HOW CAN WE HELP?

Search our knowledge base articles

SentinelOne Collector for Syslog

Last Updated: Mar 13, 2025

The Syslog Collector lets you ingest syslog events.

The collector is built on top of the SentinelOne Collector, syslog-ng (https://www.syslog-ng.com/), and Docker Compose (https://docs.docker.com/compose/).

Only Linux hosts are currently supported because the host networking driver (https://docs.docker.com/network/drivers/host/) is used.

The SentinelOne Collector uses the addEvents API (https://community.sentinelone.com/s/article/000006773) to upload messages at a rate of approximately 2 MB/sec per log file, with a total throughput up to 12 MB/sec.

Prerequisites:

1. Docker (https://docs.docker.com/get-docker/) and Docker Compose must be installed on the host that runs the collector.

> **Note**
>
> Do not install Docker from your distribution repository because they can be outdated. For more information about installing Docker, see Install Docker Engine (https://docs.docker.com/engine/install/) in the *Docker Documentation*.

2. A Singularity™ Data Lake **Log Access Write** key or an API token with SDL write permissions.

   To generate the key:

   a. At the top left of the Console, click the arrow to open the Scopes panel and select a

scope.

b. In Singularity Data Lake, open the menu next to your user name and select > API Keys.

c. Copy the write key value from the Log Access Keys section, or create a new one.

3. The URL for the Singularity™ Data Lake console, for example *xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)*. See Services and Ports for Management (https://community.sentinelone.com/s/article/000004961) for the Singularity™ Data Lake URL for your Datacenter.

To Install the Syslog Collector:

1. Create a `syslog.yaml` file to configure the collector. See Collector Configuration below for a full description and examples.

   The minimal configuration must set `api-token` with your write key, and `destination` with the Singularity™ Data Lake console URL.

   An example that sets two syslog sources, `cisco-router` and `cisco-firewall`. The `cisco-router` is identified by a syslog-parsed `hostname` that begins with "router" (`router*`), and the `cisco-firewall` is identified by a syslog-parsed `appname` that begins with "firewall" (`firewall*`). Parsers are set for both sources.

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
     parser: ciscoRouter
     matchers:
     - attribute: hostname
       matcher: router*
  - cisco-firewall:
     parser: ciscoFirewall
     matchers:
     - attribute: appname
       matcher: firewall*
```

   The default listening ports are `tcp-601` , `udp-514` , and `tls-6514` .

2. Set the `syslog.crt` and `syslog.key` properties. Self-signed certificates are supported, but we recommend certificates signed by a Certificate Authority (CA), or by a public CA.

   The key and certificate can be generated with OpenSSL (https://www.openssl.org/docs/man1.0.2/man1/openssl-s_client.html). Replace `<host>` with the fully qualified domain name (FQDN), or the non-reserved IP address of the syslog collector host system:

```
openssl req -x509 -nodes -newkey rsa:4096 -keyout syslog.key -out
syslog.crt -subj '/CN=<host>' -days 3650
```

3. Download the latest `docker-compose.yml` file for the collector:

```
curl -o docker-compose.yml https://app.scalyr.com/scalyr-
repo/stable/latest/syslog-collector/docker-compose-latest.yml
(https://app.scalyr.com/scalyr-repo/stable/latest/syslog-
collector/docker-compose-latest.yml)
```

   When successful, the following 4 files `docker-compose.yml` , `syslog.crt` , `syslog.key` , and `syslog.yaml` should now be in the directory.

4. Launch the containers:

Confidential Release Notes

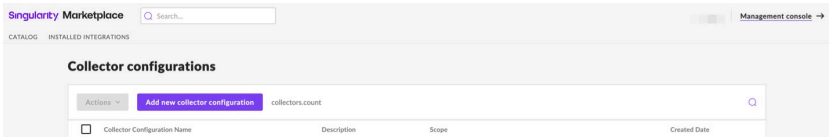
```
docker compose up
```

Make sure you set files by their full path if you are not in the same directory when you `docker compose up`. This is especially applicable for service providers such as Portainer, which has no directory context.

5. Change your firewall allowlist for the ports opened by the collector.

To create a Collector Configuration:

1. Log in to your SentinelOne Console and click Singularity™ Marketplace.
2. Click **Collector Configuration**.
3. Click **Add new collector configuration**.



4. **Enter the fields:**
   - **Name**: This is the name of the data source.
   - **Description**: Description of the data to ingest.
   - Select the **Scope**.
5. Click **Next**.
6. **Enter the fields:**
   - **Data source**
   - **Port**
   - **Parser**: Select from a list of installed parsers.
   - **Parser version**
   - **Protocol**
   - **(Optional)**: Select **Latest Version**.
7. **(Optional)**: Click Add **Another Data Source**.
8. Click **Create Collector Configuration**.
9. To view, click on the name of the added Collector Configuration.
10. You can **Copy** or **Download** the configuration data.

Collector Configuration

The name of the file is expected to be `syslog.yaml`, in the same directory as the `docker-compose.yml` file.

Format:

