Data Center Telemetry and Network Automation Using gNMI and OpenConfig

Contents

This white paper provides a technical overview of network automation and telemetry on Cisco Nexus® 9000 Series Switches using gNMI and OpenConfig. It covers the new methods used to subscribe to telemetry data from switches for real-time network visibility with gNMI and an open-source collector, Telegraf. The paper then describes gNMI automation capabilities along with use-cases that are illustrated using open source scripting tools.

Introduction

Network automation and telemetry have been at the forefront of technologies in data center networks in recent years. Network architects typically take a multipronged approach to network automation, which includes aspects of network provisioning and configuration that can be automated using scripts and tools. In addition, telemetry data and operational data from devices can be used to further automate tasks and close the loop on intent-based networking. Having a suite of network automation capabilities in the enterprise is critical for innovation and continues to be a powerful investment going forward.

The introduction of model-based network programmability on Cisco® switches in recent years can be considered trailblazing in how administrators automate network functions. The paradigm shift to data models on switches makes network automation a reality with the use of managed objects and their associated constructs using different toolchains. Cisco NX-OS now has new capabilities with OpenConfig and gRPC Network Management Interface (gNMI) support to provide an open and model-driven facility to automate data center networks.

Telemetry involves the collection of data from switches and their transmission to a receiver for monitoring. The ability to collect data in real time is essential for network visibility, which in turn helps in network operations, automation, and planning. In this paper, we describe gNMI with OpenConfig that is used to stream telemetry data from Cisco Nexus® switches. We also introduce the open-source time-series data collection agent, Telegraf, which is used to consume telemetry data. We then explore the different methods of network configuration and management using gNMI's network configuration capabilities.

gRPC and gNMI on Cisco Nexus 9000 Series Switches

There are different network configuration protocols available on Cisco Nexus 9000 Series Switches, including NETCONF, RESTCONF, and gNMI. All of these protocols use YANG as a data model to manipulate configuration and state information. Each of these protocols can use a different encoding and transport. This paper focuses on the gRPC Network Management Interface (gNMI) protocol, which leverages the gRPC Remote Procedure Call (gRPC) framework initially developed by Google. The gNMI protocol is a unified management protocol for configuration management and streaming telemetry. While NETCONF and RESTCONF are specified by the IETF, the gNMI specification is openly available at the OpenConfig repository.

Cisco Nexus switches introduced network automation using a Cisco proprietary gRPC agent in Cisco NX-OS Release 7.x. The agent, called "gRPCConfigOper", was used for model-driven network configuration with Get and Edit functions. This agent is planned to be deprecated in future releases.

Telemetry in the 7.x train was based on a dial-out model, where the switch pushed telemetry data out to telemetry receivers with gRPC as one of the transport methods supported.

Cisco NX-OS Release 9.3(5) enables gNMI support with the following gRPC operations: **CapabilityRequest, GetRequest, SetRequest,** and **SubscribeRequest**. The new gNMI feature in Release 9.3(5) includes network configuration management with Get and Set methods as part of the open gNMI specification. gNMI Get and Set are supported both with OpenConfig and with device YANG that is native to Cisco NX-OS. The gNMI Capabilities method is used to retrieve capabilities from the switch.

gNMI Subscribe offers a dial-in subscription to telemetry data on the switch. This allows a telemetry application to pull information from a switch with a Subscribe operation. The initial implementation of gNMI Subscribe in Release 9.3(1) was based on the Cisco Data Management Engine (DME) and device YANG that is specific to Cisco Nexus switches. In order to have a fully open gNMI specification, OpenConfig support was added with gNMI Subscribe in Release 9.3(5).

Cisco NX-OS Release 9.3(5) supports the complete suite of gNMI operations using OpenConfig. Cisco NX-OS Release 9.3(5) is based on gNMI version 0.5.0.

gRPC operations in gNMI

In the context of NETCONF, RESTCONF, and gNMI, the controller or network management system is referred to as the client, and the network element (the switch) is referred to as the server.

