CERTYX



AUTONOMOUS CERTIFICATE LIFECYCLE MANAGEMENT

SIEM integration

Security journal export to Splunk, Cribl, IBM QRadar, ArcSight, Microsoft Sentinel and webhooks

LANGUAGE	REVISION	DATE
English	8550060f	1 November 2026

Contents

1 What CERTYX exports 5
1.1 Pipeline 5
1.2 Transports 5
1.3 Formats 6
1.4 Schema profiles 6
2 Allowed combinations 7
2.1 Destinations of the first release 7
3 Quick start 8
4 Exported events 8
5 Before you start 9
6 Adding a destination in CERTYX 10
7 TLS and trusted roots 11
7.1 Client certificate (mTLS) 11
7.2 Revocation check 12
8 Signed HTTPS webhook 12
8.1 Receiver side 12
8.2 Verifying the signature 12
8.3 CERTYX side 14
9 Splunk HTTP Event Collector 14
9.1 Choosing the Splunk path 14
9.2 Receiver side (Splunk) 15
9.3 CERTYX side 16
9.4 props.conf 17
9.5 Splunk CIM 17
10 Syslog over TLS 18
10.1 RFC 5424 header of the canonical formats 18
10.2 Framing 19
10.3 CERTYX side (all syslog formats) 19
10.4 Generic syslog receiver (rsyslog, syslog-ng) 19
11 Cribl Stream 20
11.1 Choosing the Cribl source 20
11.2 Receiver side (Cribl syslog source) 20
11.3 Receiver side (Cribl Splunk HEC source) 20
11.4 CERTYX side 21
12 Splunk TCP-SSL input 21
12.1 Receiver side (Splunk) 21
12.2 CERTYX side 21
13 IBM QRadar 22
13.1 Choosing the LEEF format 22
13.2 Receiver side (QRadar) 22
13.3 CERTYX side 23
14 ArcSight 23
14.1 Choosing the CEF format 23
14.2 Receiver side (ArcSight SmartConnector) 23
14.3 CERTYX side 24

15 Elastic (ECS) 24
16 OCSF consumers and Amazon Security Lake 25
17 Microsoft Sentinel 25
17.1 What the table receives 25
17.2 Receiver side (Azure) 25
17.3 CERTYX side 27
18 OpenTelemetry Logs (OTLP/HTTP) 28
18.1 What the receiver gets 28
18.2 Receiver side (OpenTelemetry Collector) 29
18.3 Grafana Alloy and Loki 29
18.4 Other OTLP backends 30
18.5 CERTYX side 30
19 Connection check and test event 30
20 Delivery, retries and backlog 31
20.1 Rejected events 31
20.2 Limits 32
21 Resend from a sequence or time 32
22 Deduplication by event_id 32
23 Send only: less data to the SIEM 33
23.1 How the filters combine 34
24 Event shaping: size and layout 34
25 Field mapping 35
26 Enrichment fields 36
27 Several organizations in one SIEM 37
28 File archives 39
28.1 Setting it up 39
28.2 Files 39
28.3 Download 40
28.4 Delivery, restarts and disk space 40
29 Permissions 40
30 Troubleshooting 40
30.1 Incidents 41
30.2 Check and test results 41
30.3 Receiver shows fragments or binary data 42
30.4 Splunk HEC codes 42
30.5 Delivery status on the destination card 43
30.6 Form and permission messages 43
31 Sample events 44
31.1 HTTPS webhook 44
31.2 Splunk HEC 45
31.3 Syslog over TLS 46
31.4 Microsoft Sentinel 48
31.5 OpenTelemetry Logs 48
31.6 File archive 49
32 Field reference 50
32.1 Canonical event 50
32.2 JSON record 51

32.3 CEF fields (first release) 52
32.4 LEEF fields (first release) 52
32.5 Sentinel columns 53
33 Event-code catalogue 53
33.1 Chosen by outcome 54
33.2 Journal actions 54

CERTYX exports its security journal continuously to the SIEM you already run. This guide explains what is exported, how format, schema profile and transport are chosen, and how to configure both sides — the receiving system and CERTYX — step by step for Splunk, Cribl Stream, IBM QRadar, ArcSight, Elastic, OCSF consumers, Microsoft Sentinel, OpenTelemetry backends and your own collector.

1 What CERTYX exports

Every destination receives the same events from the security journal. A destination is defined by three separate choices:

- **Transport** — how the events travel: signed HTTPS webhook, Splunk HTTP Event Collector, syslog over TLS, the Azure Monitor Logs Ingestion API (Microsoft Sentinel), OpenTelemetry Logs over OTLP/HTTP or daily files in a server-side archive.
- **Format** — how one event is encoded: JSON, NDJSON, RFC 5424 syslog, CEF, LEEF 1.0, LEEF 2.0, Generic KV, CSV, the Sentinel table row or an OTLP logs request (protobuf or OTLP/JSON).
- **Schema profile** — which field names a JSON document uses: the CERTYX canonical event (`native`), Elastic Common Schema (`ecs`), OCSF (`ocsf`) or the canonical event with Splunk CIM indexed fields (`splunk-cim`).

Only the combinations in Allowed combinations can be saved. Every destination is available to the organization administrator in the **SIEM export** view and to the platform operator in **Security event journal → SIEM export**. Each destination keeps its own position in the journal: delivery is continuous and at least once, and nothing is dropped after a failure.

Encrypted transport only. CERTYX always uses TLS and always validates the receiver's certificate, against the operating system's trust store or exactly the root certificates you paste for the destination, and checks revocation. Plain TCP, UDP syslog and unvalidated TLS are not offered. OpenTelemetry Logs travel over OTLP/HTTP with TLS; OTLP over gRPC is not offered.

1.1 Pipeline

1. **Journal.** Every security event is written to the hash-chained security journal in the platform database. The journal is the durable queue of every destination.
2. **Canonical event.** At export time each journal entry is mapped to one canonical event, schema version `certyx.event/1.0`, together with the export context (installation ID, product version, environment, host) and the entry's enrichment (trace ID, source port, server address, role, session correlation ID — see Enrichment fields). The hashed journal record itself is never changed.
3. **Redaction.** Recorded changes pass through a redaction step that removes every property whose name indicates a password, secret, token, key, cookie or credential, at any depth; `changes_redacted` is `true` when something was removed.
4. **Filters.** Health probes, outcomes, minimum severity, categories and event codes decide whether the destination receives the event (see Send only). File archives keep every event.
5. **Field mapping.** The destination's own rules rename, drop, keep or add fields of the canonical event, once, before any format writes it (Field mapping).
6. **Serializer.** The format and schema profile turn the event into a line or document.
7. **Transport.** The batch is sent; the destination's position advances only after the receiver accepted it.

Every event carries a globally unique `event_id` = `<installation id>:<journal sequence>`, for example `7d4c2a10-…-9e1f:48213`. The installation ID is `Clm:Siem:InstallationId` when the operator sets it, otherwise a UUID generated once and stored in the platform database, so every instance and restart uses the same one. Receivers deduplicate on `event_id` (see Deduplication).

1.2 Transports

Transport	ID	What it does	Typical receiver
Signed HTTPS webhook	`https`	HTTPS `POST` of a batch, signed with HMAC-SHA256 in `X-Certyx-Signature`	Your own collector, Logstash, SOAR, serverless function, Cribl HTTP source
Splunk HTTP Event Collector	`splunk-hec`	HTTPS to `/services/collector/event` with an HEC token, optional indexer acknowledgement	Splunk Enterprise, Splunk Cloud, Cribl Splunk HEC source

Transport	ID	What it does	Typical receiver
Syslog over TLS	`syslog-tls`	RFC 5424 messages over TCP with TLS, octet counting (RFC 5425) or newline framing	Cribl Stream, QRadar, ArcSight SmartConnector, Splunk TCP input, rsyslog, syslog-ng, Logstash, Elastic Agent
Microsoft Sentinel	`azure-monitor`	Logs Ingestion API through a data collection rule, authenticated with a Microsoft Entra application	Microsoft Sentinel, Log Analytics
OpenTelemetry Logs	`otlp-http`	HTTPS `POST` of an OTLP `ExportLogsServiceRequest` (protobuf or OTLP/JSON, optional gzip and one authentication header)	OpenTelemetry Collector, Grafana Alloy and Loki, any backend with an OTLP/HTTP logs endpoint
File archive	`file`	Daily files in a server-side archive directory, downloadable from the console	Long-term retention, batch import (for example OCSF pipelines)

1.3 Formats

Format	ID	Content
JSON record	`json`	The CERTYX journal record of the first release (`Sequence` , `At` , `Category` , `EventCode` , `Tenant` , ...) plus `EventId` ; a JSON array per webhook request, the HEC `event` , or the syslog message text. Quotes inside strings are written as `\"` and characters such as `+` stay as they are (not `\u0022` or `\u002B`), so key-value extraction does not invent fields and plain-text searches match. `Changes` holds nested objects, not strings with escaped quotes (also for older events)
NDJSON	`ndjson`	One JSON document of the schema profile per line (`application/x-ndjson`), UTF-8 without byte order mark
RFC 5424	`rfc5424`	An RFC 5424 syslog message with structured data whose MSG is the JSON document of the schema profile
CEF	`cef`	ArcSight CEF 0 from the canonical event (Device Event Class ID = event code, `externalId` = event ID)
LEEF 1.0	`leef-1.0`	IBM QRadar LEEF 1.0 with TAB-separated attributes
LEEF 2.0	`leef-2.0`	IBM QRadar LEEF 2.0 with the attribute delimiter declared in the header (TAB by default)
Generic KV	`kv`	The documented `certyx-kv/1.0` profile: every canonical field as `key=value` in a fixed order; not CEF or LEEF
CSV	`csv`	Canonical CSV rows, RFC 4180, with the versioned header (file archive)
Sentinel row	`sentinel`	The columns of the Sentinel custom table, plus `EventId`
OTLP protobuf	`otlp-protobuf`	OpenTelemetry Logs request in binary protobuf (`application/x-protobuf`), one log record per canonical event
OTLP/JSON	`otlp-json`	The same request in the OTLP/JSON encoding (`application/json`)
CEF (first release)	`cef-compat`	The CEF layout of the first release plus `certyxEventId=`
LEEF (first release)	`leef-compat`	The LEEF 2.0 layout of the first release (delimiter `^`) plus `eventId=`
CSV (first release)	`csv-compat`	The CSV archive columns of the first release (`certyx-security-YYYY-MM-DD.csv`), unchanged

1.4 Schema profiles

Profile	ID	Field names
Canonical	`native`	The CERTYX canonical event `certyx.event/1.0` in snake_case (`event_id` , `event_code` , `category` , `outcome` , ...); see Field reference
Elastic Common Schema	`ecs`	ECS 8.11.0: `@timestamp` , `event.*` (`event.id` = event ID, `event.code` = event code, `event.dataset` = `certyx.security`), `user.*` , `source.ip` , `host.hostname` , `service.*` , `http.*` , `url.path` , `organization.*` , `labels.*`
OCSF	`ocsf`	OCSF 1.1.0, class chosen per event and validated: Authentication (3002), Account Change (3001), API Activity (6003), otherwise Base Event (0); `metadata.uid` = event ID
Splunk CIM	`splunk-cim`	The canonical event as HEC `event` , plus the Splunk CIM fields (Authentication and Change data models) as HEC indexed fields; HEC transport only

The complete mapping of every canonical field to CEF keys, LEEF attributes, ECS, OCSF and CIM is in the field reference document `docs/SIEM-EXPORT-FIELDS.md` of the CERTYX documentation; the main fields are summarised in Field reference.

2 Allowed combinations

Choose the transport first; the form then offers its formats and, for each format, its profiles.

Transport	Format	Profiles
`https`	`ndjson`	`native` , `ecs` , `ocsf`
`https`	`json`	`native`
`splunk-hec`	`ndjson`	`native` , `ecs` , `ocsf` , `splunk-cim`
`splunk-hec`	`json`	`native`
`syslog-tls`	`rfc5424`	`native` , `ecs` , `ocsf`
`syslog-tls`	`cef`	`native`
`syslog-tls`	`leef-1.0`	`native`
`syslog-tls`	`leef-2.0`	`native`
`syslog-tls`	`kv`	`native`
`syslog-tls`	`json`	`native`
`syslog-tls`	`cef-compat`	`native`
`syslog-tls`	`leef-compat`	`native`
`azure-monitor`	`sentinel`	`native`
`otlp-http`	`otlp-protobuf`	`native`
`otlp-http`	`otlp-json`	`native`
`file`	`ndjson`	`native` , `ecs` , `ocsf`
`file`	`csv`	`native`
`file`	`csv-compat`	`native`

Any other combination is refused with *Format, schema profile and transport: choose a transport, then one of the formats and profiles offered for it.* File archives can only be set up in the console, not in the server configuration.

2.1 Destinations of the first release

Destinations saved before format, profile and transport became separate choices — and configuration destinations that set only `Kind` — keep their layout and fields. Their syslog header carries the process ID and `[timeQuality tzKnown="1"]` where it had the placeholder `-`. Their kind is an alias of a route:

Kind (first release)	Transport	Format	Additive event ID
`webhook`	`https`	`json`	`EventId` in every record
`syslog`	`syslog-tls`	`json`	`EventId` in every record
`cef`	`syslog-tls`	`cef-compat`	`certyxEventId=` extension
`leef`	`syslog-tls`	`leef-compat`	`eventId=` attribute
`splunk-hec`	`splunk-hec`	`json`	indexed field `event_id`

Kind (first release)	Transport	Format	Additive event ID
`sentinel`	`azure-monitor`	`sentinel`	column `EventId`
`csv-archive`	`file`	`csv-compat`	none: the layout is unchanged; `Sequence` together with the installation ID gives the event ID

The event ID fields are additive: parsers written for the first release keep working. To receive the canonical event code, category and the other canonical fields, choose a canonical format (`ndjson`, `rfc5424`, `cef`, `leef-1.0`, `leef-2.0`, `kv`, `csv`). For a file archive with `event_id` per row, use the canonical `csv` or the `ndjson` archive.

3 Quick start

1. Sign in as an organization Admin with MFA and open **SIEM export** (operator: **Security event journal → SIEM export**).
2. Prepare the receiver as described in its section: Splunk, Cribl, QRadar, ArcSight, Elastic, OCSF, Sentinel or webhook.
3. Enter a destination name and identifier, choose **Transport**, **Format** and **Schema profile** (recommended starting points: Splunk `splunk-hec` × `ndjson` × `native` or `splunk-cim`; Cribl `syslog-tls` × `rfc5424` × `native`; QRadar `syslog-tls` × `leef-2.0`; ArcSight `syslog-tls` × `cef`; Elastic `https` × `ndjson` × `ecs`; OCSF pipelines `https` or `file` × `ndjson` × `ocsf`; OpenTelemetry backends `otlp-http` × `otlp-protobuf`, see OpenTelemetry Logs).
4. Fill in the address, secret and trust fields, optionally filters, delivery values and a client certificate, and save.
5. Choose **Check connection**, then **Send test event**. The result names the test event's `event_id` (a UUID); search for it in the receiver.
6. Watch the destination card: it shows the position, the events still waiting, delivered events, the last success and the last failure.

4 Exported events

The export reads the CERTYX security journal. An organization's destinations and archive receive only the events of that organization, and never platform records, even those concerning it: operator actions and requests, billing suspension and reinstatement, provisioning and re-encryption of stored secrets. The organization reads these in its console. Platform logs are a separate export the platform operator configures: operator events (scope *Platform operators*) or every event of every organization (scope *All events*).

Every canonical event has an **event code** and a **category** from the event-code catalogue, for example `AUTHN-002` (`authentication`, failed sign-in) or `CONFIG-008` (`configuration-change`, SIEM destination changed). Codes are stable and never renumbered. The ten categories are `authentication`, `authorization`, `user-management`, `configuration-change`, `data-access`, `administrative`, `security`, `application-error`, `system` and `integration`.

Outcome (filter in CERTYX)	What it contains
Sign-in events (`authentication`)	Every sign-in step as its own event: `authentication.sign-in.succeeded`, `authentication.sign-in.failed`, `authentication.sign-out`, `authentication.locked`, `authentication.mfa.enrolled`, `authentication.mfa.confirmed`, `authentication.mfa.reset`, `authentication.mfa.recovery-codes-regenerated`
Business changes (`business`)	Changes made outside a user request, for example agent health changes (`agent.health.changed`), automatic renewal decisions, background inventory and execution queue events
Successes (`success`)	Completed API, console, platform and agent requests, with the business changes they made (before/after)
Received requests (`attempt`)	The record written when a request arrives, before it runs, so a request that never completes still leaves a trace; it is not a failed attempt. Sent only when checked
Rejected requests (`rejected`)	Requests refused with HTTP 4xx (other than 401/403), for example invalid input

Outcome (filter in CERTYX)	What it contains
Access denied (`denied`)	Requests refused with HTTP 401 or 403
Server errors (`error`)	Requests that failed with HTTP 5xx (exception type and trace ID, never a stack trace)

The canonical `outcome` normalizes these to `success`, `failure` or `unknown`; the journal outcome stays available as `outcome_original`.

Sign-in events. They cover the password sign-in, password with an MFA code or a recovery code, company sign-in with OIDC or SAML and the CERTYX mobile app. Sign-outs include sessions ended by expiry, inactivity, a password, e-mail or MFA change, or a logout requested by the identity provider. Each event carries the user's e-mail address, whether that identity was verified, the organization, the source IP address, the user agent, the method, the result (`success` or `failure`) and a fixed reason code. The claimed password, MFA code, recovery code or token is never recorded.

Field	Values
Method	`password`, `password+mfa`, `password+recovery-code`, `oidc`, `saml`, `mobile-password` (also `+mfa`, `+recovery-code`), `mobile-oidc`, `mobile-saml`, `mobile-sso`, `totp` (MFA changes)
Result	`success`, `failure`
Reason (failures)	`unknown-account`, `invalid-credentials`, `disabled`, `locked`, `too-many-failures`, `mfa-required`, `mfa-invalid`, `mfa-enrollment-required`, `enrollment-expired`, `sso-required`, `not-operator`, `operator-entry`, `organization-unavailable`, `organization-mismatch`, `account-not-allowed`, `ticket-invalid`, `device-refused`, `invalid-request`, `sso-error`, `sso-disabled`, `sso-no-subject`, `sso-group-overage`, `sso-not-provisioned`, `sso-no-role`, `sso-expired`, `sso-not-configured`, `sso-unverified-domain`
Reason (sign-outs)	`user`, `credentials-changed`, `idp-logout`, `session-ended`, `administrator-reset`

A failed sign-in whose e-mail address matches no account is attributed to the organization whose verified company sign-in domain matches the address, otherwise to the platform; in that case your organization does not receive it.

Business changes. Configuration and inventory changes carry the action, the resource and the values before and after the change. A change that leaves the records as they were, such as an agent seeing the same certificate on an endpoint again (`endpoint.observe`), carries the action and the resource with empty before and after values; a trust-graph revision that only advances last-seen times is not recorded as `trust-graph.updated`. An agent health change (`agent.health.changed`, actor `system:agent-health`) carries the agent, the previous and new cause and severity, the explanation and the remediation steps. Secrets, private keys and agent output are never part of an event.

Two records per request. Every request is journaled twice: an `attempt` record before it runs (code `SYSTEM-000`, so a request that never finishes still leaves a trace) and an outcome record (`success`, `rejected`, `denied` or `error`) afterwards, both with the same correlation ID (`correlation_id`, `TraceId` in the JSON record). A destination sends the `attempt` record only when *Received requests* is checked in *Send only*: with no outcome checked it receives every outcome except `attempt`, one record per request. Check *Received requests* together with the other outcomes when the SIEM must see requests that never finish; the received record stays in the journal and in a file archive.

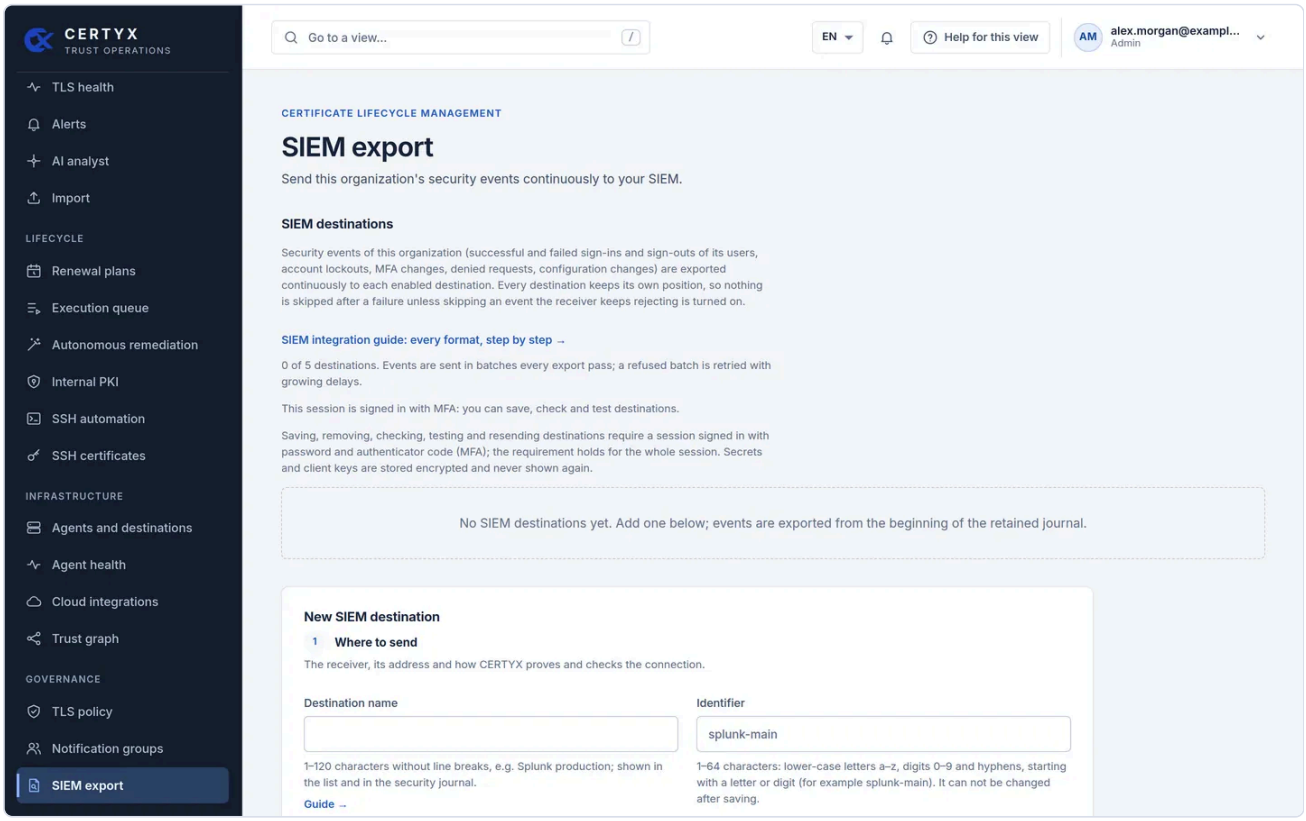
Health probes. Requests to `/health/ready` and `/health/live` (load balancer and container readiness probes) are journaled — `/health/live` not at all, `/health/ready` with both records, `Tenant` `unattributed` — but they are not sent to network destinations. The operator can send them with `Clm:Siem:ExportHealthProbes=true`. A file archive keeps them.

