gNMI - gRPC Network Management Interface

This chapter contains the following topics:

About gNMI

The gNMI protocol is an RPC-based Network Management Interface that Google created. gNMI sits on top of gRPC (Google Remote Procedure Call) messaging protocol, which acts as a transport protocol.

Cisco NX-OS supports gNMI for dial-in subscription to telemetry applications running on the Cisco Nexus 3400-S platform 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.

Cisco NX-OS Release 9.3(1) and later supports gNMI version 0.5.0. Cisco NX-OS Release 9.3(1) and later supports the following parts of gNMI version 0.5.0:

Table 1: Supported gNMI RPCs

gNMI RPC	Supported?
Get	No

gNMI RPC	Supported?
Set	No
Capabilities	Yes
Subscribe	Yes

gNMI RPC and SUBSCRIBE

The NX-OS 9.3(1) release supports gNMI version 0.5.0. Cisco NX-OS Release 9.3(1) supports the following parts of gNMI version 0.5.0.

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.
	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	No	

Optional SUBSCRIBE Flags

For the SUBSCRIBE option, some optional flags are available that modify the response to the options listed in the table. In release 9.3(1), 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
- heart-beat interval
- prefix
- qos
- suppress_redundant

Guidelines and Limitations for gNMI

Following are the guidelines and limitations for gNMI:

- Beginning with Cisco NX-OS Release 9.3(3), if you have configured a custom gRPC certificate, upon entering the **reload ascii** command the configuration is lost. It will revert to the default day-1 certificate. After entering the **reload ascii** command, the switch will reload. Once the switch is up again, you need to reconfigure the gRPC custom certificate.
- The gNMI feature supports the Subscribe and Capability gNMI 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 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 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 on the 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 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 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. 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.

Configuring gNMI

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

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	**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.
		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.

	Command or Action	Purpose
Step 4	(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.

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.

```
#show grpc gnmi service statistics

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

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

Cert notBefore : Mon Jan 27 15:34:08 PDT 2020
Cert notAfter  : Tue Jan 26 15:34:08 PDT 2021

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
```

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.

Note After the certificate expires, there are two ways to have the key/certificate to regenerate:

- Reload the switch.
- Manually delete the key/certificate in the `/opt/mtx/etc` folder and enter the **no feature grpc** and **feature grpc** commands.

Generating Key/Certificate Examples

Typically, there are two possible scenarios:

1. The server and client can use a Self-Signed Certificate. Self-Signed Certificates are less secure and are not to be used in production environments.
   a. Generating self-signed certificates on the switch:

```
1. Generating self-signed certificates on the switch:
        a. Login to the Bash shell:
        b. cd to the location where you want to store the key/cert
       c. switch# openssl req -x509 -newkey rsa:2048 -keyout self_sign2048.key -out
 self_sign2048.pem -days 365 -nodes
               Generating a 2048 bit RSA private key
                ..........................................
                ........
                writing new private key to 'self_sign2048.key'
                -----
                You are about to be asked to enter information that will be
incorporated
                into your certificate request.
                What you are about to enter is what is called a Distinguished Name
or a DN.
                There are quite a few fields but you can leave some blank
                For some fields there will be a default value,
                If you enter '.', the field will be left blank.
                -----
                Country Name (2 letter code) [AU]:US
                string is too long, it needs to be less than 2 bytes long
                Country Name (2 letter code) [AU]:US
                State or Province Name (full name) [Some-State]:CA
                Locality Name (eg, city) []:San Jose
                Organization Name (eg, company) [Internet Widgits Pty Ltd]:Cisco
Systems
```

```
                Organizational Unit Name (eg, section) []:
                Common Name (e.g. server FQDN or YOUR name) []:benton.cisco.com
                Email Address []:
                switch#
                switch# ls -al | grep self
                -rw-r--r-- 1 root root 912 Jun 6 15:16 self_sign2048.key
                -rw-r--r-- 1 root root 952 Jun 6 15:16 self_sign2048.pem
                switch#

NOTE: Steps 2 and 3 are need only with Cisco NX-OS Release 9.3(2) and earlier.
2. Modify /etc/mtx.conf file. The mtx.conf file is not persistent after reloads but
 persistent through "no feature grpc" & "feature grpc"
3. Under the "grpc" section in the file add the key and cert location
   [grpc]
   key = /bootflash/self_sign2048.key
   cert = /bootflash/self_sign2048.pem
4. Go to cli and run
        a. no feature grpc
        b. feature grpc
        c. show grpc internal gnmi service statistics
                1. Verify that the new cert and key are used by grpc process.

NOTE: If you upgraded to Cisco NX-OS Release 9.3(3) or later, edit /etc/mtx.conf.user
 and remove the entries in Step 2.
If you downgraded from Cisco NX-OS Release 9.3(3) or later to a previous release,
you need to edit /etc/mtx.conf.user and add the entries in Step 5b.
5. Optional: If you want the changes to be persistent across reloads
        a. Create a new file /etc/mtx.conf.user
        b. Add the following
        [grpc]
        key = /bootflash/self_sign2048.key
        cert = /bootflash/self_sign2048.pem
        c. To get the mtx.conf file to accept the changes either
                1. Install, activate or de-activate a MTX RPM
                2. Or reload the box
```

