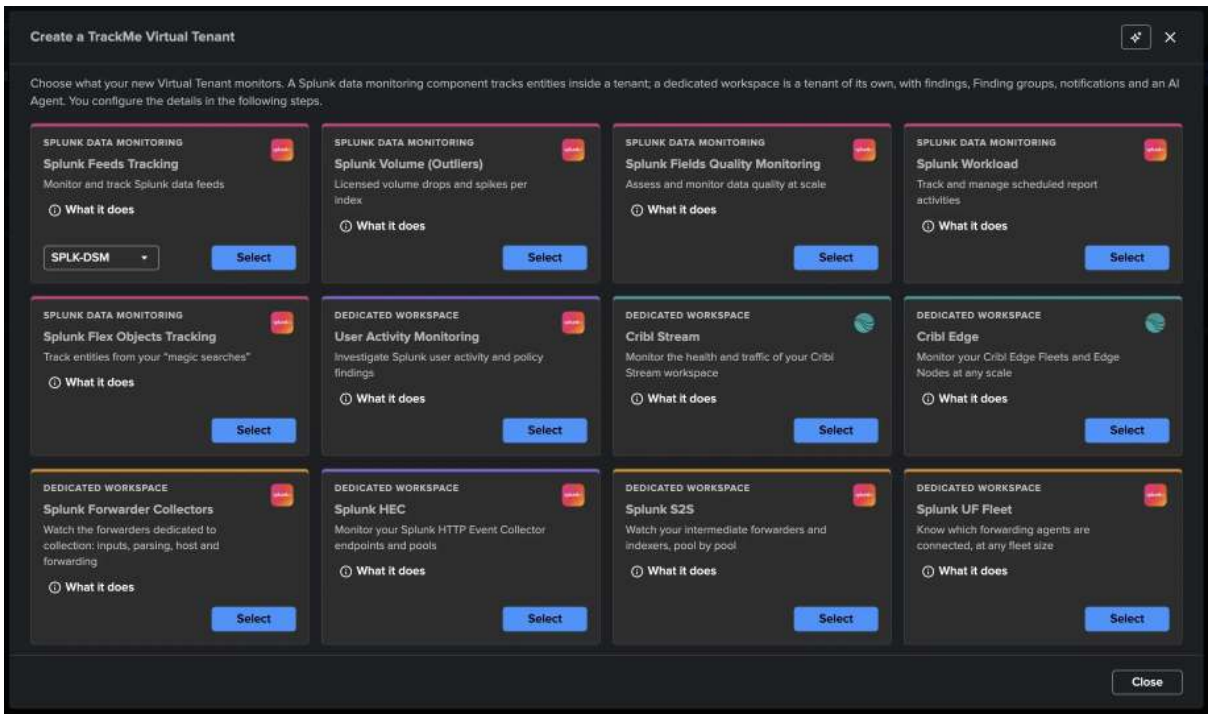
1. In Cribl Edge, turn on the **System Metrics** source of your Fleets (the Basic level is enough).
2. Route its metrics to Splunk, directly or through Cribl Stream, into a Splunk **metrics** index. Keep the metric names (`node_cpu_percent_active_all`, `node_memory_Used_percent`) and the `host` dimension, the node's host name.
3. Select that index as the tenant's **System Metrics index**, in the wizard or afterwards in **Settings → Source & Fleets**.

The System Metrics index feeds the **Node CPU** and **Node memory** policies and the CPU and memory KPIs of Edge Nodes in Topology Studio. Since 2.4.25 a node's System activity tab reads it first too, which answers in seconds. **Without it, everything else works the same**: connectivity, versions, configuration, Sources and Destinations, node disk (which comes with every heartbeat), and the node's System activity tab, which then reads the node's own metrics through the Leader when you open it.

No other index is needed: the history TrackMe keeps for the Fleets and the nodes (the trends of the workspace, the disk history) goes to the tenant's TrackMe metric index, chosen in the wizard like for any tenant.

Creating the tenant

From the *Virtual Tenants* page, open **Actions → Create a tracking tenant** and choose the **Cribl Edge** card. The wizard has four steps.

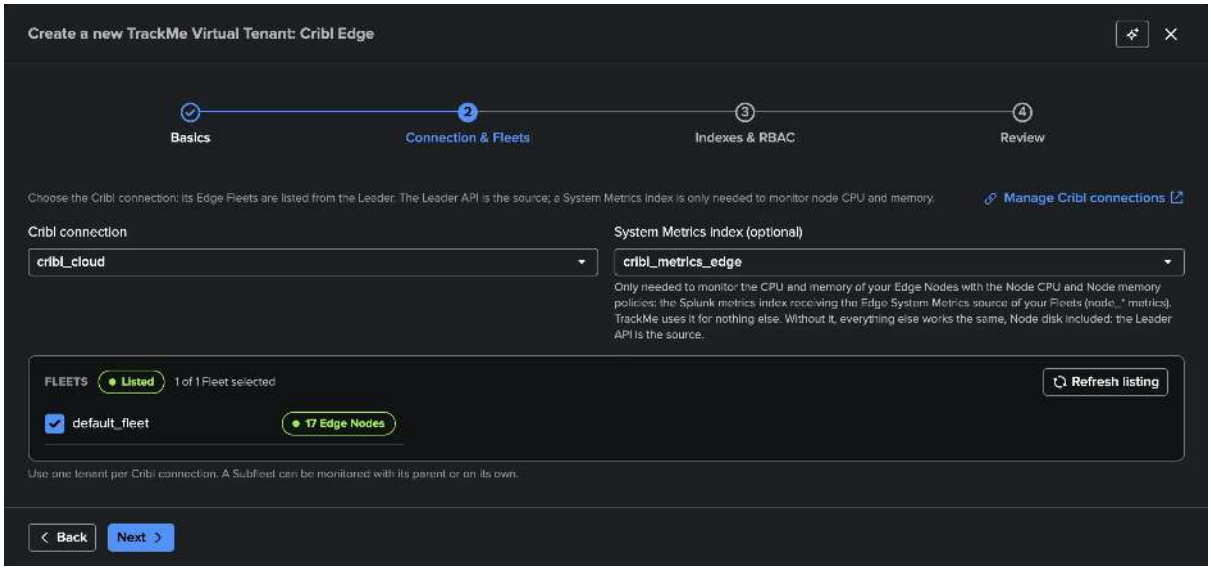


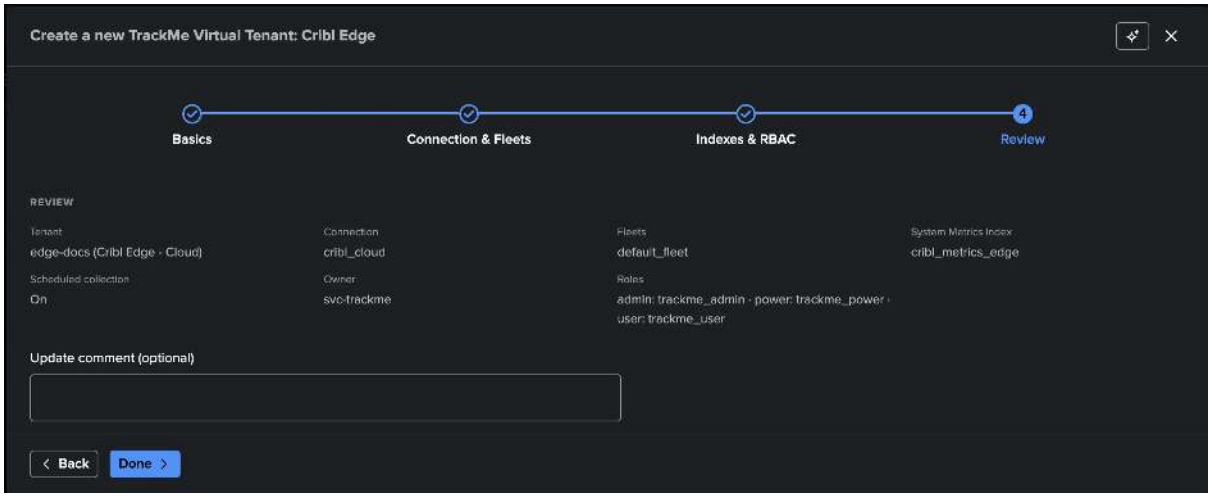
1. Basics. The **tenant identifier** (lowercase letters, digits and hyphens, up to 20 characters; it cannot be changed afterwards), an optional **alias** and **description**.

2. Connection & Fleets.

- Choose the **Cribl connection**. It is locked once the tenant exists: another Leader takes another tenant.
- TrackMe lists the **Fleets** from the Leader. On a first listing every Fleet is selected; clear the ones you do not want (1 to 100 Fleets per tenant). Each row shows its number of nodes, and a Subfleet shows "Subfleet of <parent>": a Subfleet can be monitored with its parent or on its own. **Refresh listing** lists them again.
- Choose the **System Metrics index (optional)**, or keep "None: no node CPU or memory monitoring".

The step waits for the Fleet listing. When the Leader cannot list the Fleets, the step says why in plain words (credentials refused, no right to read Cribl Edge, Leader unreachable, certificate not trusted, Leader busy) and what to do.





3. Indexes & RBAC. The tenant's indexes (summary, audit, metric, notable), its admin, power and user roles, and its owner, as for any tenant.

4. Review. Check the summary and create the tenant. TrackMe creates the tenant's collections and reports and starts the first collection: the Fleets and Edge Nodes appear in the workspace within a few minutes (**Open workspace**).

Collection is scheduled from the start: the Edge Nodes are read and the policies evaluated every 5 minutes, the Fleet configuration about once an hour. To pause it, turn off **Scheduled collection** in **Settings → Collection**.

The workspace

Open the tenant from Virtual Tenants: Tenant Home shows the **Cribl Edge** workspace. It is built with the same components as the Cribl Stream workspace, so the two look and behave the same.

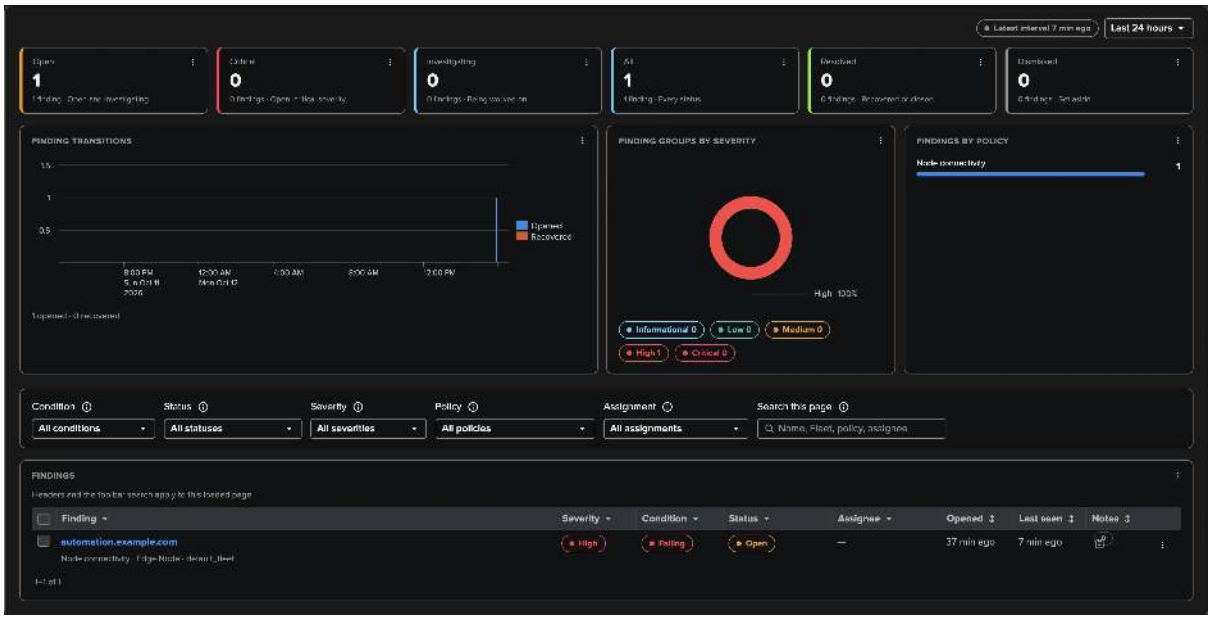


The status strip sums the tenant up: the connection, whether the **Leader** is reachable (with the time of the last check), the Fleets followed, **"N of M nodes connected"** from the latest node directory, and the age of the last **Edge Nodes** run and **Fleet configuration** run. **Refresh** reloads the workspace; **Collect now** (TrackMe administrators of the tenant) reads the Edge Nodes from the Leader and evaluates the policies at once.

Tab	What it is for
Findings	The open findings and Finding groups of the tenant, their triage, assignment and findings actions.
Fleets	One row per Fleet: connected nodes, versions, drift, Sources / Destinations health, traffic and trend. Opens the Fleet.
Edge Nodes	Every node, filtered and paged by the server. Opens the node: Node info, System activity, Data activity.
Policies	The catalogue of the eleven built-in policies, their severity, conditions and scope.
Collection health	Whether collection runs as configured, job by job, and the load on the Leader.
AI Agent	The Cribl Edge AI Agent: whether it can run, its totals, **Run a review now** and its decision log.
Settings	Source & Fleets, Collection, Notifications, AI Agent, Assignment, Finding actions.

Findings

The Findings tab is the findings centre shared by every dedicated tenant type: KPI tiles, the transitions over time, the open findings and Finding groups with their severity (Informational, Low, Medium, High, Critical), condition, status and assignee. Each row's menu investigates, triages (investigate, snooze, dismiss for a time or until it recovers, resolve, change the severity), assigns, runs a findings action, asks the AI Agent (**Ask the AI Agent…**), and **Open resource** opens the Fleet or the Edge Node the finding is about, in place. See *Findings in practice* below and *Findings, Finding groups and findings actions*.



Fleets

The tiles answer the fleet questions at a glance: **Fleets** selected, **Edge Nodes connected** (click to list the disconnected ones), nodes **Not on the Leader version**, nodes **Not on the Fleet configuration**, **Sources / Destinations unhealthy** (reported by the nodes) and, when it happens, **Connected without metrics**. The table has one row per Fleet, with its Subfleet parent, nodes, connected share, disconnected nodes, the versions running (green on the Leader version, amber otherwise, red when a version is newer than the Leader or two minor versions behind), drift counts, unhealthy Sources / Destinations, nodes without metrics, events and bytes in, and a traffic trend over the chosen range. **All Edge Nodes** opens the Edge Nodes tab.

Every count of the table opens what is behind it (2.4.25). Disconnected, a version, not on the Fleet configuration and without metrics list those nodes: each node opens its detail, and **Show in Edge Nodes** carries the filter to the Edge Nodes tab. Unhealthy Sources / Destinations open the Fleet's warning and trouble objects with the nodes reporting them.

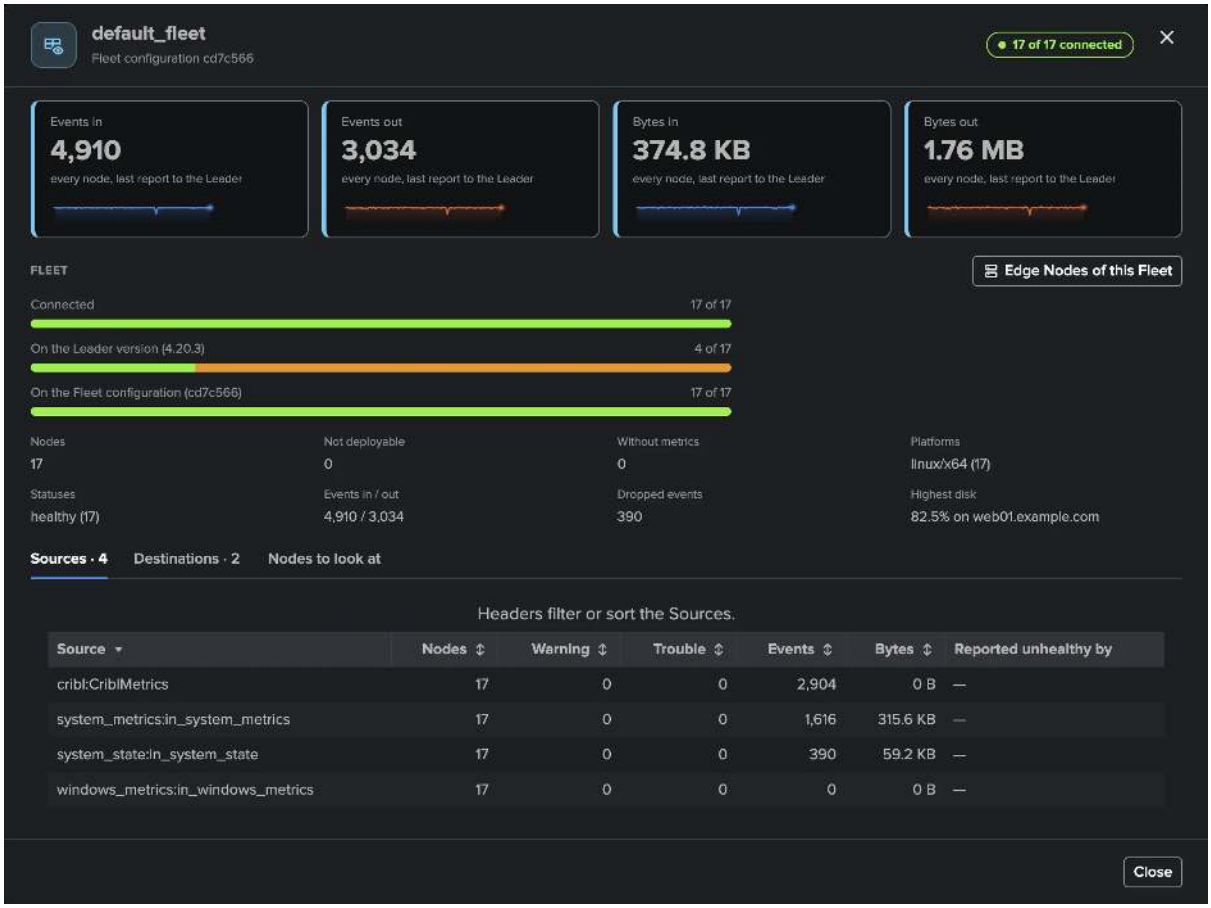
Fleet versions (2.4.25) answers what your Edge Nodes run. It opens from its button on the Fleets panel, from the **Not on the Leader version** tile, or from a Fleet's version drift, then scoped to that Fleet:

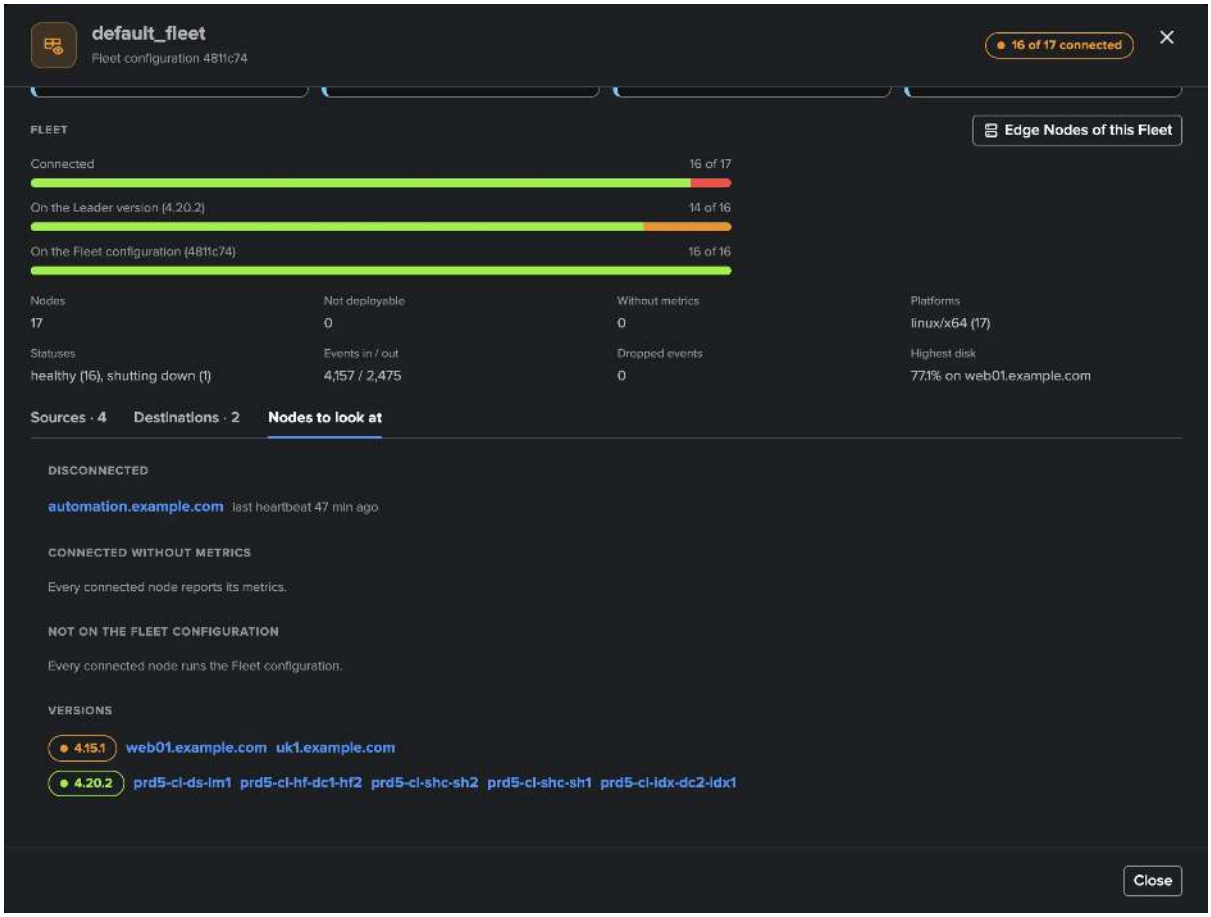
- **Overview**: the Cribl Edge versions, operating systems, OS releases and architectures of the nodes, with how many are on the Leader version;
- **Operating systems**: every operating system, release and architecture with the Edge versions running on it, searchable, with a CSV export;
- **Versions**: each Edge version compared with the Leader's, and how far behind it is;
- **Fleets**: the version mix of each Fleet;
- **Upgrades**: the nodes that changed version over the last 24 hours, 7 days and 30 days, with their previous and new versions;
- **History**: the number of nodes per Edge version and per OS release over time.

These figures are counted when the node directory is built, so the view opens at once whatever the fleet size. After an upgrade to 2.4.25, a Fleet is counted by its next live run, and the history starts from there.

Select a Fleet to open it:

- four traffic tiles (events and bytes in and out, each with its trend) and the Fleet's bars: connected, on the Leader version, on the Fleet configuration;
- its facts: nodes, not deployable, without metrics, platforms, statuses, dropped events, highest disk and on which node;
- three tabs: **Sources** and **Destinations** (per object: nodes reporting it, warning and trouble counts, events, bytes, and the nodes reporting it unhealthy) and **Nodes to look at** (disconnected nodes with their last heartbeat, connected nodes without metrics, nodes not on the Fleet configuration, and the nodes behind each version). Every node name opens the node; **Edge Nodes of this Fleet** lists them all.



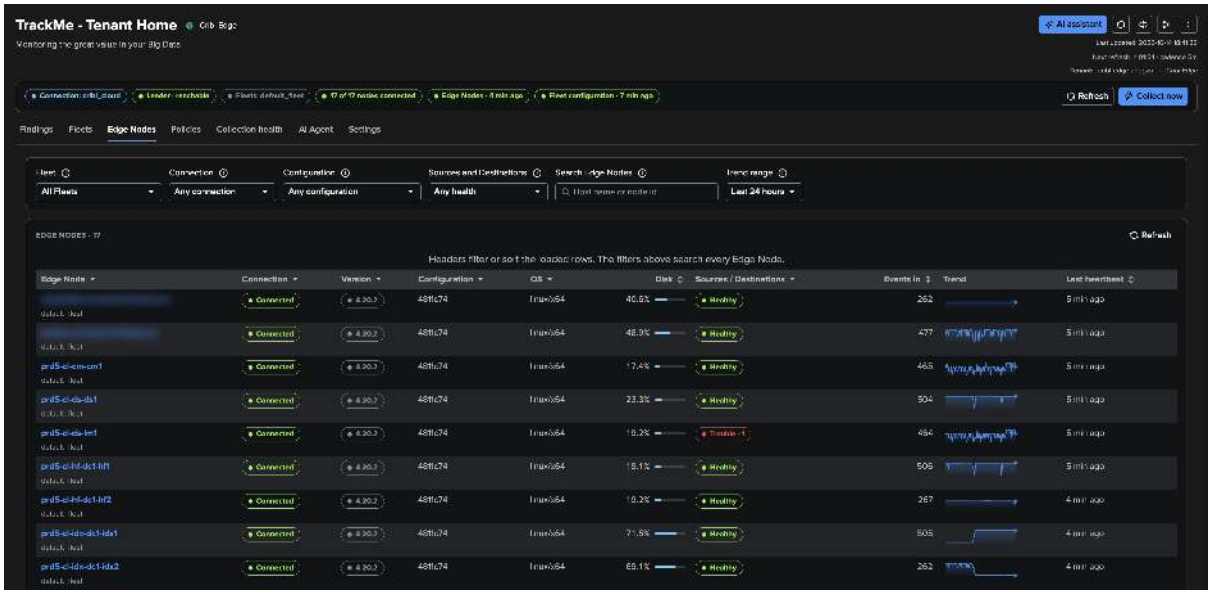


The figures of a Fleet are the sum of each node's **last report to the Leader** (Cribl measures it over a short period): they say how busy the Fleet is now, not volume totals. The trends sample that report at every collection.

Edge Nodes

Every Edge Node of the selected Fleets, **paged and filtered by the server**, so the tab stays fast with any number of nodes. Filters: **Fleet**, **Connection** (connected, disconnected, unknown), **Configuration** (on, not on the Fleet configuration, unknown), **Sources and Destinations** (the worst health a node reports), a **search** on host name or node id that applies to every node, and the **trend range**. The columns: node and Fleet, connection (with how long a node has been disconnected), version, configuration, OS, disk, Sources / Destinations health, events in, trend and last heartbeat.

A node is **disconnected** when the Leader says so, or when it has sent no heartbeat for five heartbeat periods (10 minutes at least). A connected node whose Leader record carries no metric shows **No metrics**: its Sources, Destinations and traffic are not measured, and TrackMe never lets that read as healthy.



The Edge Node detail

Select a node's name to open it (its Connection pill opens it on Node info, its Sources / Destinations pill on Data activity). The header names the node (its host name, else its node id), its OS and Edge version, and carries three pills: the **connection** (Connected / Disconnected), the **Fleet**, and the worst **health** of its Sources and Destinations (Healthy, Warning, Trouble — or *No metrics*). Three tabs follow.

Node info — four tiles: **Connection** (with the last heartbeat), **Edge version** (compared with the Leader's, flagged when newer than the Leader or two minor versions behind), **Configuration** (compared with the Fleet's target) and **Sources / Destinations** (how many, how many unhealthy). Then three blocks of facts: **Identity** (host name, node id, Fleet, whether another node id reports the same host name or the host name keeps changing, tags, first seen), **Deployment** (Edge and Leader versions, configuration, deployable, Cribl status, last heartbeat, heartbeat period) and **Machine** (operating system, release, architecture, CPUs, memory).

System activity — what Cribl's own System Activity page shows for this node:

- **Range**: *Last 15 minutes*, *Last 1 hour*, *Last 4 hours* or *Last 12 hours*, with *Read N ago* and **Refresh**.
- **Tiles**: **CPU usage** (with the average busy share and the number of CPUs), **Memory used** (used of total), **Load (1 min)** (with the 5- and 15-minute loads), **Network** (in and out per second) and **Disk used** (free of total, from the heartbeat).
- **Charts**, each with its panel menu (**View chart data**, and *Open in Search* when the figures come from the System Metrics index rather than from the node): **CPU usage** stacked by mode (User, System, I/O wait, Other — a single *Busy* line when the Fleet's System Metrics run in Basic mode), **Memory used** (Used, Cached), **Network** per second (Received, Sent; a **Bytes / Packets** toggle), **Disk I/O** per second (Read, Written), **Load average** (1, 5, 15 min) and **Processes and threads**. Units follow the values (KB/s, MB/s…).
- **Facts**: CPU (CPUs, processes and threads, busy and idle, the share of each mode, the three loads), Memory (total, used and available, cached, swap) and Network and disk (sent and received, packets, disk read and written).
- **Disk used** · **TrackMe history**: the disk of this node over any period, from the history TrackMe keeps for every node.

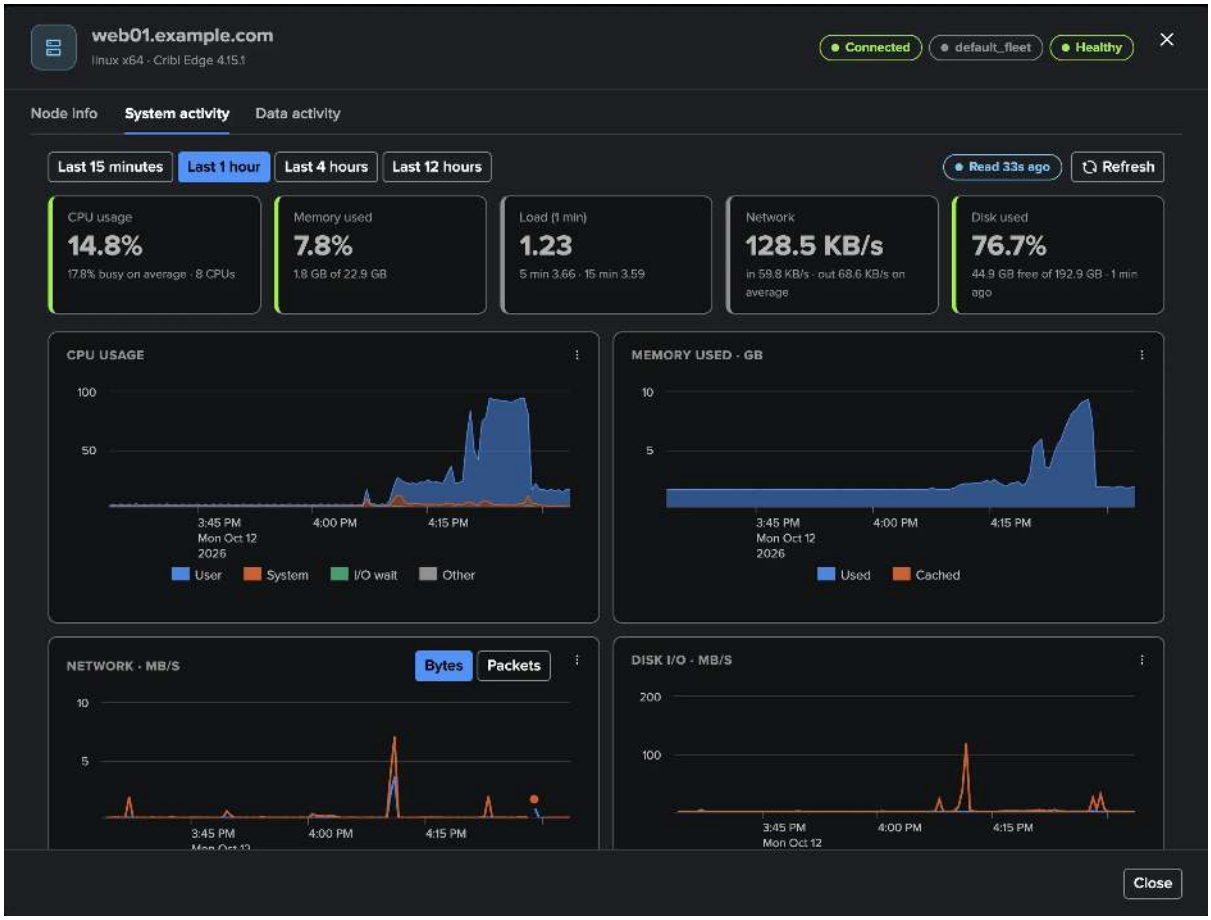
Where these figures come from. When the tenant has a **System Metrics index** (2.4.25), the tab reads it first: one search you run on the metrics the node forwarded, which answers in seconds and shows a **System Metrics index** pill. The node is matched by its host name. When the index holds nothing for that node (its Fleet does not forward the source, or names the host otherwise), the tab says so and reads the node itself.

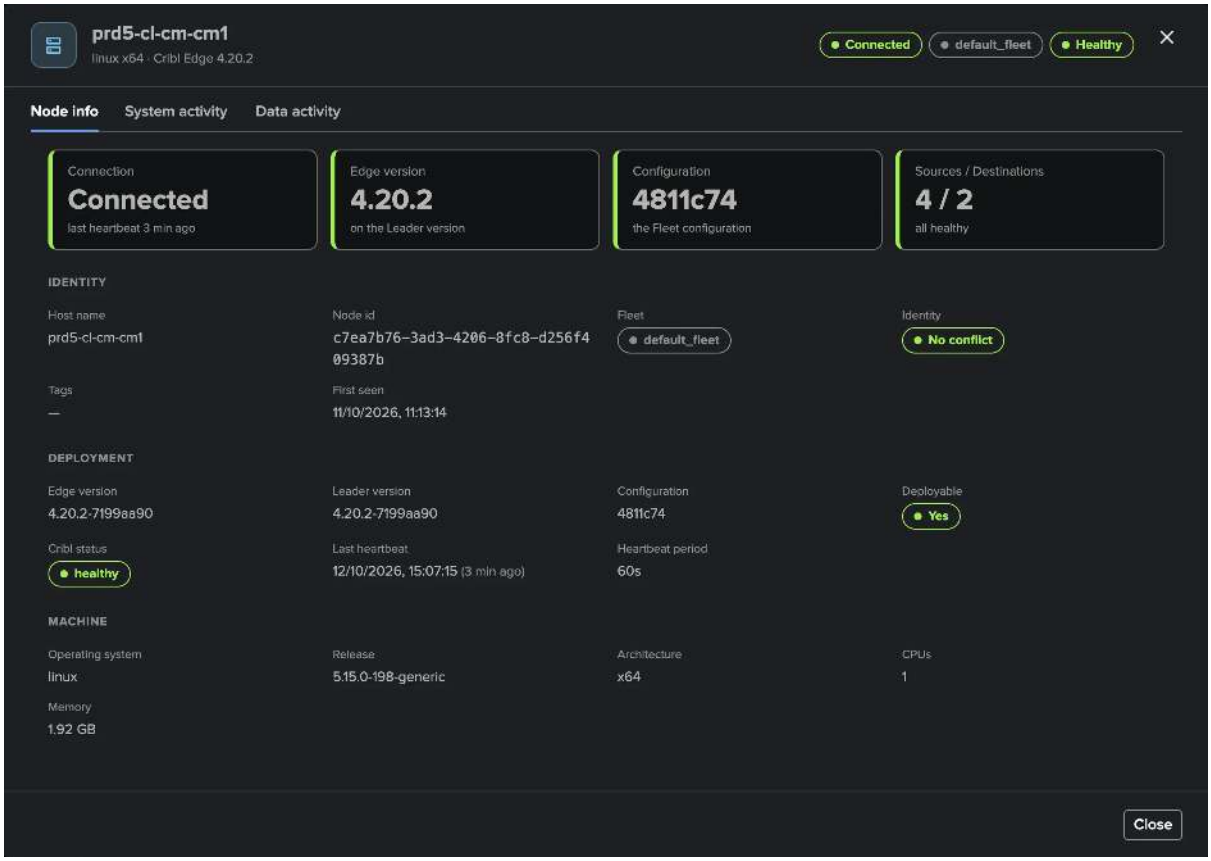
Without a System Metrics index, CPU, memory, network, disk I/O, load and processes are read **live from the node itself**, through the Cribl Leader: each Edge Node keeps its own store of its system metrics (at 10-second resolution, about the last 10 hours), and TrackMe queries that store — the same query Cribl's System Activity page makes —

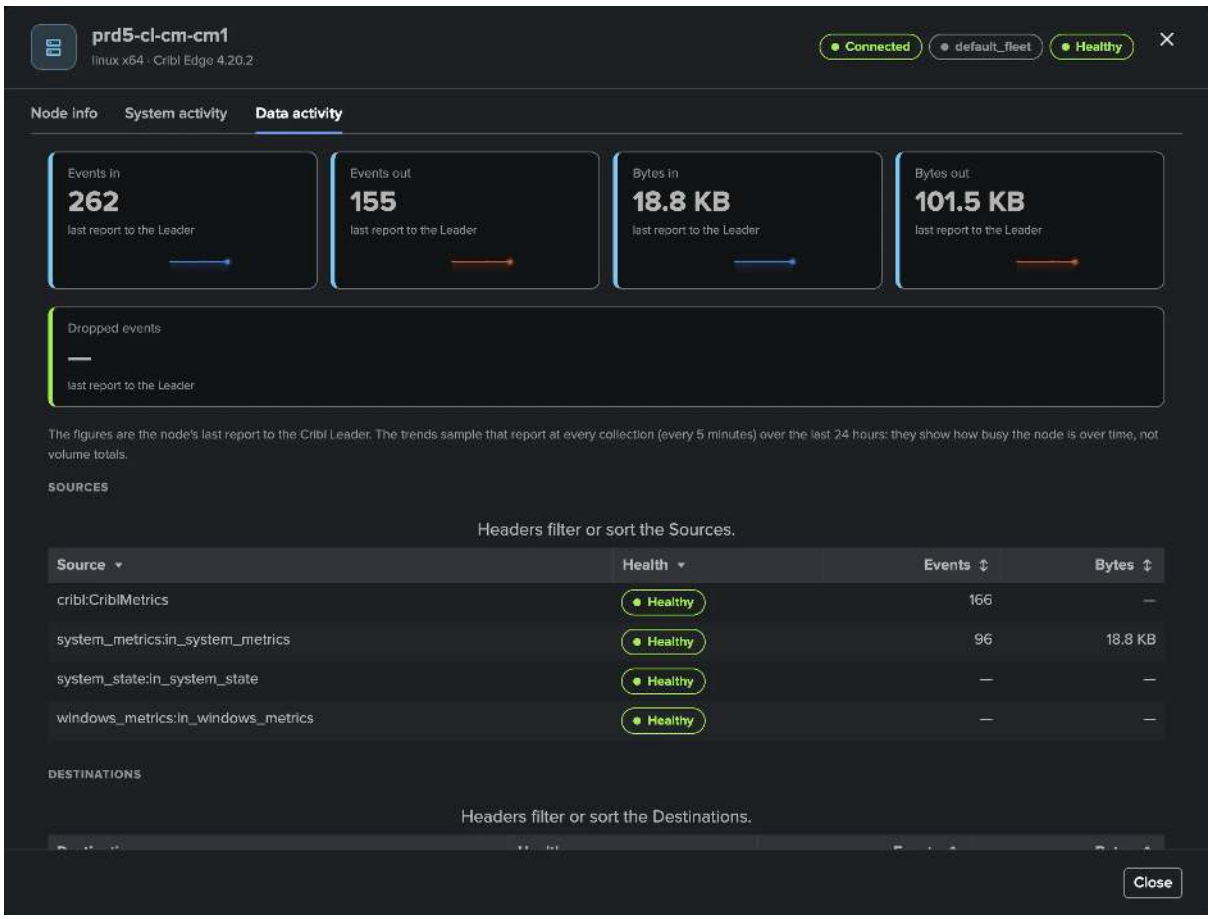
when you open the tab or change the range. This read needs no System Metrics forwarded to Splunk. Nothing is read for a node nobody opens, so the fleet-wide collection stays one bulk read whatever the fleet size. A node whose System Metrics source is off in its Fleet keeps no such store: TrackMe then reads its current figures only (CPU, memory, load, uptime), without charts. The **disk** is different: it comes with every heartbeat, for every node, and TrackMe keeps its history. A disconnected node cannot be read.

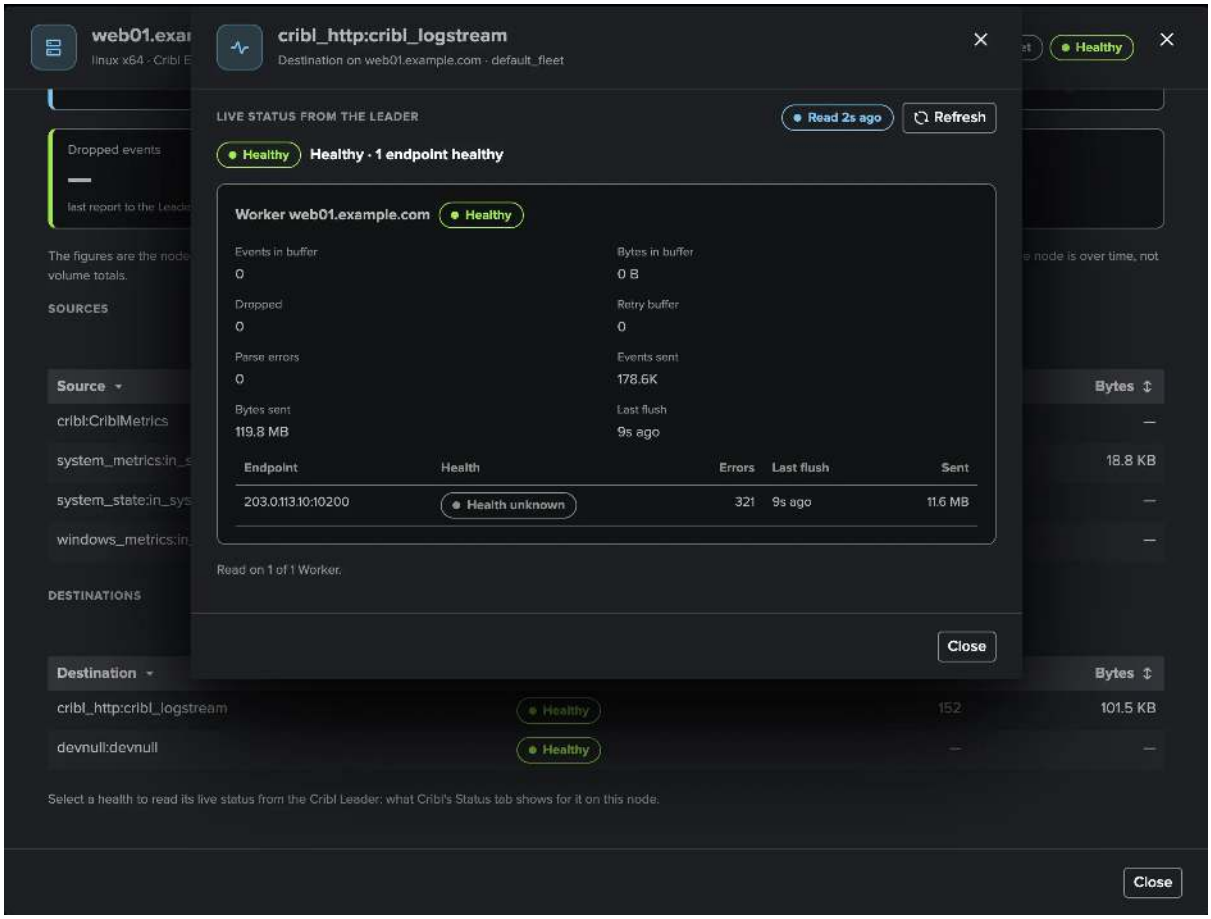
A read asked by a TrackMe administrator of the tenant starts at once (a search reads the node and the tab follows it); any other role's read is served by the next live run, within 5 minutes. The reads are counted in the Leader's request budget, at most one per node and range a minute.

Data activity — four traffic tiles (events and bytes in and out) with their 24-hour trends and **Dropped events**, all from the node's **last report to the Leader**; then the node's **Sources** and **Destinations** with their health (Healthy, Warning, Trouble) and traffic. Select a health to read the **live status** of that Source or Destination on that node from the Leader, as Cribl's Status tab shows it.




prd5-cl-cm-cm1
linux x64 · Cribl Edge 4.20.2
Connected
default_fleet
Healthy
Node info
System activity
Data activity
Connection
Connected
last heartbeat 3 min ago
Edge version
4.20.2
on the Leader version
Configuration
4811c74
the Fleet configuration
Sources / Destinations
4 / 2
all healthy
IDENTITY
Host name
prd5-cl-cm-cm1
Node id
c7ea7b76-3ad3-4206-8fc8-d256f409387b
Fleet
default_fleet
Identity
No conflict
Tags
—
First seen
11/10/2026, 11:13:14
DEPLOYMENT
Edge version
4.20.2-7199aa90
Leader version
4.20.2-7199aa90
Configuration
4811c74
Deployable
Yes
Cribl status
healthy
Last heartbeat
12/10/2026, 15:07:15 (3 min ago)
Heartbeat period
60s
MACHINE
Operating system
linux
Release
5.15.0-198-generic
Architecture
x64
CPUs
1
Memory
1.92 GB
Close


prd5-cl-cm-cm1
linux x64 · Cribl Edge 4.20.2
Connected
default_fleet
Healthy
Node info
System activity
Data activity
Events in
262
last report to the Leader
Events out
155
last report to the Leader
Bytes in
18.8 KB
last report to the Leader
Bytes out
101.5 KB
last report to the Leader
Dropped events
—
last report to the Leader
The figures are the node's last report to the Cribl Leader. The trends sample that report at every collection (every 5 minutes) over the last 24 hours: they show how busy the node is over time, not volume totals.
SOURCES
Headers filter or sort the Sources.
Source
Health
Events
Bytes
cribl:CriblMetrics
Healthy
166
—
system_metrics:in_system_metrics
Healthy
96
18.8 KB
system_state:in_system_state
Healthy
—
—
windows_metrics:in_windows_metrics
Healthy
—
—
DESTINATIONS
Headers filter or sort the Destinations.
Close

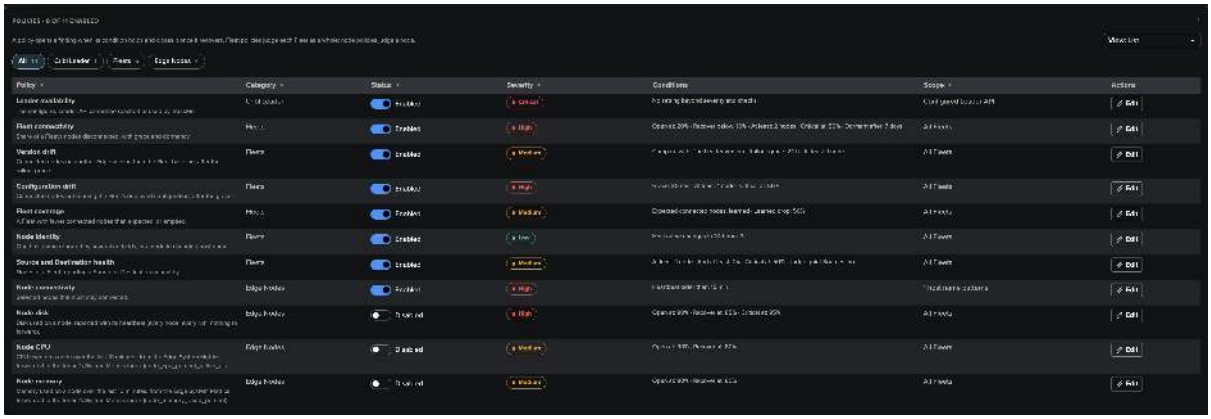


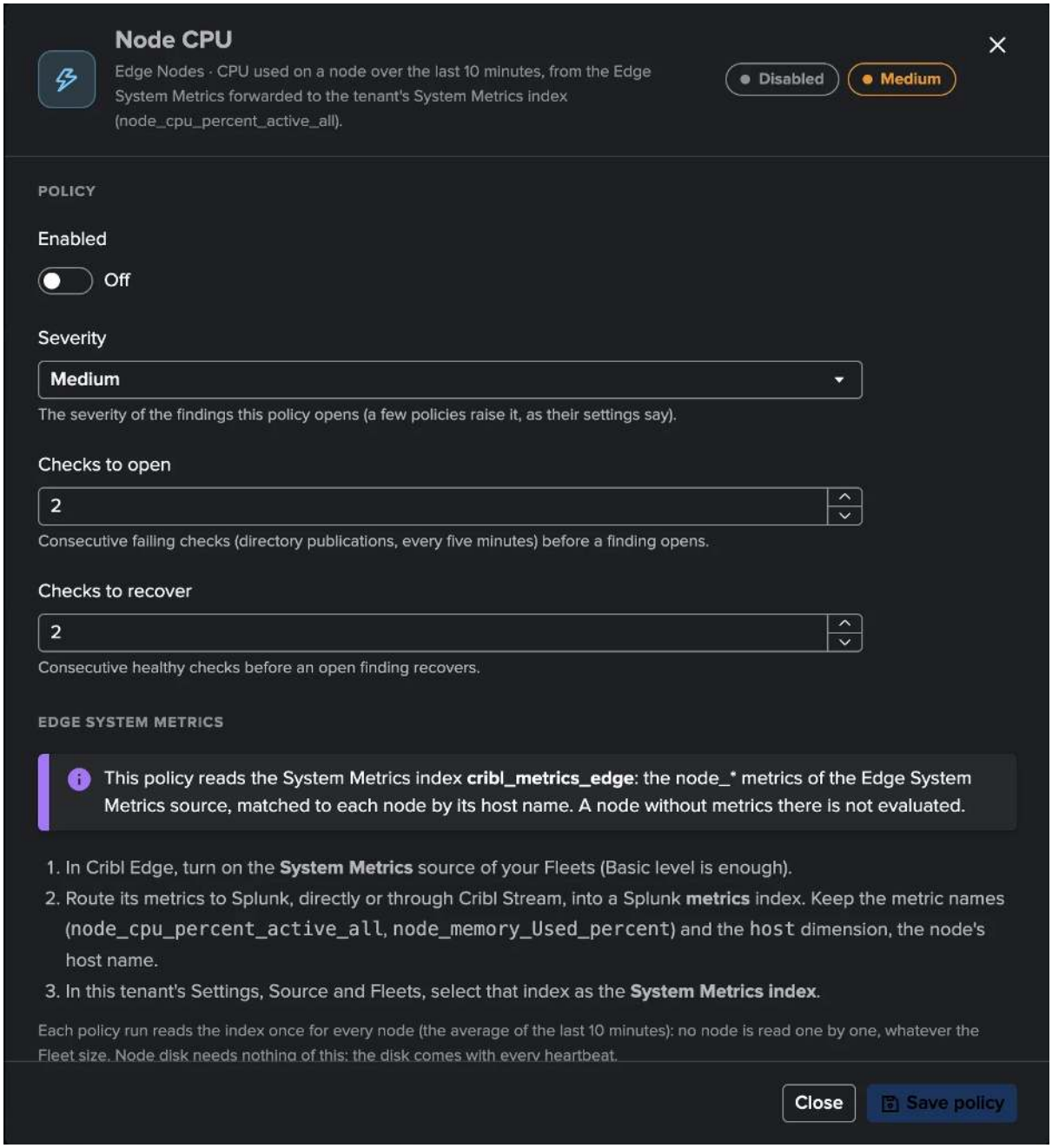
Policies

The catalogue of the eleven built-in policies, in three categories: **Cribl Leader**, **Fleets** and **Edge Nodes**. Show them as cards, compact cards or a list (the choice is shared with the Cribl Stream workspace). Each policy shows its severity, its conditions and its scope; a TrackMe administrator of the tenant turns it on or off and opens its editor:

- **Policy**: on / off, severity, the checks to open (consecutive failing directory publications, every 5 minutes) and the checks to recover;
- **Conditions**: the policy's own thresholds and graces, each with its help;
- **Scope**: the Fleets it applies to (empty: every Fleet of the tenant);
- **Bank holidays**: whether a bank holiday of the tenant's calendars holds the policy's new findings (**Hold new findings** or **Keep opening findings**; see *Bank holidays*).

Two pills tell what a policy is waiting for: **Needs nodes** on Node connectivity until you select the nodes that must stay connected (see *Nodes that must stay connected*), and **Needs System Metrics** on Node CPU and Node memory until the tenant has a System Metrics index (their editor shows the steps). Eight policies are on out of the box; **Node disk**, **Node CPU** and **Node memory** are off: node resource monitoring is a choice, because a fleet of endpoints would otherwise open a finding per full disk on day one. The full catalogue with every default is in *Built-in policies*.