5 Before you start

1. **Role and MFA.** Sign in as an organization Admin with your password and authenticator code (or through company sign-in where the identity provider reports MFA). Saving, checking, testing, resending and removing destinations require an MFA session; the view tells you exactly what to do when the session does not meet it. Sessions of the mobile app cannot change destinations. See Permissions.
2. **Platform database.** SIEM export reads the journal from the platform database. When an installation keeps the journal in a local file, the view states that export is unavailable; the operator has to configure the platform database.

3. **Network path.** The CERTYX server (not your computer) opens the connection. Allow outbound TCP from the CERTYX server to the receiver's port, and make sure the receiver's host name resolves in the DNS the CERTYX server uses.
4. **Receiver certificate.** The receiver needs a TLS server certificate whose names include the host name you enter in CERTYX. When it is issued by a private or self-signed CA, keep that CA's root certificate (PEM) at hand for *Trusted root certificates*. When the receiver requires client authentication, prepare a client certificate and its unencrypted PEM private key (see Client certificate (mTLS)).
5. **Limits.** Each organization can define up to 5 destinations (operator setting `Clm:Siem:MaxDestinationsPerOrganization`). The operator of a shared installation may restrict organization destinations to public IP addresses (`Clm:Siem:AllowPrivateDestinations`); by default internal receivers are allowed.

6 Adding a destination in CERTYX



SIEM export: the form for a new destination and the destinations of the organization (sample data)

1. Open **SIEM export** (platform operator: **Security event journal** → **SIEM export**, then choose the scope).
2. Enter **Destination name** (1–120 characters, shown in the list and in the journal) and **Identifier** (1–64 lower-case letters, digits and hyphens, for example `splunk-main`; it cannot be changed after saving).
3. Choose **Transport**, then **Format** and **Schema profile** from the choices offered for it (Allowed combinations). The box *What the receiving side needs* summarises the receiver settings; the link next to each field opens the matching section of this guide.
4. Fill in the transport's fields: HTTPS address and secret (webhook, Splunk HEC, Sentinel), syslog host, port and framing (syslog), or retention and compression (file archive). For `leef-2.0`, optionally the **LEEF delimiter**.
5. Optional: paste **Trusted root certificates** (TLS and trusted roots), a **Client certificate** with its private key (mTLS) and choose the **Revocation check** (Revocation).

The form has four numbered sections: **1 Where to send** (steps 2–5), **2 What to send** (step 6), **3 Shape and size of each event** (step 6a) and **4 Delivery** (steps 7–8). Advanced parts are folded until you open them.

1. Optional: **Send only** — outcomes, minimum severity, **Send only (categories)** and **Send only (event codes)**; under **More filters** the routes to send or never send, actors, categories and event codes never to send (Send only).

6a. Optional: **Event shaping** — a preset card (**Full**, **Balanced**, **Economical**, **Custom**) and its **Detailed settings** (Event shaping); **Field mapping** — rename, drop, keep or add fields (Field mapping); **Preview test event** shows what the destination will send and how many bytes the shaping saves. For OpenTelemetry: **Log record body**, **Authentication header name** and **Compress requests with gzip** (OpenTelemetry Logs).

1. Optional: delivery values — **Attempts before a critical alert**, **First retry delay (seconds)**, **Longest retry delay (seconds)**, **Batch size (events)**, **Backlog alert above (events)** and **Skip an event the receiver keeps rejecting after the attempts above** (Delivery). Empty fields use the server defaults.
2. Keep **Export to this destination** checked (clear it to pause a destination without losing its position) and choose **Save destination**.
3. Choose **Check connection**, then **Send test event** (Connection check and test event).

A new destination starts at the beginning of the retained journal, so the receiver first gets the history and then new events as they happen. Removing a destination deletes its position; adding it again starts from the beginning. To send a period again, use **Resend from** (Resend).

Editing. *Edit* loads a destination into the form. A secret field left empty keeps the stored secret; changing the transport requires entering the secret again (*Enter the secret again when changing the format.*). Secrets (HMAC signing secret, HEC token, client secret) and the client private key are stored encrypted, never shown again and never written to the journal. Every change is journaled as `siem.destination.*` (code `CONFIG-008`).

7 TLS and trusted roots

CERTYX validates the receiver's certificate in every connection, for every transport: the certificate must be within its validity period, must name the host you entered (DNS name or IP address in the subject alternative names), must chain to a trusted root and must pass the revocation check.

- **Without trusted roots** the operating system trust store of the CERTYX server decides. Use this for receivers with a certificate from a public CA (for example Splunk Cloud or Azure).
- **With trusted roots** exactly the pasted roots are trusted (custom trust store): paste one or more PEM blocks (`-----BEGIN CERTIFICATE-----`), at most 64 KiB, without private keys. Use this for an internal CA or a self-signed receiver certificate.

Validation cannot be turned off.

To read a receiver's certificate chain from a machine in the same network:

```
openssl s_client -connect siem.example.internal:6514 -servername siem.example.internal -showcerts </dev/null
```

The last certificate printed is usually the issuing CA; its root is the certificate to paste.

7.1 Client certificate (mTLS)

Receivers that authenticate their senders by certificate (Cribl *Authenticate client*, Logstash `ssl_client_authentication`, syslog-ng `peer-verify(required-trusted)`, a reverse proxy in front of a webhook) can require CERTYX to present a client certificate. It is available for `syslog-tls` and every HTTPS transport (`https`, `splunk-hec`, `azure-monitor`); it does not apply to file archives.

1. Issue a client certificate for CERTYX from the CA the receiver trusts (key usage *digital signature*, extended key usage *client authentication*).
2. In the destination form paste **Client certificate**: one PEM certificate (`-----BEGIN CERTIFICATE-----`), at most 64 KiB.
3. Paste its **private key**: an unencrypted PEM key (`-----BEGIN PRIVATE KEY-----`, RSA or EC), at most 64 KiB. The key is write-only: it is stored encrypted with the server's Data Protection key ring, never shown again and never journaled. Leave the field empty when editing to keep the stored key.
4. **Check connection** presents the certificate in the TLS stage.

A certificate without its matching key is refused when saving. When the stored key can no longer be decrypted (the key ring changed), delivery fails with *The client certificate or its private key can not be read; enter both again.*

Configuration destinations use `ClientCertificatePem` and `ClientKey` (a `secret:<name>` reference); see `docs/CONFIGURATION.md`.

7.2 Revocation check

CERTYX checks the revocation status of every certificate of the receiver's chain below the root online, through the CRL distribution points or the OCSP responder named in the certificates, waiting at most 10 seconds per address.

Mode	Revoked certificate	Status unknown (no CRL/OCSP address, responder unreachable)	Use for
Soft-fail (default)	refused	accepted	Most receivers, including internal CAs without a published CRL
Hard-fail	refused	refused	Receivers whose CA publishes a CRL or OCSP responder the CERTYX server can reach

The CERTYX server needs outbound HTTP access to the CRL and OCSP addresses for the check to give a result. A refused certificate fails the TLS stage of the check as `tls-untrusted` (or `tls-untrusted-roots` with pasted roots).

8 Signed HTTPS webhook

CERTYX sends `POST` requests to your HTTPS address. With format `json` the body is a JSON array of journal records (`Content-Type: application/json`); with format `ndjson` the body is one document of the chosen profile per line (`Content-Type: application/x-ndjson`, each line ended by LF). A request carries up to the destination's batch size (200 by default). The header `X-Certyx-Signature` carries `sha256=` followed by the lower-case hexadecimal HMAC-SHA256 of the exact body bytes, keyed with the signing secret. Any HTTP 2xx answer accepts the batch. HTTP 400, 413 or 422 is treated as a rejection of the data (Rejected events); every other answer, a timeout (20 seconds by default) or a TLS failure makes CERTYX send the same events again after the backoff delay (Delivery).

8.1 Receiver side

1. Publish an HTTPS endpoint, for example `https://siem.example.com/certyx`, with a certificate for that host name.
2. Read the raw request body before parsing it, compute the HMAC-SHA256 with the signing secret as the key (the secret string encoded as UTF-8, used as it is — do not decode the hexadecimal text), and compare `sha256=<hex>` with `X-Certyx-Signature` in constant time. Reject a mismatch with 401.
3. Parse the body by its content type: a JSON array (`application/json`) or one JSON document per line (`application/x-ndjson`).
4. Store the events, then answer 2xx. Answer 2xx only after the batch is stored safely.
5. Deduplicate by the event ID: `EventId` in the JSON record, `event_id` (native), `event.id` (ECS) or `metadata.uid` (OCSF). Delivery is at least once, so a batch can arrive twice after a timeout.
6. Accept the test event: its event ID is a UUID; JSON record: `Sequence` 0, `Actor` `siem-test`, `Method` `INTERNAL`, `Route` `business/siem.test`; canonical: `event_code` `INTEGRATION-001`, `category` `integration`, `action` `siem.test`, `user_id` `siem-test`.

8.2 Verifying the signature

Python (standard library):

```
import hashlib, hmac, json, os
from http.server import BaseHTTPRequestHandler, HTTPServer

SECRET = os.environ["CERTYX_SIGNING_SECRET"].encode("utf-8")
```

```
def events(body, content_type):
    if content_type.startswith("application/x-ndjson"):
        return [json.loads(line) for line in body.decode("utf-8").splitlines() if line]
    return json.loads(body)

class CertyxReceiver(BaseHTTPRequestHandler):
    def do_POST(self):
        body = self.rfile.read(int(self.headers.get("Content-Length", "0")))
        expected = "sha256=" + hmac.new(SECRET, body, hashlib.sha256).hexdigest()
        if not hmac.compare_digest(expected, self.headers.get("X-Certyx-Signature", "")):
            self.send_response(401)
            self.end_headers()
            return
        for event in events(body, self.headers.get("Content-Type", "")):
            store(event)  # your storage; deduplicate by the event ID
        self.send_response(204)
        self.end_headers()

# Run behind a TLS-terminating reverse proxy, or wrap the socket with ssl.SSLContext.
HTTPServer(("0.0.0.0", 8080), CertyxReceiver).serve_forever()
```

Node.js (Express):

```
import crypto from 'node:crypto';
import express from 'express';

const secret = process.env.CERTYX_SIGNING_SECRET;
const app = express();

app.post('/certyx', express.raw({ type: ['application/json', 'application/x-ndjson'], limit: '20mb' }), (req, res) => {
  const expected = 'sha256=' + crypto.createHmac('sha256', secret).update(req.body).digest('hex');
  const given = req.get('X-Certyx-Signature') ?? '';
  const valid = given.length === expected.length && crypto.timingSafeEqual(Buffer.from(given), Buffer.from(expected));
  if (!valid) return res.sendStatus(401);
  const text = req.body.toString('utf8');
  const events = req.is('application/x-ndjson')
    ? text.split('\n').filter(Boolean).map(line => JSON.parse(line))
    : JSON.parse(text);
  for (const event of events) store(event); // deduplicate by the event ID
  res.sendStatus(204);
});
```

C# (ASP.NET Core minimal API):

```
app.MapPost("/certyx", async (HttpRequest request) =>
{
    using var buffer = new MemoryStream();
    await request.Body.CopyToAsync(buffer);
    var body = buffer.ToArray();
    var key = Encoding.UTF8.GetBytes(Environment.GetEnvironmentVariable("CERTYX_SIGNING_SECRET")!);
    var expected = "sha256=" + Convert.ToHexString(HMACSHA256.HashData(key, body)).ToLowerInvariant();
    var given = request.Headers["X-Certyx-Signature"].ToString();
    if (!CryptographicOperations.FixedTimeEquals(Encoding.ASCII.GetBytes(expected), Encoding.ASCII.GetBytes(given)))
        return Results.Unauthorized();
    var text = Encoding.UTF8.GetString(body);
```

```
    var documents = request.ContentType?.StartsWith("application/x-ndjson") == true
        ? text.Split('\n', StringSplitOptions.RemoveEmptyEntries).Select(line => JsonDocument.Parse(line).RootElement)
        : JsonDocument.Parse(text).RootElement.EnumerateArray();
    foreach (var e in documents)
        Store(e); // deduplicate by the event ID
    return Results.NoContent();
});
```

Checking a captured body on the command line (`body.json` saved byte for byte):

```
printf 'sha256=%s\n' "$(openssl dgst -sha256 -hmac "$CERTYX_SIGNING_SECRET" -r body.json | cut -d' ' -f1)"
```

8.3 CERTYX side

1. Transport **Signed HTTPS webhook** (`https`); format `ndjson` with profile `native`, `ecs` or `ocsf`, or format `json` (the record of the first release).
2. **HTTPS address**: the absolute `https://` URL of the endpoint (at most 2048 characters, no user name or password), for example `https://siem.example.com/certyx`.
3. **HMAC signing secret**: leave it empty for a new webhook. CERTYX then generates a 64-character secret and shows it once after saving — copy it into the receiver before you close the notice. You can also enter your own secret (at most 1024 characters).
4. **New secret** on the destination card replaces the secret immediately and shows the new one once; the previous secret stops working with the next delivery, so update the receiver at the same time.
5. Optional: a client certificate when the endpoint requires mTLS.
6. Save, **Check connection**, **Send test event**.

Success looks like: the check shows DNS, TCP and TLS as passed; the test event shows that the destination accepted it and names its event ID; the receiver logged one event with that ID. Within the next pass the card shows *last success* and the number of delivered events.

9 Splunk HTTP Event Collector

The recommended way into Splunk: every batch is confirmed by Splunk, and with indexer acknowledgement CERTYX waits until the events are indexed.

9.1 Choosing the Splunk path

Goal	Transport × format × profile	Sourcetype (default)
Canonical events, searchable by `event_id`, `event_code`, `category` without parsing	`splunk-hec` × `ndjson` × `native`	`certyx:event`
Splunk CIM data models (Authentication, Change)	`splunk-hec` × `ndjson` × `splunk-cim`	`certyx:event`
ECS or OCSF documents in Splunk	`splunk-hec` × `ndjson` × `ecs` / `ocsf`	`certyx:event`
Existing searches on the first-release record	`splunk-hec` × `json` × `native`	`certyx:security`
No HEC available, syslog into an indexer	`syslog-tls`, newline framing (Splunk TCP-SSL input)	set by the input

Use a different sourcetype per profile (for example `certyx:event`, `certyx:event:cim`, `certyx:event:ecs`) when several CERTYX destinations write into the same index, so `props.conf` can treat each one on its own.

9.2 Receiver side (Splunk)

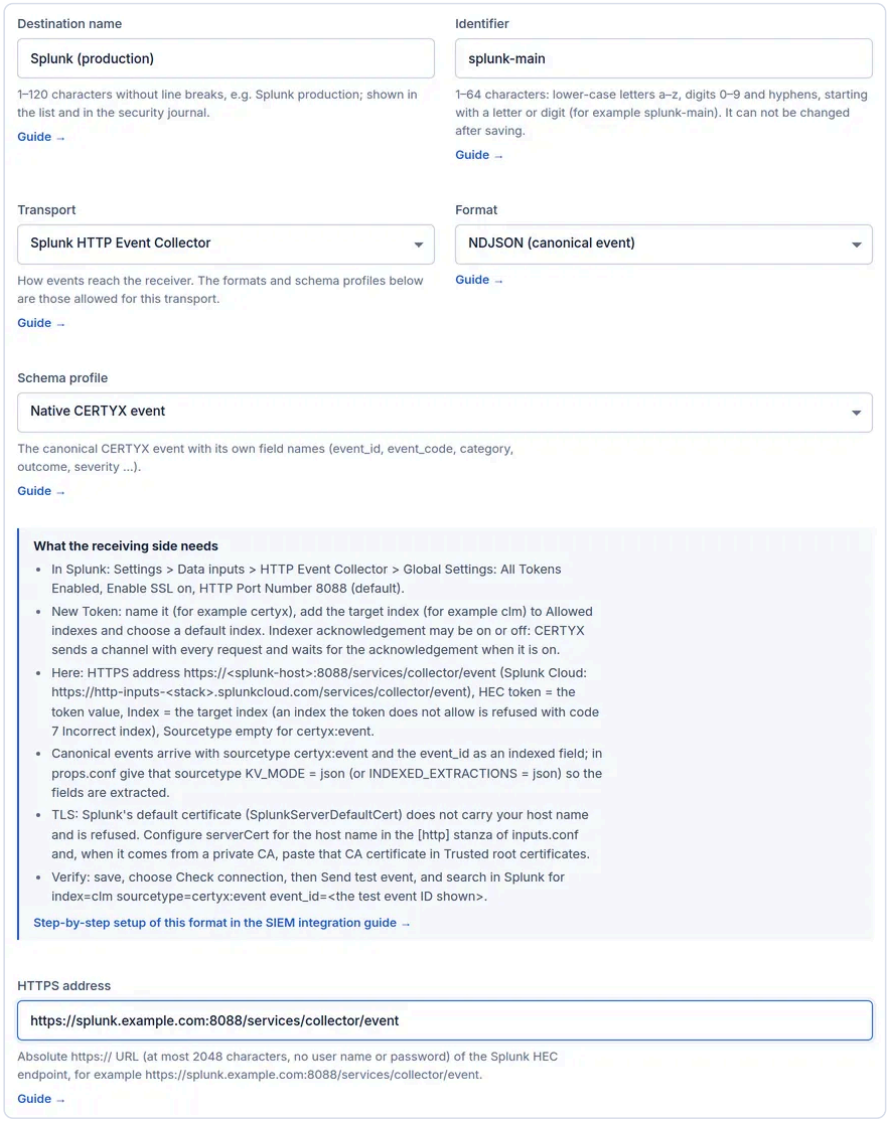
1. **Enable HEC.** *Settings* → *Data inputs* → *HTTP Event Collector* → *Global Settings*: *All Tokens* **Enabled**, **Enable SSL** checked, *HTTP Port Number* 8088 (default). Save.
2. **Create the index** (if needed): *Settings* → *Indexes* → *New Index*, for example `clm`.
3. **Create a token.** *HTTP Event Collector* → *New Token*: name `certyx`; leave *Source name override* empty; *Input Settings*: optional default *Source type* (CERTYX sets the sourcetype on every event anyway), *Allowed Indexes* includes `clm`, *Default Index* `clm`. *Enable indexer acknowledgement* may be on or off — CERTYX supports both. Copy the **Token Value**.
4. **Server certificate.** Splunk's default certificate (`SplunkServerDefaultCert`) does not carry your host name and is refused by CERTYX. Configure a certificate for the host name in the `[http]` stanza of `inputs.conf` (for example `$SPLUNK_HOME/etc/apps/splunk_httpinput/local/inputs.conf`) and restart Splunk:

```
[http]
disabled = 0
enableSSL = 1
port = 8088
serverCert = $SPLUNK_HOME/etc/auth/certyx/splunk-hec.pem
sslPassword = <password of the private key in the PEM file>
```

1. **Splunk Cloud.** Use `https://http-inputs-<stack>.splunkcloud.com/services/collector/event` (port 443). Its certificate comes from a public CA; no trusted roots are needed.
2. **Firewall.** Allow TCP 8088 (Splunk Cloud: 443) from the CERTYX server.
3. **Sourcetype settings** as in props.conf.

From a machine in the same network you can confirm that HEC answers: `curl https://splunk.example.com:8088/services/collector/health` returns `{"text":"HEC is healthy","code":17}`.

9.3 CERTYX side



A Splunk HTTP Event Collector destination being added: name, identifier, transport and collector address (sample data)

1. Transport **Splunk HTTP Event Collector** (`splunk-hec`), format and profile as chosen above.
2. **HTTPS address**: `https://<splunk-host>:8088/services/collector/event` — the `/event` endpoint, not `/raw`.
3. **HEC token**: the Token Value from Splunk (`<HEC token>`).
4. **Index (optional)**: the target index, for example `clm`. It must be one of the token's allowed indexes (otherwise Splunk refuses with code 7 *Incorrect index*); empty uses the token's default index.
5. **Sourcetype (optional)**: empty uses `certyx:event` for `ndjson` (operator setting `Clm:Siem:SplunkEventSourceType`) and `certyx:security` for `json` (`Clm:Siem:SplunkSourceType`).
6. **Trusted root certificates**: paste the CA that issued the HEC certificate when it is a private CA.
7. Save, **Check connection**, **Send test event**.

What CERTYX sends. `POST` with `Authorization: Splunk <HEC token>` , `X-Splunk-Request-Channel: <GUID>` (stable per destination; required by tokens with indexer acknowledgement, ignored by others) and one envelope per event, concatenated. Every envelope has:

- `time` — the event time, epoch seconds with milliseconds;
- `host` — the CERTYX instance that recorded the event;
- `source` — `certyx:security-journal` ;
- `sourcetype` — as configured; `index` — only when set;

- `event` — format `json`: the JSON record (JSON record); format `ndjson`: the canonical event (`native`, `splunk-cim`), the ECS document (`ecs`) or the OCSF document (`ocsf`);
- `fields` (indexed fields) — format `json`: `event_id`; format `ndjson`: always `event_id`, `event_code` and `category`, and with `splunk-cim` additionally the CIM fields (CIM).

When the answer carries an `ackId`, CERTYX polls `/services/collector/ack` every second until Splunk confirms indexing, for at most 30 seconds (operator settings `Clm:Siem:HecAckPollInterval`, `Clm:Siem:HecAckTimeout`); an unconfirmed batch is sent again. HEC codes 5, 6, 12, 13 and 15 are treated as a rejection of the data (Rejected events). Examples: Sample events.

Success looks like: *Splunk HEC accepted the test event (code 0).* with the index, sourcetype and event ID. Then search:

```
index=clm event_id=<event ID shown by the test>
index=clm sourcetype=certyx:event category=authentication outcome=failure
| stats count by username, src_ip, reason
index=clm sourcetype=certyx:security Outcome=authentication SignIn.Result=failure
| stats count by SignIn.User, SourceIp, SignIn.Reason
```

9.4 props.conf

Events from the `/event` endpoint take their time from the envelope's `time`; `TIME_PREFIX`, `TIME_FORMAT` and `TIMESTAMP_FIELDS` are not needed. Long events (many changes, national characters) need a larger `TRUNCATE` than Splunk's default of 10,000 bytes. On the search heads and indexers (or in Splunk Cloud through a custom app):

```
[certyx:event]
KV_MODE = json
TRUNCATE = 0
SHOULD_LINEMERGE = false

[certyx:security]
KV_MODE = json
TRUNCATE = 0
SHOULD_LINEMERGE = false
```

`TRUNCATE = 0` removes the limit; set a large value (for example `200000`) if your policy requires one. Indexed fields (`event_id`, `event_code`, `category`) can also be searched with the fast `field::value` syntax, for example `event_code::AUTHN-002`. A field that is both indexed and present in the JSON event can show its value twice; use `| eval event_id=mvindex(event_id,0)` or search the indexed form.

