

Telemetry

- About Telemetry, on page 1
- Telemetry Components and Terminology, on page 1
- High Availability of the Telemetry Process, on page 2
- Licensing Requirements for Telemetry, on page 2
- Guidelines and Limitations for Telemetry, on page 3
- Configuring Telemetry Using the CLI, on page 10
- Configuring Telemetry Using the NX-API, on page 32
- Cloud Scale Software Telemetry, on page 46
- Telemetry Path Labels, on page 48
- Native Data Source Paths, on page 67
- Streaming Syslog, on page 78
- Troubleshooting Telemetry, on page 83

About Telemetry

Collecting data for analyzing and troubleshooting has always been an important aspect in monitoring the health of a network.

Cisco NX-OS provides several mechanisms such as SNMP, CLI, and Syslog to collect data from a network. These mechanisms have limitations that restrict automation and scale. One limitation is the use of the pull model, where the initial request for data from network elements originates from the client. The pull model does not scale when there is more than one network management station (NMS) in the network. With this model, the server sends data only when clients request it. To initiate such requests, continual manual intervention is required. This continual manual intervention makes the pull model inefficient.

A push model continuously streams data out of the network and notifies the client. Telemetry enables the push model, which provides near-real-time access to monitoring data.

Telemetry Components and Terminology

Telemetry consists of four key elements:

- **Data Collection** — Telemetry data is collected from the Data Management Engine (DME) database in branches of the object model specified using distinguished name (DN) paths or YANG infra in branches

of paths. The data can be retrieved periodically (frequency-based) or only when a change occurs in any object on a specified path (event-based). You can use the NX-API to collect frequency-based data.

- **Data Encoding** — The telemetry encoder encapsulates the collected data into the desired format for transporting.NX-OS encodes telemetry data in the Google Protocol Buffers (GPB) and JSON format. The GPB encoder stores data in a generic key-value format. The encoder requires metadata in the form of a compiled `.proto` file to translate the data into GPB format.
- **Data Transport** — Beginning with NX-OS Release 10.6(3), HTTP Event Collector (HEC) protocol has been enabled to support splunk with JSON encoding. NX-OS transports telemetry data using HTTP for JSON encoding and the Google remote procedure call (gRPC) protocol for GPB encoding. The gRPC receiver supports message sizes greater than 4 MB. (Telemetry data using HTTPS is also supported if a certificate is configured.)
- **Telemetry Receiver** — A telemetry receiver is a remote management system or application that stores the telemetry data. In order to receive and decode the data stream correctly, the receiver requires the `.proto` file that describes the encoding and the transport services. The encoding decodes the binary stream into a key value string pair. A telemetry `.proto` file that describes the GPB encoding and gRPC transport is available on Cisco's GitLab:https://github.com/CiscoDevNet/nx-telemetry-proto

Note In the telemetry context, gRPC refers to a specific proprietary .proto. Please don't confuse with the "gRPC Agent" which hosts the standard gNMI/gNOI services.

High Availability of the Telemetry Process

High availability of the telemetry process is supported with the following behaviors:

- **System Reload** — During a system reload, any telemetry configuration and streaming services are restored.
- **Supervisor Failover** — Although telemetry is not on hot standby, telemetry configuration and streaming services are restored when the new active supervisor is running.
- **Process Restart** — If the telemetry process freezes or restarts for any reason, configuration and streaming services are restored when telemetry is restarted.

Licensing Requirements for Telemetry

Table 1: Licensing Requirements for Telemetry

Product	License Requirement
Cisco NX-OS	Telemetry requires no license. Any feature not included in a license package is bundled with the Cisco NX-OS image and is provided at no extra charge to you. For a complete explanation of the Cisco NX-OS licensing scheme, see the *Cisco NX-OS Licensing Guide*.