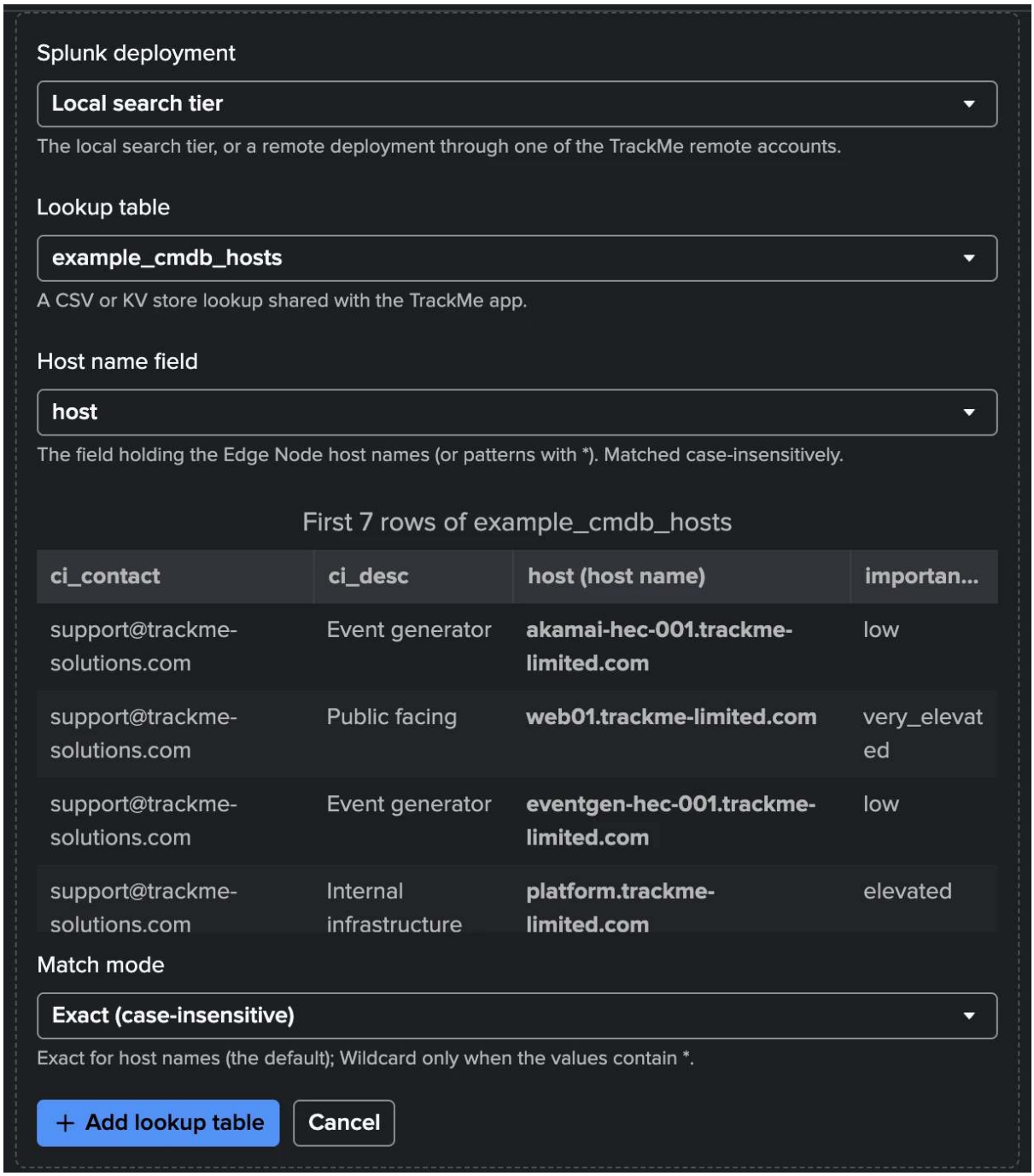



Node CPU
Edge Nodes · CPU used on a node over the last 10 minutes, from the Edge System Metrics forwarded to the tenant's System Metrics index (node_cpu_percent_active_all).
Disabled
Medium
POLICY
Enabled
Off
Severity
Medium
The severity of the findings this policy opens (a few policies raise it, as their settings say).
Checks to open
2
Consecutive failing checks (directory publications, every five minutes) before a finding opens.
Checks to recover
2
Consecutive healthy checks before an open finding recovers.
EDGE SYSTEM METRICS
This policy reads the System Metrics index cribl_metrics_edge: the node_* metrics of the Edge System Metrics source, matched to each node by its host name. A node without metrics there is not evaluated.
1. In Cribl Edge, turn on the System Metrics source of your Fleets (Basic level is enough).
2. Route its metrics to Splunk, directly or through Cribl Stream, into a Splunk metrics index. Keep the metric names (node_cpu_percent_active_all, node_memory_Used_percent) and the host dimension, the node's host name.
3. In this tenant's Settings, Source and Fleets, select that index as the System Metrics index.
Each policy run reads the index once for every node (the average of the last 10 minutes): no node is read one by one, whatever the Fleet size. Node disk needs nothing of this: the disk comes with every heartbeat.
Close
Save policy

Nodes that must stay connected

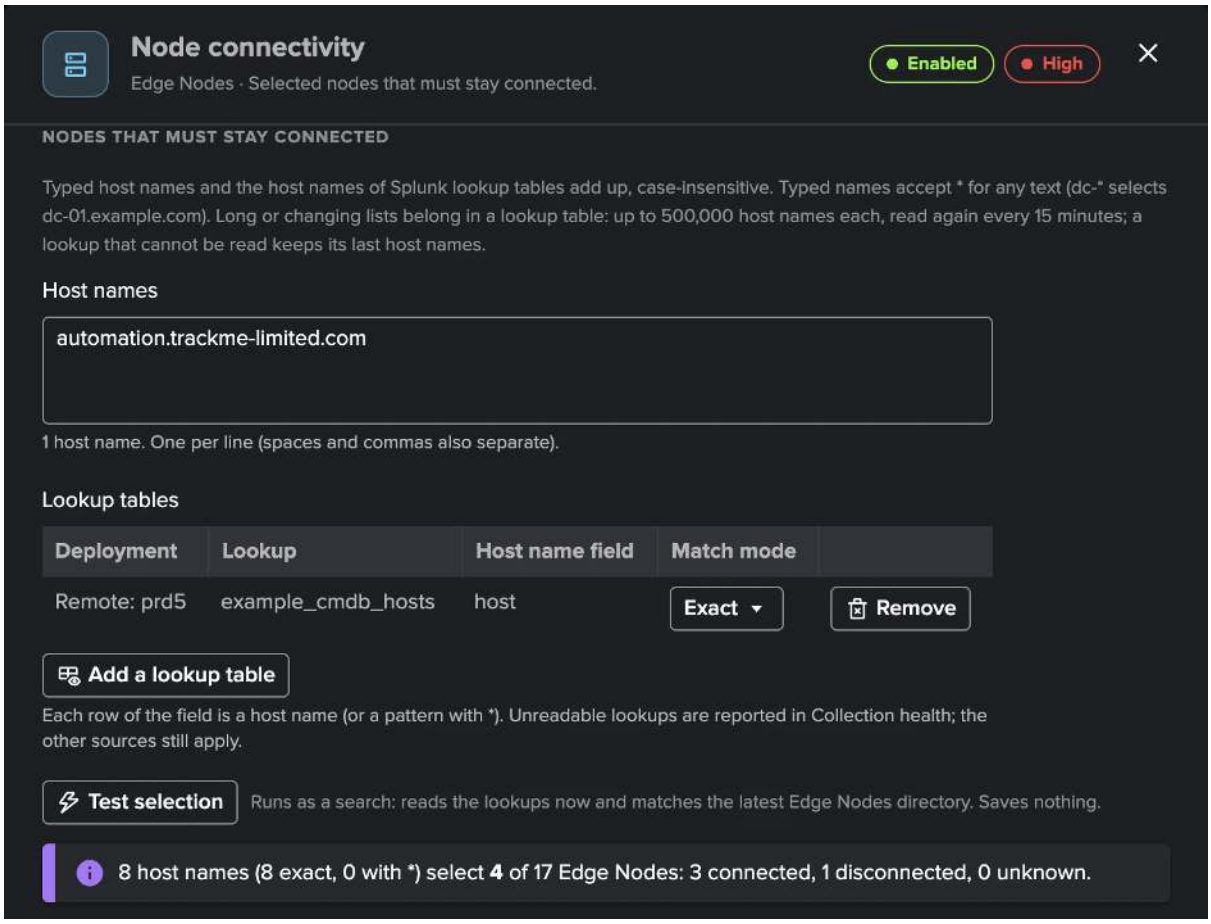
Fleet connectivity judges shares of a Fleet, so one node going down opens nothing on its own. For the nodes that must **never** drop (servers, collectors), **Node connectivity** opens a finding about the node itself. Its editor selects them from two sources that add up, case-insensitive:

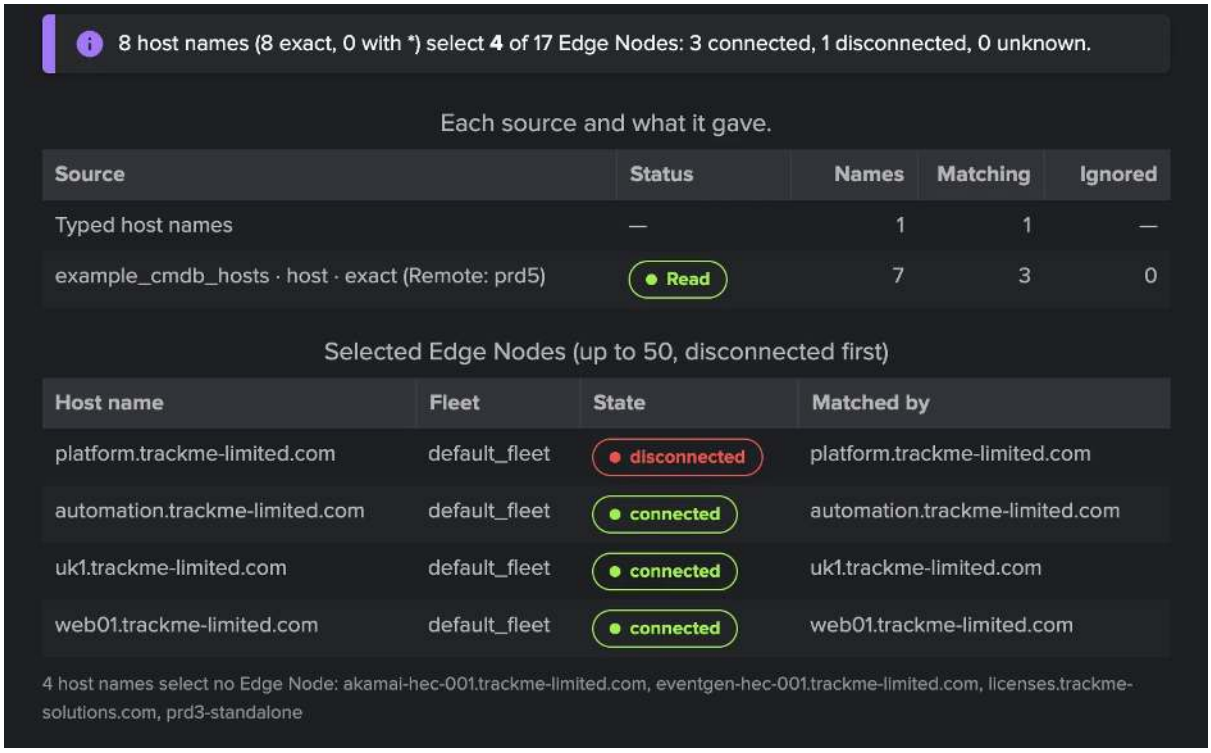
- **Host names** typed in the policy, one per line; `*` stands for any text (`dc-*`). There is no count limit.
- **Lookup tables** from Splunk, for long or changing lists such as a CMDB extract: **Add a lookup table** picks the deployment (the local search tier or a remote deployment through a TrackMe remote account), the lookup and the field holding the host names, with a preview of its first 10 rows and the field suggested. Each lookup has a **Match mode**: **Exact** (whole host names, the default) or **Wildcard** (`*` matches any text). A lookup holds up to 500,000 host names and is read again every 15 minutes; one that cannot be read keeps its last host names, and Collection health says so (`node_lookups_failing`).



Test selection checks the result before you save: it runs as a search, reads the lookups now and matches the latest

Edge Nodes directory. It shows how many nodes are selected and in which state, what each source gave, the selected nodes with what matched them, and the names that select no node. It saves nothing.





The details and limits are in *Built-in policies*, section Node connectivity selection.

Bank holidays

A Cribl Edge tenant uses the *bank holiday calendars* like every tenant: the system default calendars, unless it selects its own or none (*Manage Bank Holidays → Tenants & settings*, or *Configure tenant → Bank Holidays*). One of the four holiday effects applies to it, **Suppress new alerts**, on by default: while a holiday of the tenant's calendars is active, the policies set to hold open no new finding. Edge has no traffic model, so there is no ML training effect; SLA and monitoring windows have no effect on a Cribl Edge tenant either.

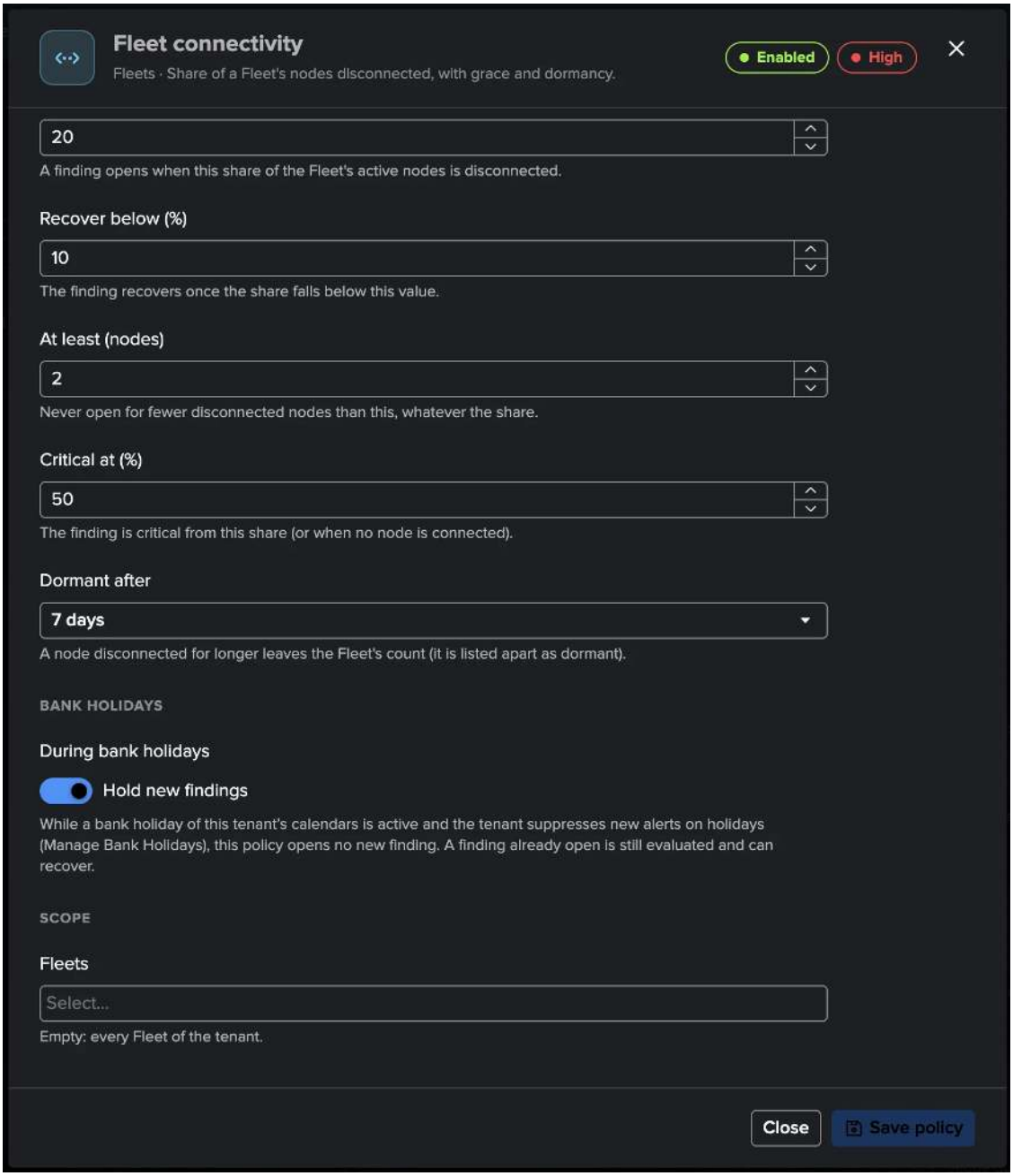
Each policy decides in its editor, section **Bank holidays**, whether a holiday holds its new findings (**Hold new findings**) or not (**Keep opening findings**). The defaults:

Policy	During bank holidays	Why
Fleet connectivity, **Fleet coverage**	Hold new findings	Laptops and desktops switched off for the holiday would otherwise open findings.
Node connectivity	Keep opening findings	It watches the nodes you selected as always connected, such as servers.
Every other Edge policy (Version drift, Configuration drift, Node identity, Source and Destination health, Node disk, Node CPU, Node memory)	Keep opening findings	An infrastructure failure on a holiday still opens its finding.
Leader availability	No setting	The Leader is the control plane: it is always judged.

While a holiday holds a policy:

- a failure of that policy opens no new finding, and sends no notification. A finding that was resolved or dismissed is not reopened by a held failure;
- a finding **already open** is still evaluated: it can recover and close during the holiday;
- after the holiday, a failure that continues counts its checks to open again before it opens its finding;
- the Policies tab says **held on bank holidays** in the conditions of each policy set to hold, and in Collection health the runs of the **Edge Nodes** and **Monitoring policies** jobs say so: *Bank holiday (names): the policies set to hold on bank holidays open no new finding while it lasts*, with how many failing checks were held in that run;
- when the calendars cannot be read, nothing is held: findings open as on any other day.

The hold is decided at the start of each 5-minute check, so a check that straddles the start of a holiday is judged normally. A dismissal for a time that ends during a holiday reopens its finding as on any other day.



Custom policies

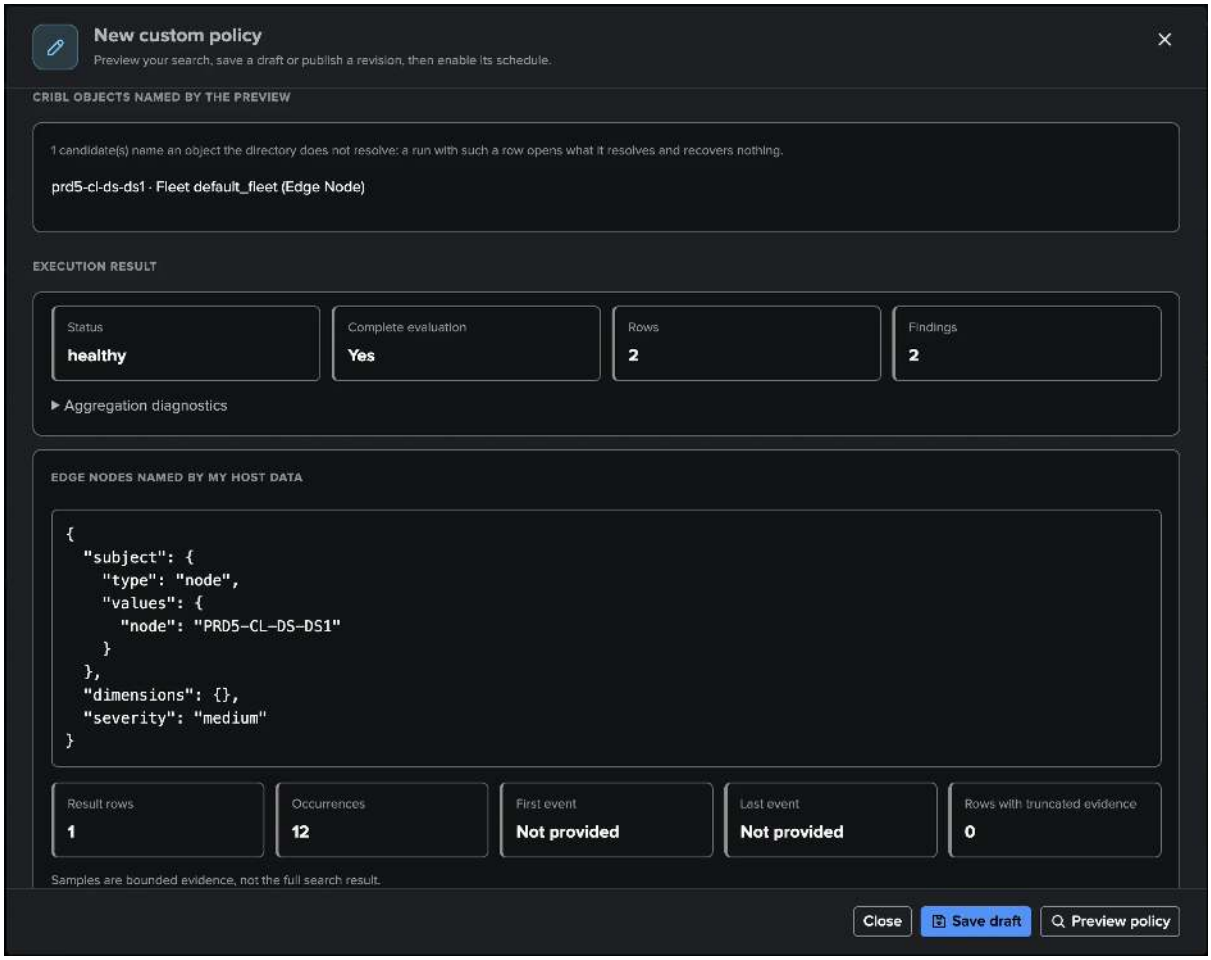
When no built-in policy covers a rule of your own, write it as an SPL search on the tenant's Splunk data: **Policies → Custom policies → Create policy**. Only the tenant's TrackMe administrators see and edit custom policies, because they hold SPL.

On a Cribl Edge tenant the object is the **Edge Node**. Each result row names a node in one result field, by its node id or its host name, matched without regard to case in the current node directory. Each node named opens a finding of the policy. That finding joins the node's Finding group and is triaged, notified, offered findings actions and reviewed by the AI Agent like a built-in one.

For example, a search over your own host or OS data that ends with `| stats count by host | rename host as node` opens a finding on every Edge Node it names, with `node` as the Node field and `count` as the occurrence count.

The editor, the preview, the version history and the lifecycle are the same as on a Cribl Stream tenant (see *Custom policies*), with one object type: **Edge Node**. Key behaviour:

- **Preview policy** shows each candidate with the node it resolves to, and reports a name that is not in the node directory as unresolved.
- A run with an unresolved row recovers nothing: that row may be the node still failing.
- **Disable**, **Archive** and **Delete permanently** close the policy's open findings.

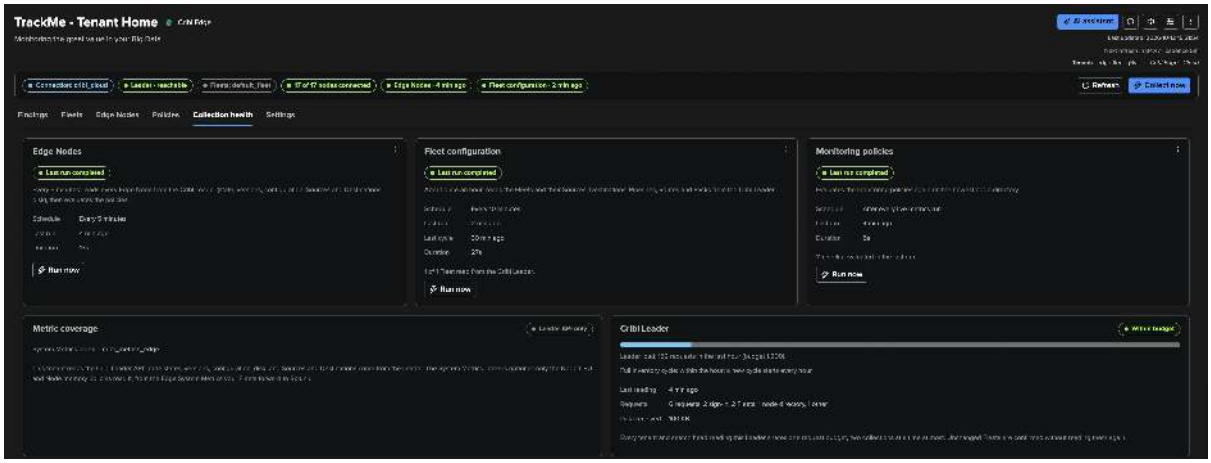


The execution health of each policy is in *Collection health → Custom policies*. A tenant holds up to 100 custom policies, 25 of them enabled; see *Custom policies*.

Collection health

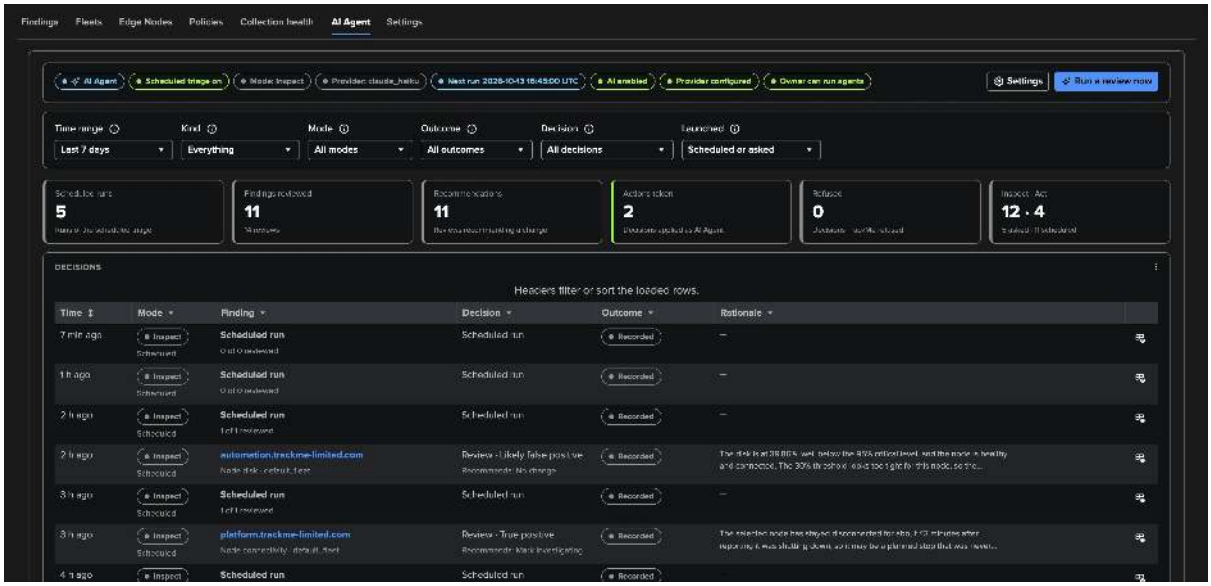
One card per job — **Edge Nodes** (every 5 minutes: reads every node from the Leader, then evaluates the policies), **Fleet configuration** (about once an hour: reads the Fleets and their Sources, Destinations, Pipelines, Routes and Packs) and **Monitoring policies** (on demand: evaluates the policies again on the newest directory) — with its last run, duration, next run and **Run now**. The tab also shows the Cribl Leader's request load against its budget, and the System Metrics index the tenant reads.

When collection does not run as configured, the tab lists the problems in plain words, and the Configuration Guardian raises **Cribl Edge collection needs attention** (`cribl_collection_health`). The Edge-specific conditions: half or more of the connected nodes report **no metrics** to the Leader, and **no Edge System Metrics** arrive in the System Metrics index or it **cannot be read**. Every condition is in *Cribl Edge reference*.



AI Agent

The **AI Agent** tab shows the Cribl Edge AI Agent: whether it can run and why not, its scheduled triage, its totals, **Run a review now** and its decision log, each decision with the facts the agent read and its rationale. See *The Cribl Edge AI Agent*.



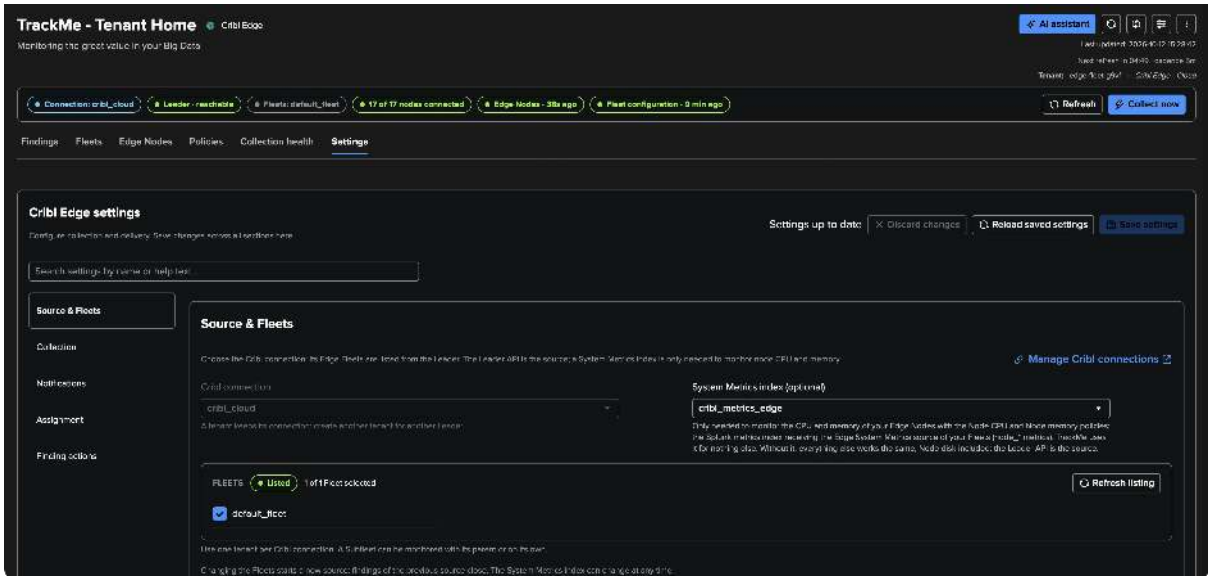
Settings

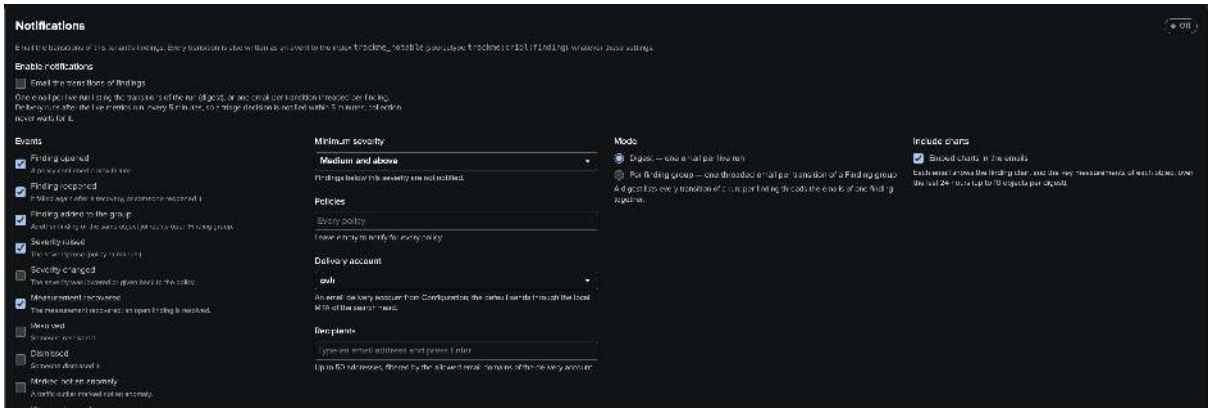
One Save covers every section:

- **Source & Fleets**: the connection (locked), the Fleets and the System Metrics index. **Changing the Fleets starts a new source**: the findings of the previous source close. The System Metrics index can change at any time without that. **Repair workspace** checks the tenant's collections and reports and keeps findings and settings.
- **Collection**: **Scheduled collection** on or off.
- **Notifications**: finding emails (events, minimum severity, policies, delivery account, recipients, digest or one email per Finding group, charts, test email).
- **AI Agent**: the scheduled triage of the Cribl Edge AI Agent, its mode, provider, limits, allowed decisions and custom instructions (see *The Cribl Edge AI Agent*).
- **Assignment**: assignment groups, their team email, ordered routing rules (by policy, severity, Fleet or name) and manual overrides.

- **Finding actions**: drilldown searches, links and webhooks offered on findings, and **Automations**: the rules that call an action automatically.

A TrackMe administrator of the tenant manages these settings; other roles read them.





Findings in practice

A Cribl Edge finding is about one object, one policy and one measurement:

The finding is about	Raised by
A Fleet	Fleet connectivity, Version drift, Configuration drift, Fleet coverage, Node identity. One finding per Fleet and policy, whatever its number of nodes; the evidence names example nodes.
A Source or a Destination of a Fleet	Source and Destination health: one finding per object, with how many nodes report it in warning or trouble.
An Edge Node	Node connectivity, Node disk, Node CPU, Node memory.
The Leader	Leader availability.

Findings work exactly as on Cribl Stream. Each object gathers its findings in one automatic **Finding group**, the unit you triage and get notified about. A finding **recovers by itself** when the measured condition clears, and closes by itself when its object is no longer listed or no longer in the tenant's scope, or when nothing evaluated it for 24 hours. Triage, notes, snooze, dismissal, the severity override and assignment are the same as for every findings-based tenant, and the Cribl Edge AI Agent can review them (see *The Cribl Edge AI Agent*). Edge has no *Stop monitoring this object* and no *Not an anomaly* (there is no model to teach): to stop judging an object, narrow the policy's scope or turn the policy off.

The shared concepts are described once:

- *Findings, Finding groups and findings actions* — findings, statuses, triage, severity, notifications;
- *Finding groups* — Finding groups, automatic and manual;
- *Findings actions* — drilldowns, links, webhooks and automation rules;
- *Webhook accounts* — the accounts a webhook action posts with;
- *Assignment and automatic routing* — assignment groups, routing rules and team emails.

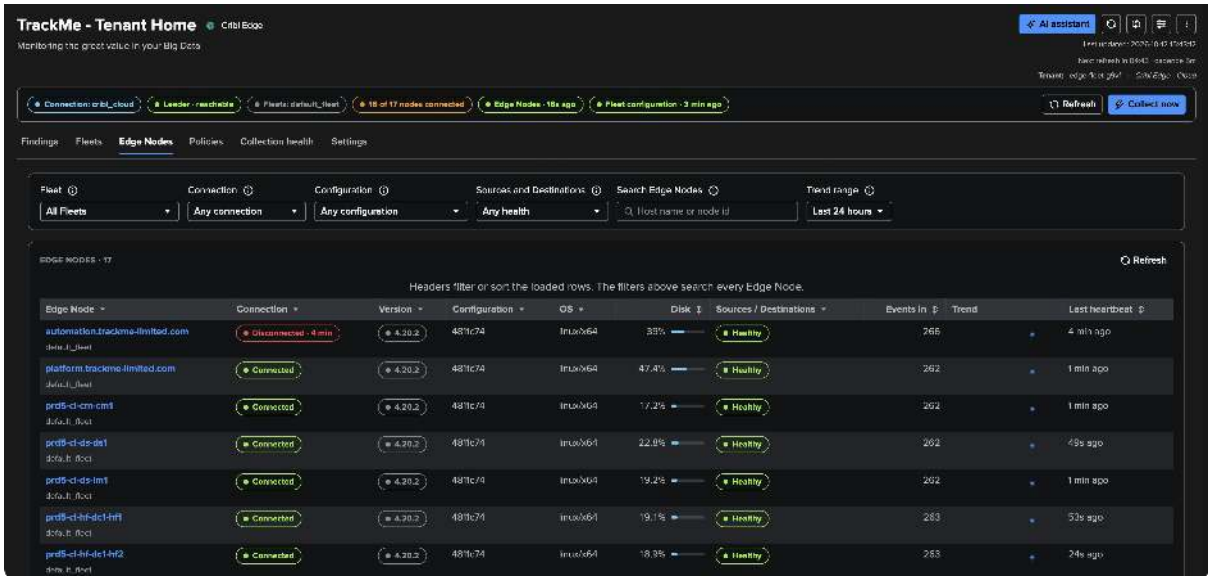
Links into the workspace. An email, an action or a dashboard can open the workspace on an object: `TenantHome?tenant_id=<tenant>&tab=nodes&node=<node key>` opens an Edge Node, `tab=fleets&fleet=<Fleet id>` opens a Fleet, and the Edge Nodes filters can be passed as parameters (`group_id`, `state`, `config_state`, `io`, `q`).

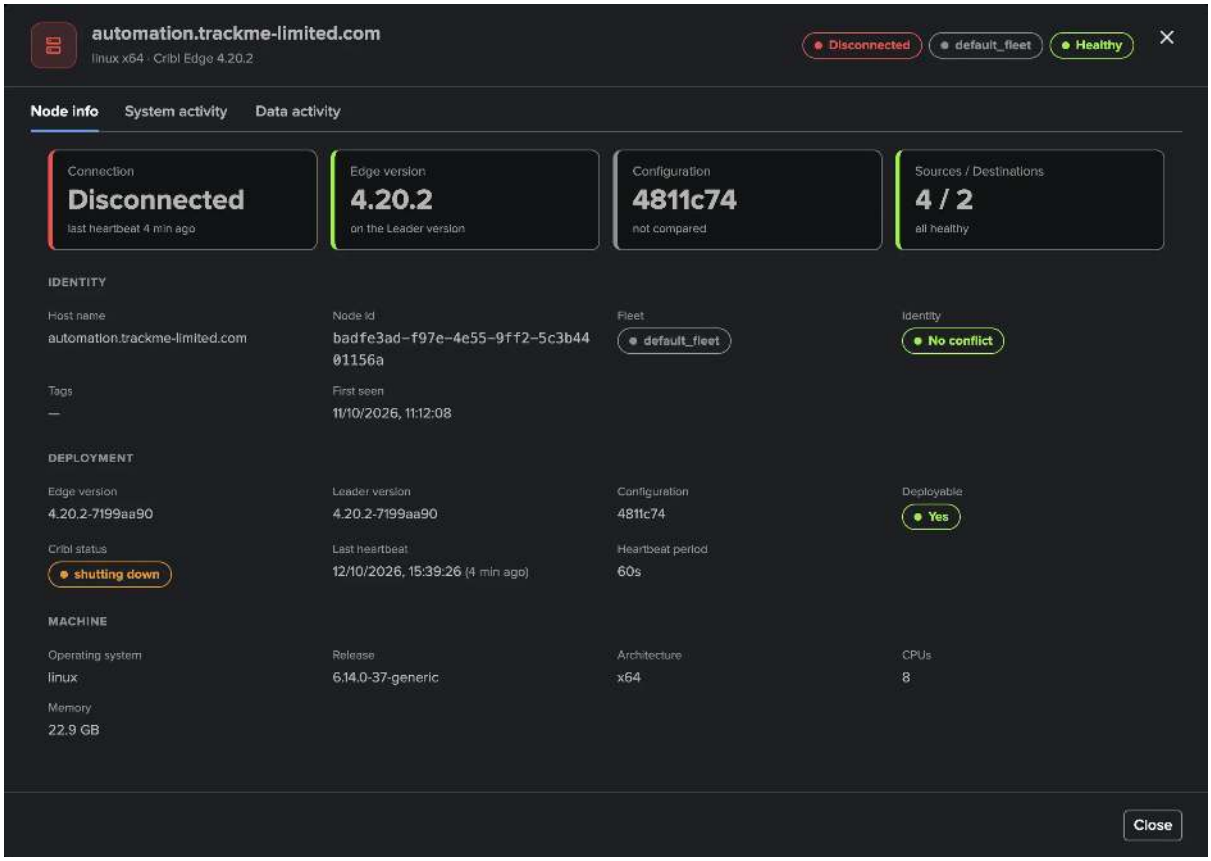
A disconnected node: what you see

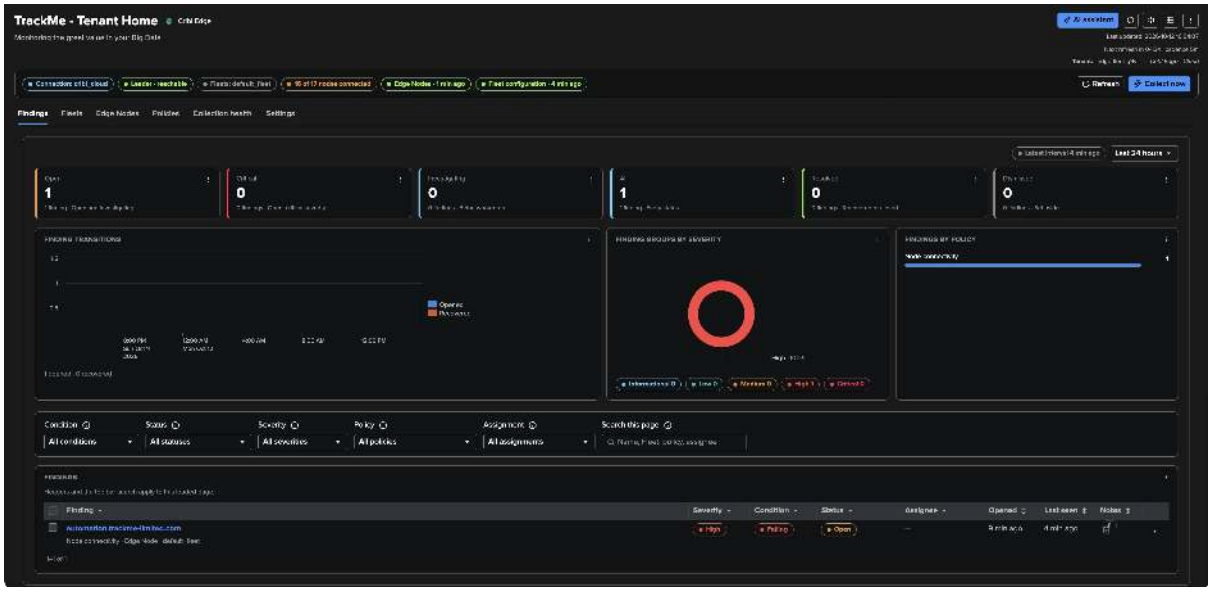
Take a Fleet of 17 nodes, one of which stops (a laptop closed, a server shut down, a service stopped):

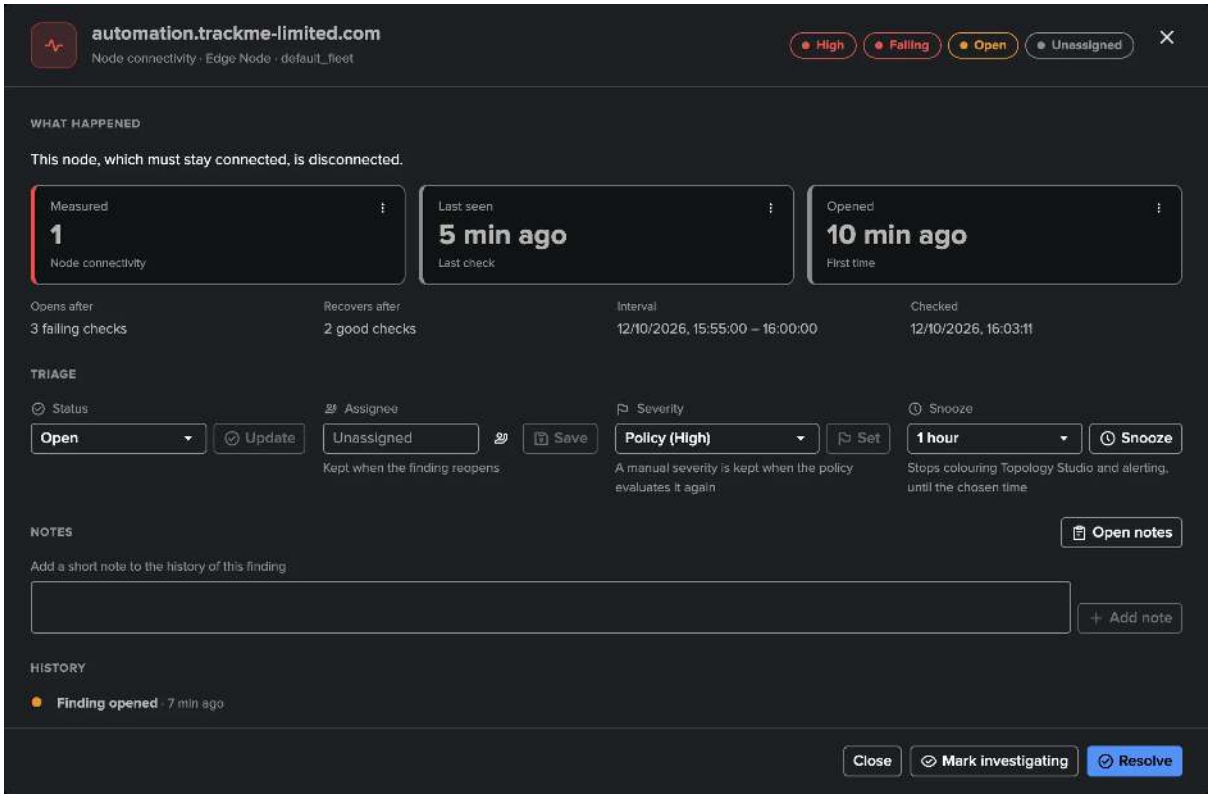
1. Within 10 minutes of its last heartbeat, the node is **disconnected** in the next node directory. The status strip reads **16 of 17 nodes connected**, the Fleets tile **Edge Nodes connected** turns amber with "1 disconnected", and the Fleet row shows the node in **Disconnected**.
2. In **Edge Nodes**, the filter **Connection: Disconnected** lists it with how long it has been disconnected; its Node info shows the last heartbeat, and System activity says the node cannot be read.
3. **No finding opens for that one node by default**, by design: **Fleet connectivity** opens when at least 20 % of the Fleet's active nodes, and at least 2 nodes, are disconnected (critical from 50 %, or when no node is connected). One node in 17 is below that.
4. For the nodes that must **never** drop (servers, collectors), select them in **Node connectivity**: typed host names (patterns like `dc-*` are accepted) or a Splunk lookup table such as a CMDB extract. A selected node without a heartbeat for 15 minutes opens a **High** finding about that node after three checks, and it recovers when the node reconnects.
5. A node disconnected for more than 7 days (setting **Dormant after**) leaves the Fleet's count, so retired nodes that nobody removed from the Leader do not hold a Fleet in alert forever.

In Topology Studio, a disconnected node without a finding is drawn **quiet grey**: it is disclosed, and it does not count in a group's healthy share.




automation.trackme-limited.com
linux x64 · Cribl Edge 4.20.2
Disconnected
default_fleet
Healthy
Node info
System activity
Data activity
Connection
Disconnected
last heartbeat 4 min ago
Edge version
4.20.2
on the Leader version
Configuration
4811c74
not compared
Sources / Destinations
4 / 2
all healthy
IDENTITY
Host name
automation.trackme-limited.com
Node id
badfe3ad-f97e-4e55-9ff2-5c3b4401156a
Fleet
default_fleet
Identity
No conflict
Tags
—
First seen
11/10/2026, 11:12:08
DEPLOYMENT
Edge version
4.20.2-7199aa90
Leader version
4.20.2-7199aa90
Configuration
4811c74
Deployable
Yes
Cribl status
shutting down
Last heartbeat
12/10/2026, 15:39:26 (4 min ago)
Heartbeat period
60s
MACHINE
Operating system
linux
Release
6.14.0-37-generic
Architecture
x64
CPUs
8
Memory
22.9 GB
Close


TrackMe - Tenant Home
Findings
Fleets
Edge Nodes
Policies
Collection health
Settings
Last 24 hours
Refresh
Collect now
Open
1
Critical
0
High
0
Medium
1
Low
0
Info
0
FINDING TRANSITIONS
FINDINGS BY SEVERITY
FINDINGS BY POLICY
Condition
All conditions
Status
All statuses
Severity
All severities
Policy
All policies
Assignment
All assignments
Search this page
Finding
Severity
Condition
Status
Assignee
Opened
Last seen
Notes
automation.trackme-limited.com
High
Failing
Open



Version drift

Cribl Edge upgrades roll out node by node, and a node left behind is a common, silent problem. **Version drift** compares the version of every connected node with a baseline, **the Leader version** by default (Cribl's own upgrade rule) or **the Fleet's most common version**:

- the Fleets tab shows the versions running in each Fleet and how many nodes are **Not on the Leader version**; the Fleet's **Nodes to look at → Versions** names the nodes behind each version, and each node's Node info compares its version with the Leader's;
- once nodes stay off the baseline for the **rollout grace** (24 hours by default, the time of a rolling upgrade), a **Version drift** finding opens on the Fleet, Medium by default. It is raised to **High** when a node runs a version **newer than the Leader** or **two minor versions or more behind** it: the cases Cribl does not support;
- the finding recovers once every connected node is on the baseline. A node that just enrolled and reports no version yet is never counted as drift.

To fix it, upgrade the nodes from the Fleet in Cribl, or change the baseline when your Fleets deliberately run their own version.



Configuration drift works the same way for deployments: connected nodes not running the Fleet's deployed configuration after a 30-minute grace open a High finding on the Fleet, critical when half the Fleet or more is behind.

Notifications

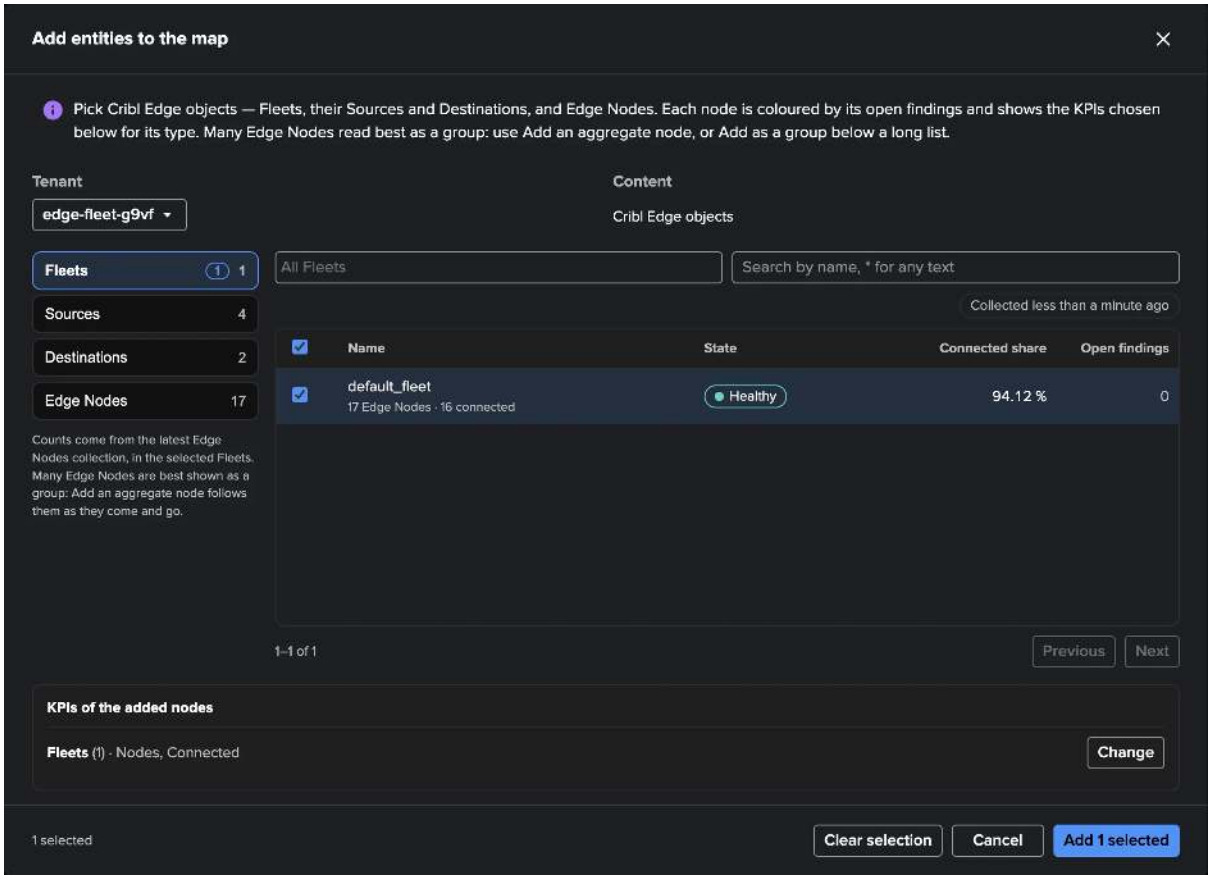
Finding notifications work as for Cribl Stream (*Findings, Finding groups and findings actions*): turn them on in **Settings → Notifications**, pick the events (opened, reopened, added to the group, escalated, recovered, dismissal ended and closed by default), a minimum severity, the policies, a delivery account and recipients, and choose a **digest** or **one email per Finding group**, with charts. Emails are sent after each live run and carry the "Cribl Edge" label and a link that opens the finding in the workspace. **Send test email** checks the delivery.