9.5 Splunk CIM

With profile `splunk-cim`, CERTYX writes the CIM fields of each event as indexed fields: the common fields `vendor_product` (`CERTYX CLM`), `app` (`clm`), `signature` (message), `signature_id` (event code), `severity` (informational, low, medium, high, critical), `severity_id`, `src`, `src_port`, `dest`, `user`, `user_id`, `session_id`; for the Authentication data model (category `authentication`) `action` (`success`, `failure`, `unknown`), `authentication_method`, `reason`; for the Change data model (categories `user-management`, `configuration-change`, and `administrative` with a change type) `action` (`created`, `modified`, `deleted`), `change_type` (`AAA` for user management, `application` otherwise), `object`, `object_category`, `object_id`, `command`, `status`, `result`; for other events `action` = the canonical action.

The canonical event in `event` uses some of the same names (`action`, `severity`, `reason`, `user_id`). For a CIM sourcetype, keep search-time JSON extraction from overwriting the CIM values, and read canonical fields with `spath` when needed:

```
[certyx:event:cim]
KV_MODE = none
TRUNCATE = 0
SHOULD_LINEMERGE = false
```

CERTYX does not ship a Splunk add-on: the event types and tags that place events into the CIM data models are not included. Create them in your own app, for example:

```
# eventtypes.conf
[certyx_authentication]
search = sourcetype=certyx:event:cim category::authentication

[certyx_change]
search = sourcetype=certyx:event:cim (category::user-management OR category::configuration-change OR category::administrative)

# tags.conf
[eventtype=certyx_authentication]
authentication = enabled

[eventtype=certyx_change]
change = enabled
```

Then verify with `| datamodel Authentication search | search sourcetype=certyx:event:cim`.

10 Syslog over TLS

All syslog formats share one transport. CERTYX opens a TLS connection to the host and port (presenting the client certificate when one is set), writes the batch and closes the connection. Syslog has no delivery acknowledgement: a completed TLS write counts as accepted.

Canonical formats (`rfc5424`, `cef`, `leef-1.0`, `leef-2.0`, `kv`) are RFC 5424 messages with the header described below; the MSG is the line of the format. **First-release formats** (`json`, `cef-compat`, `leef-compat`) keep their header layout: `<86>` (informational) or `<84>` (warning), APP-NAME `clm`, MSGID the event category (for example `authentication`; `security-journal` before 1.2.19), no structured data. Every syslog destination can choose its own MSGID (category, event code, its own text such as `security-journal`, or none) and APP-NAME in Event shaping; a receiver then filters by category without reading the message, for example `msgid=authentication` in Splunk.

10.1 RFC 5424 header of the canonical formats

`<PRI>1 TIMESTAMP HOSTNAME APP-NAME PROCID MSGID STRUCTURED-DATA MSG`

- **PRI** = facility × 8 + syslog severity. Facility from `Clm:Siem:Syslog:Facility` (default 10, authpriv; 16, local0, is the usual alternative). Severity from the canonical severity: 0–3 → 6 informational, 4–5 → 5 notice, 6–7 → 4 warning, 8 → 3 error, 9 → 2 critical, 10 → 1 alert.
- **VERSION** `1`; **TIMESTAMP** UTC with milliseconds (`2026-10-18T08:15:02.417Z`).
- **HOSTNAME** the CERTYX instance that recorded the event; **APP-NAME** `Clm:Siem:Syslog:AppName` (default `clm`); **PROCID** the process ID of the CERTYX instance; **MSGID** the event category (for example `authentication`), or what the destination's event shaping chooses (event code, own text or none); the destination may also set its own APP-NAME. Header values are printable US-ASCII; other characters become `_`.
- **STRUCTURED-DATA** always `[timeQuality tzKnown="1"]` and `[origin software="CERTYX CLM" swVersion="<version>" ip="<server address>"]` (`ip` only when known). When the operator configures an IANA Private Enterprise Number (`Clm:Siem:Syslog:IanaPrivateEnterpriseNumber`), additionally `[certyx@<PEN> eventId="…" eventCode="…" category="…" outcome="…" schemaVersion="…" organizationId="…"]` and `enterpriseId` in `origin`. CERTYX has no registered enterprise number of its own; an SD-ID with `@` must use a registered one, so without a configured number the private element is not sent. 32473 (the RFC 5612 documentation number) is refused. Values escape `"`, `\` and `]` with a backslash.
- **MSG** — `rfc5424`: the JSON document of the profile (`native`, `ecs` or `ocsf`), preceded by the UTF-8 byte order mark unless `Clm:Siem:Syslog:ByteOrderMark=false`; `cef`, `leef-1.0`, `leef-2.0`, `kv`: that line, without byte order mark.

When a receiver shows or `ï»¿` before the JSON, ask the operator to set `Clm:Siem:Syslog:ByteOrderMark=false`.

10.2 Framing

Framing tells the receiver where one message ends; it must match the receiving input.

Framing in CERTYX	On the wire	Use for
Octet counting (RFC 5425) — default	`<length> <message>`	Cribl Stream syslog source with *Octet count framing* Yes, syslog-ng, rsyslog, IBM QRadar, ArcSight
Newline (RFC 6587, Splunk TCP input)	`<message>` followed by a line feed; line feeds inside a message are written as `\n`	A Splunk `[tcp-ssl:<port>]` input, Logstash `tcp` with the `line` codec, Elastic Agent TCP input

With octet counting, `<length>` is the number of UTF-8 bytes of the message (not characters), followed by one space; there is no line feed before or after a message. CERTYX writes each record — length prefix and message together — in one write, so a record of up to 16 KiB travels in one TLS record. Both sides must use the same framing: a receiver that splits at line feeds while CERTYX counts octets shows the length of the next record glued to the end of the previous one (`… }412`), records that start with the length (`412 <84>1 …`) and records cut in the middle. See Troubleshooting.

10.3 CERTYX side (all syslog formats)

1. Transport **Syslog over TLS** (`syslog-tls`), format as needed by the receiver (`rfc5424`, `cef`, `leef-1.0`, `leef-2.0`, `kv`, or a first-release format).
2. **Syslog host**: the DNS name or IP address only — no `https://`, no `:port` (for example `cribl-worker1.example.internal`). The name must resolve on the CERTYX server and must be named in the receiver's certificate.
3. **Port**: the TLS port of the receiving input (for example 6514 or 10514).
4. **Framing**: as in the table above.
5. **LEEF delimiter** (`leef-2.0` only): one character or its hexadecimal code (`x09`, `0x5E`); empty is TAB. Allowed are TAB and printable ASCII punctuation except `=`, the pipe and the backslash.
6. **Trusted root certificates**: the CA of the receiver's certificate when it is a private CA; optionally a client certificate.
7. Save, **Check connection**, **Send test event**.

Success looks like: *Test event written over TLS (octet-counting).* with the event ID. Syslog has no delivery acknowledgement, so this proves DNS, TCP, TLS and the write. Confirm on the receiver that the event with that ID arrived in the expected index or table.

10.4 Generic syslog receiver (rsyslog, syslog-ng)

rsyslog with a TLS listener on 6514 (octet counting is recognised automatically):

```
global(DefaultNetstreamDriver="gtls"
       DefaultNetstreamDriverCAFile="/etc/rsyslog.d/tls/ca.pem"
       DefaultNetstreamDriverCertFile="/etc/rsyslog.d/tls/siem.example.internal.pem"
       DefaultNetstreamDriverKeyFile="/etc/rsyslog.d/tls/siem.example.internal.key")
module(load="imtcp" StreamDriver.Name="gtls" StreamDriver.Mode="1" StreamDriver.AuthMode="anon")
input(type="imtcp" port="6514" ruleset="certyx")
ruleset(name="certyx") { action(type="omfile" file="/var/log/certyx/security.log") }
```

To require the CERTYX client certificate, use `StreamDriver.AuthMode="x509/certvalid"` with the issuing CA in `DefaultNetstreamDriverCAFile`.

syslog-ng:

```
source s_certyx {
  syslog(transport("tls") port(6514)
         tls(key-file("/etc/syslog-ng/tls/siem.key") cert-file("/etc/syslog-ng/tls/siem.pem") peer-verify(optional-
untrusted)));
};
```

```
destination d_certyx { file("/var/log/certyx/security.log"); };
log { source(s_certyx); destination(d_certyx); };
```

With a CERTYX client certificate use `peer-verify(required-trusted)` and `ca-dir()` or `ca-file()` with the issuing CA. The JSON MSG can be parsed with the receiver's JSON parser (rsyslog `mmjsonparse` with a cookie-less configuration, syslog-ng `json-parser()`); the structured data is available as the receiver's SDATA fields.

11 Cribl Stream

Cribl receives the events, parses them and forwards them to Splunk or another destination. Cribl does not choose a Splunk index on its own; set it in the route or pipeline.

11.1 Choosing the Cribl source

Cribl source	CERTYX transport × format	Notes
Syslog (TCP with TLS)	`syslog-tls` × `rfc5424` (profile `native`, `ecs` or `ocsf`), or `cef`, `leef-2.0`, `kv`	Framing must match: CERTYX octet counting ↔ Cribl *Octet count framing* **Yes**
Splunk HEC	`splunk-hec` × `ndjson`	Cribl answers like Splunk; CERTYX sends to `https://<worker>:8088/services/collector/event`
HTTP (raw)	`https` × `ndjson`	Cribl does not verify the HMAC signature; authenticate CERTYX with a client certificate (mTLS)

11.2 Receiver side (Cribl syslog source)

1. **Certificate.** *Group Settings* → *Security* → *Certificates* → *Add certificate*: upload the certificate (with its chain) and private key for the worker's host name, for example `cribl-worker1.example.internal`. The name must be the one you enter as Syslog host in CERTYX.
2. **One source per CERTYX port.** *Data* → *Sources* → *Syslog* → *Add Source*: Input ID `certyx_tls`, Address `0.0.0.0`, *TCP Port* `10514`, *UDP Port* empty. Do not share this port with other senders that use another framing or no TLS: framing and TLS apply to every connection of the source.
3. **TLS.** In the source's *TLS Settings (TCP Only)*: **Enabled** Yes, *Certificate name* the certificate from step 1. *Authenticate client (mutual auth)*: Yes when CERTYX presents a client certificate (mTLS) — then add the CA that issued it as *CA certificate* — otherwise No. With TLS off, Cribl indexes CERTYX's TLS handshake as events (binary data starting with `\x16\x03` that contains the server name).
4. **Framing.** In the source's *Advanced Settings*: **Octet count framing** Yes (CERTYX framing *Octet counting (RFC 5425)*). Alternatively keep it No and choose *Newline* framing in CERTYX — but never mix them.
5. **Processing.** A pipeline on the source or route with: - *Parser* (type JSON object) on the field `message` for `rfc5424` and the first-release `json`; for CEF and LEEF the parser of your Cribl pack; for `kv` a *Parser* of type *Key=Value pairs*; - *Eval* with the fields `index` = `'clm'` and `sourcetype` = `'certyx:syslog'` (Cribl's Splunk destination uses these fields; without them the events land in the destination's default index).
6. **Route.** *Routing* → *Data Routes* → *Add Route*: Filter `__inputId=='syslog:certyx_tls:tcp'`, Pipeline as in step 5, Output your Splunk destination. Place it above catch-all routes.
7. **Commit and deploy** the configuration to the worker group, and allow TCP 10514 from the CERTYX server to the workers.

11.3 Receiver side (Cribl Splunk HEC source)

1. *Data* → *Sources* → *Splunk HEC* → *Add Source*: Input ID `certyx_hec`, Port `8088`, HEC endpoint `/services/collector`.
2. *Auth tokens*: add a token (`<HEC token>`) and optionally fields `index` and `sourcetype` to add to its events.
3. *TLS Settings*: Enabled Yes, the worker's certificate; mutual authentication as for the syslog source.
4. In CERTYX: transport `splunk-hec`, format `ndjson`, HTTPS address `https://cribl-worker1.example.internal:8088/services/collector/event`, the token, and an index if your Cribl route forwards it.

11.4 CERTYX side

Syslog host `cribl-worker1.example.internal` (or the load balancer name in front of the workers, which must be named in the certificate), Port `10514`, Framing **Octet counting (RFC 5425)** (or **Newline** when the Cribl source has *Octet count framing* No), Trusted root certificates when the Cribl certificate comes from a private CA. Then **Check connection** and **Send test event**, and use *Live Data* on the source in Cribl to see the event with the test event's ID.

Success looks like: the test event appears in Cribl's live capture of the source as one event whose `message` starts with the JSON document (for `rfc5424`) or the CEF/LEEF/KV line — no leading length, no binary data — and with `appname` `clm` and the category as `msgid`; then in Splunk under the index the route sets. When it is in Cribl but not in the index, check the route filter and the `index` field. Fragments or binary events: see Receiver shows fragments or binary data.

12 Splunk TCP-SSL input

Use this when the syslog stream should go directly into a Splunk indexer or heavy forwarder without HEC. Splunk splits a TCP stream at line breaks, so CERTYX must use **newline** framing.

12.1 Receiver side (Splunk)

1. Create the index (for example `clm`).
2. Install a certificate for the host name and configure the input in `inputs.conf` (for example `$SPLUNK_HOME/etc/system/local/inputs.conf`):

```
[tcp-ssl:6514]
index = clm
sourcetype = certyx:syslog
disabled = 0

[SSL]
serverCert = $SPLUNK_HOME/etc/auth/certyx/splunk-syslog.pem
sslPassword = <password of the private key in the PEM file>
requireClientCert = false
```

1. Line breaking and time stamps in `props.conf`:

```
[certyx:syslog]
SHOULD_LINEMERGE = false
LINE_BREAKER = ([\r\n]+)
TIME_PREFIX = ^<\d+>1\s
TIME_FORMAT = %Y-%m-%dT%H:%M:%S.%3NZ
TZ = UTC
MAX_TIMESTAMP_LOOKAHEAD = 30
TRUNCATE = 0
```

1. Restart Splunk and allow TCP 6514 from the CERTYX server.

12.2 CERTYX side

Transport `syslog-tls`, format `rfc5424` (or `json`, `cef`, `kv`), Syslog host = the Splunk host name in its certificate, Port `6514`, Framing **Newline**, Trusted root certificates for a private CA. **Check connection**, **Send test event**, then search (the JSON starts at the first `{` of the message):

```
index=clm sourcetype=certyx:syslog "<event ID shown by the test>"
index=clm sourcetype=certyx:syslog
```

```
| rex "(?<payload>\{.*\})$"
| spath input=payload
| search category=authentication
```

13 IBM QRadar

QRadar receives LEEF over syslog/TLS. Three LEEF formats are available:

13.1 Choosing the LEEF format

Format	Event ID in the header	Attributes	Use for
`leef-2.0` (recommended)	event code	canonical fields, delimiter TAB by default or the configured one	New QRadar integrations
`leef-1.0`	event code	canonical fields, TAB-separated	QRadar versions or DSMs that expect LEEF 1.0
`leef-compat`	`certyx:<outcome>` or `certyx:<event name>`	first-release attributes plus `eventId`, delimiter `^`	Existing QRadar content built on the first release

Headers:

```
LEEF:2.0|CERTYX|CLM|<version>|<event code>|<delimiter>|
LEEF:1.0|CERTYX|CLM|<version>|<event code>|
LEEF:2.0|CERTYX|CLM|<version>|certyx:<outcome or event name>|^|
```

The QRadar Event ID is the LEEF header's event ID: the event code (`AUTHN-002`, `CONFIG-008`, …) for `leef-1.0` and `leef-2.0`. Time: `devTime` with `devTimeFormat` `yyyy-MM-dd'T'HH:mm:ss.SSSZ`. Severity: `sev` 1–10. Main attributes: `cat` (category), `usrName`, `src`, `identSrc` (sign-in events), `proto`, `eventId` (event ID), `schemaVersion`, `action`, `outcome`, `message`, `reason`, `userId`, `authMethod`, `method`, `resource`, `httpStatus`, `organization`, `organizationName`, `organizationKind`, `correlationId`, `resourceType`, `resourceId`, `changeType`, `instance`. Values escape the backslash and the delimiter with a backslash; line breaks and tabs become spaces.

13.2 Receiver side (QRadar)

1. *Admin* → *Data Sources* → *Log Sources* → *Add* (QRadar Log Source Management app).
2. **Log Source Type**: *Universal LEEF* (or a custom DSM you create in the DSM Editor). **Protocol**: *TLS Syslog*.
3. **Log Source Identifier**: the CERTYX instance ID (`Clm:Instance:Id`; in a container deployment `CERTYX_INSTANCE_ID`, by default the console host name such as `certyx.example.com`, the same across releases) — the HOSTNAME field of every CERTYX syslog message. With several CERTYX instances, add one log source per instance ID or let QRadar auto-detect them.
4. **TLS Listen Port**: 6514 (default) or another port. **Authentication Mode**: TLS, or TLS and client authentication when CERTYX presents a client certificate. **Certificate Type**: the certificate QRadar generates, or *Provide Certificate* with a certificate for the QRadar host name.
5. Deploy the changes and allow TCP to the listen port from the CERTYX server.
6. **DSM Editor.** Map the event IDs (event codes) to QIDs; the event-code catalogue lists every code with its category. Create custom event properties for the CERTYX keys you want to search or report on, for example `eventId` (deduplication), `schemaVersion`, `organization`, `organizationName`, `correlationId`, `authMethod`, `reason`, `resourceType`, `resourceId`, `changeType`. QRadar takes the event time from `devTime` when `devTimeFormat` is present; check that *Log Activity* shows the CERTYX time, not the receive time.

13.3 CERTYX side

Transport `syslog-tls`, format `leef-2.0` (or `leef-1.0`, `leef-compat`), Syslog host = QRadar event collector host name (named in its certificate), Port = TLS Listen Port, Framing **Octet counting (RFC 5425)**, optional **LEEF delimiter** for `leef-2.0` (TAB by default; for example `^` or `x5E` when your DSM expects the caret). When QRadar uses its generated certificate or a private CA, paste that certificate's root (PEM) in **Trusted root certificates**. **Check connection**, **Send test event**, then find the event in *Log Activity* filtered by the log source (Event ID `INTEGRATION-001`). Examples: Sample events.

14 ArcSight

CEF over syslog/TLS carries ArcSight Common Event Format version 0, vendor `CERTYX`, product `CLM`.

14.1 Choosing the CEF format

Format	Device Event Class ID	Use for
`cef` (recommended)	event code (`AUTHN-002`, ...); `externalId` = event ID	New ArcSight integrations, any CEF parser
`cef-compat`	`certyx:<outcome>` or `certyx:<event name>`; `externalId` = journal sequence, `certyxEventId` = event ID	Existing ArcSight content built on the first release

Mapping of the `cef` format (unknown and empty values are left out; header fields escape the backslash and the pipe and turn line breaks into spaces; extension values escape `\`, `=`, CR and LF):

CEF key	Canonical field
Device Event Class ID	`event_code`
Name	`message`
Severity	`severity` (0–10)
`rt`	`timestamp` (epoch milliseconds)
`externalId`	`event_id`
`act`, `cat`, `outcome`, `msg`, `reason`	`action`, `category`, `outcome`, `message`, `reason`
`src`	`src_ip`
`suser`, `suid`	`username` (else `user_id`), `user_id`
`requestMethod`, `request`, `requestClientApplication`	`http_method`, `http_route`, `user_agent`
`app`, `dvchost`	`protocol`, `hostname`
`cn1` (`httpStatus`)	`http_status`
`cs1` … `cs6`	`organization_id`, `organization_name`, `correlation_id`, `resource_type`, `resource_id`, `schema_version` (labels in `cs1Label` … `cs6Label`)
`flexString1` (`changeType`), `flexString2` (`authMethod`)	`change_type`, `authentication_method`

14.2 Receiver side (ArcSight SmartConnector)

1. Install a **Syslog NG Daemon** SmartConnector (the syslog connector type that accepts TLS) on a host the CERTYX server can reach.
2. In the connector setup choose protocol **TLS** (Raw TCP over TLS) and the port, for example 6514.
3. Import the server certificate and key for the connector's host name into the connector's keystore as described in the SmartConnector documentation; enable client certificate authentication only when CERTYX presents a client certificate (mTLS).

4. Register the connector with ArcSight ESM or Logger as destination. CEF messages are parsed by the connector without a parser override; the `cs1`–`cs6` labels arrive as *Device Custom String* labels.
5. Allow TCP 6514 from the CERTYX server.

14.3 CERTYX side

Transport `syslog-tls`, format `cef` (or `cef-compat`), Syslog host = connector host name, Port 6514, Framing **Octet counting (RFC 5425)**, Trusted root certificates for a private CA. **Check connection**, **Send test event**, and look for an event with *Device Vendor* CERTYX, *Device Event Class ID* `INTEGRATION-001` and *External ID* equal to the test event's ID. Examples: Sample events.

15 Elastic (ECS)

The `ecs` profile writes Elastic Common Schema 8.11.0 documents: `@timestamp`, `message`, `ecs.version`, `event.id` (event ID), `event.code` (event code), `event.kind` `event`, `event.category`, `event.type`, `event.action`, `event.outcome`, `event.severity`, `event.created`, `event.dataset` `certyx.security`, `event.provider`, `event.reason`, `user.id`, `user.name`, `source.ip`, `host.hostname`, `service.*`, `http.request.method`, `http.request.id`, `http.response.status_code`, `url.path` (route template), `organization.id`, `organization.name`, `labels.schema_version`, `labels.certyx_category`. Unknown values are left out.

Option A — Logstash over HTTPS (recommended). CERTYX: transport `https`, format `ndjson`, profile `ecs`, a client certificate for mTLS. Logstash:

```
input {
  http {
    port => 8443
    ssl_enabled => true
    ssl_certificate => "/etc/logstash/tls/logstash.example.internal.pem"
    ssl_key => "/etc/logstash/tls/logstash.example.internal.key"
    ssl_client_authentication => "required"
    ssl_certificate_authorities => ["/etc/logstash/tls/certyx-client-ca.pem"]
    additional_codecs => { "application/x-ndjson" => "json_lines" }
    remote_host_target_field => "[@metadata][input][remote_host]"
    request_headers_target_field => "[@metadata][input][headers]"
  }
}
output {
  elasticsearch {
    hosts => ["https://elasticsearch.example.internal:9200"]
    data_stream => true
    data_stream_type => "logs"
    data_stream_dataset => "certyx.security"
    data_stream_namespace => "default"
    document_id => "%{[event][id]}"
  }
}
```

The Logstash `http` input does not check the HMAC signature; the client certificate authenticates CERTYX instead (the signature is still sent). The two `…_target_field` options keep the request metadata the input adds out of the document, so it does not mix with the document's own ECS `http`, `url` and `host` fields. The data stream is `logs-certyx.security-default`; data stream names must be lower case and must not contain `-` in the dataset.

Option B — syslog over TLS to Logstash or the Elastic Agent *Custom TCP Logs* integration with TLS: CERTYX transport `syslog-tls`, format `rfc5424`, profile `ecs`, framing **Newline**; on the receiver split lines, strip the syslog header up to the first `{` (and the byte order mark, or ask the operator to set `Clm:Siem:Syslog:ByteOrderMark=false`) and decode the JSON (Logstash

`json` filter, Agent `decode_json_fields` processor).

