

## LOGZILLA DOCUMENTATION

# NetSapiens

Rules, dashboards, and triggers for the NetSapiens SNAPsolution platform's event subscriptions: calls, CDRs, audit log, call-center agents, presence, subscribers, voicemail and messages posted to the HTTP receiver, and the syslog of its core and supporting servers

LogZilla App Store · Generated October 31, 2026 · [logzilla.ai/docs/logzilla-appstore/netsapiens](https://logzilla.ai/docs/logzilla-appstore/netsapiens)

## Overview

NetSapiens SNAPsolution is a hosted phone system (hosted PBX and unified communications) that service providers run for their business customers; each customer is a domain on the platform. The platform can send its own records to an outside system as they change: completed calls, calls in progress, the audit log of administrator actions, call-center agent and presence states, subscriber changes, voicemails, and text messages. Pointed at LogZilla, those records become searchable events with dashboards and alerts, and nothing is installed on the phone system itself. The platform's core and supporting servers can also send their own syslog to LogZilla, so each server's security configuration check, services, and system lines sit beside the call records.

## App Function

- Receive the platform's event subscriptions on the LogZilla HTTP receiver, one event per record, for all eleven record types the platform offers (`cdr`, `call`, `call_origid`, `auditlog`, `auditlog_lite`, `agent`, `presence`, `subscriber`, `voicemail`, `message`, and `messagesession`)
- Show each completed call with its customer domain, calling and called numbers, subscriber, whether it was answered, how long it lasted, and the reseller; and each call in progress with its core server, both addresses, leg types, state, codec, and the platform action that moved it along, and, when the platform passes the SIP header through, the upstream carrier the call arrived from, the calling device's hardware address, the Microsoft Teams tenant, or an integration's company ID
- Record every administrator action from the audit log as a configuration change or a portal session, with who did it, from where, and to what, and flag failed portal logins as a brute-force attempt
- Follow call-center agents (status, device state, queue), presence states, subscriber role and dial-policy changes, voicemails (caller and length), and chat and SMS messages (direction, status, carrier, sender, recipient, and the text itself)
- Add, through an optional collector, the three things the subscriptions do not carry: why each call ended, with its addresses, core server, codec, and the connection and carrier trunk it went through; the registration state of every phone; and the platform's connections with their call limits and current load
- Parse the syslog the core and supporting servers send, on the NetSapiens core server syslog port or through an HTTPS relay: the security configuration check's result per setting, the state of the database connection pool, the node, certificate, blocklist, address reputation, recording, media converter, application server, file replication, monitoring agent, and VoIP monitor services, with the failures they report, and the platform's scheduled jobs; the servers' sudo, cron, systemd, PAM, and firewall lines are parsed by the Linux apps as any host's
- Mark every event from the platform, records and server lines alike, with the `Platform` tag, so one filter selects everything a core or supporting server logged
- Rewrite the stored message of each record to a short line without timestamps or record ids, so that repeated records deduplicate
- Provide the `NetSapiens: Calls`, `NetSapiens: Audit`, `NetSapiens: Contact Center`, `NetSapiens: Registrations`, and `NetSapiens: Core Servers` dashboards

- Alert on failed portal logins, dial plan, number, and registrar changes, deleted subscribers, blocked calls, agent devices out of service, and a security setting the core servers' check reports as not compliant

## Vendor Documentation

- [NetSapiens API: Event Subscriptions](https://docs.ns-api.com/docs/event-subscriptions-1) (https://docs.ns-api.com/docs/event-subscriptions-1)
- [NetSapiens API: Create an Event Subscription](https://docs.ns-api.com/reference/createsubscription) (https://docs.ns-api.com/reference/createsubscription)
- [NetSapiens API: Limited Access API Keys](https://docs.ns-api.com/docs/limited-access-api-keys) (https://docs.ns-api.com/docs/limited-access-api-keys)
- [NetSapiens API: CDR Field Mappings \(v1 to v2 names\)](https://docs.ns-api.com/docs/cdr-field-mappings) (https://docs.ns-api.com/docs/cdr-field-mappings)
- [NetSapiens API: Read CDRs](https://docs.ns-api.com/reference/get_cdrs) (https://docs.ns-api.com/reference/get\_cdrs)
- [NetSapiens API: Get Users in Domain](https://docs.ns-api.com/reference/getusers) (https://docs.ns-api.com/reference/getusers)
- [NetSapiens API: Get Devices for User](https://docs.ns-api.com/reference/getdevices) (https://docs.ns-api.com/reference/getdevices)
- [LogZilla HTTP Event Receiver](https://www.logzilla.ai/docs/receiving-data/http-event-receiver) (https://www.logzilla.ai/docs/receiving-data/http-event-receiver)
- [LogZilla API: Getting Started](https://www.logzilla.ai/docs/logzilla-api/getting-started) (https://www.logzilla.ai/docs/logzilla-api/getting-started)
- [LogZilla: Using HTTPS](https://www.logzilla.ai/docs/administration/using-https) (https://www.logzilla.ai/docs/administration/using-https)

## Device Configuration

Four pieces make up the integration, and the first two are enough for most of the dashboards:

**Event subscriptions on the platform.** A subscription tells the platform to post every change of one record type (a *model*) to a URL, as a JSON list. One subscription is created per model, through the platform's API, with an API key the platform administrator provides.

**An ingest token and a reachable URL on LogZilla.** The platform posts to LogZilla's HTTP receiver. The URL carries the token, and every core server of the platform must be able to reach it (a subscription moves to another core server on a failure).

**The collector (optional).** A small program shipped with LogZilla that asks the platform's API, every few minutes, for the details the subscriptions do not carry (why a call ended, and which phones are registered) and posts them to the same receiver.

**The core servers' syslog (optional).** Each core or supporting server sends its own syslog to the NetSapiens core server syslog port, or through an HTTPS relay to the same receiver, marked as the platform's, so the NetSapiens: Core Servers dashboard and the security check alert fill.

The subscriptions deliver each change a few seconds after it happens (the platform checks its tables every three seconds).

### Before starting

- Administrator access to the LogZilla server (a shell as root, or `sudo`), or the URL of the LogZilla Cloud instance.
- A LogZilla address that the platform's core servers can reach over HTTPS, such as `https://logzilla.example.com`. The core servers sit on the service provider's network, so this usually means a public name and a firewall rule that allows the cluster's addresses. A plain `http://` URL works on a private network (the

vendor advises against it in production), and the platform allows a custom port in either form. A LogZilla Cloud instance is public and serves HTTPS already.

- A certificate the core servers trust on that address. The platform expects a valid certificate on the URL it posts to, which in practice means one from a public certificate authority (see [Using HTTPS](https://www.logzilla.ai/docs/administration/using-https) (<https://www.logzilla.ai/docs/administration/using-https>)); a self-signed one serves only the collector, with its `--no-verify-ssl` option.
- An API key for the platform, obtained from the platform administrator. It must have Super User scope with no domain limit and no `allowed-models` limit (a subscription for all customers and all eleven models needs that; the vendor page's example key is limited to the messaging models, which is not enough here), and it must not be read-only (the subscription calls create records). It should be limited to the public address the platform sees the LogZilla server as (behind NAT, the gateway's address; the vendor's example key lists two addresses), so that the key is useless elsewhere; the subscription commands below are then run from that server.
- Administrators create keys on the admin portal's API keys page (described in "API Keys in the Admin UI (v44+)", linked from the Limited Access API Keys page listed above; the article needs a NetSapiens customer login) or with the API's create-key call; the key value is shown once, at creation, and copied then. Hosted customers can also ask NetSapiens support to create the key.
- For the optional collector, Docker Compose (the `docker compose` plugin) on the LogZilla host, which has Docker already.
- For the core and supporting servers' syslog, the NetSapiens core server syslog port set in the Application Ports group of the system settings page (the Syslog Netsapiens Port field, for example 5527), reachable from the servers over TCP; or an HTTPS relay on each server, as in the core server syslog section.
- The Linux and Linux iptables apps installed on LogZilla. The NetSapiens app requires them: the install refuses without them, they parse the servers' system and firewall lines, and removing either removes the NetSapiens app with its dashboards and triggers (the platform keeps posting, and the records are stored unparsed until the app is installed again). The Linux PAM app is optional and parses the servers' PAM session lines.

## LogZilla

Create an ingest-only token on the LogZilla server. The last line of the output is the token; it is shown only once, so copy it now:

```
logzilla authtoken create --ingest-only
```

```
Creating INGEST token
Id: 14
ingest-0123456789abcdef0123456789abcdef0123456789abcdef
```

An ingest-only token can post events and nothing else. On LogZilla Cloud, the ingest-only token arrives with the provisioning email (see [Cloud Onboarding](https://www.logzilla.ai/docs/logzilla-cloud/cloud-onboarding) (<https://www.logzilla.ai/docs/logzilla-cloud/cloud-onboarding>)).

