gNMI Installation Guide

Version 2019.1.0, 02 October 2019

Registered Address	Support	Sales
26, Kingston Terrace, Princeton, New Jersey 08540, United States		
		+91 80 4850 5445
http://www.rtbrick.com	support@rtbrick.com	sales@rtbrick.com

©Copyright 2020 RtBrick, Inc. All rights reserved. The information contained herein is subject to change without notice. The trademarks, logos and service marks ("Marks") displayed in this documentation are the property of RtBrick in the United States and other countries. Use of the Marks are subject to RtBrick's Term of Use Policy, available at https://www.rtbrick.com/privacy. Use of marks belonging to other parties is for informational purposes only.

Table of Contents

1. gNMI Overview

The RtBrick gNMI service is designed to run on an RtBrick fullstack (RBFS) container. The service communicates with the BDS (Brick Data Store) through localhost.

There are two possibilities for communicating with the gnmi target:

a. By connection through ONL, which could be considered out-of-band management. (The ONL uses NAT forwarding to the LXC container. In the future, it might connect through CtrlD.)

b. By using inband management.

These relationships are shown in Figure 1.

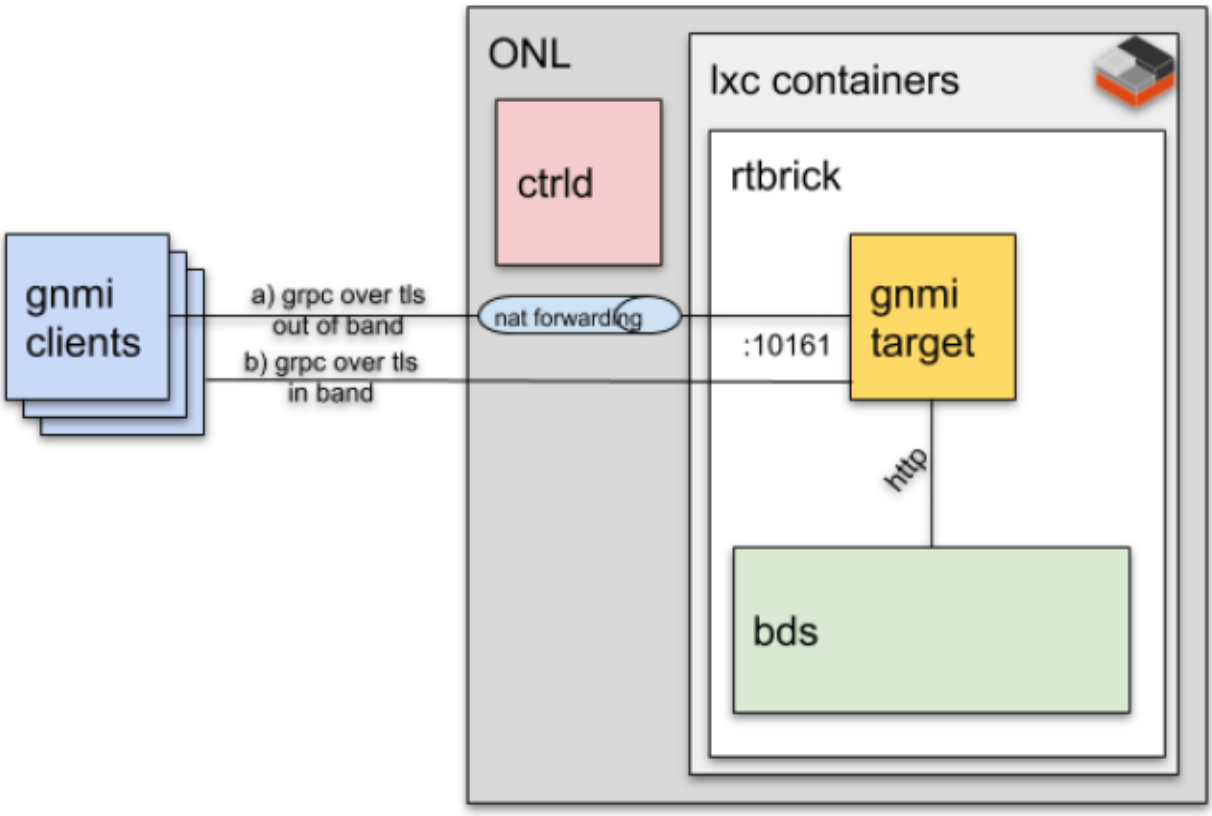

Figure 1. gNMI and Containers

The default interface and port is :10161. The gnmi target listens to port 10161 on all interfaces.

