gNMI - Management Interface

About gNMI

gNMI (gRPC Network Management Interface) is configured on gRPC. Configuring gNMI helps to edit, read the configuration, and operational state of a network device. It also enables the network devices to generate telemetry streams to a designated data collection system.

Note You can configure gNMI as a child service feature of gRPC feature only.

Cisco NX-OS supports all gNMI RPC as mentioned in the following table:

Table 1: Supported gNMI RPCs

gNMI RPC	Supported
Capabilities	Yes
Get	Yes
Set	Yes
Subscribe	Yes

To subscribe RPC, Cisco NX-OS supports the below submodes:

Table 2: Subscribe Options

Type	Sub Type	Supported	Description
Once		Yes	Switch sends current values only once for all specified paths.
Poll		Yes	Whenever the switch receives a Poll message, the switch sends the current values for all specified paths.
Stream	Sample	Yes	Once per stream sample interval, the switch sends the current values for all specified paths. The supported sample interval range is from 1 through 604,800 second. The default sample interval is 10 seconds.
	On_Change	Yes	The switch sends current values as its initial state, but updates values only when there are changes such as create, modify, or delete for the specified paths.

gNMI subscription is commonly referred to as dial-in telemetry. As gNMI requires an external customer to initiate the request in the network device.

Cisco NX-OS supports a separate telemetry feature in which the networking device pushes the telemetry data out of external receivers, it is referred to as dial-out telemetry. For more information, see the Telemetry section.

Beginning with Cisco NX-OS Release 10.4(3)F, OpenConfig path openconfig-system:/system/processes supports On-change gnmi subscriptions. This path supports both gnmi-proto and gnmi-json encodings. The subscription will remain active until the client unsubscribes the same path.

It is recommended to make the most strict possible subscription for better efficiency. For example, to monitor cpu-usage-user, you can subscribe the data directly rather than using On-target gnmi subscription for general path like /system/processes, to avoid event data.

Guidelines and Limitations for gNMI

The following are the guidelines and limitations for gNMI:

- When you subscribe an OpenConfig routing policy with a pre-existing CLI configuration as shown below, it returns an empty value due to current implementation of the OpenConfig model.

```
ip prefix-list bgp_v4_drop seq 5 deny 125.2.0.0/16 le 32
 ipv6 prefix-list bgp_v6_drop seq 5 deny cafe:125:2::/48 le 128
```

  use below path:

```
openconfig-routing-policy:/routing-policy/defined-sets/prefix-sets/prefix-set[name=bgp_v4_drop]/config
openconfig-routing-policy:/routing-policy/defined-sets/prefix-sets/prefix-set[name=bgp_v6_drop]/config
```

- For gNMI subscriptions, use of origin, use_models or both these commands are optional.
- Before Cisco NX-OS 9.3(x) Release, information about supported platforms, see *Platform Support for Programmability Features* in the guide for that release. Starting with Cisco NX-OS release 9.3(x), for information about supported platforms, see Nexus Switch Platform Support Matrix.
- The feature supports JSON and gNMI proto encoding.
- The feature does not support protobuf any encoding.
- The feature does not support a path prefix in the subscription request, but the Subscription contains an empty prefix field.

Wildcard Path

- Multilevel wildcard "..." in path is not allowed.
- wildcard '*' in the top of the path is not allowed
- wildcard '*' in the key name is not allowed
- wildcard and value are not compatible in keys

The **show grpc gnmi** command has the following guidelines and limitations:

- This command does not support xml or json output format.
- The gRPC agent retains gNMI call data for maximum of an hour after the call ends.
- If the total number of calls exceeds 2000, the gRPC agent purges ended calls based on the internal cleanup routine.

Table 3: Wildcard Support for gNMI Requests

Type of Request	Wildcard Support
gNMI GET	Yes
gNMI SET	No
gNMI SUBSCRIBE, ONCE	Yes
gNMI SUBSCRIBE, POLL	Yes
gNMI SUBSCRIBE, STREAM, SAMPLE	Yes
gNMI SUBSCRIBE, STREAM, TARGET_DEFINED	Yes
gNMI SUBSCRIBE, STREAM, ON_CHANGE	No

Scale Considerations

- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the maximum size, the collected data drop. This is applicable for gNMI ON_CHANGE mode only.

