LOGZILLA DOCUMENTATION

Forwarder and Deduplication Overview

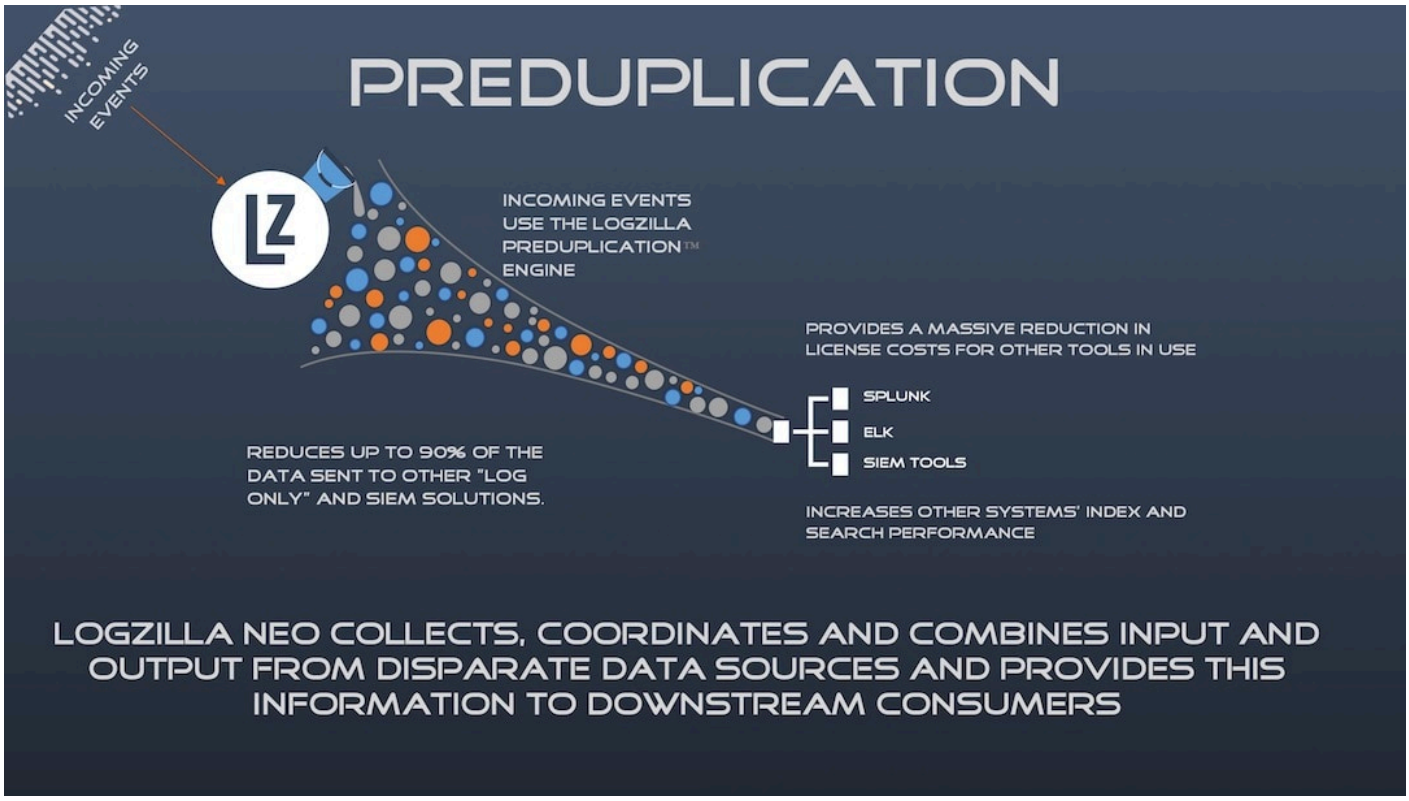
LogZilla collapses repeated events at ingest through deduplication windows, then forwards the reduced stream to syslog, file, Splunk HEC, or SNMP receivers

Forwarding To Downstream Receivers · Generated June 12, 2026 · logzilla.ai/docs/forwarding-module/dedup-forwarder-introduction