Build the URL the platform will post to. It has three parts: the LogZilla address, the path `/incoming/raw/netsapiens/` followed by the model name, and the token as the `AUTHTOKEN` query parameter:

```
https://logzilla.example.com/incoming/raw/netsapiens/cdr?AUTHTOKEN=<ingest token>
```

The path must end in the model name; that is how LogZilla knows which record type arrived. The token travels in the URL, so the URL is as confidential as a password.

Test the URL from any machine before touching the platform. A test record is accepted with 202 on the first line of the answer (on LogZilla Cloud that line may read HTTP/2 202); a wrong token is refused with 401:

```
curl -i -X POST \
  "https://logzilla.example.com/incoming/raw/netsapiens/cdr?AUTHTOKEN=<ingest token>" \
  -H "Content-Type: application/json" -d '{"test": "ok"}'
```

```
HTTP/1.1 202 Accepted
```

The test record appears in LogZilla as a NetSapiens event with the message CDR Unanswered (an unanswered call record with no domain, counted as one completed and one unanswered leg on the Calls dashboard); it is harmless and ages out with the rest.

## NetSapiens

Each model is one subscription, so the first choice is which models to subscribe to. All eleven give the full picture; a model that is not needed is left out, and a subscription can be changed or deleted later (a PUT or a DELETE on `/ns-api/v2/subscriptions/<id>` with the same key; see the API reference). The models, and what each one costs in events:

| Model         | What it carries                                                                                                                            | Volume                      |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------|
| cdr           | One record per completed call leg                                                                                                          | One per leg                 |
| call          | Every call in progress: created, changed, ended                                                                                            | High: several per call      |
| call_origid   | The same calls, keyed on the originating call id                                                                                           | High: several per call      |
| auditlog      | Every administrator action, plus the platform's own housekeeping entries                                                                   | Medium, mostly housekeeping |
| auditlog_lite | Creates and deletes of domains, users, resellers, queues, agents, conferences, and numbers (number changes only when made through the API) | Low                         |
| agent         | Call-center agent state changes                                                                                                            | High on a busy call center  |

| Model          | What it carries                    | Volume |
|----------------|------------------------------------|--------|
| presence       | Presence changes of every user     | High   |
| subscriber     | User record changes                | High   |
| voicemail      | One record per voicemail left      | Low    |
| message        | One record per chat or SMS message | Low    |
| messagesession | One record per conversation update | Low    |

The `call` records also carry the SIP headers that the platform does not map to its own fields; the app reads the upstream carrier's address, the calling device's hardware address, the Microsoft Teams tenant, and an integration's company ID from them and keeps the rest in the record.

Have the API key at hand (see *Before starting*) and the host name of one core server of the cluster, such as `core1.example.com` (any core server serves the API; the platform administrator knows their names).

Create one subscription per model, from the LogZilla server when the key is limited to its address. The loop below creates all eleven; drop any model that is not wanted. Replace the core server, the API key, the LogZilla address, and the token:

```
for model in cdr call call_origid auditlog auditlog_lite agent \
  presence subscriber voicemail message messagesession; do
  curl -X POST "https://core1.example.com/ns-api/v2/subscriptions" \
    -H "Authorization: Bearer <api key>" \
    -H "Content-Type: application/json" -w "\n" \
    -d "{\"model\": \"$model\", \"domain\": \"*\", \"user\": \"*\",
      \"reseller\": \"*\",
      \"post-url\": \"https://logzilla.example.com/incoming/raw/netsapiens/$model?AUTHTOKEN=
<ingest token>\"}"
done
```

In each call, `model` is the record type, `domain`, `user`, and `reseller` set to `*` mean every customer, user, and reseller on the platform (that is what needs the Super User key), and `post-url` is the LogZilla URL from the previous section with that model's name at the end. Each successful call answers with the new subscription's record, which, trimmed to the fields that matter, reads:

```
{ "id": "0123456789abcdef0123456789abcdef", "model": "cdr",
  "status": "pending", "posts-count": 0, "error-count": 0,
  "subscription-expires-datetime": "2046-01-01 12:00:00", ... }
```

A 409 (matching set of parameters already exists) means a subscription with the same settings already exists. A subscription created with an API key lasts twenty years unless an expiry is given (`subscription-expires-`

`datetime`); one created with a user's login token expires with that token.

Check the subscriptions after a few minutes:

```
curl "https://core1.example.com/ns-api/v2/subscriptions" \
-H "Authorization: Bearer <api key>"
```

The answer is a list with one entry per subscription; trimmed to the fields that matter, an entry reads:

```
{ "id": "0123456789abcdef0123456789abcdef", "model": "cdr",
  "post-url": "https://logzilla.example.com/incoming/raw/netsapiens/cdr?AUTHTOKEN=...",
  "status": "active", "posts-count": 42, "error-count": 0, ... }
```

Each subscription reports a `status:pending` until it has had something to post, `active` once a post succeeded, and `error` when a post failed in the last interval. `posts-count` counts every attempt and `error-count` the failures (a core server that cannot reach the URL, a certificate it does not trust, a wrong token, or an error response). An `error` status returns to `active` by itself after an interval of posts without a failure.

## Collector for call records, registrations and connections

The subscription records are the platform's own database rows. A completed call carries no reason for ending, no addresses, no core server, no codec and no trunk, and nothing reports which phones are registered or how loaded the trunks are. The collector fills these gaps by asking the API: every five minutes for the completed calls of the last hour, every two hours for the registration state of every phone, and every fifteen minutes for the platform's connections (its carrier trunks and other SIP peers) with their configured call limits and current load, which also tell which connection each completed call went through. It is a script inside the LogZilla runtime image, run with Docker Compose on the LogZilla host.

The collector's call record (`cdrs`) takes the time the call ended as its event time (a record without one keeps the received time), so a search or dashboard shows the call when it happened, not when the collector posted it: the next poll in normal operation, or later when the collector catches up after a restart or an outage. Every other record, the subscription records and the registration snapshots included, carries the time LogZilla received it.

A call that ended before the online retention window (the Live Data Retention setting, `ARCHIVE_EXPIRE_DAYS`, seven days by default) is refused by storage when it arrives, with a warning in the storage log, so a collector outage longer than that loses the calls older than the window; the rest are stored.

The collector must run from the installed LogZilla image, since it ships inside it and changes with each release. The compose file names the image by its version tag, which `logzilla version` prints and a `.env` file beside the compose file records, and it never downloads an image (`pull_policy: never`): the installed image already carries the script, and a pull of a moving tag such as `latest` names whatever release Docker Hub holds that day, not the installed one.

Create a directory for the collector on the LogZilla host (for example `/opt/netsapiens-collector`) and in it a `compose.yaml` with the following content, replacing the API key, the ingest token, and the two URLs. The `docker compose` commands below run in that directory:

```
services:
  netsapiens-collector:
    container_name: netsapiens_collector
    image: logzilla/runtime:${LOGZILLA_VERSION:?the installed runtime tag}
    pull_policy: never
    restart: unless-stopped
    environment:
      NS_API_KEY: "REPLACE_WITH_THE_NETSAPIENS_API_KEY"
      LOGZILLA_TOKEN: "ingest-REPLACE_WITH_THE_LOGZILLA_INGEST_TOKEN"
    volumes:
      - netsapiens-collector-state:/data
    command: >
      python3 /usr/lib/logzilla/bin/netsapiens-collector.py
      --api-url https://core1.example.com
      --logzilla-url http://logzilla.example.com/incoming
      --cdr-interval 300
      --devices-interval 7200
      --connections-interval 900

volumes:
  netsapiens-collector-state:
```

`--api-url` is one core server of the cluster, scheme and host only. Its host name is added to every row the collector posts, as `_api_host`, and becomes the event host of a row that names no core server (a device that has never registered, every connections row). `--logzilla-url` is the LogZilla address with `/incoming`; the example uses `http://` because a LogZilla host serves plain HTTP until HTTPS is enabled, and the collector runs on that host. The API key is the same one the subscriptions use (the collector reads call records, the connections list, and every domain's users and devices); if the key is limited to addresses, the collector's address must be among them.

Record the installed tag and start the collector, as root in that directory (`sudo -i`, then `cd` into it):

```
echo "LOGZILLA_VERSION=$(logzilla version | tr -d '\r')" > .env
docker compose up -d
```

The first line writes the `.env` file the compose file reads (`tr` drops the carriage return the command prints at a terminal); the second creates the container on that image. Watch the first polls with `docker compose logs -f`. A healthy start looks like this (timestamps and the container prefix omitted); the first call-record poll covers the last hour, and the device crawl reports when it finishes:

```
lz.netsapiens_collector INFO Starting...
lz.netsapiens_collector INFO connections: 120 fetched, 120 posted
```

```
lz.netsapiens_collector INFO cdrs: 412 fetched, 412 posted, window 2026-01-01T11:00:00Z to 2026-01-01T12:00:00Z
lz.netsapiens_collector INFO devices: 120 domains, 2400 users, 0 skipped, 2900 devices posted in 510s
```

A domain or user the key cannot read is logged and skipped on that poll, and the next poll tries again.