2. Generating a Trusted Certificate Example.

```
On any Linux server, follow the steps below.
Step 1: Create a TLS directory, and then navigate to it:
mkdir -p TLS
cd TLS
```

```
Step 2: Create three directories under mypersonalca and two prerequisite files:
mkdir -p mypersonalca/certs
mkdir -p mypersonalca/private
mkdir -p mypersonalca/crl
echo "01" > mypersonalca/serial
touch mypersonalca/index.txt
```

```
Step 3: Create the CA configuration file (ca.conf). The following is an example of the
contents of the ca.conf file:

[ ca ]
default_ca = CA_default
[ CA_default ]
dir = mypersonalca
serial = $dir/serial
database = $dir/index.txt
new_certs_dir = $dir/newcerts
certs = $dir/certs
```

```
certificate = $certs/ca.pem
private_key = $dir/private/ca.key
crl_dir = $dir/crl
default_days = 365
default_crl_days = 30
default_md = sha1
preserve = no
email_in_dn = no
nameopt = default_ca
certopt = default_ca
policy = policy_match
copy_extensions = copy
[ policy_match ]
countryName = match
stateOrProvinceName = match
organizationName = match
organizationalUnitName = optional
commonName = supplied
emailAddress = optional
[ req ]
default_bits = 2048 # Size of keys
default_keyfile = example.key # name of generated keys
default_md = sha1 # message digest algorithm
string_mask = nombstr # permitted characters
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_req
[ req_distinguished_name ]
# Variable name Prompt string
#---------------------- ----------------------------------
0.organizationName = Organization Name (company)
organizationalUnitName = Organizational Unit Name (department, division)
emailAddress = Email Address
emailAddress_max = 40
localityName = Locality Name (city, district)
stateOrProvinceName = State or Province Name (full name)
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
commonName = Common Name (hostname, IP, or your name)
commonName_max = 64
# Default values for the above, for consistency and less typing.
# Variable name Value
#------------------------------ ------------------------------
commonName_default = www.cisco.com
0.organizationName_default = Cisco
localityName_default = San Jose
stateOrProvinceName_default = CA
countryName_default = US emailAddress_default = webmaster@cisco.com
organizationalUnitName_default = NDB
[ v3_ca ]
basicConstraints = CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
[ v3_req ]
# Extensions to add to a certificate request
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
# Some CAs do not yet support subjectAltName in CSRs.
# Instead the additional names are form entries on web
# pages where one requests the certificate...
subjectAltName = @alt_names
[alt_names]
IP.1 = 1.1.1.1
```

```
IP.2 = 2.2.2.2
IP.3 = 3.3.3.3
IP.4 = 4.4.4.4
[ server ]
# Make a cert with nsCertType set to "server"
basicConstraints=CA:FALSE
nsCertType = server
nsComment = "OpenSSL Generated Server Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always
[ client ]
# Make a cert with nsCertType set to "client"
basicConstraints=CA:FALSE
nsCertType = client
nsComment = "OpenSSL Generated Client Certificate"
subjectKeyIdentifier=hash
authorityKeyIdentifier=keyid,issuer:always

Step 4: Create the TLS certificate file.
· Generate the TLS private key and Certification Authority (CA) files by entering running:

  'openssl req -x509 -nodes -days 3650 -newkey rsa:2048 -out mypersonalca/certs/ca.pem
-outform PEM -keyout mypersonalca/private/ca.key'
This step generates the TLS private key in PEM format with a key length of 2048 bits,
and the CA file

NOTE: The field values entered for this cert should be different from the ones entered
for server certificates below. These are identity attributes and should be different for
 each cert.
```