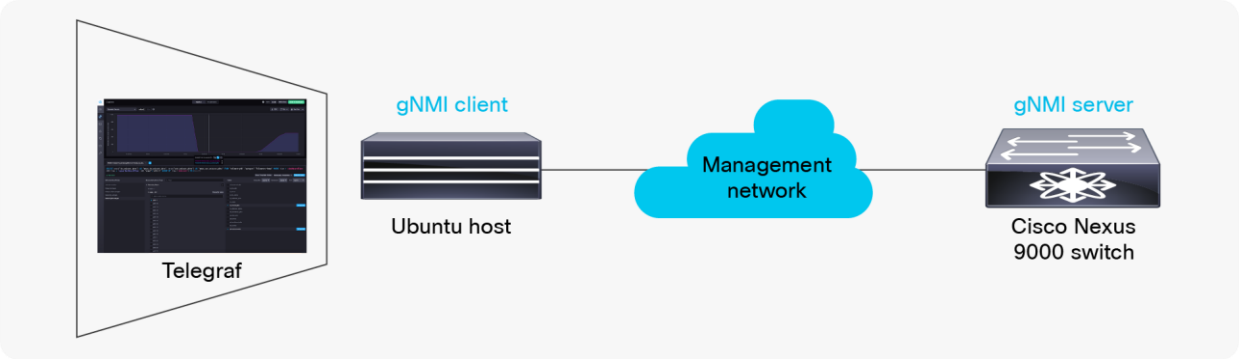

Figure 1.
gNMI server and client

gNMI CapabilitiesRequest

This operation gives a gNMI client information from the switches about the gNMI versions it uses and the data models and encodings it supports. It is up to the server (the switch in this case) to decide which YANG models and encodings it supports, and which protocol version to use.

gNMI GetRequest

When retrieving information from a gNMI server, there are two options: Get and Subscribe. The **GetRequest** is most often used when retrieving a small amount of data. The data it returns (**GetResponse**) is usually in the form of a snapshot. The **SubscribeRequest** is used for retrieving larger amounts of data.

gNMI SetRequest

The gNMI **SetRequest** is used to update, replace, and delete configurations. It consists of an ordered set of edit operations. This procedure is atomic, which means that if any of the steps fail in operation or validation, none of the operations get applied. The SetResponse typically returns a list of responses, one per operation requested.

gNMI SubscribeRequest

The **SubscribeRequest** is used for applications such as telemetry or for larger queries of data. In the case of telemetry with gNMI, the subscriber (client) specifies a frequency of delivery which could be:

ONCE: The data is returned immediately and only once for all specified paths.

POLL: The data is returned from the device when polled with the current values for all specified paths.

STREAM: The data is returned continuously. This could either be in the **SAMPLE** mode (where data is returned periodically in accordance with a sample interval) or in the **ON_CHANGE** mode where the data is returned if there is a change in values.

The details of all the requests and responses can be found in the gNMI specification.

In some instances, there can be two gNMI sessions established between client and server, one for streaming telemetry via gNMI Subscribe and the other for network configuration using gNMI Get and Set methods.

Switch prerequisites for gNMI

Cisco Nexus 9000 Series Switches support gNMI with OpenConfig beginning with Cisco NX-OS Release 9.3(5). For demo and simulation purposes, a Cisco Nexus 9300v or Cisco Nexus 9500v may also be used in place of a physical switch. In both cases, once the switch or switches are running the required software version, the appropriate RPM packages for OpenConfig for that version need to be installed on the switch. This can be downloaded from the Cisco Artifactory portal, under "open-nxos-agents."

The example configuration below illustrates this. The file "mtx-openconfig-all-<version>.lib32_n9000.rpm" is copied onto the switch bootflash, and installed on the switch as below:

```
n9300v-telemetry# install add mtx-openconfig-all-1.0.0.182-9.3.5.lib32_n9000.rpm activate
Adding the patch (/mtx-openconfig-all-1.0.0.182-9.3.5.lib32_n9000.rpm)
[####################] 100%
Install operation 1 completed successfully at Fri Jul  3 02:20:55 2020

Activating the patch (/mtx-openconfig-all-1.0.0.182-9.3.5.lib32_n9000.rpm)
[####################] 100%
Install operation 2 completed successfully at Fri Jul  3 02:21:03 2020

n9300v-telemetry# show version
<---snip--->
Active Package(s):
 mtx-openconfig-all-1.0.0.182-9.3.5.lib32_n9000
n9300v-telemetry#
```

Switch configuration for gNMI

gNMI can be enabled by the command “feature grpc.” The other gRPC configuration is summarized below:

```
n9300v-telemetry# show run grpc

!Command: show running-config grpc
!No configuration change since last restart
!Time: Tue Jul 14 16:56:37 2020

version 9.3(5) Bios:version
feature grpc

grpc gnmi max-concurrent-calls 16
grpc use-vrf default
grpc certificate gnmicert

n9300v-telemetry#
```

The max-concurrent-calls argument applies specifically to the new gNMI service and allows a maximum of 16 concurrent gNMI calls. The gRPC agent serves only the management interface by default. Adding the “use-vrf default” command allows it to accept requests from both the management and the default VRF.