2. gNMI Installation

The easiest way to install the gnmid is to use the debian package from universe.

```
# install the package
dpkg -i rtbrick-gnmi_*.deb
```

The deb package contains all the commands needed to bring up the service.

2.1. RtBrick Source List

In order to install the gNMI demon, the rtbrick source list must be on the system.

Check for the file /etc/apt/sources.list.d/rtbrick.list, and if it does not exist, then create one and add the provided rtbrick repositories.

```
deb [ arch=amd64 ] http://pkg.rtbrick.com/ubuntu/<<customer>>-staging
aries multiverse
deb [ arch=amd64 ] http://pkg.rtbrick.com/ubuntu/tools/<<customer>>-
staging aries multiverse
```

2.2. Debian Package from universe

The easiest way to install the rtbrick-gnmid is to use the debian package.

```
# re-synchronize the package index files from their sources
sudo apt-get update
# show all available package versions of the rtbrick gnmi package
apt-cache show rtbrick-gnmi | grep Version
# installs the latest version of rtbrick gnmi
apt-get install rtbrick-gnmi
```

2.3. Installation Status

You can use the command service rtbrick-gnmid status to get additional information about the service.

To verify the installed version of gnmid, use the command gnmid -version.

2.4. Log Files

The log files are stored at /var/log/rtbrick-gnmid.log, They are rotated with logrotate, and the configuration for the rotation process can be found at

/etc/logrotate.d/rtbrick-gnmid.

2.5. NAT Forwarding Rule

In order to set up NAT forwarding on the ONL, use the following command:

```
iptables -t nat -A PREROUTING -p tcp --dport 10161 -j DNAT --to-
destination 10.0.3.10:10161
iptables -L -t nat
```

2.6. What Is Installed

In the make file, the installation target shows where the different files are installed on the system. Here is a brief overview of the locations:

```
# Executeables
gnmi_target -> /usr/local/bin/gnmid
gnmid-wrapper.sh -> /usr/local/bin/gnmid-wrapper.sh
# Service file
rtbrick-gnmid-service -> /etc/init.d/rtbrick-gnmid
# Logrotation
rtbrick-gnmi-logrotate -> /etc/logrotate.d/rtbrick-gnmid
# Certificates
certs -> /etc/rtbrick/gnmi/certs/
# Configuration
bds-config.json -> /etc/rtbrick/gnmi/bds-config.json
rtbrick -> /etc/rtbrick/gnmi/rtbrick
```

3. gNMI Configuration

The service runs the gnmid-wrapper.sh script. The script calls the gnmid executable, where major configuration parameters are stored:

```
# Certificate Authority certificate file.
# Defines the root certificate authority that the server uses to verify
a
# client certificate
-ca=/etc/rtbrick/gnmi/certs/ca.crt
# Server Certificate file is the certificate which is presented to
# the client it identifies the server
-cert=/etc/rtbrick/gnmi/certs/server.crt
# Private key file to read the server certificate
-key=/etc/rtbrick/gnmi/certs/server.key
#Skip TLS validation.
-insecure
# log level
-stderrthreshold=2
# log toggle level
-v=0
# configuration file
-deviceconfig=/etc/rtbrick/gnmi/bds-config.json
# Other possible flags
# Disable TLS validation. If true, no need to specify TLS related
options.
-notls
# If specified, uses username/password credentials.
-username
# The password matching the provided username.
-password
```

3.1. BDS Configuration

The device config file alias BDS Config is used to point the gnmid process to the right endpoint to get the data from BDS.

```
{
    "base-url": "http://localhost",
    "read-additional-port-from-box-config": true,
    "box-config-folder": "/var/run/rtbrick"
}
```

The major points about this configuration type are:

- The base url is localhost.
- If the read-additional-port-from-box-config is set to true: then this means

localhost by itself is not used, but the combination localhost:port. The port value for each RTB demon is read from the additional box configuration folder (in this case /var/run/rtbrick).