  You can create focused subscriptions that manage smaller, granular data collection sets to avoid collection data drop. It is recommended to create multiple subscriptions for different, lower-level parts of the path instead of one higher-level path.

  Across all subscriptions, there is support of up to 250K aggregate MOs. Subscribing to more MOs can lead to collection data drops

- For all subscriptions, there is support of up to 250K aggregate MOs. If you subscribe to more MOs this impacts collection data drop.

Guidelines and Limitations for gNMI Subscription

- On-change subscriptions work for both gnmi and telemetry, but only one of these may be active at one time. If a subscription is made from one of these agents it will overwrite any existing subscription information.

  The following is an example for On-change gnmi subscription for openconfig-system:/system/processes:

```
Request:
./gnmi-console_enhanced_plus --host 172.22.244.142 --port 50051 -u admin -p insieme
--tls --cafile /tmp/grpc.pem --hostnameoverride ems.cisco.com  --operation=Subscribe
--submode ON_CHANGE  --xpath "openconfig-system:system/processes/process[pid=1]" -e
JSON

Response:

////// initial snapshot

Received response 1 -------------------------
/system/
{
    "processes": {
        "process": [
            {
                "pid": "1",
                "state": {
                    "pid": "1",
                    "name": "init",
                    "start-time": "1706643350384761800",
                    "cpu-usage-user": "14",
                    "cpu-usage-system": "12",
                    "cpu-utilization": 0,
                    "memory-usage": "180224",
                    "memory-utilization": 0
                }
            }
        ]
    }
}

///// first event update

/system/
{
    "processes": {
        "process": [
            {
```

```
                    "pid": "1",
                    "state": {
                        "start-time": "1706643350384761800",
                        "cpu-usage-user": "14",
                        "cpu-usage-system": "12"
                    }
                }
            ]
        }
    }
```


Configuring gNMI

Configure the gNMI feature through the gRPC gNMI commands.

Configuring gNMI Options

Before you begin

gNMI is a child service feature of gRPC feature only.

For more information, see the gPRC Agent documentation to enable the gRPC agent.

SUMMARY STEPS

1. switch# **configure terminal**
2. (Optional) **grpc gnmi max-cocrrent-call**
3. **grpc gnmi subscription target-defined min-interval**<*interval*>
4. **grpc gnmi subscription query-condition keep-data-timestamp**
5. (Optional) **grpc gnmi keepalive-timeout** <*timeout*>

DETAILED STEPS

	Command or Action	Purpose

Step 1	switch# **configure terminal**
	Example:
	`switch# config terminal`
	`switch(config)#`
		Enters the global configuration mode.

Step 2	(Optional) **grpc gnmi max-cocrrent-call** **Example:** `switch(config)# grpc gnmi max-concurrent-call 16`	Configuring this command sets the limit of simultaneous dial-in calls to the gNMI server on the switch. The default limit is 8. You can configure value with limit range of 1–16. The maximum value that you configure is on each VRF. If you set the limit value of 16 and if gNMI is configured on both management and default VRFs, each VRF supports 16 simultaneous gNMI calls. This command has no effect for ongoing or in-porgress gNMI calls. gRPC enforces a limit on new calls and the

	Command or Action	Purpose
		active calls have no impact and you can complete the active call.
Step 3	**grpc gnmi subscription target-defined min-interval**<*interval*> **Example:** `switch(config)# grpc gnmi subscription target-defined min-interval 60`	You can modify the default target-defined sample interval from 30 seconds to other required value.
Step 4	**grpc gnmi subscription query-condition keep-data-timestamp** **Example:** `switch(config)# grpc gnmi subscription query-condition keep-data-timestamp`	You can enable sample or once or poll subscriptions to get a timestamp from the database when the data is last updated. **Note** • This command is supported only for PROTO and this is not supported for JSON code. • It supports for once, poll and sample and not supported for not on_change subscriptions. • It supports for DME, YANG, and OpenConfig data sources. • For the properties which are not supported, the command defaults back to the collection time instead of the last database change time. • This command generates verbose responses for each timestamp. If you are not able to collect the messages on time, the switch drops the collection of messages.
Step 5	(Optional) **grpc gnmi keepalive-timeout** <*timeout*> **Example:** `switch(config)# grpc gnmi keepalive-timeout 1200`	Configuring this command, you can delete inactive or unauthorized connections. The gRPC agent periodically sends an empty response to the client. If this response fails to deliver to the client, then the connection stops. The default interval value is 600 seconds. You can configure to change the keepalive interval with the interval range of 600-86400 seconds.

gNMI RPCs

This section describes each gNMI RPC. You can view use of the reference client *gnmi_cli* in this section.