```
api-token: <elided> # Required.
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net) #
Required.
host: my-host # Recommended.
name: deployment-1 # Optional.
ports: # Optional.
  - udp-514
     type: rfc5424 # Optional.
  - tls-6514
source-types: # Optional.
  - my-cisco-firewalls:
     parser: ciscoRouter # Recommended.
     matchers: # Optional.
     - attribute: hostname
       matcher: router*
  - my-cisco-firewalls:
     parser: ciscoFirewall
     matchers:
     - attribute: appname
       matcher: firewall*
```

Property		Description
`api-token`		Required. The Singularity™ Data Lake **Log Access Write** key.
`destination`		The URL for the Singularity™ Data Lake console, for example *xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)*. See Services and Ports for Management (https://community.sentinelone.com/s/article/000004961) for the Singularity™ Data Lake URL for your Datacenter.
`host`		Recommended. The name of the host the collector is installed on. The name becomes a value for the `serverHost` field in the UI. When not set, defaults to the (dynamic) container name.
`name`		Optional. A name for the collector deployment. Useful when there are multiple deployments. The name becomes a value for the `syslog-collector-name` field in the UI.
`ports`		Optional. A list of listening ports. Values must be in the format `<proto>-<port>`, where proto can be `tcp`, `tls`, or `udp`. When not set, the default ports are `tcp-601`, `udp-514`, and `tls-6514`. TLS implies the use of TCP. Currently syslog-ng does not support D(atagram)TLS for use with UDP.
	`type`	Optional. Sets the syslog format. Values can be `rfc3164` or `rfc5424`. When not set, the default value is `rfc3164`.
`source-types`		A list of source types. Each name becomes a value for the `source-types` field in the UI. Lets you categorize your syslog events, and apply a different parser for each. Useful for querying.
	`parser`	Recommended. Sets the parser name. When not set, defaults to `agentSyslog`.

Property			Description
	`matchers`		Optional. A list of matching rules, to match on syslog events. If multiple rules are set, a match occurs only if *all* matches occur. If a syslog event does not match a rule, the default `agentSyslog` parser is applied. When applied, `mathchers` are sorted from the most-specific to least-specific, because only the first match applies. Specificity is determined by the number of matching rules.
		`attribute`	Optional. Sets the property to match events on. Values can be: • `proto` Matches syslog events by transport protocol (TCP, UPD, or TLS). When set, the `matcher` tag must have a value of `tcp`, `udp`, or `tls`. • `srcip` Matches syslog events by source IP address. • `destport` Matches syslog events by destination port. • `hostname` Matches syslog events by the syslog-parsed host name. • `appname` Matches syslog events by the syslog-parsed application name.
		`matcher`	Optional. Sets the value for a match to occur on `attribute`. Glob patterns (https://en.wikipedia.org/wiki/Glob_(programming)) are supported.

Configuration Examples

1. Single parser

   To set all syslog events to the "cisco-router" `source_type`, and apply the "ciscoNetworkAppliance" `parser`:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
      parser: ciscoNetworkAppliance
```

2. Match events by source IP address

   Syslog events from `192.168.*` addresses are assigned to the "cisco-router" `source_type`, and the "ciscoNetworkAppliance" `parser`. Events from other addresses have the default "agentSyslog" `parser`:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
      parser: ciscoNetworkAppliance
      matchers:
      - attribute: srcip
        matcher: 192.168.*
```

3. Multiple matchers to distinguish source-types

   You can parse access points separately with the `appname` parsed from the events. Both `source-types` are from the same subnet, and only the `appname` distinguishes access points from routers:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
      parser: ciscoNetworkAppliance
      matchers:
      - attribute: srcip
        matcher: 192.168.*
  - cisco-access-point:
      parser: ciscoAccessPoint
      matchers:
      - attribute: srcip
        matcher: 192.168.*
      - attribute: appname
        matcher: ap-*
```

   Glob patterns are not required for `matcher`. You can explicitly set values:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
      parser: ciscoNetworkAppliance
      matchers:
      - attribute: srcip
        matcher: 192.168.1.1
  - cisco-access-point:
      parser: ciscoAccessPoint
      matchers:
      - attribute: srcip
        matcher: 192.168.2.1
```

4. Match events by destination port

   In some cases, it is difficult to distinguish source types from the source. You can send the source output to different ports. Without the `proto` matcher, both TCP and UDP ports match:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
source-types:
  - cisco-router:
      parser: ciscoNetworkAppliance
      matchers:
      - attribute: destport
        matcher: 601
      - attribute: proto
        matcher: tcp
  - cisco-access-point:
      parser: ciscoAccessPoint
      matchers:
      - attribute: destport
        matcher: 514
      - attribute: proto
        matcher: udp
```