Optionally, we can also configure gNMI to use a specific port for streaming telemetry. The default port is 50051.

```
n9300v-telemetry(config)# grpc port ?
    Default 50051
```

Certificates in gNMI

Telemetry and automation with gNMI use TLS certificates to validate the client-server communication. In the example configuration below, a self-signed certificate is generated; however, other forms of certificates may be used. The certificate is uploaded onto the gNMI client and the switch. The gNMI/gRPC agent on the switch is then set to honor the certificate. In the case where the gNMI client is Telegraf, a Telegraf configuration file is set to point to the certificate.

For the switch side of the configuration, the configuration guide covers the required steps. There are two methods that can be followed. The first method is available in older releases and consists of copying the .pem file onto bootflash and manually editing the gRPC configuration file to use the .pem and .key files.

The second method was introduced with Cisco NX-OS Release 9.3(3) and is the recommended way of installing certificates. It consists of generating a public and private key pair and embedding them in a certificate that is associated with a trustpoint. The trustpoint is then referenced in the grpc certificate command above.

```
n9300v-telemetry# run bash sudo su
bash-4.3# cd /bootflash/
bash-4.3# openssl req -newkey rsa:2048 -nodes -keyout gnmi.key -x509 -days 1000 -out
gnmi.pem
```

```
bash-4.3# openssl pkcs12 -export -out gnmi.pfx -inkey gnmi.key -in gnmi.pem -certfile
gnmi.pem -password pass:abcxyz12345
bash-4.3# exit
n9300v-telemetry(config)# crypto ca trustpoint gnmicert
n9300v-telemetry(config-trustpoint)# crypto ca import gnmicert pkcs12 gnmi.pfx abcxyz12345
n9300v-telemetry(config)# grpc certificate gnmicert
```

The certificate can be verified using the command “show crypto ca certificates.” The public key in the example above (gnmi.pem) needs to be copied from the switch bootflash to the gNMI client (which could be a gNMI tool or telemetry collector such as Telegraf).

Telemetry using gRPC/gNMI

The two methods of streaming telemetry described earlier can be implemented by enabling specific features globally on Cisco Nexus switches.

- Dial-out telemetry is enabled with “feature telemetry”.
- Dial-in telemetry with gNMI is enabled with “feature grpc”.

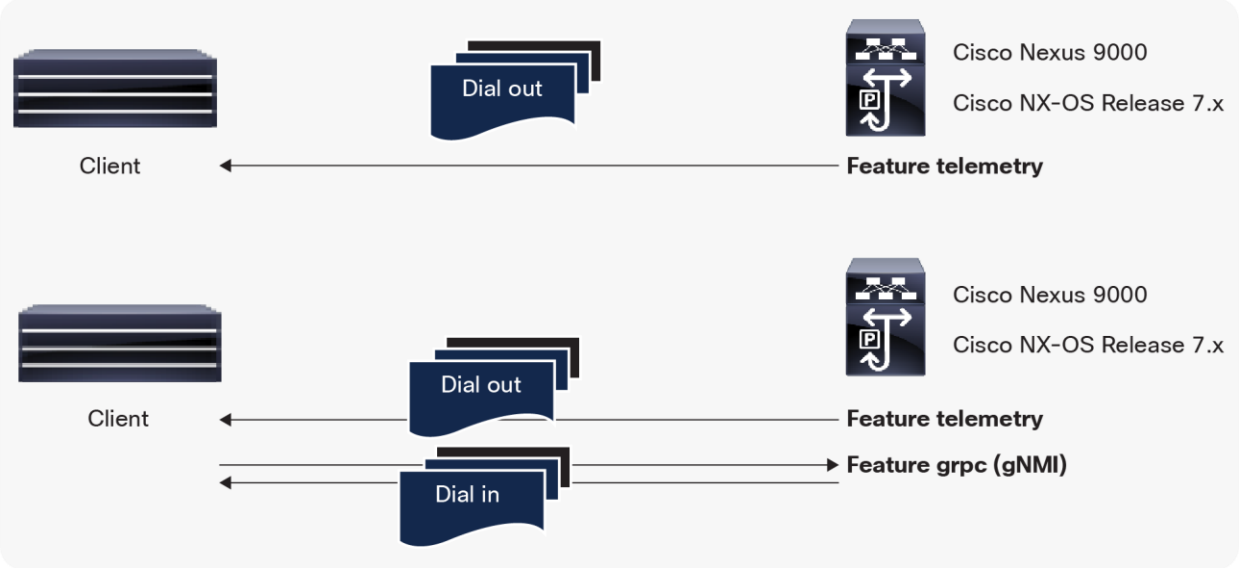

Figure 2.
Telemetry support on Cisco Nexus switches

gNMI Subscribe

The following section will explore dial-in telemetry with gNMI Subscribe using the open-source collector Telegraf.

