

gNMI-gRPC Network Management Interface

- About gNMI, on page 1
- gNMI Subscribe RPC, on page 2
- Guidelines and Limitations for gNMI, on page 5
- Configuring gNMI, on page 7
- Configuring Server Certificate, on page 10
- Generating Key/Certificate Examples , on page 11
- Examples for Generating and Configuring Key/Certificate for Cisco NX-OS Release 9.3(3) and Later, on page 11
- Verifying gNMI, on page 13
- gRPC Client-Certificate-Authentication, on page 18
- Generating New Client Root CA Certificates, on page 18
- Configuring the Generated Root CA Certificates on NX-OS Device, on page 19
- Associating Trustpoints to gRPC, on page 19
- Validating the Certificate Details, on page 20
- Verifying the Connection using Client Certificate Authentication for any gNMI Clients, on page 21
- Accounting Log for gNMI, on page 21
- Sample DME Subscription - PROTO Encoding, on page 24
- NGINX proxy for GRPC, on page 26
- Configuration Needed, on page 26
- Capabilities, on page 27
- Get, on page 30
- Set, on page 31
- Subscribe, on page 32
- Streaming Syslog, on page 37
- Troubleshooting, on page 43

About gNMI

gNMI uses gRPC (Google Remote Procedure Call) as its transport protocol.

Cisco NX-OS supports gNMI for dial-in subscription to telemetry applications running on the Cisco Nexus 9000 Series switches. Although past release supported telemetry events over gRPC, the switch pushed the telemetry data to the telemetry receivers. This method was called dial out.

With gNMI, applications can pull information from the switch. They subscribe to specific telemetry services by learning the supported telemetry capabilities and subscribing to only the telemetry services that it needs.

Table 1: Supported gNMI RPCs

gNMI RPC	Supported
Capabilities	Yes
Get	Yes
Set	Yes
Subscribe	Yes

gNMI Subscribe RPC

The Cisco NX-OS 9.3(1) release and later support the following gNMI Subscription features:

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 604800 seconds. The default sample interval is 10 seconds.

Type	Sub Type	Supported?	Description
	On_Change	Yes	The switch sends current values as its initial state, but then updates the values only when changes, such as create, modify, or delete occur to any of the specified paths.
	Target_Defined	Yes	When you create a subscription specifying the target define mode, the target must define the best type of subscription to be created.

Note Beginning with 10.2(1)F release, Target_Defined sub type subscribe option is supported.

Beginning with Cisco NX-OS Release 10.2(3)F, a new CLI command is introduced that changes the keepalive interval of the gNMI subscription. The settable limits are 600 seconds to 86400 seconds.

The command is **"[no] grpc gnmi keepalive-timeout <timeout>"**. For example, **switch(config)# grpc gnmi keepalive-timeout 600**.

The following is the an example to verify the CLI command:

```
Verify in show statistics cmd

switch(config)# sh grpc gnmi service statistics

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

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

Cert notBefore : Feb  6 01:15:06 2022 GMT
Cert notAfter  : Feb  7 01:15:06 2022 GMT
Client Root Cert notBefore : n/a
Client Root Cert notAfter  : n/a

Max concurrent calls            :  8
Listen calls                    :  1
Active calls                    :  0
KeepAlive Timeout                    :  1000
```

The following are the guidelines for the CLI command:

- The gnmi server would send an empty response to the subscription clients every given interval.
- The purpose is to detect and cleanup rogue/dangling client connections.
- By default the keepalive interval is 600 seconds.
- This command changes the interval to the user-specified value.

Optional SUBSCRIBE Flags

For the SUBSCRIBE option, some optional flags are available that modify the response to the options listed in the table. In Cisco NX-OS release 9.3(1) and later, the updates_only optional flag is supported, which is applicable to ON_CHANGE subscriptions. If this flag is set, the switch suppresses the initial snapshot data (current state) that is normally sent with the first response.

The following flags are not supported:

- aliases
- allow_aggregation
- extensions
- prefix
- qos

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

- heartbeat_interval
- suppress_redundant

A heartbeat_interval may be specified to modify the behavior of suppress_redundant in a sampled subscription. In this case, the target must generate one telemetry update per heartbeat interval regardless of whether the suppress_redundant flag is set to true. This value is specified as an unsigned 64-bit integer in nanoseconds.

The suppress_redundant field of the subscription message may be set for a sampled subscription. In the case that it is set to true, the target must not generate a telemetry update message unless the value of the path being reported on has changed since the last update was generated. Updates must only be generated for those individual leaf nodes in the subscription that have changed.

For example, a subscription to /A/B - where there are leaves C and D branching from the B node - if the value of C has changed, but D remains unchanged, an update for D must not be generated, whereas an update for C must be generated.

The following is the example for the supported Optional SUBSCRIBE Flags:

```
{
 "SubscribeRequest":
 [
  {
   "subscribe":
   {
    "subscription":
    [
     {
      "_comment" : "1st subscription path",
      "path":
      {
       "origin": "openconfig",
       "elem":
       [
        {
         "name": "interfaces/interface[name=eht1/1]"
        }
       ]
      },
      "mode": "SAMPLE",
      "heartbeat_interval": 30000000000
```

```
    "suppress-redundant": true
   },
  "mode": "STREAM",
  "allow_aggregation" : false,
  "use_models":
  [
   {
    "name": "DME",
    "organization": "Cisco Systems, Inc.",
    "version": "1.0.0"
   }
  ],
  "encoding": "JSON"
 }
}
```

The following is the support metrics for the subscribe flags:

Table 3: Support Metrics for SUBSCRIBE flags

Subscription Type	heartbeat_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

Guidelines and Limitations for gNMI

Following are the guidelines and limitations for gNMI:

- Beginning with Cisco NX-OS Release 9.3(5), Get and Set are supported.
- gNMI queries do not support wildcards in paths.
- gRPC traffic destined for a Nexus device will hit the control-plane policer (CoPP) in the default class. To limit the possibility of gRPC drops, configure a custom CoPP policy using the gRPC configured port in the management class.
- When you enable gRPC on both the management VRF and default VRF and later disable on the default VRF, the gNMI notifications on the management VRF stop working.

  As a workaround, disable gRPC completely by entering the **no feature grpc** command and reprovision it by entering the **feature grpc** command and any existing gRPC configuration commands. For example, **grpc certificate** or **grpc port**. You must also resubscribe to any existing notifications on the management VRF.
- When you attempt to subscribe an OpenConfig routing policy with a preexisting CLI configuration like the following, it returns empty values due to the 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

  using the xpath

  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
  ```

- Only server certificate authentication takes place. The client certificate is not authenticated by the server.
- If the gRPC certificate is explicitly configured, after a reload with the saved startup configuration to a prior Cisco NX-OS 9.3(x) image, the gRPC feature does not accept connections. To confirm this issue, enter the **show grpc gnmi service statistics** command and the status line displays an error like the following:

  ```
  Status: Not running - Initializing...Port not available or certificate invalid.
  ```

  Unconfigure and configure the proper certificate command to restore the service.
- The reachability in non-default VRF for gRPC is supported only over L3VNI's/EVPN and IP. However, reachability over MPLS in non-default VRF and VXLAN Flood and Learn is not supported.
- Use of origin, use_models, or both, is optional for gNMI subscriptions.
- gNMI Subscription supports Cisco DME and Device YANG data models. Beginning with Cisco NX-OS Release 9.3(3), Subscribe supports the OpenConfig model.
- For Cisco NX-OS prior to 9.3(x), 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 the Nexus Switch Platform Matrix.
- The feature supports JSON and gnmi.proto encoding. The feature does not support protobuf.any encoding.
- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the 12 MB maximum, the collected data is dropped. Applies to gNMI ON_CHANGE mode only.

  You can avoid this situation by creating more focused subscriptions that handle smaller, more granular data-collection sets. So, instead of subscribing to one higher-level path, create multiple subscriptions for different, lower-level parts of the path.
- Across all subscriptions, there is support of up to 150K aggregate MOs. Subscribing to more MOs can lead to collection data drops.
- The feature does not support a path prefix in the Subscription request, but the Subscription can contain an empty prefix field.
- The gRPC process that supports gNMI uses the HIGH_PRIO control group, which limits the CPU usage to 75% of CPU and memory to 1.5 GB.
- The **show grpc gnmi** command has the following considerations:
  - The gRPC agent retains gNMI calls for a maximum of one hour after the call has ended.
  - If the total number of calls exceeds 2000, the gRPC agent purges ended calls based on the internal cleanup routine.
- Beginning with Cisco NX-OS Release 10.2(3)F, on change subscription of Device YANG ephemeral data (Accounting-log and Multicast) is supported.

The gRPC server runs in the management VRF. As a result, the gRPC process communicates only in this VRF forcing the management interface to support all gRPC calls.

gRPC functionality now includes the default VRF for a total of two gRPC servers on each switch. You can run one gRPC server in each VRF, or run only one gRPC server in the management VRF. Supporting a gRPC

in the default VRF adds flexibility to offload processing gRPC calls from the management VRF, where significant traffic load is not desirable.

If two gRPC servers are configured, be aware of the following:

- VRF boundaries are strictly enforced, so each gRPC server process requests independent of the other. Requests do not cross between VRFs.
- The two servers are not HA or fault tolerant. One gRPC server does not back up the other, and there is no switchover or switchback between them.
- Any limits for the gRPC server are per VRF.

The following are the limitations for gNMI:

- multi-level wildcard "..." in path is not allowed
- wildcard '*' in the top of the path is not allowed
- wildcard '*' in key name is not allowed
- wildcard and value cannot be mixed in keys

The following table shows the wildcard support details for gNMI:

Table 4: 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

Configuring gNMI

Configure the gNMI feature through the **grpc gnmi** commands.

To import certificates used by the **grpc certificate** command onto the switch, see the Installing Identity Certificates section of the Cisco Nexus 9000 Series NX-OS Security Configuration Guide, Release 9.3(x).

Note When modifying the installed identity certificates or **grpc port** and **grpc certificate** values, the gRPC server might restart to apply the changes. When the gRPC server restarts, any active subscription is dropped and you must resubscribe.

SUMMARY STEPS

1. **configure terminal**
2. **feature grpc**
3. (Optional) **grpc port** *port-id*
4. **grpc certificate** *certificate-id*
5. **grpc gnmi max-concurrent-call** *number*
6. (Optional) **grpc use-vrf default**
7. **grpc gnmi subscription target-defined min-interval**
8. **grpc gnmi subscription query-condition keep-data-timestamp**

DETAILED STEPS

Procedure

	Command or Action	Purpose

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

Step 2	**feature grpc**
	Example:
	`switch-1# feature grpc`
	`switch-1(config)#`
		Enables the gRPC agent, which supports the gNMI interface for dial-in.

Step 3	(Optional) **grpc port** *port-id* **Example:** `switch-1(config)# grpc port 50051`	Configure the port number. The range of *port-id* is from 1024 to 65535. 50051 is the default. **Note** This command is available beginning with Cisco NX-OS Release 9.3(3).
Step 4	**grpc certificate** *certificate-id* **Example:** `switch-1(config)# grpc certificate cert-1`	Specify the certificate trustpoint ID. For more information, see the Installing Identity Certificates section of the Cisco Nexus 9000 Series NX-OS Security Configuration Guide, Release 9.3(x) for importing the certificate to the switch. **Note** This command is available beginning with Cisco NX-OS Release 9.3(3).

Step 5	**grpc gnmi max-concurrent-call** *number*
	Example:
	`switch-1(config)# grpc gnmi max-concurrent-call 16`
	`switch-1(config)#`
		Sets the limit of simultaneous dial-in calls to the gNMI server on the switch. Configure a limit from 1 through 16. The default limit is 8.
		The maximum value that you configure is for each VRF. If you set a limit of 16 and gNMI is configured for both management and default VRFs, each VRF supports 16 simultaneous gNMI calls.

	Command or Action	Purpose
		This command does not affect and ongoing or in-progress gNMI calls. Instead, gRPC enforces the limit on new calls, so any in-progress calls are unaffected and allowed to complete. **Note** The configured limit does not affect the gRPCConfigOper service.
Step 6	(Optional) **grpc use-vrf default** **Example:** `switch(config)# grpc use-vrf default`	Enables the gRPC agent to accept incoming (dial-in) RPC requests from the default VRF. This step enables the default VRF to process incoming RPC requests. By default, the management VRF processes incoming RPC requests when the gRPC feature is enabled. **Note** Both VRFs process requests individually, so that requests do not cross between VRFs.
Step 7	**grpc gnmi subscription target-defined min-interval** **Example:** `switch(config)# grpc gnmi subscription target-defined min-interval ? <1-65535> Default 30`	Allows the user to modify the default target-defined sample interval from 30 seconds to other value.
Step 8	**grpc gnmi subscription query-condition keep-data-timestamp**	This command enables sample/once/poll subscriptions to get timestamp from database when the data was last updated. **Note** • This feature is supported only for PROTO and not JSON encoding. • This feature is supported only for once, poll, sample, and not on_change. • This feature is supported for DME, YANG, and OpenConfig data sources. • Most of the other properties are supported, but for the properties which are not supported, this feature will default back to the collection time instead of last database change time. **Note** This feature would generate verbose responses for each timestamp. If the user client is not able to consume the messages on time, the switch may need to drop the collection/messages.

Configuring Server Certificate

When you configured a TLS certificate and imported successfully onto the switch, the following is an example of the **show grpc gnmi service statistics** command output.

```
switch(config)# sh grpc gnmi service statistics

============= gRPC Endpoint
Vrf : management
Server address : [::]:50051

Cert notBefore : Nov 5 16:48:58 2015 GMT
Cert notAfter : Nov 5 16:48:58 2035 GMT
Client Root Cert notBefore : n/a
Client Root Cert notAfter : n/a

Max concurrent calls : 8
Listen calls : 1
Active calls : 0
KeepAlive Timeout : 120

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

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

Max gNMI::Get concurrent : 6
Max grpc message size : 25165824
gNMI Synchronous calls : 3
gNMI Synchronous errors : 3
gNMI Adapter errors : 3
gNMI Dtx errors : 0
```

gNMI communicates over gRPC and uses TLS to secure the channel between the switch and the client. The default hard-coded gRPC certificate is no longer shipped with the switch. The default behavior is a self-signed key and certificate which is generated on the switch as shown below with an expiration date of one day.

When the certificate is expired or failed to install successfully, you will see the 1-D default certificate. The following is an example of the **show grpc gnmi service statistics** command output.