Creating a PEM server certificate signed by the CA.

```
Create a Certificate Signing Request (CSR) and client key
cd TLS ; mkdir server-certs ; cd server-certs
openssl genrsa -des3 -out server.key 4096
openssl req -new -key server.key -out server.csr
NOTE: The field values entered for this cert should be different from the ones entered
for CA cert above. These are identity attributes and should be different for each cert.

Sign a CSR to client certificate by CA
openssl x509 -req -days 365 -in server.csr -CA ../mypersonalca/certs/ca.pem -CAkey
../mypersonalca/private/ca.key -set_serial 02 -out server.crt
NOTE: The -set_serial value should be different for each new certificate generated by a
 CA!

---------------------------------------------------------------------------------------
Server certificate to be sent from client-side for GRPC:

The server certificate that is sent from client-side for GRPC, must have the entire
chain, i.e., the CA certificate AND the identity certificate. For this, the CA cert and
 identity cert files must be bundled into one to form a certificate chain.

  cd server-certs
  cat ../mypersonalca/certs/ca.pem server.crt > server-cert-chain.pem

---------------------------------------------------------------------------------------
Creating a PKCS12 cert file from PEM for importing the certificate to the switch:

   cd server-certs
   openssl pkcs12 -export -out server.pfx -inkey server.key -in server.crt -certfile
../mypersonalca/certs/ca.pem -password pass:cisco123
For the CLI in section 1.1:

crypto ca import mytrustpoint pkcs12 <certfile> <password>,

<certfile> would be server.pfx
```

```
<password> would be cisco123
```

Verifying gNMI

To verify the gNMI configuration, enter the following command:

Command	Description

show grpc gnmi service statistics
 	
`==========`
`gRPC Endpoint`
`==========`
 	
`Vrf : management`
`Server address : [::]:50051`
 	
`Cert notBefore : Thu Mar 12 13:37:49 PDT 2020`
`Cert notAfter : Fri Mar 13 13:37:49 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`
	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.

Clients

There are available clients for gNMI subscription. One such client is located at https://github.com/influxdata/telegraf/tree/master/plugins/inputs/cisco_telemetry_gnmi.

Sample DME Subscription - JSON Encoding

```
gnmi-console --host <iip> --port 50051 -u <user> -p <pass> --tls --
operation=Subscribe --rpc bl_paylod/once/01_subscribe_bgp_dme.json

[Subscribe]--------------------------------
### Reading from file ' bl_paylod/once/01_subscribe_bgp_dme.json '

### 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: ONCE
use_models {
name: "DME"
organization: "Cisco Systems, Inc."
version: "1.0.0"
}
}

Received response 1 --------------------------
update {
timestamp: 1549061991079
update {
path {
elem {
name: "sys"
}
elem {
name: "bgp"
}
}
\"\" } } } ] } } ] } } ] } } ] } } }"
}
duplicates: 1093487956
}
}
/Received -------------------------------------
Received response 2 --------------------------
sync_response: true
/Received -------------------------------------
(_gnmi) [root@tm-ucs-1 gnmi-console]#
```

-

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]#
```

Subscribe

Guidelines and Limitations for Subscribe

Following are the guidelines and limitations for Subscribe:

- 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 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.

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

Origin

- DME: Subscribing to DME model
- device: Subscribing to YANG 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"
                }
            }
        ]
}
```

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:

- 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 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

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: "0.5.0"

hostname user$
```

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>
          <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#.
```

Innovium Path Telemetry

About Innovium Path Telemetry

Innovium Path Telemetry (IPT) is a telemetry feature which makes a truncated copy of the original packet and adds IPT metadata at each node that has IPT enabled.

- Source: At this node a copy of the original packet is made and IPT probe-marker, base header and hop information are added after the original packets L2-L4 headers.
- Transit (1 or more): At this node, IPT packets are identified based on the probe-marker value and another hop information is added after the last node hop information. There can be multiple transit nodes for an IPT packet.
- Sink: This is the last node, and here hop information is added to all IPT packets based on the probe-marker value. Collector headers are added at the beginning of the packet and sent to the collector.

Guidelines and Limitations for Innovium Path Telemetry

Innovium Path Telemetry (IPT) has the following configuration guidelines and limitations:

- On the sink node, IPT packets are accounted for on the sink interface queue besides being accounted for on the collector interface queue.