Telemetry collector for gNMI: Telegraf

Telegraf is an open-source server agent used for collecting and reporting metrics and events. With the appropriate input plug-ins in place, Telegraf is able to subscribe to a switch or switches and collect telemetry data over gNMI or other protocols. It can send this data to a time series database called InfluxDB. The data can then be rendered with an application called Chronograf. The different components are summarized below:

- **Telegraf:** a server agent for collecting and reporting metrics
- **InfluxDB:** a time-series database
- **Chronograf:** a GUI (graphical user interface) for the InfluxData platform, which works on templates and libraries
- **Kapacitor:** a data-processing engine

Telegraf uses input and output plug-ins. The output plug-ins are a method for sending data to InfluxDB. The input plug-ins are used to specify different sources of telemetry data that Telegraf can subscribe to receive data from, including Cisco Nexus switches.

A partial configuration for the output plug-in is shown below. It includes the host IP address that Telegraf is installed on, and a database name and credentials for InfluxDB. This information will be fed into Chronograf.

```
# Configuration for sending metrics to InfluxDB
[[outputs.influxdb]]
   urls = ["http://172.25.75.91:8086"]
   database = "telemetrydb"
   username = "telemetry"
   password = "metrics"
```

There are specific plug-ins for Telegraf to work with Cisco Nexus 9000 switches. The two plug-ins **cisco_mdt_telemetry** (the Cisco model-driven telemetry plug-in) and gnmi (the Cisco gNMI plug-in) are integrated into the Telegraf release; no specific configuration is required to install them. The **cisco_mdt_telemetry** plug-in is based on dial-out telemetry or a push model. The **gnmi** plug-in is based on dial-in telemetry or a pull model.

Here is the configuration for the **gnmi** input plug-in, which includes the switch parameters. Note that the path specifies an origin of “openconfig”. The other options are to use device YANG or DME as the origin and path for the gNMI subscription. The encoding can also be specified here. The Cisco Nexus Programmability Guide contains more information about supported encoding formats with gNMI.

```
[[inputs.gnmi]]
  ## Address and port of the GNMI GRPC server
  addresses = ["172.25.75.81:50051"]
  ## define credentials
  username = "admin"
  password = "abcxyz12345"
  ## GNMI encoding requested (one of: "proto", "json", "json_ietf")
   encoding = "proto"
  ## enable client-side TLS and define CA to authenticate the device
   enable_tls = true
   tls_ca = "/etc/telegraf/gnmi.pem"
   insecure_skip_verify = true

[[inputs.gnmi.subscription]]
 ## Name of the measurement that will be emitted
 name = "Telemetry-Demo"
 ## Origin and path of the subscription
    origin = "openconfig"
    path = "/interfaces/interface/state/counters"
```

Chronograf can be accessed using the host IP address and port 8888. It will need to be set to point to the InfluxDB database that was specified in the output plug-in section of the Telegraf configuration file.

Verification of gNMI on the switch

Verify gNMI/gRPC on the switch as below to check the configured gNMI status with certificate registration and to verify that the gNMI subscription was successful.

```
n9300v-telemetry# show grpc gnmi service statistics

=============
gRPC Endpoint
=============

Vrf            : management
Server address : [::]:50051

Cert notBefore : Jul 10 19:56:47 2020 GMT
Cert notAfter  : Jul 10 19:56:47 2021 GMT

Max concurrent calls           :  16
Listen calls                   :  1
Active calls                   :  0

Number of created calls        :  4
```

```
Number of bad calls             :  0

Subscription stream/once/poll   :  3/0/0

Max gNMI::Get concurrent        :  5
Max grpc message size           :  8388608
gNMI Synchronous calls          :  0
gNMI Synchronous errors         :  0
gNMI Adapter errors             :  0
gNMI Dtx errors                 :  0
<---snip--->
n9300v-telemetry#
```

Note the output above showing the gRPC port number and VRF in use. It also shows that the certificate is installed successfully with the dates of the Cert being indicated.

```
n9300v-telemetry# show grpc internal gnmi subscription statistics  | b YANG
1               YANG                  36075             0                 0

2               DME                   0                 0                 0

3               NX-API                0                 0                 0
<---snip--->
n9300v-telemetry#
```

The above output shows increments on the statistics for YANG every time we have a successful gNMI subscription, since gNMI with OpenConfig uses the underlying YANG model.

