LOGZILLA DOCUMENTATION

Downstream Syslog Receivers

Configure LogZilla to forward deduplicated events to downstream syslog, file, Splunk HEC, or SNMP destinations using per-forwarder YAML files in forwarder.d

Forwarding To Downstream Receivers · Generated June 11, 2026 · logzilla.ai/docs/forwarding-module/downstream-syslog-receivers

Syslog Module

The Forwarder module allows forwarding all or specific matched events to a downstream log receiver. Deduplication occurs at ingest to reduce repeated events before forwarding. The downstream receiver is not limited to *syslog*; additional destination types are supported in addition to *syslog*: *file*, *splunk-hec*, and *snmp*.

Enable The Module

To enable the Forwarder Module, run the following command on the LogZilla server. See also System Commands (https://www.logzilla.ai/docs/command-line-tools/system-commands).

```
sudo logzilla settings update FORWARDER_ENABLED=true
```

1. Configure rule(s)

LogZilla creates a main forwarder configuration file automatically when the forwarder starts. While it can contain global options and forwarders, the recommended practice is to keep the global file minimal and place individual forwarders as separate files under `/etc/logzilla/forwarder.d/`. Use one forwarder per file for simpler administration.

Examples of the Main Forwarder Configuration File

Here are some sample configurations for the main forwarder configuration file:

Forward to host A

```
---
window_size: 60
forwarders:
- type: syslog
  target: 192.168.0.114:514
  transport: tcp
  unsent_buffer_limit: 250000
  protocol: rfc5424
  rules:
  - match:
      field: counter
      op: gt
      value: 1
    rewrite:
```

```
      message: $MESSAGE ORIGIN="$HOST" LZ_Dedup_Count="$COUNTER"
  - match:
      field: counter
      op: le
      value: 1
    rewrite:
      message: $MESSAGE ORIGIN="$HOST"
fast_forward_first: true
```

Forward to host B

```
---
window_size: 60
forwarders:
- type: syslog
  target: 192.168.0.117:514
  transport: udp
  protocol: bsd
  rules:
  - match:
      field: counter
      op: gt
      value: 1
    rewrite:
      message: $MESSAGE ORIGIN="$HOST" LZ_Dedup_Count="$COUNTER"
  - match:
      field: counter
      op: le
      value: 1
    rewrite:
      message: $MESSAGE ORIGIN="$HOST"
fast_forward_first: true
```

Forward to file

```
---
window_size: 1
fast_forward_first: true
forwarders:
- match:
    field: cisco_mnemonic
    value: BGP-5-ADJCHANGE
  type: file
  target: "/var/log/logzilla/sec/simple.log"
  format: tsv
  separator: "\t"
  fields:
  - last_occurrence
```

```
  - host
  - message
```

Examples of Individual Forwarder Configuration Files

Individual forwarder configuration files can be used. The files can consist of JSON (`.json`) or YAML (`.yaml`). Each file defines a particular forwarder configuration, one forwarder per file, using the same syntax and options as specified in the `forwarders` configuration element as explained above.

For example, `non-dedup.yaml` might contain the following:

```
window_size: 0
type: file
path: /var/log/logzilla/non-dedup.log
```

IMPORTANT: Source host marker

Downstream receivers such as Splunk need to know which host the event originated from. Append a key/value pair in the rewrite rule to carry the original host. The recommended key name is `ORIGIN` (examples above). See Forwarding to Splunk (https://www.logzilla.ai/docs/forwarding-module/forwarding-to-splunk) for example transforms.

Forwarder Main Configuration

format is also supported, in which case end it in `.json`):

`match` This is a filter that defines which events should be forwarded

- its syntax is exactly the same like the one used in rewrite rules Match Conditions in Rewrite Rules (https://www.logzilla.ai/docs/data-transforms/rewrite-rules). This is a global filter, affecting all forwarders; it can also be set in individual forwarders.

`window_size` This is the default value of window_size, if not given for a particular forwarder. It is the time in seconds of keeping every message while looking for its duplicates. The higher number set here, the better deduplication will work, but also the longer delay will be introduced (every message is kept for as much seconds before being forwarded to defined target). Setting this to 0 disables deduplication completely.

`fast_forward_first` The default value of fast_forward_first, if not given for particular forwarder. This defines the behavior for the first unique occurrence in the window. When true (default), the first occurrence will be forwarded without delay, while all following duplicates will be collected and forwarded at the end of the window. When false, first occurrence will be deduped alongside all the duplicates that follow. This setting delays forwarding of every event by `window_size` seconds to allow prospective deduplication.

`forwarders` This section defines forwarders. Multiple forwarders and mixing Syslog and SNMP trap destinations may be used. Every element of the `forwarders` table has a mandatory field `type` which defines what type of forwarder it is - currently `snmp`, `syslog`, `file`, and `splunk-hec` are supported. Other fields depend on the forwarder type.

For example, the following would forward to both an SNMP Trap receiver and a Syslog receiver:

```
---
forwarders:
- oid_map:
  - oid: ".1.2.0"
    src: facility
    type: s
  - oid: ".1.3.0"
    src: severity
    type: i
  - oid: ".1.4.0"
    src: cisco_mnemonic
    type: s
  - oid: ".1.5.0"
    src: message
    type: s
  - oid: ".1.99.0"
    src: counter
    type: i
  oid_prefix: 1.3.6.1.4.1.9.9.41.1.2.3
  target: snmp-server:162
  trap_oid: 1.3.6.1.4.1.2021.991
  type: snmp
- protocol: bsd
  rules:
  - match:
      field: counter
      op: gt
      value: 1
    rewrite:
      message: "$MESSAGE LZ_dedupCount=$COUNTER"
  target: central-log-collector:514
  transport: tcp
  type: syslog
```

Configuration for Each Forwarder

Every element of forwarders array has two mandatory fields: `type` and `target`. Each type might have more mandatory fields.