Capabilities

The capabilities of RPC returns to the list of capabilities of the gNMI service. The response message to an RPC request includes the gNMI service version, version data models, and data encode supported by the server.

Guidelines and Limitations for Capabilities

The following are the guidelines and limitations for capabilities:

Example Client Output

Below mentioned the example of client output for Capabilities. Considering the feature *openconfig* is enabled.

This examples shows the support to YANG model, gNMI version, and the supported encodes.

```
$ ./gnmi_cli -a 172.1.1.1:50051 -ca_crt ./grpc.pem -insecure -capabilities
supported_models: <
  name: "Cisco-NX-OS-device"
  organization: "Cisco Systems, Inc."
  version: "2019-11-13"
>
supported_models: <
  name: "openconfig-acl"
  organization: "OpenConfig working group"
  version: "1.0.0"
>
supported_models: <
  name: "openconfig-bgp-policy"
  organization: "OpenConfig working group"
  version: "4.0.1"
>
.
.
.
supported_models: <
  name: "openconfig-spanning-tree"
  organization: "OpenConfig working group"
  version: "0.2.0"
>
supported_models: <
  name: "openconfig-system"
  organization: "OpenConfig working group"
  version: "0.3.0"
>
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"
```

About Get

You can use `Get RPC` to retrieve a snapshot of the data tree from the switch. You can request multiple paths in a single request. According to gNMI Path conventions, you can use simple form of XPATH. See Schema path encoding conventions for gNMI. For more information about `GET` operation, see the *Retrieving Snapshots of State Information* section in the gRPC Network Management Interface.

Guidelines and Limitations for Get

The following are guidelines and limitations for Get:

- `GetRequest` encode supports only the JSON format.
- For the `GetRequest`.type, only DataType CONFIG and STATE have correlation and the expression in YANG format. The operation is not supported.
- You cannot have both OpenConfig (OC) YANG and device YANG paths in a single request. A single request has only one path.
- `GetRequest` for root path ("/": everything from **all** models) is not permitted.
- gNMI Get returns all default value. See Report-all mode RFC 6243 [4].
- `Get` does not support the `Cisco-NX-OS-syslog-oper` model.
- To retrieve `openconfig-procmon` data, you can send a query to the path `/system/processes` or `/system`.

Note Before Cisco NX-OS Release 10.3(x), you cannot retrieve data from the `/system` path.

- The following optionals are not supported:
  - Path prefix
  - Path alias
  - Wildcards in the path
- You can request 10 paths in a single `GetReqeuest`.
- If the return value of size in `GetResponse` is more than 12 MB, the system displays the error status `grpc::RESOURCE_EXHAUSTED`.
- The maximum receive buffer size of gRPC is 8 MB.
- If you perform an `Get` operation for a switch which has more configuration, the gRPC process consumes all available memory. If the memory exhausts, the below syslog is generated:

```
MTX-API: The memory usage is
        reaching the max memory resource limit (3072) MB
```

If the memory exhausts consecutively, the below syslog is generated:

```
The process has become unstable and
        the feature should be restarted.
```

Cisco recommends you to restart the gRPC feature for normal functioning of gNMI transactions.

- The maximum number of total concurrent sessions for `Get` is 75% of maximum configured concurrent calls. For an example, if the MTX concurrent calls are configured to 16, the maximum number of total concurrent sessions for `Get` is 12.
- The total number of concurrent sessions for `Get` and `Set` is configured max gNMI concurrent-1.

  For an example, if gNMI concurrent calls are configured to 16, the maximum number of total concurrent sessions for `Get` and `Set` is 15.

About Set

You can use `Set RPC` to change the configuration of the switch. You can delete, replace, and update the switch. All these operations in a single `Set` request that is considered as a transaction which is a successful operation or the switch is set to original state.

The `Set` operations are performed in the order that is specified in the `Set Request` configuration. If a path is requested multiple times, the changes are performed even if the paths overwrite each other path. The final state of the data is achieved with the final operation in the transaction. You can edit the paths that are specified in the `Set Request` such as delete, replace, update fields are configuration data paths.

