
OpenNTI Documentation

Release 0.0.1

Damien Garros, Efrain Gonzalez, Michael Pergament, Pablo Sagre

November 22, 2016

1	How to report feedback / participate in the project	3
2	User Documentation	5
3	Additional Documentation	7
3.1	How to install or upgrade	7
3.2	Customize OpenNTI	8
3.3	Architecture description [WIP]	8
3.4	Data Collection Agent	9
3.5	Data Streaming Collector	10
3.6	Events	11
3.7	Dashboard Generator	11
3.8	Dashboard Library	13
3.9	Troubleshooting Guide	13

OpenNTI is a container packaged with all tools needed to collect and visualize time series data from network devices. Data can be collected from different sources:

- **Data Collection Agent** : Collect data on devices using CLI/Shell or Netconf
- **Data Streaming Collector** : Take all data streamed by Juniper devices as Input (JTI, Analyticsd, soon Open-config with gRPC)
- **Statsd interface** : Accept any Statsd packets

It's pre-configured with all tools and with a default dashboard .. **Send it data, it will graph it**

Thanks to docker, it can run pretty much anywhere on server, on laptop ... on the device itself

More detailed description of a project can be found [here](#) (including a series of videos on how to use it):

How to report feedback / participate in the project

For any issues please open an issue on [Github](#). For comments, suggestions or questions please use our [Google-Group](#)

To participate, please:

- Fork the project
- Send us a pull request

Note: if you are planning significant changes, please start a discussion first.

Contributions are more than Welcome

User Documentation

- *Installation*
- *Dashboards*
- *Troubleshooting*

Additional Documentation

- *Architecture*
- *Input Plugins*

3.1 How to install or upgrade

3.1.1 Requirements

The requirements is to have docker and docker-compose installed on your Linux server/machine. Instructions to install are available below

- docker
- docker-compose

It's also available for:

- Mac
- Windows

3.1.2 How to Install/Start

OpenNTI is available on [DockerCloud](#) and this project provide scripts to easily download/start/stop it.

```
git clone https://github.com/Juniper/open-nti.git
cd open-nti
make start
```

Note: On Ubuntu, you'll have to add "sudo" before the last command

By default it will start 3 containers and it's working in **non-persistent mode**, once you stop it all data are gone. It's possible to start the main container in **persistent mode** to save the database outside the container, by using the startup script `make start-persistent`. *Persistent mode on Mac OS requires at least v1.12*

3.1.3 How to update

It's recommended to upgrade the project periodically, both the files from github.com and the containers from Docker Hub. You can update easily with

```
make update
```

3.2 Customize OpenNTI

3.2.1 Customize container's name and ports

All port numbers and names used by start/stop scripts are centralized in one file : **open-nti.params**, you can easily adapt this file with your own port numbers or names.

Note: It's mandatory if you are planning to run multiple instances of OpenNTI on the same server.

3.2.2 Customize the container itself

If you want to make some modifications, you can always build the container yourself using the script `make build`.

Note: The first time you run `make build`, it will take 10-15min to download and compile everything but after that it will be very fast

3.3 Architecture description [WIP]

OpenNTI architecture is designed to be modular. the main components are a Timeserie Database(influxdb) and a graphical interface (grafana)

Based on the need, containers can be added or removed to add functionalities.

3.3.1 Docker compose

All containers are started using `docker-compose.yml`

```
./docker.start.sh
```

You can create your own docker-compose file and pass it

```
./docker.start.sh <my docker compose file>
```

3.3.2 List of available Plugins

JTI

Event / Syslog

Input plugin container

- <https://github.com/Juniper/open-nti-input-syslog>
- <https://github.com/Juniper/open-nti-input-jti>

3.4 Data Collection Agent

3.4.1 Configuration

data/hosts.yaml In `data/hosts.yaml` you need to provide the list of devices you want to pull information from. For each device, you need to indicate the name and one or multiple *tags* (at least one). Tags will be used later to know which credentials should be used for this device and which commands need to be executed.

```
<hostA>: <tag1> <tag4>
<hostB>: <tag1> <tag4>
<hostC>: <tag2> <tag4> <tag5>
<hostD>: <tag1> <tag4> <--- Those tags relate the Hosts with the credentials and the commands to use
```