Supported types: splunk-hec, snmp, syslog, file

Common Options

The following options can be used in every forwarder type:

`match` As explained above, the filter. Only events matching this filter will be forwarded

`window_size` As explained above, this is the time in seconds of keeping every message while looking for its duplicates. The higher number set here, the better deduplication will work, but also the longer delay will be introduced (every message is kept for as much seconds before being forwarded to defined target). Setting this to 0 disables deduplication completely.

`fast_forward_first` As explained above, this defines the behavior for the first unique occurrence in the window. When true (default), the first occurrence will be forwarded without delay, while all following duplicates will be collected and forwarded at the end of the window. When false, first occurrence will be deduped alongside all the duplicates that follow.

`rules` This allows you to use rules in the same way as for rewrite rules. You can specify any list of rules, that will be applied to the event in order before it is forwarded - and which can update its fields. See Rewrite Rules (https://www.logzilla.ai/docs/data-transforms/rewrite-rules) for details on rewrite rules.

Forwarder Options for Particular Forwarder Types

Syslog

`target` This is host and port of the target syslog server.

`transport` Either `tcp` or `udp`. The `tcp` transport can operate in either blocking or non-blocking mode depending on the configuration.

`unsent_buffer_limit` The maximum number of events (post predup) that will be buffered in case the destination is down. If the destination comes back up before the buffer overflows, events will be forwarded in the original order. If the destination remains down and the buffer exceeds the limit, additional events will not be buffered. Defaults to 25000. Applies only to `tcp` transport. Note that buffering is enabled **after** the forwarder realizes that the destination is down, which might be significantly later depending on network communication.

`protocol` Either `bsd` for the classic (RFC3164) protocol or the newer `rfc5424` protocol

`octet_count` Use the octet counting framing method for sending messages.

File

This forwarder saves all forwarded events in a file, in json or TSV format, one line per event.

`target` The path to the file where events are to be saved. This is a path in the container `lz_forwarder_module`, so this file can be accessed with `logzilla shell` into the forwarder container. If desired, this file can be saved directly on the host file system if the file is put in a path inside the `/var/log/logzilla/` directory, because that directory (and subdirectories) is shared between the host and the LogZilla docker container.

`format` Defaults to `json`, in which case it always save whole event. Another option is `tsv` which uses tab separated values, but other separators can be specified (defaults to TAB); with TSV format a list of fields that are written to output file can be provided.

`separator` For the TSV format this is the string used to separate fields (defaults to TAB).

`fields` The array of fields to be written in TSV format (defaults to `["host", "program", "message"]`).

`rotate_period` The time in seconds after which log file will be renamed with `.0` appended (so if it's `fwd.log` it will become `fwd.log.0`), and then the original path will be reopened as an empty file. As appropriate, each previous `.0` file will overwritten so there is always no more than just two log files - the previous and the current one. The default value of 0 disables rotation completely.

splunk-hec
This forwarder sends events in JSON format to Splunk HTTP Event Collector. The receiving splunk instance should be configured to: have a Splunk HEC source enabled; have a HEC token; globally enable the HTTP Event Collector. Both HTTP and HTTPS transports are supported (HTTPS is recommended for production deployments). Documentation for these Splunk settings is available in the Splunk HTTP Event Collector documentation (https://docs.splunk.com/Documentation/Splunk/latest/Data/UsetheHTTPEventCollector). (This has been successfuly tested on Splunk 8.0.1).

`target` The address in the format `HOST`, `HOST:PORT`, `http://HOST:PORT`, or `https://HOST:PORT`, where `HOST` and `PORT` are replaced with the actual values of the host name and TCP/UDP port. If `PORT` is omitted, the default Splunk value of `8088` is used. For clarity, example targets include `https://splunk.example.com:8088`.

`token` The HEC token as specified in Splunk.

SNMP
This forwarder sends an SNMP Trap for each matching event. A list of variables that will be added to the trap can be defined, with values copied from the particular fields of event.

`target` This is the host and port of the SNMP server.

`trap_oid` This will be set as the type of outgoing SNMP trap.

`oid_prefix` Whenever `oid` in the map starts with a dot, it will be prefixed with this prefix.

`oid_map` This is the list of variables that will be added to the trap. For every variable you define:

`type` For now only `i` (32 bit integer) and `s` (string) are supported.

`oid` The object id of this variable; if it starts with a dot then it is prefixed with `oid_prefix`.

`src` The name of the event field variable in which the value will be set.

`value` if no `src` is defined, a constant can be configured here that will be copied for this value.

2. Add the Forwarder Configuration(s)

Recommended: place individual forwarder configuration files in `/etc/logzilla/forwarder.d/` (one forwarder per file). After changes, verify and reload:

```
logzilla forwarder print
logzilla forwarder reload
```

Advanced users may choose to manage the global file at `/etc/logzilla/forwarder.yaml` or `/etc/logzilla/forwarder.json`. In most cases, prefer the `forwarder.d` layout. See Data Commands (https://www.logzilla.ai/docs/command-line-tools/data-commands) for CLI details.

3. Apply forwarder configuration

Reload the forwarder after creating or changing configuration files:

```
sudo logzilla forwarder reload
```