For more information on `Set` operations, see *Modifying State* section of gNMI specification.

Guidelines and Limitations for Set

The following are guidelines and limitations for Set:

- `SetRequest` encode supports only JSON format.
- You cannot have both OpenConfig (OC) YANG and device YANG paths in a single request. A single request has only one path.
- The following optionals are not supported:
  - Path prefix
  - Path alias
  - Wildcards in the path
- You can request 20 paths in a single `SetReqeuest`.
- The maximum receive buffer size of gRPC is 8 MB.
- The maximum number of total concurrent sessions for `Get` and `Set` is configured maximum gNMI concurrent calls. For an example, if the gNMI concurrent calls are configured to 16, the maximum number of total concurrent sessions for `Get` and `Set` is 15.
- If you perform a Set::Delete RPC operation for a switch, if a configuration is bulky, an MTX warning message is generated as shown below:

```
Configuration size for this
        namespace exceeds operational limit. Feature may become unstable and require
        restart.
```

About Subscribe

You can use `Subscribe RPC` to register a telemetry stream for specific paths. When there is change in the registered paths, the switch pushes the notification to for the change in configuration or the path state.

You can use an optional flag available for `Subscribe RPC`. Beginning with Cisco NX-OS Release 9.3(1), the `UPDATES_ONLY` optional flag is supported and applicable only for `ON_CHANGE` subscriptions. If this optional flag is configured, the switch suppresses the current state and a notification is sent with the first response.

You can use these flags to modify the response to options listed in the following table.

Table 4: Support Metrics for `SUBSCRIBE` Flags

Subscription Type	hearbeat_interval	suppress_redundant
`ON_CHANGE`	Origin: Device YANG, OpenConfig YANG, DME	N/A
`SAMPLE`	Origin: Device YANG, OpenConfig YANG, DME	Origin: Device YANG, OpenConfig YANG

Beginning with Cisco NX-OS Release 10.2(3)F, the following optional flags are supported:

- `hearbeat_interval`
- `suppress_redundant`

You can use a `hearbeat_interval` flag to modify the behavior of `suppress_redundant` in a `sample` subscription. In this case, the target generates one telemetry update per `hearbeat_interval`, despite of `suppress_redundant` flag status is set to true. The value is specified as an unsigned 64-bit integer in nano seconds.

You can use the `suppress_redundant` flag for a `sample` subscription. In this case, the set status is true. The target generates a telemetry update message when the value of the path reported has changed the last update is generated. Updates are generated for the individual leaf nodes for which the subscription has changed.

For an example, subscriptions name 'A' and 'B' which has leaf nodes 'C' and 'D' branch from 'B' node. If the value of 'C' is changed, and not the value of 'D'. An update is generated only for 'C' and not for 'D'.

Guidelines and Limitations

The following are guidelines and limitations for Set subscription:

- The following flags are not supported:
  - Aliases
  - allow_aggregation
  - extensions
  - prefix
  - QoS

- Make sure that all paths within the same subscription request must have the same `sample` interval. If the same path requires different sample intervals, you must create multiple subscriptions.

gNMI Subscribe Options

The following table lists the subscribe option modes:

Table 5:

Type	Subscription Type	Description
`ONCE`		Subscribe to receive data when and close the session.
`POLL`		Subscribe to retain an active session. When you raise a poll request, ensure that you enter data for each request.
`STREAM`	`SAMPLE`	Subscribe to receive data at a specific cadence. The payload value is in nano seconds. 1 second =1000000000.
	`ON_CHANGE`	Subscribe to receive a snapshot and receive data only when there is change in tree.
`TARGET_DEFINED`		Subscribe to find the best subscription which can be created.

To Set Modes

Each mode requires 2 settings, such as inside subscription and outside subscription. The following mentioned the combination of subscriptions:

- ONCE: `SAMPLE`, `ONCE`
- POLL: `SAMPLE`, `POLL`
- STREAM: `SAMPLE`, `STREAM`
- ON_CHANGE: `ON_CHANGE`, `STREAM`
- TARGET_DEFINED: `TARGET_DEFINED`, `STREAM`

Origin

- `DME`: Subscribing to DME model
- `DEVICE`: Subscribing to YANG model
- `OPENCONFIG`: Subscribing to OpenConfig model

Name

- DME: Subscribing to DME model
- Cisco-NX-OS-device: Subscribing to YANG model

Encoding

- JSON: Stream is sent in JSON format.
- PROTO: Stream is sent in original proto format.