Deduplication and Forwarder Overview

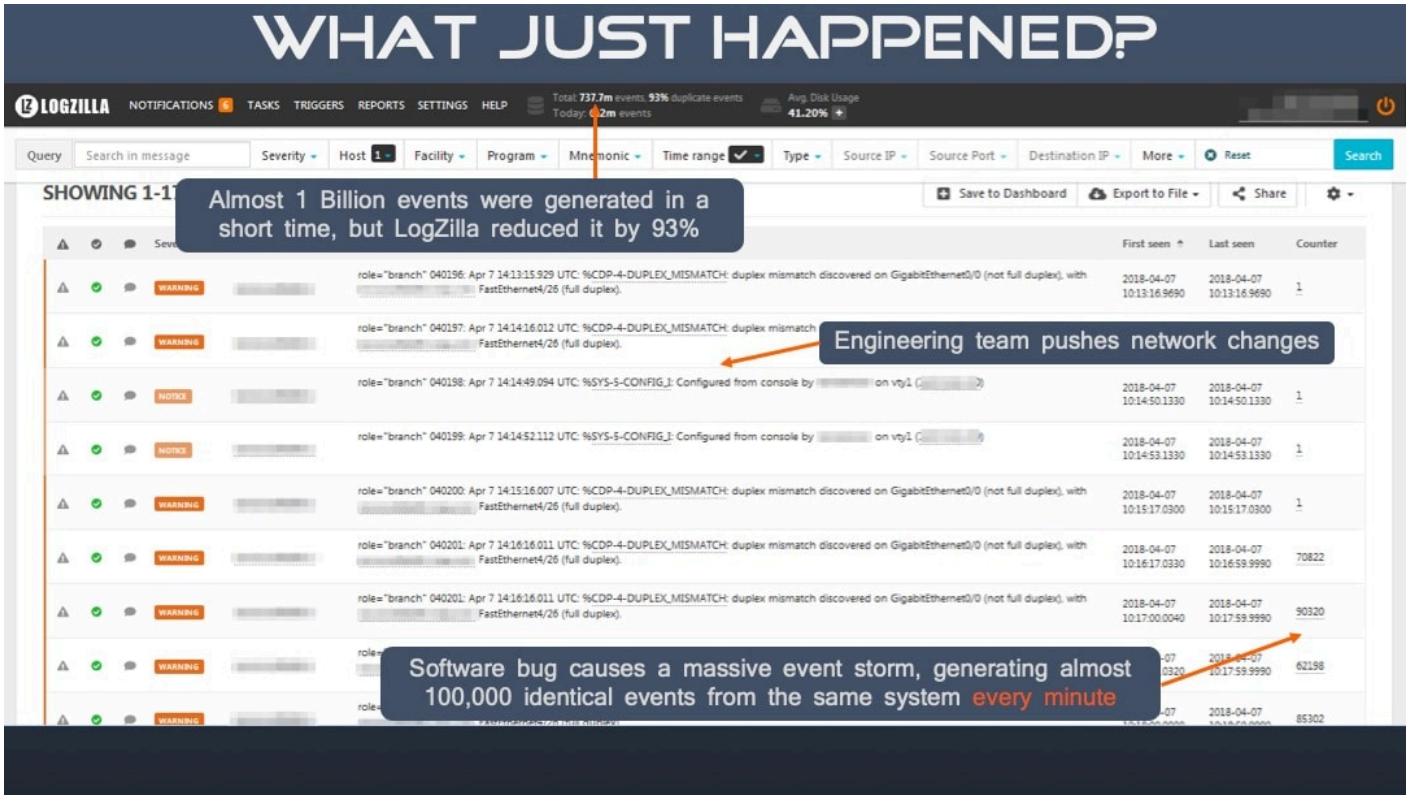
LogZilla performs deduplication at ingest to collapse repeated events into a single forwarded event per window, with a count of occurrences. This reduces downstream CPU, storage, and licensing impact during event storms, while preserving the signal needed for operations.

Deduplication process



In large environments, devices may emit the same event repeatedly (for example, during a link flap or authentication failure). Deduplication at ingest ensures downstream receivers handle a small, meaningful set of events rather than an overwhelming flood.

Event storm example



In the scenario above, hundreds of thousands of duplicate events could be reduced to just a handful, with a counter indicating how many times each unique event occurred.

Forwarding options

LogZilla can forward events as a log or a trap, regardless of how the event was received (syslog, HTTP receiver, traps, or app inserts). The module applies deduplication first, then applies any forwarder rules, and finally delivers to the configured destination(s).

Forwarder matching and rules

Forwarders use the same `match` structure as rewrite rules to decide which events to send to a destination. Each forwarder can include an optional `match` list; only events that satisfy all match entries are forwarded.

For a full description of `match` semantics, common fields, and supported operators, see Rewrite Rules (https://www.logzilla.ai/docs/data-transforms/rewrite-rules).

Regex matching example

Use regular expressions for complex matching against the message text:

```
# /etc/logzilla/forwarder.d/interface-monitoring.yaml
match:
  - field: message
    op: "=~"
    value: "Interface .* is (up|down)"
type: syslog
target: central-log-collector:514
name: "Interface State Change Forwarder"
```

This forwards events where the `message` indicates an interface state change, regardless of whether it transitioned up or down.

Configuration locations and workflow

- `/etc/logzilla/forwarder.yaml` (or `.json`): Global forwarder file. It is created automatically on first use with sane defaults (`window_size: 60`, `fast_forward_first: true`). Most users should not modify this file directly.
- `/etc/logzilla/forwarder.d/`: Recommended location. Place individual forwarders here, one forwarder per file (YAML or JSON). This keeps configurations modular and easier to manage.

After adding or changing forwarders:

```
# Inspect the merged forwarder configuration (useful for syntax checks)
logzilla forwarder print

# Reload the forwarder to apply changes
logzilla forwarder reload
```

To enable the Forwarder module feature flag:

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

For full CLI usage, refer to:

- System commands (https://www.logzilla.ai/docs/command-line-tools/system-commands)
- Data commands (forwarder) (https://www.logzilla.ai/docs/command-line-tools/data-commands)