The defaults suit most platforms; the table lists what to change and when:

| Option                              | Default                         | Meaning                                                                                                                                                                                                                      |
|-------------------------------------|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>--cdr-interval</code>         | 300                             | Seconds from the start of one call-record poll to the next; 0 turns the poll off                                                                                                                                             |
| <code>--cdr-overlap</code>          | 3600                            | How far back (seconds) each poll re-reads. The platform files a call record when the call ends but lists it by its start time, so a call longer than the overlap is never seen; raise it above the longest call that matters |
| <code>--devices-interval</code>     | 7200                            | Seconds from the start of one device crawl to the next; 0 turns the crawl off (all three intervals at 0 is refused)                                                                                                          |
| <code>--connections-interval</code> | 900                             | Seconds from the start of one connections poll to the next (one request per 1,000 connections, every connection posted on every poll); 0 turns the poll off, which also leaves the call records without their connection     |
| <code>--api-rate</code>             | 5                               | Most requests a second to the platform's API; the device crawl is one request per user, plus one per domain, so 25,000 users take about 83 minutes at 5 a second                                                             |
| <code>--cdr-first-window</code>     | 3600                            | How far back the very first poll looks                                                                                                                                                                                       |
| <code>--cdr-max-window</code>       | 3600                            | After an outage, the gap is read in windows of at most this many seconds, one after the other                                                                                                                                |
| <code>--state-file</code>           | /data/netsapiens-collector.json | Where the poll position is kept, so a restart neither loses nor repeats calls                                                                                                                                                |
| <code>--api-ca-bundle</code>        | system bundle                   | A private CA file, when the platform's certificate is not publicly signed; the platform's certificate is always checked                                                                                                      |

| Option                       | Default | Meaning                                                                                |
|------------------------------|---------|----------------------------------------------------------------------------------------|
| <code>--no-verify-ssl</code> | off     | Skips the certificate check of the LogZilla server only, for a self-signed HTTPS setup |
| <code>--once</code>          | off     | One poll of each kind, then exit (status 1 if any failed), for a test or a cron job    |
| <code>-d, --debug</code>     | off     | Verbose logging, for troubleshooting                                                   |

Raising `--cdr-overlap` means each poll re-reads more records (an hour of 12,000 call legs is 120 requests), so raise `--cdr-interval` with it. The device crawl posts every phone on every crawl, so the registration rows are a snapshot repeated at the interval, not change events (a phone that stays unregistered is reported on every crawl); the crawl posts as it goes, and a crawl that outlasts its interval is followed by the next one at once. The password and login fields of a phone record are dropped before posting. `docker compose stop` lets a poll in flight finish its window, up to five seconds, before the container exits.

After every LogZilla upgrade, run the same two lines again in that directory. The upgrade installs the new release under a new tag, and a container stays on the image it was created from, so until it is recreated the collector runs the previous release; a plain restart does not change that. The poll position lives in the named volume and survives the recreate, and running the two lines when nothing changed is harmless. `docker compose ps` shows the tag the collector runs on, which must equal the output of `logzilla version`. Never pull the image by hand: a pull adds nothing the collector can use, and a moving tag such as `latest` is not the installed release.

## Core server syslog

Each core or supporting server can send its own syslog to LogZilla, so that the platform's security configuration check, its services, and the server's system lines are searchable beside the call records. Two paths carry it; either marks every line as the platform's, which is what the app reads.

**The NetSapiens core server syslog port.** On the LogZilla server, open the Application Ports group of the system settings page and set the **Syslog Netsapiens Port** field (the NetSapiens core server syslog port, for example 5527), then save; the syslog and parser services reload, and both TCP and UDP listeners open on that port. Point each core or supporting server's syslog daemon at it over TCP. With rsyslog, a file in `/etc/rsyslog.d/` holds one line:

```
*.* @logzilla.example.com:5527
```

With syslog-ng, a destination and a log path:

```
destination d_logzilla {
    network ("logzilla.example.com" port (5527) transport ("tcp"));
}
```

```
};
log { source(s_src); destination(d_logzilla); };
```

The server's host name travels in the syslog header; it must be the core server's full name, the name the platform reports as the core server, so that the server's syslog and its call records share one host (rsyslog's `$PreserveFQDN on`, or syslog-ng's `use-fqdn (yes)`). Plain syslog to the default port is parsed as any host's and never gets the Platform tag.

**An HTTPS relay on the core server.** Where the server cannot reach the port, or on LogZilla Cloud, where there is no Application Ports page, the server's syslog daemon posts to `/incoming` in the receiver's structured format with two fields set on every event: `extra_fields._source_type` set to `netsapiens`, the mark the port would have added (the same pattern as [Application Ports on LogZilla Cloud](https://www.logzilla.ai/docs/logzilla-cloud/application-ports-on-cloud) (<https://www.logzilla.ai/docs/logzilla-cloud/application-ports-on-cloud>)), and `host` set to the server's full name. The relay authenticates with an ingest token; a separate ingest-only token per server (`logzilla authtoken create --ingest-only`, as in the LogZilla section) keeps each server's access revocable on its own. On a core server running Ubuntu 22.04, syslog-ng's `http()` destination posts the events in batches and keeps a disk buffer for the time LogZilla is unreachable. With the `syslog-ng-core` and `syslog-ng-mod-http` packages installed, a file in `/etc/syslog-ng/conf.d/` holds the destination, with the server's own name as the host:

```
destination d_logzilla {
    http(
        url("https://logzilla.example.com/incoming")
        method("POST")
        headers(
            "Content-Type: application/json",
            "Authorization: token YOUR_TOKEN"
        )
        body-prefix("{\"events\": [\n")
        delimiter(",\n")
        body('$(format-json
            --pair priority=int($PRI)
            --pair host="core1.example.com"
            --pair program="$PROGRAM"
            --pair message="$MESSAGE"
            --pair extra_fields._source_type="netsapiens"
        )')
        body-suffix("\n]}\n")
        batch-lines(1000)
        batch-timeout(1000)
        disk-buffer(mem-buf-length(10000) disk-buf-size(1073741824)
            reliable(no))
    );
};
log { source(s_src); destination(d_logzilla); };
```

rsyslog's `omhttp` module (from rsyslog's own package repository where the distribution does not ship it) can post the same shape with a disk-assisted queue, one event per request: its batch formats do not produce the `{"events": [...]}` envelope, so the template builds the envelope around a single event and batching stays off.

On a core server running Ubuntu 18.04, where neither module is available from the distribution, LogZilla can provide a proof-of-concept relay script that reads rsyslog's `omprog` output and posts the events. It is not part of the product and is not supported; a supported path is the port or one of the two relays above.

On either path the servers' syslog comes from journald, whose rate limit (`RateLimitBurst=` and `RateLimitIntervalSec=` in `/etc/systemd/journal.conf`) drops a service's lines above the burst in the interval; the platform's cron jobs and services stay well under the defaults, so no change is needed unless a service logs far more.

What the app reads from each program a core or supporting server logs (Vendor is NetSapiens on the platform's own programs):

| Program | Service                          | What the app sets                                                                                                                                        |
|---------|----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| NSSEC   | the security configuration check | Vendor, Event Class Compliance; on a result line the setting, its result, and Event Type Audit                                                           |
| NSCDR   | the CDR service                  | Vendor, Event Class System; the pool state on its statistics line; CDR Export Failed on a failed export                                                  |
| NSNODE  | the node service                 | Vendor, Event Class System; Subscription Delivery Failed on a failed event subscription post, with the endpoint as DstIP when the line names its address |
| NSCERT  | the certificate manager          | Vendor, Event Class System                                                                                                                               |
| APIBAN  | the blocklist updater            | Vendor, Event Class Security; Action and SrcIP on an address added or removed, Updated on the count that closes a run                                    |
| IPAWARE | the address reputation service   | Vendor, Event Class Security; the leading timestamp removed, the level as the severity, SrcIP on a registrar or probed address                           |
| LICF    | the recording service            | Vendor, Event Class System; Init File Error when the service cannot read its init file                                                                   |

| Program                                                 | Service                      | What the app sets                                                                                                                                                                                                                        |
|---------------------------------------------------------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NMC                                                     | the media converter          | as LICE; the encoder's banner and progress lines it relays<br>(ffmpeg version, the indented build, metadata, and stream lines, Guessed Channel Layout, Input #, Output #, Stream mapping:, Press [q], size=, video:) carry Platform only |
| tomcat9                                                 | the application server       | Vendor, Event Class System; Database Unreachable or Name Resolution Failed on the line that names the failure                                                                                                                            |
| INSIGHT                                                 | the monitoring agent         | Vendor, Event Class System; its leading ts= timestamp removed and its level= as the severity; Metrics Collection Failed when an integration cannot be scraped, Metrics Delivery Failed when the metrics cannot be pushed                 |
| NFR                                                     | the file replication service | Vendor, Event Class System; the leading timestamp of its journal lines removed; File Replication Failed on the first line of a failed copy                                                                                               |
| voipmonitor                                             | the VoIP monitor             | Vendor, Event Class System; Capture Device Down when its capture interface is not up, Transcription Failed when a call's transcription fails                                                                                             |
| CRON                                                    | the system's cron            | the platform job as NetSapiens Cron Job; the Linux app parses the line                                                                                                                                                                   |
| sudo, kernel, systemd, sshd, the pam_unix session lines | the system                   | Platform only here; the Linux apps parse them                                                                                                                                                                                            |
| rsyslogd, snmpd, snapd, postfix, systemd-logind         | the system                   | Platform only                                                                                                                                                                                                                            |

The supporting servers send the same way and are handled the same way: the app reads each line's program, never its host, so it does not tell a supporting server from a core; the programs in this table are parsed as the table says on any server, and any other program is stored as any host's, with Platform added.