5. Set listening ports

   Only use TCP, on port 514, with the rfc5424 syslog format:

```
api-token: <elided>
destination: xdr.us1.sentinelone.net (http://xdr.us1.sentinelone.net)
ports:
  - tcp-514
     type: rfc5424
source-types:
  - cisco-router:
     parser: ciscoRouter
     matchers:
     - attribute: hostname
       matcher: router*
  - cisco-firewall:
     parser: ciscoFirewall
     matchers:
     - attribute: appname
       matcher: firewall*
```

Uninstall the Collector

To stop the containers:

```
docker compose down
```

Troubleshooting

Test Listening Ports

By default, the listening ports are mapped to all interfaces on the host. You can test with Netcat (https://netcat.sourceforge.net/) or Logger (bsdutils) (https://manpages.debian.org/testing/bsdutils/logger.1.en.html) for the clear-text ports, and OpenSSL s_client (https://www.openssl.org/docs/man1.0.2/man1/openssl-s_client.html) for TLS-enabled ports:

To test a clear-text port:

```
echo "<1>$(date '+%b %d %H:%M:%S') localhost test[$((RANDOM % 100))]:
hello world TCP" | nc -v 127.0.0.1 601 # TCP port 601
echo "<1>$(date '+%b %d %H:%M:%S') localhost test[$((RANDOM % 100))]:
hello world UDP" | nc -v 127.0.0.1 601 # UDP port 514
```

To test a clear-text port with Logger:

```
logger -s -n 127.0.0.1 -P 601 -T hello world
```

To test a TLS-enabled port:

```
echo "<1>$(date '+%b %d %H:%M:%S') localhost test[$((RANDOM % 100))]:
hello world" | openssl s_client -connect 127.0.0.1:6514 -verifyCAfile
syslog.crt
```

Make Sure all Containers are Running

Run the `docker ps` command to show only running containers:

```
sudo docker ps
```

Three images should show in the output:

```
scalyr/scalyr-agent-docker-json:<version>
scalyr/syslog-collector-syslog:<version>
scalyr/syslog-collector-config-generator:<version>
```

If these do not show:

1. Make sure the latest YAML file (https://app.scalyr.com/scalyr-repo/stable/latest/syslog-collector/docker-compose-latest.yml) is in use.

2. View the container logs to troubleshoot:

```
sudo docker compose logs
```

Make Sure Ports are Open and the Host is Listening

The ports set must be open, and the host must be listening on them (not inside the container). The default ports are TCP601 and UDP514.

For Linux, run the command:

```
sudo ss -ltupn
```

If the default ports are used, `*:514` and `*:601` shows in the output.

If not, make sure other applications are not trying to use the same ports.

Send a Test Message with Attributes

1. Configure the `syslog.yaml` from the host, or from other hosts that can reach the syslog host, with a hostname attribute. For example:

```
# From syslog.yaml
   matchers:
    - attribute: hostname
      matcher: router*
```

The `agent.json` configuration will reflect the hostname. For example there will be a path to the `router*` log file:

```
/var/log/syslog-collector/syslog-collector-router*-*-eb7c9107a1.log
```

2. Send a message to the IP address and port of the host.

For Linux, with echo and netcat:

```
echo "<1>$(date '+%b %d %H:%M:%S') routerEdge test[$((RANDOM % 100))]:
Test message to ip address" | nc -N -v <ip address of host> <port>
```

For Linux, with logger:

```
logger -s -n <ip address of host> -P <port> -T Test message from logger
```

To test a TLS-enabled port on Linux:

```
echo "<1>$(date '+%b %d %H:%M:%S') routerEdge test[$((RANDOM % 100))]:
Test message to TLS port" | openssl s_client -connect <ip address of
host>:6514 -verifyCAfile syslog.crt
```

The connection and message must succeed.

3. If the connection and message do not succeed, make sure the firewall does not block connections, and enable those that are blocked.

For generic Linux, iptables (https://linux.die.net/man/8/iptables) can be used to configure the Linux kernel firewall (NetFilter).