Visualization of telemetry data with gNMI Subscription

The time-series telemetry data can be viewed using Chronograf, as shown in Figures 3 and 4. The graphical view on Chronograf can be obtained by navigating to the measurement specified in the Telegraf configuration file.

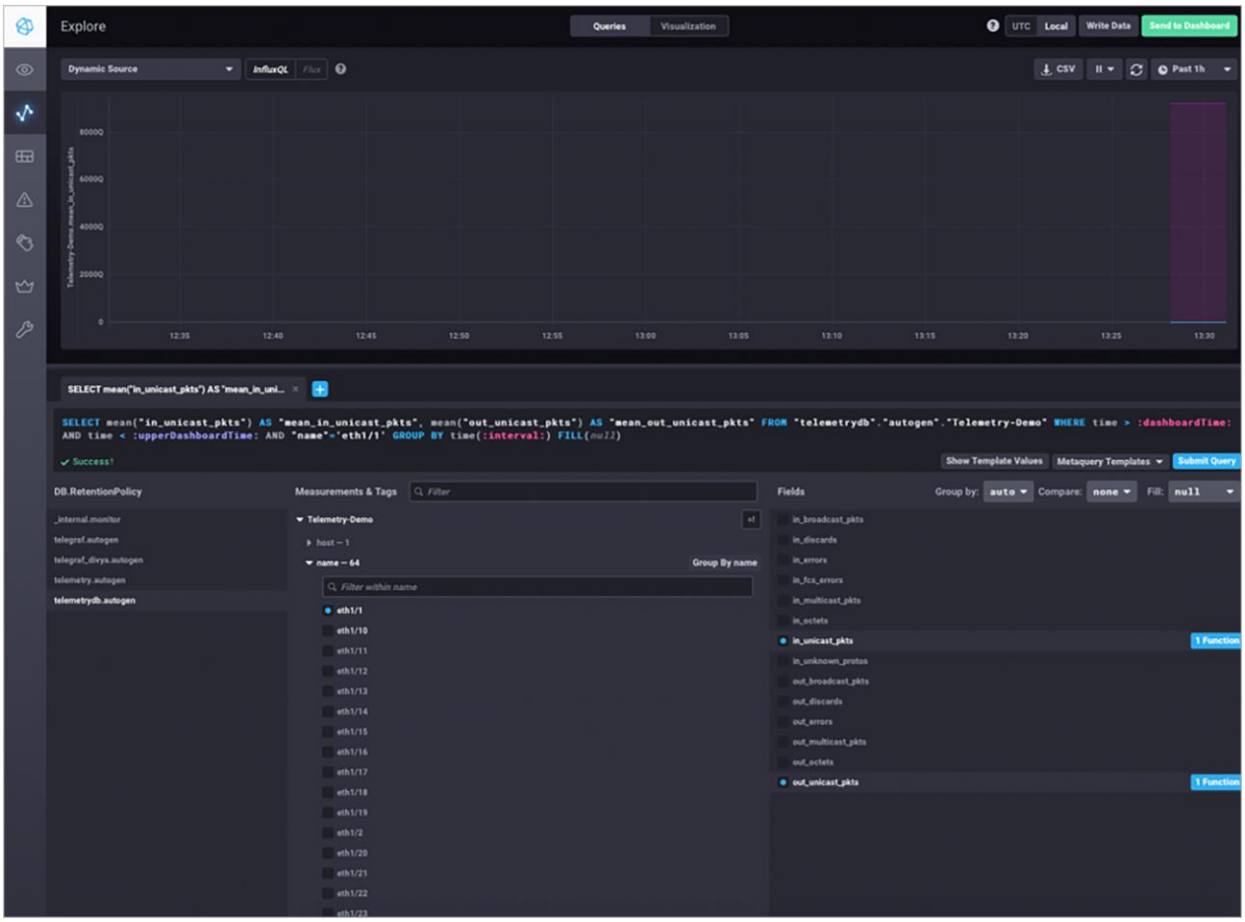

Figure 3.
Chronograf – setting up the queries and parameters to monitor

Figure 4.
Chronograf – interface counter graphs and packet counts on time-series view

The graphical view above shows interface statistics collected by gNMI for unicast packets in and out of a particular interface that has traffic going on it. The data can be viewed and further modeled using queries to make it more granular and specific.

Network automation with gNMI

gNMI Get and Set

The configuration and retrieval of data on Cisco Nexus 9000 switches using gNMI and gRPC can be summarized in Figure 5. With NX-OS Release 9.3(5), the different gRPC operations are supported for configuration management. Note that support for the proprietary gRPC agent on Cisco Nexus 9000 switches will be deprecated, and ongoing support will be present for the open gNMI specification introduced in Release 9.x.