Guidelines and Limitations for Telemetry

Telemetry has the following configuration guidelines and limitations:

- For information about supported platforms, see the Nexus Switch Platform Matrix.
- Cisco NX-OS releases that support the data management engine (DME) Native Model support Telemetry.
- Support is in place for the following:
  - DME data collection
  - NX-API data sources
  - Google protocol buffer (GPB) encoding over Google Remote Procedure Call (gRPC) transport
  - JSON encoding over HTTP
- The smallest sending interval (cadence) supported is five seconds for a depth of 0. The minimum cadence values for depth values greater than 0 depends on the size of the data being streamed out. Configuring any cadences below the minimum value may result in undesirable system behavior.
- Telemetry supports up to five remote management receivers (destinations). Configuring more than five remote receivers may result in undesirable system behavior.
- Telemetry can consume up to 20% of the CPU resource.
- In releases earlier than NX-OS Release 10.4(1)F, the time properties in DME config MOs are printed in local time, but the time zone offset is always 00.00. For example, the time is displayed as 2024-05-21T14:01:03.012+00:00.

  Beginning with NX-OS Release 10.5(1)F, the timestamps in DME config MOs like currentTime, modTs, and so on are displayed in the localtime-timezone_offset format. For example, the time is displayed as 2024-05-21T14:01:03.012-04:00.
- Beginning with Cisco NX-OS Release 10.4(2)F, telemetry is supported on the Cisco Nexus 93400LD-H1 platform switches.
- Beginning with Cisco NX-OS Release 10.4(3)F, Telemetry is supported on 92348GC-X.
- Beginning with Cisco NX-OS Release 10.4(2)F, telemetry is supported on the Cisco Nexus N9KC9364C-H1 platform switches.
- Beginning with Cisco NX-OS Release 10.6(1)F, telemetry is supported on the Cisco Nexus N9336C-SE1 platform switches.
- Beginning with Cisco NX-OS Release 10.6(1)F, the Securing NX-OS with Cisco Live Protect feature supports event logs that can be exported using the telemetry feature with a new sensor path for nxsecure.

Configuration Commands After Downgrading to an Older Release

After a downgrade to an older release, some configuration commands or command options can fail because the older release may not support them. When downgrading to an older release, unconfigure and reconfigure the telemetry feature after the new image comes up. This sequence avoids the failure of unsupported commands or command options.

Beginning with Cisco NX-OS Release 10.6(1)F, the DME model supports RTP flow monitoring. Telemetry configurations that reference the newly introduced RTP flow-monitoring DME operational objects are not compatible with earlier Cisco NX-OS releases. Such configurations are treated as a strict incompatibility during downgrade. Remove the affected telemetry configuration before downgrading to a release that does not support these DME objects.

The following example shows this procedure:

- Copy the telemetry configuration to a file:

```
switch# show running-config | section telemetry
feature telemetry
telemetry
destination-group 100
ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB use-chunking size 4096
sensor-group 100 path sys/bgp/inst/dom-default depth 0
subscription 600
dst-grp 100
snsr-grp 100 sample-interval 7000
switch# show running-config | section telemetry > telemetry_running_config switch# show
 file bootflash:telemetry_running_config
feature telemetry
telemetry
destination-group 100
ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB use-chunking size 4096
sensor-group 100 path sys/bgp/inst/dom-default depth 0
subscription 600
dst-grp 100
snsr-grp 100 sample-interval 7000
switch#
```

- Execute the downgrade operation. When the image comes up and the switch is ready, copy the telemetry configurations back to the switch.

```
switch# copy telemetry_running_config running-config echo-commands
`switch# config terminal`
`switch(config)# feature telemetry`
`switch(config)# telemetry`
`switch(config-telemetry)# destination-group 100`
`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB `
`switch(conf-tm-dest)# sensor-group 100`
`switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0`
`switch(conf-tm-sensor)# subscription 600`
`switch(conf-tm-sub)# dst-grp 100`
`switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000`
`switch(conf-tm-sub)# end`
Copy complete, now saving to disk (please wait)...
Copy complete. switch#
```

gRPC Error Behavior

The switch client disables the connection to the gRPC receiver if the gRPC receiver sends 20 errors. Unconfigure then reconfigure the receiver's IP address under the destination group to enable the gRPC receiver.

Errors include:

- The gRPC client sends the wrong certificate for secure connections.
- The gRPC receiver takes too long to handle client messages and incurs a timeout. Avoid timeouts by processing messages using a separate message processing thread.

Support for gRPC Chunking

- Cisco NX-OS supports gRPC chunking. For streaming to occur successfully, you must enable chunking if gRPC has to send an amount of data greater than 12 MB to the receiver.