Example

```
mx-edge011: edge mx madrid bgp mpls
mx-agg011: agg mx madrid bgp isis
qfx-agg022: agg qfx munich bgp
qfx5100-02: tor qfx madrid isis
```

Note: The default configuration assumes that hosts defined in `hosts.yaml` can be resolved with DNS. If your hosts don't have a DNS entry, it's possible to indicate the IP address in the `hosts.yaml` file instead of the name:

```
192.168.0.1: edge mx madrid bgp mpls
```

To avoid using IP addresses in the dashboard, you can use the device hostname defined in the configuration instead of the value defined in `hosts.yaml` by setting the parameter `use_hostname` to `true` in `open-nti.variables.yaml`:
`use_hostname: True`

data/credentials.yaml

You need to provide at least one credential profile for your devices:

```
jdi_lab:
  username: '*login*'           (Single quote is to force to be imported as string)
  password: '*password*'       (Single quote is to force to be imported as string)
  method: password             (other supported methods 'key' and 'enc_key' for ssh Key-Based Authent.)
  key_file: './data/*key_file*' (optional: only applies if method key or enc_key is used, it must be local)
  tags: tag1 tag2
```

data/commands.yaml

```
generic_commands: <--- You can name the group as best fits you
  commands: |
    show version | display xml <--- There is no limit on how many commands can be added into a group
    show isis statistics | display xml <--- Before adding a command, confirm that there is a related command
    show system buffers
    show system statistics icmp | display xml
    show route summary | display xml
  tags: tag1 tag2
```

3.4.2 Execution periodic

To collect data periodically with the **Data Collection Agent**, you need to setup a cron job inside the container. As part of the project, open-nti is providing some scripts to easily add/remove cron jobs **inside** the container **from** the host.

Scripts provided:

- **make cron-add:** Create a new cron job inside the container
- **make cron-show:** Show all cron jobs configured inside the container
- **make cron-delete:** Delete a cron job inside the container for a specific tag

To start cron job to execute commands specified above for specific tag every minute:

```
make cron-add TAG=lab
```

To start cron job for more than one tag at the same time:

```
make cron-add TAG='lab prod'
```

To start cron job to execute commands specified above for specific tag every 5 minutes:

```
make cron-add TAG=tag1 TIME=5m
```

To start cron job to execute commands specified above for specific tag every hour:

```
make cron-add TAG=tag1 TIME=1h
```

To stop cron job for specific tag:

```
make cron-show TAG=tag1
```

Note: If you want to configure the cron job yourself, open-nti use this command: `/usr/bin/python /opt/open-nti/open-nti.py -s --tag <tag>`

3.4.3 Junos Parsers

3.5 Data Streaming Collector

Currently the collector accept:

- Analyticsd (QFX5k) streams in JSON/UDP on port **UDP/50020**
- Juniper Telemetry Interface (MX/PTX) streams in GPB/UDP on port **UDP/50000**

Important: it's important that all devices have the correct time defined, it's recommended to configure NTP everywhere

3.5.1 statsd interface

open-nti is using telegraf to support statsd Statsd is a popular tool to send metrics over the network, it has been designed by etsy.

More information below:

- https://github.com/etsy/statsd/blob/master/docs/metric_types.md
- <https://github.com/influxdata/telegraf/tree/master/plugins/inputs/statsd>

Here is an example of how to insert statsd data into the Database

```
root@d3e82264a08b:/# echo "opennti,device=qfx5100,type=int.rx:100|g" | nc -w 1 -u 127.0.0.1 8125
```

opennti define the serie device=qfx5100,type=int.rx will be converted as tagl 100 is the value g indicate gauge

3.6 Events

By default dashboards are configured to display some “events” that are stored in the database into the serie “events” Their are multiple ways to record entry in the events serie

3.6.1 Insert events via syslog

open-nti will access events in the syslog format on port **UDP/6000**. The goal is not to send all syslog but only relevant information like Commit or Protocol Flaps

To send only one syslog at commit time you can use the configuration below

```
set system syslog host 192.168.99.100 any any
set system syslog host 192.168.99.100 match UI_COMMIT_COMPLETED
set system syslog host 192.168.99.100 port 6000
```

3.6.2 Insert events in the database directly

It’s possible to insert events with just a HTTP POST request to the database, here is an example using curl

```
curl -i -XPOST 'http://10.92.71.225:8086/write?db=juniper' --data-binary 'events,type=Error text="BG'
curl -i -XPOST 'http://10.92.71.225:8086/write?db=juniper' --data-binary 'events,device=qfx5100-01,t
```

Note: any system that knows how to generate a HTTP POST request can inject an event. its very utile if you have a script/tool that run some tests to keep track of when major events happen

3.7 Dashboard Generator

OpenNTI integrate a Dashboard generator based on Python and Jinja2.

