gNMI Adapter for ConfD

tail-f a Cisco company

Table of Contents

gNMI Adapter for ConfD

1. Introduction

ConfD is a data model driven management plane framework which provides a variety of northbound interfaces (NBIs): NETCONF, RESTCONF, JSON-RPC WebAPI, CLI, and SNMP. ConfD also provides application development APIs in a variety of language bindings. One of these APIs is the Management Agent API (MAAPI) which enables the development of additional NBIs.

Recently, the gNMI (gRPC based) management protocol, which makes use of YANG data models, has become popular; primarily around OpenConfig. gNMI has mainly seen adoption for streaming telemetry. While gNMI does support configuration management, the limited configuration support of gNMI compared to NETCONF has made it less popular for configuration management. While ConfD does not directly provide a gNMI NBI, as mentioned above, it does provide APIs that can be used to implement (full or partial) gNMI functionality.

This application note describes a new example project called confdgnmi which has been started at ConfD-Developer on GitHub. The project shows how to implement an example gNMI Adapter using existing ConfD APIs.

2. What is the gNMI Interface?

Per the gNMI specification, gNMI is "a gRPC-based protocol for the modification and retrieval of configuration from a target device, as well as the control and generation of telemetry streams from a target device to a data collection system. The intention is that a single gRPC service definition can cover both configuration and telemetry - allowing a single implementation on the target, as well as a single NMS element to interact with the device via telemetry and configuration RPCs."

"All messages within the gRPC service definition are defined as protocol buffers (specifically proto3). gRPC service definitions are expected to be described using the relevant features of the protobuf IDL. The protobuf definition of gNMI is maintained in the openconfig/gnmi GitHub repository."

The gNMI service definition consists of four gRPC messages:

```
service gNMI {
  rpc Capabilities(CapabilityRequest) returns (CapabilityResponse);
  rpc Get(GetRequest) returns (GetResponse);
  rpc Set(SetRequest) returns (SetResponse);
  rpc Subscribe(stream SubscribeRequest) returns (stream
  SubscribeResponse);
}
```

The gNMI specification provides a detailed description of gNMI. Even though its high-level interface is simple (only four messages), the Request and Response parameters/messages may be complex. The most complex message is Subscribe, with streaming support for both Request and Response.

3. gNMI Adapter demo project

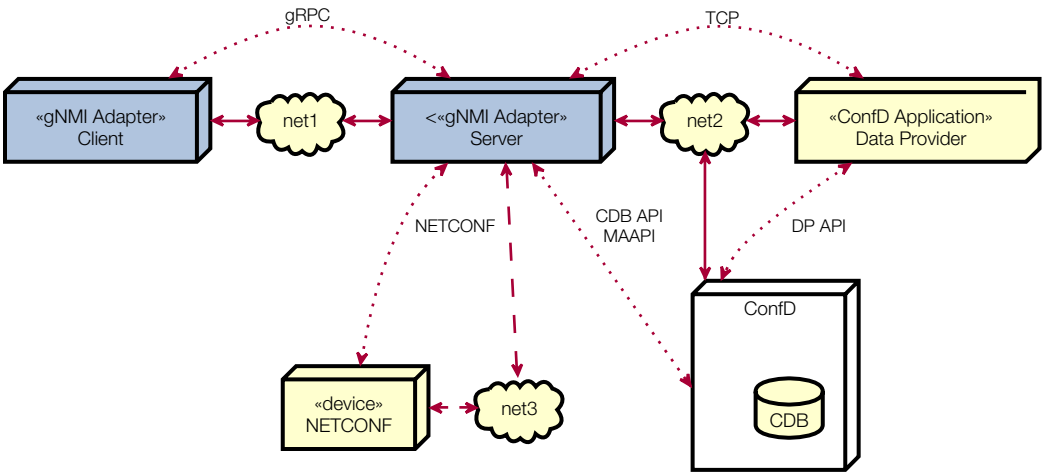
A new demo project called ConfD gNMI Adapter has been started at the ConfD-Developer on GitHub. This example project implements the gNMI Adapter over existing ConfD APIs.

This example gNMI Adapter project does not aim to provide full, production grade gNMI functionality. Rather, it should be seen as a starting point. The code can be directly used (or used with minor modification) with all gNMI messages. User and Developer documentation, description of dependencies, build process, usage scenarios, diagrams, and limitations are described in the project's documentation. The project includes automated tests to ensure that existing functionality is not broken during development.

The project can be already used (or used with minor updates) with all gNMI messages. It enables simple telemetry against ConfD devices.

3.1 Architecture Diagram

The following diagram shows the gNMI Adapter high-level architecture:



gNMI Adapter Server - This connects to ConfD and uses its API interfaces (MAAPI, CDB API) to provide the gNMI functionality. Alternatively, the design can be adapted to other non-ConfD devices with known management interfaces, e.g. devices supporting NETCONF, RESTCONF or mixed interfaces (dashed arrows).

gNMI Adapter Client - A gNMI client developed for testing and presentation of the functionality.