- Egress remarking of Innovium Path Telemetry (IPT) packets is not supported. Egress remarking of original packets should not be used with IPT since IPT data cannot be guaranteed in this case.
- SPAN and ERSPAN cannot monitor IPT packets.
- Innovium Path Telemetry is supported on Cisco Nexus 3408-S and 3432D-S switches.
- For Innovium Path Telemetry to work correctly, TCAM carving for ingress and egress is required.
- Only unicast, non-fragmented TCP and UDP packets are supported.
- Flow of Interest (FoI) is supported.
- The FoI ACL is shared with buffer latency and buffer drop features.
- Deny ACEs within the ACL are not supported.
- A maximum of one monitor can be applied.
- IPT/buffer drop and buffer latency share same ACL. If buffer drop and/or buffer latency have FoI with action telemetry_path, even though FoI is not applied to IPT, filter action happens for IPT as well.

Configuring Flow of Interest ACL for IPT

This procedure describes how to configure a flow of interest.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enter global configuration mode.
Step 2	**ip access-list** *list-name* **Example:** `switch(config)# ip access-list ipt-foi-v4`	Configure the access list.

Step 3	*seq-number* **{permit | deny}** *protocol {source-ip-prefix | source-ip-mask} {destination-ip-prefix | destination-ip-mask}* **telemetry_path**
	Example:
	`switch(config-acl)# 10 permit ip 10.1.1.1/32 11.1.1.1/32 telemetry_path`
	`switch(config-acl)# 10 permit ipv6 10:1:1::/64 20:1:1::/64 telemetry_path`
		Specify packets to forward. IPT enabled. The value of *seq-number* is from 1 to 4294967295.

Step 4	*seq-number* **{permit | deny}** *protocol {source-ip-prefix | source-ip-mask} {destination-ip-prefix | destination-ip-mask}* **telemetry_path**	Specify packets to forward. IPT enabled. The value of *seq-number* is from 1 to 4294967295.

	Command or Action	Purpose

	Example:
	`switch(config-acl)# 20 permit ip 30.1.1.2/32 30.1.1.2/32 telemetry_path`
	`switch(config-acl)# 20 permit ipv6 30:1:1::/64 30:1:1::/64 telemetry_path`	

Configuring the TCAM Region for Innovium Path Telemetry

This procedure configures a TCAM region in support of the Flow of Interest (FOI) option.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enter global configuration mode.
Step 2	**hardware access-list tcam region hw-telemetry** *size* **Example:** `switch(config)# hardware access-list tcam region hw-telemetry 128`	Configure the TCAM region for the hardware telemetry size. The values of *size* are 128 or 256.
Step 3	**copy running-config startup-config** **Example:** `switch(config)# copy running-config startup-config`	Saves the running configuration to the startup configuration.
Step 4	**reload** **Example:** `switch(config)# reload`	Reboots the switch.

Configuring the Source Node

This procedure configures the IPT probe marker, base header, and hop information.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.

	Command or Action	Purpose
Step 2	**feature hardware-telemetry** **Example:** `switch(config)# feature hardware-telemetry`	Enable hardware telemetry.
Step 3	**hardware-telemetry int-clone-md** **Example:** `switch(config)# hardware-telemetry int-clone-md`	Enable INT Clone MD configuration.
Step 4	**int-clone-md probe-marker** *pm_value*int-clone-md probe-marker **Example:** `switch(config-int-clone-md)# int-clone-md probe-marker 0x4d2`	Configure a probe marker value. For *pm_value*, the range is from 1 - 281474976710655.
Step 5	**int-clone-md source record** *src_rec* **Example:** `switch(config-int-clone-md)# int-clone-md source record src_rec1`	Define a source record.
Step 6	**interface ethernet** *slot/chassis* **Example:** `switch(config-int-clone-md-source-record)# interface ethernet 1/23`	Configure interfaces.
Step 7	**exit** **Example:** `switch(config-int-clone-md-source-record)# exit`	Exit current configuration mode.
Step 8	**int-clone-md source monitor** *src_mon* **Example:** `switch(config-int-clone-md)# int-clone-md source monitor src_mon1`	Define a source record.
Step 9	**record** *src_rec* **Example:** `switch(config-int-clone-md-source-monitor)# record src_rec1`	Add a record.
Step 10	**filter ip access-list** *ipt* **Example:** `switch(config-int-clone-md-source-monitor)# filter ip access-list ipt`	Applies the previously configured FoI ACL.