Option C — file archive with format `ndjson` and profile `ecs`, imported in batches (for example with Filebeat or Elastic Agent reading the downloaded files).

Deduplication. Use `event.id` as the document `_id` (`document_id` above): a resent event is refused as a duplicate (`409`) instead of indexed twice. A data stream checks the `_id` only within its current backing index; after a rollover a late duplicate can be indexed again, so search with `event.id` when exact counts matter. Alternatively the Logstash `fingerprint` filter with `source => "[event][id]"` can produce the `_id`.

16 OCSF consumers and Amazon Security Lake

The `ocsf` profile writes OCSF 1.1.0 JSON documents. For each event CERTYX chooses a class and validates its required attributes:

Class	Events
Authentication (3002)	category `authentication`: activity 1 Logon (sign-in), 2 Logoff (sign-out), 99 Other
Account Change (3001)	account and SCIM user changes, MFA reset: 1 Create, 5 Disable, 6 Delete, 11 MFA Factor Disable, 99 Other
API Activity (6003)	other request events: 1 Create (POST), 2 Read (GET/HEAD), 3 Update (PUT/PATCH), 4 Delete (DELETE), 99 Other
Base Event (0)	everything else, and the fall-back

Every document has `class_uid`, `category_uid`, `activity_id`, `type_uid`, `severity_id`, `status_id`, `time` (epoch milliseconds), `message`, `metadata.uid` (event ID), `metadata.event_code`, `metadata.version`, `metadata.product` and `unmapped.certyx_schema_version`, `unmapped.certyx_category`, `unmapped.certyx_action`. When an event does not have the required attributes of its class — for example an anonymous request without `actor.user`, or a request without a source address — it is written as a Base Event, and the reasons are listed in `unmapped.certyx_ocsf_fallback`. CERTYX never claims a class without validation; the validator checks presence and type of the required attributes, not the complete OCSF schema.

Delivery. Transport `https` × `ndjson` × `ocsf` to your collector, `splunk-hec` × `ndjson` × `ocsf`, `syslog-tls` × `rfc5424` × `ocsf`, or the file archive `file` × `ndjson` × `ocsf` (`certyx-events-YYYY-MM-DD.ndjson`).

Amazon Security Lake. A Security Lake custom source accepts OCSF in Apache Parquet, one OCSF event class per custom source, written to the source's S3 location. CERTYX delivers OCSF JSON; the conversion is the receiver's job: run a collector (for example behind the HTTPS webhook, or reading the downloaded `ndjson` archive) that groups the documents by `class_uid`, converts them to Parquet and writes them to the custom source of that class. Events written as Base Event (`class_uid` 0) need a custom source of their own or must be filtered out by your pipeline. Deduplicate on `metadata.uid`.

17 Microsoft Sentinel

CERTYX writes to a custom Log Analytics table through the Azure Monitor Logs Ingestion API: a data collection endpoint (DCE) receives the data, a data collection rule (DCR) maps the stream to the table, and a Microsoft Entra application authenticates CERTYX.

17.1 What the table receives

Transport `azure-monitor`, format `sentinel`: one row per event with the columns of the first release plus `EventId` (string, `<installation id>:<journal sequence>`). Declare `EventId` in the table and the stream to store it; a rule without it keeps ingesting the other columns. Column list: Sentinel columns.

17.2 Receiver side (Azure)

1. **Workspace.** A Log Analytics workspace with Microsoft Sentinel enabled.

2. **Data collection endpoint.** *Monitor → Data Collection Endpoints → Create*, in the region of the workspace. Copy its **Logs Ingestion** URL, for example `https://certyx-dce-a1b2.westeurope-1.ingest.monitor.azure.com`.
3. **Table and rule.** In the workspace: *Tables → Create → New custom log (DCR-based)*: table name `CertyxSecurity` (Azure adds `_CL`), *Create a new data collection rule*, select the endpoint from step 2, upload the sample below, keep the transformation `source`. Or create them with the Azure CLI (below).
4. **Immutable ID.** Open the rule → *JSON View* → copy `immutableId` (`dcr-` followed by 32 hexadecimal characters). The stream name is `Custom-CertyxSecurity_CL`.
5. **Application.** *Microsoft Entra ID → App registrations → New registration*: name `certyx-siem-export`, single tenant, no redirect URI. Copy the **Directory (tenant) ID** and **Application (client) ID**. *Certificates & secrets → New client secret*: copy the secret's **Value** (not its Secret ID) and note its expiry.
6. **Permissions.** The application needs no API permissions and no admin consent. Grant it the role **Monitoring Metrics Publisher** on the data collection rule: rule → *Access control (IAM) → Add role assignment* → Monitoring Metrics Publisher → the application. A new role assignment can take several minutes to take effect; until then the API answers 403.
7. **Network.** Allow outbound HTTPS from the CERTYX server to `login.microsoftonline.com` and to the DCE host.

Sample for the table wizard (one record with every column):

```
[{"TimeGenerated":"2026-10-11T08:15:02.417Z","EventId":"<installation id>:48213","Sequence":48213,"Organization":"3f2c9a6e-8d1b-4c55-9a0e-2b7f1d4c8e90","OrganizationName":"Example Ltd","OrganizationKind":"organization","Actor":"anonymous","SourceIp":"203.0.113.24","AgentFingerprint":"","Method":"POST","Route":"/auth/customer/login","Status":401,"Outcome":"authentication","TraceId":"0HN7J3K2L1M0:00000003","ErrorType":"","Instance":"certyx-web-1","EventHash":"9C1F","Changes":[],"User":"jane.doe@example.com","UserVerified":true,"UserOrganization":"3f2c9a6e-8d1b-4c55-9a0e-2b7f1d4c8e90","AuthAction":"authentication.sign-in.failed","AuthMethod":"password","AuthResult":"failure","Reason":"invalid-credentials","UserAgent":"Mozilla/5.0"}]
```

Azure CLI alternative (replace the values in angle brackets):

```
az monitor log-analytics workspace table create --resource-group <rg> --workspace-name <workspace> \
  --name CertyxSecurity_CL --columns TimeGenerated=datetime EventId=string Sequence=long Organization=string \
  OrganizationName=string OrganizationKind=string Actor=string SourceIp=string AgentFingerprint=string \
  Method=string Route=string Status=int Outcome=string TraceId=string ErrorType=string Instance=string \
  EventHash=string Changes=dynamic User=string UserVerified=boolean UserOrganization=string AuthAction=string \
  AuthMethod=string AuthResult=string Reason=string UserAgent=string

az monitor data-collection rule create --resource-group <rg> --location <region> \
  --name certyx-dcr --rule-file certyx-dcr.json

az role assignment create --assignee <application-client-id> --role "Monitoring Metrics Publisher" \
  --scope /subscriptions/<subscription>/resourceGroups/<rg>/providers/Microsoft.Insights/dataCollectionRules/certyx-dcr
```

`certyx-dcr.json` (the stream declares the same columns as the table):

```
{
  "location": "<region>",
  "properties": {
    "dataCollectionEndpointId":
"/subscriptions/<subscription>/resourceGroups/<rg>/providers/Microsoft.Insights/dataCollectionEndpoints/<dce>",
    "streamDeclarations": {
      "Custom-CertyxSecurity_CL": {
        "columns": [
          { "name": "TimeGenerated", "type": "datetime" }, { "name": "EventId", "type": "string" },
          { "name": "Sequence", "type": "long" }, { "name": "Organization", "type": "string" },
          { "name": "OrganizationName", "type": "string" }, { "name": "OrganizationKind", "type": "string" },
          { "name": "Actor", "type": "string" }, { "name": "SourceIp", "type": "string" },
```

```
          { "name": "AgentFingerprint", "type": "string" }, { "name": "Method", "type": "string" },
          { "name": "Route", "type": "string" }, { "name": "Status", "type": "int" },
          { "name": "Outcome", "type": "string" }, { "name": "TraceId", "type": "string" },
          { "name": "ErrorType", "type": "string" }, { "name": "Instance", "type": "string" },
          { "name": "EventHash", "type": "string" }, { "name": "Changes", "type": "dynamic" },
          { "name": "User", "type": "string" }, { "name": "UserVerified", "type": "boolean" },
          { "name": "UserOrganization", "type": "string" }, { "name": "AuthAction", "type": "string" },
          { "name": "AuthMethod", "type": "string" }, { "name": "AuthResult", "type": "string" },
          { "name": "Reason", "type": "string" }, { "name": "UserAgent", "type": "string" }
        ]
      }
    },
    "destinations": {
      "logAnalytics": [
        { "name": "workspace", "workspaceResourceId":
"/subscriptions/<subscription>/resourceGroups/<rg>/providers/Microsoft.OperationalInsights/workspaces/<workspace>" }
      ]
    },
    "dataFlows": [
      { "streams": ["Custom-CertyxSecurity_CL"], "destinations": ["workspace"], "transformKql": "source", "outputStream":
"Custom-CertyxSecurity_CL" }
    ]
  }
}
```

A rule created before `EventId`, `OrganizationName`, `OrganizationKind` or the sign-in columns existed keeps ingesting the other columns; add the missing columns to the table and the stream to store them.

17.3 CERTYX side

1. Transport **Microsoft Sentinel** (`azure-monitor`), format `sentinel`.
2. **HTTPS address**: the Logs Ingestion URL of the data collection endpoint.
3. **Data collection rule (immutable ID)**: `dcr-…` from the rule's JSON view.
4. **Stream**: `Custom-CertyxSecurity_CL`.
5. **Microsoft Entra tenant ID** and **Application (client) ID**: GUIDs from the app registration.
6. **Client secret**: the secret's Value. Renew it in Entra before it expires and enter the new value here.
7. Save, **Check connection**, **Send test event**.

CERTYX obtains a token with the client-credentials flow from `https://login.microsoftonline.com/<tenant>/oauth2/v2.0/token` (scope `https://monitor.azure.com//.default`) and posts the batch to `<endpoint>/dataCollectionRules/<immutable ID>/streams/<stream>?api-version=2023-01-01`. Tokens are reused until shortly before they expire. Sovereign clouds use the authority host the operator sets in `Clm:Siem:SentinelAuthorityHost`.

Success looks like: the destination accepted the test event and its event ID is shown. Data appears in the table after the ingestion delay (typically a few minutes):

```
CertyxSecurity_CL
| where Actor == "siem-test"

CertyxSecurity_CL
| summarize arg_max(TimeGenerated, *) by EventId
| where Outcome == "authentication" and AuthResult == "failure"
| summarize Failures = count() by User, SourceIp, Reason, bin(TimeGenerated, 15m)
```

18 OpenTelemetry Logs (OTLP/HTTP)

CERTYX sends the canonical event as OpenTelemetry log records over OTLP/HTTP to any receiver that implements the OTLP logs endpoint: the OpenTelemetry Collector, Grafana Alloy, Grafana Loki or a backend's own OTLP endpoint. Use it to correlate security events with the traces and metrics of your applications; the classic SIEM formats stay available next to it.

18.1 What the receiver gets

Transport `otlp-http`, format `otlp-protobuf` (binary protobuf, `Content-Type: application/x-protobuf`, the OTLP/HTTP default) or `otlp-json` (OTLP/JSON, `Content-Type: application/json`), profile `native`. Each batch is one `POST` of an `ExportLogsServiceRequest` (opentelemetry-proto `collector/logs/v1`) to the URL exactly as entered — CERTYX does not append `/v1/logs` — optionally with `Content-Encoding: gzip`. The protobuf message is encoded by CERTYX itself against the field numbers of opentelemetry-proto v1 (no protobuf library).

The request has one `ResourceLogs` per distinct resource (in practice one per organization of the batch), one `ScopeLogs` (scope `certyx.security-journal`, version = product version) and one `LogRecord` per event:

OTLP	From the canonical event
Resource `service.name`	`certyx`
Resource `service.version`, `deployment.environment.name`, `host.name`, `host.ip`	`version`, `environment`, `hostname`, `host_ip` (array)
Resource `certyx.organization.id`, `certyx.organization.name`, `certyx.organization.kind`	`organization_id`, `organization_name`, `organization_kind`
`Timestamp`, `ObservedTimestamp`	`timestamp`, `observed_timestamp` (Unix nanoseconds)
`SeverityNumber`, `SeverityText`	`severity` 0–3 → 9 `INFO`, 4–6 → 13 `WARN`, 7–8 → 17 `ERROR`, 9–10 → 21 `FATAL`
`Body`	`message`; with *Log record body: Canonical event* the whole canonical document as a key-value list
`TraceId`, `SpanId`	`trace_id`, `span_id` (W3C IDs of the request, enrichment); left out when unknown
`event.name`	`action`
`user.id`, `user.name`, `user.roles`, `session.id`, `user_agent.original`	`user_id`, `username`, `user_role` (array), `session_id`, `user_agent`
`client.address`, `client.port`, `server.address`, `server.port`	`src_ip`, `src_port`, `dst_ip`, `dst_port`
`http.request.method`, `http.route`, `http.response.status_code`, `network.protocol.name`, `error.type`	`http_method`, `http_route` (the route template), `http_status`, `protocol`, `error_type`
`certyx.event.id`, `certyx.event.code`, `certyx.event.category`, `certyx.event.outcome`, `certyx.event.severity`, …	`event_id`, `event_code`, `category`, `outcome`, `severity`; every other canonical field as `certyx.<field>`; `certyx.changes` is the compact JSON text of the changes
Custom fields of a field mapping	Attributes with their own names

Unknown values are left out. In OTLP/JSON 64-bit integers and times are decimal strings and trace and span IDs are hexadecimal strings; in protobuf they are integers and bytes. Deduplicate on the attribute `certyx.event.id`.

Answers. 2xx without `partial_success` delivers the batch. A 2xx with `partial_success.rejected_log_records` above zero is a partial success: as the OTLP specification requires, CERTYX does **not** resend the request (the receiver kept the accepted records). The position moves past the batch, the rejection is journaled as `siem.event.rejected` (code `INTEGRATION-013`) with the destination, the number of refused records, the event IDs and journal sequences of the request (OTLP does not say which records were refused, so all of the request are listed; for more than 200 events the first and the last) and the receiver's `error_message` (printable, at most 200 characters), and the incident `siem-rejected` is raised with the count; it stays open until an operator closes it. **Nothing is lost silently**: the events stay in the journal and in file archives, and after the receiver is fixed Resend from the first listed sequence sends them again (the receiver deduplicates the accepted ones by `certyx.event.id`).

HTTP 400, 413 and 422 split the batch down to the refused event (Rejected events); 401, 403, 404, 415, 429 and 5xx retry the whole batch with backoff, and a `Retry-After` on 429 or 503 is honoured (the next attempt waits at least that long, at most the destination's longest retry delay).

18.2 Receiver side (OpenTelemetry Collector)

The `otlp` receiver of the OpenTelemetry Collector (contrib distribution for the `bearertokenauth` extension) with TLS on port 4318 and a bearer token:

```
extensions:
  bearertokenauth:
    token: ${env:CERTYX_OTLP_TOKEN}
receivers:
  otlp:
    protocols:
      http:
        endpoint: 0.0.0.0:4318
        tls:
          cert_file: /etc/otelcol/tls/server.crt
          key_file: /etc/otelcol/tls/server.key
          # client_ca_file: /etc/otelcol/tls/certyx-client-ca.pem
        auth:
          authenticator: bearertokenauth
processors:
  batch: {}
exporters:
  debug:
    verbosity: detailed
service:
  extensions: [bearertokenauth]
  pipelines:
    logs:
      receivers: [otlp]
      processors: [batch]
      exporters: [debug]
```

1. Give the receiver a certificate whose names include the host name you enter in CERTYX; paste a private CA under *Trusted root certificates* in CERTYX.
2. For mutual TLS set `client_ca_file` to the CA of the CERTYX client certificate and enter that certificate and its key in CERTYX (mTLS); the bearer token is then optional.
3. Replace the `debug` exporter with the exporter of your backend.
4. In CERTYX: URL `https://collector.example.com:4318/v1/logs`, header name `Authorization` (the default), secret `Bearer <token>`.

18.3 Grafana Alloy and Loki

Grafana Alloy receives OTLP with `otelcol.receiver.otlp` and forwards to Loki's native OTLP endpoint (Loki 3, path `/otlp/v1/logs`; the exporter appends `/v1/logs`):

```
otelcol.receiver.otlp "certyx" {
  http {
    endpoint = "0.0.0.0:4318"
    tls {
      cert_file = "/etc/alloy/tls/server.crt"
      key_file  = "/etc/alloy/tls/server.key"
    }
```

```
  }
  output {
    logs = [otelcol.exporter.otlphttp.loki.input]
  }
}

otelcol.exporter.otlphttp "loki" {
  client {
    endpoint = "http://loki:3100/otlp"
  }
}
```

Loki keeps some resource attributes such as `service.name` as index labels and the other attributes as structured metadata, with dots replaced by underscores, for example `{service_name="certyx"} | certyx_event_code="AUTHN-002"`. CERTYX can also post directly to `https://<loki>/otlp/v1/logs` when Loki is reachable over TLS and needs at most one header; a multi-tenant Loki that additionally needs `X-Scope-OrgID` must be fronted by Alloy or the Collector.

18.4 Other OTLP backends

Datadog, Elastic and other vendors document their own OTLP intake. The generic, specification-based path is the OpenTelemetry Collector or Alloy above with the exporter the vendor documents. A backend that offers an OTLP/HTTP logs endpoint over HTTPS and authenticates with one header can be used directly: enter the logs URL, the header name and the header value exactly as the vendor documents them. CERTYX sends exactly one authentication header and no other vendor-specific headers.

18.5 CERTYX side

1. **Transport** *OpenTelemetry Logs (OTLP/HTTP)*; **Format** *OTLP protobuf* (recommended) or *OTLP/JSON*; profile *native*.
2. **HTTPS address**: the full logs URL of the receiver, for example `https://collector.example.com:4318/v1/logs`.
3. **Log record body**: *Message* (the event message) or *Canonical event (key-value list)*. All fields are attributes either way.
4. **Authentication header name** (optional; empty is `Authorization`) and **Authentication header value** as the secret (for example `Bearer <token>` or an API key). The value is stored encrypted and never shown again; without a value no header is sent. `Host`, `Content-Type`, `Content-Length`, `Content-Encoding`, `Transfer-Encoding`, `Connection`, `Cookie` and `Proxy-` headers are refused.
5. **Compress requests with gzip** (recommended for remote receivers).
6. Trusted roots, client certificate and revocation (TLS), *Send only*, field mapping and the delivery values work as for the other transports.
7. Save, **Check connection** and **Send test event**: the result `otlp-accepted` names the event ID; search the backend for `certyx.event.id` with that value.

19 Connection check and test event

Check connection runs the stages of the destination's transport without sending anything: DNS → TCP → TLS (with the client certificate and the revocation check) for every network transport; for a file archive the stage *Archive* checks that the archive directory can be written. **Send test event** runs the same stages and then delivers one synthetic event without moving the destination's position. The first failing stage ends the check. Each stage shows its result and the facts it found:

Stage	Passed	Facts shown
DNS	`dns-ok`, `dns-literal`	The addresses the host resolves to on the CERTYX server
TCP	`tcp-ok`	The connected address and port
TLS	`tls-ok`	TLS version, certificate subject, issuer and expiry

Stage	Passed	Facts shown
Delivery	`hec-accepted`, `syslog-written`, `http-accepted`, `otlp-accepted`	Splunk index and sourcetype; syslog framing; the test event's ID
Archive	`archive-ok`	Number of files, the space they use and the free space

The test event. Its `event_id` is a new UUID (it has no journal sequence), `event_code` `INTEGRATION-001`, `category` `integration`, `action` `siem.test`, `user_id` / `Actor` `siem-test`, outcome business; in the JSON record `Sequence` 0, `Method` `INTERNAL`, `Route` `business/siem.test`. It passes through the destination's format and profile like every other event, so it shows exactly what the receiver will get. The result names the event ID: search for it in the receiver.

For syslog destinations a successful test also states what the receiver must do with the chosen framing (octet counting: the receiving input must have octet counting on; newline: it must split at line breaks). When a receiver accepts TCP but does not complete the TLS handshake — its input has TLS off — the check reports `tls-timeout` or `tls-handshake-failed`, and the export records *The receiver accepted the TCP connection but did not complete a TLS handshake …* instead of a timeout. A file archive has no test event.

The result of the last check stays on the destination card and is written to the security journal (`siem.destination.checked`, `INTEGRATION-003`; `siem.destination.tested`, `INTEGRATION-002`). Tokens, secrets and response bodies never appear in it.

20 Delivery, retries and backlog

- **Durable queue.** The security journal is the queue: every destination has its own position (cursor) in it, which advances only after the receiver accepted a batch. Events are sent in journal order. Nothing is dropped when a receiver is unreachable.
- **At least once.** After a timeout or a lost connection the receiver may already have stored a batch that CERTYX sends again; the resent events carry the same `event_id`. Deduplicate on it (Deduplication).
- **Backoff.** After a failed delivery the destination waits before the next attempt: the **first retry delay** (default 30 seconds, `Clm:Siem:Delivery:InitialDelay`), doubled with every further consecutive failure up to the **longest retry delay** (default 15 minutes, `Clm:Siem:Delivery:MaxDelay`), each spread randomly by ±20 % (`Clm:Siem:Delivery:Jitter`) so destinations failing together do not retry together. A receiver that answers 429 or 503 with `Retry-After` (Splunk HEC, HTTPS, Sentinel, OTLP) is waited for at least that long, at most the longest retry delay. The first successful delivery resets the backoff.
- **Incident** `siem-delivery`**.** A failure raises this incident for the platform operator as a warning; after **Attempts before a critical alert** consecutive failures (default 10, `Clm:Siem:Delivery:MaxAttempts`) it becomes critical. It resolves after the next successful delivery.
- **Backlog.** When more events wait for a destination than its **Backlog alert above (events)** limit (default 100,000, `Clm:Siem:Delivery:BacklogLimit`), the incident `siem-backlog` is raised. Nothing is dropped (overflow policy *keep and alert*): the events stay in the journal and are sent when the destination accepts them again. The incident resolves when the backlog falls below the limit.
- **Batch size.** Up to 200 events per destination and pass (`Clm:Siem:BatchSize`); a console destination can set its own **Batch size (events)** (1–5000). For Splunk HEC keep a batch below the receiver's `max_content_length` (1,000,000 bytes is the smallest common setting).
- **Per destination.** Attempts, first retry delay (1–3600 s), longest retry delay (1–86400 s, not shorter than the first), batch size, backlog alert (1–100,000,000) and *Skip an event the receiver keeps rejecting after the attempts above* can be set in the destination form; empty fields use the server defaults. They do not apply to file archives.

20.1 Rejected events

Some answers say that the receiver refuses the data itself, whatever the retry: HTTP 400, 413 and 422, and the Splunk HEC codes 5 (no data), 6 (invalid data format), 12 (event field is required), 13 (event field cannot be blank) and 15 (error in handling indexed fields). Then CERTYX:

1. splits the batch in halves, recursively, until the single refused event is found;
2. delivers every event before it and stops the position just before it;
3. names its `event_id` in the last failure and in the `siem-delivery` incident (*The receiver rejects event …*).