Data Provider - Source of external state/operational data. The TCP connection between the **Server** and the **Data Provider** is non-ConfD related. It enables the detection of changes of state/operational data not stored in the ConfD CDB database. This is described later in this document.

3.2 Implementation

The gNMI Adapter is implemented in Python and uses ConfD's Python language binding.

3.2.1 Dependencies

It is expected that Python3 is installed. The python and pip commands are from the Python3 environment. If not, use python3 or pip3 instead (or use, e.g., "sudo apt-get install python-is-python3").

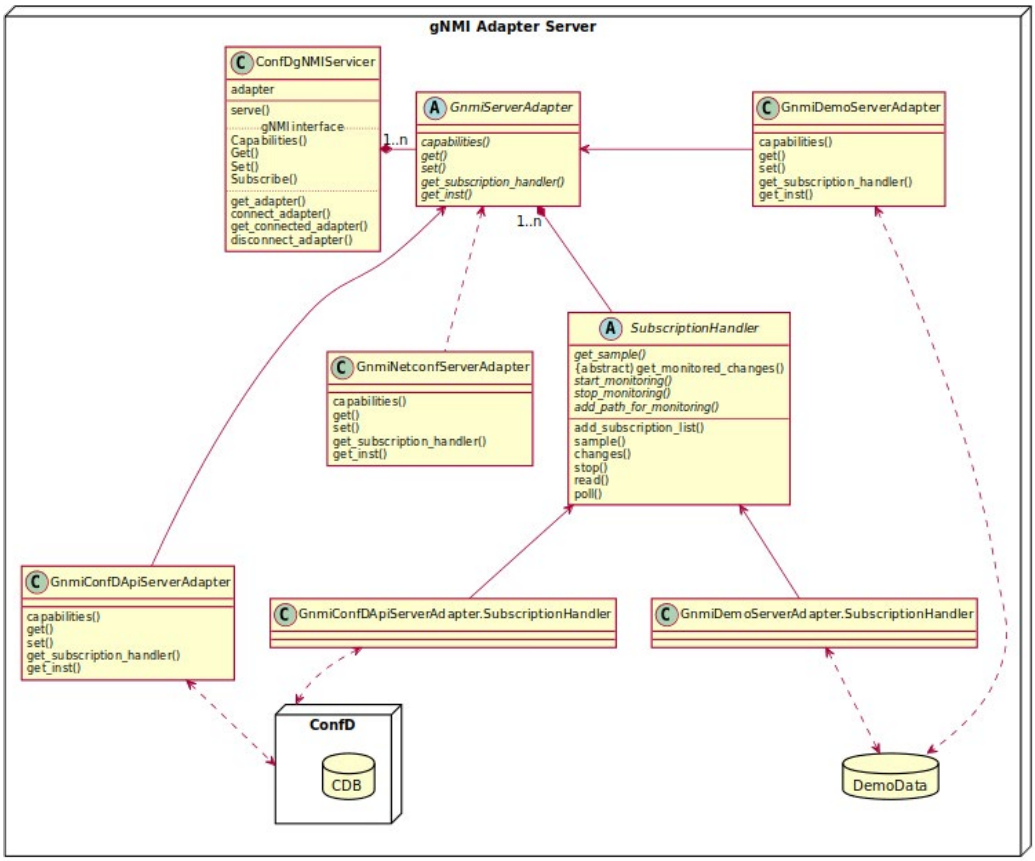
To build the gRPC/gNMI Python binding, the grpcio-tools Python package is needed ("pip install grpcio-tools"). For automated tests, Pytest is used ("pip install pytest").

ConfD Premium or ConfD Basic has to be installed according to the description in the package (README). To set up the ConfD environment, source confdrc as usual.

GNU Make is used for building the application and for running the tests. It should be available in most Linux distributions ("apt-get install build-essential").

3.2.2 Components

The following diagram shows the component/class structure of the gNMI Adapter server: There are two abstract classes called GnmiServerAdapter and SubscriptionHandler with interfaces to support gNMI. Derived classes implement the interface methods. The current implementation adds gNMI support for Demo and ConfD API variant. By adding a new derived class, we can add support for other implementations, e.g. NETCONF, if ConfD is not being used.



3.2.3 External Changes

State (operational) data can be stored in the ConfD CDB database or it can be provided to ConfD by an external application (called a Data provider). In the first case of data stored in CDB, ConfD can automatically monitor the changes to the operational data and inform the gNMI Adapter server the changes via subscriptions. In the second case of data is provided by a Data Provider, ConfD cannot monitor changes to the operational data. In this case, since the Data Provider application has all the information about changes to the operational data, it can inform the gNMI Adapter directly. A simple socket-based protocol, which is described in the documentation, was added to the gNMI Adapter server for this functionality . It is shown as a TCP connection between the gNMI Adapter server and the Data Provider in the architecture diagram above.