- The box-config-folder is where the configuration is typically found.

3.2. Configuration Through CTRLD

```
{
    "base-url": "http://<boxurl>:<ctrld-port>/api/application-rest-
proxy/<container-name>/%[1]s",
    "read-additional-port-from-box-config": false,
    "box-config-folder": "./rtbrick"
}
```

The major points about this configuration type are:

- The base-url is as follows:

```
http://<boxurl>:<ctrld-port>/api/application-rest-proxy/<container-
name>/%[1]s
```

- Typically, the CTRLD-port is 19091.
- The ending of %[1]s inserts the RTB demon name.
    - The read-additional-port-from-box-config is false, because there is no additional port needed.
    - The box-config-folder and configuration file must be provided for each demon.

A typical demon configuration file, located through box-config-folder, might look like this:

```
{
    "bd_info": {
        "process_name": "fwdd",
        "rest_port": 5002
     }
}
```

4. gNMI Implementation Details

This section shows some interesting aspects in dealing with a connection-oriented protocol like gRPC, and how this implementation avoids some pitfalls of protocols like this. It also gives an overview of the gNMI model implemented, and describes some mapping details between the models and the RBFS models.

4.1. Congestion Prevention

This idea comes from Rob Shakier and Carl Lebsack of Google.

Since gRPC is a connection-oriented protocol, it could happen that the process cannot push data out as fast as the data is taken in. This means the system becomes congested, and queues all the data for a client that can't receive the information fast enough.

This possibility is shown in Figure 1.

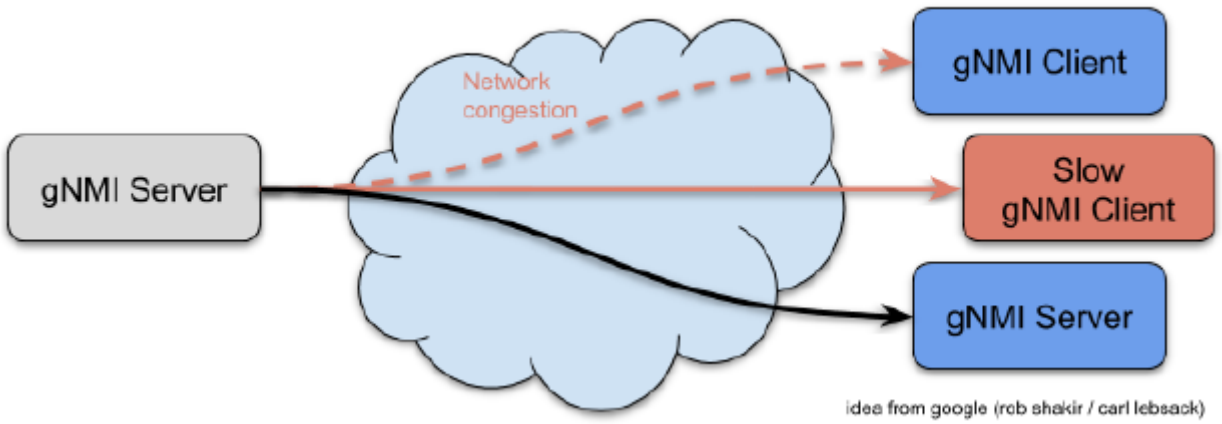

Figure 2. Network Congestion Due to a Slow Client

There could be a couple of different reasons for the slow client. There could be some network congestion between the server and the client, or the client is poorly performing and is not receiving data very fast. The system has to be protected against that. Also, TCP's guaranteed delivery can have negative effects on performance.

The solution is an application level coalesce-queue. This allows detection of the backpressure from the server side with blocking sent over gRPC. Figure 2 shows how such a queue works.