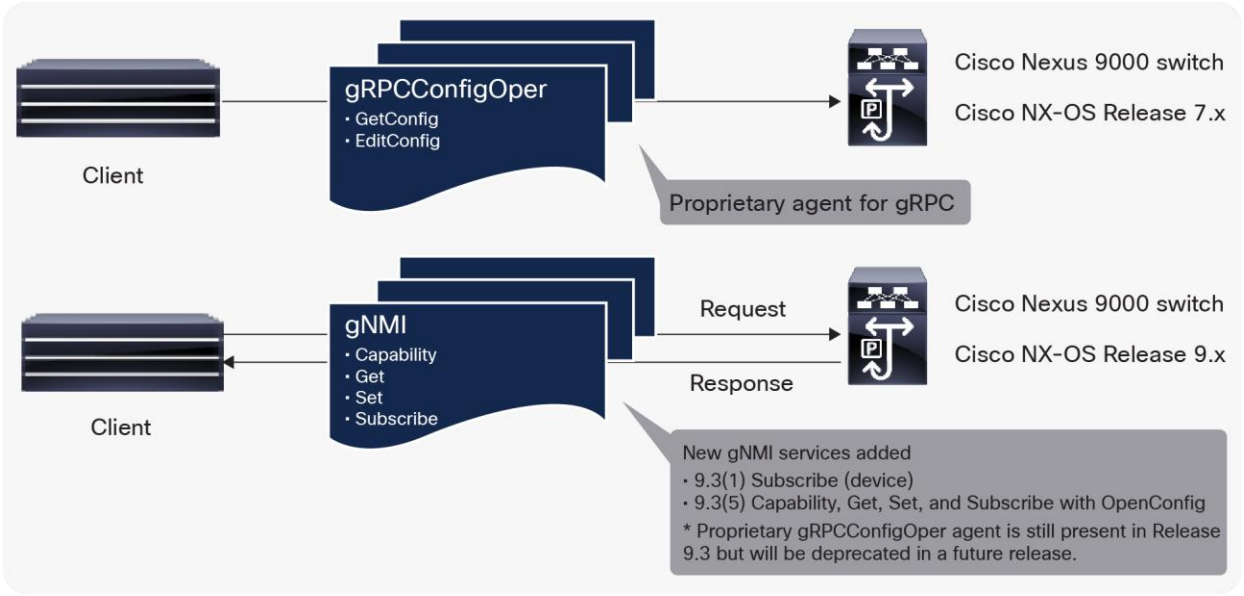

Figure 5.
gRPC and gNMI support on Cisco Nexus 9000 switches

Guidelines and limitations for gNMI Get/Set

The following applies to gNMI Get and Set operations on Cisco Nexus 9000 switches.

- The gNMI Get and Set methods support only JSON encoding.
- The supported data types are CONFIG, STATE, and ALL.
- A single request cannot have both OpenConfig YANG and device YANG model paths.
- A single GetRequest can have up to 10 paths. A single SetRequest can have up to 20 paths.

Verification of gNMI Operations using cisco-gnmi

In order to illustrate network automation with gNMI Get/Set, we use a tool called cisco-gnmi as an example. This tool includes a library " cisco-gnmi-python," which wraps the gNMI implementation to facilitate ease of use of Python programs with different Cisco implementations (Cisco IOS® XE, Cisco IOS XR, and Cisco NX-OS) of gNMI. It also includes a CLI form of the tool, which can be used to implement gNMI functionality without the use of a Python script.

gNMI Capabilities

The output below is an example of how we can retrieve gNMI **Capabilities** using cisco-gnmi. This gives us information from the switch about the gNMI versions it uses and the data models and encodings it supports. The -ssl_target_override parameter overrides the hostname of the host. The credentials to access the switch are specified along with the certificate and the gRPC port number that is configured on the switch.

```
divya@Ubuntu-host:~$ cisco-gnmi capabilities -os NX-OS -root_certificates ./gnmi.pem -
ssl_target_override divya 172.25.75.81:50051
Username: admin
Password:
INFO:root:supported_models {
  name: "Cisco-NX-OS-device"
  organization: "Cisco Systems, Inc."
  version: "2020-07-20"
}
supported_models {
  name: "openconfig-acl"
  organization: "OpenConfig working group"
  version: "1.1.1"
}
supported_models {
  name: "openconfig-bfd"
  organization: "OpenConfig working group"
  version: "0.2.0"
}
←--- snip --->
supported_models {
  name: "openconfig-telemetry"
  organization: "OpenConfig working group"
```

```
  version: "0.5.1"
}
supported_models {
  name: "openconfig-vlan"
  organization: "OpenConfig working group"
  version: "3.0.2"
}
supported_models {
  name: "DME"
  organization: "Cisco Systems, Inc."
}
supported_models {
  name: "Cisco-NX-OS-Syslog-oper"
  organization: "Cisco Systems, Inc."
  version: "2019-08-15"
}
supported_encodings: JSON
supported_encodings: PROTO
gNMI_version: "0.5.0"

divya@Ubuntu-host:~$
```

A snapshot of the OpenConfig model for interfaces can be seen in Figure 6.