3.2.4 Build and Run

To build the gNMI Python binding, ConfD's fxs, files and initial configuration:

```
make clean all
```

The gNMI Adapter can be run in *demo* or *ConfD API* mode.

In the *demo* mode ConfD not required to be running. The gNMI Adapter partially emulates the ietf-interfaces.yang YANG data model and initial configuration. This mode is useful for testing, development, etc.

If you want to run the gNMI Adapter with ConfD then API mode is used. You can use the Makefile's "start" target to start ConfD with sample initial configuration.

Rebuild everything, start ConfD, and load sample configuration:

```
make stop clean all start
```

3.2.5 gNMI Adapter Server

To Start gNMI Adapter server in (ConfD) API mode:

```
./src/confd_gnmi_server.py -t api
```

3.2.6 Automated Tests

You can find pytest tests in the tests directory.

To run all tests:

```
./test.sh -s -v tests/
```

3.3 Usage Examples

Once the gNMI Adapter server is running, you can invoke gNMI operations with the client application.

Some examples from the documentation:

Get capabilities:

```
./src/confd_gnmi_client.py -o capabilities
```

Get values of list entries:

```
./src/confd_gnmi_client.py -o get --prefix /interfaces \
      --path interface
```

Set value of leaf element:

```
./src/confd_gnmi_client.py -o set --prefix /interfaces \
      --path interface[name=if_8]/type --val fastEther
```

Subscribe ONCE for list entry:

```
./src/confd_gnmi_client.py -o subscribe \
      --prefix /interfaces \
      --path interface[name=state_if_8]
```

Subscribe (POLL) for list entry:

```
./src/confd_gnmi_client.py -o subscribe -s POLL \
      --pollcount=2 --prefix /interfaces \
      --path interface[name=state_if_8]
```

Subscribe (STREAM) for list entry:

```
./src/confd_gnmi_client.py -o subscribe -s STREAM \
      --readcount=3 --prefix /interfaces \
      --path interface[name=state_if_8]
```

3.4 Limitations

The example gNMI Adapter implementation has some known limitations:

- Only `leaf, list` entries and `lists` (with one `key`) are supported
- All values (`TypedValues`) are strings (`string yal`).
- gNMI Path is converted to XPath or ConfD formatted path with simple string operations (no datamodel knowledge used).
- Not all gNMI subscription parameters are supported.
- Current implementation works only against ConfD's RUNNING DB.

4. ConfD-Developer on GitHub

The gNMI Adapter project can be found under ConfD-Demos in ConfD Developer on GitHub: https://github.com/ConfD-Developer/ConfD-Demos.

The `confdgnmi` folder contains source code, documentation, and code for tests.

The *GitHub workflow* Action checks the basic integrity of the project:

- Build
- Statistic analysis
- Automated tests (than do not require ConfD binding)

5. Summary and Lessons Learned

The gNMI protocol is based on gRPC and which allows the use of a management API in many environments and programming languages. The gNMI functionality can be added to ConfD with use of an external application. It may not be as efficient and performant as if it were added directly to the ConfD engine, but in many cases (like basic telemetry), it is more than sufficient.

If we take a look at the gNMI specification, we can see it evolved around the OpenConfig project. Some messages may seem complicated or duplicated (e.g. some variants of Subscribe behave as Get). There are too many parameters for implementation of STREAM subscriptions. Transport of data elements as gRPC messages (protobuf data) or bytes seems nice and performant, but breaks in general approach as server and client have to fully understand the YANG data model to encode and decode messages.
Text-based messages (JSON) do not differ too much from existing NETCONF or RESTCONF standards.

The Python implementation works well and for basic usage it does not present any performance issues. In case there is no extensive data processing and transfer (e.g. only basic telemetry), it is sufficient for most scenarios. Ports to the other languages (like C++) should be possible.

A drawback of gNMI is that it is not standardized as an IETF RFC. There is a NETCONF standard for model driven telemetry called YANG Push. There are pros and cons for both variants. The ConfD network programmability approach focuses around NETCONF. This example project demonstrates that gNMI support is possible as well. The gNMI Adapter is written in a modular (plugin) way, and it could be adapted to add support to other (not ConfD) management agents or interfaces, for example, to create a pure gNMI over NETCONF adapter.

6. For More Information

For more information about ConfD, visit https://www.tail-f.com

For more information about gNMI, visit
https://github.com/openconfig/reference/blob/master/rpc/gnmi

ConfD Developer on GitHub: https://github.com/ConfD-Developer

ConfD gNMI Adapter project on ConfD Developer:
https://github.com/ConfD-Developer/ConfD-Demos/tree/master/confdgnmi

tail-f a Cisco company

www.tail-f.com
info@tail-f.com

Corporate Headquarters

Sveavagen 25
111 34 Stockholm
Sweden
+46 8 21 37 40