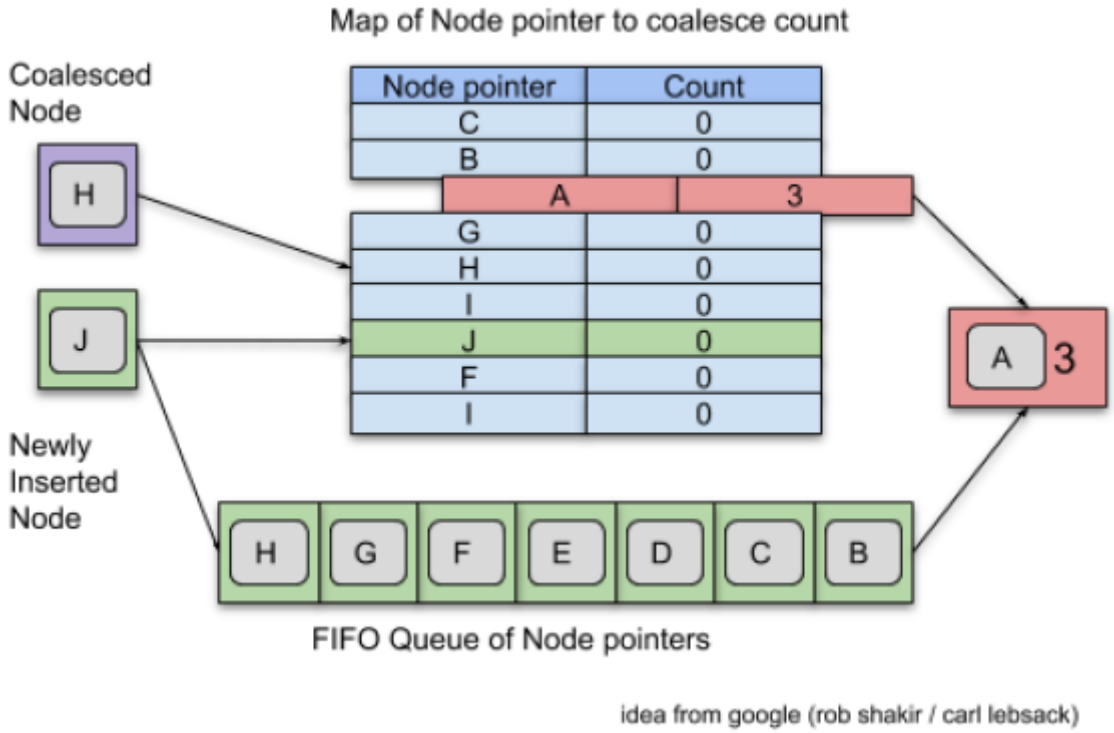

Figure 3. Schema of a coalesce-queue

The idea is only to deliver the freshest data, so the old stale data does not have to be queued. If there is an update for a key which is already in the queue, but has not yet sent, then the newer data will be sent.

For each gNMI client there is a coalescing queue provided, as shown in Figure 3.