  The gRPC user must do the gRPC chunking. The gRPC client side does the fragmentation, and the gRPC server side does the reassembly. Telemetry is still bound to memory and data can be dropped if the memory size is more than the allowed limit of 12 MB for telemetry. In order to support chunking, use the telemetry `.proto` file that is available at Cisco's GibLab, which has been updated for gRPC chunking, as described in

  The chunking size is from 64 through 4096 bytes.

  Following shows a configuration example through the NX-API CLI:

```
feature telemetry !
telemetry
destination-group 1
ip address 171.68.197.40 port 50051 protocol gRPC encoding GPB use-chunking size 4096
destination-group 2
ip address 10.155.0.15 port 50001 protocol gRPC encoding GPB use-chunking size 64
sensor-group 1 path sys/intf depth unbounded
sensor-group 2 path sys/intf depth unbounded
subscription 1
dst-grp 1
snsr-grp 1 sample-interval 10000
subscription 2
dst-grp 2 snsr-grp 2 sample-interval 15000
```

  Following shows a configuration example through the NX-API REST:

```
{
"telemetryDestGrpOptChunking": {
"attributes": {
"chunkSize": "2048",
"dn": "sys/tm/dest-1/chunking"
}
```

  The following error message appears on systems that do not support gRPC chunking, such as the Cisco MDS series switches:

```
switch# use-chunking size 200
ERROR: Operation failed: [chunking support not available]
```

- Beginning with Cisco NX-OS Release 10.6(4)M, you can use the **no grpc-retry** command in telemetry destination-host configuration mode to disable automatic gRPC retry.

  Following is a configuration example:

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 192.0.2.10 port 50007 protocol gRPC encoding GPB
switch(conf-tm-dest-host)# no grpc-retry
```

NX-API Sensor Path Limitations

NX-API can collect and stream switch information not yet in the DME using **show** commands. However, using the NX-API instead of streaming data from the DME has inherent scale limitations as outlined:

- The switch backend dynamically processes NX-API calls such as **show** commands,
- NX-API spawns several processes that can consume up to a maximum of 20% of the CPU.

- NX-API data translates from the CLI to XML to JSON.

The following is a suggested user flow to help limit excessive NX-API sensor path bandwidth consumption:

1. Check whether the **show** command has NX-API support. You can confirm whether NX-API supports the command from the VSH with the pipe option:

   ```
   show
           <command> | json
   ```

   or `show <command> | json pretty`.



Note Avoid commands that take the switch more than 30 seconds to return JSON output.

2. Refine the show command to include any filters or options.

   Avoid enumerating the same command for individual outputs; for example, show vlan id 100 , show vlan id 101 , and so on. Instead, use the CLI range options; for example, show vlan id 100-110,204 , whenever possible to improve performance.

   If only the summary or counter is needed, then avoid dumping a whole show command output to limit the bandwidth and data storage that is required for data collection.

3. Configure telemetry with sensor groups that use NX-API as their data sources. Add the show commands as sensor paths
4. Configure telemetry with a cadence of five times the processing time of the respective show command to limit CPI usage.
5. Receive and process the streamed NX-API output as part of the existing DME collection.

Telemetry VRF Support

Telemetry VRF support allows you to specify a transport VRF, which means that the telemetry data stream can egress through front-panel ports and avoid possible competition between SSH or NGINX control sessions.

You can use the **use-vrf** *vrf-name* command to specify the transport VRF.

The following example specifies the transport VRF:

The following is an example of use-vrf as a POST payload:

```
{
"telemetryDestProfile": {
"attributes": {
"adminSt": "enabled"
},
"children": [
{
"telemetryDestOptVrf": { "attributes": {
"name": "default"
}
}
}
]
}
}
```

In a given configuration, only one VRF can be used to route data for a specific IP address and port combination. The VRF that will be used is determined by the vrf configuration done at the end.

For example:

```
telemetry
  destination-group 1
    ip address 91.1.1.1 port 50007
    use-vrf default
  destination-group 2
    ip address 91.1.1.1 port 50007
    use-vrf test
  sensor-group 1
    data-source DME
    path sys/fm
  subscription 1
    dst-grp 1
    dst-grp 2
    snsr-grp 1 sample-interval 1000
```

In the above configuration, both destination-group 1 and destination-group 2 have the same destination IP address and port (91.1.1.1 50007), but different VRFs are configured (default and test). In this scenario, if destination-group 2 is configured last, the data sent to 91.1.1.1 50007 will be routed using the **test** VRF. If there are no VRF configured for a destination-group, it will implicitly use **management** VRF.

Certificate Trustpoint Support

Telemetry supports the trustpoint keyword.

The following is the command syntax:

```
switch(config-telemetry)# certificate ?
trustpoint    specify trustpoint label
WORD          .pem certificate filename (Max Size 256)
switch(config-telemetry)# certificate trustpoint
WORD          trustpoint label name (Max Size 256)
switch(config-telemetry)# certificate trustpoint trustpoint1 ?
WORD  Hostname associated with certificate (Max Size 256)
switch(config-telemetry)#certificate trustpoint trustpoint1 foo.test.google.fr
```

Destination Hostname Support

Telemetry supports the **host** keyword in destination-group command.

The following is the example for the destination hostname support:

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ?
certificate Specify certificate
host Specify destination host
ip Set destination IPv4 address
ipv6 Set destination IPv6 address
...
switch(conf-tm-dest)# host ?
A.B.C.D|A:B::C:D|WORD  IPv4 or IPv6 address or DNS name of destination
switch(conf-tm-dest)#
switch(conf-tm-dest)# host abc port 11111 ?
protocol  Set transport protocol
switch(conf-tm-dest)# host abc port 11111 protocol ?
HTTP
UDP
gRPC
```

```
switch(conf-tm-dest)# host abc port 11111 protocol gRPC ?
encoding  Set encoding format
switch(conf-tm-dest)# host abc port 11111 protocol gRPC encoding ?
Form-data    Set encoding to Form-data only
GPB          Set encoding to GPB only
GPB-compact  Set encoding to Compact-GPB only
JSON         Set encoding to JSON
XML          Set encoding to XML
switch(conf-tm-dest)# host ip address 1.1.1.1 port 2222 protocol HTTP encoding JSON
<CR>
```

Support for Node ID

Telemetry supports a custom Node ID string for a telemetry receiver through the **use-nodeid** command. By default, the host name is used, but support for a node ID enables you to set or change the identifier for the `node_id_str` of the telemetry receiver data.

You can assign the node ID through the telemetry destination profile, by using the **usenode-id** command. This command is optional.

The following example shows configuring the node ID.

```
switch(config)# telemetry
switch(config-telemetry)# destination-profile
switch(conf-tm-dest-profile)# use-nodeid test-srvr-10
switch(conf-tm-dest-profile)#
```

The following example shows a telemetry notification on the receiver after the node ID is configured.

```
Telemetry receiver:
==================================
node_id_str: "test-srvr-10"
subscription_id_str: "1" encoding_path:
"sys/ch/psuslot-1/psu" collection_id:
3896 msg_timestamp: 1559669946501
```

Use the **use-nodeid** sub-command under the **host** command. The destination level **use-nodeid** configuration precedes the global level configuration. The following example shows the command syntax:

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 172.19.216.78 port 18112 protocol http enc json
switch(conf-tm-dest-host)# use-nodeid ?
WORD  Node ID (Max Size 128)
switch(conf-tm-dest-host)# use-nodeid session_1:18112
```

The following example shows the output from the Telemetry receiver:

```
>> Message size 923
Telemetry msg received @ 23:41:38 UTC Msg Size: 11
node_id_str : session_1:18112 collection_id : 3118
data_source : DME
encoding_path : sys/ch/psuslot-1/psu collection_
start_time : 1598485314721
collection_end_time : 1598485314721
data :
```

Support for Streaming of YANG Models

Telemetry supports the YANG ("Yet Another Next Generation") data modeling language. Telemetry supports data streaming for both device YANG and OpenConfig YANG.

Support for Proxy

Telemetry supports the **proxy** keyword in the host command. The following is the command syntax:

```
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# host 172.19.216.78 port 18112 protocol http enc json
switch(conf-tm-dest-host)# proxy ?
A.B.C.D|A:B::C:D|WORD IPv4 or IPv6 address or DNS name of proxy server
<1-65535> Proxy port number, Default value is 8080 username Set proxy authentication username

password Set proxy authentication password
```

gRPC Asynchronous Mode

The gRPC asynchronous mode is available only under the **host** command. In normal stream condition, this mode allows the receivers to stream data in **mdtDialout** call without exiting or receiving **WriteDone()** call.

The following is the command syntax:

```
nxosv-1(config-telemetry)# destination-group 1
nxosv-1(conf-tm-dest)# host 172.22.244.130 port 50007 ?
nxosv-1(conf-tm-dest-host)# grpc-async ?
```

Single Telemetry Collection for Sensor Groups

Beginning with Cisco NX-OS 10.5(2)F, users can use a new CLI option **merge-subscriptions** under **telemetry** CLI to create a single telemetry collection for the sensor groups when they are included in multiple subscriptions unless a parser file is configured in any of its destination groups.

- You cannot use this configuration for event subscriptions.
- Filter file configuration under destination group and merge-subscriptions are not compatible with each other. It is not blocked in the NX-OS. The collections will happen separately for all sensor groups in that case, which is same as earlier releases.
- The sample interval for sending data for subscriptions with a larger sample interval is determined using the following formula: min_sample_interval*floor(cur_sample_interval /min_sample_interval).
  - Min_sample_interval is the minimum sample interval for a sensor group in all the subscription.
  - Cur_sample interval is the sample interval of a particular subscription for that sensor group.
- This option is not available in the earlier releases, remove this configuration before downgrading to avoid compatibility check failure.

Support for HEC Protocol

Beginning with Cisco NX-OS 10.6(3), telemetry supports HEC Protocol that users can use to stream telemetry data to a Splunk HTTP Event Collector (HEC). To configure the protocol, users need to configure a HEC token, an index name, and a specific HEC REST API endpoint URL via the CLI. If not specified, index-name defaults to "main", and the endpoint defaults to "services/collector".



Note Only JSON encoding is supported for the HEC protocol.

Configuring Telemetry Using the CLI

Configuring with CLI

The following steps enable streaming telemetry and configuring the source and destination of the data stream.

SUMMARY STEPS

1. **configure terminal**
2. **feature telemetry**
3. **feature nxapi**
4. **nxapi use-vrf management**
5. **telemetry**
6. **[no] merge-subscriptions**
7. (Optional) **certificate** *certificate_path host_URL*
8. **sensor-group** *sgrp_id*
9. **path** *sensor_path* **depth unbounded [filter-condition** *filter]* **[alias** *path_alias]*
10. (Optional) **data-source native**
11. (Optional) **path event-nxsecure**
12. **destination-group** *dgrp_id*
13. (Optional) **ip address** *ip_address* port *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol*
14. (Optional) **ipv6 address** *ipv6_address port* **port protocol** *procedural-protocol* **encoding** *encoding-protocol*
15. (Optional) **source-interface** *interface*
16. *ip_version* **address ip_address port** *portnum*
17. (Optional) **use-chunking size** *chunking_size*
18. **subscription sub_id**
19. **snsr-grp** *sgrp_id* **sample-interval** *interval*
20. **dst-grp** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal`	Enter the global configuration mode.
Step 2	**feature telemetry**	Enable the streaming telemetry feature.
Step 3	**feature nxapi**	Enable NX-API.

	Command or Action	Purpose
Step 4	**nxapi use-vrf management** **Example:** `switch(config)#` `switch(config)# nxapi use-vrf management` `switch(config)`	Enable the VRF management to be used for NX-API communication. **Note** The following warnings are seen previous to 10.2(3)F release as ACLs are able to filter only netstack packets: **Warning** Warning: Management ACLs configured will not be effective for HTTP services. Please use iptables to restrict access." **Note** Beginning with 10.2(3)F, ACLs are able to filter both netstack and kstack packets which are coming to the management vrf. The following warnings are displayed: **Warning** Warning: ACLs configured on non-management VRF will not be effective for HTTP services on that VRF."
Step 5	**telemetry** **Example:** `switch# telemetry` `switch(config-telemetry)#`	Enter configuration mode for streaming telemetry.
Step 6	**[no] merge-subscriptions** **Example:** `switch# merge-subscriptions`	Creates a single collection for all the sensor groups that are included in multiple subscriptions.
Step 7	(Optional) **certificate** *certificate_path host_URL* **Example:** `switch# certificate /bootflash/server.key localhost`	Use an existing SSL/TLS certificate. For EOR devices, the certificate also has to be copied to the standby SUP.