```
#show grpc gnmi service statistics

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

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

Cert notBefore : Wed Mar 11 19:43:01 PDT 2020
Cert notAfter  : Thu Mar 12 19:43:01 PDT 2020

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
```

With an expiration of one day, you can use this temporary certificate for quick testing. For long term a new key/certificate must be generated.

Generating Key/Certificate Examples

Follow these examples to generate Key/Certificates:

- Examples for Generating and Configuring Key/Certificate for Cisco NX-OS Release 9.3(3) and Later

Examples for Generating and Configuring Key/Certificate for Cisco NX-OS Release 9.3(3) and Later

The following is an example for generating key/certificate.

Note This task is an example of how a certificate can be generated on a switch. You can also generate a certificate in any Linux environment. In a production environment, you should consider using a CA signed certificate.

For more information on generating identity certificates, see the Installing Identity Certificates section of the *Cisco Nexus 9000 Series NX-OS Security Configuration Guide, Release 9.3(x)*.

Procedure

Step 1 Generate the selfsigned key and pem files.

```
switch# run bash sudo su
bash-4.3# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out self_sign2048.pem -days
 365 -nodes
```

Step 2 After generating the key and pem files, you must bundle the key and pem files for use in the trustpoint CA Association.

```
switch# run bash sudo su
bash-4.3# cd /bootflash/
bash-4.3# openssl pkcs12 -export -out self_sign2048.pfx -inkey self_sign2048.key -in self_sign2048.pem
 -certfile self_sign2048.pem -password pass:Ciscolab123!
bash-4.3# exit
```

Step 3 Set up the trustpoint CA Association by inputting in the pkcs12 bundle into the trustpoint.

```
switch(config)# crypto ca trustpoint mytrustpoint
switch(config-trustpoint)# crypto ca import mytrustpoint pkcs12 self_sign2048.pfx Ciscolab123!
```

Step 4 Verify the setup.

```
switch(config)# show crypto ca certificates
Trustpoint: mytrustpoint
certificate:
subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
```

```
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
serial=0413
notBefore=Nov  5 16:48:58 2015 GMT
notAfter=Nov  5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient

CA certificate 0:
subject= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
issuer= /C=US/O=Cisco Systems, Inc./OU=CSG/L=San Jose/ST=CA/street=3700 Cisco
Way/postalCode=95134/CN=ems.cisco.com/serialNumber=FGE18420K0R
serial=0413
notBefore=Nov  5 16:48:58 2015 GMT
notAfter=Nov  5 16:48:58 2035 GMT
SHA1 Fingerprint=2E:99:2C:CE:2F:C3:B4:EC:C7:E2:52:3A:19:A2:10:D0:54:CA:79:3E
purposes: sslserver sslclient
```

Step 5 Configure gRPC to use the trustpoint.

```
switch(config)# grpc certificate mytrustpoint
switch(config)# show run grpc

!Command: show running-config grpc
!Running configuration last done at: Thu Jul  2 12:24:02 2020
!Time: Thu Jul  2 12:24:05 2020

version 9.3(5) Bios:version 05.38
feature grpc

grpc gnmi max-concurrent-calls 16
grpc use-vrf default
grpc certificate mytrustpoint
```

Step 6 Verify gRPC is now using the certificate.

```
switch# show grpc gnmi service statistics

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

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

Cert notBefore : Nov 5 16:48:58 2015 GMT
Cert notAfter : Nov 5 16:48:58 2035 GMT

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

Number of created calls : 953
Number of bad calls : 0

Subscription stream/once/poll : 476/238/238

Max gNMI::Get concurrent : 5
Max grpc message size : 8388608
gNMI Synchronous calls : 10
gNMI Synchronous errors : 0
gNMI Adapter errors : 0
```

```
gNMI Dtx errors : 0
```

Verifying gNMI

To verify the gNMI configuration, enter the following command:

Command	Description
show grpc gnmi service statistics	Displays a summary of the agent running status, respectively for the management VRF, or the default VRF (if configured). It also displays: • Basic overall counters • Certificate expiration time **Note** If the certificate is expired, the agent cannot accept requests.
show grpc gnmi rpc summary	Displays the following: • Number of capability RPCs received. • Capability RPC errors. • Number of Get RPCs received. • Get RPC errors. • Number of Set RPCs received. • Set RPC errors. • More error types and counts.

Command	Description

show grpc gnmi transactions	The **show grpc gnmi transactions** command is the most dense and contains considerable information. It is a 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:
	• RPC – This shows the type of RPC that was received (Get, Set, Capabilities)
	• DataType – For a Get only. Has values ALL, CONFIG, and STATE.
	• Session – shows 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 -- shows timestamp of when the RPC was received by the gNMI handler.
	• Duration – time delta in ms from receiving the request to giving response.
	• Status – the status code of the operation returned to the client `(0 = Success, !0 == error)`
	This section is data that is kept per path within a single gNMI transaction. For example, a single Get or Set
	• subtype – for a Set RPC, shows the specific operation that is requested per path (Delete, Update, Replace). For Get, there is no subtype.
	• dtx – shows that this path was processed in DTX "fast" path or not. A dash '-' means no, an asterisk '*' means yes.
	• st – Status for this path. The meaning is as follows:
	  • OK: path is valid and processed by infra successfully.
	  • ERR: path is either invalid or generated error by infra
	  • --: path not processed yet, might or might not be valid and has not been sent to infra yet.
	• path – the path

show grpc gnmi service statistics Example

```
=============
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
```

show grpc gnmi rpc summary Example

```
=============
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
```

show grpc gnmi transactions Example

```
=============
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          -          2293989720      04/01 07:43:45       183          0
subtype: dtx:  st: path:
Replace  -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo6]

Set          -          2297110560      04/01 07:43:41       184          0
subtype: dtx:  st: path:
Update   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo7]

Set          -          0               04/01 07:43:39       0            10

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=lo791]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo792]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo793]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo794]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo795]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo796]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo797]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo798]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo799]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo800]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo801]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo802]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo803]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo804]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo805]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo806]
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          -          2294408864      04/01 07:43:17       176          13
subtype: dtx:  st: path:
Delete   -     ERR /System/intf-items/lb-items/LbRtdIf-list/descr


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]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo777]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo778]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo779]
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo780]
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=lo4]/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

Set          -          3551604840      04/01 07:42:54       35           0
subtype: dtx:  st: path:
Delete   -     OK  /System/intf-items/lb-items/LbRtdIf-list[id=lo1]

Set          -          2362201592      04/01 07:42:52       13           13
subtype: dtx:  st: path:
Delete   -     ERR /System/intf-items/lb-items/LbRtdIf-list[id=lo3]/lbrtdif-items
/operSt


Set          -          0               04/01 07:42:47       0            3
subtype: dtx:  st: path:
Delete   -     ERR /System/*

Set          -          2464158360      04/01 07:42:46       172          3
subtype: dtx:  st: path:
Delete   -     ERR /system/processes/shabang

Set          -          2295440864      04/01 07:42:46       139          3
subtype: dtx:  st: path:
Delete   -     ERR /System/invalid/path

Set          -          3495739048      04/01 07:42:44       10           0


Get          ALL        3444580832      04/01 07:42:40       3            0
subtype: dtx:  st: path:
-        -     OK  /System/bgp-items/inst-items/disPolBatch

Get          ALL        0               04/01 07:42:36       0            3
subtype: dtx:  st: path:
-        -     --  /system/processes/process[pid=1]

Get          ALL        3495870472      04/01 07:42:36       2            0
subtype: dtx:  st: path:
-        *     OK  /system/processes/process[pid=1]

Get          ALL        2304485008      04/01 07:42:36       33           0
subtype: dtx:  st: path:
-        *     OK  /system/processes

Get          ALL        2464159088      04/01 07:42:36       251          0
subtype: dtx:  st: path:
-        -     OK  /system
```

```
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
```

gRPC Client-Certificate-Authentication

Beginning with 10.1(1) release, an additional authentication method is provided for gRPC. gRPC services prior to 10.1(1) release supported only the server certificate. Starting from 10.1(1), authentication is enhanced to add support for client certificate as well so that gRPC allows to verify both server certificate and client certificate. This enhancement provides password-less authentication for different Clients.

Generating New Client Root CA Certificates

The following is the example for generating a new certificate to the client root:

- Trusted Certificate Authorities (CA)

Perform the following steps when you use a trusted CA such as a DigiCert:

SUMMARY STEPS

1. Download the CA certificate file.
2. Import to NX-OS using the steps in Cisco NX-OS Security Configuration Guide.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Download the CA certificate file.	
Step 2	Import to NX-OS using the steps in Cisco NX-OS Security Configuration Guide.	• To create a trustpoint label, use steps in Creating a Trustpoint CA Association • To authenticate the trustpoint using the trusted CA certificates, use steps in Authenticating the CA. **Note** Use the CA Certificate from cat [CA_cert_file].

Configuring the Generated Root CA Certificates on NX-OS Device

When you have generated a new certificate to the client root successfully, following are the sample commands to configure them in the switch, and their output.