## Verification

Place or receive a call on the platform, then search LogZilla for events with the `Vendor` field set to `NetSapiens`. The `call` records appear within a few seconds, the `cdr` record when the call ends, and the `NetSapiens : Calls` dashboard fills. With the collector running, the same call's `cdrs` record follows within one poll interval (five minutes by default) with its disconnect reason, listed at the time the call ended, beside the `cdr` record rather than at the top of the list, and the `NetSapiens : Registrations` dashboard fills as the first device crawl runs.

With the core servers sending to the port or through a relay, search for events with `Platform` set to `NetSapiens` and the host set to a core server's name: the server's `sudo` commands, `CRON` jobs, and `systemd` unit starts and stops appear within seconds with `Vendor` set to `Linux`, the security check's lines (program `NSSEC`) appear on its next run with `Vendor` set to `NetSapiens`, and the `NetSapiens : Core Servers` dashboard fills.

## Troubleshooting

| Symptom                                                                        | Likely cause                                                                               | What to do                                                                                                                                                                                                                                                          |
|--------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| No events at all, and the subscriptions report pending                         | Nothing has changed on the platform yet                                                    | Place a call; <code>call</code> records appear within a few seconds                                                                                                                                                                                                 |
| The subscriptions report <code>error</code> and <code>error-count</code> grows | The core servers cannot reach the URL, do not trust its certificate, or the token is wrong | Run the <code>curl</code> test from section from outside the platform; fix the firewall, DNS, certificate, or token until it answers 200                                                                                                                            |
| The <code>curl</code> test answers 401                                         | The token in the URL is wrong or was revoked                                               | Check the token against the one created in the LogZilla : <code>NetSapiens</code> dashboard; if revoked, create a new one with the <code>post-url</code> of events (a <code>PUT</code> on <code>/ns-api/v2/subscriptions</code> ) or delete and re-create the token |
| The <code>curl</code> test answers 422                                         | The body is not a JSON object or list of objects                                           | The platform's posts are not valid JSON; 422 only appears on hosts with the collector running                                                                                                                                                                       |

| Symptom                                                                                                                                                                           | Likely cause                                                                                                                                                        | What to do                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Events arrive, but the <code>host</code> is the LogZilla proxy's address                                                                                                          | A proxy or load balancer between the platform and LogZilla is not counted in <code>AUTH_TRUSTED_PROXY_HOPS</code>                                                   | Set <code>AUTH_TRUSTED_PROXY_HOPS</code> in the Auth and Session to the number of additional hops in front of LogZilla; see <code>ContainerRecords</code> records name the core  |
| <code>docker compose logs</code> repeats <code>can't open file '/usr/lib/logzilla/bin/netsapiens-collector.py' with No such file or directory</code> while the container restarts | <code>.env</code> names an older release without the script                                                                                                         | Run the two lines again than the release that should be installed. NetSapiens app must be installed                                                                              |
| <code>docker compose up</code> fails with required variable <code>LOGZILLA_VERSION</code> is missing a value                                                                      | No <code>.env</code> file in the collector's directory, or <code>logzilla version</code> printed nothing because the lines did not run as root on the LogZilla host | Run the two lines from the collector section, as root                                                                                                                            |
| <code>docker compose up</code> fails with <code>No such image: logzilla/runtime:&lt;tag&gt;</code>                                                                                | <code>.env</code> names a version that is not installed (a typo, or the tag of a release removed since)                                                             | Run the two lines again names the installed tag                                                                                                                                  |
| After an upgrade, <code>docker compose ps</code> shows a tag other than the output of <code>logzilla version</code>                                                               | The two lines were not run after the upgrade; a restart alone keeps the old image                                                                                   | Run the two lines again <code>docker compose up --detach</code> container on the install                                                                                         |
| The collector logs <code>cdns poll failed</code> , <code>devices poll failed</code> or <code>connections poll failed</code>                                                       | The API URL, the key, its address limit, or a privately signed platform certificate                                                                                 | Check that <code>--api-url</code> is correct, the key has Superuser authority and the collector's address is correct for the key; for a private key, pass <code>--api-key</code> |
| The collector logs <code>connections list not loaded</code> and the call records carry no NetSapiens Connection                                                                   | The key cannot read the connections list, or the platform does not serve it                                                                                         | The call records post wait for the next connection and the next check the key's scope, or the platform does not serve it. <code>connections-interval</code> stop asking          |

| Symptom                                                                                                                                                      | Likely cause                                                                                                                                                                                     | What to do                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| The <code>NetSapiens: Registrations</code> dashboard is empty for a long time                                                                                | The first device crawl is still running                                                                                                                                                          | One request per user, per domain, at 5 a second: take about 83 minutes; the crawl goes                                                    |
| A message appears once with a count of two                                                                                                                   | Two identical messages between the same parties inside the deduplication window                                                                                                                  | Expected; see Caveats                                                                                                                     |
| A core server's lines arrive without the <code>Platform</code> tag                                                                                           | The server sends plain syslog to the default port instead of the NetSapiens core server syslog port, or the relay does not set <code>extra_fields._source_type</code> to <code>netsapiens</code> | Point the server at the Application Ports setting relay's <code>body()</code> template feed to the default port host's and never gets the |
| A core server's <code>sudo</code> or firewall lines carry <code>Platform</code> but no <code>Vendor</code>                                                   | The Linux or Linux iptables app is not installed                                                                                                                                                 | Install both apps; the NetSapiens requires them                                                                                           |
| A core server's <code>pam_unix</code> session lines carry <code>Platform</code> only                                                                         | The Linux PAM app is not installed                                                                                                                                                               | Install it if the session lines without it they are stored on any host                                                                    |
| The <code>NetSapiens: Core Servers</code> dashboard lists <code>Subscription Delivery Failed</code> with the LogZilla server's address as <code>DstIP</code> | The core servers cannot post the event subscriptions to LogZilla                                                                                                                                 | Run the <code>curl</code> test from section from a core server: firewall, DNS, certificate; it answers <code>202</code>                   |

## Incoming Log Format

Every post from the platform is one HTTP request to the model's URL with a JSON list of records in the body:

```
POST /incoming/raw/netsapiens/<model>?AUTHTOKEN=<ingest token>
Content-Type: application/json

[{"<field>": "<value>", ...}, {"<field>": "<value>", ...}]
```

The HTTP receiver makes one event per record: `program` is `http_receiver` until the rule runs, `host` is the address the receiver sees, the record's JSON is the message, and every field of the record is kept, as text, in the event's `extra_fields` (a

JSON null becomes None), with `_url_app` set to `netsapiens` and `_url_path` to the model name. The rule then sets the program to `NetSapiens`, picks the fields below into tags, and rewrites the message to a short line. Subscription records carry the platform's v1 field names with underscores; the collector's records carry the v2 names with hyphens. Fields not listed stay in `extra_fields` with the event.

The models and the fields the rule reads:

- **cdr** (one record per completed call leg): `domain`, `orig_from_uri`, `orig_to_user`, `orig_sub`, `term_sub`, `time_answer`, `time_talking`, `territory`, `tag`
- **call** (calls in progress, posted on creation, change, and removal): `hostname`, `orig_ip`, `term_ip`, `orig_type`, `term_type`, `orig_call_info`, `remove`, `by_action`, `media`, `orig_domain`, `term_domain`, `orig_user`, `term_user`, `orig_from_user`, `orig_to_user`, `orig_territory`, `term_territory`, `unknownHeaders` (the SIP headers the platform passes through, one JSON object in a text field, from which the rule reads `P-Source-Device` as an address, `P-Emergency-Info` as an `eui-48-addr=` hardware address, and `X-MS-TenantId` and `X-TRE-CompanyID` as UUIDs, in any letter case; a header of another form sets nothing, and a field that is not a JSON object leaves the record parsed as without it)
- **call\_origid** (the same calls, keyed on the originating call id): `orig_domain`, `term_domain`, `orig_user`, `term_user`, `orig_from_uri`, `orig_to_user`, `orig_call_info`, `remove`
- **auditlog** and **auditlog\_lite**: `action`, `target_object`, `target_user`, `target_domain`, `target_host`, `by_user`, `by_ip`, `by_client`
- **agent**: `entry_status`, `entry_device`, `huntgroup_name`, `huntgroup_domain`, `device_aor`
- **presence**: `presence`, `user`, `domain`
- **subscriber**: `scope`, `dial_policy`, `user`, `domain`
- **voicemail**: `user`, `domain`, `orig_from_uri`, `duration`
- **message**: `domain`, `ses_domain`, `ses_user`, `from_num`, `term_num`, `direction`, `type`, `status`, `carrier`, `media_type`, `media_size`, `rx_hostname`, `text`
- **messagesession**: `domain`, `user`, `remote`, `last_status`, `last_mesg_type`, `muted`, `rx_hostname`, `last_mesg`
- **cdrs** (the collector, GET `/ns-api/v2/cdrs`, hyphenated v2 names): `domain`, `reseller`, `core-server`, `call-direction`, `call-answer-datetime`, `call-disconnect-datetime`, `call-talking-duration-seconds`, `call-disconnect-reason-text`, `call-orig-from-user`, `call-orig-to-user`, `call-orig-user`, `call-term-user`, `call-orig-ip-address`, `call-term-ip-address`, `call-audio-codec`, `call-tag`, `_api_host`, `_connection`, `_carrier_trunk` (added by the collector; the last two from the record's `call-orig-match-uri` and `call-term-match-uri`)
- **devices** (the collector, GET `/ns-api/v2/domains/{domain}/users/{user}/devices`): `device`, `user`, `domain`, `device-sip-registration-state`, `device-sip-registration-ip-address`, `device-sip-registration-user-agent`, `device-sip-registration-datetime`, `core-server`, `_api_host` (added by the collector)