Configuring Example of Payload

The following section lists client examples for a payload:

Subscribe to DME Stream/Sample

```
{
    "SubscribeRequest":
    [
        {
            "subscribe":
            {
                "subscription":
                [
                    {
                        "path":
                        {
                            "origin": "DME",
                            "elem":
                            [
                                {
                                    "name": "sys"
                                },
                                {
                                    "name": "bgp"
                                 }
                             ]
                        },
                        "mode": "SAMPLE"
                    }
                ],
                "mode": "ONCE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "name": "DME",
                        "organization": "Cisco Systems, Inc.",
                        "version": "1.0.0"
                    }
                ],
                "encoding": "JSON"
            }
        }
    ]
}
```

Subscribe to OpenConfig YANG/Sample

```
{
    "SubscribeRequest":
    [
        {
            "subscribe":
            {
                "subscription":
                [
                    {
                        "path":
                        {
                            "origin": "openconfig",
                            "elem":
                            [
                                {
                                    "name": "interfaces"
                                }
                            ]
                        },
                        "mode": "SAMPLE",
                        "Sample_interval": 10000000000
                    }
                ],
                "mode": "ONCE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "name": "openconfig-interfaces",
                        "organization": "OpenConfig working group",
                        "version": "0.8.1"
                    }
                ],
                "encoding": "JSON"
            }
        }
    ]
}
```

About Subscribing Custom Syslog Stream

Cisco NX-OS supports the original and OpenConfig model. For gNMI or, subscribe `ON_CHANGE`, a custom YANG model is designed to stream the syslog events. You can configure this feature for the Cisco Nexus 9000 Series switches with minimum memory of 8 GB and above.

Guidelines and Limitations

The following are guidelines and limitations for Syslog Stream:

- An invalid syslog is not supported, such as a syslog with a filter or query condition is not supported.
- Syslog stream supports only below paths:
  - Cisco-NX-OS-Syslog-oper: syslog
  - Cisco-NX-OS-Syslog-oper: syslog messages
- Only stream sample and POLL modes are supported.

- Encoding formats that are supported are JSON and PROTO.

Original YANG Syslog Model

The following example shows the configuration of the YANG Syslog Model:

Note By default, the time-zone field is empty. The time zone field is set when you configure `clock format show-timezone syslog`.

```
PYANG Tree for Syslog Native Yang Model:
>>> pyang -f tree Cisco-NX-OS-infra-syslog-oper.yang module: Cisco-NX-OS-syslog-oper
+--ro syslog
+--ro messages
+--ro message* [message-id]
+--ro message-id int32
+--ro node-name? string
+--ro time-stamp? uint64
+--ro time-of-day? string
+--ro time-zone? string
+--ro category? string
+--ro group? string
+--ro message-name? string
+--ro severity? System-message-severity
+--ro text? string
```

Subscribe Request

The following is an example for the subscribe request:

```
{
    "SubscribeRequest":
    [
        {
            "subscribe":
            {
                "subscription":
                [
                    {
                        "path":
                        {
                            "origin": "syslog-oper",
                            "elem":
                              [
                                {
                                    "name": "syslog"
                                },
                                {
                                    "name":"messages"
                                }
                            ]
                        },
                        "mode": "ON_CHANGE"
                    }
                ],
                "mode": "ON_CHANGE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "name": "Cisco-NX-OS-Syslog-oper",
```

```
                        "organization": "Cisco Systems, Inc.",
                        "version": "0.0.0"
                    }
                ],
                "encoding":"JSON"
            }
        }
    ]
}
```