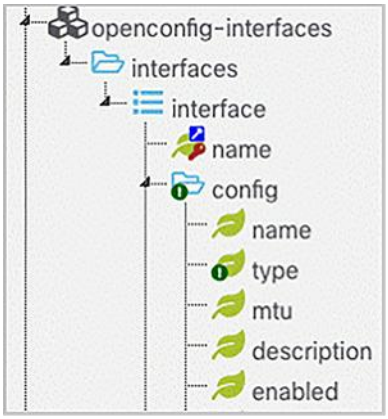

Figure 6.
OpenConfig model for interfaces

gNMI Get

The following CLI can be used to retrieve data from the switch using gNMI **Get**, for example. It specifies the path, type, and encoding for which data is requested. As seen in the xpath definition, OpenConfig is the underlying data model used to retrieve the state of an interface (the tool also supports device YANG as the data model that is specific to Cisco NX-OS). The type of information being retrieved – for example, the data type – could be **config**, **state**, or **all** in NX-OS. In the example below, JSON is specified as the encoding since it is what is currently supported on NX-OS.

```
divya@Ubuntu-host:~$ cisco-gnmi get -encoding JSON -data_type ALL -os NX-OS -
root_certificates ./gnmi.pem -ssl_target_override divya -xpath
"/interfaces/interface[name='mgmt0']" 172.25.75.81:50051
Username: admin
Password:
INFO:root:notification {
  timestamp: 1596066432721403475
  update {
    path {
      origin: "openconfig"
      elem {
        name: "interfaces"
      }
      elem {
        name: "interface"
        key {
          key: "name"
          value: "mgmt0"
        }
      }
    }
    val {
      json_val:
"[{\"name\":\"mgmt0\",\"config\":{\"enabled\":true,\"mtu\":1500,\"name\":\"mgmt0\",\"type\":
\"ethernetCsmacd\"},\"state\":{\"admin-status\":\"UP\",\"last-
change\":\"1595937502596000000\",\"oper-
status\":\"UP\",\"enabled\":true,\"mtu\":1500,\"name\":\"mgmt0\",\"type\":\"ethernetCsmacd\"
},\"subinterfaces\":{\"subinterface\":[{\"index\":0,\"config\":{\"index\":0},\"ipv4\":{\"add
resses\":{\"address\":[{\"ip\":\"172.25.75.81\",\"config\":{\"ip\":\"172.25.75.81\",\"prefix
-length\":23}}]},\"proxy-arp\":{\"config\":{\"mode\":\"DISABLE\"}}}}]}}]"
    }
  }
}
divya@Ubuntu-host:~$
```

gNMI Set

The example output below demonstrates a gNMI **Set** operation, which can be used to update, replace, or delete configurations on switches. The configuration to be applied is included in a JSON file called int_trunk. The **SetRequest** operation typically includes a path similar to the previous example used for gNMI Get. It also includes a value that is the data to be applied on the switch; in this case, the file int-trunk.json, which is included below:

```
{
"openconfig-interfaces:interfaces": {
"interface" :
[
{
"name": "eth1/44",
"ethernet": {
"switched-vlan": {
"config": {
"access-vlan": 1,
"interface-mode": "TRUNK",
"native-vlan": 1,
"trunk-vlans": [
"1..4094"
]
}
}
}
}
]
}
}
```

```
divya@Ubuntu-host:~$ cisco-gnmi set 172.25.75.81:50051 -os NX-OS -root_certificates
./gnmi.pem -ssl_target_override divya -update_json_config ./int-trunk.json
Username: admin
Password:
INFO:root:response {
  path {
    origin: "openconfig"
    elem {
      name: "openconfig-interfaces:interfaces"
    }
  }
  op: UPDATE
}
```

```
timestamp: 1596066885213855739


divya@Ubuntu-host:~$
```

