cisco-gNMI-dynamic-subscription

June 19, 2021

0.0.1 Preparation of the environment

TASK 17 (OPTIONAL)

Enable logging at the debug level:

[1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
import logging
logger = logging.getLogger()
logger.setLevel(logging.DEBUG)
logging.debug("test")
```

```
DEBUG:root:test
```

TASK 18

Import the Cisco gNMI package and its client module:

[2]:
```
from cisco_gnmi import ClientBuilder
```

0.0.2 Connectivity to the device

TASK 19

[3]:
```
builder = ClientBuilder('10.58.50.234:57000')
# builder = ClientBuilder('198.18.134.72:57777')
builder.set_os('IOS XR')
builder.set_secure_from_target()
builder.set_ssl_target_override()
builder.set_call_authentication('cisco', 'cisco123')
# builder.set_call_authentication('cisco', 'cisco')

client = builder.construct()
```

```
DEBUG:cisco_gnmi.builder:Using IOS XR wrapper.
```

[3]:
```
<cisco_gnmi.builder.ClientBuilder at 0x10d829b20>
```

[3]:
```
<cisco_gnmi.builder.ClientBuilder at 0x10d829b20>
```

[3]: <cisco_gnmi.builder.ClientBuilder at 0x10d829b20>

[3]: <cisco_gnmi.builder.ClientBuilder at 0x10d829b20>

```
DEBUG:cisco_gnmi.builder:Using secure channel.
DEBUG:cisco_gnmi.builder:Using username/password call authentication.
DEBUG:cisco_gnmi.builder:Using SSL/metadata authentication composite
credentials.
DEBUG:cisco_gnmi.util:Using ems.cisco.com as certificate CN.
WARNING:cisco_gnmi.builder:Overriding SSL option from certificate could increase
MITM susceptibility!
```

0.0.3 Configuration of model-driven telemetry streaming (dynamic, dial-in)

Configure the IOS XR device for model-driven telemetry streaming using the **gNMI dial-in** mechanism. In this case, the MDT receiver (i.e., the host that runs the Jupyter notebook - `198.18.134.50`) will contact the device (**dial-in**) and exchange SYN – SYN-ACK – ACK with the device while establishing the connection.

If the connection is successfully created, the device will start streaming telemetry data towards the collection point.

The collection will stop when the receiver cancels the subscription (e.g., *Interrupt the kernel* of the Jupyter notebook) or when the session terminates (e.g., when polling once).

Here, we subscribe to updates of the operational state (`Cisco-IOS-XR-infra-statsd-oper`) of the management interface (`[interface-name="MgmtEth0/RP0/CPU0/0"]`). Specifically, we are looking for generic counters regarding this interface : - (multicast / broadcast) packets received / sent - bytes received / sent - output / input drops - output / input queue drops - CRC errors - …

Notice that the subscription method (`subscribe_xpaths()`) has a few parameters:

- `xpath_subscriptions`: the xpath(s) to use for subscription
- `request_mode`: whether to stream continuously, once, or to poll data. Can be one of: STREAM, ONCE, POLL
- `sub_mode`: whether to sample regularly the data, or to stream the data only when a change is happening. Can be one of: TARGET_DEFINED, ON_CHANGE, SAMPLE
- `encoding`: the encoding format of the returned data. Can be one of: JSON, BYTES, PROTO, ASCII, JSON_IETF
- `sample_interval`: the time interval at which samples of data should be streamed [ns]
- `suppress_redundant`: whether to avoid sending duplicate data
- `heartbeat_interval`: the enforced time interval after which data should be sent when `suppress_redundant` is in use [ns]

In this example, we expect to receive data regularly, every 60 seconds.

TASK 20

Create the subscription:

[ ]:
```
# Subscribe
```

```
subscribe_reply = client.
 ↪subscribe_xpaths(xpath_subscriptions='Cisco-IOS-XR-infra-statsd-oper:
 ↪infra-statistics/interfaces/interface[interface-name="MgmtEth0/RP0/CPU0/0"]/
 ↪generic-counters',
                                          request_mode='STREAM',
                                          sub_mode='SAMPLE',
                                          encoding='PROTO',
                                          sample_interval=60000000000, # Every␣
 ↪60 seconds
                                          suppress_redundant=False,
                                          heartbeat_interval=None
)

# Print the MDT data
for m in subscribe_reply:
    print(m)
```

TASK 21

Observe the incoming data, including the timestamps.

TASK 22

Go back to the `cisco-gNMI-main` notebook and confirm that the model-driven telemetry streaming is active on the device.