Example Response

The following is an output response example for PROTO encode:

```
[Subscribe]--------------------------------
Sat Aug 24 14:38:06 2019
### Generating request : 1 -----------
### Comment : STREAM request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE

subscribe {
subscription {
path {
origin: "syslog-oper"
elem {
name: "syslog"
}
elem {
name: "messages"
}
}
mode: ON_CHANGE
}

use_models {
name: "Cisco-NX-OS-Syslog-oper"
organization: "Cisco Systems, Inc."
version: "0.0.0"
}
encoding: PROTO
}

Thu Nov 21 14:26:41 2019
Received response 3 --------------------------
update {
timestamp: 1574375201665688000
prefix {
origin: "Syslog-oper"
elem {
name: "syslog"
}
elem {
name: "messages"
}
}
…
update {
path {
elem {
name: "time-stamp"
}
}
val {
uint_val: 1574375200000
```

```
}
}
update {
path {
elem {
name: "severity"
}
}
val {
uint_val: 5
}
}
}

/Received -------------------------------------
```

The following is an example of output response for JSON encode:

```
[Subscribe]------------------------------
### Reading from file ' testing_bl/stream_on_change/OC_SYSLOG.json '

Tue Nov 26 11:47:00 2019
### Generating request : 1 -----------
### Comment : STREAM request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE
subscribe {
subscription {
path {
origin: "syslog-oper"
elem {
name: "syslog"
}
elem {
name: "messages"
}
}
mode: ON_CHANGE
}
use_models {
name: "Cisco-NX-OS-Syslog-oper"
organization: "Cisco Systems, Inc."
version: "0.0.0"
}
}

Tue Nov 26 11:47:15 2019
Received response 5 --------------------------
update {
timestamp: 1574797636002053000
prefix {
}
update {
path {
origin: "Syslog-oper"
elem {
name: "syslog"
}
}
val {
json_val: "[ { \"messages\" : [[
{\"message-id\":657},{\"node-name\":\"task-n9k-1\",\"time-stamp\":\"1574797635000\",\"time-of-day\":\"Nov
 26 2019
11:47:15\",\"severity\":3,\"message-name\":\"HDR_L2LEN_ERR\",\"category\":\"ARP\",\"group\":\"ARP\",\"text\":\"arp
 [30318] Received packet with incorrect layer 2 address length (8 bytes), Normal pkt with
```

```
S/D MAC: 003a.7d21.d55e ffff.ffff.ffff eff_ifc mgmt0(9), log_ifc mgmt0(9), phy_ifc
mgmt0(9)\",\"time-zone\":\"\"} ]] } ]"
}
}
}
```

References

You can view available clients for gNMI subscriptions. For more information, see https://github.com/influxdata/telegraf/tree/master/plugins/inputs/cisco_telemetry_gnmi.

Troubleshooting gNMI

You can run show commands to view gNMI status. To verify specific gNMI configurations, enter the following commands that are listed in the below table:

Table 6: Show Commands

Command	Description
`show grpc gnmi service statistics`	Displays the summary of the agent running status, respectively for the management VRF, or the default configured VRF. It also displays: • Basic overall counters • Certificate expiration time
`show grpc gnmi rpc summary`	Displays the following: • Number of capability RPCs received. • Capability of RPC errors. • Number of Get RPCs received. • Get RPC errors. • Number of Set RPCs received. • Set RPC errors.

Command	Description
`show grpc gnmi transactions`	

Command	Description
	The show grpc gnmi transactions command is the densest and contains considerable information. It is history buffer of the most recent 50 gNMI transactions that are received by the switch. As new RPCs come in, the oldest history entry is removed from the end. The following explains what is displayed: This command displays the detailed information. It shows the history of the latest 50 gNMI transactions that are received by the switch. An entry of new RPCs, the history of the oldest entry of RPCs is removed. The following shows the information which is displayed: • RPC - This shows the type of RPC that was received (Get, Set, Capabilities) • DataType - For a Get only. It has the values ALL, CONFIG, and STATE. • Session - Displays the unique session-id that is assigned to this transaction. It can be used to correlate data that is found in other log files. • Time In - Displays the timestamp when the RPC was received by the gNMI handler. • Duration - Time delta in milliseconds from receiving the request to giving a response. • Status – Displays the status code of the operation returned to the client `(0 = Success, !0 == error)` The following shows the data per path within a single gNMI transaction. For a single `Get` or `Set`: • **Subtype** – For a `Set RPC`, it displays the specific operation that is requested per path (Delete, Update, Replace). For `Get`, there is no subtype. • **Dtx** – Displays that this path is processed in DTX fast path or not. A dash (-) indicates no, an asterisk (*) indicates yes. • **St** – Displays the status for this path. The different status that is displayed as shown below: • **OK**: The path is valid and processed by infra successfully. • **ERR**: The path is either invalid or generated error by infra. • **--**: The path is not processed yet, or not a

Command	Description
	valid and not sent to infra yet.