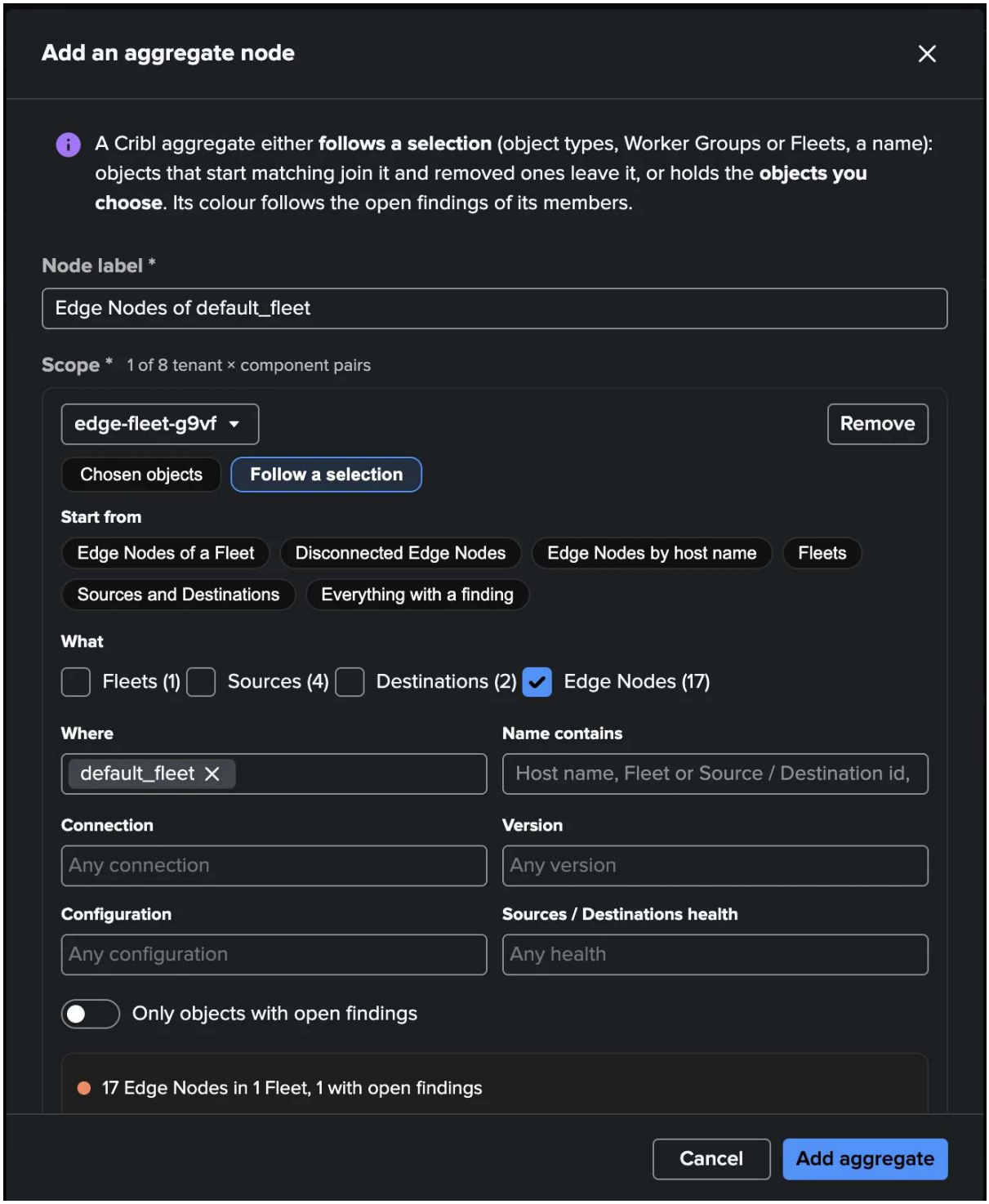
Every transition is also written as an indexed event (`trackme:cribl:finding` and `trackme:cribl:finding_group`, `object_category=cribl_edge`), whatever the notification settings, for your own Splunk alerts, Enterprise Security or SOAR: see *Cribl Edge reference*.

Topology Studio

Cribl Edge objects are Topology Studio objects (*Topology Studio*): **Fleets**, **Sources**, **Destinations** and **Edge Nodes** of an Edge tenant.

- **Add entities**: pick Fleets, Sources and Destinations of a Fleet, or Edge Nodes (with the Edge Nodes filters), each type with its count and state.
- **Groups** (*Add an aggregate node*, Cribl Edge form): the recommended way to show nodes. Start from **Edge Nodes of a Fleet**, **Disconnected Edge Nodes**, **Edge Nodes by host name**, **Fleets**, **Sources and Destinations** of a Fleet or **Everything with a finding**, then narrow by Fleet, host name pattern, connection, version, configuration or Sources / Destinations health. A group follows its nodes as they come and go, at any fleet size, and shows its health and healthy share (groups carry no summed traffic or averaged CPU).
- **Status**: findings first (red or orange; blue when only snoozed). Otherwise green when present and current. A **disconnected node without a finding is quiet grey**, and a Source or Destination without traffic is quiet.
- **KPIs**: for a node, connected, disk, events and bytes in and out, dropped events, unhealthy Sources / Destinations; for a Fleet, nodes, connected and connected %, drift counts, traffic, highest disk; for a Source or Destination, nodes, warning, trouble, events, bytes. Fleets and nodes have 24-hour sparklines. **Node CPU and memory** are read on demand from the System Metrics index when a view shows them.
- **Topology Alerts** (*Topology Alerts*) work on Edge objects and groups.
- **Create with AI** (*Build with AI — a topology from a description*) offers Edge tenants ("· Cribl Edge") with three starting points: **Fleet overview**, **Edge problems wall** and **Sources and Destinations of a Fleet**. Groups with node filters are built by hand.
- Opening a node or a Fleet from a view opens the Edge workspace on it.


Add entities to the map
Pick Cribl Edge objects — Fleets, their Sources and Destinations, and Edge Nodes. Each node is coloured by its open findings and shows the KPIs chosen below for its type. Many Edge Nodes read best as a group: use Add an aggregate node, or Add as a group below a long list.
Tenant
edge-fleet-g9vf
Content
Cribl Edge objects
Fleets 1
Sources 4
Destinations 2
Edge Nodes 17
Counts come from the latest Edge Nodes collection, in the selected Fleets. Many Edge Nodes are best shown as a group: Add an aggregate node follows them as they come and go.
All Fleets
Search by name, * for any text
Collected less than a minute ago
Name
State
Connected share
Open findings
default_fleet
17 Edge Nodes · 16 connected
Healthy
94.12 %
0
1–1 of 1
Previous
Next
KPIs of the added nodes
Fleets (1) - Nodes, Connected
Change
1 selected
Clear selection
Cancel
Add 1 selected


Add an aggregate node
A Cribl aggregate either follows a selection (object types, Worker Groups or Fleets, a name): objects that start matching join it and removed ones leave it, or holds the objects you choose. Its colour follows the open findings of its members.
Node label *
Edge Nodes of default_fleet
Scope * 1 of 8 tenant × component pairs
edge-fleet-g9vf
Remove
Chosen objects
Follow a selection
Start from
Edge Nodes of a Fleet
Disconnected Edge Nodes
Edge Nodes by host name
Fleets
Sources and Destinations
Everything with a finding
What
Fleets (1)
Sources (4)
Destinations (2)
Edge Nodes (17)
Where
default_fleet
Name contains
Host name, Fleet or Source / Destination id,
Connection
Any connection
Version
Any version
Configuration
Any configuration
Sources / Destinations health
Any health
Only objects with open findings
17 Edge Nodes in 1 Fleet, 1 with open findings
Cancel
Add aggregate



The Cribl Edge AI Agent

The **Cribl Edge AI Agent** reviews Edge findings the way an operator would, from facts TrackMe already holds rather than from guesses. It is its own agent, separate from the Cribl Stream AI Agent, built on the same framework as the other findings AI Agents (*AI Agents for findings*), with Edge's own evidence and decisions. It assesses each finding — **true positive**, **likely false positive**, **expected behaviour**, **node or Fleet not in use** or **needs investigation** — recommends a decision, and keeps one current **[AI Agent]** note on the finding.

- **Inspect** only recommends: nothing changes but the note.
- **Act** applies the recommendation, but only what the tenant allows (see *What it may decide*).

What it reads

The agent never runs a search and never calls Cribl, and no machine learning is involved: its facts are what TrackMe already holds for the tenant.

- **The finding**: its evidence and Edge detail (distributions, example nodes), its history and episodes, its Finding group, and the decisions already taken on it by an operator or by the agent.
- **Its Fleet** in the published node directory: nodes, connected and disconnected (and for how long), version and configuration drift, the rollout, and the Sources and Destinations reported unhealthy.
- **Its Edge Node**, for a node finding: state, Edge version and its risk against the Leader, disk, and the health of its Sources and Destinations.
- **Whether the node or the Fleet is still in the node directory.** Only a current publication of the directory can prove that a node or a Fleet is gone, and only once the finding has gone an hour without being evaluated: one publication missing a node (a Leader restart, a transient listing) proves nothing. A Fleet cleared in *Settings → Source & Fleets* counts as gone in the same way.
- **The tenant's Collection health**, so that a stale directory or a failing job is not read as a healthy fleet.

It knows the usual Edge patterns: version drift during a rollout, or configuration drift right after a deploy, is often expected; a node newer than the Leader or two minor versions behind it is a real risk; a retired node or Fleet is not in use; a stale directory needs investigation.

What it may decide

In act mode, each decision has its own setting in *Settings → AI Agent*, and TrackMe checks its conditions again before applying it, whatever the agent wrote:

Decision	Default	Condition checked by TrackMe
Mark investigating	Allowed	The finding is open.
Dismiss for a time	Allowed, 7 days at most (up to 90)	A **critical** or **high** finding that is still failing only when its node or Fleet is gone from the node directory.
Resolve	Not allowed	Only after a measured recovery, and never while another finding of its Finding group still fails.

A Finding group of several findings is decided as a whole. Node findings group by host name within a Fleet, so a reinstalled or cloned node (a new node id, the same host name) shares a group with the old one: the old node being gone never lets the agent dismiss a critical or high finding of the live node.

There is no *Not an anomaly* and no *Stop monitoring* on Edge: there is no traffic model to teach and no object monitoring to turn off. The agent **never** dismisses permanently, assigns, changes a severity or a policy, changes the nodes selected in Node connectivity or the Fleets the tenant collects. It makes at most one decision per finding and Finding group in a run.

Asking the AI Agent

Ask the AI Agent… is in the finding and Finding group menus and the investigation view, and **Run a review now** on the **AI Agent** tab reviews the open findings. The agent always inspects first: you read its assessment, the evidence and its rationale, then choose **Run its recommendations…** to let it act on that inspection, within 30 minutes. The tenant's power and admin roles can ask it and let it act; its settings belong to a TrackMe administrator of the tenant.

The AI Assistant of the Edge Tenant Home can also propose a Cribl Edge AI Agent review, as a consent card you confirm (see *AI Assistant*).

Asking needs the AI Assistant to be enabled and at least one *AI provider* configured.

Scheduled triage

The agent can review the open and investigating findings of the tenant **every hour** on its own. Scheduled triage is **off by default**; a TrackMe administrator of the tenant turns it on in *Settings → AI Agent*. Each run:

- reviews up to 10 findings (1 to 50), one per Finding group, the most severe first, within a time budget of 3000 seconds that fits the hourly cadence;
- skips a finding reviewed in the last day (the review interval), unless a new episode started or its Finding group changed;
- runs in **inspect** mode unless you choose **act**;
- **never overrides a person**: a finding or Finding group that an operator decided on, or that has an assignee, is left to the operators;
- applies the shared filters of the automated AI reviews, with the finding severity as the priority: by default only critical and high findings are selected (see *AI Agents automation*).

Scheduled triage runs only when the AI Assistant is enabled, an AI provider is configured, the tenant is enabled, and the tenant owner is a Splunk administrator account (not `nobody`). The Configuration Guardian warns about a tenant with scheduled triage on and a `nobody` owner (`ai_automation_owner_is_system_user`).

Agent settings

The agent's settings are in the workspace, **Settings → AI Agent**: scheduled triage on or off, mode, AI provider, review interval, findings per run, time budget per run, the allowed decisions with the longest dismissal, and the custom instructions. Every setting and its default is in *Cribl Edge reference*.

The **Manage AI Agents automation** page (*Artificial Intelligence* menu, *AI Agents automation*) lists the Cribl Edge tenant with its own agent: a row with whether it is on, its mode and provider, an **Enable / Disable** toggle, and an edit tab with the core settings (scheduled triage, custom instructions, mode, provider, review interval and runtime). The allowed decisions stay in the workspace. Saving schedules or unschedules the agent's hourly report at once.

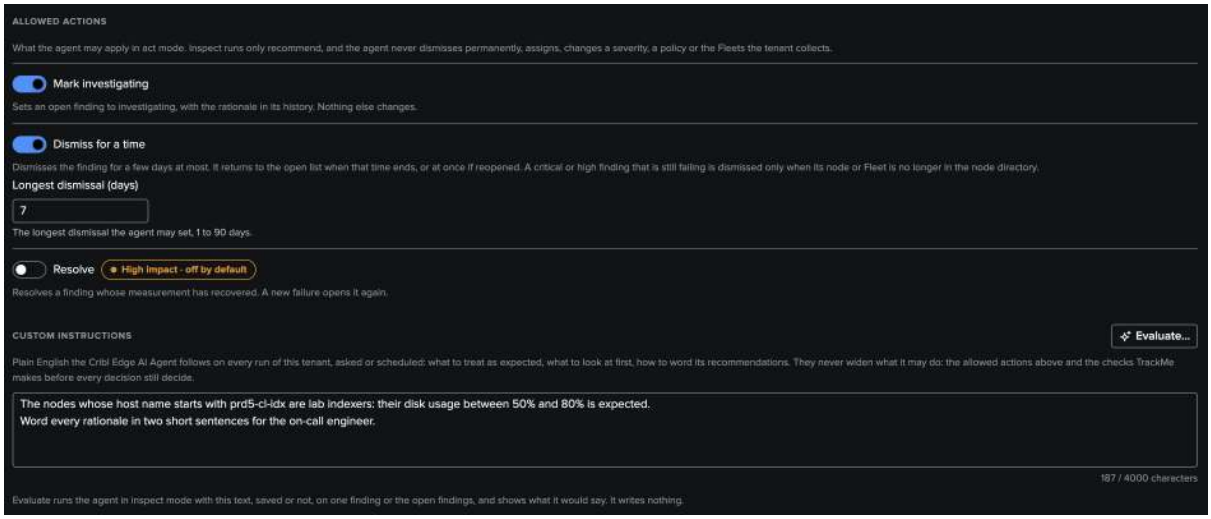
Allowing more (turning the triage on, act mode, a decision, a longer dismissal, custom instructions) needs a licence that allows changes; restricting it never does.

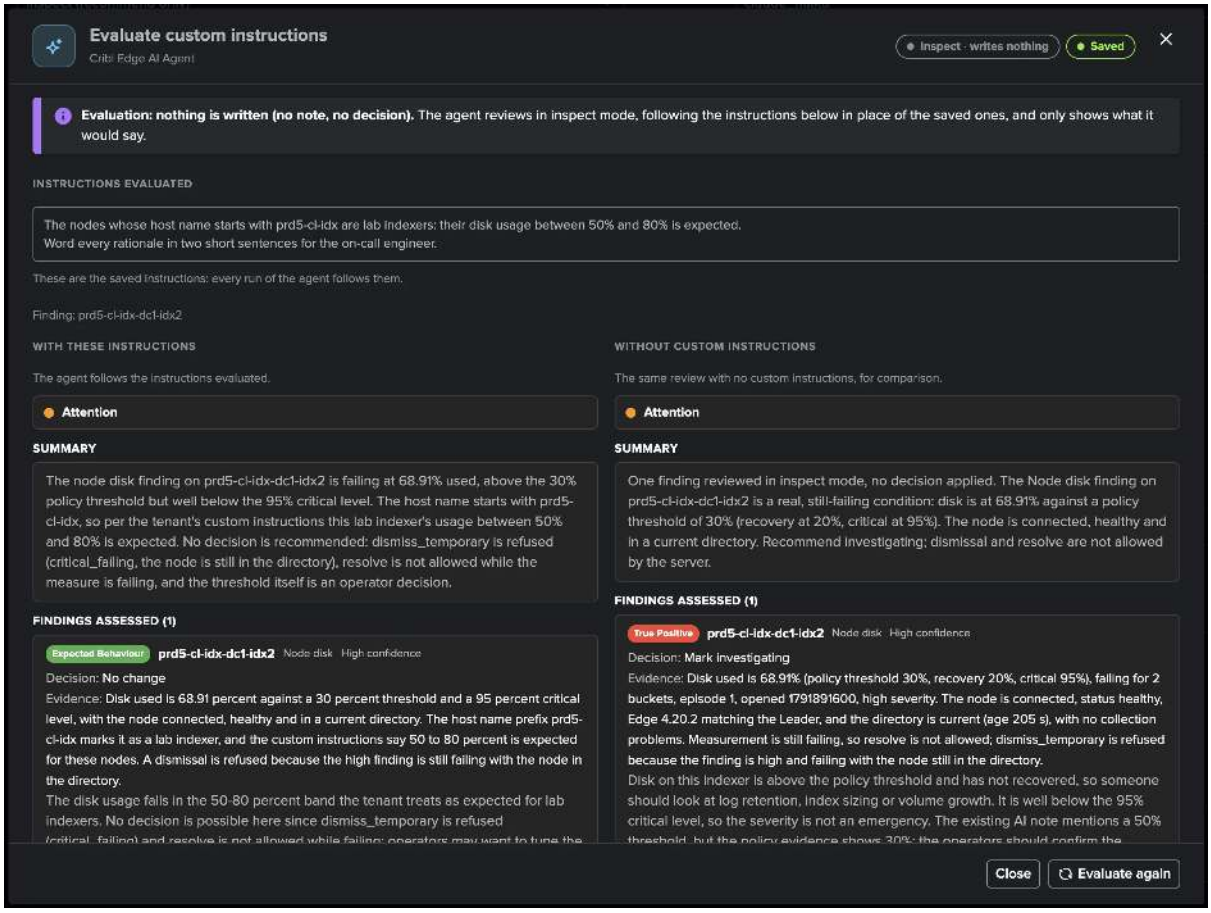
Custom instructions and Evaluate

Custom instructions are plain English that the tenant's administrators write for the agent, in *Settings → AI Agent → Custom instructions* (also in Manage AI Agents automation). The agent follows them on every run, asked or scheduled. For example: "The nodes named `lab-*` are lab machines: a disconnection outside working hours is expected."

- At most 4000 characters; line breaks are kept.
- They **never widen what the agent may do**: the allowed decisions and the conditions TrackMe checks before every decision still decide.

Evaluate…, next to the field, tries a draft before you save it: it runs the agent in **inspect** mode with the draft, on one open finding or on a review of the open findings (at most 10, the most severe first), and can run the same review **without** custom instructions side by side. An evaluation **writes nothing** — no note, no decision, no review record — and is recorded in the audit with the status `evaluation`. See *AI Agents for findings*.



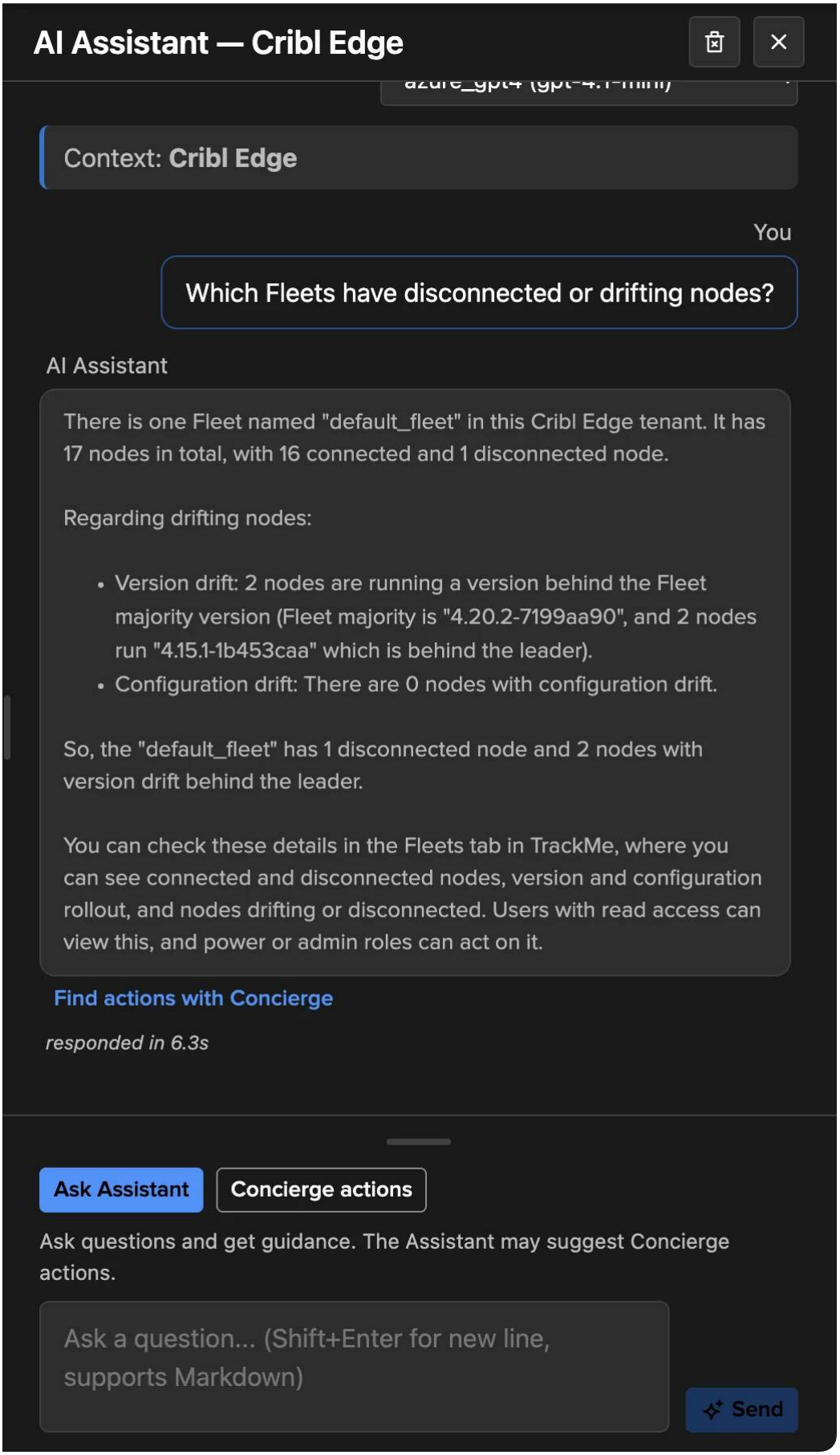


What the AI Agent did

- **The AI Agent tab**: its status, totals, and the decision log; each row opens the facts the agent read and its rationale.
- **The finding**: its history shows each decision under the actor **AI Agent**, and its notes hold the current `[AI Agent]` assessment.
- **The audit dashboards** of *Auditing AI activity* label its runs **Cribl Edge AI Agent**.
- **SPL**: every run is indexed to the tenant summary index with the sourcetype `trackme:ai_agent:cribl_edge_agent:<inspect|act>`.

AI Assistant

On an Edge tenant's Tenant Home, the AI Assistant (*AI Assistant & Concierge*) knows the workspace, the policies and their settings, the Fleets, the node summary, the Leader version and the age of the node directory, and reads the open findings: ask it why a Fleet is in drift, what a policy checks, or what to do about a finding. When a finding or the open findings would benefit from a review, it proposes a **Cribl Edge AI Agent** review as a consent card: nothing runs until you confirm. In Topology Studio, the assistant and the AI builder know Edge tenants and objects.


AI Assistant — Cribl Edge
Context: Cribl Edge
You
Which Fleets have disconnected or drifting nodes?
AI Assistant
There is one Fleet named "default_fleet" in this Cribl Edge tenant. It has 17 nodes in total, with 16 connected and 1 disconnected node.
Regarding drifting nodes:
Version drift: 2 nodes are running a version behind the Fleet majority version (Fleet majority is "4.20.2-7199aa90", and 2 nodes run "4.15.1-1b453caa" which is behind the leader).
Configuration drift: There are 0 nodes with configuration drift.
So, the "default_fleet" has 1 disconnected node and 2 nodes with version drift behind the leader.
You can check these details in the Fleets tab in TrackMe, where you can see connected and disconnected nodes, version and configuration rollout, and nodes drifting or disconnected. Users with read access can view this, and power or admin roles can act on it.
Find actions with Concierge
responded in 6.3s
Ask Assistant
Concierge actions
Ask questions and get guidance. The Assistant may suggest Concierge actions.
Ask a question... (Shift+Enter for new line, supports Markdown)
Send

What the Cribl Edge tenant does not do

- **No machine learning**: no traffic ML models, no model training, no Expected traffic.
- **No Concierge actions**: the AI Assistant answers questions and can propose a Cribl Edge AI Agent review; changes are made in the workspace or by the AI Agent within its allowed decisions.
- **No per-policy Actions or Automations tabs** in the policy editor: findings actions and automation rules are defined in **Settings → Finding actions**, for the whole tenant.
- **No Stop monitoring this object** and no **Not an anomaly** on findings.
- **No Topology Studio templates** for Edge and **no numeric rollups** on Edge groups (health and healthy share only).
- **No Cribl internal metrics** are read: the Leader API is the source, plus the optional System Metrics index for node CPU and memory.

Cribl Edge and Cribl Stream compared

	Cribl Stream tenant	Cribl Edge tenant
Follows	Worker Groups, Workers and Worker processes, Sources, Destinations, Pipelines, Routes, the Leader	Fleets and Subfleets, Edge Nodes, the Sources and Destinations of each Fleet, the Leader
Main source	Cribl internal metrics in a Splunk metric index (required), plus the Leader API	The Leader API node directory, plus an optional System Metrics index
Built-in policies	16 (14 on out of the box)	11 (8 on out of the box)
Custom policies (SPL)	Sources, Destinations, Pipelines, Routes, Worker Groups, Worker machines and processes	Edge Nodes
Judgement	Per object and measurement, traffic models	Per Fleet (shares of nodes, graces), per selected node, thresholds
Machine learning	Traffic ML models, Expected traffic	None
Collection jobs	Inventory, live metrics, model training	Inventory (Fleet configuration), live (Edge Nodes)
Size of one tenant	Up to about 20 000 monitored objects	Any fleet size, no node limit
Workspace tabs	Findings, Monitoring, Traffic ML models, Infrastructure, Deployment, Routes & pipelines, Inventory, Policies, Collection health, AI Agent, Settings	Findings, Fleets, Edge Nodes, Policies, Collection health, AI Agent, Settings
Findings, Finding groups, notifications, assignment, findings actions	Yes	Yes, the same
Policy editor tabs	General, Actions, Automations	General (actions and automations in Settings)
Object-level triage	Stop monitoring this object, Not an anomaly	—
AI	AI Assistant, Concierge actions, Cribl Stream AI Agent	AI Assistant, Cribl Edge AI Agent
AI Agent decisions	Mark investigating, Dismiss for a time, Not an anomaly, Resolve, Stop monitoring	Mark investigating, Dismiss for a time, Resolve
Bank holidays	Exclude from ML training; Suppress new alerts (Traffic outliers by default, per policy)	Suppress new alerts (Fleet connectivity and Fleet coverage by default, per policy)
Topology Studio	Objects, groups, templates, Topology Alerts	Objects, groups (health only), Topology Alerts, Create with AI starting points
Configuration Guardian	`cribl_collection_health`	`cribl_collection_health` (titled Cribl Edge, with the Edge conditions)

See *Cribl Stream* for the Cribl Stream tenant, and *Cribl Edge* for the Cribl Edge hands-on walkthrough.

Next: the reference

Every policy with its defaults, the jobs and their schedules, the node directory and its bounds, the System Metrics contract, the settings, bank holidays, the AI Agent, the indexed finding events, the permissions, the Configuration Guardian conditions and a troubleshooting table: *Cribl Edge reference*.

Cribl Edge reference

The reference of the Cribl Edge tenant: the built-in policies and their defaults, the collection jobs, the node directory and its bounds, the System Metrics contract, the settings, bank holidays, the AI Agent, the indexed finding events, the permissions, the Configuration Guardian conditions, the REST endpoints and a troubleshooting table. The user

guide is *Cribl Edge*; the findings model shared by every dedicated tenant type is *Findings, Finding groups and findings actions*.

Built-in policies

A Cribl Edge tenant has **eleven** built-in policies: ten Edge policies and the Leader availability policy it shares with Cribl Stream. **Eight are on** out of the box; **Node disk**, **Node CPU** and **Node memory** are off (node resource monitoring is opt-in). Your own SPL policies on Edge Nodes are described in *Custom policies*.

Cribl Leader

Policy	What it detects	Default	Severity
Leader availability	The Leader API cannot be reached or used at a live run.	On	Critical

Fleets

Policy	What it detects	Default	Severity
Fleet connectivity	A share of the Fleet's active nodes is disconnected.	On	High → Critical
Version drift	Connected nodes stay off the baseline version after a rollout.	On	Medium → High
Configuration drift	Connected nodes do not run the Fleet's deployed configuration.	On	High → Critical
Fleet coverage	Fewer connected nodes than expected.	On	Medium → Critical
Node identity	Cloned nodes sharing a host name, or host names that keep changing.	On	Low
Source and Destination health	Nodes of a Fleet report a Source or Destination unhealthy.	On	Medium → Critical

Edge Nodes

Policy	What it detects	Default	Severity
Node connectivity	A selected node is disconnected or its heartbeat is too old.	On	High
Node disk	Disk used on every connected node, from its heartbeat.	Off	High → Critical
Node CPU	CPU used on a node over the last 10 minutes.	Off	Medium
Node memory	Memory used by the node's processes over the last 10 minutes.	Off	Medium

Default settings

What each policy applies to, its checks to open and to recover (one check per published node directory, every 5 minutes), and its own settings with their defaults.

Policy	One finding per	Open / recover	Settings and notes (defaults)
Leader availability	Leader	2 / 2	Checked at every live run, for the whole tenant. No Fleet scope.
Fleet connectivity	Fleet	3 / 2	Open at 20 % of active nodes and at least 2 nodes, recover below 10 %, critical at 50 % or when no node is connected. Nodes disconnected longer than the dormancy period (7 days; 1, 7 or 30) leave the count; a Fleet with no active node reads good.
Version drift	Fleet	1 / 2	Baseline: the Leader version (or the Fleet's most common version). Rollout grace 24 h (0–720), at least 1 node; the grace confirms. High when a node is newer than the Leader or 2 minor versions or more behind it. A node reporting no version yet is not counted.
Configuration drift	Fleet	1 / 2	Grace 30 min (0–10,080), at least 1 node, critical at 50 % of connected nodes.
Fleet coverage	Fleet	3 / 2	Expected: a configured minimum, or a learned size (the second highest daily peak of 7 days, less the allowed drop of 50 %). Reads *learning* until the Fleet has a peak of 2 nodes or more on two days. Critical when the Fleet has no node at all.
Node identity	Fleet	1 / 2	Host name changes in 24 hours: 3 (1–10).
Source and Destination health	Source or Destination of a Fleet	3 / 2	At least 1 node and 0 % of the nodes running it; critical at 50 % reporting trouble. A quiet Source (no traffic on any node) is skipped unless *judge quiet Sources* is on; a Destination is always judged.
Node connectivity	Edge Node	3 / 2	Heartbeat older than 900 s (60–86,400). Nodes come from typed host names and Splunk lookup tables; opens nothing until nodes are selected (*Needs nodes*). See *Node connectivity selection*.
Node disk	Edge Node	2 / 2	Open at 90 %, recover at 85 %, critical at 95 %. Nothing to forward.
Node CPU	Edge Node	2 / 2	Open at 90 %, recover at 80 %. Needs the System Metrics index (`node_cpu_percent_active_all`).
Node memory	Edge Node	2 / 2	Open at 90 %, recover at 80 %. Needs the System Metrics index (`node_memory_Used_percent`).

Settings every policy has. On / off; **severity** (Low, Medium, High, Critical; the policies above raise it in the cases listed); **checks to open** and **checks to recover** (1 to 12 consecutive directory publications, one every 5 minutes; recover 2 by default); and, except Leader availability, a **Fleet scope** (empty: every Fleet of the tenant) and **During bank holidays** (**Hold new findings** or **Keep opening findings**; see *Bank holidays*).

How a check works. A check is one publication of the node directory: the cohort, connectivity and disk policies judge the directory as it was published, never twice within 5 minutes. Grace starts and learned Fleet sizes are kept between runs.

How findings close by themselves. A finding recovers when the measured condition clears for the checks to recover. It closes automatically when its object is no longer listed by the Leader, when it leaves the tenant's scope (a Fleet deselected, a node no longer selected, a policy turned off), when the tenant's source changes (connection or Fleets), or when nothing evaluated it for 24 hours. A condition TrackMe cannot decide (*unknown*, for example a Fleet whose configuration target the Leader does not report) is never written as a recovery.

Bank holidays

A Cribl Edge tenant uses the bank holiday calendars of its scope, set in *Manage Bank Holidays → Tenants & settings* or *Configure tenant → Bank Holidays*: the system default calendars (`bank_holidays_scope = inherit`), no calendar, or the calendars it selects (`bank_holidays_calendars`). See *Bank holidays*.

Effect	On a Cribl Edge tenant
Suppress new alerts (`bank_holidays_suppress_a` inherits the system default, on)	While a holiday of the tenant's calendars is active, the policies set to hold open no new finding. A finding already open is still evaluated and can recover and close; a resolved or dismissed finding is not reopened by a held failure.
Exclude from ML training, exclude from SLA, monitoring windows	Do not apply (no traffic model, no entity) and cannot be set on a Cribl Edge tenant.

Per policy (*During bank holidays* in the policy editor):

Policy	Default
Fleet connectivity, Fleet coverage	Hold new findings
Node connectivity	Keep opening findings (it watches nodes selected as always connected)
Version drift, Configuration drift, Node identity, Source and Destination health, Node disk, Node CPU, Node memory	Keep opening findings
Leader availability	No setting (always judged)

A policy configuration saved before this setting existed reads its default. The calendars are read only while an enabled policy is set to hold. The hold is decided at the start of each check; when the calendars cannot be read nothing is held (a warning is logged). A held failure records no finding and sends no notification; the runs of the Edge Nodes and Monitoring policies jobs report the active holidays and the number of failing checks held. The drift graces (Version drift, Configuration drift) keep counting through a holiday.

Custom policies

A custom policy is an SPL search on the tenant's Splunk data whose result rows each name an Edge Node. How to write one: *Custom policies*.

- **Object type**: Edge Node, one result field holding the node id or the host name, matched without regard to case in the current node directory. A name that is not listed, or a host name shared by several nodes, is unresolved.
- **Runs, lifecycle and limits**: the same as on a Cribl Stream tenant, see *Custom policies*. Each enabled policy has its own scheduled report, `trackme_cribl_custom_<policy id>_tenant_<tenant>`, which runs as the tenant owner.
- A node that leaves the node directory also closes its custom findings.
- **Health**: *Collection health → Custom policies*, and the `custom_*` problems of the Configuration Guardian check `cribl_collection_health` (see *Configuration Guardian*).
- **Logs**: `index=_internal sourcetype=trackme:custom_commands:trackmecriblcustom`, one line per run or preview.

Collection jobs

Each Cribl Edge tenant has three collection reports, named `trackme_cribl_edge_<job>_<generation>_tenant_<tenant_id` run as the tenant owner, and the AI Agent report (below):

Job (Collection health label)	Schedule	What it does
`live` (**Edge Nodes**)	Every 5 minutes, at a minute offset of the tenant's own	Reads the Leader's node list in bulk, publishes the node directory and the Fleet figures, writes the history points, then evaluates every policy (chained in the same run), drains the notification emails, the automatic assignment rules and the webhook automations, and serves the pending System activity and live status reads. Time limit 240 s.
`inventory` (**Fleet configuration**)	Every 10 minutes; a new full cycle at most hourly	Reads the Fleets and their configuration (Sources, Destinations, Pipelines, Routes, Packs). Unchanged Fleets are confirmed without being read again. Time limit 240 s.
`policies` (**Monitoring policies**)	On demand only	Evaluates the policies again on the newest directory (**Run now** in Collection health). Time limit 150 s.
`trackme_cribl_` (AI Agent)	Hourly, when enabled	Scheduled triage by the Cribl Edge AI Agent. Off by default: the report stays unscheduled until the triage is turned on (see *AI Agent*).

There is no model training job. **Scheduled collection** off (Settings → Collection) unschedules the reports; **Collect now** and **Run now** still work. Every call to the Leader draws from the connection's shared **Leader request budget** (see *Cribl connections*); a run that would exceed it waits and resumes at the next run.

With an expired licence, scheduled collection follows the licence enforcement ladder like the Cribl Stream tenant: it continues while the deployment is read-only and stops when the ladder stops the tenants.

The node directory and scale

How the directory is built. The live run reads the Leader's Edge Nodes page by page and stores each node with its connection state, heartbeat, versions, configuration, OS, capacity, disk, tags, identity, and the health and last-period traffic of its Sources and Destinations. The pages add to one row per Fleet; the directory is **published** only once complete, so a reader never sees half of it, and a run stopped by its time limit continues at the next run. A very large directory may take several runs to build: the previous publication stays current while the build progresses.

Node state. *Connected*; *disconnected* when the Leader says so or when the node sent no heartbeat for five heartbeat periods (600 s at least); *unknown* otherwise. Node connectivity uses its own heartbeat limit for the selected nodes.

No limit on the number of nodes. Nothing caps the nodes of a tenant; every node is current within one live interval (no rotation, no sampling). The bounds that exist are on selections and on what one screen or record holds:

Bound	Value
Fleets per tenant	1 to 100
Edge Nodes per page of the Edge Nodes tab	100 (filters and search apply to every node)
Sources and Destinations listed per node	64 per direction (the rest counted)
Sources and Destinations kept per Fleet	100 per direction (the healthy, quietest ones leave first)
Versions listed per Fleet	20
Typed host names of Node connectivity	No count limit: the policy holds up to 384 KiB of configuration (about 15,000 names)
Lookup tables of Node connectivity	20 per policy, up to 500,000 distinct host names each (a larger lookup is refused, never truncated), read again every 15 minutes
Hosts read from the System Metrics index per policy run	120 000
Edge Nodes in one filtered Topology Studio group	20 000 (narrow the group beyond)
Edge Node entities per tenant in one Topology Studio view	100 (use groups for more)
Unresolved findings per tenant	5 000 (beyond, new failures are not recorded; Collection health says so)

History kept by TrackMe. At every publication, in the tenant's TrackMe metric index:

- `trackme.cribl.leader.edge_fleet.*` per Fleet: `nodes`, `connected`, `disconnected`, `events_in`, `bytes_in`, `events_out`, `bytes_out`, `dropped_events`, `version_drift_nodes`, `config_drift_nodes`;
- `trackme.cribl.leader.edge_node.*` for **every node**: `events_in`, `bytes_in`, `events_out`, `bytes_out`, `dropped_events`, `connected`, `disk_used_percent` — about seven points per node every 5 minutes, roughly 2 000 per node a day.

They draw the trends of the Fleets and Edge Nodes tabs, the disk history of a node, and the Topology Studio sparklines. Traffic figures are each node's **last report** to the Leader (a gauge), never volume totals.

System Metrics contract

The optional **System Metrics index** is a Splunk **metrics** index receiving the Edge System Metrics source of the tenant's Fleets. TrackMe reads it for the Node CPU and Node memory policies and for the CPU and memory KPIs of Edge Nodes in Topology Studio, and for nothing else.

Item	Contract
Metrics read	`node_cpu_percent_active_all` (CPU used, every core; available in Basic mode) and `node_memory_Used_percent` (memory used by the node's processes)
Node matching	The `host` dimension, matched to the node's **full** host name, case aside (never a short-name guess). A node with no row is not measured; an open finding on it ages out.
Window	The last 10 minutes, averaged per host, at every policy run
Cost	**One** bounded `mstats` search per policy run for every node, whatever the fleet size (at most 120 000 hosts)
Without the index	Node CPU and Node memory evaluate nothing (*Needs System Metrics*); the CPU and memory KPIs in Topology Studio read *System Metrics index not set*. Everything else works the same.

The index is not part of the tenant's source: choosing, changing or clearing it never closes findings. The **System activity** tab of a node does not use it either: it reads the node's own metrics store through the Leader, on demand.

Settings reference

Setting (where)	Default	Meaning
Cribl connection (wizard; Settings → Source & Fleets)	—	The connection to the Edge Leader. Locked after creation.
Fleets (same)	Every Fleet listed at creation	1 to 100 Fleets or Subfleets. Changing them starts a new source: the findings of the previous source close.
System Metrics index (same)	None	The metrics index of the Edge System Metrics (Node CPU and Node memory). Can change at any time without starting a new source.
Scheduled collection (Settings → Collection)	On	Runs the live job every 5 minutes and the inventory about hourly.
Bank holidays (Manage Bank Holidays → Tenants & settings; Configure tenant → Bank Holidays)	The system default calendars; Suppress new alerts on	The tenant's calendars and its one holiday effect. See *Bank holidays*.
Notifications (Settings → Notifications)	Off	Events opened, reopened, added to the group, escalated, recovered, dismissal ended and closed; minimum severity Medium; all policies; digest; charts on. A delivery account and recipients are required.
Assignment (Settings → Assignment)	None	Assignment groups, team email, ordered routing rules (policy, severity, Fleet, name), manual overrides. See *Assignment and automatic routing*.
Finding actions (Settings → Finding actions)	None	Drilldown, link and webhook actions, and the Automations that call them. See *Findings actions*.
AI Agent (Settings → AI Agent)	Scheduled triage off	The Cribl Edge AI Agent. See *AI Agent*.
Policies (Policies tab)	See *Built-in policies*	Saved for the tenant with their own revision, including *During bank holidays*.

The tenant's indexes, roles and owner are set in the wizard and edited like any tenant's.

AI Agent

The Cribl Edge AI Agent triages the tenant's findings from what TrackMe holds: the finding and its Edge detail, history, episodes and Finding group; its Fleet and Edge Node in the published node directory; whether the node or Fleet is still in a current publication of the directory; Collection health; and its own past decisions. It runs no search and calls no Cribl API. The user guide is in *Cribl Edge*; the shared model in *AI Agents for findings*.

Assessments: true positive, likely false positive, expected behaviour, node or Fleet not in use, needs investigation.

Settings (*Settings → AI Agent*; the core ones also in **Manage AI Agents automation**, *AI Agents automation*, and in *Configure tenant*), stored in the tenant record:

Setting	Field	Default	Range and notes
Scheduled triage	`ai_cribl_edge_agent_enabled`	Off (`0`)	Asking the agent needs only AI to be available.
Mode	`ai_cribl_edge_agent_mode`	Inspect	`inspect` (recommend only) or `act` (apply allowed decisions); scheduled runs.
AI provider	`ai_cribl_edge_agent_provide`	First available	An AI provider account name.
Review interval (days)	`ai_cribl_edge_agent_min_day`	1	0 to 365; a new episode or a changed Finding group is reviewed at once.
Findings per run	`ai_cribl_edge_agent_max_fin`	10	1 to 50, one per Finding group, most severe first.
Time budget per run (seconds)	`ai_cribl_edge_agent_max_runt`	3 000	60 to 86 400.
Custom instructions	`ai_cribl_edge_agent_custom_`	Empty	At most 4 000 characters, line breaks kept; followed by every run, asked or scheduled; never widen what the agent may do.
Allow Mark investigating	`ai_cribl_edge_agent_allow_i`	Allowed (`1`)	An open finding only.
Allow Dismiss for a time	`ai_cribl_edge_agent_allow_d`	Allowed (`1`)	A critical or high finding still failing only when its node or Fleet is gone from the node directory; never across the nodes of a Finding group.
Longest dismissal (days)	`ai_cribl_edge_agent_dismiss_`	7	1 to 90.
Allow Resolve	`ai_cribl_edge_agent_allow_r`	Not allowed (`0`)	Only after a measured recovery, and not while a member of its Finding group fails.

The shared filters of the automated AI reviews, `ai_automated_priority_filter` (by default critical and high findings) and `ai_automated_filter_expression`, apply to the scheduled selection with the finding severity as the priority. There is no *Not an anomaly* and no *Stop monitoring* decision on Edge. The agent never dismisses permanently, assigns, changes a severity, a policy, the node selection of Node connectivity or the collected Fleets.

Gone from the node directory. True only when a **current** publication of the directory no longer lists the node or the Fleet (or the Fleet is no longer selected), and the finding has not been evaluated for an hour. Unknown when the directory is not published or not current; never decided for the Leader availability finding. A Source or Destination finding is judged gone only with its Fleet.

Scheduled triage runs the report `trackme_cribl_edge_agent_tracker_tenant_<tenant_id>` hourly, at a minute of the tenant's own, when all of these hold: the AI Assistant is enabled, an AI provider is configured, the tenant is enabled, and scheduled triage is on. The report runs as the tenant owner, which must be a Splunk administrator account (`admin` or `sc_admin`), not `nobody`. Saving the settings schedules or unschedules it at once, and the tenant health tracker checks the same conditions every five minutes. A scheduled run never decides on a finding or Finding group an operator decided on or assigned.

Evaluate runs the agent in inspect mode on an unsaved draft of the custom instructions (one open finding, or a review of at most 10 open findings), optionally next to a run without instructions. It writes no note, no decision and no review record, and is indexed with the status `evaluation`.

Audit. Every run is indexed to the tenant summary index with the sourcetype `trackme:ai_agent:cribl_edge_agent:inspect` or `trackme:ai_agent:cribl_edge_agent:act`, with `launched_by` (UI, AI Assistant or automation) and `automated`; the audit dashboards label it **Cribl Edge**

AI Agent. Decisions appear in the finding's history under the actor **AI Agent**, and the reviewed finding carries one current `[AI Agent]` note.

```
index=trackme_summary sourcetype="trackme:ai_agent:cribl_edge_agent:*"␣
↪tenant_id="edge-example"
| table _time, sourcetype, launched_by, automated, status
```

Indexed finding events

Every transition of a finding and of a Finding group is written as an event, whatever the notification settings:

Item	Value
Index	The tenant's notable index (`trackme_notable` by default)
Sourcetypes	`trackme:cribl:finding` (a finding) and `trackme:cribl:finding_group` (a Finding group)
`object_category`	`cribl_edge` (indexed, with `tenant_id`)
Transitions	opened, reopened, recovered, closed, resolved, dismissed, dismissal ended, investigating, assigned / unassigned, snoozed / unsnoozed, escalated, severity changed; groups add member added and members changed
Key fields	`finding_id`, `transition`, `transition_label`, `transition_id` (deduplicate on it), `policy_id`, `policy_name`, `severity`, `display_severity`, `status`, `condition`, `reason`, `group_id`, `worker_group` (the Fleet), `entity_name`, `value`, `unit`, `assignee`, `actor`

Example: the Edge findings opened or reopened in the last day.

```
index=trackme_notable sourcetype=trackme:cribl:finding object_category=cribl_edge␣
↪tenant_id="edge-example"
    transition IN ("opened", "reopened")
| table _time, policy_name, worker_group, entity_name, severity, reason, finding_id
```

Permissions

Each tier needs the TrackMe capability **and** one of the tenant's matching roles; a Splunk administrator (`admin`, `sc_admin`) holds every tier on every tenant.

Action	Who
Read every tab; ask for a node's System activity or a Source / Destination live status; run drilldown and link actions	Any role of the tenant (`trackmeuseroperations`)
Triage findings and Finding groups (investigate, assign, snooze, dismiss, resolve, change the severity, group by hand), notes; send a webhook action; ask the AI Agent and let it act on its inspection	The tenant's power and admin roles (`trackmepoweroperations`)
Policies (on / off, severity, conditions, scope, during bank holidays)	A TrackMe administrator of the tenant (`trackmeadminoperations` and a tenant admin role)
Source & Fleets, Collection, **Collect now** / **Run now**, Repair workspace, Notifications, AI Agent settings, Assignment settings, Finding actions and Automations	A TrackMe administrator of the tenant
Create or delete the tenant; manage Cribl connections and Webhook accounts	TrackMe administrators; Cribl connections: Splunk administrators (see *Cribl connections* and *Webhook accounts*)

A System activity read asked by a TrackMe administrator of the tenant starts at once; any other role's read is served

by the next live run.

The tenant's bank holiday calendars and effect are set on the Bank Holidays page like for every tenant (*Bank holidays*). The **scheduled** AI Agent needs a tenant owner that is a Splunk administrator account (not `nobody`). Turning the AI Agent on, act mode, allowing a decision, a longer dismissal or custom instructions need a licence that allows changes; restricting the agent never does.

Node connectivity selection

Node connectivity watches the nodes that must stay connected, from two sources that add up:

Source	Rules
Typed host names	One per line (spaces and commas also separate), case-insensitive. `*` stands for any text (`dc-*` selects `dc-01.example.com`); `*` alone is refused. No count limit: the policy holds up to 384 KiB of configuration, about 15,000 names.
Splunk lookup tables	Up to 20 per policy, each a CSV or KV store lookup shared with the TrackMe app, on the local search tier or on a remote deployment through a TrackMe remote account, with the field holding the host names. Up to 500,000 distinct host names per lookup; a larger one is refused, never truncated. Multivalue fields give one name per value; values that cannot be a host name are counted as ignored.
Match mode (per lookup)	**Exact** (default): whole host names, case-insensitive; a value holding `*` is ignored. **Wildcard**: `*` matches any text; the editor warns about the cost of patterns with `*` at both ends. A lookup saved without a mode reads Exact.

Refresh. The policies run reads each lookup again every 15 minutes, the oldest first, and keeps its host names between reads. A lookup that cannot be read keeps its last good host names, so an outage of a remote search tier never deselects nodes; a lookup never read successfully selects nothing. Collection health reports `node_lookups_failing` when a lookup is unreadable, too large or empty.

Test selection runs as a search: it reads every lookup now (no cached names) and the latest Edge Nodes directory, then shows how many nodes the names select by state (connected, disconnected, unknown) inside and outside the policy's Fleet scope, each source with its status and its names matching or ignored, up to 50 selected nodes (disconnected first) with what matched them, and the names that select no node. It saves nothing. Policy managers of the tenant use it, like the rest of the editor.

Host names are selectors: they are not copied into the evidence of the findings.

Configuration Guardian

The tenant check `cribl_collection_health` raises **Cribl Edge collection needs attention** (warning) when collection does not run as configured, and clears by itself on the first run without a problem. The Collection health tab shows the same conditions.

Condition	In plain words
edge_node_metrics_missin	Half or more of the connected Edge Nodes report no metric to the Leader: their Sources, Destinations and traffic are not measured. Check the nodes' metrics settings in Cribl Edge, then **Collect now**.
system_metrics_not_arriving	No Edge System Metrics (node_*) reached the System Metrics index in the last 10 minutes: Node CPU and Node memory evaluate nothing. Check that the Fleets' System Metrics source is on and routed to that index, with the node host name as host.
system_metrics_unreadable	The System Metrics index could not be searched, or holds more hosts than one run reads.
node_lookups_failing	A lookup table of Node connectivity could not be read, holds more than 500,000 host names, or is empty, at the last policies run. A lookup that cannot be read keeps its last host names (no node is deselected); one never read selects nothing. Check the lookup, its sharing with the TrackMe app, and the remote account when it is on a remote deployment.
custom_schedule_drift, custom_policy_failing, custom_missed_schedule, custom_recovery_required	A custom policy's report is missing or out of date, its runs keep failing or missed three schedules, or a run holds execution capacity until **Verify and recover**. The same conditions as on a Cribl Stream tenant: see *Custom policies* and the Cribl Stream reference's Configuration Guardian table.
live_never_ran, live_not_running	No Edge Nodes run since the source was configured, or none for 20 minutes. Check that the reports exist, are scheduled and that their owner can run searches.
live_failing, policies_failing, inventory_failing	A job keeps failing (30 minutes for live and policies, 90 minutes for the inventory: never on a single failed run).
inventory_never_ran, inventory_not_running, inventory_not_published, inventory_cycle_slow	The Fleet configuration is not read, or its cycle takes more than a day at the Leader's request budget.
worker_directory_stale	The Edge Nodes directory is not refreshed: connections and versions are out of date. The Edge policies stop rather than judge an old directory.
capacity_reached	A capacity limit was reached (for example more than 5 000 unresolved findings).
notification_delivery_failing	Notifications are on and the emails have failed for 30 minutes. Collection never waits for delivery.