```
switch(config)# crypto ca trustpoint my_client_trustpoint
enticate my_client_trustpoint
switch(config-trustpoint)# crypto ca authenticate my_client_trustpoint
input (cut & paste) CA certificate (chain) in PEM format;
end the input with a line containing only END OF INPUT :
-----BEGIN CERTIFICATE-----
MIIDUDCCAjigAwIBAgIJAJLisBKCGjQOMA0GCSqGSIb3DQEBCwUAMD0xCzAJBgNV
BAYTAlVTMQswCQYDVQQIDAJDQTERMA8GA1UEBwwIU2FuIEpvc2UxDjAMBgNVBAoM
BUNpc2NvMB4XDTIwMTAxNDIwNTYyN1oXDTQwMTAwOTIwNTYyN1owPTELMAkGA1UE
BhMCVVMxCzAJBgNVBAgMAkNBMREwDwYDVQQHDAhTYW4gSm9zZTEOMAwGA1UECgwF
Q2lzY28wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDEX7qZ2EdogZU4
EW0NSpB3EjY0nSlFLOw/iLKSXfIiQJD0Qhaw16fDnnYZj6vzWEa0ls8canqHCXQl
gUyxFOdGDXa6neQFTqLowSA6UCSQA+eenN2PIpMOjfdFpaPiHu3mmcTI1xP39Ti3
/y548NNORSepApBNkZ1rJSB6Cu9AIFMZgrZXFqDKBGSUOf/CPnvIDZeLcun+zpUu
CxJLA76Et4buPMysuRqMGHIX8CYw8MtjmuCuCTHXNN31ghhgpFxfrW/69pykjU3R
YOrwlSUkvYQhtefHuTHBmqym7MFoBEchwrlC5YTduDzmOvtkhsmpogRe3BiIBx45
AnZdtdi1AgMBAAGjUzBRMB0GA1UdDgQWBBSh3IqRrm+mtB5GNsoLXFb3bAVg5TAf
BgNVHSMEGDAWgBSh3IqRrm+mtB5GNsoLXFb3bAVg5TAPBgNVHRMBAf8EBTADAQH/
MA0GCSqGSIb3DQEBCwUAA4IBAQAZ4Fpc6lRKzBGJQ/7oK1FNcTX/YXkneXDk7Zrj
8W0RS0Khxgke97d2Cwl5P5reXO27kvXsnsz/VZn7JYGUvGSlxTlcCb6x6wNBr4Qr
t9qDBu+LykwqNOFe4VCAv6e4cMXNbH2wHBVS/NSoWnM2FGZ10VppjEGFm6OM+N6z
8n4/rWslfWFbn7T7xHH+Nl0Ffc+8q8h37opyCnb0ILj+a4rnyus8xXJPQb05DfJe
ahPNfdEsXKDOWkrSDtmKwtWDqdtjSQC4xioKHoshnNgWBJbovPlMQ64UrajBycwV
z9snWBm6p9SdTsV92YwF1tRGUqpcI9olsBgH7FUVU1hmHDWE
-----END CERTIFICATE-----
END OF INPUT
Fingerprint(s): SHA1 Fingerprint=0A:61:F8:40:A0:1A:C7:AF:F2:F7:D9:C7:12:AE:29:15:52:9D:D2:AE


Do you accept this certificate? [yes/no]:yes
switch(config)#

NOTE: Use the CA Certificate from the .pem file content.

switch# show crypto ca certificates
Trustpoint: my_client_trustpoint
CA certificate 0:
subject=C = US, ST = CA, L = San Jose, O = Cisco
issuer=C = US, ST = CA, L = San Jose, O = Cisco
serial=B7E30B8F4168FB87
notBefore=Oct  1 17:29:47 2020 GMT
notAfter=Sep 26 17:29:47 2040 GMT
SHA1 Fingerprint=E4:91:4E:D4:41:D2:7D:C0:5A:E8:F7:2D:32:81:B3:37:94:68:89:10
purposes: sslserver sslclient
```

Associating Trustpoints to gRPC

When you have configured a new certificate to the client root successfully, the following is the output example for associating trustpoints to gRPCs on the switch:

Note Configuring or removing the root certificate for client authentication will cause gRPC process to restart.

```
# switch(config)# feature grpc

switch(config)# grpc client root certificate my_client_trustpoint
switch(config)# show run grpc

!Command: show running-config grpc
!Running configuration last done at: Wed Dec 16 20:18:35 2020
!Time: Wed Dec 16 20:18:40 2020

version 10.1(1) Bios:version N/A
feature grpc

grpc gnmi max-concurrent-calls 14
grpc use-vrf default
grpc certificate my_trustpoint
grpc client root certificate my_client_trustpoint
grpc port 50003
```

Validating the Certificate Details

When you have successfully associated the trustpoints to gRPC on the switch, the following is the output example for validating the certificate details:

```
switch# show grpc gnmi service statistics

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

Vrf : management
Server address : [::]:50003

Cert notBefore : Mar 13 19:05:24 2020 GMT
Cert notAfter : Nov 20 19:05:24 2033 GMT
Client Root Cert notBefore : Oct 1 17:29:47 2020 GMT
Client Root Cert notAfter : Sep 26 17:29:47 2040 GMT

Max concurrent calls : 14
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 : 0
gNMI Synchronous errors : 0
gNMI Adapter errors : 0
gNMI Dtx errors : 0
```

Verifying the Connection using Client Certificate Authentication for any gNMI Clients

The client certificate requests with a private key (pkey) and ca chain (cchain). The password is now optional.

```
Performing GetRequest, encoding = JSON to 172.19.199.xxx with the following gNMI Path
 -------------------------
 [elem {
  name: "System"
}
elem {
  name: "bgp-items"
}
]
The GetResponse is below
-------------------------

notification {
  timestamp: 1608071208072199559
  update {
    path {
      elem {
        name: "System"
      }
      elem {
        name: "bgp-items"
      }
    }
    val {
      json_val: ""
    }
  }
}
```

For removing trustpoint reference from gRPC (no command) use the following command:

```
 [no] grpc client root certificate <my_client_trustpoints>
switch(config)# no grpc client root certificate my_client_trustpoint
```

The command will remove the trustpoint reference only from gRPC agent, but the trustpoints CA certificates will NOT be removed. Connections that use client certificate authentication to gRPC server on switch will not establish, but basic authentication with username and password will go through.



Note If the client's certificate is signed by intermediate CAs, but not directly by the root CA that is imported from the above config, the grpc client needs to supply the full cert chain, including the user, intermediate CA cert, and the root CA cert.

Accounting Log for gNMI

In the GNMI, SET RPC changes the configuration on the switch. For the `SET` requests such as `UPDATE`, `REPLACE` or `DELETE`, gNMI would emit the corresponding accounting log. It would include both the original received request, as well as the eventual changes applied to the switch.

You can see the accounting log using the **show accounting log** command.

Consider the following example request.

Performing 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 accounting log shall include the following items:

- Changes Applied to the Switch:

Item	Description
Context	Session ID and user
Operation	**COMMIT/ABORT**
Database	Running or Candidate
ConfigMO	MO tree's text representation. Up to 3K characters.
Status	`SUCCESS/FAILED`

Example:

```
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)
```

- Original Received Request:

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. Up to 3K characters.
Status	`SUCCESS/FAILED`

Example:

```
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, based on the failed scenarios, a user may not observe both the logs.