	Command or Action	Purpose
Step 11	**sampling rate** sr-value **Example:** `switch(config-int-clone-md-source-monitor)# sampling rate 1`	Specify sampling rate for INT Clone MD.
Step 12	**filter ip access-list** *access-list-name* **Example:** `switch(config-buffer-drop-monitor)# filter ip access-list ipt`	Configure the IPv4 filter.
Step 13	**filter ipv6 access-list** *access-list-name* **Example:** `switch(config-buffer-drop-monitor)# filter ipv6 access-list acl2`	Configure the IPv6 filter.
Step 14	**exit** **Example:** `switch(config-int-clone-md-source-monitor)# exit`	Exit current configuration mode.
Step 15	**int-clone-md system source monitor** *src_mon* **Example:** `switch(config-int-clone-md)# int-clone-md system source monitor src_mon1`	Specify source monitor to be applied.

Configuring the Transit Node

This procedure configures the **probe-marker** value.

Before you begin

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**feature hardware-telemetry** **Example:** `switch(config)# feature hardware-telemetry`	Enable hardware telemetry.

	Command or Action	Purpose
Step 3	**hardware-telemetry int-clone-md** **Example:** `switch(config)# hardware-telemetry int-clone-md`	Enable INT Clone MD configuration.
Step 4	**int-clone-md probe-marker** *pm_value* **Example:** `switch(config-int-clone-md)# int-clone-md probe-marker 0x4d2`	Configure a probe marker value. The range for *pm_value* is from 1 - 281474976710655.

Configuring the Sink Node

This procedure configures the hop information to all IPT packets based on the **probe-marker** value.

Before you begin

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enters global configuration mode.
Step 2	**feature hardware-telemetry** **Example:** `switch(config)# feature hardware-telemetry`	Enable hardware telemetry.
Step 3	**hardware-telemetry int-clone-md** **Example:** `switch(config)# hardware-telemetry int-clone-md`	Enable INT Clone MD configuration.
Step 4	**int-clone-md probe-marker** *pm_value* **Example:** `switch(config-int-clone-md)# int-clone-md probe-marker 0x4d2`	Configure a probe marker value. The range of *pm_value* is from 1 - 281474976710655.
Step 5	**int-clone-md sink collector** *sink_col* **Example:** `switch(config-int-clone-md)# int-clone-md sink collector sink_col`	Configure sink collector.
Step 6	**source ipv4** *ipaddr* **Example:**	Configure IP address.

	Command or Action	Purpose
	switch(config-int-clone-md-sink-collector)# **source ipv4 192.0.2.1**	
Step 7	**dscp** *dscp-val* **Example:** switch(config-int-clone-md-sink-collector)# **dscp 33**	Configure DSCP value. The range of *dscp-val* is 0-2147483647.
Step 8	**ttl** *ttl-val* **Example:** switch(config-int-clone-md-sink-collector)# **ttl 60**	Configure TTL value. The range of *ttl-val* is 1-255.
Step 9	**exit** **Example:** switch(config-int-clone-md-sink-collector)# **exit**	Exit current configuration mode.
Step 10	**int-clone-md sink record** *sink_rec* **Example:** switch(config-int-clone-md)# **int-clone-md sink record sink_rec1**	Define a sink record.
Step 11	**interface ethernet** *slot/chassis* **Example:** switch(config-int-clone-md-sink-record)# **interface Ethernet1/23**	Configure interface.
Step 12	**exit** **Example:** switch(config-int-clone-md-sink-collector)# **exit**	Exit current configuration mode.
Step 13	**int-clone-md sink monitor** *sink_mon* **Example:** switch(config-int-clone-md)# **int-clone-md sink monitor sink_mon1**	Define a sink monitor.
Step 14	**collector** *sink_col* **Example:** switch(config-int-clone-md-sink-monitor)# **collector sink_col1**	Define a sink monitor.
Step 15	**record** *sink_rec* **Example:** switch(config-int-clone-md-sink-monitor)# **record sink_rec1**	Add a record.

	Command or Action	Purpose
Step 16	**exit** **Example:** `switch(config-int-clone-md-sink-collector)# exit`	Exit current configuration mode.
Step 17	**int-clone-md system sink monitor** *sink_mon* **Example:** `switch(config-int-clone-md)# int-clone-md system sink monitor sink_mon1`	Sink monitor to be applied.