Each alert carries its recommended actions. See *Configuration Guardian*.

Also relevant: ai_automation_owner_is_system_user warns when the AI Agent's scheduled triage is on and the tenant owner is nobody or a system account, so the scheduled runs cannot start.

REST endpoints

The workspace uses the /trackme/v2/cribl/tenant/* endpoints shared with Cribl Stream (workspace, configure, collect, collect_result, repair, policies, findings, finding, triage, finding_groups, group_triage, group_members, notifications, notification_test) plus the Edge reads below, which answer cribl_edge_tenant_required on a Cribl Stream tenant. Every endpoint documents itself with describe=true and is listed in the REST API Reference page.

Endpoint	Purpose
`GET edge_overview`	Directory head, summary and the Fleet rows
`GET edge_fleet`	One Fleet (`group_id`): figures, Sources and Destinations, example nodes
`GET edge_nodes`	A page of nodes (up to 100), filters `group_id`, `state`, `version`, `config_state`, `platform`, `io`, `q`
`GET edge_node`	One node (`worker_id`) in full
`GET / POST edge_node_machine`	Read the latest System activity of a node over a range (15m, 1h, 4h, 12h) / ask for one
`GET / POST edge_io_status`	Read / ask for the live status of a Source or Destination on a node

The AI Agent uses the `cribl/tenant` operations shared with the Cribl Stream AI Agent, which serve the agent of the tenant's product: `agent_settings` (GET; POST saves them), `agent_findings`, `agent_evidence`, `agent_decisions` (reads), `agent_decide` and `agent_review` (writes of a run). Runs are started, followed and cancelled through `ai_findings_agent/agent` (`mode` inspect or act, an optional `finding_id`; `evaluate_instructions` for an evaluation), `ai_findings_agent/agent_status` and `ai_findings_agent/agent_cancel`.

Example:

```
| trackme url="/services/trackme/v2/cribl/tenant/workspace" mode="get"␣
↪body="{\"tenant_id\":\"edge-example\"}"
```

Troubleshooting

Symptom	What to check
The wizard cannot list the Fleets	The message names the cause: credentials refused, no right to read Cribl Edge, Leader unreachable, certificate not trusted (add the CA chain to the connection), Leader busy (retry). Test the connection in **Manage Cribl connections**.
"This connection lists no Edge Fleet"	The connection points at a workspace without Edge, or its API client cannot read Cribl Edge.
The workspace shows no Fleet or node	The first live run has not completed: **Collect now**, then **Collection health**.
A selected Fleet shows "No longer listed"	The Fleet was removed or renamed on the Leader: clear it in Settings → Source & Fleets and save.
Nodes show **No metrics**	Their Leader record carries no metric: their Sources, Destinations and traffic are not measured. Check their metrics settings in Cribl Edge.
A disconnected node opened no finding	Expected: Fleet connectivity judges shares (20 % and 2 nodes by default). Add the node's host name to **Node connectivity** for nodes that must never drop.
Node connectivity shows **Needs nodes**	Add host names or a lookup table in its editor: until then it opens nothing. Use **Test selection** to check which nodes they select.
Collection health: `node_lookups_failing`	See the Configuration Guardian table above. A lookup that cannot be read keeps its last host names, so no node is deselected meanwhile.
Node CPU / Node memory show **Needs System Metrics**, or find nothing	Set the System Metrics index; check `system_metrics_not_arriving` in Collection health, the metric names and that `host` is the node's full host name.
System activity shows only current figures, without charts	The node's System Metrics source is off in its Fleet: turn it on in Cribl Edge.
System activity says the node cannot be read	The node is disconnected, or the Leader request budget is exhausted for now: the next live run tries again.
Version drift opened while nodes are upgrading	The rollout grace (24 h by default) has passed. Raise it, or compare with the Fleet's most common version.
Fleet coverage reads *learning*	It needs daily peaks of 2 nodes or more on two days, or a configured **Expected connected nodes**.
Findings closed after changing the Fleets	Expected: a change of Fleets (or of connection) starts a new source.
Collection health: `edge_node_metrics_missing`	See the Configuration Guardian table above.
Fleet connectivity or Fleet coverage opened no finding during a holiday	Expected: they hold new findings on the bank holidays of the tenant's calendars by default. Choose **Keep opening findings** in the policy editor, or change the tenant's calendars.
The scheduled AI Agent does not run	AI Assistant disabled, no AI provider, scheduled triage off, the tenant disabled, or a `nobody` tenant owner. The AI Agent tab says which; give the tenant a Splunk administrator owner when needed.
The AI Agent did not dismiss a critical finding it judged expected	Expected: a critical or high finding still failing is dismissed only when its node or Fleet is gone from a current node directory, and never across the nodes of a Finding group. Decide it yourself, or let it recover.
A run failed and the reason is unclear	The collector log, `index=_internal sourcetype=trackme:custom_commands:trackmecriblcollect` (file `trackme_cribl_collect.log`): one summary line per run, a WARNING for an external failure (the Leader, its network, the Splunk tier), an ERROR for an internal TrackMe failure.

Monitor data sources and hosts with Cribl signals

Use a small stream of activity metrics to monitor the logs passing through Cribl. Cribl summarizes each host, index and sourcetype over a short window. TrackMe reads those summaries with `mstats` to discover data sources (DSM) and hosts (DHM), track event delay and alert when previously discovered subjects become stale.

This source is designed for estates where repeated searches across the original Splunk indexes are costly to operate. The original logs continue along their existing routes. **We recommend a separate monitoring route** for control over coverage and metrics delivery. Destination post-processing is an alternative when you want to observe all eligible logs sent to a selected Splunk destination.

Important

A signal means **Cribl observed the data**. It does not prove that Splunk successfully indexed the original logs. Use separate destination and indexing monitoring where that assurance is required.

Note

The `cribl_signals` hybrid tracker source is included in TrackMe 2.4.18. If you are testing a pre-release build, use one that includes this source; an earlier 2.4.18 build may not contain it.

How it works

```
Incoming logs: host, original index / sourcetype, parsed event time
  |
  +-- Activity route (Final off): copy -> preserve metadata / capture observation␣
↪time
  |     -> Aggregate per host / index / sourcetype, independently on each worker␣
↪process
  |        Window: 60s default; 5m optional; non-cumulative counts
  |        Values: event count, max observation epoch, max source event epoch
  |     -> Splunk metrics index (cribl_metrics), three numeric measurements per␣
↪summary
  |     -> TrackMe scheduled mstats: sum counts, max epochs; scope with dimension␣
↪filters
  |        +-- DSM: normally one entity per original index / sourcetype
  |        +-- DHM: normally one entity per host, retaining its feeds
  |     -> Event delay / observation age, count and delay history, retained␣
↪entities
  |     -> Thresholds, count-based outliers, impact score, SLA and alerts
  |
  +-- Original event -> existing pipelines -> original log destinations

No new signals: retained timestamps age, so silence can become an alert.
Signals do not prove original-log indexing or measure Splunk ingestion latency.
```

There is still **one pipeline per route**. Add a separate route before the existing matching log routes and turn its **Final** setting off. Cribl processes a copy through the activity pipeline while the original event continues to the following routes. Existing route pipelines do not need to be replaced. See Cribl route behavior.

The selected point must already have the host, original index, sourcetype and parsed source timestamp you want to monitor. Later changes to those fields are not reflected in signals produced earlier in the flow.

Choose where to observe activity

Both options produce the same metrics and work with the same TrackMe source. Choose one observation point for each population to avoid counting it twice.

Tip

Start with the separate non-final route approach. It gives you more control and flexibility:

- Select the sources or feeds to monitor, then expand coverage gradually.
- Adjust or disable the monitoring route without editing existing log routes or destination post-processing.
- Send metrics to a dedicated destination, independently of the logs.
- Inspect and troubleshoot the monitoring branch separately.

One monitoring route can cover several feeds through its filter; you do not need a separate pipeline for every source. Both approaches still share Cribl worker resources. This recommendation is about operational control, not a guarantee of lower overhead.

Choice	Separate non-final route (recommended)	Destination post-processing
Coverage	Events matching the monitoring route, before later route pipelines.	Eligible logs reaching the selected Splunk destination after route processing.
Configuration	Add a monitoring route; existing log route pipelines remain in place.	Attach or integrate one pipeline per destination; all its routes are covered.
Original logs	The monitoring branch handles a copy. Its aggregation has passthrough off.	Aggregation has passthrough on. Original timestamps and payloads are preserved.
Metrics delivery	A separate destination can centralize metrics independently of logs.	Logs and summaries use the same HEC destination and its queues.
Best fit	Default choice for controlled rollout, flexible scope and independent metrics delivery.	An explicit requirement to cover all eligible outgoing logs at selected destinations.

Follow steps 1–3 below for the recommended routing option. For destination coverage, use *Alternative: observe at the Splunk destination* instead of steps 2–3, then continue with step 4. Neither option proves indexing of the original logs.

The pipeline processes every selected event, but emits **window summaries rather than one metric event per original log**. Each worker process produces its own partial summaries. Splunk combines counts with `sum` and timestamps with `max`.

Set up a first tracker

1. Prepare Splunk and choose the scope

Create a **metrics index**, for example `cribl_metrics`, and a Splunk HEC destination that can write to it. The TrackMe search account needs read access to that index. Cribl's internal metrics may share the index, but they do not automatically provide the activity metrics in this recipe.

Start with one known source or feed. Confirm that events at the chosen route have non-empty `host`, `index` and `sourcetype` fields, and a numeric `_time` containing the parsed source event time in **epoch seconds**. Correct parsing before producing signals; do not replace a missing event time with the current time, which would make stale data appear fresh.

The examples use neutral names so other monitoring consumers can reuse the same metrics. The Cribl API add-on is not needed to produce or consume them.

2. Create the activity pipeline

Create a pipeline named `telemetry_activity_signals`. Add the following functions in order. Keep each function's **Final** setting off so the aggregate can reach the final Eval function. The route-level Final setting is separate.

A. Drop incomplete records from the activity branch

Add a **Drop** function with this filter:

```
!host || !index || !sourcetype ||
!Number.isFinite(Number(_time)) || Number(_time) <= 0
```

This excludes invalid subjects from the copied monitoring branch. The original event continues along the existing log routes. Monitor rejected records during onboarding; unexpected rejections indicate a parsing or metadata problem.

B. Preserve the original metadata and record observation time

Add an **Eval** function, filter `true`, with these added fields:

Field	Value expression
`data_index`	`index`
`data_sourcetype`	`sourcetype`
`event_time`	`Number(_time)`
`observed_at`	`Date.now() / 1000`

Add a **second Eval** function, filter `true`, setting `_time` to `observed_at`. This makes aggregation windows follow activity in Cribl while preserving the source timestamp separately. Old events replayed today therefore enter today's observation windows.

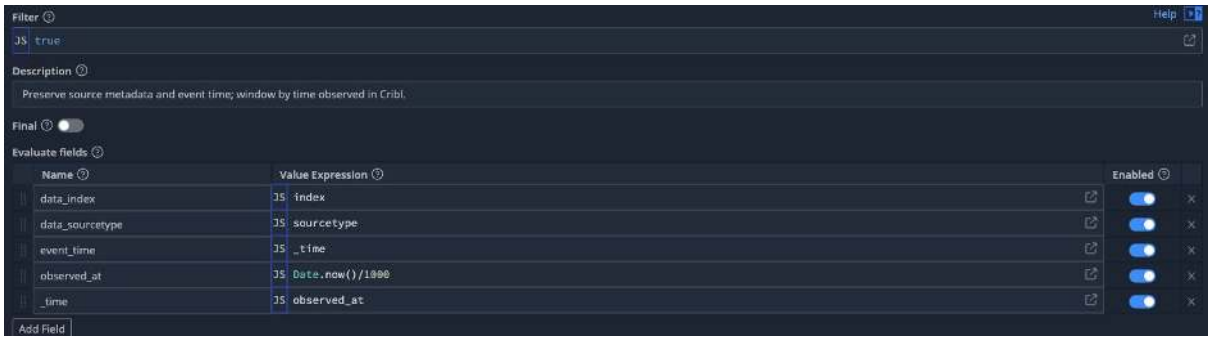


Fig. 1: Metadata preservation in the example. Its input already has numeric `_time` and its assignments share one Eval. For a new setup, follow the two Eval functions above, including `Number(_time)`, after the validation Drop.

Optionally remove unneeded payload fields from the copied branch before aggregation, retaining `_time`, the three grouping dimensions and both epoch fields. Retain any additional dimensions used by your trackers too.

C. Aggregate into activity metrics

Add an **Aggregations** function, filter `true`. Add these three expressions as separate aggregates:

```
count().as('telemetry.activity.events.count')
max(observed_at).as('telemetry.activity.last.observed.epoch')
max(event_time).as('telemetry.activity.last.event.epoch')
```

Use the following settings as a starting configuration:



Fig. 2: Enter each expression separately, then add the three grouping fields. Keep the function's Final switch off so its output reaches the next Eval.

Setting	Value
Group by fields	`host`, `data_index`, `data_sourcetype`
Time window	`60s` (default). For large estates, consider `5m` to reduce metric points at the cost of less frequent updates. See *five-minute sizing and tradeoffs*.
Cumulative aggregations	Off
Passthrough mode	Off
Metrics mode	On
Treat dots as literals	On
Lag tolerance / idle bucket time limit	`2s` / `2s`
Flush on stream close	Off
Aggregation memory limit	`64MB` as an initial aggregation limit

These settings are a starting configuration; size the aggregation limit for your workload. Keep counts non-cumulative so that `sum` does not repeatedly count earlier events. The memory limit is per worker process, not a cap on total worker memory. Monitor flush behavior and memory at your own cardinality. See Cribl Aggregations.

Keep **60 seconds** for the initial setup. For a large estate that can accept less frequent updates, see *Choose the aggregation window: 60 seconds or five minutes* for a five-minute option and its expected reduction in metric points.

D. Select the metrics index

After Aggregations, add an **Eval** function setting `index` to the expression `'cribl_metrics'`. Preserve the generated metric metadata, including `__criblMetrics`. Do not overwrite `host`, `data_index` or `data_sourcetype`.

The Splunk HEC destination uses Cribl's metric metadata to format metric points. Its multi-metric option affects serialization, not the three logical measurements in each summary. Check any destination post-processing as well. See the Splunk HEC event format.

3. Add a separate non-final route

Create a route named `telemetry-activity-signals` immediately before the existing log routes that handle your selected events:

- **Filter:** your source/feed selection, excluding internal metrics and the generated activity stream.
- **Pipeline:** `telemetry_activity_signals`.
- **Destination:** the Splunk HEC destination for your metrics index.
- **Final: Off**.

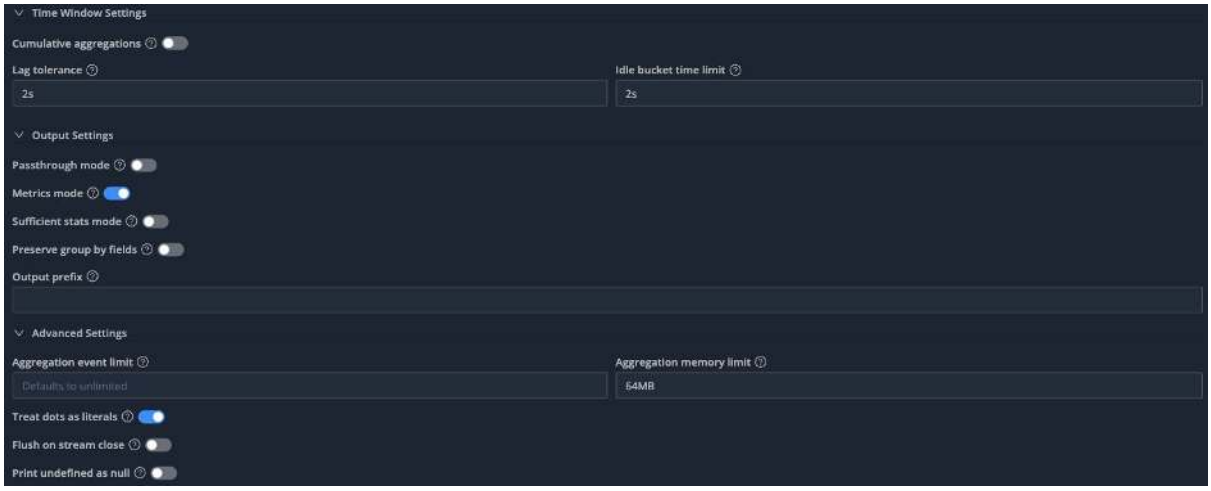


Fig. 3: Expand Time Window Settings, Output Settings and Advanced Settings. Enable Metrics mode and Treat dots as literals; leave cumulative aggregation and passthrough off.



Fig. 4: Set `index` to `'cribl_metrics'`. The example also adds optional `source` and `sourcetype` labels for the generated stream; the original sourcetype remains in `data_sourcetype`.

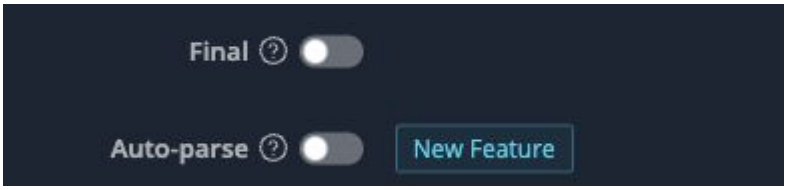


Fig. 5: In the route editor, turn Final off. This is the route-level switch, separate from the Final switch inside each pipeline function.

For example, after substituting your actual input ID:

```
__inputId === 'splunk_hec:example_logs' && index !== 'cribl_metrics'
```

Inspect the route order: an earlier matching final route prevents events from reaching this branch. Do not add a second activity route matching the same events, or count metrics will be duplicated. Keep the original routes' settings and destinations as they are.

Use Data Preview to check the field names and numeric values, then commit and deploy to a small scope. Confirm that original pipeline traffic continues and that fresh metrics arrive after the window flushes.

A separate route still shares workers and buffering resources. Configure and test the metadata destination's backpressure and queue policy so a metadata outage does not unexpectedly stall the original log flow.

Alternative: observe at the Splunk destination

Destination post-processing is a valid alternative when broad destination coverage is the priority. One attachment covers eligible logs from all routes using that destination, after their route pipelines. Repeat the attachment for each destination you intend to monitor. Prefer the separate-route approach when you need finer scope control, gradual rollout or independent metrics delivery.

This changes the observation boundary: records dropped earlier are absent, and prior sampling, cloning or metadata changes affect the resulting counts and identities. Broader coverage also means processing more traffic; qualify worker overhead and delivery behavior for that scope before expanding it.

Important

Do not attach the routing recipe above unchanged. Its aggregation consumes the copied events, and its final Eval changes every output to the metrics index. On the live log path, use **passthrough On** and restrict metric formatting to the newly generated summaries.

1. Create a separate destination pipeline

Download `telemetry_activity_destination.json`. Create a pipeline with that ID and paste the JSON into **Manage as JSON**, or import it. This is an example for a metrics index named `cribl_metrics`; adjust both the exclusion filter and summary index if you use another name.

The six functions run in this order, all with **Final off**:

Step	Function	Purpose
1	Eval: select eligible logs	Require host, index, sourcetype and a valid source epoch. Exclude existing metrics and the metrics index. Save the original timestamp internally.
2	Eval: observation window	Temporarily set `_time` to `Date.now()/1000` for eligible logs.
3	Aggregations: passthrough On	Group by host, index and sourcetype. Emit the same count and two maximum epochs over 60 seconds. Its Evaluate fields setting adds `__activity_summary=true` to summaries only.
4	Eval: restore original logs	Restore the original `_time` and remove its temporary field. Ineligible logs pass through without entering aggregation.
5	Eval: summary dimensions	Only on marked summaries, copy index to `data_index` and sourcetype to `data_sourcetype`.
6	Eval: summary destination	Only on marked summaries, set the metrics index and update `__criblMetrics` to use host, data_index and data_sourcetype as dimensions. Remove the temporary summary marker.

Reserve the example's `__activity_*` fields for this pipeline. The recipe preserves original `_raw`, index, sourcetype and source timestamps; Cribl can still add its usual pipeline bookkeeping. Existing transformations continue to apply. If you add grouping dimensions, add them to both the aggregation group-by list **and** the final metric dimension list in the JSON.

View the complete destination pipeline JSON

```
{
  "id": "telemetry_activity_destination",
  "conf": {
    "output": "default",
    "streamtags": [],
    "groups": {},
    "asyncFuncTimeout": 1000,
    "description": "Preserve original logs and emit 60-second activity summaries␣
↪to a Splunk metrics index.",
    "functions": [
      {
        "id": "eval",
        "filter": "typeof __criblMetrics === 'undefined' && index !==
↪'cribl_metrics' && !!host && !!index && !!sourcetype &&␣
↪Number.isFinite(Number(_time)) && Number(_time) > 0",
        "conf": {
          "add": [
            {
              "name": "__activity_original_time",
              "value": "_time"
            }
          ]
        }
      },
      {
        "id": "eval",
        "filter": "typeof __activity_original_time !== 'undefined'",
        "conf": {
          "add": [
            {
              "name": "_time",
              "value": "Date.now()/1000"
            }
          ]
        }
      },
      {
        "id": "aggregation",
        "filter": "typeof __activity_original_time !== 'undefined'",
        "conf": {
          "passthrough": true,
          "preserveGroupBys": false,
          "sufficientStatsOnly": false,
          "metricsMode": true,
          "timeWindow": "60s",
          "aggregations": [
            "count().as('telemetry.activity.events.count')",
            "max(_time).as('telemetry.activity.last.observed.epoch')",
            "max(Number(__activity_original_time)).as(
↪'telemetry.activity.last.event.epoch')"
          ],
          "cumulative": false,
          "shouldTreatDotsAsLiterals": true,
          "flushOnInputClose": false,
          "printUndefineds": false,
```

(continues on next page)

(continued from previous page)

```
        "groupbys": [
          "host",
          "index",
          "sourcetype"
        ],
        "lagTolerance": "2s",
        "idleTimeLimit": "2s",
        "flushMemLimit": "64MB",
        "add": [
          {
            "name": "__activity_summary",
            "value": "true"
          }
        ]
      }
    },
    {
      "id": "eval",
      "filter": "typeof __activity_original_time !== 'undefined'",
      "conf": {
        "add": [
          {
            "name": "_time",
            "value": "__activity_original_time"
          }
        ],
        "remove": [
          "__activity_original_time"
        ]
      }
    },
    {
      "id": "eval",
      "filter": "__activity_summary === true",
      "conf": {
        "add": [
          {
            "name": "data_index",
            "value": "index"
          },
          {
            "name": "data_sourcetype",
            "value": "sourcetype"
          }
        ]
      }
    },
    {
      "id": "eval",
      "filter": "__activity_summary === true",
      "conf": {
        "add": [
          {
            "name": "index",
            "value": "'cribl_metrics'"
          },
```

(continues on next page)

(continued from previous page)

```
            {
              "name": "source",
              "value": "'cribl:telemetry:activity'"
            },
            {
              "name": "sourcetype",
              "value": "'telemetry:activity'"
            },
            {
              "name": "__criblMetrics",
              "value": "[{dims: ['host', 'data_index', 'data_sourcetype'], values:␣
↪__criblMetrics[0].values, types: __criblMetrics[0].types}]"
            }
          ],
          "remove": [
            "__activity_summary"
          ]
        }
      }
    ]
  }
}
```

2. Preview before attachment

Use representative samples, including multiple hosts/feeds, replayed timestamps, missing metadata and pre-existing metric events. Compare original log payloads, metadata and timestamps before and after. Eligible logs should produce additional summaries; other events must remain on the normal output path. Check the metric view for separate host/feed series and the exact three required dimension names.

In our synthetic preview, **100 logs across three feeds produced 103 outputs**: 100 original logs and three summaries with counts **80, 15 and 5**. Original source timestamps remained unchanged; summary windows used current observation time. After mapping the internal metric dimensions, preview showed nine metric series: three metrics for each of three feeds. This validates the sample pipeline transformation, not a live destination rollout or an overhead benchmark.

3. Attach to the destination and qualify delivery

Open the Splunk HEC destination and select the pipeline under its post-processing setting. If a post-processing pipeline already exists, integrate these functions at the intended observation point within it; do not overwrite existing processing. Ensure earlier Final functions do not skip the activity functions. Repeat this configuration for each destination you intend to cover.

Use the HEC event endpoint and ensure its token permits both the original event indexes and the metrics index. Cribl uses `__criblMetrics` to distinguish metrics from logs on this shared output. Existing field-removal or serialization functions must preserve that metadata and the three dimensions on generated summaries. Then commit/deploy in a controlled scope and verify **both** original logs and metrics in Splunk before expanding coverage. See Cribl pipeline attachment and Splunk HEC event format.

Avoid duplicate observations. Do not enable the route and destination recipe for the same events into the same signal population. If logs fan out to several Splunk destinations, observing all copies inflates event counts. Choose one boundary, or emit a stable destination dimension and scope separate trackers to it. Do not sum those destination copies as unique source events.

This option shares the destination's delivery path: its queue or outage can delay signals together with logs. Observation still occurs before successful indexing. If metrics must go to an independent central destination, the separate-route option is usually simpler. TrackMe configuration is identical once the required metrics and dimensions arrive.

4. Verify the signals in Splunk

Before creating a tracker, check that the three metrics and their dimensions arrive in the metrics index. These searches inspect the signals stored in Splunk; they do not confirm indexing of the original logs.

Discover the metrics with mcatalog

Set the search time picker to **Last 60 minutes**, then run:

```
| mcatalog values(metric_name) AS metrics, values(_dims) AS dimension
  WHERE index=cribl_metrics metric_name=telemetry.activity.*
```

The result should list `telemetry.activity.events.count`, `telemetry.activity.last.observed.epoch` and `telemetry.activity.last.event.epoch`. `values(_dims)` lists dimension names, such as `data_index` and `data_sourcetype`; it does not list their values or calculate event counts.

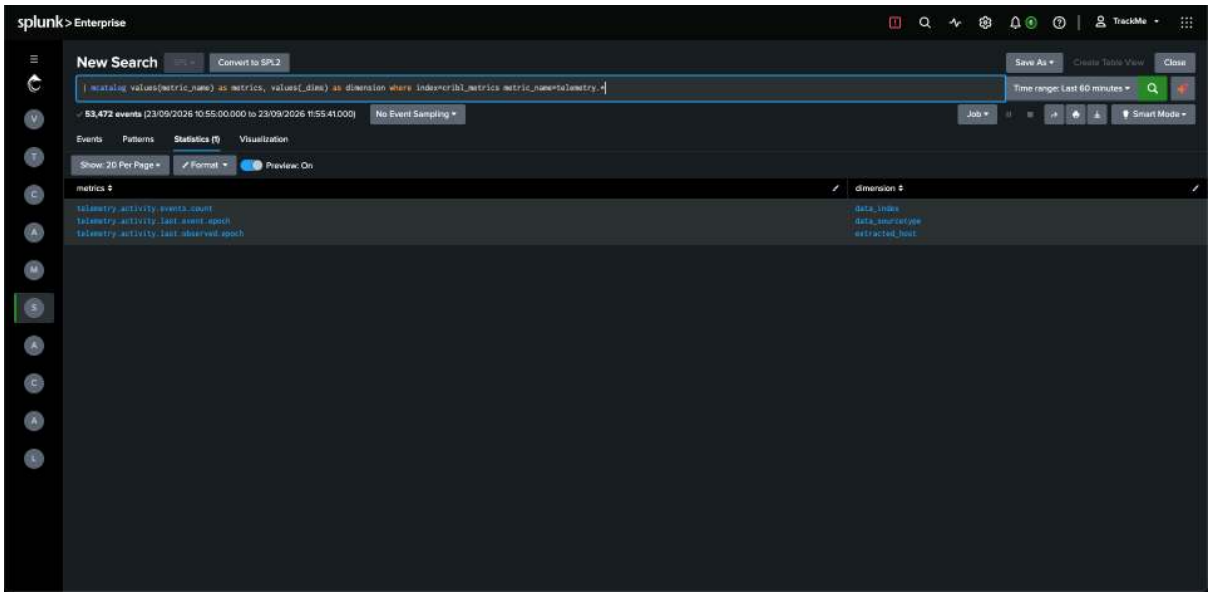


Fig. 6: Discover the metric names and dimensions before running the tracker query.

Inspect individual metric points with mpreview

Use `mpreview` to inspect the stored values and metadata. Start with the last observation timestamp:

```
| mpreview index=cribl_metrics
  filter="metric_name=telemetry.activity.last.observed.epoch"
```

Inspect the latest source event timestamp in the same way:

```
| mpreview index=cribl_metrics
  filter="metric_name=telemetry.activity.last.event.epoch"
```

Finally, inspect the event count:

```
| mpreview index=cribl_metrics
  filter="metric_name=telemetry.activity.events.count"
```

The screenshots show an additional `extracted_host` dimension in this example. The original host is also present as Splunk's `host` metadata beneath each preview row. Verify that `host` identifies the original producer; the tracker query below uses `host`, `data_index` and `data_sourcetype`.

`mpreview` returns a preview of metric points, with a default target of five points per time series per metrics index file. Its displayed result count is not the number of original log events. Keep a bounded time range and, for a large estate,

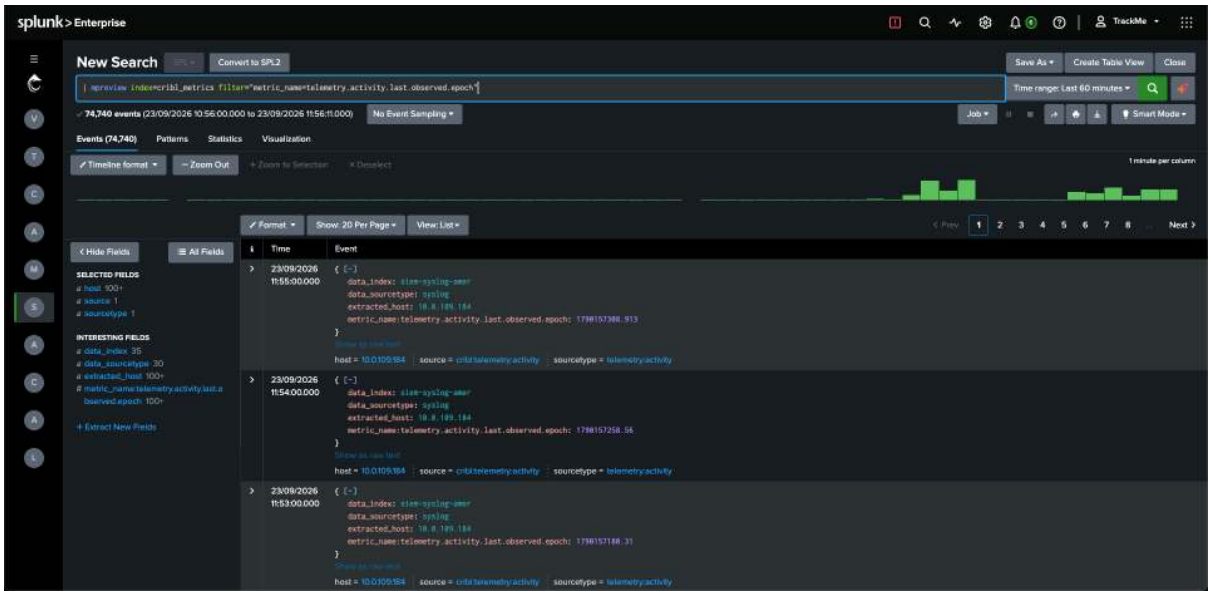


Fig. 7: The numeric metric value is the latest observation epoch recorded by Cribl for that aggregation window.

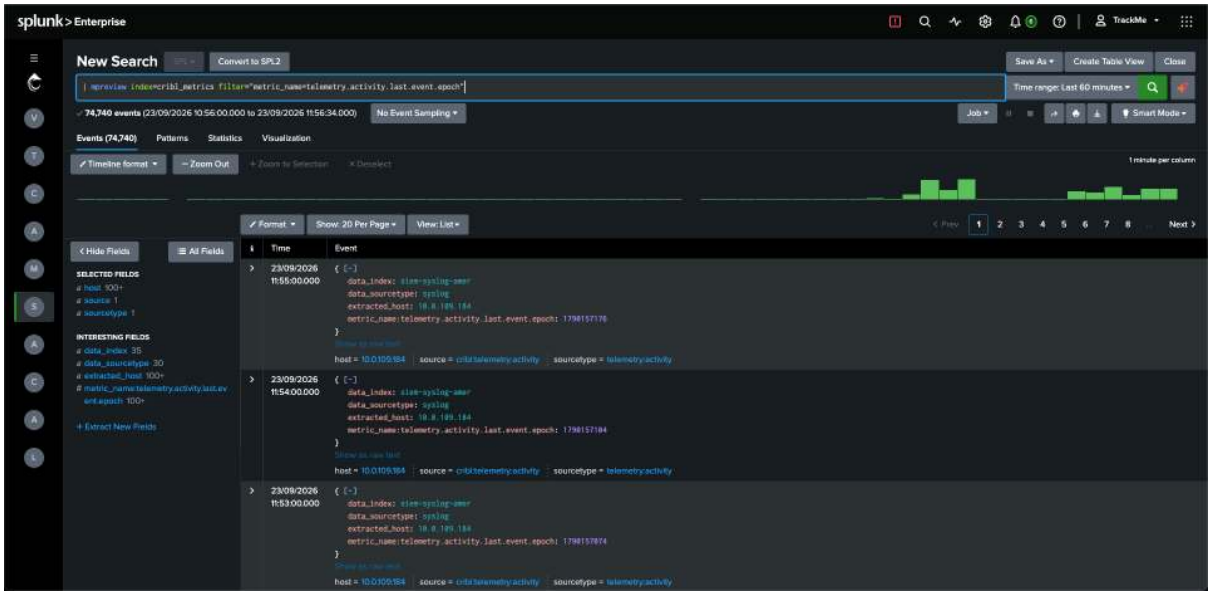


Fig. 8: This metric carries the latest source event timestamp. The displayed row time is the metric point's timestamp, not this metric's epoch value.

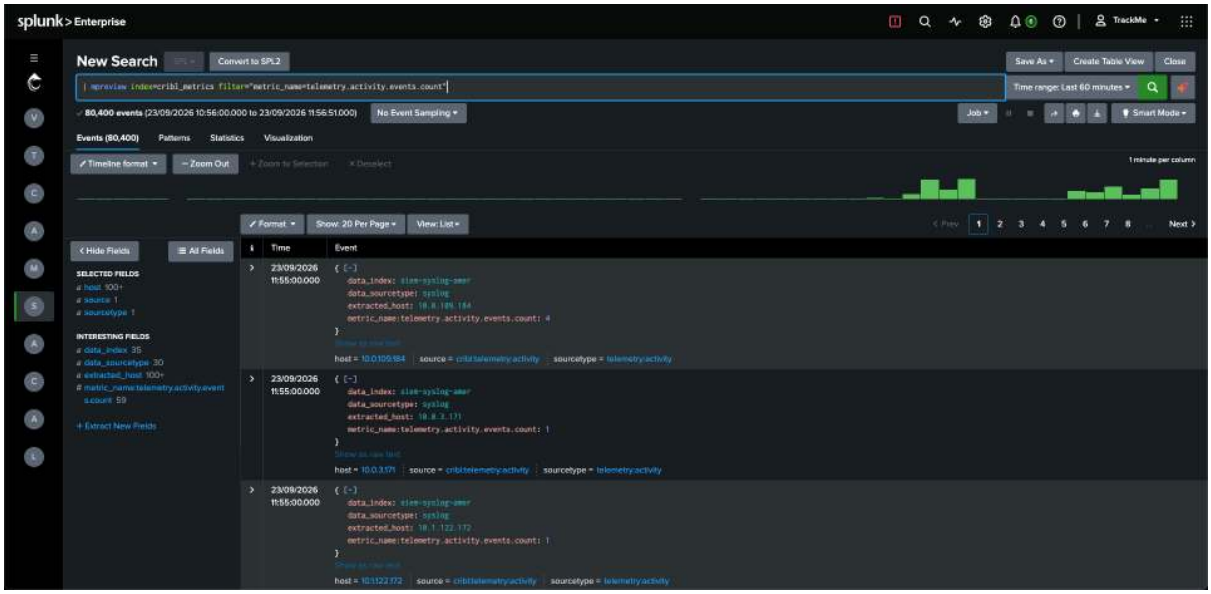


Fig. 9: Each value counts source events in one emitted aggregation summary.

add a known host to the filter, for example `filter="metric_name=telemetry.activity.events.count host=example-host"`. Use `mstats` for the event-count totals. See Splunk's mpreview reference and mcatalog reference.

Aggregate activity with mstats

Run this search to produce one row per subject:

```
| mstats
    sum(telemetry.activity.events.count) AS event_count
    max(telemetry.activity.last.observed.epoch) AS last_seen
    max(telemetry.activity.last.event.epoch) AS last_event
  WHERE index=cribl_metrics earliest=-15m latest=now
  BY host data_index data_sourcetype
| eval delay_seconds=round(now()-last_event,1),
       inactivity_seconds=round(now()-last_seen,1)
```

Each row should identify an original host/index/sourcetype combination. `index=cribl_metrics` selects the **storage index**; `data_index` names the **original log index**. Check a known subject's count and timestamps against the input, allowing for flush, delivery and search-window boundaries.

The two ages answer different questions:

- **Event delay:** `now() - last_event`. How old is the newest source event timestamp represented in the signals?
- **Inactivity:** `now() - last_seen`. How long since Cribl observed activity for this subject?

`last_seen - last_event` is not event-level latency: the maxima may come from different events. Neither value is Splunk's original-log indexing time.

5. Create the DSM or DHM hybrid tracker

In the TrackMe tenant, create a hybrid tracker and select **Cribl signals** as the search mode. Choose the Splunk deployment containing the metrics index. Use the default constraint:

```
index=cribl_metrics
```

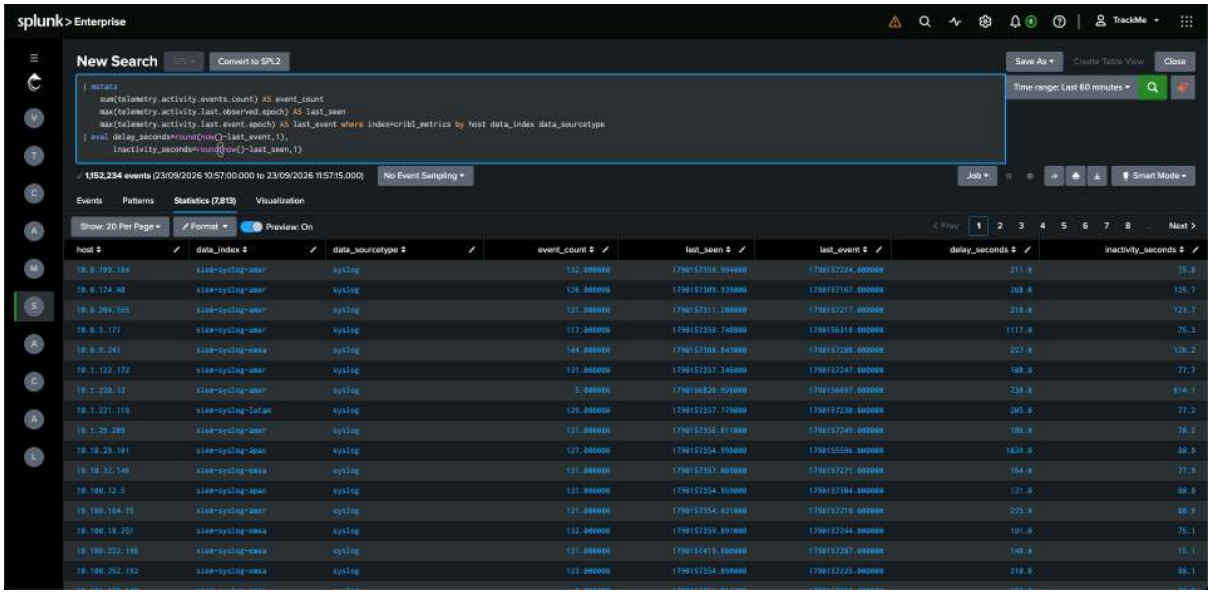


Fig. 10: Aggregated activity by host, original index and sourcetype. The screenshot uses the Last 60 minutes time picker; the query above explicitly uses the last 15 minutes.

This is an `mstats WHERE` constraint, not a complete SPL search. Add dimension filters when needed, for example:

```
index=cribl_metrics estate="example" region="emea"
```

Those dimensions must exist on the generated metrics. Add them to the producer's group-by list before using them for filtering or custom tracker grouping. Do not group by `*`: unnecessary dimensions increase cardinality and cost.

Keep the initial observation lookback at **-15m to now**, test discovery, review the results, then choose the schedule and create the tracker. The lookback must cover aggregation, delivery and scheduling jitter. It selects summary timestamps, not the source event timestamps stored as metric values; index-time controls do not apply to this source.

- **DSM** normally creates one entity per original index/sourcetype. Custom grouping and merged-sourcetype mode follow the existing DSM conventions.
- **DHM** normally creates one entity per host and keeps its index/sourcetype combinations. Custom host identifiers and additional combination dimensions must also exist in the producer's metric dimensions.

Use a separate tenant for an initial comparison with existing trackers. Filters scope input data; they do not automatically become entity identity. If two trackers discover the same identity in one tenant, they update the same entity. Use separate tenants or disjoint identities for overlapping estates, repeated hostnames, or comparisons between signals and indexed-log monitoring.

What changes in the entity view

The regular DSM/DHM overview shows **event delay**, **time since Cribl observation** and the **last known event count** in formatted single-value cards, alongside **impact score** and **SLA percentage**. Use the time picker to explore the recorded event-count and delay timeline. DHM also retains its index, sourcetype and configured extra-dimension breakdowns. Chart search and refresh actions, delay thresholds, monitoring windows, retained entities and the normal alert lifecycle remain available.

Splunk ingestion latency is marked as unavailable. Latency columns and DSM sampling controls are hidden. **Performance Metrics** retains supported count, delay and DSM host-count options; count-based outlier detection remains available when enough history exists. Raw sampling and parsing-quality views are excluded.

The search icon, row action and bulk search can investigate original logs using the recorded index, sourcetype and host or custom identifier. Verify the search deployment and time range: centralized metrics do not imply that the original logs are searchable there. No result is not, by itself, proof of data loss.

AI Assistant and the relevant Advisors receive the Cribl observation context. They distinguish event delay from observation age, treat latency as unavailable, and can help with delay thresholds, count-based models and history readiness. The AI Feeds Advisor remains available when AI is enabled. Monitor Cribl's own workers and destinations separately with the *Cribl infrastructure integration*.

Counts are totals over the last successful tracker lookback. They are not a cumulative counter, and overlapping tracker runs must not be added together. The history's five-minute counts describe observation windows; the newest bucket can be partial and missing buckets are not synthesized as zero.

A subject with no recent signal produces no new `mstats` row. TrackMe retains previously discovered entities and evaluates their stored timestamps, so silence can become an alert. Replayed older data does not move retained maxima backwards. A host that never appeared still needs expected-host inventory monitoring.

Scale, overhead and validation

The monitoring workload moves from searches over distributed original-log metadata to searches over a dedicated summary dataset. `tstats` already uses indexed metadata; the benefit comes from reducing the dataset and consolidating the monitoring searches, not from avoiding raw-payload reads that `tstats` never required.

Cribl adds per-event evaluation, aggregation state and summary delivery work. CPU demand follows input event rate and processing complexity; memory also follows active grouping cardinality and flush behavior. Output volume can be estimated from active feeds, aggregation windows and emitted summaries, but worker requirements need measurements at representative traffic and peak load.

For **200,000 hosts**, **two active feeds per host**, a **60-second window** and **one contributing process per feed/window**, the arithmetic is approximately **6,667 summaries/s**, containing **20,000 numeric measurements/s**. A 300-second window reduces the nominal output rate to one fifth under those assumptions, while increasing reporting delay. Neither scenario predicts worker count or license cost.

Read the *overhead and sizing guide* for worker resource considerations and a repeatable validation method. Use *Splunk storage estimation* to convert metric points into daily disk growth, retained volume and storage cost.

Before expanding the scope, validate peak event rate, active feed cardinality, worker-process fan-out, memory, output volume and search duration. Also test silence, replay, restart and metadata-destination failure. TrackMe still maintains its own entities and host/feed combinations; this source does not remove those capacity considerations.

Choose the aggregation window: 60 seconds or five minutes

60 seconds remains the default in this guide and its downloadable recipe. It provides responsive feedback during setup and troubleshooting. Consider **five minutes** for a large estate when reducing metric volume matters more than seeing new activity within about a minute.

With continuous activity and one summary per feed, worker process and window, the comparison is:

Table 2: Nominal output per active feed and contributing worker process

Measurement	60 seconds	Five minutes
Summaries per hour	60	12
Metric points per hour (three metrics)	180	36
Reduction in metric points	Baseline	**80% fewer**

A metric point here means one numeric measurement, not one HEC request or serialized event. The reduction assumes the same active feeds and contributing processes in both cases. Sparse activity and early flushes can change the result. See *Five-minute windows: estimated savings* for estate-wide examples.

How to try five minutes

1. In the activity pipeline, open **Aggregations** and change **Time window** from `60s` to `5m` (equivalently `300s`). In the function's JSON configuration, this is `"timeWindow": "5m"`. Update any description that still says 60 seconds.

2. Keep the metric names, grouping dimensions and non-cumulative counts. Preserve the recipe's passthrough setting: **off for the monitoring route**, **on for destination post-processing**. Keep **Flush on stream close** off.
3. Review the other flush settings. The recipe uses a **2-second idle bucket limit**; idle, event-limit or memory-limit flushes can affect the output rate. Start with the existing limits and measure the cadence before tuning them. A five-minute window alone does not guarantee one summary every five minutes.
4. Check the tracker lookback, schedule and delay thresholds before deployment. **-15m to now** remains a starting lookback, but it must cover aggregation, delivery and scheduling jitter in your environment. Running the tracker more frequently does not make Cribl flush sooner.
5. Save, commit and deploy to a limited scope. Allow several complete windows to settle, then compare count-metric samples, represented event counts, worker memory and CPU, destination queues and signal freshness. Use the searches in the *sizing guide*.

The metric contract and TrackMe source configuration stay the same. Both observation approaches support this setting; do not enable both for the same population during the comparison.

What changes for monitoring

The stored event and observation epochs keep their precision. **Their visibility is delayed**: an event arriving near the start of a five-minute window may wait almost five minutes for that window to end, followed by flush, delivery and tracker scheduling delays. This is not a guaranteed five-minute end-to-end bound. Event delay still means `now() - last_event`, but the tracker cannot use a newer timestamp until its summary arrives. Allow for that freshness budget in alert thresholds and test both normal activity and silence.

Counts also arrive in coarser batches. A longer lookback can retain visibility, but cannot recover minute-by-minute detail from five-minute summaries. Inspect the count timeline and count-based outlier models after changing the window; allow representative history at the new cadence and retrain models if needed.

Fewer summaries can reduce serialization, transport and downstream indexing work. Every selected event still needs processing, and a longer window may retain more distinct feeds in memory and produce larger flush bursts. **An 80% reduction in metric points is not an 80% reduction in worker CPU or memory.** Measure both before expanding coverage.

Troubleshooting

Symptom	Check
No signal rows	Route order and filter, deployment status, metric-index type, HEC index permissions, exact dotted metric names, and all required dimensions.
Counts higher than expected	Duplicate activity routes, simultaneous route and destination observation, destination fan-out, re-ingestion/retries, cumulative aggregation, and whether compared searches cover identical observation windows.
Increasing event delay with recent observation	Old data, replay or timestamp parsing. Cribl may be actively receiving events whose source timestamps are stale.
Many subjects become stale together	The signal pipeline, destination and tracker schedule, as well as the original sources. A broken monitoring path can resemble an estate outage.
Unexpected merged subjects	Reused identities across estates and missing identity dimensions. Dimension filters alone do not create distinct TrackMe entities.

Cribl signals: overhead and sizing

Use this guide to estimate metric output and assess the additional work on Cribl workers. The *Cribl signals recipe* aggregates events as they pass through Cribl; it does not query the original Splunk log dataset. For daily disk growth, replication and retention, see *Splunk storage estimation*.

`Download the sizing assumptions as Markdown.`

What drives the overhead

There are three main contributors:

- **Per-event processing:** preserve identity and timestamps and update the aggregation for every eligible event. Original log volume alone is not enough to estimate CPU: the same byte volume can represent very different event rates.
- **Aggregation state:** retain active host/index/sourcetype groups in each contributing worker process. Additional dimensions and longer windows can increase the number of groups held at once.
- **Summary delivery:** serialize, buffer and send the generated metrics. Longer windows can reduce this work, but early flushes and process distribution affect the actual output rate.

The recipe's 64 MB aggregation limit is per worker process, not a total worker memory cap. Cloning, other functions, serialization and destination queues also consume memory. Increasing the window is therefore not a guarantee of lower CPU or peak memory. Use the formulas below for volume planning and the measurement procedure to determine resource requirements.

Estimate the summary volume

Use the following variables:

- **H:** hosts in scope.
- **K:** mean index/sourcetype feeds per host.
- **A:** fraction of feeds active within an aggregation window.
- **F:** mean number of processes contributing a summary for each active feed/window. This is not the total worker count.
- **T:** aggregation window in seconds.
- **M:** numeric metrics per summary; this recipe uses three.

```
Logical feeds                  = H × K
Summary equivalents / second  ≈ H × K × A × F / T
Numeric measurements / second ≈ H × K × A × F × M / T
Numeric measurements / day    ≈ measurements / second × 86,400
```

This assumes one flush per active process/feed/window. Early flushes, routing duplicates, retries and changes to dimensions can alter the result. Worker load balancing and connection distribution can change F. Increasing the window also changes which feeds are active, so A need not remain constant.

The following are **arithmetic scenarios**, not measured deployment estimates. They assume 200,000 hosts, every feed active (A=1), one contributing process (F=1), and three metrics (M=3):

Table 3: Volume scenarios

Feeds / host	Logical feeds	Window	Summaries / s	Measurements / s	Measurements / day
1	200,000	60s	3,333	10,000	864 million
2	400,000	60s	6,667	20,000	1.728 billion
5	1,000,000	60s	16,667	50,000	4.320 billion
1	200,000	300s	667	2,000	172.8 million
2	400,000	300s	1,333	4,000	345.6 million
5	1,000,000	300s	3,333	10,000	864 million

If two processes usually see each feed during a window, double these values. Sparse feeds reduce them. A longer window reduces nominal output, but every eligible original event still needs processing, freshness becomes visible later, and more distinct feeds may remain in aggregation state.

Five-minute windows: estimated savings

The guide keeps a **60-second default**. A **five-minute window** is an optional volume reduction for estates that can tolerate slower updates. Follow *Choose the aggregation window: 60 seconds or five minutes* for the configuration steps and monitoring tradeoffs.

Holding active feed count and process fan-out constant, with one flush per active process/feed/window:

```
Five-minute points = 60-second points × 60 / 300
                   = 60-second points × 0.2
Reduction          = 1 - 0.2 = 80%
```

For the **200,000-host, two-feed** scenario above (A=1, F=1, M=3):

Table 4: Estimated daily output, not a measured production result

Measurement	60 seconds	Five minutes	Daily reduction
Summary equivalents	576 million	115.2 million	460.8 million
Metric points (numeric measurements)	1.728 billion	345.6 million	**1.3824 billion**

These are fewer samples of the same metrics and dimensions, not 80% fewer distinct time series or monitored entities. An otherwise identical continuously active estate with 5,000 hosts and two feeds per host would go from **43.2 million to 8.64 million metric points/day**. This smaller scenario is also arithmetic, not a measured deployment result.

The 80% ratio is most useful for continuously active feeds. A feed that sends only one event every ten minutes may still need one summary per event with either window, so it may see little reduction. More processes can contribute to a feed over five minutes, and early flushes can create additional summaries. Re-estimate A and F for each window instead of applying 80% to every deployment.

Validate with settled, equal-duration measurements covering several complete five-minute windows and comparable input. Exclude the configuration-change interval and allow delivery to complete. Compare the **number of count-metric samples**, not just the sum of their values: fewer samples should still represent the same input event population over equivalent, fully delivered windows. Check peak memory and flush-time queue pressure as well as average CPU.

Do not infer CPU from host count

For illustration, **100 TB/day** of original logs corresponds to approximately **1.16 million events/s** at an assumed mean event size of 1,000 bytes (decimal units), distributed across the estate's workers. Per-event work and active grouping state must both be measured; scaling host count alone does not predict them.

For the two-feed, 60-second scenario, there are 576 million summary equivalents per day. An **assumed** effective serialized size of 300–1,000 bytes per summary, including all three measurements and dimensions, gives **172.8–576 GB/day** of additional metadata. These are sensitivity inputs, not measured encoding sizes. They exclude protocol overhead and do not predict compression, license usage, index storage, replication or retention cost.