Configuration Examples for gRPC Commands

gRPC gNMI Service Statistics

The following shows the sample output for the command `show grpc gnmi service statistics`:

```
=============
gRPC Endpoint
=============

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

Cert notBefore : Mar 13 19:05:24 2020 GMT
Cert notAfter : Nov 20 19:05:24 2033 GMT

Max concurrent calls : 8
Listen calls : 1
Active calls : 0

Number of created calls : 1
Number of bad calls : 0

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

Max gNMI::Get concurrent : 5
Max grpc message size : 8388608
gNMI Synchronous calls : 74
gNMI Synchronous errors : 0
gNMI Adapter errors : 0
gNMI Dtx errors : 0
```

gRPC gNMI rPC Summary

The following shows the sample output for the command `show grpc gnmi service statistics`:

```
=============
gRPC Endpoint
=============

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

Cert notBefore : Mar 31 20:55:02 2020 GMT
Cert notAfter  : Apr  1 20:55:02 2020 GMT

Capability rpcs    : 1
Capability errors  : 0
Get rpcs           : 53
Get errors         : 19
Set rpcs           : 23
Set errors         : 8
Resource Exhausted : 0
Option Unsupported : 6
Invalid Argument   : 18
Operation Aborted  : 1
```

```
Internal Error     : 2
Unknown Error      : 0

RPC Type        State      Last Activity  Cnt Req    Cnt Resp   Client
--------------- ---------- -------------- ---------- ---------- -----------------
-----------------------
Subscribe       Listen     04/01 07:39:21          0          0
```

gRPC gNMI Transactions

The following shows the sample output for the command `show grpc gnmi transactions`:

```
=============
gRPC Endpoint
=============

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

Cert notBefore : Mar 31 20:55:02 2020 GMT
Cert notAfter  : Apr  1 20:55:02 2020 GMT

RPC          DataType   Session         Time In              Duration(ms) Status
------------ ---------- --------------- -------------------- ------------ ------
Set          -          2361443608      04/01 07:43:49       173          0
subtype: dtx:  st: path:
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo789]

Set          -          3445444384      04/01 07:43:33       3259         0
subtype: dtx:  st: path:
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo789]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo790]
…
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo807]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo808]

Set          -          2297474560      04/01 07:43:26       186          0
subtype: dtx:  st: path:
Update   -     OK  /System/ipv4-items/inst-items/dom-items/Dom-list[name=foo]/rt-
items/Route-list[prefix=0.0.0.0/0]/nh-items/Nexthop-list[nhAddr=192.168.1.1/32][n
hVrf=foo][nhIf=unspecified]/tag


Set          -          0               04/01 07:43:11       0            3
subtype: dtx:  st: path:
Update   -     --  /System/intf-items/lb-items/LbRtdIf-list[id=lo4]/descr
Update   -     ERR /system/processes

Set          -          2464255200      04/01 07:43:05       708          0
subtype: dtx:  st: path:
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo2]
Replace  -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr
Replace  -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo4]/descr
Replace  -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo5]/descr
Update   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr
Update   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo5]/descr


Set          -          3491213208      04/01 07:42:58       14           0
subtype: dtx:  st: path:
Replace  -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/descr

...

Get          ALL        2293232352      04/01 07:42:35       258          0
```

```
subtype: dtx:  st: path:
-        -     OK  /system

Get          ALL        0               04/01 07:42:33       0            12
subtype: dtx:  st: path:
-        -     --  /intf-items
```

Gathering Debug Logs

gNOI is a child service of the gRPC agent. For more information, see gRPC Agent chapter.

Accounting Log for gNMI

In gNMI, `Set RPC` changes the configuration on the switch. For the `SET` requests such as `UPDATE`, `REPLACE`, or `DELETE` configuration, gNMI generates the correspoding accouting logs. These logs include both original request and the changes made on the switch.

You can use `show accouting log` command to view the accouting logs.