Default: the refused event is retried with backoff and the incident stays open until the receiver accepts it (correct the receiver — for example the index, the field limits or the size limit — and the next attempt delivers it).

Opt-in *Skip an event the receiver keeps rejecting after the attempts above* **(destination form, or `Clm:Siem:Delivery:SkipRejected` for every destination): after Attempts before a critical alert** attempts the refused event is skipped, the position moves past it, and the skip is journaled as `siem.event.skipped` (code `INTEGRATION-011`) with the event ID, the sequence, the number of attempts and the reason. The event stays in the journal and in file archives; it can be sent again with Resend after the receiver is fixed.

Every other failure — unreachable receiver, TLS failure, HTTP 401, 403, 404, 429 or 5xx, other HEC codes — retries the whole batch.

20.2 Limits

- **Pace.** One pass every 30 seconds (`Clm:Siem:Interval`), 20-second timeout per request or connection (`Clm:Siem:HttpTimeout`); a destination in backoff is skipped until its delay is over.
- **Independence.** Each destination has its own position, backoff and status; a failing destination does not delay the others.
- **History.** A new destination starts at the beginning of the retained journal of its scope.
- **Not observed.** Sign-ins blocked before CERTYX (TLS handshake, firewall, rate limits at the ingress) cannot be exported; behind a proxy the source IP is the proxy's address.
- **Configuration destinations.** Destinations the operator defines in the server configuration are listed read-only in the view of their scope; they use the server delivery defaults. File archives can only be set up in the console.
- **Organization identification.** Every format carries the organization ID, name and kind and the instance (Several organizations in one SIEM).

21 Resend from a sequence or time

Resend from on a destination's card sends events again from a journal sequence or from a point in time, for example after the receiver lost data or after a rejected event was skipped.

1. Choose **Resend from** on the destination card.
2. Enter either a journal **sequence** (1 or higher; the part after the colon of an `event_id`) or a **time** — not both.
3. Confirm. The destination's position moves back to just before that point; the next passes send every event from there again, in order. The backoff and the rejected-event state are cleared.

Rules:

- The position only moves back, never forward: a point the destination has not delivered yet is refused (*the destination has not delivered that far yet*).
- A file archive cannot resend: it keeps every event; download its files instead.
- Each resend is journaled as `siem.destination.replayed` (code `INTEGRATION-012`) with the previous and the new position.
- The receiver gets the events a second time with the same `event_id`; deduplicate on it.
- Filters apply as for every pass: events the destination's *Send only* excludes are not sent.

Permissions. An organization Admin with an MFA session resends the organization's own destinations. The platform operator resends destinations of the *Platform operators* and *All events* scopes; this is a control-room change and asks for the operator step-up code. Mobile app sessions cannot resend.

API: `POST /api/administration/siem/{id}/replay` (organization) or `POST /platform/api/siem/{scope}/{id}/replay` (platform, scope `platform` or `all`, step-up code in `X-Step-Up-Code`) with `{"fromSequence": 48000}` or `{"fromTime": "2026-10-18T06:00:00Z"}`; the answer is `{"position": …}`.

22 Deduplication by event_id

Delivery is at least once: after a timeout, a lost connection, an unconfirmed HEC acknowledgement or a resend, the receiver can get an event twice. Both copies have the same event ID, which is unique across installations (`<installation id>:<journal sequence>`; a UUID for test events).

Receiver	Where the event ID is	How to deduplicate
Splunk	indexed field `event_id` (every HEC format), `event_id` in the canonical event	`dedup event_id` in searches; for summaries `stats latest(*) by event_id`
Elastic	`event.id` (ECS), `event_id` (native)	use it as the document `_id` (`document_id => "%{[event][id]}"` or the `fingerprint` filter); duplicates are refused with 409
QRadar	LEEF attribute `eventId`	custom event property `eventId`; QRadar does not deduplicate on its own — use it in searches and rules (for example *group by eventId*)
ArcSight	CEF `externalId` (`cef`), `certyxEventId` (`cef-compat`)	ArcSight does not drop duplicates on its own; group or filter by External ID in rules and reports
Sentinel	column `EventId`	`summarize arg_max(TimeGenerated, *) by EventId`
OCSF consumers	`metadata.uid`	key the stored record on it
Webhook / own collector	`EventId`, `event_id`, `event.id` or `metadata.uid` by profile	unique key in your store
File archive	`event_id` column (`csv`), `event_id` / `event.id` / `metadata.uid` (`ndjson`); `Sequence` (`csv-compat`)	archives never contain a row twice except after a crash at a day change (Delivery, restarts and disk space)

The installation ID part changes only when the operator changes `Clm:Siem:InstallationId`; keep it stable, or receivers no longer recognise resent events as duplicates.

23 Send only: less data to the SIEM

External SIEM and log services are often licensed by ingested volume. *Send only* reduces what one destination receives, while a file archive keeps the complete record on your own storage.

- **Outcomes** (check boxes): the destination receives the checked outcomes; checked choices are combined with *or*. *Configuration and business changes* adds every event that carries a configuration or business change (for example `siem.destination.updated`, user and role changes, renewal decisions, `agent.health.changed`), whatever its outcome check box. Example for a licence-saving SIEM feed: *Access denied*, *Server errors*, *Sign-in events* and *Configuration and business changes*.
- **Minimum severity** (list): only events at or above the severity are sent. Severity (0–10): received (`attempt`) 2, success and business change 3, rejected 5, denied and failed sign-in 7, server error and lockout 8.
- **Send only (categories)**: one or more of the ten categories.
- **Send only (event codes)**: codes from the event-code catalogue, for example `AUTHN-002`, at most 200.
- **Fields sent** (under *Shape and size*, folded): every field of the chosen format with a check box. The destination receives exactly the checked fields; an unchecked field is stored as `excludeFields` in the shaping. It works for every transport and format except the fixed `cef-compat` and `leef-compat` layouts (choose `cef` or `leef` there). Required stay on: for canonical formats the schema's required fields (`schema_version`, `event_id`, `event_code`, `category`, `action`, `outcome`, `outcome_original`, `severity`, `severity_label`, `message`, the three timestamps, `application`, `service_name`, `organization_kind`), for Sentinel `TimeGenerated`; in the `json` record every field can be removed. **Select all** and **Only required** set all boxes at once. *Preview test event* shows the result.
- **More filters** (folded): **Send only these routes** and **Never send these routes** (one per line; a route covers every route that starts with it, for example `/agent/v1/discovery/telemetry` for the agents' regular reports), **Never send events of these actors** (for example `pending`, requests before authentication), **Never send these categories** and **Never send these event codes**. An exclusion always wins.

- Health probes are never sent unless `Clm:Siem:ExportHealthProbes` is on.

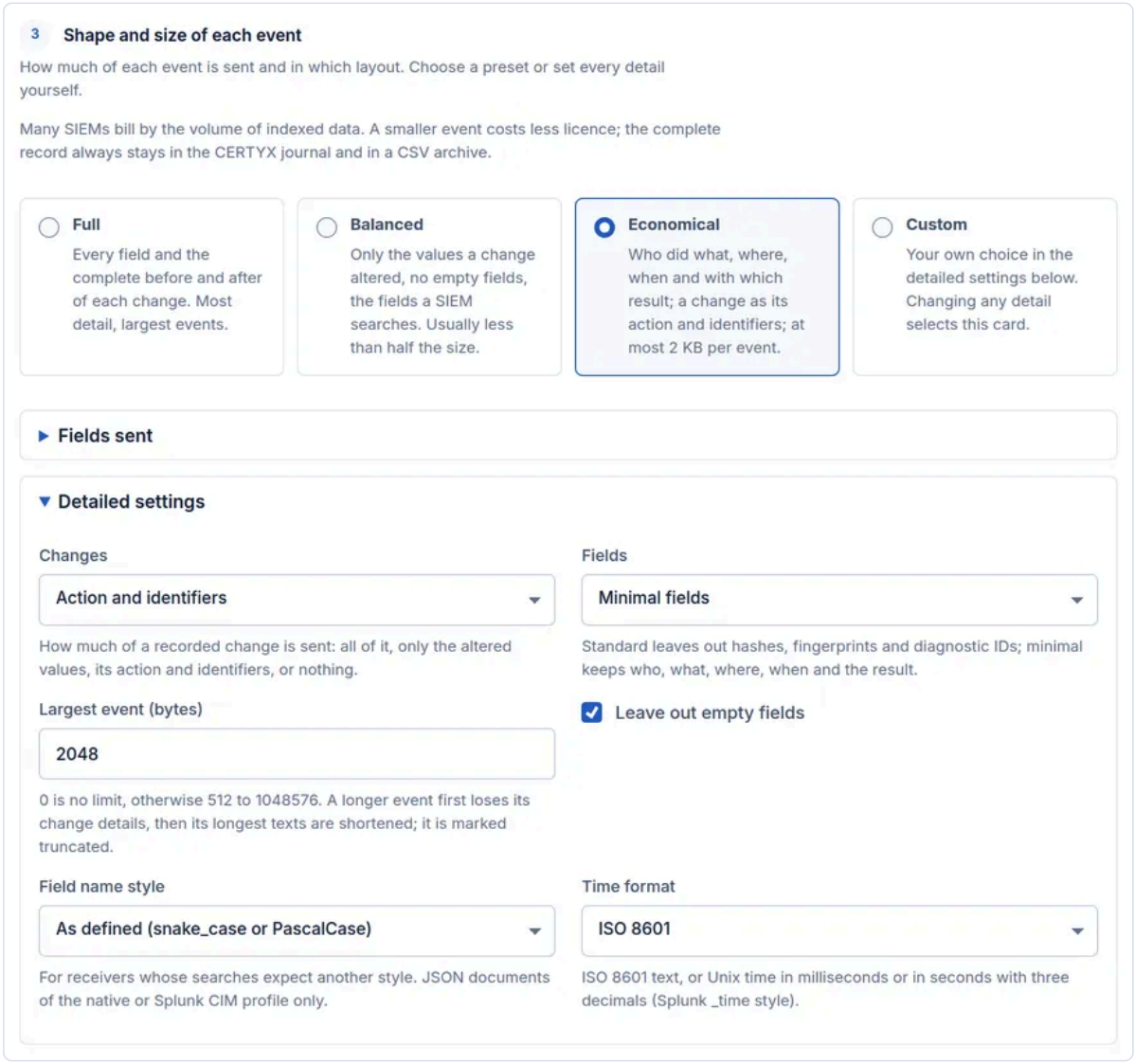
23.1 How the filters combine

1. Health probes are removed (unless exported).
2. Outcomes (*or* among the checked ones) and minimum severity select the events.
3. Categories and event codes narrow that result: an event passes when its category **or** its code is listed. With neither list set, every category and code passes.
4. *More filters* remove what is left out: routes outside *Send only these routes*, and every excluded route, actor, category and event code.

Example: outcomes empty (all), minimum severity 0, categories `authentication`, event codes `CONFIG-008` → every sign-in event and every SIEM destination change.

Events left out are not lost: the destination's position still advances past them (the journal keeps them), and a file archive of the same scope keeps them. Changing the filter applies to events exported from then on; already exported events are not sent again unless you resend.

24 Event shaping: size and layout



Event shaping: the preset cards and the detailed settings of a destination (sample data)

Many SIEM and log services charge by the volume of indexed data. *Send only* decides which events a destination receives; **event shaping** decides how large and in which layout each of them arrives. Every destination has its own shaping, so one SIEM can receive compact events while another receives everything. The complete record always stays in the CERTYX journal and in a file archive; shaping never changes the hash chain.

Presets (cards in section 3 of the form):

Preset	Changes	Fields	Empty fields	Largest event
Full (as before shaping existed)	complete before and after	all	sent	no limit
Balanced	only the changed values	standard	left out	no limit
Economical	action and identifiers	minimal	left out	2 KB
Custom	your choice	your choice	your choice	your choice

Choosing a card fills the **Detailed settings** with its values; changing any of them selects **Custom**. Measured on an agent observation of three certificates in the `json` format: Full 4,578 bytes, Balanced 591 bytes (−87 %), Economical 378 bytes (−92 %); a plain agent report: 706, 462 and 390 bytes. **Preview test event** shows the size of the test event with and without shaping.

Detailed settings:

- **Changes** — *Complete before and after*; *Only the changed values* (`before` and `after` are replaced by `changed`, for example `{"certificate.Owner":{"before":"Unassigned","after":"Team A"}}`; nested objects as dotted paths); *Action and identifiers* (the plain values of a change, every object and list left out); *No changes*.
- **Fields** — *All fields*; *Standard fields* (leaves out hashes, the agent fingerprint, span and request IDs, host IP and export bookkeeping); *Minimal fields* (who did what, where, when and with which result). The required canonical fields (`schema_version`, `event_id`, `event_code`, `category`, `action`, `outcome`, `outcome_original`, `severity`, `severity_label`, `message`, the three times, `application`, `service_name`, `organization_kind`) and `hostname` (it names the instance in the Splunk HEC envelope, the syslog header and the OTLP resource) are always sent, and a field the field mapping keeps or adds stays. In the `json` record the minimal set is `Sequence`, `At`, `Category`, `EventCode`, `Tenant`, `Actor`, `SourceIp`, `Method`, `Route`, `Status`, `Outcome`, `EventId` and `SignIn`; standard adds `OrganizationName`, `Instance`, `TraceId`, `ErrorType` and `Changes`.
- **Leave out empty fields** — fields without a value (null, empty text, empty list or object) are not written.
- **Largest event (bytes)** — 0 is no limit, otherwise 512 to 1,048,576. A longer event first keeps only the action and identifiers of its changes, then loses its changes, then its longest texts are shortened (ending in `…`); it then carries `truncated: true` (`Truncated` in the `json` record). The size is measured on the event document; line formats are a little shorter or longer.
- **Field name style** — *As defined*, `snake_case`, `camelCase` or `PascalCase` for the top-level fields (values inside changes and sign-in details keep their names). **Time format** — ISO 8601, Unix time in milliseconds, or in seconds with three decimals (Splunk `_time` style). Both apply only where a JSON document of the native or Splunk CIM profile is written (`json`, `ndjson`, `rfc5424`); ECS, OCSF and the line formats keep their own names and times.
- **Syslog MSGID** — *Event category* (default), *Event code*, *Your own text* (1–32 printable characters without spaces, for example `security-journal` to keep searches written before 1.2.19) or *None* (`-`); **Syslog APP-NAME** — empty sends `clm`. Only syslog destinations have them.

Shaping applies to every network destination and format; a file archive always keeps the complete record. Changing the shaping applies to events exported from then on. The saved destination card names the shaping, for example *Economical · exclusions: 1 · MSGID Event code*.

25 Field mapping

A network destination with a canonical format (`ndjson`, `rfc5424`, `cef`, `leef-1.0`, `leef-2.0`, `kv`, `otlp-protobuf`, `otlp-json`) can map the fields of the canonical event before any format writes them. The rules are part of the destination: saved with it, versioned with it (a save at an outdated version is refused) and journaled with the destination before and after the change (`siem.destination.updated`).

Rule	Effect
Rename (`rename`)	Moves a canonical field to a new name (a custom field)
Drop (`drop`)	Removes a canonical field (deny-list)
Keep (`keep`)	Allow-list: with at least one Keep rule every optional field that is neither kept nor renamed is removed
Add constant (`add`)	Adds a custom field with a fixed text of at most 256 characters

Keep rules apply first, then rename, drop and add in their order. The fields `schema_version`, `event_id`, `event_code`, `category`, `action`, `outcome`, `outcome_original`, `severity`, `severity_label`, `message`, `timestamp`, `observed_timestamp`, `export_timestamp`, `application`, `service_name` and `organization_kind` are never removed or renamed, so identification, time, category and outcome stay in every event. A new name has 1–64 characters, starts with a letter and uses letters, digits, underscores and dots (CEF: letters and digits only; ECS and OCSF: no dots); it must not start with `certyx`, repeat another new name or equal a canonical field or a key the format writes itself (for example `dst` in CEF or `server.address` in OTLP). At most 32 rules; each field appears in at most one rename or drop rule.

Where the result appears: removed fields are left out of every format (KV leaves their keys out); custom fields follow the format's own fields — inline in the JSON document (NDJSON, RFC 5424 MSG, HEC `event`), under `labels` in ECS (as text), under `unmapped` in OCSF, as extensions in CEF and attributes in LEEF, as keys in KV and as log record attributes in OTLP. The HEC indexed fields and the Splunk CIM fields keep their standard names. The JSON record, the Sentinel rows, the layouts of the first release and file archives (which keep the complete event) have no field mapping.

Preview. Preview test event sends the form as filled in, not yet saved, to the server, which renders the scope's test event after the rules in the chosen transport, format and encoding: webhook body, HEC envelope, syslog message, Sentinel row, OTLP request (protobuf shown in its OTLP/JSON encoding, with the size of the binary body) or archive line. Nothing is sent to the receiver and nothing is stored. An invalid rule is refused with the reason.

API: the save body carries the rules in `fieldMapping`, for example

```
"fieldMapping": [
  {"op": "rename", "field": "user_id", "to": "account"},
  {"op": "drop", "field": "host_ip"},
  {"op": "add", "to": "region", "value": "eu-central"}
]
```

and `POST /api/administration/siem/preview` (organization) or `POST /platform/api/siem/{scope}/preview` (platform; no step-up code, it changes nothing) takes the same body as a save and answers `{transport, format, profile, eventId, contentType, payload, bytes, canonical}`.

Format `kv` with the rules above (user_id renamed to `account`, `host_ip` dropped, `region` added), test event:

```
<86>1 2026-10-19T04:20:37.893Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM"
swVersion="1.0.0"] schema_version=certyx.event/1.0 event_id=f4293b79-ca2c-4fd2-a9d5-6490b28aee56 event_code=INTEGRATION-001
category=integration action=siem.test outcome=success outcome_original=business severity=3 severity_label=low
message="siem.test by siem-test" timestamp=2026-10-19T04:20:37.893Z observed_timestamp=2026-10-19T04:20:37.893Z
export_timestamp=2026-10-19T04:20:37.893Z application="CERTYX CLM" version=1.0.0 environment=production hostname=certyx-node-1
service_name=clm component=siem organization_id=platform organization_name=- organization_kind=platform username=- user_role=-
session_id=- authentication_method=- user_verified=- user_agent=- src_ip=- src_port=- http_method=- http_route=- http_status=-
protocol=- correlation_id=f4293b79-ca2c-4fd2-a9d5-6490b28aee56 request_id=f4293b79-ca2c-4fd2-a9d5-6490b28aee56 trace_id=-
resource_type=- resource_id=- change_type=- reason=- error_type=- sequence=- event_hash=- agent_fingerprint=- changes="
[{\"action\":\"siem.test\",\"requestedBy\":\"admin\"}]" changes_redacted=- dst_ip=- dst_port=- span_id=- account=siem-test
region=eu-central
```

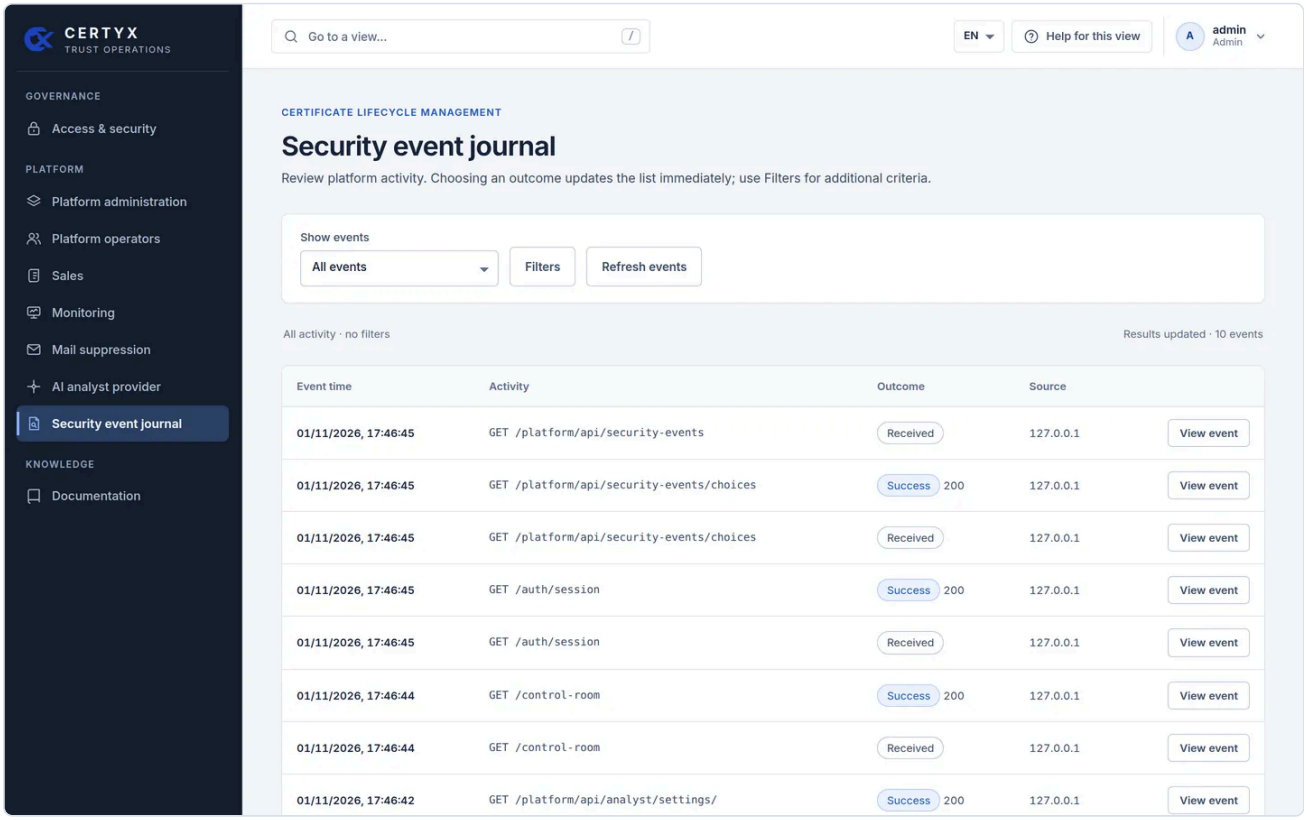
26 Enrichment fields

Since this release CERTYX records these facts about each request beside the journal entry (in a separate table keyed by the sequence, outside the hash chain, so the journal hash stays verifiable):

Field	Content	Notes
`trace_id`	W3C trace ID of the request activity	An incoming `traceparent` header is continued, so the ID matches the caller's trace
`src_port`	Client source port	Behind a proxy the proxy's port
`host_ip`	Server address that accepted the connection	Behind a proxy or in a container network the local interface address; RFC 5424 `origin ip`
`dst_port`	Server port that accepted the connection	`dst_ip` is the same address as `host_ip`; behind a reverse proxy or in a container network the local listener port (for example 8080), not the public port
`span_id`	W3C span ID of the request activity	With `trace_id` the TraceId and SpanId of an OTLP log record
`user_role`	Role claim of the signed-in principal	
`session_id`	Session correlation ID: for a server-side session (console or mobile app) the first 32 hexadecimal characters of SHA-256 of `clm-session` and the SHA-256 of the session identifier; for an older browser session of SHA-256 of `clm-session`, the subject and the sign-in time	The same for every request of one session; never the session cookie, the session identifier or the stored session key

Availability. Entries recorded before this release, business entries (changes outside a request) and test events have no enrichment; their fields are unknown (left out in JSON, empty in CSV, `-` in KV). Requests of agents and API keys, and mobile app sessions started before this release, carry only part of them (no `session_id`). `dst_port` and `span_id` are recorded from the OpenTelemetry release on. A forwarded host header is never used for the destination, because a client can set it. Agent and endpoint events that name an observed target (a discovery alert names its endpoint `host:port`) take `dst_ip` and `dst_port` from that target; a host name gives no `dst_ip`. These fields appear in the canonical formats and profiles only; the first-release layouts are unchanged.