Measure actual destination bytes and retained Splunk footprint before costing the design. See *Cribl signals: Splunk storage estimation* for the storage formulas. Additional provisioned worker capacity may be unnecessary while there is sufficient headroom, but the branch still consumes resources. Compare that cost with the measured reduction in distributed monitoring searches using your own commercial terms.

Repeat the measurements in your environment

Measure a settled window with the activity branch off and another with it on, using the same input replay where possible. Keep original log processing active. Repeat the cycle; compare the busiest process as well as the group average.

This search measures a 15-minute activity window. Use explicit time bounds and the corresponding duration when comparing runs:

```
| mstats
    count(telemetry.activity.events.count) AS summaries
    sum(telemetry.activity.events.count) AS represented_events
  WHERE index=cribl_metrics earliest=-15m latest=now
  BY host data_index data_sourcetype
| stats dc(host) AS hosts count AS feeds
        sum(summaries) AS summaries sum(represented_events) AS represented_events
| eval measurements=summaries*3,
       summaries_per_second=round(summaries/900,2),
       measurements_per_second=round(measurements/900,2),
       represented_events_per_second=round(represented_events/900,2)
```

For worker resources, the following example uses Cribl internal metric names and dimensions. Adapt them to your exporter and preserve both **node identity** and **process identity** when comparing several workers:

```
| mstats
    avg(cribl.logstream.system.cpu_perc) AS cpu_pct
    count(cribl.logstream.system.cpu_perc) AS cpu_samples
    avg(cribl.logstream.system.mem_heap_used) AS heap_bytes
    avg(cribl.logstream.system.mem_rss) AS rss_bytes
    max(cribl.logstream.aggregators.total_memory_usage) AS aggregation_peak_bytes
  WHERE index=cribl_metrics earliest=-15m latest=now
  BY group event_host cribl_wp
| eval heap_mib=round(heap_bytes/1048576,2),
       rss_mib=round(rss_bytes/1048576,2),
       aggregation_peak_mib=round(aggregation_peak_bytes/1048576,2)
```

See Cribl internal metrics for the underlying measurements. Missing dimensions can remove rows from a grouped search, so first confirm the names present in your metric index.

Before publishing a capacity estimate

1. Vary event rate and active feed cardinality independently with representative event sizes, peak traffic and process distribution.
2. Record CPU, memory, aggregation state, emitted bytes, destination pressure and queue behavior. Test restarts and failure of the metrics destination.
3. Compare equivalent `mstats` and existing tracker searches: same subjects, time semantics and polling frequency. Record wall time, concurrency and search-resource demand. A screenshot of different searches is not a benchmark.
4. Test silent subjects, replay, invalid/future timestamps, repeated identities across estates and failure of the signal path itself.
5. Measure the complete TrackMe lifecycle at target entity and combination counts, including persistence and scheduling, before claiming estate capacity.

The production proposition is a smaller, centralized activity dataset with a clear observation point. Exact resource cost and estate capacity depend on the workload, worker distribution and available headroom.

Destination post-processing scope

The same output-rate arithmetic applies to destination signals, but use the **eligible traffic reaching those destinations** as the input population. Earlier filtering or sampling changes that population. Measurements for a limited route do not establish the overhead of observing an entire estate.

CPU work occurs for every eligible input event; memory follows active aggregation groups and windows per worker process. Destination fan-out can repeat that work and create duplicate summaries. A longer window mainly reduces output frequency; it does not remove per-event evaluation or guarantee lower peak memory.

For destination sizing, compare equal input rates, event sizes and feed cardinalities with the added functions disabled and enabled. Include existing post-processing, original-log throughput, worker CPU/heap, aggregation state, queue/backpressure, output errors and signal freshness. Report the number of observed destinations and copies per event. Keep original-log delivery within the deployment's normal targets while increasing scope.

Volume such as TB/day alone cannot determine CPU cost or worker count: event rate, metadata cardinality, process distribution and remaining worker capacity are also required.

Cribl signals: Splunk storage estimation

Estimate the **additional storage for Cribl activity metrics** from the number of active feeds, how often summaries are emitted, and the effective stored size of each metric point. Then apply retention and capacity headroom. This gives a repeatable calculation that can be adjusted to your estate.

For example, **150,000 hosts with three feeds each**, a **five-minute window**, **1.9 summaries per active feed/window** and **64 stored bytes per point** give approximately **47.3 GB/day across the cluster**. Thirty days of retention is about **1.42 TB**, or **2.03 TB provisioned at 70% target utilization**. These are conditional estimates: the sections below explain each input and how to replace it with a value measured in your environment.

Important

64 bytes per point is an illustrative planning coefficient, not a fixed Splunk storage rate. It is a rounded reference value informed by limited measurements with RF=3/SF=2, rather than a validated long-term capacity rate. Use the sensitivity examples and calibrate before committing capacity. Dimension cardinality, compression, bucket lifecycle and replication affect the result. The higher examples are not guaranteed upper bounds.

This page covers locally retained metrics on a Splunk Enterprise indexer cluster. Original logs, TrackMe's separate history indexes, compute, network, backups and Splunk licensing are outside the calculation. SmartStore requires separate object-storage and local-cache sizing. For Splunk Cloud, use the contracted retention and storage model rather than treating these disk estimates as a subscription price.

Use the *setup guide* for configuration and the *overhead guide* for Cribl worker sizing.

What is being counted?

A **feed** is a distinct combination of host, original index and sourcetype, plus any additional grouping dimensions. A host sending three sourcetypes to one index normally represents three feeds. Changing identity dimensions creates additional combinations.

The recipe emits **three metric points per summary**:

- Event count for the aggregation.
- Latest observation timestamp in Cribl.
- Latest source-event timestamp observed.

A metric point is one numeric measurement. It is not an original log event, an HTTP request, or necessarily one serialized HEC event: multiple measurements can be delivered together. In particular, `sum(telemetry.activity.events.count)` counts represented original events, while `count(telemetry.activity.events.count)` counts samples of that metric.

Daily metric points and disk growth

```
Points/day = H × K × A × E × (86,400 / W) × M
GB/day     = Points/day × B / 1,000,000,000
```

Table 5: Inputs to the estimate

Input	Meaning	Worked example
H	Hosts in scope	150,000
K	Mean feeds per host	3
A	Fraction of those feeds active in an aggregation window	1 (all active)
E	Mean emitted summaries per active feed/window, across all contributing processes	1.9
W	Configured aggregation window in seconds	300 (five minutes)
M	Numeric metrics per summary	3
B	Effective cluster-wide stored bytes per logical metric point, including replicas	64 (illustrative)

E=1 means one summary for each active feed in each window. A feed handled by several worker processes or flushed early can produce more than one summary. **E=1.9 is an example, not a default or a limit.** Measure it from the emitted samples. The overhead guide uses **F** for the contributing-process factor, assuming one flush per process/feed/window; E also accounts for repeated flushes.

K and A should describe the same population and time window. Avoid combining all-time distinct feeds with a short-window activity fraction from a different population. For changing workloads, calculate points separately for representative periods and add the results.

```
150,000 × 3 × 1 × 1.9 × (86,400 / 300) × 3
    = 738,720,000 points/day

738,720,000 × 64 / 1,000,000,000
    = 47.27808 GB/day across the cluster
```

Do not multiply this result by RF or SF again: B already includes the cluster's physical copies. All GB and TB on this page are decimal (1 GB = 10^9 bytes; 1 TB = 10^12 bytes). A GiB is 2^30 bytes; convert units before combining values from dashboards or storage invoices.

Examples for 150,000 hosts

The following estimates assume a five-minute window, all feeds active, three metrics per summary and B=64 bytes/point, including the assumed RF=3/SF=2 cluster footprint. They show the effect of feed count and summary frequency.

Table 6: Additional cluster disk growth (GB/day)

Feeds per host	E=1 summary/feed/window	E=1.9 summaries/feed/window
1	8.3	15.8
2	16.6	31.5
3	24.9	47.3
5	41.5	78.8

Test the estimate against different storage coefficients. For the three-feed, E=1.9 example, changing B gives:

Table 7: Stored-size sensitivity

Cluster bytes per point	Additional GB/day
64	47.3
128	94.6
256	189.1

These coefficients are planning inputs, not a confidence interval. Measure B with representative dimension values, grouping cardinality and index settings. Changing RF/SF requires a new coefficient or the component model below.

Aggregation window and freshness

Holding every other input constant in the three-feed, E=1.9, B=64 example:

Table 8: Effect of the configured window

Window	Additional GB/day
60 seconds	236.4
Five minutes	47.3
Ten minutes	23.6
Fifteen minutes	15.8

Moving from 60 seconds to five minutes gives one fifth as many configured windows, or **80% fewer nominal points**. This is conditional on A, E and B remaining unchanged. Sparse activity, early flushes, process fan-out and bucket overhead can change the actual reduction.

Longer windows delay fresh information and can retain more active aggregation state. They do not remove per-event processing or guarantee lower Cribl memory usage. The recipe keeps 60 seconds as its default; consider five minutes when lower volume is more important than faster visibility. Ten- and fifteen-minute rows illustrate the arithmetic, not a recommended default. See *Choose the aggregation window: 60 seconds or five minutes* for configuration, flush settings and tracker lookback.

Replication and search factor

For a fully replicated, single-site cluster using local storage, a useful component model is:

```
Cluster bytes ≈ RF × R + SF × I + O
RF=3, SF=2    ≈ 3 × R + 2 × I + O
```

R is the compressed raw-journal size of one copy, I is the searchable index-file size of one copy, and O is other cluster-wide bucket and filesystem overhead. Apply the terms to the same data population. Searchable copies are included within RF: **RF=3/SF=2 does not mean six full copies**. See Splunk's bucket-copy model.

Metrics use dedicated index structures, and floating-point compression is enabled by default. However, do not assume that replicated metrics have no raw journal: `metric.stubOutRawdataJournal` does not take effect for indexes with `repFactor=auto` in an indexer cluster. See the Splunk indexes.conf reference.

This is why multiplying JSON payload size by six, or applying a generic event index compression ratio, is not a reliable metrics-storage calculation. Prefer a measured B for the intended configuration. For multisite clusters, use the effective site replication/search policy and copy distribution.

Retention, provisioned volume and price

```
Retained GB    ≈ daily GB × retention days
Provisioned GB ≈ retained GB / target utilization
```

At 47.27808 GB/day, using an illustrative target utilization of 70%:

Table 9: Additional storage for the worked example

Retention	Retained TB	Provisioned TB at 70%
30 days	1.42	2.03
90 days	4.26	6.08
180 days	8.51	12.16

Retention is an approximate steady-state calculation; bucket boundaries and workload changes affect the actual footprint. On four evenly loaded indexers, 2.03 TB is approximately **507 GB additional provisioned storage per indexer**. Add existing indexes and account for imbalance, maintenance and recovery. The 70% value is an example planning choice, not a Splunk requirement or a substitute for a deployment-specific capacity margin.

If storage is billed on provisioned capacity at a price P per GB/month:

```
Monthly storage charge ≈ provisioned GB × P
30-day example         ≈ 2,026 × P
```

If billing is based on actual occupied storage, use average billable occupancy instead. Add any separate IOPS, throughput, backup or transfer charges. Daily disk growth, retained capacity and monthly cost are different quantities; **this calculation is not a Splunk license-usage estimate**.

Calibrate the estimate in your environment

1. Use a dedicated metrics index or another method that isolates only the signal metrics. Keep the grouping dimensions, aggregation settings and replication policy representative of the planned deployment.
2. Choose matching start and end times over several complete aggregation windows. Count the logical metric samples once through a normal distributed search. Do not sum independent searches of replica copies.
3. Measure that index's footprint across **all indexer peers**, including all physical copies, at both boundaries. Wait for replication to settle and use consistent units and measurement methods. Dashboard index size and filesystem allocated bytes are not interchangeable.
4. Ensure no freezing, deletion, tier movement or unrelated data changed the measured footprint. A nearly full index or shared volume can remove old data while new data arrives, making net growth understate storage demand.
5. Repeat over representative periods, ideally several days, covering normal bucket rollover and busy traffic. A new index's one-time size divided by its sample count is only an initial indication, not a stable growth coefficient.

Use explicit, matching time-picker boundaries with this search (replace the index name if necessary):

```
| mstats
    count(telemetry.activity.events.count) AS p_count
    count(telemetry.activity.last.observed.epoch) AS p_seen
    count(telemetry.activity.last.event.epoch) AS p_event
  WHERE index=cribl_signals
| eval total_points=p_count+p_seen+p_event
```

Allow delivery to complete for the selected interval. Check all three counts: they should be consistent with the recipe. Unexpected differences should be investigated before using the result for sizing. The index name above selects the metrics storage index, not the original-log `data_index` dimension.

For intervals without data expiry or other footprint changes:

```
B ≈ (cluster bytes at end − cluster bytes at start) / new logical points
```

Measure E by dividing emitted count-metric samples by the number of active feed/window combinations over the same period, using the configured window and all grouping dimensions. Window alignment and flush timing can affect a short measurement. Recalibrate after changing dimensions, window/flush settings, worker distribution or the Splunk storage configuration.

The result is a traceable storage estimate: metric output drives daily growth, retention determines the retained dataset, and headroom and commercial terms determine provisioned capacity and price.

Cribl API add-on and Flex use cases (annex)

An earlier way to monitor Cribl

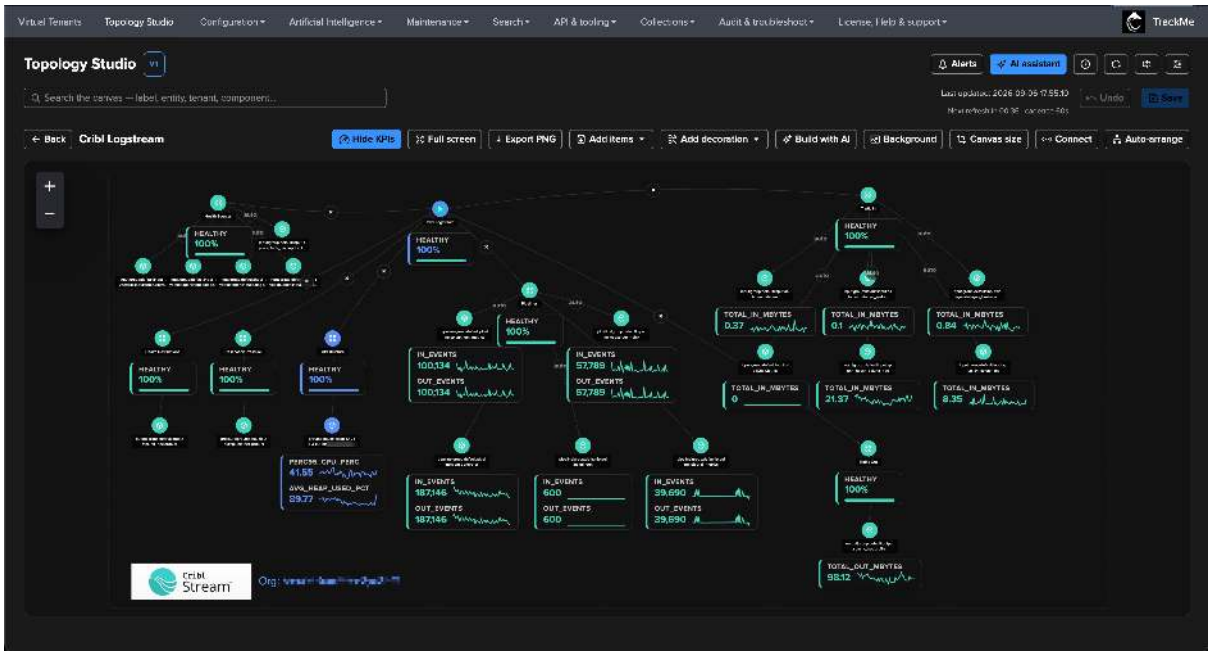
This page documents the integration that came **before** the native Cribl tenants: the `TA-trackme-cribl` add-on (SPL commands for the Cribl API) and the Flex Objects library of Cribl use cases. Both remain supported for custom monitoring built on your own SPL inside an entity tenant.

To monitor a Cribl Stream deployment or a Cribl Edge fleet, use the dedicated tenants described in *Cribl Stream & Cribl Edge*: they need no tracker to write and no add-on.

The sections below cover the earlier, Flex-based integration and the `TA-trackme-cribl` add-on.

Flex Objects use cases. TrackMe can also turn Cribl's internal metrics in Splunk into monitored Flex entities, with live health, traffic trends and alerts across pipelines, sources, destinations and worker infrastructure. Follow the *Cribl monitoring with Flex Objects guide* for the use-case library, setup instructions and topology walkthrough.

Explore the Flex Objects Cribl use cases



About the Cribl Stream & Edge API add-on

The `TA-trackme-cribl` add-on provides SPL commands to interact with the Cribl API of Cribl Stream and Cribl Edge. It is separate from the Cribl connections of the native Cribl Stream and Cribl Edge tenants, which are declared in TrackMe itself.

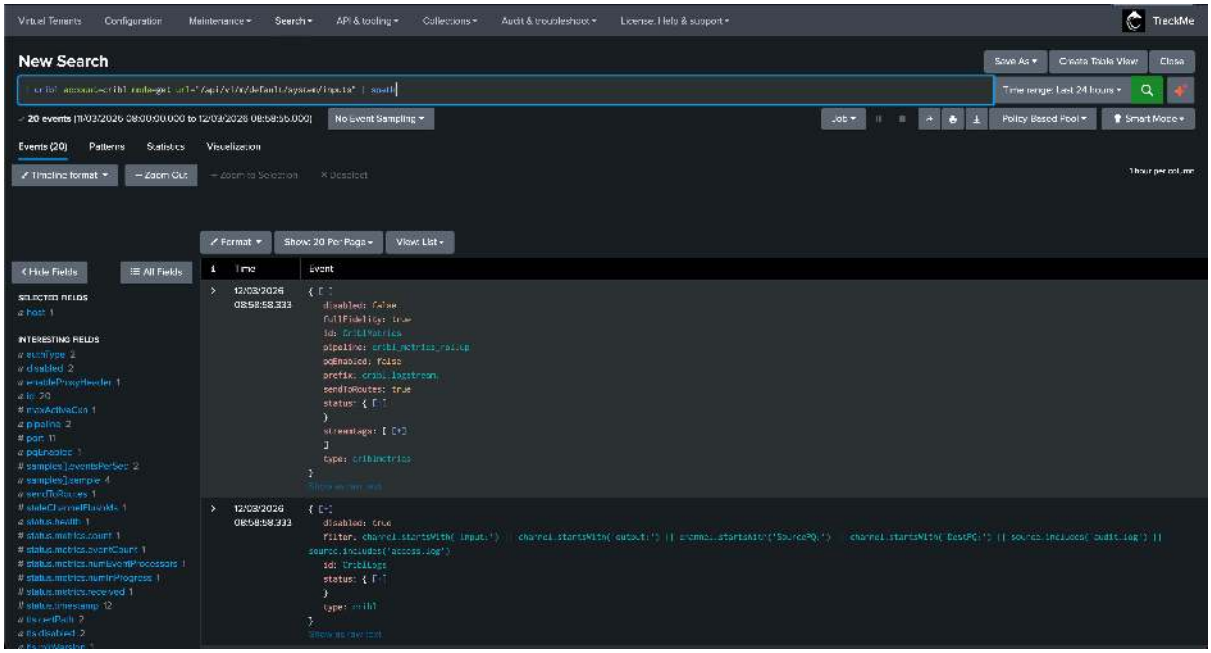
TA-trackme-cribl add-on overview

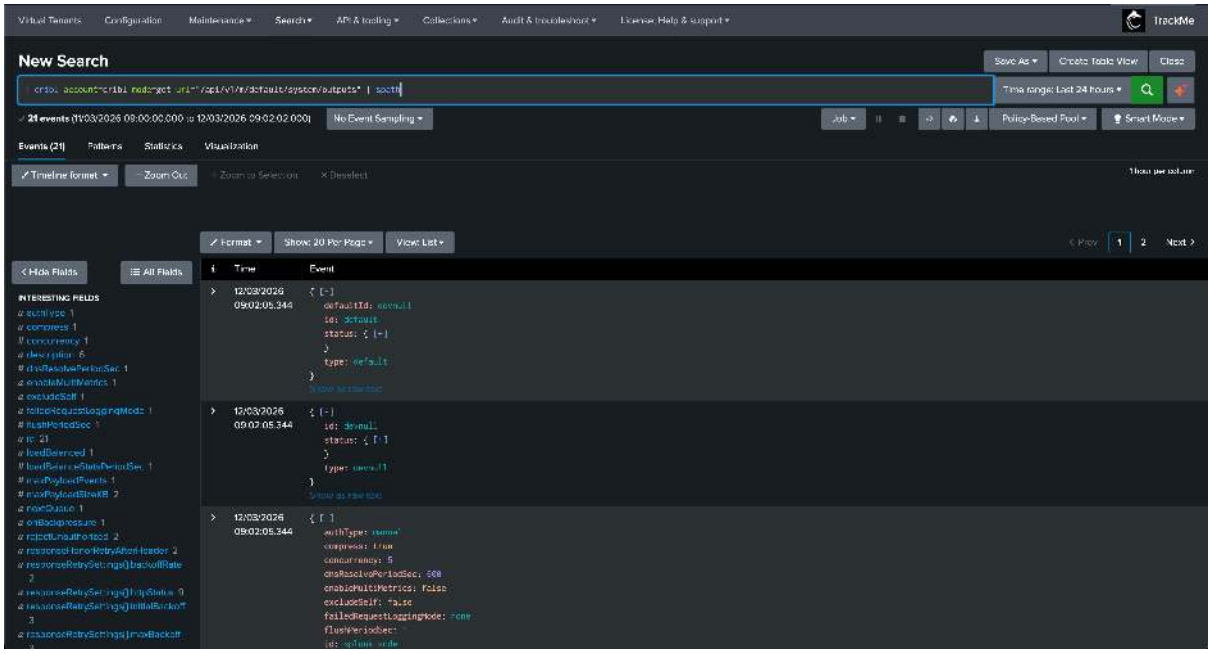
The **TA-trackme-cribl** Splunk Add-on provides seamless API integration between Splunk and Cribl, enabling you to interact with Cribl's API directly from Splunk using native SPL commands. This integration supports both on-premise Cribl deployments and Cribl Cloud environments, and covers both **Cribl Stream** (LogStream) and **Cribl Edge** products.

inputs/outputs details & status:

```
| cribl account=cribl mode=get url="/api/v1/m/default/system/inputs" | spath
```

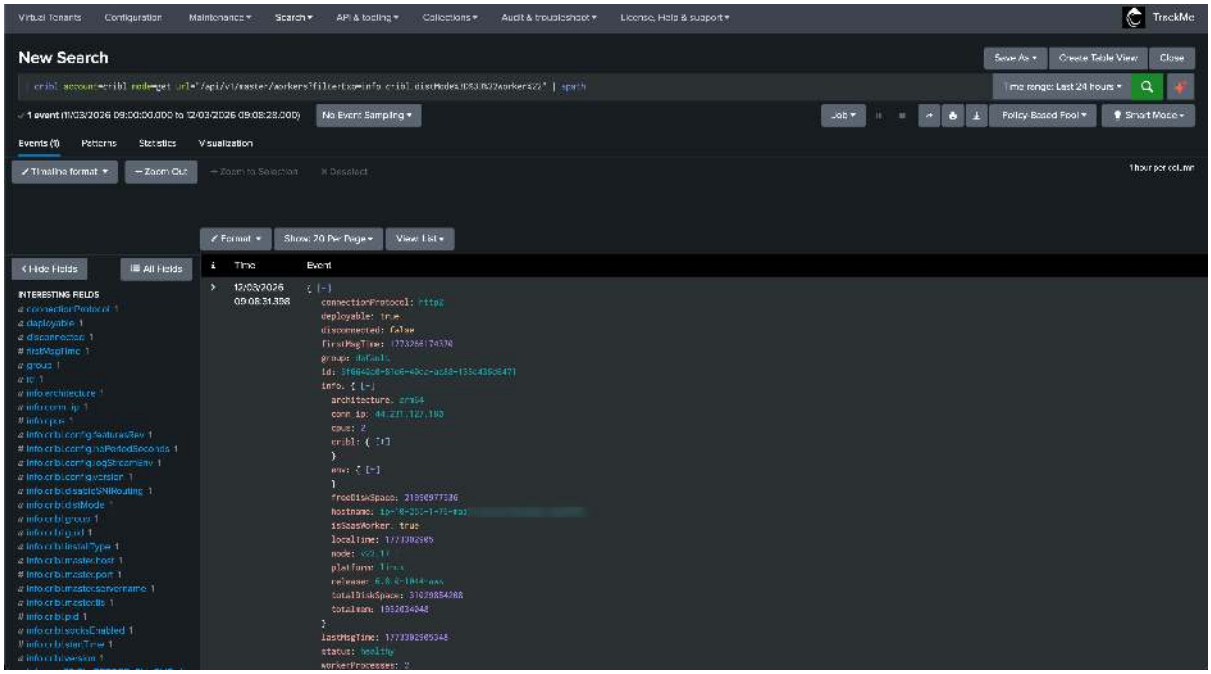
```
| cribl account=cribl mode=get url="/api/v1/m/default/system/outputs" | spath
```





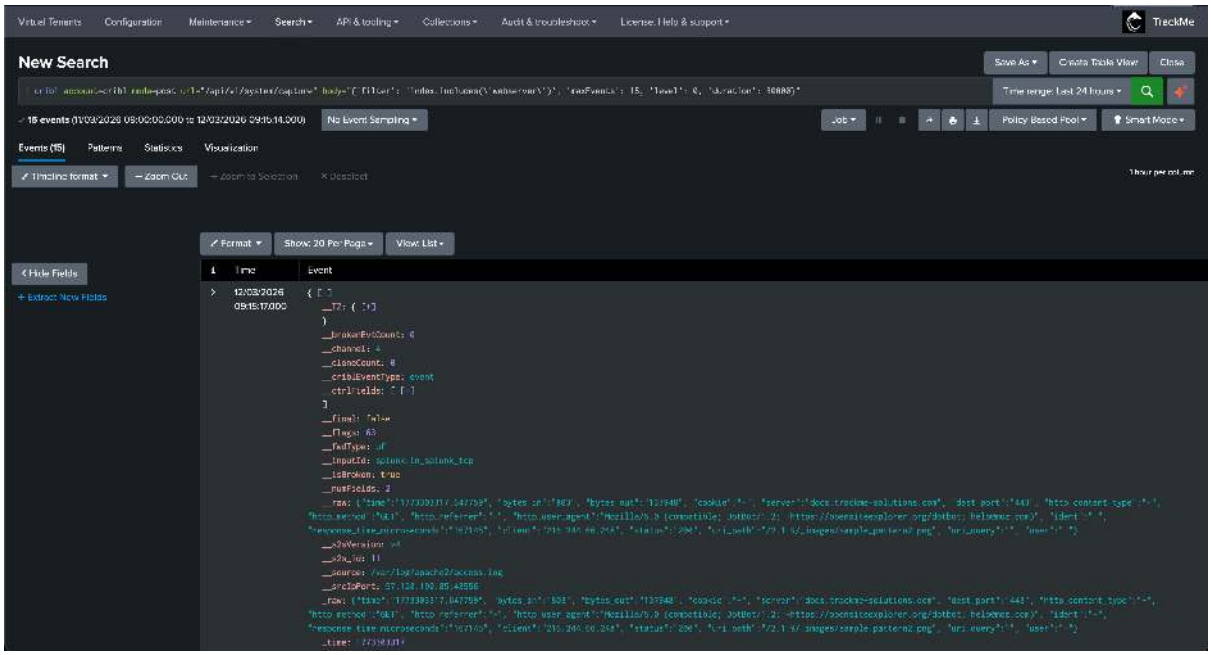
workers informations & status:

```
| cribl account=cribl mode=get url="/
↪api/v1/master/workers?filterExp=info.cribl.distMode%3D%3D%22worker%22" | spath
```



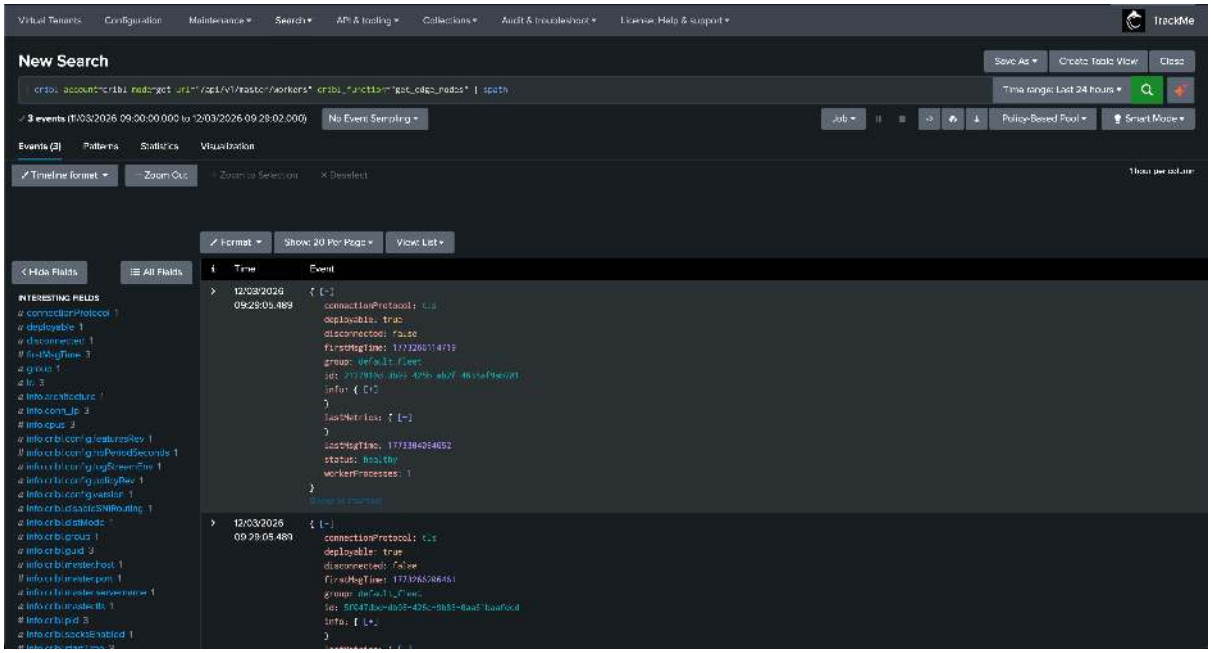
events capture by Splunk index:

```
| cribl account=cribl mode=post url="/api/v1/system/capture" body="{'filter':
↪'index.includes(\'webserver\')', 'maxEvents': 15, 'level': 0, 'duration': 30000}"
```



Cribl Edge fleet nodes status:

```
| cribl account=cribl mode=get url="/api/v1/master/workers" cribl_function=
↪"get_edge_nodes" | spath
```



Key Features

- **Cribl Stream & Edge Support**: Full support for both Cribl Stream (LogStream) and Cribl Edge API interactions from a single Add-on
- **Multi-Environment Support**: Configure and manage connections to both on-premise Cribl deployments and Cribl Cloud
- **Native SPL Integration**: Access Cribl API through the built-in `cribl` command in Splunk
- **Real-Time Metrics**: Retrieve and chart real-time metrics from Cribl for complex operations like pipeline monitoring
- **Edge Fleet Management**: Built-in functions for Cribl Edge — list Edge fleets, retrieve Edge nodes status, and get Edge fleet configurations
- **Role-Based Access Control**: Implement least-privilege access with granular role permissions
- **Predefined Functions**: Built-in functions for common operations like retrieving pipeline metrics, worker information, Edge fleet/node status, and system statistics
- **SSL Certificate Support**: Configure custom SSL certificates for secure connections
- **Connectivity Testing**: Automated connectivity testing during account creation and editing

Requirements

Application Dependencies
: No dependencies on other Splunk applications required.

Network Requirements
: The Splunk Search Head must have network connectivity to the Cribl API endpoints (typically HTTPS over port 443).

Authentication
: Supports both on-premise (username/password) and Cloud (client_id/client_secret) authentication methods.

Installation

Installation Target
Install on Search Head(s), standalone or Search Head Cluster.

Download Page
TrackMe Solutions Downloads

Splunk Cloud
Install the application as a private application on the Cloud Search heads, Appinspect vetting will be performed automatically.

Configuration

Account Configuration

Once deployed, access the Configuration UI and create a Cribl API account. The same account works for both Cribl Stream and Cribl Edge API calls — no separate account is needed.

When creating a Cribl account, you'll need to configure the following parameters:

Parameter	Description
Name	Enter a unique name for this account (default: `cribl`)
cribl_deployment_type	The type of Cribl deployment: Cloud or on-premise
cribl_cloud_organization_id	Organization ID for Cribl Cloud deployments
cribl_cloud_workspace	Workspace name for Cribl Cloud deployments (default: `main`)
cribl_onprem_leader_url	Leader URL for on-premise deployments (format: `https://<hostname>:<port>`)
cribl_client_id	Username for on-premise or client_id for Cloud
cribl_client_secret	Password for on-premise or client_secret for Cloud
rbac_roles	Comma-separated list of Splunk roles a user must be a true member of to use this account
cribl_ssl_verify	Enable/disable SSL verification (mandatory for Cloud, can be disabled for on-premise testing)
cribl_ssl_certificate_path	Path to PEM file for custom or self-signed certificates

When creating or updating the account, the application automatically tests connectivity and authentication against the Cribl API. If the connection fails, the UI will refuse the operation and return the exception.

Access Control

Capabilities & Built-in Roles

- Capability: `criblapi`
- Built-in role: `cribl_api`

Role-Based Access Control
Configure access on a per-account basis using the `rbac_roles` parameter. This is a true membership — the user must be a direct member of one of these roles (not inheriting). Default roles include: admin, sc_admin, trackme_user, trackme_power, trackme_admin.

Usage

General Notes

- When working with metrics, the Splunk time range filter is automatically taken into account
- The Cribl API allows up to the last 3 hours of metrics with a granular 10-second time window

- Beyond 3 hours, Cribl API rolls metrics with a 10-minute span — the `cribl` command handles this automatically
- When working with configuration, use `spath` to extract fields from the JSON API response
- You can perform `POST` calls using `mode=post` and the `body` option
- Add `run_test=True` to any command to report the response time (`runtime_sec`) and status (success or failure), useful for connectivity validation and monitoring

Cribl Stream API

Testing Connectivity

Test the Cribl API connection:

```
| cribl account=cribl mode=get url="/
↪api/v1/master/groups?fields=workerCount&product=stream" run_test=True
```

Event Capture

Capture events by index:

```
| cribl account=cribl mode=post url="/api/v1/system/capture" body="{'filter':
↪'index.includes(\'webserver\')', 'maxEvents': 15, 'level': 0, 'duration': 30000}"
```

Capture events by input ID:

```
| cribl account=cribl mode=post url="/api/v1/system/capture" body="{'filter': '__
↪inputId==\'splunk_hec:in_splunk_hec\'', 'maxEvents': 15, 'level': 0, 'duration':␣
↪30000}"
```

Worker Information

Get worker count and information:

```
| cribl account=cribl mode=get url="/
↪api/v1/master/groups?fields=workerCount&product=stream" | spath
```

Get extended worker stats:

```
| cribl account=cribl mode=get url="/
↪api/v1/master/workers?filterExp=info.cribl.distMode%3D%3D%22worker%22" | spath
```

Get worker count summary:

```
| cribl account=cribl mode=get url="/
↪api/v1/master/summary/workers?filterExp=info.cribl.distMode%3D%3D%22worker%22" |␣
↪spath
```

System Configuration

Get groups:

```
| cribl account=cribl mode=get url="/api/v1/master/groups" | spath
```

Get groups (Cribl Stream only):

```
| cribl account=cribl mode=get url="/api/v1/master/groups?product=stream" | spath
```

Get all groups configuration in a single request (Cribl Stream only):

```
| cribl account=cribl mode=post url="/api/v1/master/groups?product=stream"␣
↪cribl_function="get_groups_conf" | spath
```

Get inputs:

```
| cribl account=cribl mode=get url="/api/v1/m/default/system/inputs" | spath
```

Get outputs:

```
| cribl account=cribl mode=get url="/api/v1/m/default/system/outputs" | spath
```

Get routes:

```
| cribl account=cribl mode=get url="/api/v1/m/default/routes" | spath
```

Get pipelines:

```
| cribl account=cribl mode=get url="/api/v1/m/default/pipelines" | spath
```

Get packs:

```
| cribl account=cribl mode=get url="/api/v1/m/default/packs" | spath
```

Metrics and Monitoring

Get system metrics:

```
| cribl account=cribl mode=post url="/api/v1/system/metrics/query" cribl_function=
↪"get_global_metrics"
```

Get pipeline metrics:

```
| cribl account=cribl mode=post url="/api/v1/system/metrics/query" cribl_function=
↪"get_pipelines_metrics"
```

Get destination metrics:

```
| cribl account=cribl mode=post url="/api/v1/system/metrics/query" cribl_function=
↪"get_destinations_metrics"
```

Get routes metrics:

```
| cribl account=cribl mode=post url="/api/v1/system/metrics/query" cribl_function=
↪"get_routes_metrics"
```

System Information

Get system messages:

```
| cribl account=cribl mode=get url="/api/v1/system/messages" | spath
```

Get banners:

```
| cribl account=cribl mode=get url="/api/v1/system/banners" | spath
```

Get recent actions:

```
| cribl account=cribl mode=get url="/api/v1/ui/recentActions" | spath
```

Cribl Edge API

The Cribl Edge API uses the same authentication and account configuration as Cribl Stream. No separate account is needed — the same Cribl account works for both Stream and Edge API calls.

The following built-in functions and API calls allow full interaction with Cribl Edge fleets and nodes.

Get Edge Fleets

List all Edge fleets:

```
| cribl account=cribl mode=get url="/api/v1/master/groups?product=edge"␣
↪cribl_function="get_edge_fleets" | spath
```

Get Edge Nodes Status

This is the main built-in function for retrieving the status of all Edge nodes across all fleets. Each Edge node is returned as an individual Splunk event, making it easy to search, filter, chart, and alert on in Splunk.

Each event contains the full node information including: hostname, GUID, connection status, fleet assignment, last heartbeat, Edge version, config version, sources status, destinations status, and more.

```
| cribl account=cribl mode=get url="/api/v1/master/workers" cribl_function=
↪"get_edge_nodes" | spath
```

Extract key fields for a table view:

```
| cribl account=cribl mode=get url="/api/v1/master/workers" cribl_function=
↪"get_edge_nodes" | spath | table id, info.cribl.hostname, info.cribl.guid, info.
↪cribl.version, info.cribl.group, info.cribl.os, workerProcesses{}.status
```

Filter Edge nodes for a specific fleet:

```
| cribl account=cribl mode=get url="/api/v1/master/workers" cribl_function=
↪"get_edge_nodes" | spath | search info.cribl.group="default_fleet"
```

Get Edge Fleet Configurations

Retrieve all Edge fleet configurations in a single request:

```
| cribl account=cribl mode=get url="/api/v1/master/groups?product=edge"␣
↪cribl_function="get_edge_fleets_conf" | spath
```

Edge Fleet-Specific Configuration

Get inputs for a specific fleet:

```
| cribl account=cribl mode=get url="/api/v1/m/default_fleet/system/inputs" | spath
```

Get outputs for a specific fleet:

```
| cribl account=cribl mode=get url="/api/v1/m/default_fleet/system/outputs" | spath
```

Get routes for a specific fleet:

```
| cribl account=cribl mode=get url="/api/v1/m/default_fleet/routes" | spath
```

Get pipelines for a specific fleet:

```
| cribl account=cribl mode=get url="/api/v1/m/default_fleet/pipelines" | spath
```

Get All Workers and Edge Nodes

Retrieve all workers (Stream and Edge) in a single call:

```
| cribl account=cribl mode=get url="/api/v1/master/workers" | spath
```

Testing Edge Connectivity

Test Edge API connectivity:

```
| cribl account=cribl mode=get url="/api/v1/master/groups?product=edge"␣
↪run_test=True
```

Troubleshooting

Logs

Internal REST API Logs
: Available at: `index=_internal sourcetype="cribl:rest_api"`

Custom Command Logs
: Available at: `index=_internal sourcetype="cribl:custom_commands:cribl"`

Connectivity Testing

Test connectivity before account creation:

```
| cribl mode=post url="/services/cribl/v1/test_cribl_connectivity" body="{
↪'cribl_deployment_type': 'cloud', 'cribl_client_id': '<redacted_client_id>',
↪'cribl_client_secret': '<redacted_client_secret>'}"
```

Important Notes

- **Time Range Support**: The integration automatically respects Splunk time range filters when working with metrics
- **API Limitations**: Cribl API provides up to 3 hours of metrics with 10-second granularity, beyond which metrics are rolled up to 10-minute spans
- **JSON Parsing**: Use `spath` command to extract fields from JSON API responses
- **Performance Testing**: Add `run_test=True` to any command to get response time and status information
- **Stream & Edge Shared Authentication**: The same Cribl account works for both Stream and Edge API calls — no separate configuration needed

License and Support

License
: This application is licensed under the TrackMe EULA. See TrackMe License for details.

Terms & Conditions
: Available at TrackMe Terms & Conditions

Support

Support information available at TrackMe Support

CHAPTER

THIRTEEN

VARIOUS

13.1 Third-party components credits

13.1.1 Python Backend Dependencies

annotated-types

- Description: Reusable constraint types to use with typing.Annotated
- License: MIT
- Version: 0.8.0

```
The MIT License (MIT)

Copyright (c) 2022 the contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

anthropic

- Description: The official Python library for the anthropic API
- License: Apache-2.0
- Version: 2.10.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
```

(continues on next page)

(continued from previous page)

```
1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."
```

(continues on next page)

(continued from previous page)

```
      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

anyio

- Description: High-level concurrency and networking framework on top of asyncio or Trio
- License: MIT
- Version: 4.16.0

```
The MIT License (MIT)

Copyright (c) 2018 Alex Grönholm

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

argon2-cffi

- Description: Argon2 for Python
- License: MIT
- Version: 25.1.0

```
The MIT License (MIT)

Copyright (c) 2015 Hynek Schlawack and the argon2-cffi contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
```

(continues on next page)

(continued from previous page)

```
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

argon2-cffi-bindings

- Description: Low-level CFFI bindings for Argon2
- License: MIT
- Homepage: https://github.com/hynek/argon2-cffi-bindings
- Version: 21.2.0

```
The MIT License (MIT)

Copyright (c) 2021 Hynek Schlawack

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

attrs

- Description: Classes Without Boilerplate
- License: MIT
- Version: 26.1.0

```
The MIT License (MIT)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2015 Hynek Schlawack and the attrs contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

certifi

- Description: Python package for providing Mozilla's CA Bundle.
- License: MPL-2.0
- Homepage: https://github.com/certifi/python-certifi
- Version: 2026.10.23

```
This package contains a modified version of ca-bundle.crt:

ca-bundle.crt -- Bundle of CA Root Certificates

This is a bundle of X.509 certificates of public Certificate Authorities
(CA). These were automatically extracted from Mozilla's root certificates
file (certdata.txt).  This file can be found in the mozilla source tree:
https://hg.mozilla.org/mozilla-central/file/tip/security/nss/lib/ckfw/builtins/
↪certdata.txt
It contains the certificates in PEM format and therefore
can be directly used with curl / libcurl / php_curl, or with
an Apache+mod_ssl webserver for SSL client authentication.
Just configure this file as the SSLCACertificateFile.#

***** BEGIN LICENSE BLOCK *****
This Source Code Form is subject to the terms of the Mozilla Public License,
v. 2.0. If a copy of the MPL was not distributed with this file, You can obtain
one at http://mozilla.org/MPL/2.0/.

***** END LICENSE BLOCK *****
@(#) $RCSfile: certdata.txt,v $ $Revision: 1.80 $ $Date: 2011/11/03 15:11:58 $
```

cffi

- Description: Foreign Function Interface for Python calling C code.
- License: MIT
- Version: 2.0.0

```
Except when otherwise stated (look for LICENSE files in directories or
information at the beginning of each file) all software and
documentation is licensed as follows:

    MIT No Attribution

    Permission is hereby granted, free of charge, to any person
    obtaining a copy of this software and associated documentation
    files (the "Software"), to deal in the Software without
    restriction, including without limitation the rights to use,
    copy, modify, merge, publish, distribute, sublicense, and/or
    sell copies of the Software, and to permit persons to whom the
    Software is furnished to do so.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
    OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
    THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
    FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
    DEALINGS IN THE SOFTWARE.
```

charset-normalizer

- Description: The Real First Universal Charset Detector. Open, modern and actively maintained alternative to Chardet.
- License: MIT
- Version: 3.5.2

```
MIT License

Copyright (c) 2025 TAHRI Ahmed R.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

click

- Description: Composable command line interface toolkit
- License: BSD-3-Clause
- Version: 8.5.1

```
Copyright 2014 Pallets

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1.  Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

2.  Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

3.  Neither the name of the copyright holder nor the names of its
    contributors may be used to endorse or promote products derived from
    this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

croniter

- Description: croniter provides iteration for datetime object with cron like format
- License: MIT
- Version: 7.0.0

```
Copyright (C) 2010-2012 Matsumoto Taichi

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
```

(continues on next page)

(continued from previous page)

```
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

cryptography

- Description: cryptography is a package which provides cryptographic recipes and primitives to Python developers.
- License: Apache-2.0 OR BSD-3-Clause
- Version: 50.0.1

```
This software is made available under the terms of *either* of the licenses
found in LICENSE.APACHE or LICENSE.BSD. Contributions to cryptography are made
under the terms of *both* these licenses.
```

cssselect2

- Description: CSS selectors for Python ElementTree
- License: BSD-3-Clause
- Version: 0.10.1

```
BSD 3-Clause License

Copyright (c) 2012-2018, Simon Sapin and contributors (see AUTHORS).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

defusedxml

- Description: XML bomb protection for Python stdlib modules
- License: PSFL
- Homepage: https://github.com/tiran/defusedxml
- Version: 0.7.1

```
PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF
hereby grants Licensee a nonexclusive, royalty-free, world-wide
license to reproduce, analyze, test, perform and/or display publicly,
prepare derivative works, distribute, and otherwise use Python
alone or in any derivative version, provided, however, that PSF's
License Agreement and PSF's notice of copyright, i.e., "Copyright (c)
2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008 Python Software Foundation;
All Rights Reserved" are retained in Python alone or in any derivative
version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative work available to others as provided herein, then
Licensee hereby agrees to include in any such work a brief summary of
the changes made to Python.

4. PSF is making Python available to Licensee on an "AS IS"
basis.  PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR
IMPLIED.  BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND
DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS
FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON WILL NOT
INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS
A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON,
OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material
breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any
relationship of agency, partnership, or joint venture between PSF and
Licensee.  This License Agreement does not grant permission to use PSF
trademarks or trade name in a trademark sense to endorse or promote
products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python, Licensee
agrees to be bound by the terms and conditions of this License
Agreement.
```

distro

- Description: Distro - an OS platform information API
- License: Apache License, Version 2.0
- Homepage: https://github.com/python-distro/distro
- Version: 1.9.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
```

(continues on next page)

(continued from previous page)

```
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

docstring_parser

- Description: Parse Python docstrings in reST, Google and Numpydoc format
- License: MIT
- Version: 0.18.0

```
The MIT License (MIT)

Copyright (c) 2018 Marcin Kurczewski

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

filetype

- Description: Infer file type and MIME type of any file/buffer. No external dependencies.
- License: MIT
- Homepage: https://github.com/h2non/filetype.py
- Version: 1.2.0

```
The MIT License (MIT)

Copyright (c) 2016 Tomás Aparicio

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

google-auth

- Description: Google Authentication Library
- License: Apache 2.0
- Homepage: https://github.com/googleapis/google-cloud-python/tree/main/packages/google-auth
- Version: 2.62.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
```

(continues on next page)

(continued from previous page)

```
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

google-genai

- Description: GenAI Python SDK
- License: Apache-2.0
- Version: 3.6.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
```

(continues on next page)

(continued from previous page)

```
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

googleapis-common-protos

- Description: Common protobufs used in Google APIs
- License: Apache-2.0
- Version: 1.76.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.
```

(continues on next page)

(continued from previous page)

```
      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

grpcio

- Description: HTTP/2-based RPC framework
- License: Apache-2.0
- Version: 1.85.3

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
```

(continues on next page)

(continued from previous page)

```
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

h11

- Description: A pure-Python, bring-your-own-I/O implementation of HTTP/1.1
- License: MIT
- Homepage: https://github.com/python-hyper/h11
- Version: 0.16.0

```
The MIT License (MIT)

Copyright (c) 2016 Nathaniel J. Smith <njs@pobox.com> and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

httpcore

- Description: A minimal low-level HTTP client.
- License: BSD-3-Clause
- Version: 1.0.9

```
Copyright © 2020, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

httpcore2

- Description: A minimal low-level HTTP client.
- License: BSD-3-Clause
- Version: 2.16.2

```
Copyright © 2026 to present Pydantic Services Inc. and individual contributors.
Copyright © 2020, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
```

(continues on next page)

(continued from previous page)

```
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

httpx

- Description: The next generation HTTP client.
- License: BSD-3-Clause
- Version: 0.28.1

```
Copyright © 2019, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,␣
↪are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list␣
↪of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this␣
↪list of conditions and the following disclaimer in the documentation and/or␣
↪other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors may␣
↪be used to endorse or promote products derived from this software without␣
↪specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND␣
↪ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED␣
↪WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR␣
↪ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES␣
↪(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;␣
↪LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON␣
↪ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT␣
↪(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS␣
↪SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

httpx-sse

- Description: Consume Server-Sent Event (SSE) messages with HTTPX.
- License: MIT
- Version: 0.4.3

```
MIT License

Copyright (c) 2022 Florimond Manca

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

httpx2

- Description: The next generation HTTP client.
- License: BSD-3-Clause
- Version: 2.16.2

```
Copyright © 2026 to present Pydantic Services Inc. and individual contributors.
Copyright © 2019, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,␣
↪are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list␣
↪of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice, this␣
↪list of conditions and the following disclaimer in the documentation and/or␣
↪other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its contributors may␣
↪be used to endorse or promote products derived from this software without␣
↪specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND␣
↪ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED␣
↪WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR␣
↪ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES␣
↪(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;␣
↪LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON␣
↪ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT␣
↪(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS␣
↪SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

idna

- Description: Internationalized Domain Names in Applications (IDNA)
- License: BSD-3-Clause
- Version: 3.20

```
BSD 3-Clause License

Copyright (c) 2013-2026, Kim Davies and contributors.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED
TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

importlib_metadata

- Description: Read metadata from Python packages
- License: Apache-2.0
- Version: 10.2.0

```
Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and␣
↪distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright␣
```

(continues on next page)

(continued from previous page)

```
↪owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities␣
↪that control, are controlled by, or are under common control with that entity.␣
↪For the purposes of this definition, "control" means (i) the power, direct or␣
↪indirect, to cause the direction or management of such entity, whether by␣
↪contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the␣
↪outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions␣
↪granted by this License.

"Source" form shall mean the preferred form for making modifications, including␣
↪but not limited to software source code, documentation source, and configuration␣
↪files.

"Object" form shall mean any form resulting from mechanical transformation or␣
↪translation of a Source form, including but not limited to compiled object code,␣
↪generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made␣
↪available under the License, as indicated by a copyright notice that is included␣
↪in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is␣
↪based on (or derived from) the Work and for which the editorial revisions,␣
↪annotations, elaborations, or other modifications represent, as a whole, an␣
↪original work of authorship. For the purposes of this License, Derivative Works␣
↪shall not include works that remain separable from, or merely link (or bind by␣
↪name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version␣
↪of the Work and any modifications or additions to that Work or Derivative Works␣
↪thereof, that is intentionally submitted to Licensor for inclusion in the Work␣
↪by the copyright owner or by an individual or Legal Entity authorized to submit␣
↪on behalf of the copyright owner. For the purposes of this definition, "submitted
↪" means any form of electronic, verbal, or written communication sent to the␣
↪Licensor or its representatives, including but not limited to communication on␣
↪electronic mailing lists, source code control systems, and issue tracking␣
↪systems that are managed by, or on behalf of, the Licensor for the purpose of␣
↪discussing and improving the Work, but excluding communication that is␣
↪conspicuously marked or otherwise designated in writing by the copyright owner␣
↪as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of␣
↪whom a Contribution has been received by Licensor and subsequently incorporated␣
↪within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License,
↪ each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-
↪charge, royalty-free, irrevocable copyright license to reproduce, prepare␣
↪Derivative Works of, publicly display, publicly perform, sublicense, and␣
↪distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License,␣
↪each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-
```

(continues on next page)

(continued from previous page)

```
↪charge, royalty-free, irrevocable (except as stated in this section) patent␣
↪license to make, have made, use, offer to sell, sell, import, and otherwise␣
↪transfer the Work, where such license applies only to those patent claims␣
↪licensable by such Contributor that are necessarily infringed by their␣
↪Contribution(s) alone or by
   [...truncated, see full license at the project homepage...]
```

jiter

- Description: Fast iterable JSON parser.
- License: MIT
- Homepage: https://github.com/pydantic/jiter/
- Version: 0.13.0

```
The MIT License (MIT)

Copyright (c) 2022 to present Samuel Colvin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

jsonpatch

- Description: Apply JSON-Patches (RFC 6902)
- License: BSD-3-Clause
- Homepage: https://github.com/stefankoegl/python-json-patch
- Version: 1.47

```
Copyright (c) 2011 Stefan Kögl <stefan@skoegl.net>

Redistribution and use in source and binary forms, with or without modification,␣
↪are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this␣
↪list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this␣
↪list of conditions and the following disclaimer in the documentation and/or␣
```

(continues on next page)

(continued from previous page)

```
↪other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may␣
↪be used to endorse or promote products derived from this software without␣
↪specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND␣
↪ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED␣
↪WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR␣
↪ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES␣
↪(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;␣
↪LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON␣
↪ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT␣
↪(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS␣
↪SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

jsonpointer

- Description: Identify specific nodes in a JSON document (RFC 6901)
- License: BSD-3-Clause
- Homepage: https://github.com/stefankoegl/python-json-pointer
- Version: 3.12

```
Copyright (c) 2011 Stefan Kögl <stefan@skoegl.net>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS'' AND ANY EXPRESS OR
IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES
OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF
THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

jsonschema

- Description: An implementation of JSON Schema validation for Python
- License: MIT
- Version: 4.26.0

```
Copyright (c) 2013 Julian Berman

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

jsonschema-specifications

- Description: The JSON Schema meta-schemas and vocabularies, exposed as a Registry
- License: MIT
- Version: 2025.9.1

```
Copyright (c) 2022 Julian Berman

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

langchain

- Description: Building applications with LLMs through composability
- License: MIT
- Version: 1.5.2

```
MIT License

Copyright (c) LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langchain-anthropic

- Description: Integration package connecting Claude (Anthropic) APIs and LangChain
- License: MIT
- Version: 1.7.11

```
MIT License

Copyright (c) 2023 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langchain-core

- Description: Building applications with LLMs through composability
- License: MIT
- Version: 1.8.3

```
MIT License

Copyright (c) LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langchain-google-genai

- Description: An integration package connecting Google's genai package and LangChain
- License: MIT
- Version: 4.5.1

```
MIT License

Copyright (c) 2023 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langchain-openai

- Description: An integration package connecting OpenAI and LangChain
- License: MIT
- Version: 1.8.3

```
MIT License

Copyright (c) 2023 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langchain-protocol

- Description: Python bindings for the LangChain agent streaming protocol
- License: MIT
- Version: 0.0.21

```
MIT License

Copyright (c) 2026 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langgraph

- Description: Building stateful, multi-actor applications with LLMs
- License: MIT
- Version: 1.2.23

```
MIT License

Copyright (c) 2024 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langgraph-checkpoint

- Description: Library with base interfaces for LangGraph checkpoint savers.
- License: MIT
- Version: 4.3.2

```
MIT License

Copyright (c) 2024 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langgraph-prebuilt

- Description: Library with high-level APIs for creating and executing LangGraph agents and tools.
- License: MIT
- Version: 1.1.3

```
MIT License

Copyright (c) 2024 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langgraph-sdk

- Description: SDK for interacting with LangGraph API
- License: MIT
- Version: 0.4.14

```
MIT License

Copyright (c) 2024 LangChain, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

langsmith

- Description: Client library to connect to the LangSmith Observability and Evaluation Platform.
- License: MIT
- Version: 0.15.0

```
Copyright 2026 langChain

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

lazy-importer

- Description: A context manager for portable, opt-in lazy imports
- License: Apache-2.0
- Version: 0.1.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.
```

(continues on next page)

(continued from previous page)

```
    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

licensing

- Description:
- License: MIT
- Homepage: https://devolens.com

- Version: 0.55

```
MIT License

Copyright (c) Cryptolens AB and Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

lxml

- Description: Powerful and Pythonic XML processing library combining libxml2/libxslt with the ElementTree API.
- License: BSD-3-Clause
- Homepage: https://lxml.de/
- Version: 6.1.3

```
BSD 3-Clause License

Copyright (c) 2004 Infrae. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

  1. Redistributions of source code must retain the above copyright
     notice, this list of conditions and the following disclaimer.