- **connections** (the collector, GET /ns-api/v2/connections): description, domain (\* for a platform-wide connection, as a carrier trunk usually is, which then carries no Domain tag), connection-is-carrier-trunk, limits-max-active-calls-total, limits-max-active-calls-orig, limits-max-active-calls-term, active-calls-orig-current, active-calls-term-current, active-calls-max-this-month, \_api\_host (added by the collector). Credential fields and nested values are dropped from every collector row before posting; the registration object of a connection, which holds the trunk's credentials, is one

The stored message follows one shape per model, so that the same thing happening twice deduplicates: CDR <outcome> <calling> -> <called> domain <domain> <talk bucket>, Call <state> <leg path> <calling> -> <called> on <core> via <action>, Audit <action> <object> <target> by <actor> from <address>, Agent <status> <device> <agent> queue <queue>, Presence <state> <user>@<domain>, Subscriber <user>@<domain> <role>, Voicemail <user>@<domain> from <caller> <length bucket>, CDR <outcome> <direction> <calling> -> <called> domain <domain> on <core> reason <reason> (collector), Device <device> <state> from <address> on <core> <user agent> (collector), Connection <name> carrier trunk domain <domain> active <n> of <limit> peak <p> this month (collector; active <n>, <side> <a> of <limit> when a side limit is the tightest, active <n> alone when no limit is set), Message <type> <direction> <sender> -> <recipient> user <user>@<domain> status <status> via <carrier> media <media type> <size> bytes on <core>: <text>, and Message session <user>@<domain> with <remote> <type> <status> muted on <core>: <last text> (muted only when the session is muted). A part whose field is empty is left out, and call\_origid records carry only Call <state> <calling> -> <called>.

A core or supporting server's line arrives on the NetSapiens core server syslog port as plain syslog, with the program, facility, severity, and message the server logged (syslog-ng adds extra\_fields.\_source\_type set to netsapiens on that port), or from a relay as a structured post that carries the same fields:

```
{"events": [{"priority": 14, "host": "core1.example.com", "program": "NSSEC", "message": "PerDevicePasswords is compliant: yes", "extra_fields": {"_source_type": "netsapiens"}}]}
```

The message is stored as sent, except that the leading timestamps of the address reputation service (program IP-AWARE), the monitoring agent (INSIGHT, its ts= field), and the file replication service's journal lines (NFR) are removed so that repeated lines deduplicate, and the level the first two state becomes the event's severity; the platform's own programs are parsed by the NetSapiens app and the system's by the Linux apps.

## Parsed Metadata Fields

| Tag Name                 | Example              | Description                                                                                                                                                                                                                                                                             |
|--------------------------|----------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Vendor                   | NetSapiens           | NetSapiens on every record and on the platform's own programs, except the media converter's encoder output, which carries Platform only; a server's sudo, CRON, systemd, and kernel lines, and the pam_unix session lines of sudo and CRON, carry the Linux of the app that parsed them |
| Product                  | SNAPsolution         | SNAPsolution wherever Vendor is NetSapiens                                                                                                                                                                                                                                              |
| NetSapiens Model         | cdr                  | Subscription model, or collector path (cdrs, devices), the record came from                                                                                                                                                                                                             |
| Platform                 | NetSapiens           | On every event from the platform: the records and each syslog line a core or supporting server sends, whatever app parsed the line                                                                                                                                                      |
| NetSapiens Check         | PerDevicePasswords   | Setting the security configuration check (NSSEC) reports a result for                                                                                                                                                                                                                   |
| NetSapiens Check Result  | Compliant            | Compliant or Non-Compliant, as the check reports the setting                                                                                                                                                                                                                            |
| NetSapiens Pool State    | Available            | State of the database connection pool from its statistics line: Queuing, Idle, Exhausted, or Available; the counts stay in the message                                                                                                                                                  |
| NetSapiens APIBan Result | Updated              | Updated when a blocklist updater run ends with the count of addresses the firewall blocks; Failed only for an updater line logged at error severity or more severe, which the updater's notice-level output does not reach                                                              |
| NetSapiens Cron Job      | sms/processQueuedSMS | Platform job a core server's cron ran: the API object and action, or a platform script such as apiban/apiban2ufw.sh                                                                                                                                                                     |

| Tag Name                       | Example                         | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NetSapiens<br>Service<br>Error | Subscription<br>Delivery Failed | Failure a platform service logged, once per failure on the line that names it: Subscription Delivery Failed (an event subscription post that did not reach its endpoint; the endpoint's address, when the line names one, in DstIP), Database Unreachable and Name Resolution Failed (the application server), CDR Export Failed, Init File Error (the recording or media converter service), File Replication Failed (a file copy between platform servers), Capture Device Down and Transcription Failed (the VoIP monitor), Metrics Collection Failed and Metrics Delivery Failed (the monitoring agent) |
| Event Class                    | VoIP                            | VoIP for call (cdr, cdrs, call, call_origid), device, agent, presence, subscriber, voicemail, and message records; Config for audit changes; Auth for failed portal logins and logouts; Compliance for the security check, System for the CDR, node, certificate, recording, media converter, application server, file replication, monitoring agent, and VoIP monitor services, Security for the blocklist updater and the address reputation service                                                                                                                                                      |
| Event Type                     | Configuration                   | Configuration for audit create, update, and delete; Session for logout and failed login; Audit for a security check result                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| Domain                         | tenant1.example.com             | NetSapiens domain, the customer tenant; a record at the platform's system domain * carries no Domain tag                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| User                           | 1001                            | Subscriber extension of the record, or the portal login of an audit entry (HC)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Calling<br>Number              | 15550100300                     | Caller, the user part of the From URI, or the sender of a message (HC)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Called<br>Number               | 1002                            | Dialed number, or the recipient of a message (HC)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |

| Tag Name                     | Example           | Description                                                                                                                                                                                                                                  |
|------------------------------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Call Outcome                 | Answered          | Answered or Unanswered (cdr, cdrs)                                                                                                                                                                                                           |
| Call Direction               | Inbound           | Inbound, Outbound, Missed, On-net, or Unknown, from the platform's direction code (cdrs)                                                                                                                                                     |
| NetSapiens Disconnect Reason | Term: Bye         | Why the leg ended (cdrs)                                                                                                                                                                                                                     |
| Duration Category            | Normal (30s-5m)   | Talk time of a completed leg or length of a voicemail                                                                                                                                                                                        |
| NetSapiens Reseller          | reseller1         | Reseller territory of the domain                                                                                                                                                                                                             |
| NetSapiens Call Tag          | blocked           | Tag the platform put on a completed leg                                                                                                                                                                                                      |
| NetSapiens Core              | core1.example.com | Core server of a call, an audit entry, a message or message session, or a registration, also the event host                                                                                                                                  |
| SrcIP                        | 203.0.113.10      | Originating address of a call, the actor's address of an audit entry, the registration address of a device, or on a core server's line an address the blocklist updater blocked or unblocked or the address reputation service reported (HC) |
| DstIP                        | 10.0.0.20         | Terminating address of a call, or on a core server's line the endpoint an event subscription post could not reach (HC)                                                                                                                       |
| NetSapiens Carrier IP        | 203.0.113.40      | Address of the upstream carrier that an active call arrived from, from the P-Source-Device SIP header the platform passes through; SrcIP keeps the originating address, and a call that did not arrive from a carrier has none (call)        |
| SrcMAC                       | 02:00:5e:10:00:01 | Hardware address that the device placing an active call states in its P-Emergency-Info SIP header, lowercase with colons (call)(HC)                                                                                                          |