- Invalid Request:

  The invalid request would be rejected without making a configuration change, thus only the original request would be logged.

  Example:

```
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"}] (FAILED)
```

- Request Fails due to Various Configuration Restrictions:

  In this case, both the failed configuration attempt and the original request would be logged.

  Example:

```
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)
```

- Request Fails to Commit:

  Both configuration attempt and the original request would be logged successfully along with the failed commit.

  Example:

```
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)
```

Sample DME Subscription - PROTO Encoding

```
gnmi-console --host >iip> --port 50051 -u <user> -p <pass> --tls --
operation=Subscribe --rpc /root/gnmi-console/testing_bl/once/61_subscribe_bgp_dme_gpb.json

[Subscribe]--------------------------------
### Reading from file ' /root/gnmi-console/testing_bl/once/61_subscribe_bgp_dme_gpb.json '
Wed Jun 26 11:49:17 2019
### Generating request : 1 -----------
### Comment : ONCE request
### Delay : 2 sec(s) ...
### Delay : 2 sec(s) DONE
subscribe {
subscription {
path {
origin: "DME"
elem {
name: "sys"
}
elem {
name: "bgp"
}
}
mode: SAMPLE
}
mode: ONCE
use_models {
name: "DME"
organization: "Cisco Systems, Inc."
version: "1.0.0"
}
encoding: PROTO
}
Wed Jun 26 11:49:19 2019
Received response 1 --------------------------
```

```
update {
timestamp: 1561574967761
prefix {
elem {
name: "sys"
}
elem {
name: "bgp"
}
}
update {
path {
elem {
}
elem {
name: "version_str"
}
}
val {
string_val: "1.0.0"
}
}
update {
path {
elem {
}
elem {
name: "node_id_str"
}
}
val {
string_val: "n9k-tm2"
}
}
update {
path {
elem {
}
elem {
name: "encoding_path"
}
}
val {
string_val: "sys/bgp"
}
}
update {
path {
elem {
}
elem {
/Received --------------------------------------
Wed Jun 26 11:49:19 2019
Received response 2 --------------------------
sync_response: true
/Received --------------------------------------
(_gnmi) [root@tm-ucs-1 gnmi-console]#
```

NGINX proxy for GRPC

gNMI and gNOI requests are handled by the gRPC agent running on the switch. NGINX is the HTTP server used to access NX-API services. Starting from release 10.3(3), NGINX can act as GRPC proxy by receiving the gNMI and gNOI requests and forwarding them to the GRPC agent. This can be useful for certain use cases.

- GRPC port blocked: The GRPC agent listens on port 50051. If this port is blocked by firewall, GRPC clients can access the gRPC services through the HTTPS port 443.
- Increased VRF support: Currently GRPC services are accessible only via default and management VRFs. NGINX proxy can provide access on any VRF.

Configuration Needed

Now a GRPC client can connect to the GRPC agent directly, or connect to the NGINX server, which proxy the GRPC requests to the GRPC agent.

GRPC Services Through NGINX

All server and client authentication will be handled by NGINX. Just enable GRPC and configure NGINX server certificate and/or client certificates.

- Server certificate authentication: configure the NXAPI server certificate. See Using NX-API CLI for details.

```
feature grpc
feature nxapi
nxapi certificate httpscrt certfile bootflash:nxapi.crt
nxapi certificate httpskey keyfile bootflash:nxapi.key password cisco123

nxapi certificate enable
```

- Client certificate authentication: configure the NXAPI server and client certificates. See Using NX-API CLI for details.

```
feature grpc
feature nxapi
nxapi certificate httpscrt certfile bootflash:nxapi.crt
nxapi certificate httpskey keyfile bootflash:nxapi.key password cisco123

nxapi certificate enable
crypto ca trustpoint grpcClientCA
crypto ca authenticate grpcClientCA
nxapi client certificate authentication
```

GRPC Services Directly to The GRPC Agent

All server and client authentication will be handled by the GRPC agent. The regular GRPC server and client certificate configurations apply in this case.

Capabilities

About Capabilities

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

Guidelines and Limitations for Capabilities

Following are the guidelines and limitations for Capabilities:

- Beginning with Cisco NX-OS Release 9.3(3), Capabilities supports the OpenConfig model.
- For information about supported platforms, see Nexus Switch Platform Matrix.
- The gNMI feature supports Subscribe and Capability as options of the gNMI service.
- The feature supports JSON and gnmi.proto encoding. The feature does not support protobuf.any encoding.
- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the 12-MB maximum, the collected data is dropped.

  You can avoid this situation by creating more focused subscriptions that handle smaller, more granular data-collection sets. So, instead of subscribing to one higher-level path, create multiple subscriptions for different, lower-level parts of the path.
- All paths within the same subscription request must have the same sample interval. If the same path requires different sample intervals, create multiple subscriptions.
- The feature does not support a path prefix in the Subscription request, but the Subscription can contain an empty prefix field.
- The feature supports Cisco DME and Device YANG data models.
- The gRPC process that supports gNMI uses the HIGH_PRIO cgroup, which limits the CPU usage to 75% of CPU and memory to 1.5 GB.
- The **show grpc gnmi** command has the following considerations:
  - The commands are not XMLized in this release.
  - The gRPC agent retains gNMI calls for a maximum of 1 hour after the call has ended.
  - If the total number of calls exceeds 2000, the gRPC agent purges ended calls based an internal cleanup routine.

The gRPC server runs in the management VRF. As a result, the gRPC process communicates only in this VRF forcing the management interface to support all gRPC calls.

gRPC functionality now includes the default VRF for a total of 2 gRPC servers on each Cisco Nexus 9000 switch. You can run one gRPC server in each VRF, or run only one gRPC server in the management VRF. Supporting a gRPC in the default VRF adds flexibility to offload processing gRPC calls from the management VRF, where significant traffic load might not be desirable.

If two gRPC servers are configured, be aware of the following:

- VRF boundaries are strictly enforced, so each gRPC server processes requests independent of the other, and requests do not cross between VRFs.
- The two servers are not HA or fault tolerant. One gRPC server does not back up the other, and there is no switchover or switchback between them.
- Any limits for the gRPC server are per VRF.

Example Client Output for Capabilities

In this example, all the OpenConfig model RPMs have been installed on the switch.