For RHEL distributions, firewalld (https://firewalld.org/documentation/) can be used:

```
sudo firewall-cmd --zone=public --add-port=514/tcp --permanent
sudo firewall-cmd --zone=public --add-port=6514/tcp --permanent
sudo firewall-cmd --zone=public --add-port=601/tcp --permanent
```

For Ubuntu distributions, UFW (https://help.ubuntu.com/community/UFW) can be used:

```
sudo ufw allow 514/tcp
sudo ufw allow 6514/tcp
sudo ufw allow 601/tcp
```

4. Connect to the SentinelOne Collector container:

```
sudo docker exec -it s1collector_scalyr-agent_1 /bin/bash
```

Open the `/etc/scalyr-agent-2/agent.json` file:

```
cat /etc/scalyr-agent-2/agent.json
```

In the `logs` section, make sure there is a `path` to the hostname log file set in Step 1. For example, a path to the `router*` log file:

```
{
  "implicit_agent_process_metrics_monitor": false,
  "implicit_metric_monitor": false,
  "api_key": "YOUR_KEY_HERE",
  "scalyr_server": "xdr.us1.sentinelone.net
(http://xdr.us1.sentinelone.net)",
  "server_attributes": {
    "syslog-collector-version": "2.0.4"
  },
  "logs": [
    {
      "path": "/var/log/syslog-collector/syslog-collector-router*-*-
eb7c9107a1.log",
      "attributes": {
        "parser": "ciscoRouter",
        "source_type": "cisco-router"
      }
    },
    {
      "path": "/var/log/syslog-collector/syslog-collector-*-firewall*-
eb7c9107a1.log",
      "attributes": {
        "parser": "ciscoFirewall",
        "source_type": "cisco-firewall"
      }
    }
  ]
```

List all log files below the syslog-collector path:

```
ls -la /var/log/syslog-collector/
```

Make sure there is a file with a path to the test message sent in Step 2. For example, the "routerEdge test" messages should have a path similar to `/var/log/syslog-collector/syslog-collector-routerEdge-test-eb7c9107a1.log` . Note that `eb7c9107a1` is random, and will be different.

Open the path to the log file with the test message sent in Step 2. For example:

```
cat  /var/log/syslog-collector/syslog-collector-routerEdge-test-
eb7c9107a1.log
```

The test message must show in the file.

4. If the test message does not show, then it did not reach syslog. Inspect the network connection between the host and the syslog host or container.

Make sure that the firewall does not block connections.

Run the command in the container to get the status of the SentinelOne Collector.

```
scalyr-agent-2 -v status
```

Make sure syslog messages are picked up by the `scalyr-agent` and sent to the Singularity™ Data Lake. For example, in the output:

```
Path /var/log/scalyr-agent-2/agent.log: copied 24876 bytes (110 lines), 0
bytes pending, last checked Wed Jan  3 08:26:32 2024 UTC
Glob: /var/log/syslog-collector/syslog-collector-router*-*-
eb7c9107a1.log:: last scanned for glob matches at Wed Jan  3 08:25:40 2024
UTC
  /var/log/syslog-collector/syslog-collector-routerEdge-test-
eb7c9107a1.log: copied 158 bytes (3 lines), 0 bytes pending, last checked
Wed Jan  3 08:26:32 2024 UTC
Glob: /var/log/syslog-collector/syslog-collector-*-firewall*-
eb7c9107a1.log:: last scanned for glob matches at Wed Jan  3 08:25:40 2024
UTC
Glob: /var/log/scalyr-agent-2/agent-worker-session-*.log:: last scanned
for glob matches at Wed Jan  3 08:25:40 2024 UTC
```

If a line for the test message is not present:

```
/var/log/syslog-collector/syslog-collector-routerEdge-test-eb7c9107a1.log:
copied 158 bytes (3 lines), 0 bytes pending, last checked Wed Jan  3
08:26:32 2024 UTC
```

But there is a "Glob" line to pick up messages for the log file with the test message:

```
Glob: /var/log/syslog-collector/syslog-collector-router*-*-
eb7c9107a1.log:: last scanned for glob matches at Wed Jan  3 08:25:40 2024
UT
```

Then the SentinelOne Collector did not pick up the log file. Make sure the `matches` rules are correct, and that the message has attributes that match the log filename rules.

For other errors related to the `scalyr-agent` , see the docker compose logs:

```
sudo docker compose logs | grep scalyr_agent
```

Or, in the container:

```
sudo docker exec -it s1collector_scalyr-agent_1 /bin/bash
cat /var/log/scalyr-agent-2/agent.log
```


Related Articles

SentinelOne Collector Requirements
(/s/article/000006806)

SentinelOne Collector Syslog Rotation
(/s/article/000009004)

The SentinelOne Collector

(/s/article/000006807)

SentinelOne Collector Installation Quick Start
(/s/article/000006805)

Install the SentinelOne Collector on Windows
(/s/article/000006804)