27 Several organizations in one SIEM



Security event journal of the control room, where the platform-wide SIEM destinations are managed (sample data)

Every exported event identifies its organization, so several organizations — or the operator's *All events* scope — can share one SIEM index. The fields are added when the event is exported; the hash-chained journal record itself is not changed, so the hash stays verifiable.

Meaning	Canonical	JSON record / HEC (`cef-compat`)	CEF (`cef-compat`)	LEEF	Sentinel	CSV (`csv-compat`)
Organization ID	`organization_id`	`Tenant`	`cs1`	`organization`	`Organization`	`OrganizationId`
Organization name	`organization_name`	`OrganizationName`	`cs2`	`organizationName`	`OrganizationName`	`OrganizationName`
Kind: `organization`, `platform`, `unattributed`	`organization_kind`	`OrganizationKind`	—	`organizationKind`	`OrganizationKind`	`OrganizationKind`
CERTYX instance	`hostname`	`Instance` (HEC also `host`)	`dvchost`	`instance`	`Instance`	`Instance`

In `cef-compat` the organization is `cs1` and its name and kind are `certyxOrganizationName` and `certyxOrganizationKind`. `platform` marks operator events and platform sign-in failures; `unattributed` marks requests without a signed-in organization (before sign-in, anonymous calls, health probes). The name is the organization's name in the platform's customer list; it is empty (unknown) when the installation has no platform administration. Use the ID, not the name, for access rules and deduplication — a name can change.

Splunk:

```
index=clm sourcetype=certyx:event organization_kind=organization
| stats count by organization_name, organization_id, category
```

Cribl, to route organizations to their own index: an *Eval* with `index = 'clm_' + organization_id.substring(0, 8)` (or a lookup from the organization ID to an index), placed after the JSON *Parser*. Splunk role-based access can then restrict each organization's users to its index.

28 File archives

The `file` transport writes every event of its scope to daily files on the CERTYX server instead of sending it over the network. Use it to keep the full security record for a retention period outside the SIEM, and send the SIEM only what it needs (Send only).

Format × profile	File name	Content
`csv-compat`	`certyx-security-YYYY-MM-DD.csv`	The columns of the first release (below)
`csv`	`certyx-events-YYYY-MM-DD.csv`	Canonical columns in the fixed order of the schema version, `schema_version` first, `event_id` second
`ndjson` × `native` / `ecs` / `ocsf`	`certyx-events-YYYY-MM-DD.ndjson`	One document of the profile per line

28.1 Setting it up

1. **Organization:** an Admin signed in with MFA opens **SIEM export**. **Platform:** the operator opens **Security event journal** → **SIEM export** and chooses the scope *Platform operators* or *All events*.
2. Destination name and identifier (for example `csv-archive`), Transport **File archive**, Format `csv-compat`, `csv` or `ndjson` (with its profile).
3. **Retention (days)**: 1–3650. Daily files older than this are deleted automatically.
4. Optional **Compress closed days with gzip** (`.csv.gz`, `.ndjson.gz`).
5. Save, then **Check connection**: it checks that the archive directory can be written and shows the number of files, the space they use and the free space on the volume.

An archive keeps every event: it has no outcome, severity, category or code filter, no delivery values, no client certificate and no test event. Each archive counts toward the destination limit of its scope.

Location. The organization cannot choose a path: its files live in an area the platform operator configures, `<Clm:Siem:ArchiveRoot>/organizations/<organization ID>/<archive identifier>/` (default `ArchiveRoot`: `siem-archive` in the CERTYX data directory). Platform-wide archives live in `<Clm:Siem:PlatformArchiveDirectory>/<platform|all>/<archive identifier>/` (default `<ArchiveRoot>/platform`). Directories and files are created readable only by the CERTYX service account; a path that contains a symbolic link is refused.

28.2 Files

- One file per UTC day on which events were written. A new archive's first file holds the retained journal history, then each day's events.
- **CSV (`csv-compat`, `csv`)**: RFC 4180, UTF-8 without byte order mark, a header row, comma separator, CRLF line ends, quotes doubled. Control characters inside a value become spaces (`csv-compat`) or stay inside the quotes (`csv`), so every record is one CSV record. `csv-compat` quotes every cell. `csv` distinguishes unknown from empty: an unknown value is an empty unquoted field, a known empty value is `""`.
- **Formula-injection safe:** a cell that starts with `=`, `+`, `-`, `@` (also after spaces), a tab or a carriage return is prefixed with `'`, so spreadsheet programs show it as text.
- `csv-compat` columns: `Sequence`, `At` (UTC), `OrganizationId`, `OrganizationName`, `OrganizationKind`, `Instance`, `Actor`, `SourceIp`, `AgentFingerprint`, `Method`, `Route`, `Status`, `Outcome`, `Severity`, `TraceId`, `ErrorType`, `SignInAction`, `SignInUser`, `SignInMethod`, `SignInResult`, `SignInReason`, `Changes` (JSON array), `PreviousHash`, `Hash`.
- `csv` columns: the canonical fields in the order of Field reference; a new field is only ever appended with a new schema version.
- **NDJSON (`ndjson`)**: UTF-8 without byte order mark, one JSON document per line ended by LF, no header.
- **Sealing and integrity.** When the UTC day is over, the day's file is sealed at the next export pass: compressed with gzip when compression is on, and its SHA-256 is added to the manifest `SHA256SUMS` in the same directory. Check a downloaded archive with `sha256sum -c SHA256SUMS` (the current day's file is not in the manifest until it is sealed). In `csv-compat`, `Hash`

and `PreviousHash` link each row to the journal chain verified in **Security event journal** → **Verify integrity**; the canonical formats carry `event_hash`.

- **Retention.** At every export pass, files of days older than the retention are deleted with their manifest lines; each deletion is journaled as `siem.archive.expired` (code `INTEGRATION-009`) with the file names (actor `system:siem-archive`).

28.3 Download

On the archive's card, **Archive files** lists every file with its size, whether it is sealed and its SHA-256. **Download** fetches one file, **Download all (zip)** every file with the manifest. Downloads require a session signed in with MFA (as for changes) and are refused for mobile app sessions; each download is journaled as `siem.archive.downloaded` (code `DATA-003`) with the file names. Listing the files needs only the Admin role (organization) or platform administration (operator). The operator can also copy the files directly from the archive volume.

28.4 Delivery, restarts and disk space

- The archive uses the same durable position as the network destinations: rows are written and flushed to disk, then the position advances (at least once, nothing skipped). After a crash between the two, the next pass first reads the last complete row of the open file (its sequence, from `Sequence` or from the event ID), removes a row cut by the crash and continues after it, so a file never contains a row twice. A row can appear in two files only if the day changed during such a crash.
- Up to `Clm:Siem:BatchSize` events (default 200) per pass, one pass every `Clm:Siem:Interval` (default 30 seconds); a new archive catches up with the history at that pace.
- **Disk space:** plan roughly 0.5–1 KB per event uncompressed for `csv-compat`, more for `csv` and `ndjson` (more fields); gzip typically reduces it to a tenth. Monitor the volume — a full disk or a missing volume makes the archive fail with *The CSV archive directory can not be written (...)*, raises the `siem-delivery` incident and keeps the position, so the archive continues after the space is freed.
- In a multi-instance installation every instance must mount the same archive directory (the export runs on one instance at a time, but not always the same one).
- Removing the archive destination stops writing and deletes its position; the files stay in the directory until the operator deletes them. Adding it again with the same identifier continues in the same directory.

29 Permissions

Action	Organization destinations	Platform scopes (*Platform operators*, *All events*)
View destinations, status, archive file list	Admin role	Platform administration
Check connection, send test event	Admin with an MFA session	Platform operator with an MFA session
Save, edit, remove, new secret	Admin with an MFA session	Platform operator with an MFA session and the operator step-up code
Resend from a sequence or time	Admin with an MFA session	Platform operator with an MFA session and the operator step-up code
Download archive files	Admin with an MFA session	Platform operator with an MFA session

Mobile app sessions cannot change, test, resend or download. Every change, check, test, resend, skip and download is written to the security journal (codes `CONFIG-008`, `INTEGRATION-002`, `INTEGRATION-003`, `INTEGRATION-012`, `INTEGRATION-011`, `DATA-003`).

30 Troubleshooting

30.1 Incidents

Incident	Severity	Meaning	What to do
`siem-delivery`	warning; critical after *Attempts before a critical alert* consecutive failures	A destination's last delivery failed; the reason is on the card (and names the event ID for a rejected event)	Run *Check connection*; fix the receiver; for a rejected event see Rejected events
`siem-backlog`	warning	More events wait than the destination's backlog alert limit; nothing is dropped	Fix the delivery failure, or raise the batch size or the limit when the receiver is merely slow

Both incidents resolve on their own: `siem-delivery` after the next successful delivery, `siem-backlog` when the backlog falls below the limit.

30.2 Check and test results

Code	Meaning	What to do
`dns-failed`	The CERTYX server cannot resolve the host name	Correct the name, or add it to the DNS the CERTYX server uses; internal names must resolve on the server, not only on your computer
`dns-private`	The host resolves to a private, loopback or link-local address and the operator restricted organization destinations to public addresses	Use a public address, or ask the operator to allow private destinations (`Clm:Siem:AllowPrivateDestinations`)
`tcp-refused`	Nothing listens on the port, or a firewall rejects the connection	Enable the receiving input on this port; check the port number
`tcp-unreachable`	No network route from the CERTYX server to the address	Check routing and firewalls between the server and the receiver
`tcp-timeout`	No answer within the timeout (10 seconds by default)	A firewall probably drops the connection; allow TCP from the CERTYX server
`tls-timeout`	TCP connected, but no TLS handshake	The input is plain TCP; enable TLS on it (Cribl: TLS settings of the source; Splunk: `[tcp-ssl:<port>]` or HEC with Enable SSL)
`tls-handshake-failed`	The receiver closed the connection during the handshake	Enable TLS on the input; offer TLS 1.2 or 1.3; when the receiver requires a client certificate, add one (mTLS) or check that it is issued by a CA the receiver trusts
`tls-name-mismatch`	The certificate does not name the host entered	Enter a host name the certificate names, or install a certificate for that name (Splunk's `SplunkServerDefaultCert` never matches)
`tls-expired`	The receiver's certificate is outside its validity period	Renew the receiver's certificate
`tls-untrusted`	The certificate does not chain to the system trust store, or it is revoked (with hard-fail: its revocation status is unknown)	Paste the root CA certificate (PEM) in Trusted root certificates; replace a revoked certificate; with hard-fail make the CRL/OCSP address reachable or choose soft-fail
`tls-untrusted-roots`	The certificate does not chain to the roots pasted for this destination, or fails the revocation check	Paste the root that actually issued the receiver's certificate; as above for revocation
`hec-refused`	Splunk refused the event with an HEC code	See the HEC code table below
`hec-status`	Splunk answered an HTTP error without an HEC code (often a proxy or a wrong path)	Use `/services/collector/event` ; check proxies in between

Code	Meaning	What to do
`hec-not-acknowledged`	Splunk accepted the batch but did not confirm indexing within 30 seconds	Check indexer health and acknowledgement on the token; the batch is sent again
`entra-refused`	Microsoft Entra refused the application credentials	Check tenant ID, application (client) ID and the secret's Value (not its ID); renew an expired secret
`http-status`	The receiver answered an HTTP error	400/422: the receiver refuses the data (see Rejected events); 401/403: enter the token or secret again, for Sentinel check the role assignment; 404: wrong path, DCR immutable ID or stream; 413: the batch is too large for the receiver — lower the batch size; 429/503: busy, retried with backoff; 5xx: check the receiver's logs
`delivery-failed`	The connection or write failed after the TLS stage	See the reason shown (for example *Destination did not answer in time.*)
`archive-unwritable`	The archive directory cannot be written	Ask the operator to check the archive volume, its permissions and free space (`Clm:Siem:ArchiveRoot`)
`archive-link`	The archive path contains a symbolic link	The operator must configure an archive path without symbolic links

30.3 Receiver shows fragments or binary data

What you see in the SIEM (raw)	Cause	What to do
Records that start with a number and a space (`412 <84>1 2026-…`), a number glued to the end of a record (`… }4`), records cut in the middle (`<86>1 2026-10-11T1`)	CERTYX uses octet counting, the receiver splits at line feeds (Cribl syslog source with *Octet count framing* No, the default)	Set *Octet count framing* Yes on the source, or choose *Newline* framing in CERTYX — the same on both sides
Binary events starting with `\x16\x03` (often shown as `\x16\x03\x01…`) that contain the receiver's host name (for example `cribl-worker1…`)	The receiving port has TLS off; it indexes CERTYX's TLS ClientHello	Enable TLS on the source/input (Cribl: *TLS Settings → Enabled*); CERTYX never sends plain TCP
, `ï»¿` or before the JSON of an `rfc5424` message	The receiver shows the UTF-8 byte order mark (RFC 5424 MSG-UTF8) as text	Strip it in the receiver's parser, or ask the operator to set `Clm:Siem:Syslog:ByteOrderMark=false`
Events from `/health/ready` with `Tenant unattributed` every few seconds	A load balancer probes readiness; versions before this release exported the probes	Current versions do not send probes (`Clm:Siem:ExportHealthProbes` , default off)
Each request appears twice (`attempt` , then `success` / `denied` /…)	*Received requests* is checked in *Send only*	Uncheck it; with no box checked the destination receives one record per request
The same event appears twice with the same event ID	At-least-once delivery after a timeout or a resend	Deduplicate on the event ID (Deduplication)
`host` or `source` values such as `http:clm` or the Cribl worker address	Set by the forwarding pipeline (for example Cribl's HEC output), not by CERTYX	Use `hostname` / `Instance` and the organization fields from the event, or set `host` with an Eval in Cribl
Splunk fields with two identical values (`event_id` , `action`)	The field is both an HEC indexed field and part of the JSON event	See props.conf and Splunk CIM

30.4 Splunk HEC codes

Code	Splunk text	What to do
1	Token disabled	Enable the token in *HTTP Event Collector*
2	Token is required	Enter the HEC token
3	Invalid authorization	Enter the token value exactly as Splunk shows it
4	Invalid token	The stored token is unknown to Splunk; enter a valid token
5	No data	Treated as a rejection of the data (Rejected events); check proxies that could strip the body
6	Invalid data format	Use `/services/collector/event` (not `/raw`); treated as a rejection of the data

Code	Splunk text	What to do
7	Incorrect index	Add the index to the token's Allowed indexes, or correct the index name
8	Internal server error	Check the Splunk indexer health
9	Server is busy	Splunk is overloaded; delivery is retried with backoff
10	Data channel is missing	A proxy probably removes the `X-Splunk-Request-Channel` header
11	Invalid data channel	Disable and re-enable indexer acknowledgement on the token
12, 13	Event field is required / cannot be blank	Use `/services/collector/event` ; treated as a rejection of the data
14	ACK is disabled	No action needed; delivery is not affected
15	Error in handling indexed fields	Treated as a rejection of the data; check limits on indexed fields in Splunk
16	Query string authorization is not enabled	No action needed; CERTYX uses header authorization
18, 19, 20	HEC is unhealthy	Queues full or acknowledgement unavailable; delivery is retried with backoff

30.5 Delivery status on the destination card

Last failure	What to do
Destination could not be reached.	Run Check connection; it names the failing stage
TLS validation of the destination failed.	Run Check connection; see the TLS codes above
Destination did not answer in time.	The receiver did not answer within 20 seconds; check its load or the network
HTTP <status>	See `http-status` above
Splunk HEC answered HTTP <status> with code <n> (...)	See the HEC code table
The receiver rejects event <event ID>: ...	See Rejected events
Microsoft Entra refused the application credentials.	See `entra-refused`
The stored secret can not be decrypted; enter it again.	Edit the destination and enter the secret again (the server's key ring changed)
The client certificate or its private key can not be read; enter both again.	Edit the destination and paste the client certificate and its key again
The destination resolves to a private, loopback or link-local address; ...	See `dns-private`
The receiver accepted the TCP connection but did not complete a TLS handshake ...	The receiving input has TLS off; see Receiver shows fragments or binary data
The CSV archive directory can not be written (...)	Ask the operator to check the archive volume, its permissions and free space (`Clm:Siem:ArchiveRoot`)
The CSV archive path contains a symbolic link; ...	The operator must configure an archive path without symbolic links

A failed batch is retried after the backoff delay and raises the `siem-delivery` incident (Delivery). The position does not advance until the receiver accepted the batch.

30.6 Form and permission messages

- Every refused field names the expected format, for example *Syslog host: enter only the host name and put the port (for example 10514) in the Port field.*
- *Format, schema profile and transport: choose a transport, then one of the formats and profiles offered for it.* — the combination is not allowed (Allowed combinations).
- *LEEF delimiter: ...* — only `leef-2.0` declares a delimiter; TAB or a punctuation character other than `=`, the pipe and the backslash, or its hexadecimal code.

- *Send only (categories) | Send only (event codes)* — only categories from the list; codes from the catalogue, at most 200.
- *Attempts before a critical alert*, *First retry delay (seconds)*, *Longest retry delay (seconds)*, *Batch size (events)*, *Backlog alert above (events)* — whole numbers in the ranges of Delivery; the longest delay is not shorter than the first.
- *Client certificate: …* — one PEM certificate with a matching unencrypted PEM private key, at most 64 KiB each; a new certificate needs its key.
- *A file archive keeps every event: delivery retries, rejected-event skipping and the client certificate do not apply.*
- *Resend from: …* — a sequence (1 or higher) or a time, not both, at or before the destination's position; not for file archives.
- *Enter the secret again when changing the format.* — shown when the transport changes; a stored secret belongs to its transport.
- *The destination changed meanwhile, already exists, or the destination limit is reached.* — reload the view; remove an unused destination when the limit is reached.
- *Enter the secret before sending a test event.* — webhook, Splunk HEC and Sentinel need their secret first.
- The MFA notice names the reason: no confirmed authenticator, enrolment not confirmed, session started before MFA, company sign-in without reported MFA, or password-only session. Follow the instruction shown, then sign in again.

31 Sample events

Each sample is the test event of a running CERTYX server in that format and profile (event code `INTEGRATION-001`); installation-specific values differ in your installation.

The samples were captured on 2026-10-19 from a CERTYX server with the default syslog settings (facility authpriv, APP-NAME `clm`, no enterprise number, so no `certyx@…` element) and `Clm:Siem:Environment=production`. Syslog samples show one message without its octet-count prefix; `<BOM>` stands for the UTF-8 byte order mark. A file archive receives no test event, so its samples are the first journal entries the export wrote.

31.1 HTTPS webhook

Format `json` (JSON array of records):


```
[{"Sequence":0,"At":"2026-10-19T04:20:31.2998647+00:00","Category":"integration","EventCode":"INTEGRATION-
001","Tenant":"platform","OrganizationName":"","OrganizationKind":"platform","Instance":"certyx-node-1","Actor":"siem-
test","SourceIp":"system","AgentFingerprint":"","Method":"INTERNAL","Route":"business/siem.test","Status":200,"Outcome":"busin
ess","TraceId":"ecb63792-0af6-4902-b5c1-4a8b409b82ce","ErrorType":"","Changes":
[{"action":"siem.test","requestedBy":"admin"}],"Hash":"","EventId":"ecb63792-0af6-4902-b5c1-4a8b409b82ce"}]
```


Format `ndjson`, profile `native`:


```
{"schema_version":"certyx.event/1.0","event_id":"c6030d2e-e437-4a96-a118-688cc908e6f9","event_code":"INTEGRATION-
001","category":"integration","action":"siem.test","outcome":"success","outcome_original":"business","severity":3,"severity_la
bel":"low","message":"siem.test by siem-test","timestamp":"2026-10-19T04:20:31.764Z","observed_timestamp":"2026-10-
19T04:20:31.764Z","export_timestamp":"2026-10-19T04:20:31.764Z","application":"CERTYX
CLM","version":"1.0.0","environment":"production","hostname":"certyx-node-
1","service_name":"clm","component":"siem","organization_id":"platform","organization_kind":"platform","user_id":"siem-
test","correlation_id":"c6030d2e-e437-4a96-a118-688cc908e6f9","request_id":"c6030d2e-e437-4a96-a118-688cc908e6f9","changes":
[{"action":"siem.test","requestedBy":"admin"}]}
```


Format `ndjson`, profile `ecs`:


```
{"@timestamp":"2026-10-19T04:20:32.164Z","message":"siem.test by siem-test","ecs":{"version":"8.11.0"},"event":
{"id":"81e697f9-214e-46c7-9b22-785f48e900a6","code":"INTEGRATION-001","kind":"event","category":["network"],"type":
["connection"],"action":"siem.test","outcome":"success","severity":3,"created":"2026-10-
19T04:20:32.164Z","dataset":"certyx.security","provider":"siem"},"user":{"id":"siem-test"},"host":{"hostname":"certyx-node-
```

```
1"},"service":{"name":"clm","version":"1.0.0","environment":"production"},"organization":{"id":"platform"},"labels":
{"schema_version":"certyx.event/1.0","certyx_category":"integration"}}
```


Format `ndjson`, profile `ocsf`:


```
{"class_uid":0,"category_uid":0,"activity_id":99,"type_uid":99,"severity_id":1,"status_id":1,"time":1792383632558,"message":"s
iem.test by siem-test","metadata":{"version":"1.1.0","uid":"d2bbe2f2-be7a-4963-903a-94a2c7790742","event_code":"INTEGRATION-
001","log_name":"security-journal","product":
{"name":"CLM","vendor_name":"CERTYX","version":"1.0.0"},"correlation_uid":"d2bbe2f2-be7a-4963-903a-94a2c7790742"},"unmapped":
{"certyx_schema_version":"certyx.event/1.0","certyx_category":"integration","certyx_action":"siem.test"}}
```