  2. Redistributions in binary form must reproduce the above copyright
     notice, this list of conditions and the following disclaimer in
     the documentation and/or other materials provided with the
     distribution.

  3. Neither the name of Infrae nor the names of its contributors may
     be used to endorse or promote products derived from this software
     without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL INFRAE OR
```

(continues on next page)

(continued from previous page)

```
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

mcp

- Description: Model Context Protocol SDK
- License: MIT
- Version: 1.30.0

```
MIT License

Copyright (c) 2024 Anthropic, PBC

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

minio

- Description: MinIO Python SDK for Amazon S3 Compatible Cloud Storage
- License: Apache-2.0
- Homepage: https://github.com/minio/minio-py
- Version: 7.2.20

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
```

(continues on next page)

(continued from previous page)

```
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

numpy

- Description: Fundamental package for array computing in Python
- License: BSD-3-Clause
- Version: 2.2.3

```
Copyright (c) 2005-2024, NumPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
       notice, this list of conditions and the following disclaimer.

    * Redistributions in binary form must reproduce the above
       copyright notice, this list of conditions and the following
       disclaimer in the documentation and/or other materials provided
       with the distribution.

    * Neither the name of the NumPy Developers nor the names of any
       contributors may be used to endorse or promote products derived
       from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

----

The NumPy repository and source distributions bundle several libraries that are
compatibly licensed.  We list these here.

Name: lapack-lite
Files: numpy/linalg/lapack_lite/*
License: BSD-3-Clause
```

(continues on next page)

(continued from previous page)

```
  For details, see numpy/linalg/lapack_lite/LICENSE.txt

Name: dragon4
Files: numpy/_core/src/multiarray/dragon4.c
License: MIT
  For license text, see numpy/_core/src/multiarray/dragon4.c

Name: libdivide
Files: numpy/_core/include/numpy/libdivide/*
License: Zlib
  For license text, see numpy/_core/include/numpy/libdivide/LICENSE.txt


Note that the following files are vendored in the repository and sdist but not
installed in built numpy packages:

Name: Meson
Files: vendored-meson/meson/*
License: Apache 2.0
  For license text, see vendored-meson/meson/COPYING

Name: spin
Files: .spin/cmds.py
License: BSD-3
  For license text, see .spin/LICENSE

Name: tempita
Files: numpy/_build_utils/tempita/*
License: MIT
  For details, see numpy/_build_utils/tempita/LICENCE.txt

----

This binary distribution of NumPy also bundles the following software:


Name: OpenBLAS
Files: numpy.libs/libscipy_openblas*.so
Description: bundled as a dynamically linked library
Availability: https://github.com/OpenMathLib/OpenBLAS/
License: BSD-3-Clause
  Copyright (c) 2011-2014, The OpenBLAS Project
  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions are
  met:

     1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

     2. Redistributions in binary form must reproduce the above copyright
        notice, this list of conditions and the following disclaimer in
        the documentation and/or other materials provided with the
        distribution.
     3. Neither the name of the OpenBLAS project nor the names of
```

(continues on next page)

(continued from previous page)

```
       its contributors may be used to endorse or promote products
       derived from this software without specific prior written
       permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL

   [...truncated, see full license at the project homepage...]
```

openai

- Description: The official Python library for the openai API
- License: Apache-2.0
- Version: 3.55.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
```

(continues on next page)

(continued from previous page)

```
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-api

- Description: OpenTelemetry Python API
- License: Apache-2.0
- Version: 1.45.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
    "Licensor" shall mean the copyright owner or entity authorized by
    the copyright owner that is granting the License.

    "Legal Entity" shall mean the union of the acting entity and all
    other entities that control, are controlled by, or are under common
    control with that entity. For the purposes of this definition,
    "control" means (i) the power, direct or indirect, to cause the
    direction or management of such entity, whether by contract or
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
```

(continues on next page)

(continued from previous page)

```
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-exporter-otlp-common

- Description: OpenTelemetry OTLP HTTP export utilities
- License: Apache-2.0
- Version: 0.66b2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).
```

(continues on next page)

(continued from previous page)

```
      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-exporter-otlp-proto-common

- Description: OpenTelemetry Protobuf encoding
- License: Apache-2.0
- Version: 1.45.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
```

(continues on next page)

(continued from previous page)

```
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
```

(continues on next page)

(continued from previous page)

```
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-exporter-otlp-proto-grpc

- Description: OpenTelemetry Collector Protobuf over gRPC Exporter
- License: Apache-2.0
- Version: 1.45.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
```

(continues on next page)

(continued from previous page)

```
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-proto

- Description: OpenTelemetry Python Proto
- License: Apache-2.0
- Version: 1.45.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.
```

(continues on next page)

(continued from previous page)

```
      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
```

(continues on next page)

(continued from previous page)

```
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-sdk

- Description: OpenTelemetry Python SDK
- License: Apache-2.0
- Version: 1.45.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
```

(continues on next page)

(continued from previous page)

```
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

opentelemetry-semantic-conventions

- Description: OpenTelemetry Semantic Conventions
- License: Apache-2.0
- Version: 0.66b2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
```

(continues on next page)

(continued from previous page)

```
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.
```

(continues on next page)

(continued from previous page)

```
   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

orjson

- Description: Fast, correct Python JSON library supporting dataclasses, datetimes, and numpy
- License: MPL-2.0 AND (Apache-2.0 OR MIT)
- Version: 3.11.6

ormsgpack

- Description: Fast, correct Python msgpack library supporting dataclasses, datetimes, and numpy
- License: Apache-2.0 OR MIT
- Version: 1.12.2

packaging

- Description: Core utilities for Python packages
- License: BSD
- Version: 24.0

```
This software is made available under the terms of *either* of the licenses
found in LICENSE.APACHE or LICENSE.BSD. Contributions to this software is made
under the terms of *both* these licenses.
```

pillow

- Description: Python Imaging Library (fork)
- License: MIT-CMU
- Version: 12.2.0

```
The Python Imaging Library (PIL) is

    Copyright © 1997-2011 by Secret Labs AB
    Copyright © 1995-2011 by Fredrik Lundh and contributors

Pillow is the friendly PIL fork. It is

    Copyright © 2010 by Jeffrey 'Alex' Clark and contributors

Like PIL, Pillow is licensed under the open source MIT-CMU License:

By obtaining, using, and/or copying this software and/or its associated
documentation, you agree that you have read, understood, and will comply
with the following terms and conditions:

Permission to use, copy, modify and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appears in all copies, and that
both that copyright notice and this permission notice appear in supporting
documentation, and that the name of Secret Labs AB or the author not be
```

(continues on next page)

(continued from previous page)

```
used in advertising or publicity pertaining to distribution of the software
without specific, written prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS.
IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR BE LIABLE FOR ANY SPECIAL,
INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE
OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.


----

AOM

Copyright (c) 2016, Alliance for Open Media. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in
   the documentation and/or other materials provided with the
   distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.


----

BROTLI

Copyright (c) 2009, 2010, 2013-2016 by the Brotli Authors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.  IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.


----

BZIP2


--------------------------------------------------------------------------

This program, "bzip2", the
   [...truncated, see full license at the project homepage...]
```

protobuf

- Description:
- License: 3-Clause BSD License
- Homepage: https://developers.google.com/protocol-buffers/
- Version: 7.36.2

```
Copyright 2008 Google Inc.  All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

    * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
    * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

(continues on next page)

(continued from previous page)

```
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Code generated by the Protocol Buffer compiler is owned by the owner
of the input file used when generating it.  This code is not
standalone and requires a support library to be linked with it.  This
support library is itself covered by the above license.
```

pyasn1

- Description: Pure-Python implementation of ASN.1 types and DER/BER/CER codecs (X.208)
- License: BSD-2-Clause
- Version: 0.6.5

pyasn1_modules

- Description: A collection of ASN.1-based protocols modules
- License: BSD
- Homepage: https://github.com/pyasn1/pyasn1-modules
- Version: 0.4.2

```
Copyright (c) 2005-2020, Ilya Etingof <etingof@gmail.com>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

  * Redistributions of source code must retain the above copyright notice,
    this list of conditions and the following disclaimer.

  * Redistributions in binary form must reproduce the above copyright notice,
    this list of conditions and the following disclaimer in the documentation
    and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS
INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN
CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
```

pycparser

- Description: C parser in Python
- License: BSD-3-Clause
- Version: 3.12

```
pycparser -- A C parser in Python

Copyright (c) 2008-2027, Eli Bendersky
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of the copyright holder nor the names of its contributors may
  be used to endorse or promote products derived from this software without
  specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

pycryptodome

- Description: Cryptographic library for Python
- License: BSD, Public Domain
- Homepage: https://www.pycryptodome.org
- Version: 3.23.0

pydantic

- Description: Data validation using Python type hints
- License: MIT
- Version: 2.13.5

```
The MIT License (MIT)

Copyright (c) 2017 to present Pydantic Services Inc. and individual contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

pydantic_core

- Description: Core functionality for Pydantic validation and serialization
- License: MIT
- Homepage: https://github.com/pydantic/pydantic
- Version: 2.46.5

```
The MIT License (MIT)

Copyright (c) 2022 Samuel Colvin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

pydantic-settings

- Description: Settings management using Pydantic
- License: MIT
- Version: 2.16.3

```
The MIT License (MIT)

Copyright (c) 2022 Samuel Colvin and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
```

(continues on next page)

(continued from previous page)

```
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

pygal

- Description: A Python svg graph plotting library
- License: GNU LGPL v3+
- Homepage: https://www.pygal.org/
- Version: 3.1.3

```
GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.


  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.
```

(continues on next page)

(continued from previous page)

```
  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.

  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
```

(continues on next page)

(continued from previous page)

```
   document.

   c) For a Combined Work that displays copyright
   [...truncated, see full license at the project homepage...]
```

PyJWT

- Description: JSON Web Token implementation in Python
- License: MIT
- Version: 2.15.1

```
The MIT License (MIT)

Copyright (c) 2015-2022 José Padilla

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

PySocks

- Description: A Python SOCKS client module. See https://github.com/Anorov/PySocks for more information.
- License: BSD
- Homepage: https://github.com/Anorov/PySocks
- Version: 1.7.1

```
Copyright 2006 Dan-Haim. All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:
1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.
3. Neither the name of Dan Haim nor the names of his contributors may be used
   to endorse or promote products derived from this software without specific
   prior written permission.
```

(continues on next page)

(continued from previous page)

```
THIS SOFTWARE IS PROVIDED BY DAN HAIM "AS IS" AND ANY EXPRESS OR IMPLIED
WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO
EVENT SHALL DAN HAIM OR HIS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA
OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT
OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMANGE.
```

python-dateutil

- Description: Extensions to the standard Python datetime module
- License: BSD-3-Clause
- Version: 4.0.0

```
Copyright 2017- Paul Ganssle <paul@ganssle.io>
Copyright 2017- dateutil contributors (see AUTHORS file)

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.

The above license applies to all contributions after 2017-12-01, as well as
all contributions that have been re-licensed (see AUTHORS file for the list of
contributors who have re-licensed their code).
--------------------------------------------------------------------------------
dateutil - Extensions to the standard Python datetime module.

Copyright (c) 2003-2011 - Gustavo Niemeyer <gustavo@niemeyer.net>
Copyright (c) 2012-2014 - Tomi Pieviläinen <tomi.pievilainen@iki.fi>
Copyright (c) 2014-2016 - Yaron de Leeuw <me@jarondl.net>
Copyright (c) 2015-     - Paul Ganssle <paul@ganssle.io>
Copyright (c) 2015-     - dateutil contributors (see AUTHORS file)

All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of the copyright holder nor the names of its
      contributors may be used to endorse or promote products derived from
```

(continues on next page)

(continued from previous page)

```
      this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The above BSD License Applies to all code, even that also covered by Apache 2.0.
```

python-dotenv

- Description: Read key-value pairs from a .env file and set them as environment variables
- License: BSD-3-Clause
- Version: 1.2.4

```
Copyright (c) 2014, Saurabh Kumar (python-dotenv), 2013, Ted Tieken (django-dotenv-
↪rw), 2013, Jacob Kaplan-Moss (django-dotenv)

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

- Neither the name of django-dotenv nor the names of its contributors
  may be used to endorse or promote products derived from this software
  without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

python-multipart

- Description: A streaming multipart parser for Python
- License: Apache-2.0
- Version: 0.0.34

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
```

(continues on next page)

(continued from previous page)

```
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

PyYAML

- Description: YAML parser and emitter for Python
- License: MIT
- Homepage: https://pyyaml.org/
- Version: 6.0.3

```
Copyright (c) 2017-2021 Ingy döt Net
Copyright (c) 2006-2016 Kirill Simonov

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

referencing

- Description: JSON Referencing + Python
- License: MIT
- Version: 0.37.0

```
Copyright (c) 2022 Julian Berman

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

regex

- Description: Alternative regular expression module, to replace re.
- License: Apache-2.0 AND CNRI-Python
- Version: 2026.1.15

```
This work was derived from the 're' module of CPython 2.6 and CPython 3.1,
copyright (c) 1998-2001 by Secret Labs AB and licensed under CNRI's Python 1.6
license.

All additions and alterations are licensed under the Apache 2.0 License.


                                 Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
```

(continues on next page)

(continued from previous page)

```
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare
   [...truncated, see full license at the project homepage...]
```

reportlab

- Description: The Reportlab Toolkit
- License: BSD license (see license.txt for details), Copyright (c) 2000-2025, ReportLab Inc.
- Homepage: https://www.reportlab.com/
- Version: 4.5.1

```
#############################################################################
↪##
#
#    Copyright (c) 2000-2024, ReportLab Inc.
#    All rights reserved.
#
#    Redistribution and use in source and binary forms, with or without␣
↪modification,
#    are permitted provided that the following conditions are met:
#
#            *       Redistributions of source code must retain the above␣
↪copyright notice,
#                    this list of conditions and the following disclaimer.
#            *       Redistributions in binary form must reproduce the above␣
↪copyright notice,
#                    this list of conditions and the following disclaimer in the␣
↪documentation
#                    and/or other materials provided with the distribution.
#            *       Neither the name of the company nor the names of its␣
↪contributors may be
#                    used to endorse or promote products derived from this␣
↪software without
#                    specific prior written permission.
#
#    THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"␣
↪AND
#    ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
#    WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED.
#    IN NO EVENT SHALL THE OFFICERS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,␣
↪INDIRECT,
#    INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT␣
↪LIMITED
#    TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR␣
↪PROFITS;
#    OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,␣
↪WHETHER
#    IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)␣
↪ARISING
#    IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE␣
↪POSSIBILITY OF
#    SUCH DAMAGE.
#
#############################################################################
↪##
```

requests

- Description: Python HTTP for Humans.
- License: Apache-2.0
- Version: 2.34.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
```

(continues on next page)

(continued from previous page)

```
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

requests-toolbelt

- Description: A utility belt for advanced users of python-requests
- License: Apache 2.0
- Homepage: https://toolbelt.readthedocs.io/
- Version: 1.0.0

```
Copyright 2014 Ian Cordasco, Cory Benfield

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       https://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.
```

rl-renderPM

- Description: Bitmap Render Acclerator for ReportLab
- License: BSD license (see LICENSE.txt for details), Copyright (c) 2000-2022, ReportLab Inc.
- Homepage: http://www.reportlab.com/
- Version: 4.0.3

rpds-py

- Description: Python bindings to Rust's persistent data structures (rpds)
- License: MIT
- Version: 0.27.1

```
Copyright (c) 2023 Julian Berman

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

scipy

- Description: Fundamental algorithms for scientific computing in Python
- License: BSD-3-Clause
- Version: 1.15.2

```
Copyright (c) 2001-2002 Enthought, Inc. 2003-2024, SciPy Developers.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above
   copyright notice, this list of conditions and the following
   disclaimer in the documentation and/or other materials provided
   with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived
   from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
```

(continues on next page)

(continued from previous page)

```
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

----

This binary distribution of SciPy can also bundle the following software
(depending on the build):


Name: OpenBLAS
Files: scipy.libs/libscipy_openblas*.so
Description: bundled as a dynamically linked library
Availability: https://github.com/OpenMathLib/OpenBLAS/
License: BSD-3-Clause-Attribution
  Copyright (c) 2011-2014, The OpenBLAS Project
  All rights reserved.

  Redistribution and use in source and binary forms, with or without
  modification, are permitted provided that the following conditions are
  met:

     1. Redistributions of source code must retain the above copyright
        notice, this list of conditions and the following disclaimer.

     2. Redistributions in binary form must reproduce the above copyright
        notice, this list of conditions and the following disclaimer in
        the documentation and/or other materials provided with the
        distribution.
     3. Neither the name of the OpenBLAS project nor the names of
        its contributors may be used to endorse or promote products
        derived from this software without specific prior written
        permission.

  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
  ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE
  LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
  USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


Name: LAPACK
Files: scipy.libs/libscipy_openblas*.so
Description: bundled in OpenBLAS
Availability: https://github.com/OpenMathLib/OpenBLAS/
License: BSD-3-Clause-Attribution
  Copyright (c) 1992-2013 The University of Tennessee and The University
                          of Tennessee Research Foundation.  All rights
```

(continues on next page)

(continued from previous page)

```
                          reserved.
  Copyright (c) 2000-2013 The University of California Berkeley. All
                          rights reserved.
  Copyright (c) 2006-2013 The University of Colorado Denver.  All rights

   [...truncated, see full license at the project homepage...]
```

sniffio

- Description: Sniff out which async library your code is running under
- License: MIT OR Apache-2.0
- Version: 1.3.1

```
This software is made available under the terms of *either* of the
licenses found in LICENSE.APACHE2 or LICENSE.MIT. Contributions to are
made under the terms of *both* these licenses.
```

solnlib

- Description: The Splunk Software Development Kit for Splunk Solutions
- License: Apache-2.0
- Version: 8.2.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
```

(continues on next page)

(continued from previous page)

```
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

sortedcontainers

- Description: Sorted Containers – Sorted List, Sorted Dict, Sorted Set
- License: Apache 2.0
- Homepage: http://www.grantjenks.com/docs/sortedcontainers/
- Version: 2.4.0

```
Copyright 2014-2019 Grant Jenks

Licensed under the Apache License, Version 2.0 (the "License");
```

(continues on next page)

(continued from previous page)

```
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

splunk-sdk

- Description: Splunk Software Development Kit for Python
- License: Apache-2.0
- Version: 3.0.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).
```

(continues on next page)

(continued from previous page)

```
      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

splunktaucclib

- Description:
- License: Apache-2.0
- Version: 8.2.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
```

(continues on next page)

(continued from previous page)

```
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

sse-starlette

- Description: SSE plugin for Starlette
- License: BSD-3-Clause
- Version: 3.7.0

```
Copyright © 2020, [sysid](https://sysid.github.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

starlette

- Description: The little ASGI library that shines.
- License: BSD-3-Clause
- Version: 1.10.0

```
Copyright © 2018, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
```

(continues on next page)

(continued from previous page)

```
* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

svglib

- Description: A pure-Python library for reading and converting SVG
- License: LGPL-3.0-or-later
- Version: 2.4.0

```
GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.


  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.
```

(continues on next page)

(continued from previous page)

```
  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.

  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
```

(continues on next page)

(continued from previous page)

```
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright
   [...truncated, see full license at the project homepage...]
```

tenacity

- Description: Retry code until it succeeds
- License: Apache 2.0
- Homepage: https://github.com/jd/tenacity
- Version: 9.1.4

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.
```

(continues on next page)

(continued from previous page)

```
      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

tiktoken

- Description: tiktoken is a fast BPE tokeniser for use with OpenAI's models
- License: MIT
- Version: 0.11.0

```
MIT License

Copyright (c) 2022 OpenAI, Shantanu Jain

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
```

(continues on next page)

(continued from previous page)

```
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

tinycss2

- Description: A tiny CSS parser
- License: BSD-3-Clause
- Version: 1.5.1

```
BSD 3-Clause License

Copyright (c) 2013-2020, Simon Sapin and contributors.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

truststore

- Description: Verify certificates using native system trust stores
- License: MIT
- Version: 0.10.5

```
The MIT License (MIT)

Copyright (c) 2022 Seth Michael Larson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

typing_extensions

- Description: Backported and Experimental Type Hints for Python 3.9+
- License: PSF-2.0
- Version: 4.16.0

```
A. HISTORY OF THE SOFTWARE
==========================

Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see https://www.cwi.nl) in the Netherlands
as a successor of a language called ABC.  Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see https://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team.  In October of the same
year, the PythonLabs team moved to Digital Creations, which became
Zope Corporation.  In 2001, the Python Software Foundation (PSF, see
https://www.python.org/psf/) was formed, a non-profit organization
created specifically to own Python-related Intellectual Property.
Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see https://opensource.org for
```

(continues on next page)

(continued from previous page)

```
the Open Source Definition).  Historically, most, but not all, Python
releases have also been GPL-compatible; the table below summarizes
the various releases.

    Release         Derived     Year        Owner       GPL-
                    from                                compatible? (1)

    0.9.0 thru 1.2              1991-1995   CWI         yes
    1.3 thru 1.5.2  1.2         1995-1999   CNRI        yes
    1.6             1.5.2       2000        CNRI        no
    2.0             1.6         2000        BeOpen.com  no
    1.6.1           1.6         2001        CNRI        yes (2)
    2.1             2.0+1.6.1   2001        PSF         no
    2.0.1           2.0+1.6.1   2001        PSF         yes
    2.1.1           2.1+2.0.1   2001        PSF         yes
    2.1.2           2.1.1       2002        PSF         yes
    2.1.3           2.1.2       2002        PSF         yes
    2.2 and above   2.1.1       2001-now    PSF         yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under
    the GPL.  All Python licenses, unlike the GPL, let you distribute
    a modified version without making your changes open source.  The
    GPL-compatible licenses make it possible to combine Python with
    other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible,
    because its license has a choice of law clause.  According to
    CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1
    is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's
direction to make these releases possible.


B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON
===============================================================

Python software and documentation are licensed under the
Python Software Foundation License Version 2.

Starting with Python 3.8.6, examples, recipes, and other code in
the documentation are dual licensed under the PSF License Version 2
and the Zero-Clause BSD license.

Some software incorporated into Python is under different licenses.
The licenses are listed with code falling under that license.


PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
```

(continues on next page)

(continued from previous page)

```
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,
analyze, test, perform and/or display publicly, prepare derivative works,
distribute, and otherwise use Python alone or in any derivative version,
provided, however, that PSF's License Agreement and PSF's notice of copyright,
i.e., "Copyrig
   [...truncated, see full license at the project homepage...]
```

typing-inspection

- Description: Runtime typing introspection tools
- License: MIT
- Version: 0.4.5

```
MIT License

Copyright (c) Pydantic Services Inc. 2025 to present

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

tzdata

- Description: Provider of IANA time zone data
- License: Apache-2.0
- Homepage: https://github.com/python/tzdata
- Version: 2026.6

```
Apache Software License 2.0

Copyright (c) 2020, Paul Ganssle (Google)
Copyright (c) 2026, Stan Ulbrych

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
```

(continues on next page)

(continued from previous page)

```
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

urllib3

- Description: HTTP library with thread-safe connection pooling, file post, and more.
- License: MIT
- Version: 2.8.0

```
MIT License

Copyright (c) 2008-2020 Andrey Petrov and contributors.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

uuid_utils

- Description: Fast, drop-in replacement for Python's uuid module, powered by Rust.
- License: BSD-3-Clause
- Version: 0.14.1

```
Copyright © 2023, Amin Alaee.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
```

(continues on next page)

(continued from previous page)

```
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

uvicorn

- Description: The lightning-fast ASGI server.
- License: BSD-3-Clause
- Version: 0.54.0

```
Copyright © 2017-present, [Encode OSS Ltd](https://www.encode.io/).
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

webencodings

- Description: Character encoding aliases for legacy web content
- License: BSD-3-Clause
- Version: 0.6.1

```
BSD 3-Clause License

Copyright (c) 2012, Simon Sapin
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

websockets

- Description: An implementation of the WebSocket Protocol (RFC 6455 & 7692)
- License: BSD-3-Clause
- Version: 17.4

```
Copyright (c) Aymeric Augustin and contributors

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of the copyright holder nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.
```

(continues on next page)

(continued from previous page)

```
THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

xxhash

- Description: Python binding for xxHash
- License: BSD
- Homepage: https://github.com/ifduyue/python-xxhash
- Version: 3.6.0

```
Copyright (c) 2014-2024, Yue Du
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

zipp

- Description: Backport of pathlib-compatible object wrapper for zip files
- License: MIT
- Version: 4.1.2

```
MIT License

Copyright (c) 2026 <copyright holders>

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and
```

(continues on next page)

(continued from previous page)

```
associated documentation files (the "Software"), to deal in the Software without␣
↪restriction, including
without limitation the rights to use, copy, modify, merge, publish, distribute,␣
↪sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is furnished to␣
↪do so, subject to the
following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial
portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND␣
↪NONINFRINGEMENT. IN NO
EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR␣
↪OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION␣
↪WITH THE SOFTWARE OR THE
USE OR OTHER DEALINGS IN THE SOFTWARE.
```

zstandard

- Description: Zstandard bindings for Python
- License: BSD-3-Clause
- Version: 0.25.0

```
Copyright (c) 2016, Gregory Szorc
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
this list of conditions and the following disclaimer in the documentation
and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors
may be used to endorse or promote products derived from this software without
specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

13.1.2 JavaScript Frontend Dependencies

@babel/code-frame

- Description: Generate errors that contain a code frame that point to source locations.
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-code-frame
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/generator

- Description: Turns an AST into code.
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-generator
- Version: 7.29.13

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
```

(continues on next page)

(continued from previous page)

```
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-annotate-as-pure

- Description: Helper function to annotate paths and nodes with #__PURE__ comment
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-helper-annotate-as-pure
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-globals

- Description: A collection of JavaScript globals for Babel internal usage
- License: MIT
- Homepage: https://github.com/babel/babel
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
```

(continues on next page)

(continued from previous page)

```
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-module-imports

- Description: Babel helper functions for inserting module loads
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-helper-module-imports
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-plugin-utils

- Description: General utilities for plugins to use
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-helper-plugin-utils
- Version: 7.29.7

```
MIT License
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-string-parser

- Description: A utility package to parse strings
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-helper-string-parser
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/helper-validator-identifier

- Description: Validate identifier/keywords name
- License: MIT
- Homepage: https://github.com/babel/babel
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/parser

- Description: A JavaScript parser
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-parser
- Version: 7.29.8

```
Copyright (C) 2012-2014 by various contributors (see AUTHORS)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

@babel/plugin-syntax-jsx

- Description: Allow parsing of jsx
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-plugin-syntax-jsx
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/runtime

- Description: babel's modular runtime helpers
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-runtime
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
```

(continues on next page)

(continued from previous page)

```
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/template

- Description: Generate an AST from a string template.
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-template
- Version: 7.29.7

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/traverse

- Description: The Babel Traverse module maintains the overall tree state, and is responsible for replacing, removing, and adding nodes
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-traverse
- Version: 7.29.8

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
```

(continues on next page)

(continued from previous page)

```
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@babel/types

- Description: Babel Types is a Lodash-esque utility library for AST nodes
- License: MIT
- Homepage: https://babel.dev/docs/en/next/babel-types
- Version: 7.29.8

```
MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@dnd-kit/accessibility

- Description: A generic toolkit to help with accessibility
- License: MIT
- Homepage: https://github.com/clauderic/dnd-kit
- Version: 3.1.1

```
MIT License

Copyright (c) 2021, Claudéric Demers

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@dnd-kit/core

- Description: dnd kit – a lightweight React library for building performant and accessible drag and drop experiences
- License: MIT
- Homepage: https://github.com/clauderic/dnd-kit
- Version: 6.3.1

```
MIT License

Copyright (c) 2021, Claudéric Demers

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@dnd-kit/modifiers

- Description: Translate modifier presets for use with *@dnd-kit* packages.
- License: MIT
- Homepage: https://github.com/clauderic/dnd-kit
- Version: 7.0.0

```
MIT License

Copyright (c) 2021, Claudéric Demers

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@dnd-kit/sortable

- Description: Official sortable preset and sensors for dnd kit
- License: MIT
- Homepage: https://github.com/clauderic/dnd-kit
- Version: 10.0.0

```
MIT License

Copyright (c) 2021, Claudéric Demers

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@dnd-kit/utilities

- Description: Internal utilities to bee shared between *@dnd-kit* packages
- License: MIT
- Homepage: https://github.com/clauderic/dnd-kit
- Version: 3.2.2

```
MIT License

Copyright (c) 2021, Claudéric Demers

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@emotion/is-prop-valid

- Description: A function to check whether a prop is valid for HTML and SVG elements
- License: MIT
- Homepage: https://github.com/emotion-js/emotion/tree/main/packages/is-prop-valid
- Version: 1.4.0

```
MIT License

Copyright (c) Emotion team and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@emotion/memoize

- Description: emotion's memoize utility
- License: MIT
- Homepage: https://github.com/emotion-js/emotion/tree/main/packages/memoize
- Version: 0.9.0

```
MIT License

Copyright (c) Emotion team and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@emotion/stylis

- Description: A custom build of Stylis
- License: MIT
- Homepage: https://github.com/emotion-js/emotion/tree/master/packages/stylis
- Version: 0.8.5

```
MIT License

Copyright (c) Emotion team and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
```

(continues on next page)

(continued from previous page)

```
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@emotion/unitless

- Description: An object of css properties that don't accept values with units
- License: MIT
- Homepage: https://github.com/emotion-js/emotion/tree/master/packages/unitless
- Version: 0.7.5

```
MIT License

Copyright (c) Emotion team and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@hnw/date-tibetan

- Description: Tibetan Calendar
- License: MIT
- Homepage: https://github.com/hnw/date-tibetan
- Version: 1.0.2

```
MIT License

Copyright (c) 2025 Yoshio HANAWA
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@jridgewell/gen-mapping

- Description: Generate source maps
- License: MIT
- Homepage: https://github.com/jridgewell/sourcemaps/tree/main/packages/gen-mapping
- Version: 0.3.13

```
Copyright 2024 Justin Ridgewell <justin@ridgewell.name>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@jridgewell/resolve-uri

- Description: Resolve a URI relative to an optional base URI
- License: MIT
- Homepage: https://github.com/jridgewell/resolve-uri
- Version: 3.1.2

```
Copyright 2019 Justin Ridgewell <jridgewell@google.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@jridgewell/sourcemap-codec

- Description: Encode/decode sourcemap mappings
- License: MIT
- Homepage: https://github.com/jridgewell/sourcemaps/tree/main/packages/sourcemap-codec
- Version: 1.6.0

```
Copyright 2024 Justin Ridgewell <justin@ridgewell.name>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@jridgewell/trace-mapping

- Description: Trace the original position through a source map
- License: MIT
- Homepage: https://github.com/jridgewell/sourcemaps/tree/main/packages/trace-mapping
- Version: 0.3.31

```
Copyright 2024 Justin Ridgewell <justin@ridgewell.name>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@juggle/resize-observer

- Description: Polyfills the ResizeObserver API and supports box size options from the latest spec
- License: Apache-2.0
- Homepage: https://juggle.studio/resize-observer/
- Version: 3.4.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.
```

(continues on next page)

(continued from previous page)

```
    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

@loki/create-async-callback

- Description: Loki async story management helper
- License: MIT
- Homepage: https://github.com/oblador/loki/tree/master/packages/create-async-callback
- Version: 0.35.1

```
The MIT License (MIT)

Copyright (c) 2017 Joel Arvidsson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@loki/integration-core

- Description: Loki storybook integration core lib
- License: MIT
- Homepage: https://github.com/oblador/loki/tree/master/packages/integration-core
- Version: 0.35.0

```
The MIT License (MIT)

Copyright (c) 2017 Joel Arvidsson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@loki/integration-react-native

- Description: Loki react-native integration
- License: MIT
- Homepage: https://github.com/oblador/loki/tree/master/packages/integration-react-native
- Version: 0.35.0

```
The MIT License (MIT)

Copyright (c) 2017 Joel Arvidsson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@mapbox/jsonlint-lines-primitives

- Description: Validate JSON
- License: MIT
- Homepage: https://github.com/mapbox/jsonlint
- Version: 2.0.3

```
MIT License

Copyright (c) 2012 Zachary Carter

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@mapbox/point-geometry

- Description: a point geometry with transforms
- License: ISC
- Homepage: https://github.com/mapbox/point-geometry
- Version: 1.1.0

```
Copyright (c) 2024, Mapbox

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

@mapbox/tiny-sdf

- Description: Browser-side SDF font generator
- License: BSD-2-Clause
- Homepage: https://github.com/mapbox/tiny-sdf#readme
- Version: 2.2.0

```
BSD-2-Clause
Copyright (c) 2016-2024 Mapbox, Inc.

Redistribution and use in source and binary forms, with or without modification,␣
↪are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this␣
↪list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this␣
↪list of conditions and the following disclaimer in the documentation and/or␣
↪other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND␣
↪ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED␣
↪WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR␣
↪ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES␣
↪(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;␣
↪LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON␣
↪ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT␣
↪(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS␣
↪SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

@mapbox/unitbezier

- Description: unit bezier curve interpolation
- License: BSD-2-Clause
- Homepage: https://github.com/mapbox/unitbezier
- Version: 1.0.0

```
BSD-2-Clause

Copyright (C) 2008 Apple Inc. All Rights Reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY APPLE INC. ``AS IS'' AND ANY
EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL APPLE INC. OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Ported from Webkit
http://svn.webkit.org/repository/webkit/trunk/Source/WebCore/platform/graphics/UnitBezier.h
```

@mapbox/vector-tile

- Description: Parses vector tiles
- License: BSD-3-Clause
- Homepage: https://github.com/mapbox/vector-tile-js
- Version: 3.0.0

```
Copyright (c) 2024, Mapbox

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
```

(continues on next page)

(continued from previous page)

```
    * Neither the name of Mapbox nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

@maplibre/geojson-vt

- Description: Slice GeoJSON data into vector tiles efficiently
- License: ISC
- Homepage: https://github.com/maplibre/geojson-vt
- Version: 6.1.1

```
The MIT License (MIT)

Copyright (c) 2025 MapLibre contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

----------------------------------------------------------------------------

ISC License

Copyright (c) 2024, Mapbox

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

@maplibre/maplibre-gl-style-spec

- Description: a specification for maplibre styles
- License: ISC
- Homepage: https://maplibre.org/maplibre-style-spec/
- Version: 26.4.2

```
Copyright (c) 2020, MapLibre contributors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of MapLibre GL JS nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


--------------------------------------------------------------------------------

Contains code from mapbox-gl-js v1.13 and earlier

Version v1.13 of mapbox-gl-js and earlier are licensed under a BSD-3-Clause license

Copyright (c) 2020, Mapbox
Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
```

(continues on next page)

(continued from previous page)

```
  this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of Mapbox GL JS nor the names of its contributors
  may be used to endorse or promote products derived from this software
  without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


--------------------------------------------------------------------------------

Contains code from glfx.js

Copyright (C) 2011 by Evan Wallace

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT
   [...truncated, see full license at the project homepage...]
```

@maplibre/mlt

- License: (MIT OR Apache-2.0)
- Homepage: https://github.com/maplibre/maplibre-tile-spec/#readme
- Version: 1.2.1

```
===== MIT License =====
MIT License

Copyright (c) 2024 MapLibre contributors

Permission is hereby granted, free of charge, to any
person obtaining a copy of this software and associated
documentation files (the "Software"), to deal in the
```

(continues on next page)

(continued from previous page)

```
Software without restriction, including without
limitation the rights to use, copy, modify, merge,
publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software
is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice
shall be included in all copies or substantial portions
of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF
ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED
TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT
SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR
IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.

===== Apache License 2.0 =====
                             Apache License
                       Version 2.0, January 2004
                    http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

   "License" shall mean the terms and conditions for use, reproduction,
   and distribution as defined by Sections 1 through 9 of this document.

   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.
```

(continues on next page)

(continued from previous page)

```
   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control sys
   [...truncated, see full license at the project homepage...]
```

@maplibre/vt-pbf

- Description: Serialize mapbox vector tiles to binary protobufs in javascript.
- License: MIT
- Homepage: https://github.com/maplibre/vt-pbf
- Version: 4.3.2

```
The MIT License (MIT)

Copyright (c) 2025 MapLibre contributors

Copyright (c) 2015 Anand Thakker

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------

Contains geojson_wrapper.js from https://github.com/mapbox/mapbox-gl-js

Copyright (c) 2014, Mapbox

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of Mapbox GL JS nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

@popperjs/core

- Description: Tooltip and Popover Positioning Engine
- License: MIT
- Homepage: github:popperjs/popper-core
- Version: 2.11.8

```
The MIT License (MIT)

Copyright (c) 2019 Federico Zivolo

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
```

(continues on next page)

(continued from previous page)

```
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

@react-spring/animated

- Description: Animated component props for React
- License: MIT
- Homepage: https://github.com/pmndrs/react-spring#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@react-spring/core

- License: MIT
- Homepage: https://github.com/pmndrs/react-spring#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@react-spring/rafz

- Description: react-spring's fork of rafz one frameloop to rule them all
- License: MIT
- Homepage: https://github.com/pmndrs/react-spring/tree/main/packages/rafz#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@react-spring/shared

- Description: Globals and shared modules
- License: MIT
- Homepage: https://github.com/pmndrs/react-spring#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

(continues on next page)

(continued from previous page)

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@react-spring/types

- Description: Internal package with TypeScript stuff
- License: MIT
- Homepage: https://github.com/pmndrs/react-spring#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@react-spring/web

- License: MIT
- Homepage: https://github.com/pmndrs/react-spring#readme
- Version: 9.7.5

```
MIT License

Copyright (c) 2018-present Paul Henschel, react-spring, all contributors
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@reduxjs/toolkit

- Description: The official, opinionated, batteries-included toolset for efficient Redux development
- License: MIT
- Homepage: https://redux-toolkit.js.org
- Version: 2.12.0

```
MIT License

Copyright (c) 2018 Mark Erikson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@splunk/async-dynamic-options-evaluator

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

```

(continues on next page)

(continued from previous page)

```
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/audit-adaptive-delay

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-ai-advisor

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-ai-advisor-scheduled

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-ai-assistant

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-ai-routines

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-data-sampling

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-kvstore

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-sla

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-svc-usage

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-tenants-ops

- License: UNLICENSED
- Version: 0.0.1

@splunk/audit-trackers-perf

- License: UNLICENSED
- Version: 0.0.1

@splunk/backuprestore

- License: UNLICENSED
- Version: 0.0.1

@splunk/charting-bundle

- Description: @splunk/charting bundle
- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/configuration-guardian

- License: UNLICENSED
- Version: 0.0.1

@splunk/dashboard-context

- Description: React contexts and APIs for dashboard state, plugins, data sources, and interactions
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
```

(continues on next page)

(continued from previous page)

```
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-core

- Description: Core dashboard rendering components for layouts, visualizations, inputs, and action menus
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-definition

- Description: Dashboard definition models, schemas, and utilities for editing and validation
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-event-handlers

- Description: Built-in dashboard event handlers for tokens, time ranges, tabs, and custom URLs
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-icons

- Description: Custom icon components used across the dashboard authoring UI
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-inputs

- Description: Dashboard input components and schemas for text, select, multiselect, number, button, and time range controls
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

```

(continues on next page)

(continued from previous page)

```
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-layouts

- Description: Dashboard layout components and APIs for absolute, grid, and input layouts
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-presets

- Description: Preset mappings that wire dashboard visualizations, layouts, inputs, event handlers, and data sources
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-search

- Description: Search and data source orchestration for dashboard queries, refresh, and subscriptions
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-state

- Description: Redux state management, reducers, middleware, and providers for dashboards
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-telemetry

- Description: Telemetry APIs and collectors for dashboard usage, performance, and search metrics
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-types

- Description: TypeScript types for dashboard APIs, definitions, events, layouts, inputs, and state
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-ui

- Description: Shared React components for dashboard authoring, editing, and management experiences
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-utils

- Description: Shared utilities for dashboard exports, expressions, data handling, and helper logic
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
```

(continues on next page)

(continued from previous page)

```
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/dashboard-validation

- Description: Dashboard definition validation components, hooks, and worker-backed validators
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/datasource-utils

- Description: Utilities and base types for working with dashboard data sources and DataSet results
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/datasources

- Description: Splunk search and snapshot datasource implementations for dashboard queries
- License: Apache-2.0
- Version: 29.9.0

```
Copyright 2026 Cisco Systems, Inc.
```

(continues on next page)

(continued from previous page)

```
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/license

- License: UNLICENSED
- Version: 0.0.1

@splunk/logs-inspector

- License: UNLICENSED
- Version: 0.0.1

@splunk/maintenance-kdb

- License: UNLICENSED
- Version: 0.0.1

@splunk/maintenance-mode

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-ai-providers

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-ai-tenants

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-bank-holidays

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-brokers

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-cribl-connections

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-email-accounts

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-remote-accounts

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-s3-accounts

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-system-settings

- License: UNLICENSED
- Version: 0.0.1

@splunk/manage-webhook-accounts

- License: UNLICENSED
- Version: 0.0.1

@splunk/moment

- Description: Splunk timezone and formatting plugins for moment.js
- License: Apache-2.0
- Version: 0.7.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
```

(continues on next page)

(continued from previous page)

```
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

@splunk/react-events-viewer

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/react-field-summary

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/react-icons

- Description: Library of various React icons
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/react-icons
- Version: 5.17.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
    "Licensor" shall mean the copyright owner or entity authorized by
    the copyright owner that is granting the License.

    "Legal Entity" shall mean the union of the acting entity and all
    other entities that control, are controlled by, or are under common
    control with that entity. For the purposes of this definition,
    "control" means (i) the power, direct or indirect, to cause the
    direction or management of such entity, whether by contract or
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
```

(continues on next page)

(continued from previous page)

```
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/react-page

- Description: Load React components into the latest layout from Splunk Enterprise
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/react-page
- Version: 8.5.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).
```

(continues on next page)

(continued from previous page)

```
      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/react-search

- Description: Splunk search component in React
- License: Apache-2.0
- Version: 8.0.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

    "License" shall mean the terms and conditions for use, reproduction,
    and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
    "Licensor" shall mean the copyright owner or entity authorized by
    the copyright owner that is granting the License.

    "Legal Entity" shall mean the union of the acting entity and all
    other entities that control, are controlled by, or are under common
    control with that entity. For the purposes of this definition,
    "control" means (i) the power, direct or indirect, to cause the
    direction or management of such entity, whether by contract or
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
```

(continues on next page)

(continued from previous page)

```
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this se
   [...truncated, see full license at the project homepage...]
```

@splunk/react-time-range

- Description: A Splunk time range picker built with React.
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/react-time-range
- Version: 12.1.2

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
```

(continues on next page)

(continued from previous page)

```
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/react-ui

- Description: Library of React components that implement the Splunk design language
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/react-ui
- Version: 5.18.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
```

(continues on next page)

(continued from previous page)

```
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.
```

(continues on next page)

(continued from previous page)

```
   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/remote-accounts-overview

- License: UNLICENSED
- Version: 0.0.1

@splunk/rest-api-reference

- License: UNLICENSED
- Version: 0.0.1

@splunk/search-job

- Description: Create and access Splunk Enterprise search jobs
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/search-job
- Version: 3.1.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
```

(continues on next page)

(continued from previous page)

```
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby
[...truncated, see full license at the project homepage...]
```

@splunk/splunk-utils

- Description: Library of common Splunk Enterprise utilities
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/splunk-utils

- Version: 4.2.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
```

(continues on next page)

(continued from previous page)

```
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/support-diag

- License: UNLICENSED
- Version: 0.0.1

@splunk/tenant-home

- License: UNLICENSED
- Version: 0.0.1

@splunk/themes

- Description: Theme variables and mixins for the Splunk design language
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/themes
- Version: 1.13.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
```

(continues on next page)

(continued from previous page)

```
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
```

(continues on next page)

(continued from previous page)

```
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/time-range-utils

- License: Apache-2.0
- Version: 3.4.1

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

    "License" shall mean the terms and conditions for use, reproduction,
    and distribution as defined by Sections 1 through 9 of this document.

    "Licensor" shall mean the copyright owner or entity authorized by
    the copyright owner that is granting the License.

    "Legal Entity" shall mean the union of the acting entity and all
    other entities that control, are controlled by, or are under common
    control with that entity. For the purposes of this definition,
    "control" means (i) the power, direct or indirect, to cause the
    direction or management of such entity, whether by contract or
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
```

(continues on next page)

(continued from previous page)

```
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   copyright license to reproduce, prepare Derivative Works of,
   publicly display, publicly perform, sublicense, and distribute the
   Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
   worldwide, non-exclusive, no-charge, royalty-free, irrevocable
   (except as stated in this se
   [...truncated, see full license at the project homepage...]
```

@splunk/topology-studio

- License: UNLICENSED
- Version: 0.0.1

@splunk/trackme-ai-assistant

- License: UNLICENSED
- Version: 0.0.1

@splunk/trackme-ai-routines

- License: UNLICENSED
- Version: 0.0.1

@splunk/trackme-notifications

- License: UNLICENSED
- Version: 0.0.1

@splunk/trackme-preferences

- License: UNLICENSED
- Version: 0.0.1

@splunk/ui-utils

- Description: Library of common UI utilities
- License: Apache-2.0
- Homepage: https://splunkui.splunk.com/Packages/ui-utils
- Version: 1.15.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
```

(continues on next page)

(continued from previous page)

```
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunk/virtual-tenants

- License: UNLICENSED
- Version: 0.0.1

@splunk/visualization-canvas

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-color-palettes

- Description: Color palette for visualizations
- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-context

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-encoding

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

(continues on next page)

(continued from previous page)

```
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-icons

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-migrations

- Description: Migration scripts for visualization definitions
- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-schemas

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0
```

(continues on next page)

(continued from previous page)

```
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualization-themes

- Description: Dashboard themes for visualizations
- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualizations

- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunk/visualizations-shared

- Description: Visualizations Shared Files
- License: Apache-2.0
- Version: 28.9.1

```
Copyright 2026 Splunk Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```

@splunkdev/cloud-auth-client

- Description: Splunk Cloud Node Authentication Library for front-end web applications
- License: Apache-2.0
- Homepage: https://dev.splunk.com/scs/docs/overview/tools/tools_cloud-auth/
- Version: 3.5.3

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.
```

(continues on next page)

(continued from previous page)

```
      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunkdev/cloud-auth-common

- Description: Splunk Cloud Common Authentication Library for JavaScript
- License: Apache-2.0
- Homepage: https://dev.splunk.com/scs/docs/overview/tools/tools_cloud-auth/
- Version: 2.3.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION
```

(continues on next page)

(continued from previous page)

```
   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
```

(continues on next page)

(continued from previous page)

```
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@splunkdev/cloud-sdk

- Description: Splunk Cloud SDK for Javascript
- License: Apache-2.0
- Version: 15.0.0

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
      the copyright owner that is granting the License.

      "Legal Entity" shall mean the union of the acting entity and all
      other entities that control, are controlled by, or are under common
      control with that entity. For the purposes of this definition,
      "control" means (i) the power, direct or indirect, to cause the
      direction or management of such entity, whether by contract or
      otherwise, or (ii) ownership of fifty percent (50%) or more of the
      outstanding shares, or (iii) beneficial ownership of such entity.

      "You" (or "Your") shall mean an individual or Legal Entity
      exercising permissions granted by this License.

      "Source" form shall mean the preferred form for making modifications,
      including but not limited to software source code, documentation
      source, and configuration files.

      "Object" form shall mean any form resulting from mechanical
      transformation or translation of a Source form, including but
      not limited to compiled object code, generated documentation,
      and conversions to other media types.

      "Work" shall mean the work of authorship, whether in Source or
```

(continues on next page)

(continued from previous page)

```
      Object form, made available under the License, as indicated by a
      copyright notice that is included in or attached to the work
      (an example is provided in the Appendix below).

      "Derivative Works" shall mean any work, whether in Source or Object
      form, that is based on (or derived from) the Work and for which the
      editorial revisions, annotations, elaborations, or other modifications
      represent, as a whole, an original work of authorship. For the purposes
      of this License, Derivative Works shall not include works that remain
      separable from, or merely link (or bind by name) to the interfaces of,
      the Work and Derivative Works thereof.

      "Contribution" shall mean any work of authorship, including
      the original version of the Work and any modifications or additions
      to that Work or Derivative Works thereof, that is intentionally
      submitted to Licensor for inclusion in the Work by the copyright owner
      or by an individual or Legal Entity authorized to submit on behalf of
      the copyright owner. For the purposes of this definition, "submitted"
      means any form of electronic, verbal, or written communication sent
      to the Licensor or its representatives, including but not limited to
      communication on electronic mailing lists, source code control systems,
      and issue tracking systems that are managed by, or on behalf of, the
      Licensor for the purpose of discussing and improving the Work, but
      excluding communication that is conspicuously marked or otherwise
      designated in writing by the copyright owner as "Not a Contribution."

      "Contributor" shall mean Licensor and any individual or Legal Entity
      on behalf of whom a Contribution has been received by Licensor and
      subsequently incorporated within the Work.

   2. Grant of Copyright License. Subject to the terms and conditions of
      this License, each Contributor hereby grants to You a perpetual,
      worldwide, non-exclusive, no-charge, royalty-free, irrevocable
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

@standard-schema/spec

- Description: A family of specs for interoperable TypeScript
- License: MIT
- Homepage: https://standardschema.dev
- Version: 1.1.0

```
MIT License

Copyright (c) 2024 Colin McDonnell

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
```

(continues on next page)

(continued from previous page)

```
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@standard-schema/utils

- Description: The official runtime utils for Standard Schema
- License: MIT
- Homepage: https://github.com/standard-schema/standard-schema
- Version: 0.3.0

```
MIT License

Copyright (c) 2024 Fabian Hiller

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

@types/geojson

- Description: TypeScript definitions for geojson
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/geojson
- Version: 7946.0.16

```
MIT License
```

(continues on next page)

(continued from previous page)

```
    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/hast

- Description: TypeScript definitions for hast
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/hast
- Version: 2.3.10

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/json-schema

- Description: TypeScript definitions for json-schema
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/json-schema

- Version: 7.0.15

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/mdast

- Description: TypeScript definitions for mdast
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/mdast
- Version: 3.0.15

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/node

- Description: TypeScript definitions for node
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/node
- Version: 26.6.6

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/pako

- Description: TypeScript definitions for pako
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/pako
- Version: 2.0.4

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/prop-types

- Description: TypeScript definitions for prop-types
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/prop-types
- Version: 15.7.15

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/react

- Description: TypeScript definitions for react
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/react
- Version: 18.3.31

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/set-cookie-parser

- Description: TypeScript definitions for set-cookie-parser
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped
- Version: 0.0.6

```
MIT License

    Copyright (c) Microsoft Corporation. All rights reserved.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/unist

- Description: TypeScript definitions for unist
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/unist
- Version: 2.0.11

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
```

(continues on next page)

(continued from previous page)

```
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@types/use-sync-external-store

- Description: TypeScript definitions for use-sync-external-store
- License: MIT
- Homepage: https://github.com/DefinitelyTyped/DefinitelyTyped/tree/master/types/use-sync-external-store
- Version: 0.0.6

```
MIT License

    Copyright (c) Microsoft Corporation.