The following is an example of client output for Capabilities.

```
hostname user$ ./gnmi_cli -a 172.19.193.166:50051 -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-interfaces"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-aggregate"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-ethernet"
  organization: "OpenConfig working group"
  version: "2.0.0"
>
supported_models: <
  name: "openconfig-if-ip"
  organization: "OpenConfig working group"
  version: "2.3.0"
>
supported_models: <
  name: "openconfig-if-ip-ext"
  organization: "OpenConfig working group"
  version: "2.3.0"
>
supported_models: <
  name: "openconfig-lacp"
  organization: "OpenConfig working group"
  version: "1.0.2"
>
```

```
supported_models: <
  name: "openconfig-lldp"
  organization: "OpenConfig working group"
  version: "0.2.1"
>
supported_models: <
  name: "openconfig-network-instance"
  organization: "OpenConfig working group"
  version: "0.11.1"
>
supported_models: <
  name: "openconfig-network-instance-policy"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-ospf-policy"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-platform"
  organization: "OpenConfig working group"
  version: "0.12.2"
>
supported_models: <
  name: "openconfig-platform-cpu"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-platform-fan"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-platform-linecard"
  organization: "OpenConfig working group"
  version: "0.1.1"
>
supported_models: <
  name: "openconfig-platform-port"
  organization: "OpenConfig working group"
  version: "0.3.2"
>
supported_models: <
  name: "openconfig-platform-psu"
  organization: "OpenConfig working group"
  version: "0.2.1"
>
supported_models: <
  name: "openconfig-platform-transceiver"
  organization: "OpenConfig working group"
  version: "0.7.0"
>
supported_models: <
  name: "openconfig-relay-agent"
  organization: "OpenConfig working group"
  version: "0.1.0"
>
supported_models: <
  name: "openconfig-routing-policy"
  organization: "OpenConfig working group"
  version: "2.0.1"
```

```
>
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:

hostname user$
```

Get

About Get

The purpose of the Get RPC is to allow a client to retrieve a snapshot of the data tree from the device. Multiple paths may be requested in a single request. A simplified form of XPATH according to the gNMI Path Conventions, Schema path encoding conventions for gNMI are used for the path.

For detailed information on the Get operation, refer to the Retrieving Snapshots of State Information section in the gNMI specification: gRPC Network Management Interface (gNMI)

Guidelines and Limitations for Get

The following are guidelines and limitations for Get and Set:

- GetRequest.encoding supports only JSON.
- For GetRequest.type, only DataType CONFIG and STATE have direct correlation and expression in YANG. OPERATIONAL is not supported.

- A single request cannot have both OpenConfig (OC) YANG and device YANG paths. A request must have only OC YANG paths or device YANG paths, but not both.
- GetRequest for root path ("/": everything from **all** models) is not allowed.
- gNMI Get returns all default values (ref. report-all mode in RFC 6243 [4]).
- Subscribe supports the model `Cisco-NX-OS-syslog-oper`.
- Get does not support the model `Cisco-NX-OS-syslog-oper`.
- To retrieve `openconfig-procmon` data, the query can be sent either to the path `/system/processes` or `/system`. The path `/system` will not retrieve the data in the releases prior to 10.3.0 release.
- The following optional items are not supported:
  - Path prefix
  - Path alias
  - Wildcards in path
- A single GetRequest can have up to 10 paths.
- If the size of value field to be returned in GetResponse is over 12 MB, the system returns error status`grpc::RESOURCE_EXHAUSTED`.
- The maximum gRPC receive buffer size is set to 8 MB.
- Performing a Get operation when a large configuration is applied to the switch might cause the gRPC process to consume all available memory. If a memory exhaustion condition is hit, the following syslog is generated:

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

  If this condition is hit several times consecutively, the following syslog is generated:

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

  We recommend that you restart the gRPC feature at this point to continue normal processing of gNMI transactions.
- The maximum number of total concurrent sessions for Get is 75% of the maximum concurrent calls configured. For instance, if the MTX concurrent calls are configured to 16, the maximum number of total concurrent sessions for Get will be 12.
- The combined number of concurrent sessions for Get and Set is the currently configured max gNMI concurrent-1. For instance, if gnmi concurrent calls are configured to 16, the maximum number of total concurrent sessions for Get and Set will be 15.

Set

About Set

The Set RPC is used by a client to change the configuration of the device. The operations, which may be applied to the device data, are (in order) delete, replace, and update. All operations in a single Set request are

treated as a transaction, meaning that all operations are successful or the device is rolled-back to the original state. The Set operations are applied in the order that is specified in the SetRequest. If a path is mentioned multiple times, the changes are applied even if they overwrite each other. The final state of the data is achieved with the final operation in the transaction. It is assumed that all paths specified in the SetRequest::delete, replace, update fields are CONFIG data paths and writable by the client.

For detailed information on the Set operation, refer to the Modifying State section of the gNMI Specification https://github.com/openconfig/reference/blob/1cf43d2146f9ba70abb7f04f6b0f6eaa504cef05/rpc/gnmi/gnmi-specification.md.

Guidelines and Limitations for Set

The following are guidelines and limitations for Set:

- SetRequest.encoding supports only JSON.
- A single request cannot have both OpenConfig (OC) YANG and device YANG paths. A request must have only OC YANG paths or device YANG paths, but not both.
- Subscribe supports the model `Cisco-NX-OS-syslog-oper`.
- The following optional items are not supported:
  - Path prefix
  - Path alias
  - Wildcards in path
- A single SetRequest can have up to 20 paths.
- The maximum gRPC receive buffer size is set to 8 MB.
- The combined number of concurrent sessions for Get and Set is the currently configured max gNMI concurrent-1. For instance, if gNMI concurrent calls are configured to 16, the maximum number of total concurrent sessions for Get and Set will be 15.
- For the Set::Delete RPC, an MTX log message warns if the configuration being operated on may be too large:

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

Subscribe

Guidelines and Limitations for Subscribe

Following are the guidelines and limitations for Subscribe:

- If you configure a routing-policy **prefix-list** using the CLI and request gNMI Subscription for the routing-policy OpenConfig model, it is not supported. For example, when you attempt to subscribe an OpenConfig routing policy with a preexisting CLI configuration like the following, it returns empty values due to the 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
Using the example paths,
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
```

- Beginning with Cisco NX-OS Release 9.3(3), Subscribe supports the OpenConfig model.
- For information about supported platforms, see the Nexus Switch Platform Matrix.
- The gNMI feature supports Subscribe and Capability RPCs.
- The feature supports JSON and gnmi.proto encoding. The feature does not support protobuf.any encoding.
- Each gNMI message has a maximum size of 12 MB. If the amount of collected data exceeds the 12 MB maximum, the collected data is dropped.

  You can avoid this situation by creating more focused subscriptions that handle smaller, more granular data-collection sets. So, instead of subscribing to one higher-level path, create multiple subscriptions for different, lower-level parts of the path.
- All paths within the same subscription request must have the same sample interval. If the same path requires different sample intervals, create multiple subscriptions.
- The feature does not support a path prefix in the Subscription request, but the Subscription can contain an empty prefix field.
- The feature supports Cisco DME and Device YANG data models.
- The gRPC process that supports gNMI uses the HIGH_PRIO cgroup, which limits the CPU usage to 75% of CPU and memory to 1.5 GB.
- The **show grpc gnmi** command has the following considerations:
  - The commands are not XMLized in this release.
  - The gRPC agent retains gNMI calls for a maximum of 1 hour after the call has ended.
  - If the total number of calls exceeds 2000, the gRPC agent purges ended calls based an internal cleanup routine.

The gRPC server runs in the management VRF. As a result, the gRPC process communicates only in this VRF forcing the management interface to support all gRPC calls.

gRPC functionality now includes the default VRF for a total of 2 gRPC servers on each Cisco Nexus 9000 switch. You can run one gRPC server in each VRF, or run only one gRPC server in the management VRF. Supporting a gRPC in the default VRF adds flexibility to offload processing gRPC calls from the management VRF, where significant traffic load might not be desirable.

If two gRPC servers are configured, be aware of the following:

- VRF boundaries are strictly enforced, so each gRPC server processes requests independent of the other, and requests do not cross between VRFs.
- The two servers are not HA or fault tolerant. One gRPC server does not back up the other, and there is no switchover or switchback between them.
- Any limits for the gRPC server are per VRF.

gNMI Payload

gNMI uses a specific payload format to subscribe to:

- DME Streams
- YANG Streams

Subscribe operations are supported with the following modes:

- ONCE: Subscribe and receive data once and close session.
- POLL: Subscribe and keep session open, client sends poll request each time data is needed.
- STREAM: Subscribe and receive data at specific cadence. The payload accepts values in nanoseconds 1 second = 1000000000.
- ON_CHANGE: Subscribe, receive a snapshot, and only receive data when something changes in the tree.
- TARGET_DEFINED: Determines the best type of subscription that can be created.

Setting modes:

- Each mode requires 2 settings, inside sub and outside sub
- 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 will be send in JSON format.
- PROTO = Stream will be sent in protobuf.any format.

Sample gNMI Payload for DME Stream



Note Different clients have their own input format.