31.2 Splunk HEC

Format `json` (record of the first release, indexed field `event_id`):


```
{"time":1792383633.006,"host":"certyx-node-1","source":"certyx:security-journal","sourcetype":"certyx:security","event":
{"Sequence":0,"At":"2026-10-19T04:20:33.0060384+00:00","Category":"integration","EventCode":"INTEGRATION-
001","Tenant":"platform","OrganizationName":"","OrganizationKind":"platform","Instance":"certyx-node-1","Actor":"siem-
test","SourceIp":"system","AgentFingerprint":"","Method":"INTERNAL","Route":"business/siem.test","Status":200,"Outcome":"busin
ess","TraceId":"94ac6f46-f534-4fdc-95a5-ff0199cbc888","ErrorType":"","Changes":
[{"action":"siem.test","requestedBy":"admin"}],"Hash":"","EventId":"94ac6f46-f534-4fdc-95a5-ff0199cbc888"},"fields":
{"event_id":"94ac6f46-f534-4fdc-95a5-ff0199cbc888"}}
```


Format `ndjson`, profile `native`:


```
{"time":1792383633.454,"host":"certyx-node-1","source":"certyx:security-journal","sourcetype":"certyx:event","event":
{"schema_version":"certyx.event/1.0","event_id":"cbd516bc-2c12-497d-909d-66417ae5b605","event_code":"INTEGRATION-
001","category":"integration","action":"siem.test","outcome":"success","outcome_original":"business","severity":3,"severity_la
bel":"low","message":"siem.test by siem-test","timestamp":"2026-10-19T04:20:33.454Z","observed_timestamp":"2026-10-
19T04:20:33.454Z","export_timestamp":"2026-10-19T04:20:33.454Z","application":"CERTYX
CLM","version":"1.0.0","environment":"production","hostname":"certyx-node-
1","service_name":"clm","component":"siem","organization_id":"platform","organization_kind":"platform","user_id":"siem-
test","correlation_id":"cbd516bc-2c12-497d-909d-66417ae5b605","request_id":"cbd516bc-2c12-497d-909d-66417ae5b605","changes":
[{"action":"siem.test","requestedBy":"admin"}]},"fields":{"event_id":"cbd516bc-2c12-497d-909d-
66417ae5b605","event_code":"INTEGRATION-001","category":"integration"}}
```


Format `ndjson`, profile `ecs`:


```
{"time":1792383633.853,"host":"certyx-node-1","source":"certyx:security-journal","sourcetype":"certyx:event","event":
{"@timestamp":"2026-10-19T04:20:33.853Z","message":"siem.test by siem-test","ecs":{"version":"8.11.0"},"event":
{"id":"4bdad006-45f0-494e-a088-1857180b973c","code":"INTEGRATION-001","kind":"event","category":["network"],"type":
["connection"],"action":"siem.test","outcome":"success","severity":3,"created":"2026-10-
19T04:20:33.853Z","dataset":"certyx.security","provider":"siem"},"user":{"id":"siem-test"},"host":{"hostname":"certyx-node-
1"},"service":{"name":"clm","version":"1.0.0","environment":"production"},"organization":{"id":"platform"},"labels":
{"schema_version":"certyx.event/1.0","certyx_category":"integration"}},"fields":{"event_id":"4bdad006-45f0-494e-a088-
1857180b973c","event_code":"INTEGRATION-001","category":"integration"}}
```


Format `ndjson`, profile `ocsf`:


```
{"time":1792383634.227,"host":"certyx-node-1","source":"certyx:security-journal","sourcetype":"certyx:event","event":
{"class_uid":0,"category_uid":0,"activity_id":99,"type_uid":99,"severity_id":1,"status_id":1,"time":1792383634227,"message":"s
iem.test by siem-test","metadata":{"version":"1.1.0","uid":"05b4a4e8-c15b-439f-8051-d00980aca411","event_code":"INTEGRATION-
001","log_name":"security-journal","product":
{"name":"CLM","vendor_name":"CERTYX","version":"1.0.0"},"correlation_uid":"05b4a4e8-c15b-439f-8051-d00980aca411"},"unmapped":
```

{"certyx_schema_version":"certyx.event/1.0","certyx_category":"integration","certyx_action":"siem.test"}},"fields":
{"event_id":"05b4a4e8-c15b-439f-8051-d00980aca411","event_code":"INTEGRATION-001","category":"integration"}}

Format `ndjson`, profile `splunk-cim`:

{"time":1792383634.591,"host":"certyx-node-1","source":"certyx:security-journal","sourcetype":"certyx:event","event":
{"schema_version":"certyx.event/1.0","event_id":"38cd7f4b-5f1b-44c3-909c-695c0c9caf3c","event_code":"INTEGRATION-
001","category":"integration","action":"siem.test","outcome":"success","outcome_original":"business","severity":3,"severity_la
bel":"low","message":"siem.test by siem-test","timestamp":"2026-10-19T04:20:34.591Z","observed_timestamp":"2026-10-
19T04:20:34.591Z","export_timestamp":"2026-10-19T04:20:34.591Z","application":"CERTYX
CLM","version":"1.0.0","environment":"production","hostname":"certyx-node-
1","service_name":"clm","component":"siem","organization_id":"platform","organization_kind":"platform","user_id":"siem-
test","correlation_id":"38cd7f4b-5f1b-44c3-909c-695c0c9caf3c","request_id":"38cd7f4b-5f1b-44c3-909c-695c0c9caf3c","changes":
[{"action":"siem.test","requestedBy":"admin"}]},"fields":{"event_id":"38cd7f4b-5f1b-44c3-909c-
695c0c9caf3c","event_code":"INTEGRATION-001","category":"integration","vendor_product":"CERTYX
CLM","app":"clm","signature":"siem.test by siem-test","signature_id":"INTEGRATION-
001","severity":"informational","severity_id":"3","dest":"certyx-node-1","user":"siem-test","user_id":"siem-
test","action":"siem.test"}}

31.3 Syslog over TLS

Format `json` (first release, fixed header):

<86>1 2026-10-19T04:20:34.960Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"] {"Sequence":0,"At":"2026-10-
19T04:20:34.9600107+00:00","Category":"integration","EventCode":"INTEGRATION-
001","Tenant":"platform","OrganizationName":"","OrganizationKind":"platform","Instance":"certyx-node-1","Actor":"siem-
test","SourceIp":"system","AgentFingerprint":"","Method":"INTERNAL","Route":"business/siem.test","Status":200,"Outcome":"busin
ess","TraceId":"e4828ba9-d4aa-46ef-bff7-82465c8f9598","ErrorType":"","Changes":
[{"action":"siem.test","requestedBy":"admin"}],"Hash":"","EventId":"e4828ba9-d4aa-46ef-bff7-82465c8f9598"}

Format `rfc5424`, profile `native`:

<86>1 2026-10-19T04:20:35.330Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM"
swVersion="1.0.0"] <BOM>{"schema_version":"certyx.event/1.0","event_id":"75620e9f-9fe1-4f59-84d4-
5c1263f6c1fe","event_code":"INTEGRATION-
001","category":"integration","action":"siem.test","outcome":"success","outcome_original":"business","severity":3,"severity_la
bel":"low","message":"siem.test by siem-test","timestamp":"2026-10-19T04:20:35.330Z","observed_timestamp":"2026-10-
19T04:20:35.330Z","export_timestamp":"2026-10-19T04:20:35.330Z","application":"CERTYX
CLM","version":"1.0.0","environment":"production","hostname":"certyx-node-
1","service_name":"clm","component":"siem","organization_id":"platform","organization_kind":"platform","user_id":"siem-
test","correlation_id":"75620e9f-9fe1-4f59-84d4-5c1263f6c1fe","request_id":"75620e9f-9fe1-4f59-84d4-5c1263f6c1fe","changes":
[{"action":"siem.test","requestedBy":"admin"}]}

Format `rfc5424`, profile `ecs`:

<86>1 2026-10-19T04:20:35.699Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM"
swVersion="1.0.0"] <BOM>{"@timestamp":"2026-10-19T04:20:35.699Z","message":"siem.test by siem-test","ecs":
{"version":"8.11.0"},"event":{"id":"ae9bc3c0-bc81-4d1e-9540-11fc3940029b","code":"INTEGRATION-001","kind":"event","category":
["network"],"type":["connection"],"action":"siem.test","outcome":"success","severity":3,"created":"2026-10-
19T04:20:35.699Z","dataset":"certyx.security","provider":"siem"},"user":{"id":"siem-test"},"host":{"hostname":"certyx-node-
1"},"service":{"name":"clm","version":"1.0.0","environment":"production"},"organization":{"id":"platform"},"labels":
{"schema_version":"certyx.event/1.0","certyx_category":"integration"}}

Format `rfc5424`, profile `ocsf`:

```
<86>1 2026-10-19T04:20:36.066Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM" swVersion="1.0.0"] <BOM>{"class_uid":0,"category_uid":0,"activity_id":99,"type_uid":99,"severity_id":1,"status_id":1,"time":1792383636066,"message":"siem.test by siem-test","metadata":{"version":"1.1.0","uid":"f4dbd1be-ac9d-49b1-b952-07226fedb56e","event_code":"INTEGRATION-001","log_name":"security-journal","product":{"name":"CLM","vendor_name":"CERTYX","version":"1.0.0"},"correlation_uid":"f4dbd1be-ac9d-49b1-b952-07226fedb56e"},"unmapped":{"certyx_schema_version":"certyx.event/1.0","certyx_category":"integration","certyx_action":"siem.test"}}
```

Format `cef`:

```
<86>1 2026-10-19T04:20:36.426Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM" swVersion="1.0.0"] CEF:0|CERTYX|CLM|1.0.0|INTEGRATION-001|siem.test by siem-test|3|rt=1792383636426 externalId=1ff31932-0574-411b-ae54-e5198a04aea0 act=siem.test cat=integration outcome=success msg=siem.test by siem-test suser=siem-test suid=siem-test dvchost=certyx-node-1 cs1Label=organizationId cs1=platform cs3Label=correlationId cs3=1ff31932-0574-411b-ae54-e5198a04aea0 cs6Label=schemaVersion cs6=certyx.event/1.0
```

Format `leef-1.0`:

```
<86>1 2026-10-19T04:20:36.792Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM" swVersion="1.0.0"] LEEF:1.0|CERTYX|CLM|1.0.0|INTEGRATION-001|devTime=2026-10-19T04:20:36.792+0000	devTimeFormat=yyyy-MM-dd'T'HH:mm:ss.SSSZ	cat=integration	sev=3	usrName=siem-test	eventId=cc2733fd-b537-4cec-98f2-80c0acf4e3c6	schemaVersion=certyx.event/1.0	action=siem.test	outcome=success	message=siem.test by siem-test	userId=siem-test	organization=platform	organizationKind=platform	correlationId=cc2733fd-b537-4cec-98f2-80c0acf4e3c6	instance=certyx-node-1
```

Format `leef-2.0` with the destination delimiter `^` (without a delimiter TAB is used, declared as `x09`):

```
<86>1 2026-10-19T04:20:37.152Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM" swVersion="1.0.0"] LEEF:2.0|CERTYX|CLM|1.0.0|INTEGRATION-001|^|devTime=2026-10-19T04:20:37.152+0000^devTimeFormat=yyyy-MM-dd'T'HH:mm:ss.SSSZ^cat=integration^sev=3^usrName=siem-test^eventId=4ead979d-6b67-496c-931a-d0c0ff1fc02a^schemaVersion=certyx.event/1.0^action=siem.test^outcome=success^message=siem.test by siem-test^userId=siem-test^organization=platform^organizationKind=platform^correlationId=4ead979d-6b67-496c-931a-d0c0ff1fc02a^instance=certyx-node-1
```

Format `kv` (`certyx-kv/1.0`):

```
<86>1 2026-10-19T04:20:37.523Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"][origin software="CERTYX CLM" swVersion="1.0.0"] schema_version=certyx.event/1.0 event_id=acbe1e97-a048-43f7-b350-ae2dba0f2dfb event_code=INTEGRATION-001 category=integration action=siem.test outcome=success outcome_original=business severity=3 severity_label=low message="siem.test by siem-test" timestamp=2026-10-19T04:20:37.523Z observed_timestamp=2026-10-19T04:20:37.523Z export_timestamp=2026-10-19T04:20:37.523Z application="CERTYX CLM" version=1.0.0 environment=production hostname=certyx-node-1 host_ip=- service_name=clm component=siem organization_id=platform organization_name=- organization_kind=platform user_id=siem-test username=- user_role=- session_id=- authentication_method=- user_verified=- user_agent=- src_ip=- src_port=- http_method=- http_route=- http_status=- protocol=- correlation_id=acbe1e97-a048-43f7-b350-ae2dba0f2dfb request_id=acbe1e97-a048-43f7-b350-ae2dba0f2dfb trace_id=- resource_type=- resource_id=- change_type=- reason=- error_type=- sequence=- event_hash=- agent_fingerprint=- changes="[{\"action\":\"siem.test\",\"requestedBy\":\"admin\"}]" changes_redacted=- dst_ip=- dst_port=- span_id=-
```

Format `cef-compat` (first release):

```
<86>1 2026-10-19T04:20:38.263Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"] CEF:0|CERTYX|CLM|1.0.0|certyx:business|business INTERNAL business/siem.test|3|rt=1792383638263 suser=siem-test requestMethod=INTERNAL request=business/siem.test outcome=business externalId=0 dvchost=certyx-node-1 cn1Label=httpStatus cn1=200 cs1Label=organization cs1=platform certyxOrganizationName= certyxOrganizationKind=platform cs2Label=traceId
```

```
cs2=0d0b9f13-888d-4f45-bca4-ec48d749f919 cs3Label=eventHash cs3= cs4Label=agentFingerprint cs4= cs5Label=errorType cs5=
cs6Label=changes cs6=[{"action":"siem.test","requestedBy":"admin"}] certyxEventId=0d0b9f13-888d-4f45-bca4-ec48d749f919
```

Format `leef-compat` (first release):

```
<86>1 2026-10-19T04:20:38.733Z certyx-node-1 clm 1 integration [timeQuality tzKnown="1"]
LEEF:2.0|CERTYX|CLM|1.0.0|certyx:business|^|devTime=2026-10-19T04:20:38.733+0000^devTimeFormat=yyyy-MM-
dd'T'HH:mm:ss.SSSZ^usrName=siem-
test^cat=business^sev=3^method=INTERNAL^resource=business/siem.test^httpStatus=200^organization=platform^organizationName=^org
anizationKind=platform^traceId=2938c9ab-5183-4e4f-81fa-
deab9119266a^sequence=0^eventHash=^agentFingerprint=^errorType=^instance=certyx-node-1^changes=
[{"action":"siem.test","requestedBy":"admin"}]^eventId=2938c9ab-5183-4e4f-81fa-deab9119266a
```

31.4 Microsoft Sentinel

Format `sentinel` (Logs Ingestion row):

```
[{"TimeGenerated":"2026-10-
19T04:20:39.0967775Z","Sequence":0,"Organization":"platform","OrganizationName":"","OrganizationKind":"platform","Actor":"siem
-
test","SourceIp":"system","AgentFingerprint":"","Method":"INTERNAL","Route":"business/siem.test","Status":200,"Outcome":"busin
ess","TraceId":"70b73efa-1bea-4dfe-9e68-d72ba95b82a7","ErrorType":"","Instance":"certyx-node-1","EventHash":"","Changes":
[{"action":"siem.test","requestedBy":"admin"}],"User":"","UserVerified":false,"UserOrganization":"","AuthAction":"","AuthMetho
d":"","AuthResult":"","Reason":"","UserAgent":"","EventId":"70b73efa-1bea-4dfe-9e68-d72ba95b82a7"}]
```

31.5 OpenTelemetry Logs

Format `otlp-json`, log record body *Message* (test event):

```
{"resourceLogs":[{"resource":{"attributes":[{"key":"service.name","value":{"stringValue":"certyx"}},
{"key":"service.version","value":{"stringValue":"1.0.0"}},{"key":"deployment.environment.name","value":
{"stringValue":"production"}},{"key":"host.name","value":{"stringValue":"certyx-node-1"}},
{"key":"certyx.organization.id","value":{"stringValue":"platform"}},{"key":"certyx.organization.kind","value":
{"stringValue":"platform"}}]},"scopeLogs":[{"scope":{"name":"certyx.security-journal","version":"1.0.0"},"logRecords":
[{"timeUnixNano":"1792383639470357800","observedTimeUnixNano":"1792383639470357800","severityNumber":9,"severityText":"INFO","
body":{"stringValue":"siem.test by siem-test"},"attributes":[{"key":"certyx.schema_version","value":
{"stringValue":"certyx.event/1.0"}},{"key":"certyx.event.id","value":{"stringValue":"eaf0e504-9ff5-4214-8758-5eebb3e5b907"}},
{"key":"certyx.event.code","value":{"stringValue":"INTEGRATION-001"}},{"key":"certyx.event.category","value":
{"stringValue":"integration"}},{"key":"event.name","value":{"stringValue":"siem.test"}},{"key":"certyx.event.outcome","value":
{"stringValue":"success"}},{"key":"certyx.event.outcome_original","value":{"stringValue":"business"}},
{"key":"certyx.event.severity","value":{"intValue":"3"}},{"key":"certyx.event.severity_label","value":{"stringValue":"low"}},
{"key":"certyx.export_timestamp","value":{"stringValue":"2026-10-19T04:20:39.470Z"}},{"key":"certyx.application","value":
{"stringValue":"CERTYX CLM"}},{"key":"certyx.service_name","value":{"stringValue":"clm"}},{"key":"certyx.component","value":
{"stringValue":"siem"}},{"key":"user.id","value":{"stringValue":"siem-test"}},{"key":"certyx.correlation_id","value":
{"stringValue":"eaf0e504-9ff5-4214-8758-5eebb3e5b907"}},{"key":"certyx.request_id","value":{"stringValue":"eaf0e504-9ff5-4214-
8758-5eebb3e5b907"}},{"key":"certyx.changes","value":{"stringValue":"
[{\"action\":\"siem.test\",\"requestedBy\":\"admin\"}]"}}]}]}]}]}
```

A journal request entry (the `POST` of a test, with trace and span ID, client and server address), exported by the export pass over `otlp-json`:

```
{"resourceLogs":[{"resource":{"attributes":[{"key":"service.name","value":{"stringValue":"certyx"}},
{"key":"service.version","value":{"stringValue":"1.0.0"}},{"key":"deployment.environment.name","value":
{"stringValue":"production"}},{"key":"host.name","value":{"stringValue":"certyx-node-1"}},{"key":"host.ip","value":
```

{"arrayValue":{"values":[{"stringValue":"127.0.0.1"}]}}},{"key":"certyx.organization.id","value":{"stringValue":"default"}},
{"key":"certyx.organization.kind","value":{"stringValue":"organization"}}]},"scopeLogs":[{"scope":{"name":"certyx.security-
journal","version":"1.0.0"},"logRecords":
[{"timeUnixNano":"1792383631369104500","observedTimeUnixNano":"1792383631369104500","severityNumber":9,"severityText":"INFO","
body":{"stringValue":"POST /platform/api/siem/{scope}/{id}/test 200 success"},"attributes":
[{"key":"certyx.schema_version","value":{"stringValue":"certyx.event/1.0"}},{"key":"certyx.event.id","value":
{"stringValue":"e828c2f0-1089-48fe-838a-9c5267fb560b:31"}},{"key":"certyx.event.code","value":{"stringValue":"INTEGRATION-
002"}},{"key":"certyx.event.category","value":{"stringValue":"integration"}},{"key":"event.name","value":
{"stringValue":"siem.destination.tested"}},{"key":"certyx.event.outcome","value":{"stringValue":"success"}},
{"key":"certyx.event.outcome_original","value":{"stringValue":"success"}},{"key":"certyx.event.severity","value":
{"intValue":"3"}},{"key":"certyx.event.severity_label","value":{"stringValue":"low"}},
{"key":"certyx.export_timestamp","value":{"stringValue":"2026-10-19T04:20:41.828Z"}},{"key":"certyx.application","value":
{"stringValue":"CERTYX CLM"}},{"key":"certyx.service_name","value":{"stringValue":"clm"}},{"key":"certyx.component","value":
{"stringValue":"platform"}},{"key":"user.id","value":{"stringValue":"7e636f39d1f54da2a7b53c583a2c16ab"}},
{"key":"user.roles","value":{"arrayValue":{"values":[{"stringValue":"Admin"}]}}},{"key":"session.id","value":
{"stringValue":"2387475c1fe53858e83262b02ab8c789"}},{"key":"client.address","value":{"stringValue":"127.0.0.1"}},
{"key":"client.port","value":{"intValue":"38350"}},{"key":"server.address","value":{"stringValue":"127.0.0.1"}},
{"key":"server.port","value":{"intValue":"5291"}},{"key":"http.request.method","value":{"stringValue":"POST"}},
{"key":"http.route","value":{"stringValue":"/platform/api/siem/{scope}/{id}/test"}},
{"key":"http.response.status_code","value":{"intValue":"200"}},{"key":"network.protocol.name","value":{"stringValue":"http"}},
{"key":"certyx.correlation_id","value":{"stringValue":"0HNPDJ0V49OUH:00000002"}},{"key":"certyx.request_id","value":
{"stringValue":"0HNPDJ0V49OUH:00000002"}},{"key":"certyx.sequence","value":{"intValue":"31"}},
{"key":"certyx.event_hash","value":{"stringValue":"3C7261A625BABD9DCFC9918736DBE1825E73B63BCAF264DAF54D419EE4C6FECC"}},
{"key":"certyx.changes","value":{"stringValue":"[{\"action\":\"siem.destination.tested\",\"before\":
{\"scope\":\"platform\",\"id\":\"https-json\"},\"after\":{\"scope\":\"platform\",\"id\":\"https-
json\",\"Ok\":true,\"Error\":null,\"steps\":[{\"Step\":\"dns\",\"Ok\":true,\"Code\":\"dns-literal\"},
{\"Step\":\"tcp\",\"Ok\":true,\"Code\":\"tcp-ok\"},{\"Step\":\"tls\",\"Ok\":true,\"Code\":\"tls-ok\"},
{\"Step\":\"delivery\",\"Ok\":true,\"Code\":\"http-accepted\"}]}},{\"action\":\"request.resource\",\"values\":
{\"scope\":\"platform\",\"id\":\"https-
json\"}}]"}}],"traceId":"5484ed97229f3acd5acff7a0bce7b454","spanId":"ae1a1295fae542f9"}]}]}]}

Format `otlp-protobuf` sends the same request in binary protobuf; the test event above was 1030 bytes.