The following example shows `SetRequest`, encoding = JSON to localhost with the following gNMI paths:

```
---
<<<<<<< set_delete >>>>>>>
[]
<<<<<<< set_replace >>>>>>>
[] []
<<<<<<< set_update >>>>>>>
[elem { name: "System"
} elem {
name: "tm-items"
} elem {
name: "certificate-items"
}
] [json_val: "{\"hostname\": \"test\", \"trustpoint\": \"foo\"}"
]
The SetRequest response is below ------------------------response { path {
elem {
name: "System"
} elem {
name: "tm-items"
} elem {
name: "certificate-items"
}
} op: UPDATE
} timestamp: 1656512303065384369
---
```

The below table shows the options can be configured on the switches. The accouting logs include the following items.

Item	Description
Context	Session ID and user
Operation	Commit or Abort
Database	Running or Candidate

Item	Description
ConfigMO	MO tree's text representation. Maximum up to 3000 characters.
Status	Success or Failed

The following shows the sample of accouting logs.

```
Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd=(COMMIT),database=[candidate],
configMo=[<topSystem childAction="" dn="sys" status="created,modified"><telemetryEntity
childAction="" rn="tm" status="created,modified"><telemetryCertificate childAction=""
hostname="test" rn="certificate" status="created,modified"
trustpoint="foo"/></telemetryEntity></topSystem>] (SUCCESS)
Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd=(COMMIT:CANDIDATE-TO-RUNNING),
database=[running] (SUCCESS)
```

The below table shows the original received request on the switch.

Item	Description
Context	Session ID and user
Operation	gNMI:SET:UPDATE, gNMI:SET:REPLACE, gNMI:SET:DELETE, COMMIT:CANDIDATE-TO-RUNNING
Source IP	gNMI Client IP
Path	gNMI path in the text format
Payload	Received JSON Request. Maximum up to 3000 characters.
Status	Success or Failed

```
Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd=(GNMI:SET:UPDATE),sourceIp=[192.168.1.2],
path=[/System/tm-items/certificate-items],payload=[{"hostname":"test","trustpoint":"foo"}]

(SUCCESS)
```

In case of failed request and based on failed scenario, you cannot view both the logs.

Invalid Requests

If you raise a request which is invalid, this request will be rejected without any confiugartion changes and only the initial request will be logged. The following shows the example of invalid request logs.

```
Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd=(GNMI:SET:UPDATE),
sourceIp=[192.168.1.2],path=[/System/tm-items/certificateitems],
payload=[{"hostname":"test","trustpoint":"foo"}] (FAILED)
```

Failed Requests

If you raise a request and if it fails due to various configuration restrictions, in such case both the original and failed configuration request is logged. The following example shows the logs.

```
Wed Jun 29 20:52:15
2022:type=update:id=1429663200:user=admin:cmd=(COMMIT),database=[candidate],
configMo=[<topSystem childAction="" dn="sys"
status="created,modified"><telemetryEntity childAction="" rn="tm"
status="created,modified"><telemetryCertificate childAction="" filename="foo"
hostname="test" rn="certificate" status="created,modified,replaced"
trustpoint="foo"/></telemetryEntity></topSystem>] (FAILED)

Wed Jun 29 20:52:15
2022:type=update:id=1429663200:user=admin:cmd=(GNMI:SET:REPLACE),
sourceIp=[192.168.1.2],path=[/System/tm-items/certificate-items],
payload=[{"hostname":"test","trustpoint":"foo","filename":"foo"}] (FAILED)
```

If you raise a request and if it fails to commit, in such case the original request is logged with the failed request. The following example shows the logs.

```
Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd
(COMMIT),database=[candidate],configMo=[<topSystem childAction="" dn="sys"
status="created,modified"><telemetryEntity childAction="" rn="tm"
status="created,modified"><telemetryCertificate childAction="" hostname="test"
rn="certificate" status="created,modified" trustpoint="foo"/></telemetryEntity></topSystem>]
 (SUCCESS)

Wed Jun 29 14:18:23
2022:type=update:id=1430425712:user=admin:cmd=(GNMI:SET:UPDATE),
sourceIp=[192.168.1.2],path=[/System/tm-items/certificate-items],
payload=[{"hostname":"test","trustpoint":"foo"}] (SUCCESS)

Wed Jun 29 20:34:06
2022:type=update:id=1429665744:user=admin:cmd=(COMMIT:CANDIDATE-TO-RUNNING),
database=[running] (FAILED)
```