    Permission is hereby granted, free of charge, to any person obtaining a copy
    of this software and associated documentation files (the "Software"), to deal
    in the Software without restriction, including without limitation the rights
    to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
    copies of the Software, and to permit persons to whom the Software is
    furnished to do so, subject to the following conditions:

    The above copyright notice and this permission notice shall be included in all
    copies or substantial portions of the Software.

    THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
    IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
    FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
    AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
    LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
    OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
    SOFTWARE
```

@xmldom/xmldom

- Description: A pure JavaScript W3C standard-based (XML DOM Level 2 Core) DOMParser and XMLSerializer module.
- License: MIT
- Homepage: https://github.com/xmldom/xmldom
- Version: 0.8.15

```
Copyright 2019 - present Christopher J. Brody and other contributors, as listed␣
↪in: https://github.com/xmldom/xmldom/graphs/contributors
Copyright 2012 - 2017 @jindw <jindw@xidea.org> and other contributors, as listed␣
```

(continues on next page)

(continued from previous page)

```
↪in: https://github.com/jindw/xmldom/graphs/contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

ace-builds

- Description: Ace (Ajax.org Cloud9 Editor)
- License: BSD-3-Clause
- Homepage: https://github.com/ajaxorg/ace-builds
- Version: 1.44.0

```
Copyright (c) 2010, Ajax.org B.V.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of Ajax.org B.V. nor the
      names of its contributors may be used to endorse or promote products
      derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL AJAX.ORG B.V. BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

ajv

- Description: Another JSON Schema Validator
- License: MIT
- Homepage: https://ajv.js.org
- Version: 8.20.0

```
The MIT License (MIT)

Copyright (c) 2015-2021 Evgeny Poberezkin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

ajv-errors

- Description: Custom error messages in JSON Schemas for Ajv validator
- License: MIT
- Homepage: https://github.com/epoberezkin/ajv-errors#readme
- Version: 3.0.0

```
MIT License

Copyright (c) 2017 Evgeny Poberezkin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

argparse

- Description: CLI arguments parser. Native port of python's argparse.
- License: Python-2.0
- Homepage: nodeca/argparse
- Version: 2.0.1

```
A. HISTORY OF THE SOFTWARE
==========================

Python was created in the early 1990s by Guido van Rossum at Stichting
Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands
as a successor of a language called ABC.  Guido remains Python's
principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for
National Research Initiatives (CNRI, see http://www.cnri.reston.va.us)
in Reston, Virginia where he released several versions of the
software.

In May 2000, Guido and the Python core development team moved to
BeOpen.com to form the BeOpen PythonLabs team.  In October of the same
year, the PythonLabs team moved to Digital Creations, which became
Zope Corporation.  In 2001, the Python Software Foundation (PSF, see
https://www.python.org/psf/) was formed, a non-profit organization
created specifically to own Python-related Intellectual Property.
Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see http://www.opensource.org for
the Open Source Definition).  Historically, most, but not all, Python
releases have also been GPL-compatible; the table below summarizes
the various releases.

    Release         Derived     Year        Owner       GPL-
                    from                                compatible? (1)

    0.9.0 thru 1.2              1991-1995   CWI         yes
    1.3 thru 1.5.2  1.2         1995-1999   CNRI        yes
    1.6             1.5.2       2000        CNRI        no
    2.0             1.6         2000        BeOpen.com  no
    1.6.1           1.6         2001        CNRI        yes (2)
    2.1             2.0+1.6.1   2001        PSF         no
    2.0.1           2.0+1.6.1   2001        PSF         yes
    2.1.1           2.1+2.0.1   2001        PSF         yes
    2.1.2           2.1.1       2002        PSF         yes
    2.1.3           2.1.2       2002        PSF         yes
    2.2 and above   2.1.1       2001-now    PSF         yes

Footnotes:

(1) GPL-compatible doesn't mean that we're distributing Python under
    the GPL.  All Python licenses, unlike the GPL, let you distribute
```

(continues on next page)

(continued from previous page)

```
    a modified version without making your changes open source.  The
    GPL-compatible licenses make it possible to combine Python with
    other software that is released under the GPL; the others don't.

(2) According to Richard Stallman, 1.6.1 is not GPL-compatible,
    because its license has a choice of law clause.  According to
    CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1
    is "not incompatible" with the GPL.

Thanks to the many outside volunteers who have worked under Guido's
direction to make these releases possible.


B. TERMS AND CONDITIONS FOR ACCESSING OR OTHERWISE USING PYTHON
===============================================================

PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2
--------------------------------------------

1. This LICENSE AGREEMENT is between the Python Software Foundation
("PSF"), and the Individual or Organization ("Licensee") accessing and
otherwise using this software ("Python") in source or binary form and
its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby
grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce,
analyze, test, perform and/or display publicly, prepare derivative works,
distribute, and otherwise use Python alone or in any derivative version,
provided, however, that PSF's License Agreement and PSF's notice of copyright,
i.e., "Copyright (c) 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010,
2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020 Python Software␣
↪Foundation;
All Rights Reserved" are retained in Python alone or in any derivative version
prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on
or incorporates Python or any part thereof, and wants to make
the derivative
   [...truncated, see full license at the project homepage...]
```

astronomia

- Description: An astronomical library
- License: MIT
- Homepage: https://github.com/commenthol/astronomia
- Version: 4.3.0

```
The MIT License (MIT)

Copyright (c) 2013 Sonia Keys
Copyright (c) 2016 Commenthol

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
```

(continues on next page)

(continued from previous page)

```
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

babel-plugin-styled-components

- Description: Improve the debugging experience and add server-side rendering support to styled-components
- License: MIT
- Homepage: https://styled-components.com/docs/tooling#babel-plugin
- Version: 2.3.0

```
MIT License

Copyright (c) 2016-present Vladimir Danchenkov and Maximilian Stoiber

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

bail

- Description: Throw a given error
- License: MIT
- Homepage: wooorm/bail
- Version: 2.0.2

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

batch-processor

- Description: Batch processing in JS
- License: MIT
- Homepage: https://github.com/wnr/batch-processor
- Version: 1.0.0

```
The MIT License (MIT)

Copyright (c) 2015 Lucas Wiener

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

bignumber.js

- Description: A library for arbitrary-precision decimal and non-decimal arithmetic
- License: MIT
- Homepage: https://github.com/MikeMcl/bignumber.js

- Version: 9.3.1

caldate

- Description: Calendar date for date-holidays
- License: ISC
- Homepage: https://github.com/commenthol/caldate
- Version: 2.1.0

```
ISC License (ISC)

Copyright (c) 2016, commenthol

Permission to use, copy, modify, and/or distribute this software for
any purpose with or without fee is hereby granted, provided that the
above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR
BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES
OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
SOFTWARE.
```

camelize

- Description: recursively transform key strings to camel-case
- License: MIT
- Homepage: https://github.com/ljharb/camelize
- Version: 1.0.1

```
This software is released under the MIT license:

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

ccount

- Description: Count characters
- License: MIT
- Homepage: wooorm/ccount
- Version: 1.1.0

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

character-entities

- Description: HTML character entity information
- License: MIT
- Homepage: wooorm/character-entities
- Version: 1.2.4

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

(continues on next page)

(continued from previous page)

```
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

character-entities-legacy

- Description: HTML legacy character entity information
- License: MIT
- Homepage: wooorm/character-entities-legacy
- Version: 1.1.4

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

character-reference-invalid

- Description: HTML invalid numeric character reference information
- License: MIT
- Homepage: wooorm/character-reference-invalid
- Version: 1.1.4

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

chroma-js

- Description: JavaScript library for color conversions
- License: (BSD-3-Clause AND Apache-2.0)
- Homepage: https://github.com/gka/chroma.js
- Version: 3.2.0

```
chroma.js - JavaScript library for color conversions

Copyright (c) 2011-2025, Gregor Aisch
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. The name Gregor Aisch may not be used to endorse or promote products
   derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL GREGOR AISCH OR CONTRIBUTORS BE LIABLE FOR ANY␣
↪DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

-------------------------------------------------------

chroma.js includes colors from colorbrewer2.org, which are released under
the following license:

Copyright (c) 2002 Cynthia Brewer, Mark Harrower,
```

(continues on next page)

(continued from previous page)

```
and The Pennsylvania State University.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,
either express or implied. See the License for the specific
language governing permissions and limitations under the License.

------------------------------------------------------

Named colors are taken from X11 Color Names.
http://www.w3.org/TR/css3-color/#svg-color

@preserve
```

classnames

- Description: A simple utility for conditionally joining classNames together
- License: MIT
- Homepage: https://github.com/JedWatson/classnames
- Version: 2.5.1

```
The MIT License (MIT)

Copyright (c) 2018 Jed Watson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

color-blend

- Description: Blends RGBA colors with different blend modes
- License: MIT
- Homepage: Loilo/color-blend

- Version: 2.0.9

```
Copyright (c) 2016 Florian Reuschel

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

colorjs.io

- Description: Let's get serious about color
- License: MIT
- Homepage: https://colorjs.io
- Version: 0.6.1

```
MIT License

Copyright (c) 2021 Lea Verou, Chris Lilley

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

comma-separated-tokens

- Description: Parse and stringify comma-separated tokens
- License: MIT
- Homepage: wooorm/comma-separated-tokens
- Version: 2.0.3

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

commander

- Description: the complete solution for node.js command-line programs
- License: MIT
- Homepage: https://github.com/tj/commander.js
- Version: 2.20.3

```
(The MIT License)

Copyright (c) 2011 TJ Holowaychuk <tj@vision-media.ca>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

(continues on next page)

(continued from previous page)

```
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

compute-scroll-into-view

- Description: The engine that powers scroll-into-view-if-needed
- License: MIT
- Homepage: https://scroll-into-view.dev
- Version: 3.1.1

```
MIT License

Copyright (c) 2025 Cody Olsen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

cookie-storage

- Description: A Web Storage interface for Cookie
- License: MIT
- Homepage: https://github.com/bouzuya/cookie-storage
- Version: 5.0.3

```
The MIT License (MIT)

Copyright (c) 2015 bouzuya

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
```

(continues on next page)

(continued from previous page)

```
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

crypto-js

- Description: JavaScript library of crypto standards.
- License: MIT
- Homepage: http://github.com/brix/crypto-js
- Version: 4.2.0

```
# License

[The MIT License (MIT)](http://opensource.org/licenses/MIT)

Copyright (c) 2009-2013 Jeff Mott
Copyright (c) 2013-2016 Evan Vosberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

css-color-keywords

- Description: A list of all CSS color keywords.
- License: ISC
- Homepage: sonicdoe/css-color-keywords
- Version: 1.0.0

```
ISC License

Copyright (c) 2017, Jakob Krigovsky
```

(continues on next page)

(continued from previous page)

```
Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

css-to-react-native

- Description: Convert CSS text to a React Native stylesheet object
- License: MIT
- Homepage: https://github.com/styled-components/css-to-react-native#readme
- Version: 3.2.0

```
MIT License

Copyright (c) 2016 Jacob Parker and Maximilian Stoiber

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

cssesc

- Description: A JavaScript library for escaping CSS strings and identifiers while generating the shortest possible ASCII-only output.
- License: MIT
- Homepage: https://mths.be/cssesc
- Version: 3.0.0

csstype

- Description: Strict TypeScript and Flow types for style based on MDN data
- License: MIT
- Homepage: https://github.com/frenic/csstype
- Version: 3.2.3

```
Copyright (c) 2017-2018 Fredrik Nicol

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

ctrl-keys

- Description: A tiny, super fast, typescript library to handle keybindings efficiently.
- License: MIT
- Version: 1.0.6

```
MIT License

Copyright (c) 2021 Amine Ben hammou

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

d3-array

- Description: Array manipulation, ordering, searching, summarizing, etc.
- License: ISC
- Homepage: https://d3js.org/d3-array/
- Version: 3.2.4

```
Copyright 2010-2023 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

d3-axis

- Description: Displays automatic reference lines for scales.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-axis/
- Version: 2.1.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-brush

- Description: Select a one- or two-dimensional region using the mouse or touch.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-brush/
- Version: 2.1.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-collection

- Description: Handy data structures for elements keyed by string.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-collection/
- Version: 1.0.7

```
Copyright 2010-2016, Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
```

(continues on next page)

(continued from previous page)

```
* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-color

- Description: Color spaces! RGB, HSL, Cubehelix, Lab and HCL (Lch).
- License: ISC
- Homepage: https://d3js.org/d3-color/
- Version: 3.1.0

```
Copyright 2010-2022 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

d3-dispatch

- Description: Register named callbacks and call them with arguments.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-dispatch/
- Version: 2.0.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
```

(continues on next page)

(continued from previous page)

```
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-drag

- Description: Drag and drop SVG, HTML or Canvas using mouse or touch input.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-drag/
- Version: 2.0.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-ease

- Description: Easing functions for smooth animation.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-ease/
- Version: 2.0.0

```
Copyright 2010-2016 Mike Bostock
Copyright 2001 Robert Penner
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-force

- Description: Force-directed graph layout using velocity Verlet integration.
- License: ISC
- Homepage: https://d3js.org/d3-force/
- Version: 3.0.0

```
Copyright 2010-2021 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
```

(continues on next page)

(continued from previous page)

```
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

d3-format

- Description: Format numbers for human consumption.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-format/
- Version: 2.0.0

```
Copyright 2010-2015 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-hierarchy

- Description: Layout algorithms for visualizing hierarchical data.
- License: ISC
- Homepage: https://d3js.org/d3-hierarchy/
- Version: 3.1.2

```
Copyright 2010-2021 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
```

(continues on next page)

(continued from previous page)

```
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

d3-interpolate

- Description: Interpolate numbers, colors, strings, arrays, objects, whatever!
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-interpolate/
- Version: 2.0.1

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-quadtree

- Description: Two-dimensional recursive spatial subdivision.
- License: ISC
- Homepage: https://d3js.org/d3-quadtree/
- Version: 3.0.1

```
Copyright 2010-2021 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

d3-scale

- Description: Encodings that map abstract data to visual representation.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-scale/
- Version: 3.3.0

```
Copyright 2010-2015 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-scale-chromatic

- Description: Sequential, diverging and categorical color schemes.
- License: ISC
- Homepage: https://d3js.org/d3-scale-chromatic/
- Version: 3.1.0

```
Copyright 2010-2024 Mike Bostock

```

(continues on next page)

(continued from previous page)

```
Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.

Apache-Style Software License for ColorBrewer software and ColorBrewer Color␣
↪Schemes

Copyright 2002 Cynthia Brewer, Mark Harrower, and The Pennsylvania State University

Licensed under the Apache License, Version 2.0 (the "License"); you may not use
this file except in compliance with the License. You may obtain a copy of the
License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed
under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied. See the License for the
specific language governing permissions and limitations under the License.
```

d3-selection

- Description: Data-driven DOM manipulation: select elements and join them to data.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-selection/
- Version: 2.0.0

```
Copyright (c) 2010-2018, Michael Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* The name Michael Bostock may not be used to endorse or promote products
  derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL MICHAEL BOSTOCK BE LIABLE FOR ANY DIRECT,
```

(continues on next page)

(continued from previous page)

```
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-time

- Description: A calculator for humanity's peculiar conventions of time.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-time/
- Version: 2.1.1

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-time-format

- Description: A JavaScript time formatter and parser inspired by strftime and strptime.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-time-format/
- Version: 3.0.0

```
Copyright 2010-2017 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
```

(continues on next page)

(continued from previous page)

```
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-timer

- Description: An efficient queue capable of managing thousands of concurrent animations.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-timer/
- Version: 2.0.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
```

(continues on next page)

(continued from previous page)

```
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-transition

- Description: Animated transitions for D3 selections.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-transition/
- Version: 2.0.0

```
Copyright (c) 2010-2015, Michael Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* The name Michael Bostock may not be used to endorse or promote products
  derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL MICHAEL BOSTOCK BE LIABLE FOR ANY DIRECT,
INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY
OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

TERMS OF USE - EASING EQUATIONS

Open source under the BSD License.

Copyright 2001 Robert Penner
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

- Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
```

(continues on next page)

(continued from previous page)

```
- Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

d3-zoom

- Description: Pan and zoom SVG, HTML or Canvas using mouse or touch input.
- License: BSD-3-Clause
- Homepage: https://d3js.org/d3-zoom/
- Version: 2.0.0

```
Copyright 2010-2016 Mike Bostock
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the author nor the names of contributors may be used to
  endorse or promote products derived from this software without specific prior
  written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

dagre

- Description: Graph layout for JavaScript
- License: MIT
- Homepage: https://github.com/dagrejs/dagre
- Version: 0.8.5

```
Copyright (c) 2012-2014 Chris Pettitt

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

date-bengali-revised

- Description: Revised Bengali Calendar
- License: MIT
- Homepage: https://github.com/commenthol/date-bengali-revised
- Version: 2.1.0

```
MIT License

Copyright (c) 2018-present commenthol

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

date-chinese

- Description: Chinese Calendar
- License: MIT
- Homepage: https://github.com/commenthol/date-chinese
- Version: 2.2.0

```
MIT License

Copyright (c) 2016-present commenthol

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

date-easter

- Description: Calculates Easter for a given year
- License: MIT
- Homepage: https://github.com/commenthol/date-easter#readme
- Version: 1.1.0

```
The MIT License (MIT)

Copyright (c) 2014 commenthol

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
```

(continues on next page)

(continued from previous page)

```
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

date-fns

- Description: Modern JavaScript date utility library
- License: MIT
- Homepage: https://github.com/date-fns/date-fns
- Version: 2.30.0

```
MIT License

Copyright (c) 2021 Sasha Koss and Lesha Koss https://kossnocorp.mit-license.org

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

date-holidays

- Description: worldwide holidays
- License: (ISC AND CC-BY-SA-4.0)
- Homepage: https://github.com/commenthol/date-holidays
- Version: 3.39.2

```
The data contained in `holidays.yaml` is available under CC BY-SA 4.0
<http://creativecommons.org/licenses/by-sa/4.0/> as the data obtained
relies on wikipedia articles.
The required attribution can be found inside the file.

All other code is Copyright (c) 2015 commenthol and available under
ISC <http://opensource.org/licenses/ISC>

----

ISC License (ISC)

Copyright (c) 2015-present, commenthol
```

(continues on next page)

(continued from previous page)

```
Permission to use, copy, modify, and/or distribute this software for
any purpose with or without fee is hereby granted, provided that the
above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR
BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES
OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
SOFTWARE.

----

CC BY-SA 3.0 Attributions

<attribution>

https://da.wikipedia.org/wiki/Danske_helligdage
https://de.wikipedia.org/wiki/Feiertage_in_%C3%96sterreich
https://de.wikipedia.org/wiki/Feiertage_in_Deutschland
https://de.wikipedia.org/wiki/Feiertage_in_Island
https://de.wikipedia.org/wiki/Feiertage_in_Italien
https://de.wikipedia.org/wiki/Feiertage_in_Liechtenstein#Gesetzliche_Feiertage
https://de.wikipedia.org/wiki/Feiertage_in_Polen
https://de.wikipedia.org/wiki/Feiertage_in_Rum%C3%A4nien
https://de.wikipedia.org/wiki/Feiertage_in_Schweden
https://de.wikipedia.org/wiki/Feiertage_in_der_Schweiz
https://de.wikipedia.org/wiki/Kanada#Feiertage
https://de.wikipedia.org/wiki/Portugal#Feiertage
https://el.wikipedia.org/wiki/Επίσημες_αργίες_στην_Ελλάδα
https://en.wikipedia.org/wiki/2023_Singaporean_presidential_election
https://en.wikipedia.org/wiki/Bengali_calendars
https://en.wikipedia.org/wiki/Culture_of_Tonga#Public_holidays
https://en.wikipedia.org/wiki/Family_Day_(Canada)
https://en.wikipedia.org/wiki/Faroe_Islands#Public_holidays
https://en.wikipedia.org/wiki/List_of_public_holidays_in_South_Korea
https://en.wikipedia.org/wiki/Nyepi
https://en.wikipedia.org/wiki/People_Power_Revolution#Commemoration
https://en.wikipedia.org/wiki/Public_holidays_in_Albania
https://en.wikipedia.org/wiki/Public_holidays_in_Algeria
https://en.wikipedia.org/wiki/Public_holidays_in_Andorra
https://en.wikipedia.org/wiki/Public_holidays_in_Australia
https://en.wikipedia.org/wiki/Public_holidays_in_Bahrain
https://en.wikipedia.org/wiki/Public_holidays_in_Bangladesh
https://en.wikipedia.org/wiki/Public_holidays_in_Belarus
https://en.wikipedia.org/wiki/Public_holidays_in_Belgium
https://en.wikipedia.org/wiki/Public_holidays_in_Belize
https://en.wikipedia.org/wiki/Public_holidays_in_Bermuda
https://en.wikipedia.org/wiki/Public_holidays_in_Bolivia
https://en.wikipedia.org/wiki/Public_holidays_in_Bosnia_and_Herzegovina
https://en.wikipedia.org/wiki/Public_holidays_in_Brazil
https://en.wikipedia.org/wiki/Public_holidays_in_Brunei
https://en.wikipedia.org/wiki/Public_holidays_in_Bulgaria
https://en.wikipedia.org/wiki/Public_holidays_in_Canada
```

(continues on next page)

(continued from previous page)

```
https://en.wikipedia.org/wiki/Public_holidays_in_Chile
https://en.wikipedia.org/wiki/Public_holidays_in_China
https://en.wikipedia.org/wiki/Public_holidays_in_Christmas_Island
https://en.wikipedia.org/wiki/Public_holidays_in_Costa_Rica
https://en.wikipedia.org/wiki/Public_holidays_in_Croatia
https://en.wikipedia.org/wiki/Public_holidays_in_Cuba
https://en.wikipedia.org/wiki/Public_holidays_in_Cyprus
https://en.wikipedia.org/wiki/Public_holidays_in_Denmark
https://en.wikipedia.org/wiki/Public_holidays_in_Djibouti
https://en.wikipedia.org/wiki/Public_holidays_in_Ecuador
https://en.wikipedia.org/wiki/Public_holidays_in_Egypt
https://en.wikipedia.org/wiki/Public_holidays_in_Equatorial_Guinea
https://en.wikipedia.org/wiki/Public_holidays_in_Eritre
   [...truncated, see full license at the project homepage...]
```

date-holidays-parser

- Description: parser for worldwide holidays
- License: ISC
- Homepage: https://github.com/commenthol/date-holidays-parser
- Version: 3.7.0

```
ISC License (ISC)

Copyright (c) 2015-present, commenthol

Permission to use, copy, modify, and/or distribute this software for
any purpose with or without fee is hereby granted, provided that the
above copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL
WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR
BE LIABLE FOR ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES
OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION,
ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS
SOFTWARE.
```

debug

- Description: Lightweight debugging utility for Node.js and the browser
- License: MIT
- Homepage: https://github.com/debug-js/debug
- Version: 4.4.3

```
(The MIT License)

Copyright (c) 2014-2017 TJ Holowaychuk <tj@vision-media.ca>
Copyright (c) 2018-2021 Josh Junon

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software
```

(continues on next page)

(continued from previous page)

```
and associated documentation files (the 'Software'), to deal in the Software␣
↪without restriction,
including without limitation the rights to use, copy, modify, merge, publish,␣
↪distribute, sublicense,
and/or sell copies of the Software, and to permit persons to whom the Software is␣
↪furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial
portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT
LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND␣
↪NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM,␣
↪DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

decimal.js-light

- Description: An arbitrary-precision Decimal type for JavaScript.
- License: MIT
- Homepage: https://github.com/MikeMcl/decimal.js-light
- Version: 2.5.1

decode-named-character-reference

- Description: Decode named character references
- License: MIT
- Homepage: wooorm/decode-named-character-reference
- Version: 1.3.0

```
(The MIT License)

Copyright (c) Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
```

(continues on next page)

(continued from previous page)

```
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

deepmerge

- Description: A library for deep (recursive) merging of Javascript objects
- License: MIT
- Homepage: https://github.com/TehShrike/deepmerge
- Version: 4.3.1

dequal

- Description: A tiny (304B to 489B) utility for check for deep equality
- License: MIT
- Homepage: lukeed/dequal
- Version: 2.0.3

```
The MIT License (MIT)

Copyright (c) Luke Edwards <luke.edwards05@gmail.com> (lukeed.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

diff

- Description: A JavaScript text diff implementation.
- License: BSD-3-Clause
- Homepage: https://github.com/kpdecker/jsdiff
- Version: 8.0.4

```
BSD 3-Clause License

Copyright (c) 2009-2015, Kevin Decker <kpdecker@gmail.com>
```

(continues on next page)

(continued from previous page)

```
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its
   contributors may be used to endorse or promote products derived from
   this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

direction

- Description: Detect directionality: left-to-right, right-to-left, or neutral
- License: MIT
- Homepage: wooorm/direction
- Version: 1.0.4

```
(The MIT License)

Copyright (c) 2014 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
```

(continues on next page)

(continued from previous page)

```
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

discontinuous-range

- Description: for adding, subtracting, and indexing discontinuous ranges of numbers
- License: MIT
- Homepage: https://github.com/dtudury/discontinuous-range
- Version: 1.0.0

```
The MIT License (MIT)

Copyright (c) 2014 David Tudury

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

dompurify

- Description: DOMPurify is a DOM-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG. It runs as JavaScript and works in all modern browsers, as well as in Node.js (via jsdom). DOMPurify is written by security people who have vast background in web attacks and XSS. Fear not.
- License: (MPL-2.0 OR Apache-2.0)
- Homepage: https://github.com/cure53/DOMPurify
- Version: 3.4.17

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

   TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

   1. Definitions.

      "License" shall mean the terms and conditions for use, reproduction,
      and distribution as defined by Sections 1 through 9 of this document.

      "Licensor" shall mean the copyright owner or entity authorized by
```

(continues on next page)

(continued from previous page)

```
    the copyright owner that is granting the License.

    "Legal Entity" shall mean the union of the acting entity and all
    other entities that control, are controlled by, or are under common
    control with that entity. For the purposes of this definition,
    "control" means (i) the power, direct or indirect, to cause the
    direction or management of such entity, whether by contract or
    otherwise, or (ii) ownership of fifty percent (50%) or more of the
    outstanding shares, or (iii) beneficial ownership of such entity.

    "You" (or "Your") shall mean an individual or Legal Entity
    exercising permissions granted by this License.

    "Source" form shall mean the preferred form for making modifications,
    including but not limited to software source code, documentation
    source, and configuration files.

    "Object" form shall mean any form resulting from mechanical
    transformation or translation of a Source form, including but
    not limited to compiled object code, generated documentation,
    and conversions to other media types.

    "Work" shall mean the work of authorship, whether in Source or
    Object form, made available under the License, as indicated by a
    copyright notice that is included in or attached to the work
    (an example is provided in the Appendix below).

    "Derivative Works" shall mean any work, whether in Source or Object
    form, that is based on (or derived from) the Work and for which the
    editorial revisions, annotations, elaborations, or other modifications
    represent, as a whole, an original work of authorship. For the purposes
    of this License, Derivative Works shall not include works that remain
    separable from, or merely link (or bind by name) to the interfaces of,
    the Work and Derivative Works thereof.

    "Contribution" shall mean any work of authorship, including
    the original version of the Work and any modifications or additions
    to that Work or Derivative Works thereof, that is intentionally
    submitted to Licensor for inclusion in the Work by the copyright owner
    or by an individual or Legal Entity authorized to submit on behalf of
    the copyright owner. For the purposes of this definition, "submitted"
    means any form of electronic, verbal, or written communication sent
    to the Licensor or its representatives, including but not limited to
    communication on electronic mailing lists, source code control systems,
    and issue tracking systems that are managed by, or on behalf of, the
    Licensor for the purpose of discussing and improving the Work, but
    excluding communication that is conspicuously marked or otherwise
    designated in writing by the copyright owner as "Not a Contribution."

    "Contributor" shall mean Licensor and any individual or Legal Entity
    on behalf of whom a Contribution has been received by Licensor and
    subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
```

(continues on next page)

(continued from previous page)

```
      copyright license to reproduce, prepare Derivative Works of,
      publicly display, publicly perform, sublicense, and distribute the
      Work and such Derivative Works in Source or Object form.

   3. Grant of Patent License. Subject to the terms and conditions of
      this License, each Contributor hereby
   [...truncated, see full license at the project homepage...]
```

earcut

- Description: The fastest and smallest JavaScript polygon triangulation library for your WebGL apps
- License: ISC
- Homepage: https://github.com/mapbox/earcut
- Version: 3.2.3

```
ISC License

Copyright (c) 2026, Mapbox

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

element-resize-detector

- Description: Resize event emitter for elements.
- License: MIT
- Homepage: https://github.com/wnr/element-resize-detector
- Version: 1.2.4

```
The MIT License (MIT)

Copyright (c) 2016 Lucas Wiener

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

(continues on next page)

(continued from previous page)

```
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

es-toolkit

- Description: A state-of-the-art, high-performance JavaScript utility library with a small bundle size and strong type annotations.
- License: MIT
- Homepage: https://es-toolkit.dev
- Version: 1.52.0

```
MIT License

Copyright (c) 2024 Viva Republica, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

escape-string-regexp

- Description: Escape RegExp special characters
- License: MIT
- Homepage: sindresorhus/escape-string-regexp
- Version: 4.0.0

```
MIT License

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (https://sindresorhus.com)

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
```

(continues on next page)

(continued from previous page)

```
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

extend

- Description: Port of jQuery.extend for node.js and the browser
- License: MIT
- Homepage: https://github.com/justmoon/node-extend
- Version: 3.0.2

```
The MIT License (MIT)

Copyright (c) 2014 Stefan Thomas

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

fast-deep-equal

- Description: Fast deep equal
- License: MIT
- Homepage: https://github.com/epoberezkin/fast-deep-equal#readme
- Version: 3.1.3

```
MIT License

Copyright (c) 2017 Evgeny Poberezkin
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

fast-json-patch

- Description: Fast implementation of JSON-Patch (RFC-6902) with duplex (observe changes) capabilities
- License: MIT
- Homepage: https://github.com/Starcounter-Jack/JSON-Patch
- Version: 3.1.1

```
(The MIT License)

Copyright (c) 2013, 2014, 2020 Joachim Wester

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

fast-png

- Description: PNG image decoder and encoder written entirely in JavaScript
- License: MIT
- Homepage: https://github.com/image-js/fast-png#readme
- Version: 6.4.0

```
MIT License

Copyright (c) 2015 Michaël Zasso

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

fast-uri

- Description: Dependency-free RFC 3986 URI toolbox
- License: BSD-3-Clause
- Homepage: https://github.com/fastify/fast-uri
- Version: 3.1.9

```
Copyright (c) 2011-2021, Gary Court until https://github.com/garycourt/uri-js/
↪commit/a1acf730b4bba3f1097c9f52e7d9d3aba8cdcaae
Copyright (c) 2021-present The Fastify team <https://github.com/fastify/fastify
↪#team>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * The names of any contributors may not be used to endorse or promote
      products derived from this software without specific prior written
      permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDERS AND CONTRIBUTORS BE LIABLE FOR␣
↪ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
```

(continues on next page)

(continued from previous page)

```
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

                                    *   *   *

The complete list of contributors can be found at:
- https://github.com/garycourt/uri-js/graphs/contributors
```

fflate

- Description: High performance (de)compression in an 8kB package
- License: MIT
- Homepage: https://101arrowz.github.io/fflate
- Version: 0.8.3

```
MIT License

Copyright (c) 2026 Arjun Barrett

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

fscreen

- Description: Vendor agnostic access to the fullscreen spec api
- License: MIT
- Homepage: https://github.com/rafgraph/fscreen#readme
- Version: 1.2.0

```
The MIT License (MIT)

Copyright (c) 2017 Rafael Pedicini

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
```

(continues on next page)

(continued from previous page)

```
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

gl-matrix

- Description: Javascript Matrix and Vector library for High Performance WebGL apps
- License: MIT
- Homepage: http://glmatrix.net
- Version: 3.4.4

```
Copyright (c) 2015-2025, Brandon Jones, Colin MacKenzie IV.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

graphlib

- Description: A directed and undirected multi-graph library
- License: MIT
- Homepage: https://github.com/dagrejs/graphlib
- Version: 2.1.8

```
Copyright (c) 2012-2014 Chris Pettitt

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

(continues on next page)

(continued from previous page)

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

has-flag

- Description: Check if argv has a specific flag
- License: MIT
- Homepage: sindresorhus/has-flag
- Version: 4.0.0

```
MIT License

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

hash-it

- Description: Hash any object based on its value
- License: MIT
- Homepage: https://github.com/planttheidea/hash-it#readme
- Version: 5.0.2

```
The MIT License (MIT)

Copyright (c) 2015 Plant The Idea
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

hast-util-whitespace

- Description: hast utility to check if a node is inter-element whitespace
- License: MIT
- Homepage: syntax-tree/hast-util-whitespace
- Version: 2.0.1

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

hoist-non-react-statics

- Description: Copies non-react specific statics from a child component to a parent component
- License: BSD-3-Clause
- Homepage: https://github.com/mridgway/hoist-non-react-statics
- Version: 3.3.2

```
Software License Agreement (BSD License)
========================================

Copyright (c) 2015, Yahoo! Inc. All rights reserved.
----------------------------------------------------

Redistribution and use of this software in source and binary forms, with or
without modification, are permitted provided that the following conditions are
met:

  * Redistributions of source code must retain the above copyright notice, this
    list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright notice,
    this list of conditions and the following disclaimer in the documentation
    and/or other materials provided with the distribution.
  * Neither the name of Yahoo! Inc. nor the names of YUI's contributors may be
    used to endorse or promote products derived from this software without
    specific prior written permission of Yahoo! Inc.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

immer

- Description: Create your next immutable state by mutating the current one
- License: MIT
- Homepage: https://github.com/immerjs/immer#readme
- Version: 10.2.0

```
MIT License

Copyright (c) 2017 Michel Weststrate

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

(continues on next page)

(continued from previous page)

```
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

inline-style-parser

- Description: An inline style parser.
- License: MIT
- Homepage: https://github.com/remarkablemark/inline-style-parser
- Version: 0.1.1

internmap

- Description: Map and Set with automatic key interning
- License: ISC
- Homepage: https://github.com/mbostock/internmap/
- Version: 2.0.3

```
Copyright 2021 Mike Bostock

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

intl-tel-input

- Description: A JavaScript plugin for entering and validating international telephone numbers
- License: MIT
- Homepage: https://intl-tel-input.com
- Version: 25.15.1

```
The MIT License (MIT)

Copyright (c) 2014-2016 Jack O'Connor

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

iobuffer

- Description: Read and write binary data on ArrayBuffers
- License: MIT
- Homepage: https://github.com/image-js/iobuffer#readme
- Version: 5.4.0

```
The MIT License (MIT)

Copyright (c) 2015 Michaël Zasso

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

is-alphabetical

- Description: Check if a character is alphabetical
- License: MIT
- Homepage: wooorm/is-alphabetical
- Version: 1.0.4

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
```

(continues on next page)

(continued from previous page)

```
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

is-alphanumerical

- Description: Check if a character is alphanumerical
- License: MIT
- Homepage: wooorm/is-alphanumerical
- Version: 1.0.4

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

is-buffer

- Description: Determine if an object is a Buffer
- License: MIT
- Homepage: https://github.com/feross/is-buffer
- Version: 2.0.5

```
The MIT License (MIT)

Copyright (c) Feross Aboukhadijeh

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

is-decimal

- Description: Check if a character is decimal
- License: MIT
- Homepage: wooorm/is-decimal
- Version: 1.0.4

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

is-hexadecimal

- Description: Check if a character is hexadecimal
- License: MIT
- Homepage: wooorm/is-hexadecimal
- Version: 1.0.4

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

is-hotkey

- Description: Check whether a browser event matches a hotkey.
- License: MIT
- Homepage: https://github.com/ianstormtaylor/is-hotkey
- Version: 0.2.0

is-plain-obj

- Description: Check if a value is a plain object
- License: MIT
- Homepage: sindresorhus/is-plain-obj
- Version: 1.1.0

```
The MIT License (MIT)

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

is-plain-object

- Description: Returns true if an object was created by the *Object* constructor, or Object.create(null).
- License: MIT
- Homepage: https://github.com/jonschlinkert/is-plain-object
- Version: 5.1.0

```
The MIT License (MIT)

Copyright (c) 2014-2017, Jon Schlinkert.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

isomorphic-fetch

- Description: Isomorphic WHATWG Fetch API, for Node & Browserify
- License: MIT
- Homepage: https://github.com/matthew-andrews/isomorphic-fetch/issues
- Version: 3.0.0

```
The MIT License (MIT)

Copyright (c) 2015 Matt Andrews

Permission is hereby granted, free of charge, to any person obtaining a copy
```

(continues on next page)

(continued from previous page)

```
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

jalaali-js

- Description: Converts Gregorian and Jalaali calendars to each other
- License: MIT
- Homepage: https://github.com/jalaali/jalaali-js
- Version: 2.0.1

```
MIT License

Copyright (c) 2020 Behrang Norouzinia

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

jexl

- Description: Javascript Expression Language: Powerful context-based expression parser and evaluator
- License: MIT
- Homepage: https://github.com/TomFrost/jexl
- Version: 2.3.0

```
Copyright (c) 2020 Tom Shawver

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

js-tokens

- Description: A regex that tokenizes JavaScript.
- License: MIT
- Homepage: lydell/js-tokens
- Version: 4.0.0

```
The MIT License (MIT)

Copyright (c) 2014, 2015, 2016, 2017, 2018 Simon Lydell

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

js-yaml

- Description: YAML 1.2 parser and serializer
- License: MIT
- Homepage: nodeca/js-yaml

- Version: 4.3.2

```
(The MIT License)

Copyright (C) 2011-2015 by Vitaly Puzrin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

jsesc

- Description: Given some data, jsesc returns the shortest possible stringified & ASCII-safe representation of that data.
- License: MIT
- Homepage: https://mths.be/jsesc
- Version: 3.1.0

json-schema-traverse

- Description: Traverse JSON Schema passing each schema object to callback
- License: MIT
- Homepage: https://github.com/epoberezkin/json-schema-traverse#readme
- Version: 1.0.0

```
MIT License

Copyright (c) 2017 Evgeny Poberezkin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
```

(continues on next page)

(continued from previous page)

```
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

json-stringify-pretty-compact

- Description: The best of both *JSON.stringify(obj)* and *JSON.stringify(obj, null, indent)*.
- License: MIT
- Homepage: lydell/json-stringify-pretty-compact
- Version: 4.0.0

```
The MIT License (MIT)

Copyright (c) 2014, 2016, 2017, 2019, 2021, 2022 Simon Lydell

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

jsonata

- Description: JSON query and transformation language
- License: MIT
- Homepage: http://jsonata.org/
- Version: 2.2.2

```
MIT license

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

jspdf

- Description: PDF Document creation from JavaScript
- License: MIT
- Homepage: https://github.com/parallax/jsPDF
- Version: 4.2.1

```
Copyright
(c) 2010-2025 James Hall, https://github.com/MrRio/jsPDF
(c) 2015-2025 yWorks GmbH, https://www.yworks.com/

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

kdbush

- Description: A very fast static 2D index for points based on kd-tree.
- License: ISC
- Homepage: https://github.com/mourner/kdbush
- Version: 4.1.0

```
ISC License

Copyright (c) 2026, Vladimir Agafonkin

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

keycode

- Description: Convert between keyboard keycodes and keynames and vice versa.
- License: MIT
- Homepage: https://github.com/timoxley/keycode
- Version: 2.2.1

```
The MIT License (MIT)

Copyright (c) 2014 Tim Oxley

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

kleur

- Description: The fastest Node.js library for formatting terminal text with ANSI colors~!
- License: MIT
- Homepage: lukeed/kleur
- Version: 4.1.5

```
The MIT License (MIT)

Copyright (c) Luke Edwards <luke.edwards05@gmail.com> (lukeed.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

(continues on next page)

(continued from previous page)

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

lodash

- Description: Lodash modular utilities.
- License: MIT
- Homepage: https://lodash.com/
- Version: 4.18.1

```
Copyright OpenJS Foundation and other contributors <https://openjsf.org/>

Based on Underscore.js, copyright Jeremy Ashkenas,
DocumentCloud and Investigative Reporters & Editors <http://underscorejs.org/>

This software consists of voluntary contributions made by many
individuals. For exact contribution history, see the revision history
available at https://github.com/lodash/lodash

The following license applies to all parts of this software except as
documented below:

====

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

====
```

(continues on next page)

(continued from previous page)

```
Copyright and related rights for sample code are waived via CC0. Sample
code is defined as all source code displayed within the prose of the
documentation.

CC0: http://creativecommons.org/publicdomain/zero/1.0/

====

Files located in the node_modules and vendor directories are externally
maintained libraries used by this software which have their own
licenses; we recommend you read them, as their terms may differ from the
terms above.
```

longest-streak

- Description: Count the longest repeating streak of a character
- License: MIT
- Homepage: wooorm/longest-streak
- Version: 2.0.4

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <mailto:tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

loose-envify

- Description: Fast (and loose) selective *process.env* replacer using js-tokens instead of an AST
- License: MIT
- Homepage: https://github.com/zertosh/loose-envify
- Version: 1.4.0

```
The MIT License (MIT)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2015 Andres Suarez <zertosh@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

maplibre-gl

- Description: BSD licensed community fork of mapbox-gl, a WebGL interactive maps library
- License: BSD-3-Clause
- Homepage: https://maplibre.org/
- Version: 6.4.1

```
Copyright (c) 2023, MapLibre contributors

All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright notice,
      this list of conditions and the following disclaimer in the documentation
      and/or other materials provided with the distribution.
    * Neither the name of MapLibre GL JS nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

(continues on next page)

(continued from previous page)

```
--------------------------------------------------------------------------------

Contains code from mapbox-gl-js v1.13 and earlier

Version v1.13 of mapbox-gl-js and earlier are licensed under a BSD-3-Clause license

Copyright (c) 2020, Mapbox
Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice,
  this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.
* Neither the name of Mapbox GL JS nor the names of its contributors
  may be used to endorse or promote products derived from this software
  without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE,
EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


--------------------------------------------------------------------------------

Contains code from glfx.js

Copyright (C) 2011 by Evan Wallace

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT
   [...truncated, see full license at the project homepage...]
```

markdown-table

- Description: Markdown tables
- License: MIT
- Homepage: wooorm/markdown-table
- Version: 2.0.0

```
(The MIT License)

Copyright (c) 2014 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-definitions

- Description: mdast utility to find definition nodes in a tree
- License: MIT
- Homepage: syntax-tree/mdast-util-definitions
- Version: 5.1.2

```
(The MIT License)

Copyright (c) 2015-2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

(continues on next page)

(continued from previous page)

```
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-find-and-replace

- Description: mdast utility to find and replace text in a tree
- License: MIT
- Homepage: syntax-tree/mdast-util-find-and-replace
- Version: 1.1.1

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-from-markdown

- Description: mdast utility to parse markdown
- License: MIT
- Homepage: syntax-tree/mdast-util-from-markdown
- Version: 1.3.1

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-gfm

- Description: mdast extension to parse and serialize GFM (GitHub Flavored Markdown)
- License: MIT
- Homepage: syntax-tree/mdast-util-gfm
- Version: 0.1.2

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-gfm-autolink-literal

- Description: mdast extension to parse and serialize GFM autolink literals
- License: MIT
- Homepage: syntax-tree/mdast-util-gfm-autolink-literal
- Version: 0.1.3

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-gfm-strikethrough

- Description: mdast extension to parse and serialize GFM strikethrough
- License: MIT
- Homepage: syntax-tree/mdast-util-gfm-strikethrough
- Version: 0.2.3

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-gfm-table

- Description: mdast extension to parse and serialize GFM tables
- License: MIT
- Homepage: syntax-tree/mdast-util-gfm-table

- Version: 0.1.6

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-gfm-task-list-item

- Description: mdast extension to parse and serialize GFM task list items
- License: MIT
- Homepage: syntax-tree/mdast-util-gfm-task-list-item
- Version: 0.1.6

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-to-hast

- Description: mdast utility to transform to hast
- License: MIT
- Homepage: syntax-tree/mdast-util-to-hast
- Version: 12.3.0

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-to-markdown

- Description: mdast utility to serialize markdown
- License: MIT
- Homepage: syntax-tree/mdast-util-to-markdown
- Version: 0.6.5

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

(continues on next page)

(continued from previous page)

```
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

mdast-util-to-string

- Description: mdast utility to get the plain text content of a node
- License: MIT
- Homepage: syntax-tree/mdast-util-to-string
- Version: 2.0.0

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

memoize-one

- Description: A memoization library which only remembers the latest invocation
- License: MIT
- Homepage: https://github.com/alexreardon/memoize-one
- Version: 6.0.0

```
MIT License

Copyright (c) 2019 Alexander Reardon

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

micromark

- Description: small commonmark compliant markdown parser with positional info and concrete tokens
- License: MIT
- Homepage: micromark/micromark
- Version: 2.11.4

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm

- Description: micromark extension to support GFM (GitHub Flavored Markdown)
- License: MIT
- Homepage: micromark/micromark-extension-gfm
- Version: 0.3.3

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
```

(continues on next page)

(continued from previous page)

```
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm-autolink-literal

- Description: micromark extension to support GFM autolink literals
- License: MIT
- Homepage: micromark/micromark-extension-gfm-autolink-literal
- Version: 0.5.7

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm-strikethrough

- Description: micromark extension to support GFM strikethrough
- License: MIT
- Homepage: micromark/micromark-extension-gfm-strikethrough
- Version: 0.6.5

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm-table

- Description: micromark extension to support GFM tables
- License: MIT
- Homepage: micromark/micromark-extension-gfm-table
- Version: 0.4.3

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm-tagfilter

- Description: micromark extension to support GFM tagfilter
- License: MIT
- Homepage: micromark/micromark-extension-gfm-tagfilter
- Version: 0.3.0

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-extension-gfm-task-list-item

- Description: micromark extension to support GFM task list items
- License: MIT
- Homepage: micromark/micromark-extension-gfm-task-list-item
- Version: 0.3.3

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
```

(continues on next page)

(continued from previous page)

```
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

micromark-util-character

- Description: micromark utility to handle character codes
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-character
- Version: 1.2.0

micromark-util-decode-numeric-character-reference

- Description: micromark utility to decode numeric character references
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-decode-numeric-character-reference
- Version: 1.1.0

micromark-util-decode-string

- Description: micromark utility to decode markdown strings
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-decode-string
- Version: 1.1.0

micromark-util-encode

- Description: micromark utility to encode dangerous html characters
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-encode
- Version: 1.1.0

micromark-util-normalize-identifier

- Description: micromark utility normalize identifiers (as found in references, definitions)
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-normalize-identifier
- Version: 1.1.0

micromark-util-sanitize-uri

- Description: micromark utility to sanitize urls
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-sanitize-uri
- Version: 1.2.0

micromark-util-symbol

- Description: micromark utility with symbols
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-symbol
- Version: 1.1.0

micromark-util-types

- Description: micromark utility with a couple of typescript types
- License: MIT
- Homepage: https://github.com/micromark/micromark/tree/main/packages/micromark-util-types
- Version: 1.1.0

minimist

- Description: parse argument options
- License: MIT
- Homepage: https://github.com/minimistjs/minimist
- Version: 1.2.8

```
This software is released under the MIT license:

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

moment

- Description: Parse, validate, manipulate, and display dates
- License: MIT
- Homepage: https://momentjs.com
- Version: 2.31.0

```
Copyright (c) OpenJS Foundation and other contributors

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
```

(continues on next page)

(continued from previous page)

```
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

moment-timezone

- Description: Parse and display moments in any timezone.
- License: MIT
- Homepage: http://momentjs.com/timezone/
- Version: 0.6.6

```
The MIT License (MIT)

Copyright (c) JS Foundation and other contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

moo

- Description: Optimised tokenizer/lexer generator! Much performance. Moo!
- License: BSD-3-Clause
- Homepage: https://github.com/tjvr/moo
- Version: 0.5.3

```
BSD 3-Clause License

Copyright (c) 2017, Tim Radvan (tjvr)
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of the copyright holder nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

mri

- Description: Quickly scan for CLI flags and arguments
- License: MIT
- Homepage: lukeed/mri
- Version: 1.2.0

ms

- Description: Tiny millisecond conversion utility
- License: MIT
- Homepage: vercel/ms
- Version: 2.1.3

murmurhash-js

- Description: Native JS murmur hash implementation
- License: MIT
- Homepage: https://github.com/mikolalysenko/murmurhash-js
- Version: 1.0.0

nearley

- Description: Simple, fast, powerful parser toolkit for JavaScript.
- License: MIT
- Homepage: https://github.com/hardmath123/nearley
- Version: 2.20.1

```
The MIT License (MIT)

Copyright (c) 2014, 2015, 2016, 2017, 2018, 2019 Kartik Chandra, Tim Radvan

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

node-fetch

- Description: A light-weight module that brings window.fetch to node.js
- License: MIT
- Homepage: https://github.com/bitinn/node-fetch
- Version: 2.7.0

```
The MIT License (MIT)

Copyright (c) 2016 David Frank

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

numbro

- Description: Format and manipulate numbers.
- License: MIT
- Homepage: http://numbrojs.com
- Version: 2.5.0

```
Copyright (c) 2017-2019 Benjamin Van Ryseghem<benjamin@vanryseghem.com>
Copyright (c) 2015-2017 Företagsplatsen

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

numeral

- Description: Format and manipulate numbers.
- License: MIT
- Homepage: http://numeraljs.com
- Version: 2.0.6

```
Copyright (c) 2016 Adam Draper

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
```

(continues on next page)

(continued from previous page)

```
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

object-assign

- Description: ES2015 *Object.assign()* ponyfill
- License: MIT
- Homepage: sindresorhus/object-assign
- Version: 4.1.1

```
The MIT License (MIT)

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

pako

- Description: zlib port to javascript - fast, modularized, with browser support
- License: (MIT AND Zlib)
- Homepage: nodeca/pako
- Version: 2.2.0

```
(The MIT License)

Copyright (C) 2014-2017 by Vitaly Puzrin and Andrei Tuputcyn

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
```

(continues on next page)

(continued from previous page)