31.6 File archive

The test event is not written to archives; these are rows of journal events. Format `csv` (header and one row):

"schema_version","event_id","event_code","category","action","outcome","outcome_original","severity","severity_label","message
","timestamp","observed_timestamp","export_timestamp","application","version","environment","hostname","host_ip","service_name
","component","organization_id","organization_name","organization_kind","user_id","username","user_role","session_id","authent
ication_method","user_verified","user_agent","src_ip","src_port","http_method","http_route","http_status","protocol","correlat
ion_id","request_id","trace_id","resource_type","resource_id","change_type","reason","error_type","sequence","event_hash","age
nt_fingerprint","changes","changes_redacted","dst_ip","dst_port","span_id"
"certyx.event/1.0","e828c2f0-1089-48fe-838a-9c5267fb560b:1","CONFIG-028","configuration-
change","sales.mail.seeded","success","business","3","low","sales.mail.seeded mail by system:configuration","2026-10-
19T04:19:41.297Z","2026-10-19T04:19:41.297Z","2026-10-19T04:20:43.904Z","CERTYX CLM","1.0.0","production","certyx-node-
1",,"clm","sales","platform",,"platform","system:configuration",,,,,,,,,,,,"9cf31c6f3be3445ba7e988dc7146cfd4","9cf31c6f3be344
5ba7e988dc7146cfd4",,"sales.mail","mail","create",,,"1","C6E68A3969E5557A0CD49A6969FE6824130603A162AB84DDD0B9D4B1E03FE932",,"
[{""action"":""sales.mail.seeded"",""resource"":""mail"",""before"":{},""after"":
{""Enabled"":false,""Host"":"""",""Port"":587,""From"":"""",""Username"":null,""OperatorCopy"":null,""InquiryRecipient"":null}
}]","true",,,

Format `csv-compat` (header and one row):

"Sequence","At","OrganizationId","OrganizationName","OrganizationKind","Instance","Actor","SourceIp","AgentFingerprint","Metho
d","Route","Status","Outcome","Severity","TraceId","ErrorType","SignInAction","SignInUser","SignInMethod","SignInResult","Sign

```
InReason","Changes","PreviousHash","Hash"
"1","2026-10-19T04:19:41.2973423Z","platform","","platform","certyx-node-
1","system:configuration","system","","INTERNAL","business/sales.mail.seeded","200","business","3","9cf31c6f3be3445ba7e988dc71
46cfd4","","","","","","","[{""action"":""sales.mail.seeded"",""resource"":""mail"",""before"":{},""after"":
{""Enabled"":false,""Host"":"""",""Port"":587,""From"":"""",""Username"":null,""password"":
{""Set"":false,""Reference"":null},""OperatorCopy"":null,""InquiryRecipient"":null}}]","","C6E68A3969E5557A0CD49A6969FE6824130
603A162AB84DDD0B9D4B1E03FE932"
```


Format `ndjson`, profile `native` (one line):


```
{"schema_version":"certyx.event/1.0","event_id":"e828c2f0-1089-48fe-838a-9c5267fb560b:1","event_code":"CONFIG-
028","category":"configuration-
change","action":"sales.mail.seeded","outcome":"success","outcome_original":"business","severity":3,"severity_label":"low","me
ssage":"sales.mail.seeded mail by system:configuration","timestamp":"2026-10-19T04:19:41.297Z","observed_timestamp":"2026-10-
19T04:19:41.297Z","export_timestamp":"2026-10-19T04:20:43.904Z","application":"CERTYX
CLM","version":"1.0.0","environment":"production","hostname":"certyx-node-
1","service_name":"clm","component":"sales","organization_id":"platform","organization_kind":"platform","user_id":"system:conf
iguration","correlation_id":"9cf31c6f3be3445ba7e988dc7146cfd4","request_id":"9cf31c6f3be3445ba7e988dc7146cfd4","resource_type"
:"sales.mail","resource_id":"mail","change_type":"create","sequence":1,"event_hash":"C6E68A3969E5557A0CD49A6969FE6824130603A16
2AB84DDD0B9D4B1E03FE932","changes":[{"action":"sales.mail.seeded","resource":"mail","before":{},"after":
{"Enabled":false,"Host":"","Port":587,"From":"","Username":null,"OperatorCopy":null,"InquiryRecipient":null}}],"changes_redact
ed":true}
```


32 Field reference

The complete reference — every canonical field with its source and the mapping to CEF, LEEF, KV, RFC 5424, ECS, OCSF and Splunk CIM — is the field reference document `docs/SIEM-EXPORT-FIELDS.md` of the CERTYX documentation. This section summarises it.

Unknown versus empty. A field the journal does not know, or that does not apply (for example `http_status` of a business change), is *unknown*. JSON formats (NDJSON, RFC 5424 MSG, ECS, OCSF, HEC) leave unknown fields out; CSV writes unknown as an empty unquoted field and a known empty value as `""`; KV writes unknown as a bare `-` and a known empty value as `""`; CEF and LEEF leave unknown and empty values out. All times are ISO 8601 UTC with milliseconds and `Z` (CEF `rt`, OCSF `time` and the HEC `time` are the same instant as epoch values).

32.1 Canonical event

Schema version `certyx.event/1.0`. The order is the fixed column order of `csv` and `kv`.

Field	Content
`schema_version`	`certyx.event/1.0`
`event_id`	`<installation id>:<journal sequence>`; a UUID for test events
`event_code`, `category`	From the event-code catalogue
`action`	Sign-in event name, first business change, or the outcome pseudo-action (`http.read`, `http.write`, `http.denied`, …)
`outcome`, `outcome_original`	`success`, `failure` or `unknown`; the journal outcome
`severity`, `severity_label`	0–10 (attempt 2, success/business 3, rejected 5, denied 7, error 8; sign-in success 3, failure 7, lockout 8); low, medium, high, very-high
`message`	Short English sentence
`timestamp`, `observed_timestamp`, `export_timestamp`	When the event happened, was observed (equal) and was serialized for export

Field	Content
`application`, `version`, `environment`	`CERTYX CLM`, product version, environment (`Clm:Siem:Environment`, otherwise the hosting environment)
`hostname`, `host_ip`	Recording instance; server address (enrichment)
`service_name`, `component`	`clm`; first route segment after `api` or first segment of a business action
`organization_id`, `organization_name`, `organization_kind`	See Several organizations
`user_id`, `username`, `user_role`, `session_id`	Actor; sign-in user; role and session correlation ID (enrichment)
`authentication_method`, `user_verified`, `user_agent`	Sign-in events
`src_ip`, `src_port`	Source address (only a valid address); source port (enrichment)
`dst_ip`, `dst_port`	Destination: the server address and port that accepted a request (enrichment), or the observed target of a discovery event
`span_id`	W3C span ID of the request activity (enrichment)
`http_method`, `http_route`, `http_status`, `protocol`	Request entries; route template, never the raw path; `http`
`correlation_id`, `request_id`, `trace_id`	Joins attempt and outcome; ASP.NET request ID; W3C trace ID (enrichment)
`resource_type`, `resource_id`, `change_type`	Changed resource; `create`, `update`, `delete`
`reason`, `error_type`	Sign-in reason code or error type; exception type name
`sequence`, `event_hash`, `agent_fingerprint`	Journal sequence (unknown for test events); journal chain hash; agent certificate fingerprint
`changes`, `changes_redacted`	Recorded changes after redaction (JSON array; compact JSON text in CSV and KV); `true` when something was removed

32.2 JSON record

Used by format `json` on the webhook (array of records), syslog (one record per message) and Splunk HEC (`event`).

Field	Type	Content
`Sequence`	number	Position in the CERTYX journal; 0 for the test event
`EventId`	string	Event ID `<installation id>:<journal sequence>` (deduplication key); a UUID for the test event
`At`	string	Event time, ISO 8601 with offset
`Tenant`	string	Organization identifier, `platform` for operator events, `unattributed` for requests without a signed-in organization
`OrganizationName`	string	Name of the organization when the platform knows it; empty for `platform`, `unattributed` and installations without platform administration (added at export)
`OrganizationKind`	string	`organization`, `platform` or `unattributed` (added at export)
`Instance`	string	ID of the CERTYX instance that recorded the event (`Clm:Instance:Id`)
`Actor`	string	Account, company sign-in identity, agent or system actor; `anonymous` before sign-in
`SourceIp`	string	Socket peer address (`system` for background events)
`AgentFingerprint`	string	SHA-256 fingerprint of the agent's client certificate (agent requests)
`Method`	string	HTTP method, `INTERNAL` for background events
`Route`	string	Route template (never the raw path), `business/<action>` for background changes
`Status`	number	HTTP status (200 or 401 for sign-in events)
`Outcome`	string	`attempt`, `success`, `rejected`, `denied`, `error`, `business`, `authentication`
`TraceId`	string	Correlation ID of the request

Field	Type	Content
`ErrorType`	string	Exception type for server errors
`Changes`	array of objects	Each element is a JSON object `{action, before, after}` (nested, without escaped quotes; since release 1.2.12, before that each element was a string holding the object); changes outside a request and sign-in events add `resource`
`Hash`	string	SHA-256 of the entry in the CERTYX journal chain (upper-case hexadecimal)
`SignIn`	object	Sign-in events only: `Action`, `User`, `UserVerified`, `Organization`, `Method`, `Result`, `Reason`, `UserAgent`

32.3 CEF fields (first release)

Format `cef-compat`. The canonical `cef` format is described in ArcSight.

CEF key	Content
Signature ID	`certyx:<outcome>`; sign-in events `certyx:<event name>`
Name	`<outcome> <method> <route>`; sign-in events `<event name> <method>`
`rt`	Event time, epoch milliseconds
`suser`	Actor; sign-in events: the user's e-mail address
`suid`	Sign-in events: the actor
`src`	Source IP (only a valid address)
`requestMethod`, `request`	HTTP method, route template
`outcome`	Outcome; sign-in events `success` or `failure`
`externalId`	Journal sequence
`certyxEventId`	Event ID `<installation id>:<journal sequence>`
`dvchost`	CERTYX instance
`cn1` (`httpStatus`)	HTTP status
`cs1` … `cs6`	`organization`, `traceId`, `eventHash`, `agentFingerprint`, `errorType`, `changes` (JSON)
`certyxOrganizationName`, `certyxOrganizationKind`	Organization name and kind (vendor extension keys; ArcSight shows them as additional data)
`cat`, `act`, `reason`	Sign-in events: `authentication`, event name, reason code
`requestClientApplication`	Sign-in events: user agent
`flexString1` (`authMethod`), `flexString2` (`userVerified`)	Sign-in events: method, verified identity

Header fields escape `\` and the pipe; extension values escape `\`, `=` and line breaks (`\n`, `\r`).

32.4 LEEF fields (first release)

Format `leef-compat` (LEEF 2.0, delimiter `^`). The canonical `leef-1.0` and `leef-2.0` formats are described in QRadar.

LEEF attribute	Content
Event ID	`certyx:<outcome>`; sign-in events `certyx:<event name>`
`devTime`, `devTimeFormat`	Event time (`yyyy-MM-dd'T'HH:mm:ss.SSSZ`)
`usrName`	Actor; sign-in events: the user's e-mail address
`src`	Source IP (only a valid address)

LEEF attribute	Content
`cat`, `sev`	Outcome, severity 1–10
`method`, `resource`, `httpStatus`	HTTP method, route template, status
`organization`, `organizationName`, `organizationKind`, `traceId`, `sequence`, `eventHash`, `agentFingerprint`, `errorType`, `instance`, `changes`	As in the JSON record (`organization` = `Tenant`)
`eventId`	Event ID `<installation id>:<journal sequence>`
`actor`, `action`, `authMethod`, `result`, `reason`, `userAgent`, `userVerified`	Sign-in events only

Values escape `\` and `^`; line breaks and tabs become spaces.

32.5 Sentinel columns

Column	Type
`TimeGenerated`	datetime
`EventId`	string (`<installation id>:<journal sequence>`)
`Sequence`	long
`Organization`, `OrganizationName`, `OrganizationKind`, `Actor`, `SourceIp`, `AgentFingerprint`, `Method`, `Route`, `Outcome`, `TraceId`, `ErrorType`, `Instance`, `EventHash`	string
`Status`	int
`Changes`	dynamic (array of objects)
`User`, `UserOrganization`, `AuthAction`, `AuthMethod`, `AuthResult`, `Reason`, `UserAgent`	string (empty for events other than sign-in events)
`UserVerified`	boolean

A rule created before `EventId`, `OrganizationName` and `OrganizationKind` existed keeps ingesting the other columns; add them (string) to the table and the stream declaration to store them.

33 Event-code catalogue

Each journal action has a stable event code; an exact entry wins over a family, and among families the longest prefix wins. A family (prefix) row covers action names built at run time, such as `renewal.profile.<decision>`. Codes are never renumbered; new actions get new codes.

Category	Code prefix	Meaning
`authentication`	AUTHN	sign-in, sign-out, lockout, MFA steps
`authorization`	AUTHZ	refused requests (401/403) and refused sign-in networks
`user-management`	USER	accounts, SCIM users and groups, devices, MFA resets
`configuration-change`	CONFIG	SSO, networks, notification, SIEM, agent, discovery, renewal and monitoring settings
`data-access`	DATA	exports, archive downloads, revealed document passwords, reads
`administrative`	ADMIN	platform operator actions, customers, sales records, approvals and operational decisions
`security`	SECURITY	agent revocation and identity, presented certificates, certificate and discovery alerts
`application-error`	ERROR	failed and rejected requests, provisioning failures
`system`	SYSTEM	health, scans, jobs, deployments, request attempts and annotations

Category	Code prefix	Meaning
`integration`	INTEGRATION	connection checks and test messages of SIEM, notification, alert and analyst connections, imports, skipped and resent SIEM events

The category is also the RFC 5424 MSGID.

33.1 Chosen by outcome

Entries without a business action get a pseudo-action and the code of their outcome. A successful request uses `http.read` for GET and HEAD, `http.write` otherwise; an uncatalogued business action keeps its name and gets SYSTEM-001.

Code	Category	Action	Kind
AUTHN-000	authentication	`authentication`	sign-in entry without readable details
AUTHZ-000	authorization	`http.denied`	outcome denied
DATA-000	data-access	`http.read`	outcome success, GET/HEAD
ADMIN-000	administrative	`http.write`	outcome success, other methods
ERROR-000	application-error	`http.error`	outcome error
ERROR-001	application-error	`http.rejected`	outcome rejected
SYSTEM-000	system	`http.attempt`	attempt entry
SYSTEM-001	system	`business`	business entry with an uncatalogued action

33.2 Journal actions

Code	Category	Action	Kind
AUTHN-001	authentication	`authentication.sign-in.succeeded`	exact
AUTHN-002	authentication	`authentication.sign-in.failed`	exact
AUTHN-003	authentication	`authentication.sign-out`	exact
AUTHN-004	authentication	`authentication.locked`	exact
AUTHN-005	authentication	`authentication.repeated-failures`	exact
AUTHN-006	authentication	`authentication.mfa.enrolled`	exact
AUTHN-007	authentication	`authentication.mfa.confirmed`	exact
AUTHN-008	authentication	`authentication.mfa.recovery-codes-regenerated`	exact
AUTHN-009	authentication	`authentication.claimed-identity`	exact
AUTHN-010	authentication	`mobile.signin.failed`	exact
AUTHN-011	authentication	`mobile.session.created`	exact
AUTHN-012	authentication	`account.locked`	exact
AUTHN-013	authentication	`account.mfa.recovery-code-used`	exact
AUTHN-014	authentication	`account.passkey.used`	exact
AUTHN-015	authentication	`session.revoked`	exact
AUTHZ-001	authorization	`identity.signin-networks.refused`	exact
USER-001	user-management	`identity.scim.user-created`	exact
USER-002	user-management	`identity.scim.user-replaced`	exact
USER-003	user-management	`identity.scim.user-updated`	exact
USER-004	user-management	`identity.scim.user-deactivated`	exact

Code	Category	Action	Kind
USER-005	user-management	`identity.scim.user-deleted`	exact
USER-006	user-management	`identity.scim.group-created`	exact
USER-007	user-management	`identity.scim.group-replaced`	exact
USER-008	user-management	`identity.scim.group-updated`	exact
USER-009	user-management	`identity.scim.group-deleted`	exact
USER-010	user-management	`identity.scim.role-changed`	exact
USER-011	user-management	`account.created`	exact
USER-012	user-management	`account.updated`	exact
USER-013	user-management	`account.mfa.reset`	exact
USER-014	user-management	`account.mfa.recovery-codes-regenerated`	exact
USER-015	user-management	`authentication.mfa.reset`	exact
USER-016	user-management	`mfa-setup.mail.`	family (prefix)
USER-017	user-management	`customer.account.`	family (prefix)
USER-018	user-management	`mobile.device.`	family (prefix)
USER-019	user-management	`mobile.enrollment.`	family (prefix)
USER-020	user-management	`account.passkey.registered`	exact
USER-021	user-management	`account.passkey.removed`	exact
USER-022	user-management	`account.second-factor.changed`	exact
USER-023	user-management	`identity.scim.role-raised`	exact
USER-024	user-management	`account.create-refused`	exact
USER-025	user-management	`account.email-refused`	exact
CONFIG-001	configuration-change	`identity.sso.`	family (prefix)
CONFIG-002	configuration-change	`identity.signin-networks.changed`	exact
CONFIG-003	configuration-change	`identity.signin-networks.cleared-by-operator`	exact
CONFIG-004	configuration-change	`platform.signin-networks.changed`	exact
CONFIG-005	configuration-change	`customer.signin-networks.cleared`	exact
CONFIG-006	configuration-change	`notification.channel.`	family (prefix)
CONFIG-007	configuration-change	`notification.smtp.`	family (prefix)
CONFIG-008	configuration-change	`siem.destination.`	family (prefix)
CONFIG-009	configuration-change	`agent.configuration.`	family (prefix)
CONFIG-010	configuration-change	`agent.capabilities.updated`	exact
CONFIG-011	configuration-change	`agent.discovery.profile.updated`	exact
CONFIG-012	configuration-change	`discovery.scope.`	family (prefix)
CONFIG-013	configuration-change	`renewal.automatic.`	family (prefix)
CONFIG-014	configuration-change	`auto-renewal.`	family (prefix)
CONFIG-015	configuration-change	`platform.cloud-source.`	family (prefix)
CONFIG-016	configuration-change	`cloud-sources.seeded`	exact
CONFIG-017	configuration-change	`monitoring.alert-channel.`	family (prefix)

Code	Category	Action	Kind
CONFIG-018	configuration-change	`monitoring.alert-relay.`	family (prefix)
CONFIG-019	configuration-change	`monitoring.database-log-source.`	family (prefix)
CONFIG-020	configuration-change	`monitoring.settings.`	family (prefix)
CONFIG-021	configuration-change	`analyst.settings.`	family (prefix)
CONFIG-022	configuration-change	`analyst.organization.`	family (prefix)
CONFIG-023	configuration-change	`remediation.policy.`	family (prefix)
CONFIG-024	configuration-change	`recipient-group.updated`	exact
CONFIG-025	configuration-change	`policy.update.v`	family (prefix)
CONFIG-026	configuration-change	`mail.suppression.`	family (prefix)
CONFIG-027	configuration-change	`mail.events.subscription-confirmed`	exact
CONFIG-028	configuration-change	`sales.mail.`	family (prefix)
CONFIG-029	configuration-change	`execution.private-verification.`	family (prefix)
CONFIG-030	configuration-change	`execution.target.`	family (prefix)
CONFIG-031	configuration-change	`family.`	family (prefix)
CONFIG-032	configuration-change	`certificate.metadata.update`	exact
CONFIG-033	configuration-change	`ssh.metadata.updated`	exact
CONFIG-034	configuration-change	`trust-graph.updated`	exact
CONFIG-035	configuration-change	`endpoint-health.monitor.`	family (prefix)
DATA-001	data-access	`audit.exported`	exact
DATA-002	data-access	`security-events.exported`	exact
DATA-003	data-access	`siem.archive.downloaded`	exact
DATA-004	data-access	`sales.document.password-revealed`	exact
DATA-005	data-access	`analyst.request`	exact
DATA-006	data-access	`analyst.conversation.deleted`	exact
ADMIN-001	administrative	`platform.operator.`	family (prefix)
ADMIN-002	administrative	`platform.customer.changed`	exact
ADMIN-003	administrative	`platform.license.contract-override`	exact
ADMIN-004	administrative	`platform.subscription.contract-override`	exact
ADMIN-005	administrative	`customer.`	family (prefix)
ADMIN-006	administrative	`sales.`	family (prefix)
ADMIN-007	administrative	`certificate.import`	exact
ADMIN-008	administrative	`renewal.approved`	exact
ADMIN-009	administrative	`renewal.cancelled`	exact
ADMIN-010	administrative	`renewal.plan`	exact
ADMIN-011	administrative	`renewal.profile.`	family (prefix)
ADMIN-012	administrative	`alert.acknowledged`	exact
ADMIN-013	administrative	`monitoring.incident.`	family (prefix)
ADMIN-014	administrative	`agent.enrollment.ticket.created`	exact

Code	Category	Action	Kind
ADMIN-015	administrative	`agent.discovery.warning.acknowledged`	exact
ADMIN-016	administrative	`remediation.decision`	exact
ADMIN-017	administrative	`remediation.notice`	exact
ADMIN-018	administrative	`ssh.profile.`	family (prefix)
ADMIN-019	administrative	`ssh.revocation.`	family (prefix)
ADMIN-020	administrative	`ssh.certificate.imported`	exact
ADMIN-021	administrative	`profile.`	family (prefix)
SECURITY-001	security	`agent.revoked`	exact
SECURITY-002	security	`agent.enrolled`	exact
SECURITY-003	security	`agent.identity.`	family (prefix)
SECURITY-004	security	`agent.presented-certificate`	exact
SECURITY-005	security	`certificate.alert`	exact
SECURITY-006	security	`certificate.alert.escalated`	exact
SECURITY-007	security	`discovery.alert.raised`	exact
SECURITY-008	security	`data-protection.secret-purpose-migrated`	exact
ERROR-002	application-error	`customer.provisioning.step-failed`	exact
ERROR-003	application-error	`sales.customer.provisioning-failed`	exact
SYSTEM-002	system	`agent.health.changed`	exact
SYSTEM-003	system	`agent.discovery.baseline.completed`	exact
SYSTEM-004	system	`agent.discovery.networks.reported`	exact
SYSTEM-005	system	`discovery.scan.`	family (prefix)
SYSTEM-006	system	`discovery.started`	exact
SYSTEM-007	system	`discovery.resumed`	exact
SYSTEM-008	system	`endpoint.observe`	exact
SYSTEM-009	system	`ssh.job.`	family (prefix)
SYSTEM-010	system	`ssh.deployment.`	family (prefix)
SYSTEM-011	system	`relay.deployment.`	family (prefix)
SYSTEM-012	system	`lease.expired`	exact
SYSTEM-013	system	`inventory.projected`	exact
SYSTEM-014	system	`execution.result.imported`	exact
SYSTEM-015	system	`request.resource`	exact
SYSTEM-016	system	`unmatched.path`	exact
SYSTEM-017	system	`certificate.expiry`	exact
SYSTEM-018	system	`certificate.renewed`	exact
INTEGRATION-001	integration	`siem.test`	exact
INTEGRATION-002	integration	`siem.destination.tested`	exact
INTEGRATION-003	integration	`siem.destination.checked`	exact
INTEGRATION-004	integration	`notification.channel.tested`	exact

Code	Category	Action	Kind
INTEGRATION-005	integration	`notification.smtp.tested`	exact
INTEGRATION-006	integration	`monitoring.alert-channel.tested`	exact
INTEGRATION-007	integration	`analyst.connection.tested`	exact
INTEGRATION-008	integration	`sales.mail.tested`	exact
INTEGRATION-009	integration	`siem.archive.expired`	exact
INTEGRATION-010	integration	`cloud.snapshot.imported`	exact
INTEGRATION-011	integration	`siem.event.skipped`	exact
INTEGRATION-012	integration	`siem.destination.replayed`	exact
INTEGRATION-013	integration	`siem.event.rejected`	exact