Overview

syslog-ng is a modern syslog daemon that's focused on flexibility and portability. It also has an easy to use configuration format, that helps you ship your logs to Logsene in 3 steps:

1. **sources**. syslog-ng can listen to local syslog traffic on /dev/log, can tail filesand more
2. **destinations**. You can send your logs to Logsene via UDP, TCP or RFC-5425 TLS Syslog
3. **bind sources to destinations**. Once you have your sources and destinations defined, you have to define paths by linking sources and destinations with the log statement

Configure Sources

Sources enable syslog-ng collect the logs you want to send to Logsene. You can use system() to collect all the local syslog messages from that system. You can put this at the beginning of your **/etc/syslog-ng/syslog-ng.conf**, along with your configuration version. We recommend running version 3.3 or later:

```
@version:3.3
source local_logs {
    system();
};
```

To download a more recent version of syslog-ng than the one on your system, take a look at the download page of syslog-ng.

Tailing Files

syslog-ng can also listen to other inputs, such as files, TCP or UDP.

To tail a file, you'll use the file() source and point it to the right file. If that file doesn't contain syslog-formatted messages, you'll need to tell syslog-ng not to try and parse it, and if you need multi-line support, you should specify that, too:

```
source jetty_log { file("/var/log/jetty" flags(no-parse) multi-line-mode(indented)); };
```

Configure Destinations

To properly configure your destination, you'll first need to decide how to authenticate to Logsene. There are two options: with your Logsene application token (which is what we recommend), or by authorizing your public IPs from the Logsene UI.

After you choose your authentication method, you'll configure your destination according to the protocol you prefer: UDP, TCP or TLS. For all cases, your endpoint will be **logsene-receiver-syslog.sematext.com**

Token Authentication

To enable token authentication, get the token of your Logsene application from the list of Logsene applications. Then, put it in the template() statement of your Logsene destination. See working examples below. The end result is a CEE-formatted JSON syslog message that contains your token in the **logsene-app-token** field.

IMPORTANT: For JSON formatting that enables token authentication to work you'll need syslog-ng version 3.3 or later and the **libjson** syslog-ng plugin. The libjson plugin is typically provided by the **syslog-ng-json** package of your distribution.

IP-based Authentication

The alternate method for authentication is by pre-authorizing your public IP in Logsene's UI. Here is a complete guide on how to do that. If you choose this path, you don't need the template() statement from the code snippets below.

UDP

To send logs over the fire-and-forget UDP protocol, your can add a destination like the one below to **/etc/syslog-ng/syslog-ng.conf**. If you're using token-based authentication, you'll have to fill in your Logsene application token. If you've authorized your public IP, the template() statement should be removed:

**** Expand source

```
destination logsene {
    syslog("logsene-receiver-syslog.sematext.com"
      transport("udp")
      port(514)
# template() statement should be removed if you authorized your IP
    template("@cee: $(format-json --pair message=\"$MSG\" --pair logsene-app-token=\"LOGSENE_
    );
};
```

TCP

For TCP, the destination will look similar to the one for UDP, the only change is in the protocol() statement. Make sure your real token is there or remove the template() statement if you've authorized your IP:

**** Expand source

```
destination logsene {
    syslog("logsene-receiver-syslog.sematext.com"
      transport("tcp")
      port(514)
# template() statement should be removed if you authorized your IP
    template("@cee: $(format-json --pair message=\"$MSG\" --pair logsene-app-token=\"LOGSENE_
    );
};
```

TLS

For configuring RFC-5425 TLS Syslog, there are two steps. First, you need to set up the certificates:

**** Expand source

```
mkdir /opt/syslog-ng
cd /opt/syslog-ng
wget https://apps.sematext.com/cert/DigiCertCA.pem           # md5sum is 9e028401b52ca7453f6b
wget https://apps.sematext.com/cert/DigiCert_Global_Root_CA.pem  # md5sum is 3816293340b05c52

# openssl x509 -subject_hash -noout -in DigiCert_Global_Root_CA.pem
# 3513523f
ln -s DigiCert_Global_Root_CA.pem 3513523f.0

# openssl x509 -subject_hash -noout -in DigiCertCA.pem
# 85cf5865
ln -s DigiCertCA.pem 85cf5865.0
```

Then, you'll configure the destination in a similar fashion to plain TCP, except for adding tls() statement and pointing it to your newly created certificates directory and **changing the port to 10514**:

**** Expand source

```
destination logsene {
    syslog("logsene-receiver-syslog.sematext.com"
      transport("tcp")
      port(10514)
# template() statement should be removed if you authorized your IP
```

```
    template("@cee: $(format-json --pair message=\"$MSG\" --pair logsene-app-token=\"LOGSENE_
      tls(
        ca_dir("/opt/syslog-ng")
        peer_verify("required_trusted")
      )
 );
};
```

Bind Sources to Destinations

After configuring your source and destination the last step is binding the two:

```
log {
    source(local_logs); destination(logsene);
};
```

If you're tailing files or you defined other sources, you need to bind them to the **logsene** destination in order to have those messages shipped to your Logsene application as well. For example:

```
log {
    source(jetty_log); destination(logsene);
};
```

Then, restart syslog-ng and you should see your logs in the Logsene UI or Kibana.

Tag Your Logs

From your syslog messages, Logsene will populate a number of special fields, such as the **source** and **host**. You can also configure syslog-ng to add a tag to logs matching certain criteria. This is useful when you want to quickly identify a special kind of logs. For example, you could tag events that come to the "kern" facility with a severity/level of "err" as "kernel errors".

To achieve this, you can define a filter that matches such events. Then, you'd define a destination similar to the ones described above, where you'd add a **tags** field. Finally, you'd define a **log** statement where you bind your source, the newly defined filter and the newly defined destination:

```
filter user_tests { facility(kern) and level(err) };

destination logsene_tests {
    syslog("logsene-receiver-syslog.sematext.com"
      transport("tcp")
      port(514)
    template("@cee: $(format-json --pair message=\"$MSG\" --pair tags=\"kernel errors\" --pair
    );
};

log { source(all_syslog); filter(user_tests); destination(logsene_tests); flags(final); };
# main Logsene "log" statement will be defined below
```

Notice the **final** flag to this log statement - this prevents syslog-ng from sending matched events twice (once with tags and once without). Make sure you place the log statement with tags before your main Logsene log statement.