```
{
    "SubscribeRequest":
    [
        {
            "_comment" : "ONCE request",
            "_delay" : 2,
            "subscribe":
            {
                "subscription":
                [
                    {
                        "_comment" : "1st subscription path",
                        "path":
                        {
                            "origin": "DME",
                            "elem":
                            [
                                {
                                    "name": "sys"
                                },
                                {
                                    "name": "bgp"
                                 }
                             ]
                        },
                        "mode": "SAMPLE"
                    }
                ],
                "mode": "ONCE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "_comment" : "1st module",
                        "name": "DME",
                        "organization": "Cisco Systems, Inc.",
                        "version": "1.0.0"
                    }
                ],
                "encoding": "JSON"
            }
        }
    ]
}
```

Sample gNMI Payload YANG Stream

```
{
    "SubscribeRequest":
    [
        {
            "_comment" : "ONCE request",
            "_delay" : 2,
            "subscribe":
            {
```

```
                "subscription":
                [
                    {
                        "_comment" : "1st subscription path",
                        "path":
                        {
                            "origin": "device",
                            "elem":
                            [
                                {
                                    "name": "System"
                                },
                                {
                                    "name": "bgp-items"
                                }
                            ]
                        },
                                                    "mode": "SAMPLE"
                    }
                ],
                "mode": "ONCE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "_comment" : "1st module",
                        "name": "Cisco-NX-OS-device",
                        "organization": "Cisco Systems, Inc.",
                        "version": "0.0.0"
                    }
                ],
                "encoding": "JSON"
            }
        }
    ]
}
```

Sample Openconfig Payload

```
{
    "SubscribeRequest":
    [
        {
            "_comment" : "STREAM request",
            "_delay"   : 2,
            "subscribe":
            {
                "subscription":
                [
                    {
                        "_comment" : "1st subscription path",
                        "path":
                        {
                            "origin": "openconfig",
                            "elem":
                            [
                                {
                                    "name": "interfaces"
                                }
                            ]
                        },
                        "mode": "SAMPLE",
                        "sample_interval": 10000000000
                    }
```

```
                ],
                "mode": "ONCE",
                "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "_comment" : "1st module",
                        "name": "openconfig-interfaces",
                        "organization": "OpenConfig working group",
                        "version": "0.8.1"
                    }
                ],
                "encoding": "JSON"
            }
        }
    ]
}
```

Streaming Syslog

About Streaming Syslog for gNMI

gNMI Subscribe is a new way of monitoring the network as it provides a real-time view of what's going on in your system by pushing the structured data as per gNMI Subscribe request.

Beginning with the Cisco NX-OS Release 9.3(3), support is added for gNMI Subscribe functionality.

gNMI Subscribe Support Detail

- Syslog-oper model streaming
  - stream_on_change

This feature applies to Cisco Nexus 9000 Series switches with 8 GB or more of memory.

Guidelines and Limitations for Streaming Syslog - gNMI

The following are guidelines and limitations for Streaming Syslog:

- An invalid syslog is not supported. For example, a syslog with a filter or query condition
- Only the following paths are supported:
  - Cisco-NX-OS-Syslog-oper:syslog
  - Cisco-NX-OS-Syslog-oper:syslog/messages
- The following modes are not supported:
  - Stream sample
  - POLL
- A request must be in the YANG model format.
- You can use the internal application or write your own application.

- The payload comes from the controller and gNMI sends a response.
- Encoding formats are JSON and PROTO.

Syslog Native YANG Model

The YangModels are located here.



Note The time-zone field is set only when the **clock format show-timezone syslog** is entered. By default, it's not set, therefore the time-zone field is empty.

```
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 Example

The following is an example of a Subscribe request:

```
{
    "SubscribeRequest":
    [
        {
            "_comment" : "STREAM request",
            "_delay"   : 2,
            "subscribe":
            {
                "subscription":
                [
                    {
                        "_comment" : "1st subscription path",
                        "path":
                        {
                            "origin": "syslog-oper",
                            "elem":
                              [
                                {
                                    "name": "syslog"
                                },
                                {
                                    "name":"messages"
                                }
                            ]
```

```
                        },
                        "mode": "ON_CHANGE"
                    }
                ],
                "mode": "ON_CHANGE",
              "allow_aggregation" : false,
                "use_models":
                [
                    {
                        "_comment" : "1st module",
                        "name": "Cisco-NX-OS-Syslog-oper",
                        "organization": "Cisco Systems, Inc.",
                        "version": "0.0.0"
                    }
                ],
                "encoding":"JSON"
            }
        }
    ]
}
```

Sample PROTO Output

This is a sample of PROTO output.

```
############################

[Subscribe]--------------------------------

### Reading from file ' /root/gnmi-console/testing_bl/stream_on_change/OC_SYSLOG.json '

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: "message-id"
}
}
val {
uint_val: 529
}
}
update {
path {
elem {
name: "node-name"
}
}
val {
string_val: "task-n9k-1"
}
}
update {
path {
elem {
name: "message-name"
}
}
val {
string_val: "VSHD_SYSLOG_CONFIG_I"
}
}
update {
path {
elem {
```

```
name: "text"
}
}
val {
string_val: "Configured from vty by admin on console0"
}
}
update {
path {
elem {
name: "group"
}
}
val {
string_val: "VSHD"
}
}
update {
path {
elem {
name: "category"
}
}
val {
string_val: "VSHD"
}
}
update {
path {
elem {
name: "time-of-day"
}
}
val {
string_val: "Nov 21 2019 14:26:40"
}
}
update {
path {
elem {
name: "time-zone"
}
}
val {
string_val: ""
}
}
update {
path {
elem {
name: "time-stamp"
}
}
val {
uint_val: 1574375200000
}
}
update {
path {
elem {
name: "severity"
}
}
val {
```

```
uint_val: 5
}
}
}

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

-

Sample JSON Output

This is a sample JSON output.