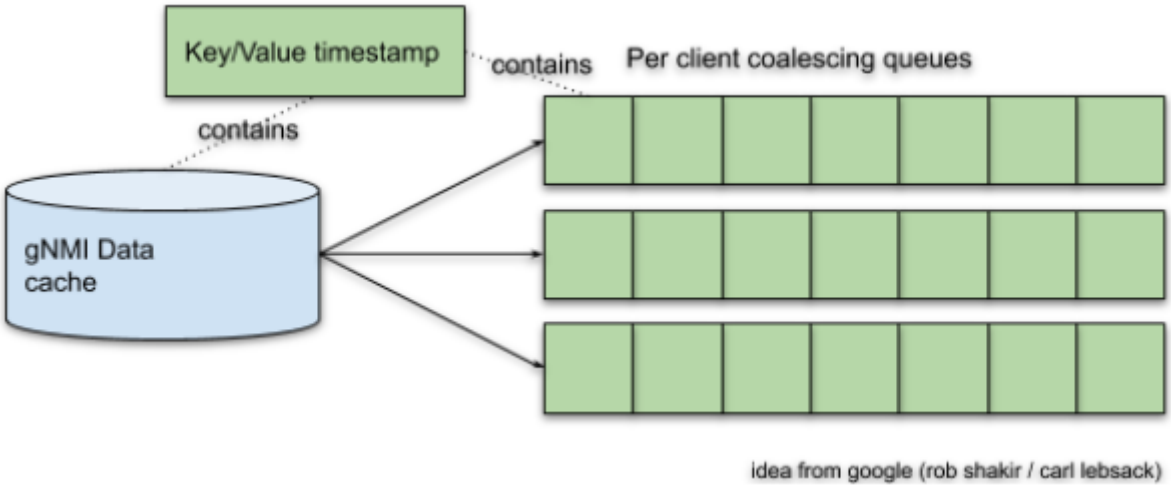

Figure 4. Queue per client

Using this scheme, one client does not affect another client.

4.2. Implementation Model Details

This gNMI implementation does not implement all attributes provided by the

openconfig models, but includes most of them. There is no list of attributes.not implemented.

All of the mentioned models can be queried using gNMI get or gNMI subscribe requests. The gNMI get request returns the queried path plus the subsequent leafs. The gNMI subscribe only returns leaf elements.

4.2.1. QoS

QoS is not implemented yet.

Additional Information can be found at QosMapping.

4.2.2. System

OpenConfig System is implemented rudimentarily.

Additional Information can be found at SystemMapping.

4.2.3. Interface

The **openconfig-interfaces, openconfig-if-ethernet, openconfig-if-ip** models are mapped to the RtBrick internal models. RtBrick distinguishes between IFP (Interface Physical), IFC (Container Interface) and IFL (Logical Interface).

In openconfig-interfaces there are two ways to describe containment:

- Aggregation is a bidirectional reference between two interfaces.
- Subinterfaces describe true containment.

Figure 4 shows the mapping between the open-config interface model possibilities and the RBFS interface model implementation.

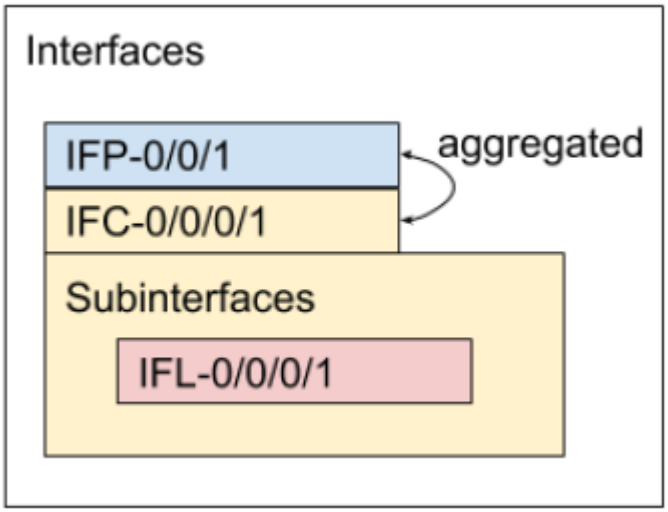

Figure 5. Interface Aggregation and Containment with Sub-interfaces

Additional Information can be found at InterfaceMapping.

4.2.4. BGP

RtBrick's BGP is mapped to the openconfig-bgp models.

Additional Information can be found at BgpMapping.

4.2.5. Platform

Additional Information can be found at PlatformMapping.

Openconfig components are divided into hardware and software components:

- Hardware components
  - OPENCONFIG_HARDWARE_COMPONENT_UNSET
  - OPENCONFIG_HARDWARE_COMPONENT_BACKPLANE
  - OPENCONFIG_HARDWARE_COMPONENT_CHASSIS
  - OPENCONFIG_HARDWARE_COMPONENT_CONTROLLER_CARD
  - OPENCONFIG_HARDWARE_COMPONENT_CPU
  - OPENCONFIG_HARDWARE_COMPONENT_FABRIC
  - OPENCONFIG_HARDWARE_COMPONENT_FAN
  - OPENCONFIG_HARDWARE_COMPONENT_FRU
  - OPENCONFIG_HARDWARE_COMPONENT_INTEGRATED_CIRCUIT
  - OPENCONFIG_HARDWARE_COMPONENT_LINECARD
  - OPENCONFIG_HARDWARE_COMPONENT_PORT