This dashboard generator can be very useful in many situations:

- Convert JTI graphs to the new variables names
- Create graphs for the new JTI sensors (LSP, FW etc ..)
- Add templating for interface
- Create Dashboard for Netconf in mode 2 & 3
- Create Dashboard on demand and more personalized

In a nutshell, it templated a grafana dashboard into multiple pieces:

The skeleton of the dashboard:

- **Rows**, composed of multiple panels or graphs
- **Graphs**,
- **Annotations**, events overlay on the graphs
- **Templatings**, drop down menu to narrow the scope

To generate a dashboard you need to create a yaml file that indicate: the title, which rows, which annotations etc ..

```
title: Data Steaming Collector ALPHA
template: "dashboard_base.j2"

tags:
  - opennti

rows:
  - int-traffic.yaml
  - int-queue.yaml
  - int-buffer.yaml

templatings:
  - host_regex.yaml
  - interface.yaml

annotations:
  - commit.yaml
  - bgp_state.yaml
```

To generate the dashboard based on this config file, you just have to call this command line

```
cd dashboards/
python gendashboard.py --file data_streaming_collector.yaml
```

The rows are defined in the directory `templates/rows/` and the graphs in the directory `templates/graphs/`. The idea is to define which template for each configuration file, so we don't need to turn everything into a variable in the templates. If 2 graphs are very different we can just have different templates.

It will keep the YAML file light and easily readable.

Note: You can browse all *rows*, *graphs*, *templatings* and *annotations* available in the [Dashboard Library](#)

3.8 Dashboard Library

3.8.1 Graphs

3.8.2 Rows

3.8.3 Annotations

3.8.4 Templatings

3.9 Troubleshooting Guide

To check if containers are running, execute the following command. By default you should have 3 containers running

```
docker ps
```

To force containers to stop, execute

```
make stop
```

To access the CLI of the main container for debug, Start a SSH session using the `insecure_key` provided in the repo and the script “`docker.cli.sh`”

```
make cli
```

For the Input containers named `__open-nti-input-*` you can access the logs directly from docker by running :

```
docker logs <container name or ID>
```

3.9.1 Data Collection Agent

Q - I configured hosts/credential/commands.yaml files but I'm not seeing anything on the dashboard

To make sure everything is working as expected, you can run the Data Collection Agent in debug mode

```
make cron-debug TAG=lab
```

3.9.2 Data Streaming Collector

Q - I'm streaming data from devices but I'm not seeing anything on the Dashboard

To reach the dashboard, traffic have to go through the following path: **Device** >(A)> **Host** >(B)> **Container** >(C)> **Fluentd** >(B)> **InfluxDB** >(E)> **Grafana**

A - Check the timestamp on the devices and on the server

Timestamp MUST match on both side, the server and the junos devices. It's the most common issue.

B - Check that traffic is reaching the Host

The best solution is to use TCPDUMP on the Host and filter on destination port

```
On Unix/Mac
tcpdump -i <ingress interface> -n dst port <dest port number>
```

C - Check that traffic is reaching the container

The best solution is to use TCPDUMP inside the container

```
./docker.cli.sh  
tcpdump -i eth0 -n dst port <dest port number>
```

RPF check might be a problem if you see incoming packets in A but not in B. If you e.g. use Src IP for which there is no route entry on host OS (Ubuntu does RPF check as default), packets would be discarded.

D - Check Fluentd**

Check fluentd logs, inside the container

```
docker logs opennti_input_jti
```

Nothing should be printed if everything is right

E - Check if data is properly reaching the database

- connect on Influxdb management interface with a browser on port 8083
- Select Juniper as database on top right corner
- Run query `show measurements` to see what is present
- Execute query for `SELECT * FROM "<measurements>"`

Note:

Destination tables will vary depending of the incoming traffic

- For MX > jnpr.jvision
 - For QFX5100/EX4300 > jnpr.analyticsd
-