```
[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\":\"\"} ]] } ]"
```

```
}
}
}

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

Troubleshooting

Gathering TM-Trace Logs

```
1. tmtrace.bin -f gnmi-logs gnmi-events gnmi-errors following are available
2. Usage:

bash-4.3# tmtrace.bin -d gnmi-events | tail -30 Gives the last 30
}
}
}
[06/21/19 15:58:38.969 PDT f8f 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id: 0,
sub_id_str: 2329, dc_start_time: 0, length: 124, sync_response:1
[06/21/19 15:58:43.210 PDT f90 3133] [3621780288][tm_ec_yang_data_processor.c:93] TM_EC:
[Y] Data received for 2799743488: 49
{
"cdp-items" : {
"inst-items" : {
"if-items" : {
"If-list" : [
{
"id" : "mgmt0",
"ifstats-items" : {
"v2Sent" : "74",
"validV2Rcvd" : "79"
}
}
]
}
}
}
}
[06/21/19 15:58:43.210 PDT f91 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id: 0,
sub_id_str: 2329, dc_start_time: 0, length: 141, sync_response:1
[06/21/19 15:59:01.341 PDT f92 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/intf-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157935518, length: 3063619, sync_response:0
[06/21/19 15:59:03.933 PDT f93 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/cdp-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157940881, length: 6756, sync_response:0
[06/21/19 15:59:03.940 PDT f94 3133] [3981658944][tm_transport_internal.c:43] dn:
Cisco-NX-OS-device:System/lldp-items, sub_id:
4091, sub_id_str: , dc_start_time: 1561157940912, length: 8466, sync_response:1
bash-4.3#
```

Gathering MTX-Internal Logs

```
1. Modify the following file with below /opt/mtx/conf/mtxlogger.cfg
```

```
<config name="nxos-device-mgmt">
  <container name="mgmtConf">
    <container name="logging">
      <leaf name="enabled" type="boolean" default="false">true</leaf>
      <leaf name="allActive" type="boolean" default="false">true<
/leaf>
      <container name="format">
        <leaf name="content" type="string" default="$DATETIME$
$COMPONENTID$ $TYPE$: $MSG$">$DATETIME$ $COMPONENTID$ $TYPE$
$SRCFILE$ @ $SRCLINE$ $FCNINFO$:$MSG$</leaf>
          <container name="componentID">
         <leaf name="enabled" type="boolean" default="true"></leaf>
          </container>
          <container name="dateTime">
         <leaf name="enabled" type="boolean" default="true"></leaf>
         <leaf name="format" type="string" default="%y%m%d.%H%M%S"><
/leaf>
           </container>
           <container name="fcn">
          <leaf name="enabled" type="boolean" default="true"></leaf>
          <leaf name="format" type="string"
default="$CLASS$::$FCNNAME$($ARGS$)@$LINE$"></leaf>
           </container>
      </container>
      <container name="facility">
          <leaf name="info" type="boolean" default="true">true</leaf>
          <leaf name="warning" type="boolean" default="true">true<
/leaf>
          <leaf name="error" type="boolean" default="true">true</leaf>

Note: Beginning with Cisco NX-OS Release 9.3(4), the following default configuration is
changed from
      default true to false. To investigate an issue which requires the debug messages,
edit
      the following configuration and toggle it to true.

          <leaf name="debug" type="boolean" default="false">true<
/leaf>
        </container>
        <container name="dest">
          <container name="console">
            <leaf name="enabled" type="boolean" default="false">true<
/leaf>
          </container>
          <container name="file">
         <leaf name="enabled" type="boolean" default="false">true<
/leaf>
    <leaf name="name" type="string" default="mtx-internal.log"><
/leaf>


        <leaf name="location" type="string" default="./mtxlogs">
/volatile</leaf>
           <leaf name="mbytes-rollover" type="uint32" default="10"
>50</leaf>
           <leaf name="hours-rollover" type="uint32" default="24"
>24</leaf>
           <leaf name="startup-rollover" type="boolean" default="
false">true</leaf>
          <leaf name="max-rollover-files" type="uint32" default="10"
>10</leaf>
        </container>
      </container>
      <list name="logitems" key="id">
```

```
        <listitem>
                <leaf name="id" type="string">*</leaf>
                    <leaf name="active" type="boolean" default="false"
>false</leaf>
         </listitem>
          <listitem>
                 <leaf name="id" type="string">MTX-EvtMgr</leaf>
                     <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
               <leaf name="id" type="string">TM-ADPT</leaf>
                   <leaf name="active" type="boolean" default="true"
>false</leaf>
        </listitem>
        <listitem>
              <leaf name="id" type="string">TM-ADPT-JSON</leaf>
                  <leaf name="active" type="boolean" default="true"
>false</leaf>
        </listitem  >
        <listitem>
                <leaf name="id" type="string">SYSTEM</leaf>
                    <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
                <leaf name="id" type="string">LIBUTILS</leaf>
                     <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
                <leaf name="id" type="string">MTX-API</leaf>
                    <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
         <listitem>
                 <leaf name="id" type="string">Model-*</leaf>
                     <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
                <leaf name="id" type="string">Model-Cisco-NX-OS-
device</leaf>
                 <leaf name="active" type="boolean" default="true"
>false</leaf>
        </listitem>
        <listitem>
                 <leaf name="id" type="string">Model-openconfig-bgp<
/leaf>
                     <leaf name="active" type="boolean" default="true"
>false</leaf>
        </listitem>
        <listitem>
               <leaf name="id" type="string">INST-MTX-API</leaf>
                   <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
                 <leaf name="id" type="string">INST-ADAPTER-NC</leaf>
                     <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
               <leaf name="id" type="string">INST-ADAPTER-RC</leaf>
```

```
                    <leaf name="active" type="boolean" default="true"
>true</leaf>
        </listitem>
        <listitem>
               <leaf name="id" type="string">INST-ADAPTER-GRPC</leaf>
                   <leaf name="active" type="boolean" default="true"
>true</leaf>
         </listitem>
      </list>
    </container>
  </container>
</config>

2. Run "no feature grpc" / "feature grpc"
3. The /volatile directory houses the mtx-internal.log, the log rolls over time so be sure
 to grab what you need before then.

bash-4.3# cd /volatile/
bash-4.3# cd /volatile -al
total 148
drwxrwxrwx 4 root root 340 Jun 21 15:47 .
drwxrwxr-t 64 root network-admin 1600 Jun 21 14:45 ..
-rw-rw-rw- 1 root root 103412 Jun 21 16:14 grpc-internal-log
-rw-r--r-- 1 root root 24 Jun 21 14:44 mtx-internal-19-06-21-14-46-21.log
-rw-r--r-- 1 root root 24 Jun 21 14:46 mtx-internal-19-06-21-14-46-46.log
-rw-r--r-- 1 root root 175 Jun 21 15:11 mtx-internal-19-06-21-15-11-57.log
-rw-r--r-- 1 root root 175 Jun 21 15:12 mtx-internal-19-06-21-15-12-28.log
-rw-r--r-- 1 root root 175 Jun 21 15:13 mtx-internal-19-06-21-15-13-17.log
-rw-r--r-- 1 root root 175 Jun 21 15:13 mtx-internal-19-06-21-15-13-42.log
-rw-r--r-- 1 root root 24 Jun 21 15:13 mtx-internal-19-06-21-15-14-22.log
-rw-r--r-- 1 root root 24 Jun 21 15:14 mtx-internal-19-06-21-15-19-05.log
-rw-r--r-- 1 root root 24 Jun 21 15:19 mtx-internal-19-06-21-15-47-09.log
-rw-r--r-- 1 root root 24 Jun 21 15:47 mtx-internal.log
-rw-rw-rw- 1 root root 355 Jun 21 14:44 netconf-internal-log
-rw-rw-rw- 1 root root 0 Jun 21 14:45 nginx_logflag
drwxrwxrwx 3 root root 60 Jun 21 14:45 uwsgipy
drwxrwxrwx 2 root root 40 Jun 21 14:43 virtual-instance
bash-4.3#.
```