```
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

parse-entities

- Description: Parse HTML character references: fast, spec-compliant, positional information
- License: MIT
- Homepage: wooorm/parse-entities
- Version: 2.0.0

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <mailto:tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

pbf

- Description: a low-level, lightweight protocol buffers implementation in JavaScript
- License: BSD-3-Clause
- Homepage: https://github.com/mapbox/pbf
- Version: 5.1.2

```
Copyright (c) 2026, Mapbox
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name of pbf nor the names of its
  contributors may be used to endorse or promote products derived from
  this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

picocolors

- Description: The tiniest and the fastest library for terminal output formatting with ANSI colors
- License: ISC
- Homepage: alexeyraspopov/picocolors
- Version: 1.1.1

```
ISC License

Copyright (c) 2021-2024 Oleksii Raspopov, Kostiantyn Denysov, Anton Verinov

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

picomatch

- Description: Blazing fast and accurate glob matcher written in JavaScript, with no dependencies and full support for standard and extended Bash glob features, including braces, extglobs, POSIX brackets, and regular expressions.
- License: MIT
- Homepage: https://github.com/micromatch/picomatch
- Version: 4.0.7

```
The MIT License (MIT)

Copyright (c) 2017-present, Jon Schlinkert.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

postcss-value-parser

- Description: Transforms css values and at-rule params into the tree
- License: MIT
- Homepage: https://github.com/TrySound/postcss-value-parser
- Version: 4.2.0

```
Copyright (c) Bogdan Chadkin <trysound@yandex.ru>

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
```

(continues on next page)

(continued from previous page)

```
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.
```

potpack

- Description: A tiny library for packing 2D rectangles (for sprite layouts)
- License: ISC
- Homepage: https://mapbox.github.io/potpack/
- Version: 2.1.0

```
ISC License

Copyright (c) 2025, Mapbox

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

prepin

- Description: An in-place pre-processor for javascript files using C style pre-compile syntax to compose custom builds from npm-packages
- License: Unlicense
- Homepage: https://github.com/commenthol/prepin#readme
- Version: 1.0.3

prop-types

- Description: Runtime type checking for React props and similar objects.
- License: MIT
- Homepage: https://facebook.github.io/react/
- Version: 15.8.1

```
MIT License

Copyright (c) 2013-present, Facebook, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

(continues on next page)

(continued from previous page)

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

property-information

- Description: Info on the properties and attributes of the web platform
- License: MIT
- Homepage: wooorm/property-information
- Version: 6.5.0

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <mailto:tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

protocol-buffers-schema

- Description: No nonsense protocol buffers schema parser written in Javascript
- License: MIT
- Homepage: https://github.com/mafintosh/protocol-buffers-schema
- Version: 3.6.1

```
The MIT License (MIT)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2014 Mathias Buus

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

punycode

- Description: A robust Punycode converter that fully complies to RFC 3492 and RFC 5891, and works on nearly all JavaScript platforms.
- License: MIT
- Homepage: https://mths.be/punycode
- Version: 2.3.1

quickselect

- Description: A tiny and fast selection algorithm in JavaScript.
- License: ISC
- Homepage: github:mourner/quickselect
- Version: 3.0.0

```
ISC License

Copyright (c) 2024, Vladimir Agafonkin

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

railroad-diagrams

- Description: A small JS+SVG library for drawing railroad syntax diagrams.
- License: CC0-1.0
- Homepage: https://github.com/tabatkins/railroad-diagrams
- Version: 1.0.0

randexp

- Description: Create random strings that match a given regular expression.
- License: MIT
- Homepage: http://fent.github.io/randexp.js/
- Version: 0.4.6

```
Copyright (C) 2011 by Roly Fentanes

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

react

- Description: React is a JavaScript library for building user interfaces.
- License: MIT
- Homepage: https://reactjs.org/
- Version: 18.3.1

```
MIT License

Copyright (c) Facebook, Inc. and its affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-datepicker

- Description: A simple and reusable datepicker component for React
- License: MIT
- Homepage: https://github.com/Hacker0x01/react-datepicker
- Version: 4.25.0

```
The MIT License (MIT)

Copyright (c) 2014-2023 HackerOne Inc and individual contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-dom

- Description: React package for working with the DOM.
- License: MIT
- Homepage: https://reactjs.org/
- Version: 18.3.1

```
MIT License

Copyright (c) Facebook, Inc. and its affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
```

(continues on next page)

(continued from previous page)

```
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-fast-compare

- Description: Fastest deep equal comparison for React. Great for React.memo & shouldComponentUpdate. Also really fast general-purpose deep comparison.
- License: MIT
- Homepage: https://github.com/FormidableLabs/react-fast-compare
- Version: 3.2.2

```
MIT License

Copyright (c) 2018 Formidable Labs
Copyright (c) 2017 Evgeny Poberezkin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-full-screen

- Description: Component and Hook for handling full screen components
- License: MIT
- Homepage: https://github.com/snakesilk/react-fullscreen
- Version: 1.1.1

```
MIT License

Copyright (c) 2017 Pontus Persson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-is

- Description: Brand checking of React Elements.
- License: MIT
- Homepage: https://reactjs.org/
- Version: 16.13.1

```
MIT License

Copyright (c) Facebook, Inc. and its affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-markdown

- Description: React component to render markdown
- License: MIT
- Homepage: remarkjs/react-markdown
- Version: 8.0.7

```
The MIT License (MIT)

Copyright (c) 2015 Espen Hovlandsdal

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-onclickoutside

- Description: An onClickOutside wrapper for React components
- License: MIT
- Homepage: https://github.com/Pomax/react-onclickoutside
- Version: 6.13.2

```
Copyright 2015-2022 Mike "Pomax" Kamermans

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-popper

- Description: Official library to use Popper on React projects
- License: MIT
- Homepage: https://popper.js.org/react-popper
- Version: 2.3.0

```
The MIT License (MIT)

Copyright (c) 2018 React Popper authors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-redux

- Description: Official React bindings for Redux
- License: MIT
- Homepage: https://github.com/reduxjs/react-redux
- Version: 9.3.0

```
The MIT License (MIT)

Copyright (c) 2015-present Dan Abramov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

react-resize-detector

- Description: React resize detector
- License: MIT
- Homepage: https://github.com/maslianok/react-resize-detector
- Version: 12.3.0

```
The MIT License (MIT)

Copyright (c) 2023 Vitalii Maslianok

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

redux

- Description: Predictable state container for JavaScript apps
- License: MIT
- Homepage: http://redux.js.org
- Version: 5.0.1

```
The MIT License (MIT)

Copyright (c) 2015-present Dan Abramov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

redux-thunk

- Description: Thunk middleware for Redux.
- License: MIT
- Homepage: https://github.com/reduxjs/redux-thunk
- Version: 3.1.0

```
The MIT License (MIT)

Copyright (c) 2015-present Dan Abramov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

remark-gfm

- Description: remark plugin to support GFM (autolink literals, strikethrough, tables, tasklists)
- License: MIT
- Homepage: remarkjs/remark-gfm
- Version: 1.0.0

```
(The MIT License)

Copyright (c) 2020 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
```

(continues on next page)

(continued from previous page)

```
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

remark-parse

- Description: remark plugin to add support for parsing markdown input
- License: MIT
- Homepage: https://remark.js.org
- Version: 10.0.2

```
(The MIT License)

Copyright (c) 2014-2020 Titus Wormer <tituswormer@gmail.com>
Copyright (c) 2011-2014, Christopher Jeffrey (https://github.com/chjj/)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

remark-rehype

- Description: remark plugin that turns markdown into HTML to support rehype
- License: MIT
- Homepage: remarkjs/remark-rehype
- Version: 10.1.0

```
(The MIT License)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

repeat-string

- Description: Repeat the given string n times. Fastest implementation for repeating a string.
- License: MIT
- Homepage: https://github.com/jonschlinkert/repeat-string
- Version: 1.6.1

```
The MIT License (MIT)

Copyright (c) 2014-2016, Jon Schlinkert.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

require-from-string

- Description: Require module from string
- License: MIT
- Homepage: floatdrop/require-from-string
- Version: 2.0.2

```
The MIT License (MIT)

Copyright (c) Vsevolod Strukchinsky <floatdrop@gmail.com> (github.com/floatdrop)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

reselect

- Description: Selectors for Redux.
- License: MIT
- Homepage: https://github.com/reduxjs/reselect
- Version: 5.3.0

```
The MIT License (MIT)

Copyright (c) 2015-2018 Reselect Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

resize-observer-polyfill

- Description: A polyfill for the Resize Observer API
- License: MIT
- Homepage: https://github.com/que-etc/resize-observer-polyfill
- Version: 1.5.1

```
The MIT License (MIT)

Copyright (c) 2016 Denis Rul

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

resolve-protobuf-schema

- Description: Read a protobuf schema from the disk, parse it and resolve all imports
- License: MIT
- Homepage: https://github.com/mafintosh/resolve-protobuf-schema
- Version: 2.1.0

```
The MIT License (MIT)

Copyright (c) 2014 Mathias Buus

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.
```

(continues on next page)

(continued from previous page)

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

ret

- Description: Tokenizes a string that represents a regular expression.
- License: MIT
- Homepage: https://github.com/fent/ret.js
- Version: 0.1.15

```
Copyright (C) 2011 by Roly Fentanes

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

rxjs

- Description: Reactive Extensions for modern JavaScript
- License: Apache-2.0
- Homepage: https://github.com/ReactiveX/RxJS
- Version: 5.5.12

```
Apache License
                           Version 2.0, January 2004
                        http://www.apache.org/licenses/

 TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

 1. Definitions.

    "License" shall mean the terms and conditions for use, reproduction,
    and distribution as defined by Sections 1 through 9 of this document.
```

(continues on next page)

(continued from previous page)

```
   "Licensor" shall mean the copyright owner or entity authorized by
   the copyright owner that is granting the License.

   "Legal Entity" shall mean the union of the acting entity and all
   other entities that control, are controlled by, or are under common
   control with that entity. For the purposes of this definition,
   "control" means (i) the power, direct or indirect, to cause the
   direction or management of such entity, whether by contract or
   otherwise, or (ii) ownership of fifty percent (50%) or more of the
   outstanding shares, or (iii) beneficial ownership of such entity.

   "You" (or "Your") shall mean an individual or Legal Entity
   exercising permissions granted by this License.

   "Source" form shall mean the preferred form for making modifications,
   including but not limited to software source code, documentation
   source, and configuration files.

   "Object" form shall mean any form resulting from mechanical
   transformation or translation of a Source form, including but
   not limited to compiled object code, generated documentation,
   and conversions to other media types.

   "Work" shall mean the work of authorship, whether in Source or
   Object form, made available under the License, as indicated by a
   copyright notice that is included in or attached to the work
   (an example is provided in the Appendix below).

   "Derivative Works" shall mean any work, whether in Source or Object
   form, that is based on (or derived from) the Work and for which the
   editorial revisions, annotations, elaborations, or other modifications
   represent, as a whole, an original work of authorship. For the purposes
   of this License, Derivative Works shall not include works that remain
   separable from, or merely link (or bind by name) to the interfaces of,
   the Work and Derivative Works thereof.

   "Contribution" shall mean any work of authorship, including
   the original version of the Work and any modifications or additions
   to that Work or Derivative Works thereof, that is intentionally
   submitted to Licensor for inclusion in the Work by the copyright owner
   or by an individual or Legal Entity authorized to submit on behalf of
   the copyright owner. For the purposes of this definition, "submitted"
   means any form of electronic, verbal, or written communication sent
   to the Licensor or its representatives, including but not limited to
   communication on electronic mailing lists, source code control systems,
   and issue tracking systems that are managed by, or on behalf of, the
   Licensor for the purpose of discussing and improving the Work, but
   excluding communication that is conspicuously marked or otherwise
   designated in writing by the copyright owner as "Not a Contribution."

   "Contributor" shall mean Licensor and any individual or Legal Entity
   on behalf of whom a Contribution has been received by Licensor and
   subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
   this License, each Contributor hereby grants to You a perpetual,
```

(continues on next page)

(continued from previous page)

```
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    copyright license to reproduce, prepare Derivative Works of,
    publicly display, publicly perform, sublicense, and distribute the
    Work and such Derivative Works in Source or Object form.

 3. Grant of Patent License. Subject to the terms and conditions of
    this License, each Contributor hereby grants to You a perpetual,
    worldwide, non-exclusive, no-charge, royalty-free, irrevocable
    (except as stated in
   [...truncated, see full license at the project homepage...]
```

sade

- Description: Smooth (CLI) operator
- License: MIT
- Homepage: lukeed/sade
- Version: 1.8.1

```
The MIT License (MIT)

Copyright (c) Luke Edwards <luke.edwards05@gmail.com> (https://lukeed.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

scheduler

- Description: Cooperative scheduler for the browser environment.
- License: MIT
- Homepage: https://reactjs.org/
- Version: 0.23.2

```
MIT License

Copyright (c) Facebook, Inc. and its affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
```

(continues on next page)

(continued from previous page)

```
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

scriptjs

- Description: Asyncronous JavaScript loader and dependency manager
- License: MIT
- Homepage: https://github.com/ded/script.js
- Version: 2.5.9

```
(The MIT License)

Copyright (c) 2011 - 2015 Dustin Diaz <dustin@dustindiaz.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

scroll-into-view-if-needed

- Description: Ponyfill for upcoming Element.scrollIntoView() APIs like scrollMode: if-needed, behavior: smooth and block: center
- License: MIT
- Homepage: https://scroll-into-view.dev
- Version: 3.1.0

```
MIT License

Copyright (c) 2023 Cody Olsen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

set-cookie-parser

- Description: Parses set-cookie headers into objects
- License: MIT
- Homepage: https://github.com/nfriedly/set-cookie-parser
- Version: 2.7.2

```
The MIT License (MIT)

Copyright (c) 2015 Nathan Friedly <nathan@nfriedly.com> (http://nfriedly.com/)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

shallowequal

- Description: Like lodash isEqualWith but for shallow equal.
- License: MIT
- Homepage: dashed/shallowequal
- Version: 1.1.0

```
MIT License

Copyright (c) 2017 Alberto Leal <mailforalberto@gmail.com> (github.com/dashed)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

slate

- Description: A completely customizable framework for building rich text editors.
- License: MIT
- Homepage: https://github.com/ianstormtaylor/slate
- Version: 0.112.0

slate-dom

- Description: Tools for building completely customizable richtext editors with React.
- License: MIT
- Homepage: https://github.com/ianstormtaylor/slate
- Version: 0.112.2

slate-react

- Description: Tools for building completely customizable richtext editors with React.
- License: MIT
- Homepage: https://github.com/ianstormtaylor/slate
- Version: 0.112.1

space-separated-tokens

- Description: Parse and stringify space separated tokens
- License: MIT
- Homepage: wooorm/space-separated-tokens
- Version: 2.0.2

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

string-format-obj

- Description: Replace tokens from a string with values of an object
- License: MIT
- Homepage: stephenplusplus/string-format-obj
- Version: 1.1.1

```
The MIT License (MIT)

Copyright (c) 2015 Stephen Sawchuk

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
```

(continues on next page)

(continued from previous page)

```
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

style-to-object

- Description: Converts inline style to object.
- License: MIT
- Homepage: https://github.com/remarkablemark/style-to-object
- Version: 0.4.4

```
The MIT License (MIT)

Copyright (c) 2017 Menglin "Mark" Xu <mark@remarkablemark.org>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

styled-components

- Description: Visual primitives for the component age. Use the best bits of ES6 and CSS to style your apps without stress
- License: MIT
- Homepage: https://styled-components.com
- Version: 5.3.11

supports-color

- Description: Detect whether a terminal supports color
- License: MIT
- Homepage: chalk/supports-color
- Version: 7.2.0

```
MIT License

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)
```

(continues on next page)

(continued from previous page)

```
Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:

The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

symbol-observable

- Description: Symbol.observable ponyfill
- License: MIT
- Homepage: blesh/symbol-observable
- Version: 1.0.1

```
The MIT License (MIT)

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)
Copyright (c) Ben Lesh <ben@benlesh.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

temporal-polyfill

- Description: A spec-compliant Temporal JavaScript polyfill in 16kb
- License: MIT
- Homepage: https://github.com/fullcalendar/temporal
- Version: 0.1.1

temporal-spec

- License: ISC
- Homepage: https://github.com/fullcalendar/temporal
- Version: 0.1.0

```
Copyright 2017, 2018, 2019, 2020 ECMA International

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

tiny-invariant

- Description: A tiny invariant function
- License: MIT
- Homepage: https://github.com/alexreardon/tiny-invariant
- Version: 1.3.1

```
MIT License

Copyright (c) 2019 Alexander Reardon

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

tiny-warning

- Description: A tiny warning function
- License: MIT
- Homepage: https://github.com/alexreardon/tiny-warning

- Version: 1.0.3

```
MIT License

Copyright (c) 2019 Alexander Reardon

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

tinycolor2

- Description: Fast Color Parsing and Manipulation
- License: MIT
- Homepage: https://github.com/bgrins/TinyColor
- Version: 1.6.0

```
Copyright (c), Brian Grinstead, http://briangrinstead.com

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

tinyqueue

- Description: The smallest and simplest JavaScript priority queue
- License: ISC
- Homepage: https://github.com/mourner/tinyqueue
- Version: 3.0.0

```
ISC License

Copyright (c) 2017, Vladimir Agafonkin

Permission to use, copy, modify, and/or distribute this software for any purpose
with or without fee is hereby granted, provided that the above copyright notice
and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF
THIS SOFTWARE.
```

tr46

- Description: An implementation of the Unicode UTS #46: Unicode IDNA Compatibility Processing
- License: MIT
- Homepage: https://github.com/jsdom/tr46
- Version: 5.1.1

```
The MIT License (MIT)

Copyright (c) Sebastian Mayr

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

trim-lines

- Description: Remove spaces and tabs around line-breaks
- License: MIT
- Homepage: wooorm/trim-lines
- Version: 3.0.1

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <mailto:tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

trough

- Description: *trough* is middleware
- License: MIT
- Homepage: wooorm/trough
- Version: 2.2.0

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
```

(continues on next page)

(continued from previous page)

```
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

tslib

- Description: Runtime library for TypeScript helper functions
- License: 0BSD
- Homepage: https://www.typescriptlang.org/
- Version: 2.8.1

```
Copyright (c) Microsoft Corporation.

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH
REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT,
INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR
OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

undici-types

- Description: A stand-alone types package for Undici
- License: MIT
- Homepage: https://undici.nodejs.org
- Version: 8.9.0

```
MIT License

Copyright (c) Matteo Collina and Undici contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

unified

- Description: Interface for parsing, inspecting, transforming, and serializing content through syntax trees
- License: MIT
- Homepage: https://unifiedjs.com
- Version: 10.1.2

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

unist-util-generated

- Description: unist utility to check if a node is generated
- License: MIT
- Homepage: syntax-tree/unist-util-generated
- Version: 2.0.1

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
```

(continues on next page)

(continued from previous page)

```
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

unist-util-is

- Description: unist utility to check if a node passes a test
- License: MIT
- Homepage: syntax-tree/unist-util-is
- Version: 4.1.0

```
(The MIT license)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

unist-util-position

- Description: unist utility to get the position of a node
- License: MIT
- Homepage: syntax-tree/unist-util-position
- Version: 4.0.4

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

unist-util-stringify-position

- Description: unist utility to serialize a node, position, or point as a human readable location
- License: MIT
- Homepage: syntax-tree/unist-util-stringify-position
- Version: 3.0.3

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

unist-util-visit

- Description: unist utility to visit nodes
- License: MIT
- Homepage: syntax-tree/unist-util-visit
- Version: 4.1.2

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
```

(continues on next page)

(continued from previous page)

```
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

unist-util-visit-parents

- Description: unist utility to recursively walk over nodes, with ancestral information
- License: MIT
- Homepage: syntax-tree/unist-util-visit-parents
- Version: 5.1.3

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

uplot

- Description: A small, fast chart for time series, lines, areas, ohlc & bars
- License: MIT
- Homepage: https://github.com/leeoniya/uPlot#readme
- Version: 1.6.32

```
The MIT License (MIT)

Copyright (c) 2022 Leon Sorokin

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

uplot-react

- Description: React wrapper for uPlot that allows you to work with charts declaratively inside your favorite framework
- License: MIT
- Homepage: https://github.com/skalinichev/uplot-wrappers#readme
- Version: 1.2.4

```
Copyright (c) 2021-present Sergey Kalinichev

MIT License

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

use-deep-compare

- Description: React hooks, except using deep comparison on the inputs, not reference equality
- License: MIT
- Homepage: https://github.com/sandiiarov/use-deep-compare
- Version: 1.3.0

```
The MIT License (MIT)
Copyright (c) 2019 Alex Sandiiarov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

use-sync-external-store

- Description: Backwards compatible shim for React's useSyncExternalStore. Works with any React that supports hooks.
- License: MIT
- Homepage: https://github.com/facebook/react
- Version: 1.6.0

```
MIT License

Copyright (c) Meta Platforms, Inc. and affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

(continues on next page)

(continued from previous page)

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

use-typed-event-listener

- Description: React Hook for DOM event listeners with TypeScript supported
- License: MIT
- Homepage: https://github.com/foray1010/use-typed-event-listener
- Version: 3.0.0

```
MIT License

Copyright (c) 2019 Alex Young

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

uuid-v4

- Description: A simple v4 UUID generator
- License: See LICENSE file
- Version: 0.1.0

uvu

- Description: uvu is an extremely fast and lightweight test runner for Node.js and the browser
- License: MIT
- Homepage: lukeed/uvu
- Version: 0.5.6

```
The MIT License (MIT)

Copyright (c) Luke Edwards <luke.edwards05@gmail.com> (lukeed.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
```

(continues on next page)

(continued from previous page)

```
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

vfile

- Description: Virtual file format for text processing
- License: MIT
- Homepage: vfile/vfile
- Version: 5.3.7

```
(The MIT License)

Copyright (c) 2015 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

vfile-message

- Description: vfile utility to create a virtual message
- License: MIT
- Homepage: vfile/vfile-message
- Version: 3.1.4

```
(The MIT License)
```

(continues on next page)

(continued from previous page)

```
Copyright (c) 2017 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

warning

- Description: A mirror of Facebook's Warning
- License: MIT
- Homepage: https://github.com/BerkeleyTrue/warning
- Version: 4.0.3

```
MIT License

Copyright (c) 2013-present, Facebook, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

webidl-conversions

- Description: Implements the WebIDL algorithms for converting to and from JavaScript values
- License: BSD-2-Clause
- Homepage: jsdom/webidl-conversions
- Version: 7.0.0

```
# The BSD 2-Clause License

Copyright (c) 2014, Domenic Denicola
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,␣
↪are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this␣
↪list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this␣
↪list of conditions and the following disclaimer in the documentation and/or␣
↪other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND␣
↪ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED␣
↪WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE␣
↪DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR␣
↪ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES␣
↪(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;␣
↪LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON␣
↪ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT␣
↪(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS␣
↪SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

whatwg-fetch

- Description: A window.fetch polyfill.
- License: MIT
- Homepage: github/fetch
- Version: 3.6.20

```
Copyright (c) 2014-2023 GitHub, Inc.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
```

(continues on next page)

(continued from previous page)

```
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

whatwg-url

- Description: An implementation of the WHATWG URL Standard's URL API and parsing machinery
- License: MIT
- Homepage: jsdom/whatwg-url
- Version: 14.2.0

```
The MIT License (MIT)

Copyright (c) Sebastian Mayr

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

xmlserializer

- Description: Serializes a document to XML/XHTML
- License: MIT
- Homepage: http://github.com/cburgmer/xmlserializer
- Version: 0.6.1

```
The MIT License (MIT)

Copyright (c) 2014 Christoph Burgmer and contributors

Permission is hereby granted, free of charge, to any person obtaining a copy of␣
↪this software and associated documentation files (the "Software"), to deal in␣
↪the Software without restriction, including without limitation the rights to use,
↪ copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the␣
↪Software, and to permit persons to whom the Software is furnished to do so,␣
↪subject to the following conditions:
```

(continues on next page)

(continued from previous page)

```
The above copyright notice and this permission notice shall be included in all␣
↪copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
↪ INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A␣
↪PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR␣
↪COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER␣
↪IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN␣
↪CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

xpath

- Description: DOM 3 XPath implemention and helper for node.js.
- License: MIT
- Homepage: https://github.com/goto100/xpath
- Version: 0.0.32

```
MIT License

Copyright (c) 2018 Cameron McCormack

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

zwitch

- Description: Handle values based on a property
- License: MIT
- Homepage: wooorm/zwitch
- Version: 1.0.5

```
(The MIT License)

Copyright (c) 2016 Titus Wormer <tituswormer@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
```

(continues on next page)

(continued from previous page)

```
'Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

CHAPTER

FOURTEEN

ARCHIVES

14.1 Migrate from TrackMe V1 to TrackMe V2

Warning

Archived content - Migration from TrackMe V1 is no longer supported

This page is kept for knowledge and reference purposes only. Migrating from TrackMe V1 to TrackMe V2 is no longer actively supported. If you are still running TrackMe V1, please contact us at contact@trackme-solutions.com for guidance on upgrading to the latest version of TrackMe.

14.1.1 Introduction

The purpose of this page is to provide a path for existing users of TrackMe V1 to migrate to TrackMe V2, as easily as possible.

Hint

About configuration losses, interruption of service and efforts

- We appreciate many fellow TrackMe V1 users invested time and efforts in fine tuning the many various configuration items in TrackMe V1
- In many cases, TrackMe became a central tool, providing deep visibility and efficient monitoring capabilities
- Although the migration requires manual operations, there is a limited amount of efforts needed for TrackMe V2 to be operation and replacing your previous deployment
- However, during this phase, existing monitoring operations will stop, until your new TrackMe V2 deployment is operational
- Review this migration manual, performing planning and backups, and upgrade peacefully
- If you missed something, you can always retrieve these information from summary events, TrackMe backups or file backups

Top information

- You **cannot** run both TrackMe V1 and V2 on the same Splunk instance / SHC
- When the application is upgraded, it replaces V1 entirely, all configuration, settings are permanently lost (apart from TrackMe and third party backups)
- **Prior to the migration** you must follow this documentation if you wish to migrate existing settings and rules

- **If for some reasons** you didn't read this documentation **before** (sic), you still can recover information from TrackMe's summary events and backups
- The documentation cannot cover every subtitle of potential customizations of TrackMe, you may need to adapt some of these to your own context

TrackMe V2 is deeply different from TrackMe V1 in all means, TrackMe V2 notably introduces a brand new key concept of Virtual Tenants which slightly influences the migration path.

Due to the highly flexible nature of TrackMe V2, there are various design questions that could be taken into account, the present document describes the general migration path and options.

Migrating or restarting from scratch?

Depending on your context, both approaches can make sense, in short:

- Starting from scratch has the advantage of starting fully clean, without taking any risks of leverage obsolete rules or settings which you could improve today
- You can directly leverage the many powerful advantages of TrackMe V2, including designing Virtual Tenants for instance, for a better experience and value
- On the opposite, you may have spent time and efforts in qualifying topics like entities priorities, it is fully understandable that you would want to migrate these settings

The migration path is in reality doing a little bit of both Worlds, you can decide at any step what makes the most sense in your context, and selectively import settings from your TrackMe V1 deployment.

14.1.2 Planning the migration

Not all users are equal in term when it comes to the level of access, Splunk Enterprise users have full backend accesses, while users in Splunk Cloud have much more limitations.

In short, the migration from an existing deployment of TrackMe V1 can be summarised as:

- Performing backups prior to the migration (KVstore backups and file-system level backups)
- Auditing local knowledge objects and customisations
- Allowing a maintenance window for the operation, if you are relying on TrackMe V1 in a Production context, no alerts will be available as long as the migration and configuration is not terminated
- Performing the application upgrade from V1 to V2
- Creating at least one TrackMe V2 Virtual Tenant to replace the migrated TrackMe V1 deployment
- Creating Hybrid Trackers, discover and perform the basis setup
- Import target key configuration items, such as entities priority or custom threshold policies
- Creating additional Knowledge Objects, such as TrackMe alerts or Elastic Sources, etc

14.1.3 Prior to the migration: Performing backups and exports

file-system level backup (For Splunk Enterprise)

Splunk Cloud users

- If you are a Splunk Cloud user, you can ignore this step
- Performing backup here is only for Splunk on-premise users, and a simple additional safety before proceeding to an upgrade as a generic recommendation

It is recommended to perform a full backup of the TrackMe application root directory prior to the upgrade.

In a basic way, this means creating a compressed archive of the existing root directory:

```
cd /opt/splunk/etc/apps
tar -czf <target_path>/trackme_v1_backup_DDMMYYYY.tgz trackme
```

TrackMe KVstores backup

TrackMe backups

- While you wouldn't be able to restore directly the V1 backups into TrackMe V2, these provides means to retrieve and access any useful settings of TrackMe
- For instance, if you later on realise that you have created some specific content such as Elastic Sources, block lists, tags and so forth, these information can be retrieved from the JSON files contained in the backup archive

TrackMe V1 has a concept of a backup job which targets its own KVstore collections, this job is by default executed once per day and stores backup archives locally on the search head.

You can ensure that this process is functional, and run an immediate backup job prior to the migration.

In TrackMe V1, this scheduled report is named:

`TrackMe - Backup KVstore collections and purge older backup files.`

You can manually run this job to ensure a backup will be taken prior to the upgrade.

Auditing TrackMe V1 objects

TrackMe V1 has 3 main types of components:

- Data sources tracking
- Data hosts tracking
- Metric hosts tracking

As part of its normal activity, TrackMe V1 generates summary events which contain the essential entity information, such as the priority of their monitor status.

There are various additional TrackMe V1 features which you may have used, mostly:

- Elastic Sources, either shared or dedicated
- Allow and block lists, which are specific collections for each component
- Data sampling custom rules
- Tag policies
- TrackMe alerts

Use the following search to detect which KVstore collections have actual records, you can ignore empty collections:

Notes:

- run this search **prior** to the migration, if TrackMe was upgraded already, this is too late so you can ignore these steps
- ignore empty collections
- some of the KVstore collections have generated and/or dynamic content, this is not relevant for the purpose of the migration

- for each of the collection resulting from the following search, you can manually copy the collection in a CSV file for the ease of re-creation and/or re-applying some of steps

```
| rest splunk_server=local /services/server/introspection/kvstore/collectionstats
| mvexpand data
| spath input=data
| rex field=ns "(?<app>.*)\.(?<collection>.*)"
| stats first(count) as count by app, collection
| where app="trackme" AND count>0
| where NOT collection in ("kv_trackme_alerts_ack", "kv_trackme_audit_changes",
↪"kv_trackme_backup_archives_info", "kv_trackme_data_sampling",
↪"kv_trackme_maintenance_mode", "kv_trackme_objects_summary",
↪"kv_trackme_summary_investigator_volume_outliers")
| rex field=collection "^kv_(?<transforms_definition>.*)"
| fields app collection transforms_definition count
```

TrackMe Collections CSV export

BEFORE TrackMe is upgraded, the easiest and simplest procedure is to manually export the principal collections into CSV lookups, which will be used to restore main configuration items.

These steps need to be taken prior to the migration, once upgraded the original TrackMe V1 collections will not exist anymore and therefore their content wilk has been deleted automatically.

Allow lists & block lists

- TrackMe V2 handles concepts of allow lists & block lists in a fundamentally different fashion, as of now, these cannot be migrated and need to be replayed if necessary

Splunk Cloud

- Splunk Cloud will normally preserve CSV lookup files created prior to the migration (unless for some technical reasons, the app is uninstalled at some points)
- As an additional measure of safety, you can as well download the CSV lookups out of Splunk when performing the following steps

Hint

Empty collections:

- If any of the following collection is empty, you can safety ignore its export and associated restoration

Proceed as follows:

Export Data Sources collection

```
| inputlookup trackme_data_source_monitoring | eval keyid=_key
| eval object=coalesce(data_name, data_host, metric_host)
| outputlookup trackme_v1_trackme_data_source_monitoring.csv
```

Export Data Hosts collection

```
| inputlookup trackme_host_monitoring | eval keyid=_key
| eval object=coalesce(data_name, data_host, metric_host)
| eval object="key:host|" . lower(object)
| outputlookup trackme_v1_trackme_host_monitoring.csv
```

Export Metric Hosts collection

```
| inputlookup trackme_metric_host_monitoring | eval keyid=_key
| eval object=coalesce(data_name, data_host, metric_host)
| eval object="key:host|" . lower(object)
| outputlookup trackme_v1_trackme_metric_host_monitoring.csv
```

Export Shared Elastic Sources collection

Elastic Shared

- In TrackMe V1, Elastic Shared sources were sharing the time range period with the scheduled period of the Elastic tracker
- In TrackMe V2, the period is properly retrieved and applied by the backend responsible from spawning the searches, the following sets to a default of -4h/+4h to allow the transition
- Review the Elastic records, adapt the earliest/latest if necessary, and make sure the index and sourcetype fields are set to useful values (which do not have to be real indexes/sourcetypes)
- Several fields have been renamed, make sure to properly execute the following export statement

```
| inputlookup trackme_elastic_sources
| rename data_name as object, elastic_data_index as elastic_index,␣
↪elastic_data_sourcetype as elastic_sourcetype
| eval earliest="-4h", latest="+4h"
| outputlookup trackme_v1_trackme_elastic_sources_shared.csv
```

Export Dedicated Elastic Sources collection

Elastic Dedicated

- Elastic Dedicated cannot be migrated from V1 to V2, the trackers have to be re-created from TrackMe V2
- However, you can export the V1 collection records to support the re-creation in the further steps

```
| inputlookup trackme_elastic_sources_dedicated | eval keyid=_key
| outputlookup trackme_v1_trackme_elastic_sources_dedicated.csv
```

Export Data Sampling custom rules

```
| inputlookup trackme_data_sampling_custom_models | eval keyid=_key
| outputlookup trackme_v1_trackme_data_sampling_custom_models.csv
```

Export Lagging policies for Data Sources & Hosts

```
| inputlookup trackme_custom_lagging_definition | eval keyid=_key
| outputlookup trackme_v1_trackme_custom_lagging_definition.csv
```

Export Lagging policies for Metric Hosts

```
| inputlookup trackme_metric_lagging_definition | eval keyid=_key
| outputlookup trackme_v1_trackme_metric_lagging_definition.csv
```

Export tag policies

```
| inputlookup trackme_tags_policies | eval keyid=_key
| outputlookup trackme_v1_trackme_tags_policies.csv
```

Export Logical groups

```
| inputlookup trackme_logical_group | eval keyid=_key
| eval object_group_members=lower(mvjoin(object_group_members, "|"))
| outputlookup trackme_v1_trackme_logical_group.csv
```

14.1.4 Upgrading to TrackMe V2

Upgrade the application

Once you are ready to migrate, perform the application upgrade as usually:

- If running Splunk Enterprise in a standalone instance, you can either upgrade through Splunk Web, or manually by extracting the compressed tarball archive, in both cases, make sure to restart Splunk post-upgrade
- If running Splunk Enterprise in a Search Head Cluster, extract TrackMe V2 to the SHC deployer, publish the bundle and wait for the Rolling Restart to complete
- If you are a Splunk Cloud Victoria user, you can upgrade to TrackMe V2 through Splunk Cloud self-services
- If you are a Splunk Cloud user on any other flavour, request the application upgrade to Cloud Operations

What will happen once TrackMe was upgraded?

- TrackMe V2 will be mostly fully virgin, the Virtual Tenants wizard will propose you to guide you through the creation of your first tenant
- Previously defined KVstore do not exist anymore, their content is now lost, this will be the case for almost all KVstore collections
- There are no more entities monitored yet
- There are no more active alerts

Create your first Virtual Tenant

Designing Virtual Tenants

Foundation Edition trial

- The Foundation Edition trial is restricted to three Virtual Tenants
- The Full Registered Edition allows to create an unlimited number of tenants

A new key concept in TrackMe V2 consists in the Virtual Tenants feature, depending on your context, Virtual Tenants allow you to:

- Dedicate entire virtual instances of TrackMe depending on your needs (think about departments, technologies, companies, teams, etc…)
- Scale by scoping Virtual Tenants to reasonable volumes of entities (customers rarely have 10,000 or 20,000 forwarders without any form of convention, context or regions!)
- And many more

For the purposes of the migration, this present documentation assumes the creation of a single Virtual Tenant which replaces your previous existing TrackMe V1 deployment.

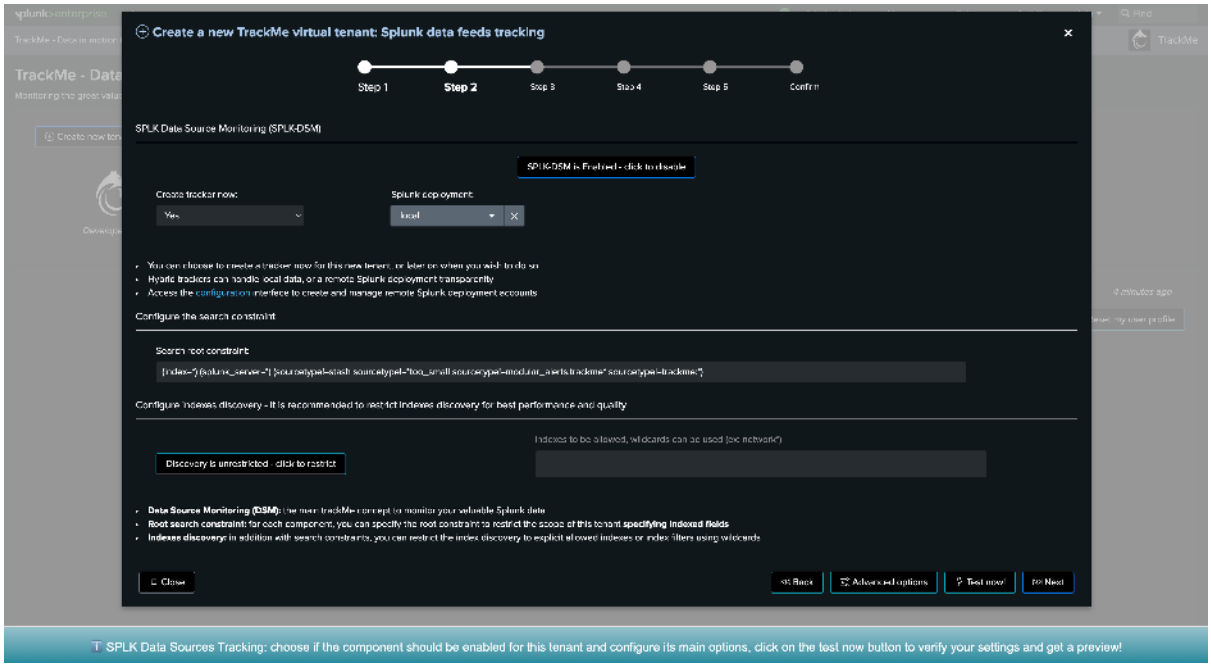
However, keep in mind that Virtual Tenants provide a key concept to address many various use cases with minimal levels of design.

For more details about the creation of Virtual Tenants for Splunk feeds tracking, see *Creating a tenant*

Use the wizard to create your new Virtual Tenant

Following the wizard and creation your first Virtual Tenant, you can define various options at the wizard phase, you can for instance:

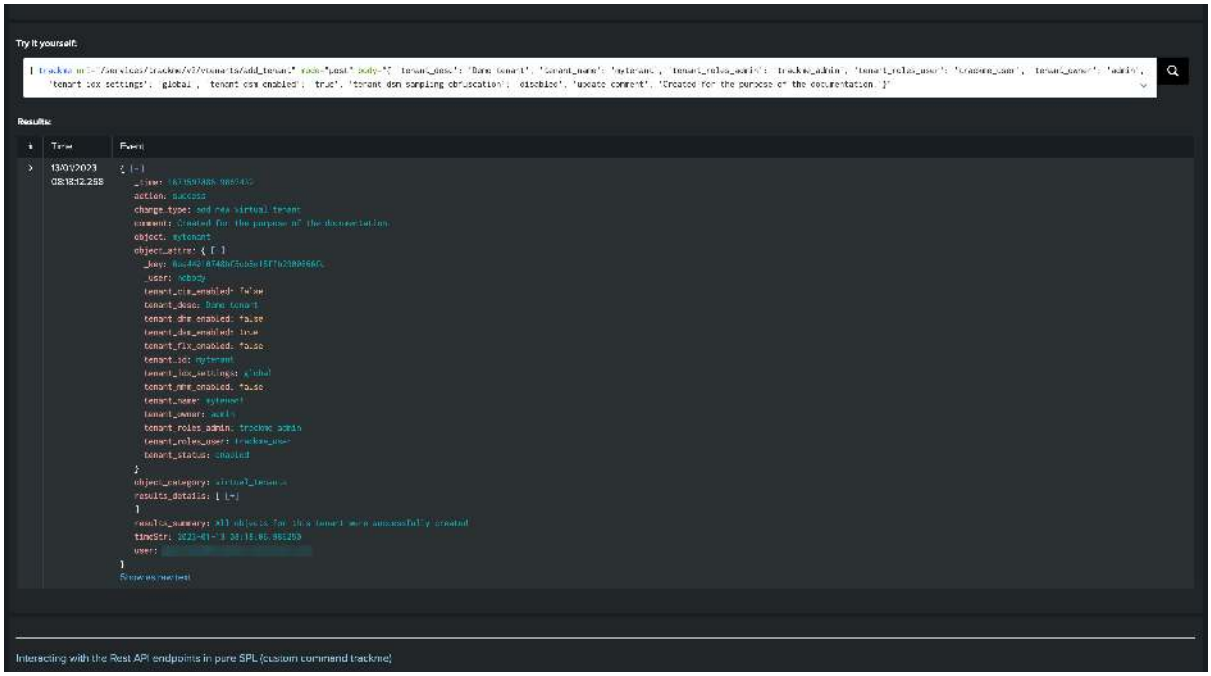
- Create a tenant with no trackers at all, then create your own Hybrid Trackers after through the wizard
- Create a tenant with trackers to be created at the stage of the Virtual tenant creation
- Choose settings such as target for indexers, ownership and RBAC



Create a new Virtual Tenant in command line, with no trackers

In TrackMe, go in a Search, and run the following command:

```
| trackme url="/services/trackme/v2/vtenants/add_tenant" mode="post" body="{\
↪"tenant_desc\":\"Splunk Feeds tracking\",\"tenant_name\":\"splk-feeds-tracking\",
↪\"tenant_roles_admin\":[\"trackme_admin\"],\"tenant_roles_user\":[\"trackme_user\
↪"],\"tenant_owner\":\"admin\",\"tenant_idx_settings\":\"global\",\
↪"tenant_dsm_enabled\":true,\"tenant_dsm_sampling_obfuscation\":\"disabled\",\
↪"tenant_dhm_enabled\":true,\"tenant_dhm_alerting_policy\":\"track_per_host\",\
↪"tenant_mhm_enabled\":true}"
```



There are many more options available through the REST API, for more information see: *REST API reference*

Create Hybrid Trackers

BEFORE restoring any parameters from TrackMe V1, ensure that you have create relevant Hybrid Trackers and started to discover and maintain entities in your brand new TrackMe V2 deployment.

For more information about how to create Hybrid Trackers, see: *Hybrid trackers*

Restore priorities, monitored state and other main settings

In most of the use cases, there are two first pieces of information that you will want to restore regarding previously known entities.

Component splk-dsm (previously called Data Source Monitoring)

For the purpose of the documentation, and if you have not exported the collection prior to the upgrade, you can still use the summary data to generate the latest known information:

Ignore this if you performed exports prior to the migration

```
index=trackme_summary sourcetype=stash source="current_state_tracking:data_source"
| stats latest(priority) as priority, latest(data_monitored_state) as␣
↪monitored_state by key, object
| outputlookup trackme_v1_trackme_data_source_monitoring.csv
```

Apply:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_dsm_tenant_splk-feeds-tracking | eval keyid=_key
| lookup trackme_v1_trackme_data_source_monitoring.csv object OUTPUT␣
↪data_max_lag_allowed as previous_data_max_lag_allowed, priority as␣
↪previous_priority, data_monitored_state as previous_monitored_state,␣
↪data_monitoring_wdays as previous_data_monitoring_wdays,␣
↪data_override_lagging_class as previous_data_override_lagging_class,␣
↪min_dcount_host as previous_min_dcount_host
| eval data_max_delay_allowed=coalesce(previous_data_max_lag_allowed,␣
↪data_max_delay_allowed)
| eval data_max_lag_allowed=coalesce(previous_data_max_lag_allowed,␣
↪data_max_lag_allowed)
| eval priority=coalesce(previous_priority, priority),␣
↪monitored_state=coalesce(previous_monitored_state, monitored_state)
| eval data_monitoring_wdays=coalesce(previous_data_monitoring_wdays,␣
↪data_monitoring_wdays),␣
↪data_override_lagging_class=coalesce(previous_data_override_lagging_class,␣
↪data_override_lagging_class), min_dcount_host=coalesce(previous_min_dcount_host,␣
↪min_dcount_host)
| outputlookup trackme_dsm_tenant_splk-feeds-tracking append=t key_field=keyid
```

Component splk-dhm (previously called Data Host Monitoring)

For the purpose of the documentation, and if you have not exported the collection prior to the upgrade, you can still use the summary data to generate the latest known information:

Ignore this if you performed exports prior to the migration

```
index=trackme_summary sourcetype=stash source="current_state_tracking:data_host"
| stats latest(priority) as priority, latest(data_monitored_state) as␣
↪monitored_state by key, object
| eval object=lower(object)
| eval object="key:host|" . lower(object)
| outputlookup trackme_v1_trackme_host_monitoring.csv
```

Apply:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_dhm_tenant_splk-feeds-tracking | eval keyid=_key
| lookup trackme_v1_trackme_host_monitoring.csv object OUTPUT data_max_lag_allowed␣
↪as previous_data_max_lag_allowed, priority as previous_priority,␣
↪data_monitored_state as previous_monitored_state, data_monitoring_wdays as␣
↪previous_data_monitoring_wdays, data_override_lagging_class as␣
↪previous_data_override_lagging_class, data_host_alerting_policy as␣
↪previous_splk_dhm_alerting_policy
| eval data_max_delay_allowed=coalesce(previous_data_max_lag_allowed,␣
↪data_max_delay_allowed)
| eval data_max_lag_allowed=coalesce(previous_data_max_lag_allowed,␣
↪data_max_lag_allowed)
| eval priority=coalesce(previous_priority, priority),␣
↪monitored_state=coalesce(previous_monitored_state, monitored_state)
| eval data_monitoring_wdays=coalesce(previous_data_monitoring_wdays,␣
```

(continues on next page)

(continued from previous page)

```
↪data_monitoring_wdays),␣
↪data_override_lagging_class=coalesce(previous_data_override_lagging_class,␣
↪data_override_lagging_class)
| eval splk_dhm_alerting_policy=coalesce(previous_splk_dhm_alerting_policy,␣
↪splk_dhm_alerting_policy)
| outputlookup trackme_dhm_tenant_splk-feeds-tracking append=t key_field=keyid
```

Push expected hosts

- You can push expected hosts from a third party lookup (such as a CMDB or the export of TrackMe V1)
- This way, if you have hosts which are not active yet, TrackMe will automatically take these into account, and update their status as soon as they are active
- See: *Expected sources & hosts — in depth*

Component splk-mhm (previously called Metric Host Monitoring)

For the purpose of the documentation, and if you have not exported the collection prior to the upgrade, you can still use the summary data to generate the latest known information:

Ignore this if you performed exports prior to the migration

```
index=trackme_summary sourcetype=stash source="current_state_tracking:metric_host"
| stats latest(priority) as priority, latest(metric_monitored_state) as␣
↪monitored_state by key, object
| eval object="key:host|" . lower(object)
| outputlookup trackme_v1_trackme_metric_host_monitoring.csv
```

Apply:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_mhm_tenant_splk-feeds-tracking | eval keyid=_key
| lookup trackme_v1_trackme_metric_host_monitoring.csv object OUTPUT priority as␣
↪previous_priority, metric_monitored_state as previous_monitored_state
| eval priority=coalesce(previous_priority, priority),␣
↪monitored_state=coalesce(previous_monitored_state, monitored_state)
| outputlookup trackme_mhm_tenant_splk-feeds-tracking append=t key_field=keyid
```

Restore lagging policies for splk-dsm/splk-dhm

If you had created lagging policies in TrackMe V1 for splk-dsm/splk-dhm, these records were stored in the following collection:

- collection: kv_trackme_custom_lagging_definition, transforms: trackme_custom_lagging_definition

You can restore the policies from an export made previously:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created
- TrackMe V2 has an additional concept in the lagging policies where delay and lag are stored independently
- For the purpose of the migration, we will apply the V1 value in both cases
- You can later modify these values if needed in TrackMe V2

Apply:

```
| inputlookup trackme_v1_trackme_custom_lagging_definition.csv
| eval value_delay=value, value_lag=value
| outputlookup trackme_common_lagging_classes_tenant_splk-feeds-tracking
```

Restore lagging policies for splk-mhm

If you had created lagging policies in TrackMe V1 for splk-mhm, these records were stored in the following collection:

- collection: kv_trackme_custom_lagging_definition, transforms: trackme_custom_lagging_definition

You can restore the policies from an export made previously:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

Apply:

```
| inputlookup trackme_v1_trackme_metric_lagging_definition.csv
| outputlookup trackme_mhm_lagging_classes_tenant_splk-feeds-tracking
```

Restore logical groups association

If you had created logical groups association in TrackMe V1 for splk-dhm/splk-mhm, these records were stored in the following collection:

- collection: kv_trackme_logical_group, transforms: trackme_logical_group

You can restore the policies from an export made previously:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created
- Make sure you carefully exported and respect the SPL syntax, the object_group_members field is expected as a pipe separated field form the export, and transformed back to an mvfield

Apply:

```
| inputlookup trackme_v1_trackme_logical_group.csv
| makemv delim="|" object_group_members
| outputlookup trackme_common_logical_group_tenant_splk-feeds-tracking
```

Restore tag policies

If you had created tag policies in TrackMe V1, these records were stored in the following collection:

- collection: kv_trackme_tags_policies, transforms: trackme_tags_policies

You can restore the policies from an export made previously:

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_v1_trackme_tags_policies.csv
| outputlookup trackme_common_tag_policies_tenant_splk-feeds-tracking
```

Restore Shared Elastic Sources

You can restore Elastic Sources quite easily, if you had created any, their records were stored in the following collection:

- collection: kv_trackme_elastic_sources, transforms: trackme_elastic_sources

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_v1_trackme_elastic_sources_shared.csv
| outputlookup trackme_dsm_elastic_shared_tenant_splk-feeds-tracking
```

Re-create Dedicated Shared Elastic Sources

TrackMe V1 dedicated Elastic Trackers cannot be re-created by simply exporting / importing the KVstore collections.

Actual knowledge objects need to be properly created, and their syntax and behaviours fundamentally changed in TrackMe V2.

As well, in various use cases, Elastic Dedicated trackers can advantageously be replaced by Hybrid Trackers which are much more scalable, and allow to manage from a few to many entities resulting from the search logic.

For more information about how to create Hybrid Trackers, see: *Hybrid trackers*

Dedicated Trackers

- Rely on the previously exported KVstore collection to review the context of Trackers that had been created in TrackMe V1
- Use the Elastic tracker wizard as needed to re-create brand new entities from TrackMe V2

Restore Data Sampling custom rules

If you had created tag policies in TrackMe V1, these records were stored in the following collection:

- collection: kv_trackme_data_sampling_custom_models, transforms: trackme_data_sampling_custom_models

notes:

- replace `splk-feeds-tracking` with the identifier of the tenant previously created

```
| inputlookup trackme_v1_trackme_data_sampling_custom_models.csv
| outputlookup trackme_dsm_data_sampling_custom_models_tenant_splk-feeds-tracking
```

Re-create TrackMe alerts

You **cannot** migrate as such TrackMe V1 alerts, nor modify the existing alerts to match TrackMe V2 requirements.

You must disable these alerts in the first phase, then create new alerts up to your needs in TrackMe V2, and finally later on, delete the TrackMe V1 alerts once you are up and running.

To identity existing active alerts:

```
| rest splunk_server=local /servicesNS/nobody/trackme/saved/searches | search␣
↪eai:acl.app="trackme" alert.track=1
| fields title, cron_schedule, schedule_window, alert.suppress.fields, alert.
↪suppress.period, disabled, next_scheduled_time, id, actions
```

For more details about the creation of TrackMe alerts, see *Stateful alerting*

End of migration

You are now done and should have a fully operational fancy TrackMe V2 deployment, well done!

Review TrackMe V2 carefully, for more information about the massively enhanced troubleshooting and administration experience in TrackNe V2:

- See *Troubleshooting TrackMe*

- See *Managing tenants*
- See *Managing tenants*

You can now get some rest and enjoy well deserved pizza, with or without pepperoni depending on your preferences.