Verifying Innovium Path Telemetry

To display the Innovium Path Telemetry configuration information enter the following command:

Command	Purpose

show ipt details
`------------------------------------`
`  IPT Enabled ***** System Details`
`------------------------------------`
`Probe Marker   : 1234`
 	
`--------------------------------------`
`Source Monitor : src_mon(1) details`
`--------------------------------------`
`In use (applied to system): YES`
`V4 acl: ipt`
`V6 acl: iptv6`
`Sampling Rate: 1`
`Record Attached: src_rec`
 	
`--------------------------------------`
`Source Record : src_rec(1) details`
`--------------------------------------`
`Total interfaces under record:  1`
`Ethernet1/23`
 	
`--------------------------------------`
	Displays Innovium Path Telemetry details.

Command	Purpose

show hardware access-list tcam region
`IPV4 PACL [ifacl] size = 512`
`IPV6 PACL [ipv6-ifacl] size = 0`
`MAC PACL [mac-ifacl] size = 0`
`IPV4 VACL [vacl] size = 0`
`IPV6 VACL [ipv6-vacl] size = 0`
`MAC VACL [mac-vacl] size = 0`
`IPV4 RACL [racl] size = 256`
`IPV6 RACL [ipv6-racl] size = 0`
`Egress IPV4 RACL [e-racl] size = 0`
`Egress IPV6 RACL [e-ipv6-racl] size = 0`
`SPAN [span] size = 0`
`Ingress L2 QOS [ing-l2-qos] size = 0`
`Ingress L3/VLAN QOS [ing-l3-vlan-qos] size = 128`
`Ingress SUP [ing-sup] size = 256`
`Egress L2 QOS [egr-l2-qos] size = 0`
`Egress L3/VLAN QOS [egr-l3-vlan-qos] size = 128`
`Ingress PACL v4 & v6 [ifacl-all] size = 0`
`Ingress RACL v4 & v6 [racl-all] size = 0`
`Ingress QoS L2/L3 [ing-l2-l3-qos] size = 0`
`HW Telemetry [hw-telemetry] size = 128`
`Egress PACL v4 v6 [e-ifacl-all] size = 0`
`Egress HW Telemetry [e-hw-telemetry] size = 128`
	Displays TCAM carving.

Command	Purpose

show hardware access-list tcam region
`show hardware access-list tcam region`
`                                IPV4 PACL`
`[ifacl] size =  512`
`                           IPV6 PACL`
`[ipv6-ifacl] size =    0`
`                            MAC PACL`
`[mac-ifacl] size =    0`
`                                 IPV4 VACL`
`[vacl] size =    0`
`                           IPV6 VACL`
`[ipv6-vacl] size =    0`
`                             MAC VACL`
`[mac-vacl] size =    0`
`                                 IPV4 RACL`
`[racl] size =  256`
`                           IPV6 RACL`
`[ipv6-racl] size =    0`
`                         Egress IPV4 RACL`
`[e-racl] size =    0`
`                      Egress IPV6 RACL`
`[e-ipv6-racl] size =    0`
`                                       SPAN`
`[span] size =    0`
`                        Ingress L2 QOS`
`[ing-l2-qos] size =    0`
`              Ingress L3/VLAN QOS`
`[ing-l3-vlan-qos] size =  128`
`                           Ingress SUP`
`[ing-sup] size =  256`
`                       Egress L2 QOS`
`[egr-l2-qos] size =    0`
`                Egress L3/VLAN QOS`
`[egr-l3-vlan-qos] size =  128`
`                   Ingress PACL v4 & v6`
`[ifacl-all] size =    0`
`                    Ingress RACL v4 & v6`
`[racl-all] size =    0`
`                  Ingress QoS L2/L3`
`[ing-l2-l3-qos] size =    0`
`                        HW Telemetry`
`[hw-telemetry] size =  128`
`                    Egress PACL v4 v6`
`[e-ifacl-all] size =    0`
	Display egress TCAM carving.

show ip access lists
`show ip access-lists`
 	
`IP access list ipt`
`        10 permit ip 1.1.1.1/32 2.2.2.2/32 telemetry_path`
`        20 permit ip 3.3.3.3/32 4.4.4.4/32 telemetry_queue`
	Displays IPv4 access lists.

Command	Purpose

show ipv6 access-lists
`show ipv6 access-lists`
 	
`IPv6 access list iptv6`
`10 permit ipv6 1:1::1:1/128 2:2::2:2/128 telemetry_path`
`20 permit ipv6 3:3::3:3/128 4:4::4:4/128 telemetry_queue`
	Displays IPv6 access lists.