| Tag Name                    | Example                                  | Description                                                                                                                                                                                                                                                                                                                |
|-----------------------------|------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NetSapiens<br>Teams Tenant  | aaaaaaaa-bbbb-4ccc-<br>8ddd-eeeeeeee0001 | Microsoft tenant ID of a Teams Direct Routing call, from the X-MS-TenantId SIP header, in lower case (call)                                                                                                                                                                                                                |
| NetSapiens<br>Company ID    | cccccccc-dddd-4eee-<br>8fff-aaaaaaaa0001 | Company ID that a third-party integration adds to a call, from the X-TRE-CompanyID SIP header, in lower case (call)                                                                                                                                                                                                        |
| NetSapiens<br>Call State    | active                                   | progressing, active, held, or ended                                                                                                                                                                                                                                                                                        |
| NetSapiens<br>Call Action   | QueueSDispatch                           | Platform action that advanced the call                                                                                                                                                                                                                                                                                     |
| NetSapiens<br>Leg Path      | gateway -> device                        | Originating and terminating leg types                                                                                                                                                                                                                                                                                      |
| NetSapiens<br>Codec         | PCMU                                     | Negotiated audio codec, G.711 names normalized (call, cdrs)                                                                                                                                                                                                                                                                |
| NetSapiens<br>Registration  | registered                               | registered, unregistered for a registration that lapsed, or never registered for a provisioned line that has not registered anywhere (devices)                                                                                                                                                                             |
| NetSapiens<br>User Agent    | Yealink SIP-T48S<br>66.85                | SIP user agent of the last registration (devices)                                                                                                                                                                                                                                                                          |
| NetSapiens<br>Connection    | Carrier East                             | The platform connection a call leg went through, or the one a connections row describes, by its description (cdrs, connections)                                                                                                                                                                                            |
| NetSapiens<br>Carrier Trunk | Carrier East                             | The connection's description when the platform marks it a carrier trunk; a leg with NetSapiens Connection but not this tag used a connection that is not a carrier trunk, and a leg with neither is on-net once the collector has the connections list, which skips a connection without a description (cdrs, connections) |
| NetSapiens<br>Capacity      | 80-95%                                   | A connection's active calls against the tightest of its configured limits: Unlimited, Under 50%, 50-80%, 80-95%, Over 95% (connections)                                                                                                                                                                                    |

| Tag Name                           | Example     | Description                                                                                                                                        |
|------------------------------------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| NetSapiens<br>Message<br>Direction | term        | term for a message to the subscriber, <code>orig</code> for one from the subscriber (message)                                                      |
| NetSapiens<br>Message Type         | sms         | Type of a message, or of a session's last message                                                                                                  |
| NetSapiens<br>Message<br>Status    | sending     | Delivery status the platform recorded (message)                                                                                                    |
| NetSapiens<br>Carrier              | bandwidth   | SMS carrier of a message                                                                                                                           |
| NetSapiens<br>Media Type           | image/jpeg  | MIME type of the media attached to a message                                                                                                       |
| NetSapiens<br>Remote<br>Number     | 15550100300 | Other party of a message session (HC)                                                                                                              |
| NetSapiens<br>Session<br>Status    | read        | Whether the session's last message was read: <code>read</code> or <code>unread</code> ( <code>messagesession</code> )                              |
| NetSapiens<br>Muted                | no          | Whether the subscriber muted the session ( <code>messagesession</code> )                                                                           |
| Action                             | create      | Audit action, or <code>Block</code> or <code>Unblock</code> for an address the blocklist updater added to or removed from the firewall's deny list |
| NetSapiens<br>Target<br>Object     | Subscriber  | Object type the audit entry changed                                                                                                                |
| NetSapiens<br>Client               | omp         | Client that made the audit entry                                                                                                                   |

| Tag Name                 | Example           | Description                                                                                  |
|--------------------------|-------------------|----------------------------------------------------------------------------------------------|
| MitreId                  | T1110             | Brute force, on a failed portal login                                                        |
| MITRE Tactic             | Credential Access | Tactic of the technique                                                                      |
| NetSapiens Agent Status  | available         | Agent availability                                                                           |
| NetSapiens Agent Device  | reg               | Agent device state: reg, pen, oos, fto, dnd, noreg                                           |
| NetSapiens Queue         | 7001              | Call queue extension of the agent entry                                                      |
| NetSapiens Presence      | inuse             | Presence state                                                                               |
| NetSapiens Scope         | Office Manager    | Portal role of the subscriber                                                                |
| NetSapiens Dial Policy   | US and Canada     | Dial policy of the subscriber                                                                |
| Compliance - <framework> | 1                 | Applied to audit changes, portal sessions, and security check results, one tag per framework |

## Log Examples

### Completed call leg (cdr)

```
[{"cdr_id": "0123456789abcdef0123456789abcdef01234567ab", "domain": "tenant1.example.com", "type": 1, "time_start": 1700000000, "time_answer": 1700000010, "time_release": 1700000110, "time_talking": 95, "duration": 110, "orig_from_uri": "sip:1001@tenant1.example.com", "orig_from_name": "Example User", "orig_req_uri": "sip:15550100200@tenant1.example.com", "orig_req_user": "15550100200", "orig_to_user": "15550100200", "orig_sub": "1001", "term_sub": null, "term_to_uri": "sip:15550100200@203.0.113.10", "by_sub": null, "territory": "reseller1", "tag": "", "hide": 0}]
```

Stored as:

```
CDR Answered 1001 -> 15550100200 domain tenant1.example.com Normal (30s-5m)
```

## Active call leg (call)

```
[{"hostname": "core1.example.com", "sessionId": "17000000000000000001", "action": "nop",
"orig_callid": "0123456789abcdef@203.0.113.10", "orig_ip": "203.0.113.10", "orig_type": "gateway",
"orig_call_info": "active", "orig_domain": "", "orig_user": "", "orig_from_user": "15550100300",
"orig_to_user": "1002", "term_ip": "10.0.0.20:5060", "term_type": "device", "term_call_info":
"active", "term_domain": "tenant1.example.com", "term_user": "1002", "term_territory": "reseller1",
"by_action": "QueueSDispatch", "media": "G.711 u-law", "time_start": "2026-01-01 12:00:00",
"time_answer": "2026-01-01 12:00:05", "unknownHeaders": "{\n    \nP-Source-Device\":
\"203.0.113.40\"\n}"}]
```

Stored as:

```
Call active gateway -> device 15550100300 -> 1002 on core1.example.com via QueueSDispatch
```

The P-Source-Device header adds the NetSapiens Carrier IP tag 203.0.113.40 (the address of the Carrier East connection in the collector example below); the stored message does not change.

## Audit entry (auditlog)

```
[{"action": "create", "by_client": "omp", "by_user": "admin1@tenant1.example.com", "by_domain":
"tenant1.example.com", "by_ip": "198.51.100.5", "target_object": "Subscriber", "target_user":
"1003@tenant1.example.com", "target_domain": "tenant1.example.com", "target_host":
"core1.example.com", "territory": "reseller1", "parameter": "", "time_stamp": "2026-01-01 12:30:00"}]
```

Stored as:

```
Audit create Subscriber 1003@tenant1.example.com by admin1@tenant1.example.com from 198.51.100.5
```

## Failed portal login (auditlog)

```
[{"action": "login-bad-pwd", "by_client": "api", "by_user": "1001@tenant1.example.com", "by_domain":
"tenant1.example.com", "by_ip": "203.0.113.50", "target_object": "oauth-password", "target_user":
"1001@tenant1.example.com", "target_domain": "tenant1.example.com", "target_host":
"core1.example.com", "territory": "reseller1", "parameter": "", "time_stamp": "2026-01-01 12:31:00"}]
```

## Agent entry (agent)

```
[{"device_aor": "1001@tenant1.example.com", "huntgroup_name": "7001", "huntgroup_domain": "tenant1.example.com", "entry_status": "available", "entry_device": "reg", "entry_option": "immediate", "entry_order": 1, "entry_priority": 1, "session_count": 0, "call_limit": "1", "auto_ans": "no", "last_update": "2026-01-01 12:40:00"}]
```

## Completed call from the collector (cdrs)

```
[{"call-parent-cdr-id": "0123456789abcdef0123456789abcdef01234567b1", "id": "0123456789abcdef0123456789abcdef01234567b1", "domain": "tenant1.example.com", "reseller": "reseller1", "core-server": "core2.example.com", "call-direction": 0, "call-leg-ordinal-index": 0, "call-start-datetime": "2026-01-01T12:10:00+00:00", "call-ringing-datetime": null, "call-answer-datetime": null, "call-disconnect-datetime": "2026-01-01T12:10:03+00:00", "call-disconnect-reason-text": "Term: 502", "call-total-duration-seconds": 3, "call-talking-duration-seconds": 0, "call-orig-from-user": "1001", "call-orig-to-user": "15550100200", "call-orig-user": "1001", "call-orig-ip-address": "10.0.0.30", "call-term-user": null, "call-term-ip-address": "203.0.113.20", "call-audio-codec": null, "call-tag": "blocked", "call-orig-match-uri": "sip:*@tenant1.example.com", "call-term-match-uri": "sip:*@carrier-east.trunk", "_api_host": "core1.example.com", "_connection": "Carrier East", "_carrier_trunk": "yes"}]
```