gNMI Subscribe

Similarly, the cisco-gnmi CLI can also be used to do the gNMI **SubscribeRequest** operation in various modes to stream telemetry data. The SubscribeRequest specifies what data is being requested, and the SubscribeResponse includes a notification message providing an update value for a subscribed data entity, and a “sync_response” indicates that all the data values corresponding to the specified paths have been transmitted at least once.

```
divya@Ubuntu-host:~$ cisco-gnmi subscribe 172.25.75.81:50051 -os NX-OS -root_certificates
./gnmi.pem -ssl_target_override divya -xpath "/interfaces/interface[name='eth1/49']/state" -
req_mode ONCE -mode SAMPLE -interval 10 -dump_json -encoding JSON
Username: admin
Password:
INFO:root:update {
  timestamp: 1598464330620267213
  prefix {
  }
  update {
    path {
      origin: "openconfig"
      elem {
        name: "interfaces"
      }
    }
    val {
      json_val: "{\"interface\":[{\"name\":\"eth1\\/49\",\"state\":{\"admin-
status\":\"UP\",\"ifindex\":436232192,\"last-change\":\"18446718873709551616\",\"oper-
status\":\"DOWN\",\"counters\":{\"in-broadcast-pkts\":\"0\",\"in-discards\":\"0\",\"in-
errors\":\"0\",\"in-fcs-errors\":\"0\",\"in-multicast-pkts\":\"0\",\"in-octets\":\"0\",\"in-
unicast-pkts\":\"0\",\"in-unknown-protos\":\"0\",\"out-broadcast-pkts\":\"0\",\"out-
discards\":\"0\",\"out-errors\":\"0\",\"out-multicast-pkts\":\"0\",\"out-
octets\":\"0\",\"out-unicast-
pkts\":\"0\"},\"description\":\"\",\"enabled\":false,\"mtu\":1500,\"name\":\"eth1\\/49\",\"t
ype\":\"ethernetCsmacd\",\"tpid\":\"TPID_0X8100\"}}]}"
    }
  }
}
INFO:root:sync_response: true
divya@Ubuntu-host:~$
```

Automation using gNMI and Python

The above CLI examples can be used to verify that the switch configuration, OpenConfig RPM packages, and certificates are in place. They also verify the successful execution of gNMI **Capabilities**, **Get**, **Set**, and **Subscribe** operations.

In addition to the CLI methods, a Python script can be used to automate configurations. In this automation example, the IEEE protocol LLDP is used to detect the presence of Linux neighbors on a switch, and then configure the corresponding ports with a predefined template. The script first applies a gNMI **Capabilities** method to check if the OpenConfig model for LLDP is supported on the switch. Next, it performs a gNMI **Get** to retrieve the state of LLDP neighbors on the switch. With this information, it is possible to extract the interfaces where a Linux host is detected and do a gNMI **Set** to set the required interfaces with "switchport mode trunk." The code defines a new class called ConfigFunctions(). Within that class, the following functions are defined: Init, lldp_capability, get_lldp_ifs, and set_trunk_host to perform the required operations. It also includes a helper function called get_gnmi_json_val to convert the data from protobuf to JSON and decode the base64 string to UTF-8, in order to be able to easily parse through it. This example is used to illustrate the use of gNMI for configuration management and automation and can be extrapolated to more complex deployment scenarios.

An example of executing a Python script to perform gNMI Capabilities, Get, and Set operations is given below.

```
divya@Ubuntu-host:~$ python3 lldp-gnmi-getpython.py
Overriding SSL option from certificate could increase MITM susceptibility!
openconfig-lldp model supported on device
Setting up Interface: eth1/3
response {
path {
origin: "openconfig"
elem {
name: "openconfig-interfaces:interfaces"
}
}
op: UPDATE
}
timestamp: 1597283880756493972
divya@Ubuntu-host:~$
```

Within this script, the xpath, which is used to specify the OpenConfig model, is invoked in the set_trunk_host function.

```
xpath = "openconfig-
lldp:lldp/interfaces/interface[name='"+interface+"']/neighbors/neighbor/state/system-
description"
```

It should be noted along with this example, that there is no support currently to convert an interface from Layer 3 to Layer 2 mode and vice versa using OpenConfig. However, this can still be done with gNMI **Set** using device YANG models.

Conclusions

Cisco Nexus 9000 Series NX-OS Release 9.3(5) supports the different operations with gNMI and gRPC, including Capabilities, Get, Set, and Subscribe. Adding complete support of gNMI with OpenConfig in NX-OS paves the way for future network needs in the areas of network automation and telemetry.

For more information

- Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI [Addison-Wesley ISBN-13: 978-0135180396]
- OpenConfig gNMI specification on GitHub
- Telegraf overview and plug-ins

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-744191-00 09/20