Stored as:

```
CDR Unanswered Outbound 1001 -> 15550100200 domain tenant1.example.com on core2.example.com reason Term: 502
```

## Device registration from the collector (devices)

```
[{"device": "1001a", "user": "1001", "domain": "tenant1.example.com", "device-sip-registration-state": "registered", "device-sip-registration-ip-address": "198.51.100.40:5060", "device-sip-registration-user-agent": "Yealink SIP-T48S 66.85", "device-sip-registration-datetime": "2026-01-01T12:00:00+00:00", "device-sip-registration-expires-datetime": "2026-01-01T12:01:00+00:00", "device-sip-latency-seconds-current": 0.021, "core-server": "core1.example.com", "_api_host": "core1.example.com"}]
```

Stored as:

```
Device 1001a registered from 198.51.100.40 on core1.example.com Yealink SIP-T48S 66.85
```

A line that has never registered, as the collector posts it:

```
[{"device": "1003a", "user": "1003", "domain": "tenant1.example.com", "device-sip-registration-state": "unregistered", "device-sip-registration-ip-address": "", "device-sip-registration-user-agent": ""}]
```

```
agent": "", "device-sip-registration-datetime": null, "device-sip-registration-expires-datetime": null, "core-server": "", "_api_host": "core1.example.com"]]
```

Stored as:

```
Device 1003a never registered
```

## Connection from the collector (connections)

```
[{"description": "Carrier East", "domain": "*", "connection-address": "203.0.113.40", "connection-is-carrier-trunk": "yes", "connection-orig-match-pattern": "sip:*@carrier-east.trunk", "limits-max-active-calls-total": 100, "active-calls-orig-current": 60, "active-calls-term-current": 25, "active-calls-max-this-month": 90, "_api_host": "core1.example.com"}]
```

Stored as(a platform-wide connection, domain \*, carries no Domain tag and no domain in the message):

```
Connection Carrier East carrier trunk active 85 of 100 peak 90 this month
```

## Voicemail (voicemail)

```
[{"cdr_id": "0123456789abcdef0123456789abcdef01234567ad", "user": "1002", "domain": "tenant1.example.com", "orig_from_uri": "15550100300@203.0.113.10", "orig_req_user": "1002", "orig_to_user": "1002", "term_sub": "1002", "duration": 42, "time_start": 1700000300, "time_answer": 1700000305, "time_release": 1700000347, "time_talking": 42, "file": "vmail/tenant1.example.com/1002/new/0123456789abcdef.wav", "newVoicemails": 3, "type": 2, "hide": 0, "territory": "reseller1"}]
```

## Text message (message)

```
[{"id": "0123456789abcdef0123456789abc", "session_id": "0123456789abcdef0123456789abcdef", "domain": "tenant1.example.com", "ses_domain": "tenant1.example.com", "ses_user": "1001", "from_uid": "15550100300", "term_uid": "1001@tenant1.example.com", "from_num": "15550100300", "term_num": "15550100200", "dialed": "15550100200", "direction": "term", "type": "sms", "status": "sending", "carrier": "bandwidth", "text": "Running ten minutes late", "media_type": "", "media_size": 0, "rx_hostname": "core1.example.com", "from_ua": "example-client/1.0", "timestamp": "2026-01-01 12:30:00", "second_old": 0, "deleted_timestamp": "", "from_attendee_id": "", "instance_id": "", "participants": null, "session_name": ""}]
```

Stored as:

```
Message sms term 15550100300 -> 15550100200 user 1001@tenant1.example.com status sending via
bandwidth on core1.example.com: Running ten minutes late
```

## Message session (messagesession)

```
[{"session_id": "0123456789abcdef0123456789abcdef", "domain": "tenant1.example.com", "user": "1001",
"remote": "15550100300", "smsani": "15550100200", "last_sender": "15550100300", "last_mesg": "Running
ten minutes late", "last_mesg_type": "sms", "last_status": "read", "last_timestamp": "2026-01-01
12:30:00", "last_update_sec": 1767270600, "last_email": "", "muted": "no", "rx_hostname":
"core1.example.com", "session_type": "", "session_name": "", "participants": null, "queue_status":
null, "queue_session_id": "", "term_queue": null, "start_timestamp": null, "agent_start_timestamp":
null, "last_agent_pickup_sec": null, "last_mesg_media_type": "", "last_mesg_media_size": 0}]
```

Stored as:

```
Message session 1001@tenant1.example.com with 15550100300 sms read on core1.example.com: Running ten
minutes late
```

## Security check result (NSSEC)

```
PerDevicePasswords is compliant: yes
```

Tagged Vendor NetSapiens, Platform NetSapiens, Event Class Compliance, Event Type Audit, NetSapiens Check PerDevicePasswords, and NetSapiens Check Result Compliant.

```
skipTftp is NOT enabled:
```

The same tags with NetSapiens Check skipTftp and NetSapiens Check Result Non-Compliant; the NetSapiens: Security Setting Not Compliant trigger fires.

## Database pool statistics (NSCDR)

```
{"level": "MONITOR", "function": "pool_stats", "message": "pool
statistics", "total_connections": 4, "free_connections": 0, "queued_requests": 3}
```

Tagged Event Class System and NetSapiens Pool State Queuing; the counts stay in the message.

## Platform cron job (CRON)

```
(root) CMD (sleep 10 && curl -sk -X POST 'https://core1.example.com/ns-api/?
object=sms&action=processQueuedSMS' --connect-timeout 10 -m 20 2>&1 &> /dev/null)
```

Tagged PlatformNetSapiens and NetSapiens Cron Job sms/processQueuedSMS by the NetSapiens app, then Vendor Linux, Product System, User root, and Command by the Linux app.

## Administrator command on a core (sudo)

```
user1 : TTY=pts/0 ; PWD=/var/log ; USER=root ; COMMAND=/bin/cat /var/log/syslog
```

Tagged PlatformNetSapiens by the NetSapiens app, then Vendor Linux, Product System, Event Class Auth, Event Type Privilege Escalation, User user1, Runas User root, and Command by the Linux app.

## Address reported by the address reputation service (IPAWARE)

```
[2026-01-02T03:04:05.678Z] WARN bad-registrar: 203.0.113.40 user_agent list is truncated at
group_concat_max_len (1024 bytes, 12 user agent(s) parsed); any user agent past the cut was not
classified
```

Stored as:

```
WARN bad-registrar: 203.0.113.40 user_agent list is truncated at group_concat_max_len (1024 bytes, 12
user agent(s) parsed); any user agent past the cut was not classified
```

Stored without the leading timestamp at severity warning (the WARN level), tagged Vendor NetSapiens, Platform NetSapiens, Event Class Security, and SrcIP 203.0.113.40.

## Blocklist change (APIBAN)

```
Adding 203.0.113.50 to DENY list
```

Tagged Event Class Security, Action Block, and SrcIP 203.0.113.50; the run's Currently UFW blocking 1834 IPs line is tagged NetSapiens APIBan Result Updated.

## Event subscription post failure (NSNODE)

```
Subscription POST Error [5f2a9c0e1b7d] : connect ETIMEDOUT 198.51.100.20
```

Tagged Event Class System, NetSapiens Service Error Subscription Delivery Failed, and DstIP 198.51.100.20.

## Application server database failure (tomcat9)

```
Error conecting to mysql DB.
Exception:com.mysql.jdbc.exceptions.jdbc4.MySQLNonTransientConnectionException: Could not create
connection to database server. Attempted reconnect 3 times. Giving up.
```

Tagged Event Class System and NetSapiens Service Error Database Unreachable; the stack frames that follow, one line each, carry the class only.

## Monitoring agent scrape failure (INSIGHT)

```
ts=2026-01-02T03:04:05.678Z caller=exporter.go:120 level=error component=integrations
integration=mysql instance=support1.example.com:3306 msg="Error from scraper" scraper=slave_status
err="Error 1227: Access denied; you need (at least one of) the SUPER, REPLICATION CLIENT privilege(s)
for this operation"
```

Stored as:

```
caller=exporter.go:120 level=error component=integrations integration=mysql
instance=support1.example.com:3306 msg="Error from scraper" scraper=slave_status err="Error 1227:
Access denied; you need (at least one of) the SUPER, REPLICATION CLIENT privilege(s) for this
operation"
```

Stored without the leading `ts=` field at severity error (the `level=error` field), tagged Vendor NetSapiens, Platform NetSapiens, Event Class System, and NetSapiens Service Error Metrics Collection Failed.

## File replication failure (NFR)

```
#011- failed to copy
#011from 'core2.example.com:/usr/local/NetSapiens/SiPbx/data/tenant1.example.com/1001/vmail/save/vm-
0102-0304.wav'
#011to   '/usr/local/NetSapiens/SiPbx/data/tenant1.example.com/1001/vmail/save/vm-0102-0304.wav'
#011err  (scp -o StrictHostKeyChecking=no -o PasswordAuthentication=no
"core2.example.com:/usr/local/NetSapiens/SiPbx/data/tenant1.example.com/1001/vmail/save/vm-0102-
0304.wav" /usr/local/NetSapiens/SiPbx/data/tenant1.example.com/1001/vmail/save/vm-0102-0304.wav >
/dev/null 2>&1)
```

One event per line (#011 is the tab the syslog daemon escapes), the command's closing ) on a line of its own after these four; the first is tagged Event Class System and NetSapiens Service Error File Replication Failed, every

other line of the block carries the class only. The service's journal lines (`gProcessFileJournalEntry (add)`) are stored without their leading timestamp.

## VoIP monitor capture device down (voipmonitor)

```
packetbuffer - eth1: libpcap error: That device is not up - wait for interface or hot restarting
```

Tagged Event Class System and NetSapiens Service Error Capture Device Down; the monitor's statistics lines (`calls [ . . . ]`) carry the class only.

## Dashboards

- **NetSapiens: Calls** - the call picture: completed and active call counts and rate, unanswered and blocked calls, calls by domain, core server, outcome, talk time, leg path, codec, and platform action, top callers and called numbers, the collector's disconnect reasons, call directions, unanswered calls by core server and by carrier trunk, active calls by the carrier address they arrived from, connection load against the configured limits (one snapshot per connection per poll, so the pie shows how often each load bucket was seen over the time range), and a live stream of call records.
- **NetSapiens: Audit** - who changed what: audit entry counts and rate, failed logins, entries by action, target object, client, actor, address, and core server, failed logins by domain, and a live stream of audit entries.
- **NetSapiens: Contact Center** - agents, presence, and messaging: agent and presence counts and rate, agents by status, device state, and queue, presence by state and domain, a messaging row with messages and unread session updates today and messages by direction, status, carrier, and domain, a live stream of message records, and a live stream of agent and presence records.
- **NetSapiens: Registrations** - the collector's phone registrations: unregistered and polled user counts, the record rate, registration state over time, registrar cores, user agents, unregistered devices by domain and user, registration addresses, and a live stream.
- **NetSapiens: Core Servers** - the core and supporting servers' own syslog: settings the security check reports as not compliant, check results, the database pool's state, blocklist updates, the servers reporting, the rate and programs of their lines, the platform's cron jobs, sudo users on the servers, firewall blocks by port and by source, events by server, the platform services' failures, the endpoints the cores' event subscription posts cannot reach, the addresses the address reputation service reports, and a live stream of the servers' lines.

Failed portal logins and logouts also appear on the **AuthOps** dashboards, and the audit changes, portal sessions, and security check results on the **ComplianceOps** dashboards, together with the other apps. A core server's system lines appear on the **Linux** dashboards as any host's and its firewall blocks on the **Linux iptables** dashboards.

## Triggers

| Trigger                                            | Description                                                                                                                                  |
|----------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| NetSapiens: Portal Login Failure                   | A portal or API login failed ( <code>login-bad-pwd</code> )                                                                                  |
| NetSapiens: Dial Plan, Number, or Registrar Change | An audit entry created, updated, or deleted a dial plan, phone number, or registrar                                                          |
| NetSapiens: Subscriber Deleted                     | A subscriber was deleted                                                                                                                     |
| NetSapiens: Call Blocked                           | A completed leg carries the platform's <code>blocked</code> tag                                                                              |
| NetSapiens: Agent Device Out of Service            | An agent entry reports its device as <code>oos</code> or <code>noreg</code>                                                                  |
| NetSapiens: Security Setting Not Compliant         | The security configuration check on a core server reported a setting as not compliant; fires for each such setting on every run of the check |

## Caveats

- Message text is kept. A `message` record's text and a `messagesession` record's last text stay in the stored event, since the platform already shows them to the operator's staff; how long LogZilla keeps message content is the operator's retention policy, not the app's. The stored line carries no id or timestamp, so two different messages with the same text between the same parties, with the same status, carrier, and core inside the deduplication window, become one event with a count of two.
- A subscription's `cdr` record is the platform's database row, not the richer call record the API serves: it has no reason for ending, no addresses, no core server, and no codec. Those come from the collector's `cdrs` record of the same call, which arrives up to one poll interval later, stamped with the time the call ended. Both records exist for a call seen by both feeds; the completed-call counts on the Calls dashboard use the `cdr` records and the collector widgets the `cdrs` records, so neither feed is counted twice, and the live stream shows both.
- A collector record is stamped with a time up to a poll interval back, so when it is older than the deduplication windows on arrival (two to three minutes at the defaults) the Processing Stats widget counts it as an out-of-time event rather than a new one. That is the normal path for these records; they are stored and shown like any other event.
- The platform lists call records by their start time but files one only when the call ends, so the collector can see a call only while its start is still inside `--cdr-overlap` (one hour by default); a longer call never arrives as a `cdrs` record. The platform also returns the detail fields only for pages of at most 100 records, and the two legs of a call between two of its own users share one record id and differ in domain and direction; the collector's page size and duplicate detection follow both.

- The collector's `cdrs` record carries the call direction as a code. The app names the codes as the API's own `type` filter selects them (0 Outbound, 1 Received, tagged `Inbound`, 2 Missed, 3 On-net); a code the platform does not document (–1 is seen) is tagged `Unknown` with the code kept in the message. The numeric `type` of a subscription's `cdr` record is not documented and is not tagged.
- The `call`, `call_origid`, `presence`, `subscriber`, and `agent` models post on every state change, so they produce many times the events of the `cdr` and `auditlog` models. The `auditlog` model also carries the platform's own `timerange_events` housekeeping entries (create and delete by the `api` client with no user), which make up most of its records. Each subscription is independent, so a model that is not needed is left out.
- The SIP header tags of a `call` record (`NetSapiens Carrier IP`, `SrcMAC`, `NetSapiens Teams Tenant`, and `NetSapiens Company ID`) are set only when the platform passed that header through, which depends on the carrier, the device, or the integration that handled the call. `SrcIP` on a `call` record stays the originating address the record names, and a carrier's address is not matched to its carrier trunk's name, which the collector's `cdrs` records carry. `SrcMAC` is the address the device states in `P-Emergency-Info`; the `P-Access-Network-Info` header names the network node the device is attached to and stays in the record untagged.
- The event host of most subscription records is the address of the core server that posted the record, because those records carry no server name: the HTTP receiver uses the address the LogZilla front proxy accepted the connection from, or, when another proxy or load balancer stands in front of LogZilla, the address behind it according to `AUTH_TRUSTED_PROXY_HOPS` in the Auth and Sessions settings group. The `call`, `auditlog`, `auditlog_lite`, `message`, and `messagesession` records and the collector's records name the core server, which becomes the host. A collector record that names no core server (a device that has never registered, every connections row) is stored under the host name of the API the collector polls (`--api-url`).
- A device the platform reports as unregistered whose row carries no registration time has never registered anywhere: a provisioned line, unused or retired but still configured. It is stored as `Device <device> never registered` with the `NetSapiens Registration` tag `never registered`, so the Registrations dashboard's unregistered counts and lists show lapsed registrations only.
- A core or supporting server's syslog reaches the app only through the NetSapiens core server syslog port or a relay that marks each event as the platform's; the same lines sent as plain syslog to the default port are parsed as any host's, without the `Platform` tag.
- The security check reports every setting on every run, so a setting that stays non-compliant raises the `NetSapiens : Security Setting Not Compliant` trigger on each run until it is fixed; the check's summary and informational lines carry no result tag.
- The blocklist updater's lines (`APIBAN`) are tagged from the forms its runs write: each address added to or removed from the firewall's deny list, and the count that closes a run; any other line is stored with its class only. An updater line logged at error severity or more severe would be a failed run, but the updater's output reaches syslog at notice.
- The address reputation service's lines (`IPAWARE`) are stored without their leading timestamp and at the severity of the level they state (`DEBUG`, `INFO`, `WARN`, or `ERROR`); a line with another level keeps the severity it arrived with.
- The node service's multi-line objects and the application server's stack traces arrive one line per event and are stored with the program's class; only the line that names a failure carries `NetSapiens Service Error`.

- Recording or media converter lines with `Init File Error` every few seconds mean that service cannot read its init file and systemd restarts it; the unit's start and stop lines are parsed by the Linux app, its restart scheduling lines carry `Platform only`.
- The supporting servers' syslog is handled as the cores': the app reads the program, never the host, so a search by host is how to tell a supporting server's lines from a core's.
- The media converter (NMC) relays its encoder's banner and progress for each conversion; those lines carry `Platform only`, while the service's own lines keep `Vendor` and the class.
- The file replication service (NFR) logs a failed copy as several lines; only the first carries `File Replication Failed`.
- The VoIP monitor's statistics lines (`voipmonitor, calls[ . . . ]`) carry their counters and do not deduplicate.
- A core server's `pam_unix` session lines (program `sudo` or `CRON`) are parsed by the Linux PAM app when it is installed and stored with `Platform only` otherwise, as on any host without that app.