Step 8	**sensor-group** *sgrp_id* **Example:** `switch# sensor-group 100` `switch(config-telemetry)#`	Create a sensor group with ID srgp_id and enter sensor group configuration mode. Alphanumeric ID values are supported. The sensor group defines nodes that will be monitored for telemetry reporting.
Step 9	**path** *sensor_path* **depth unbounded [filter-condition** *filter]* **[alias** *path_alias]* **Example:** • The following command is applicable for DME, not for NX-API or YANG:	Here unbounded means include child Managed Objects (MO) in the output. So, for POLL telemetry streams, all child MO for that path and EVENT retrieves the changes made in child MO. **Note** This is applicable for data source DME paths only.

	Command or Action	Purpose
	`switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition eq(l2BD.operSt, "down")` Use the following syntax for state-based filtering to trigger only when operSt changes from up to down, with no notifications of when the MO changes. `switch(conf-tm-sensor)# path sys/bd/bd-[vlan-100] depth 0 filter-condition and(updated(l2BD.operSt),eq(l2BD.operSt,"down"))` Use the following syntax to distinguish the path on the UTR side. `switch(conf-tm-sensor)# path sys/ch/ftslot-1/ft alias ft_1` • The following command is applicable for NX-API, not for DME or YANG: `switch(conf-tm-sensor)# path "show interface" depth 0` • The following command is applicable for device YANG: `switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items` • The following commands are applicable for OpenConfig YANG: `switch(conf-tm-sensor)# path openconfig-bgp:bgp` `switch(conf-tm-sensor)# path Cisco-NX-OS-device:System/bgp-items/inst-items alias bgp_alias` • The following command is applicable for NX-API: `switch(conf-tm-sensor)# path "show interface" depth 0 alias sh_int_alias` • The following command is applicable for OpenConfig: `switch(conf-tm-sensor)# path openconfig-bgp:bgp alias oc_bgp_alias`	Add a sensor path to the sensor group. • Beginning with the Cisco NX-OS 9.3(5) release, the alias keyword is introduced. • The depth setting specifies the retrieval level for the sensor path. Depth settings of 0 - 32 , unbounded are supported. **Note** depth 0 is the default depth. NX-API-based sensor paths can only use depth 0 . If a path is subscribed for the event collection, the depth only supports 0 and unbounded. Other values would be treated as 0. • The optional filter-condition parameter can be specified to create a specific filter for event-based subscriptions. For state-based filtering, the filter returns both when a state has changed and when an event has occurred during the specified state. That is, a filter condition for the DN sys/bd/bd-[vlan] of eq(l2Bd.operSt, "down") triggers when the operSt changes, and when the DN's property changes while the operSt remains down , such as a no shutdown command is issued while the VLAN is operationally down . • For the YANG model, the sensor path format is as follows: module_name: YANG_path, where module_name is the name of the YANG model file. For example: • For device YANG: Cisco-NX-OS-device:System/bgp-items/inst-items • For OpenConfig YANG: openconfig-bgp:bgp **Note** The depth , filter-condition , and query-condition parameters are not supported for YANG currently. For the openconfig YANG models, go to https://github.com/YangModels/yang/tree/master/vendor/cisco/nx and navigate to the appropriate folder for the latest release. Instead of installing a specific model, you can install the openconfig-all RPM which has all the OpenConfig models. For example:

	Command or Action	Purpose
		install add mtx-openconfig-bgp-1.0.0.0.0-7.0.3.IHD8.1.lib32_n9000.rpm activate
Step 10	(Optional) **data-source native** **Example:** `switch(conf-tm-sensor)# data-source native`	Sets the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 11	(Optional) **path event-nxsecure** **Example:** `switch(conf-tm-sensor)# path event-nxsecure` `switch(conf-tm-sensor)# path event-history` `switch(conf-tm-sensor)# path event-monitor`	The telemetry path sensor type allows for export of log files to external receivers. This command sets the telemetry path sensor type as one of the following three paths that are event-based which include: • `event-nxsecure` – Allows for nxsecure log files to be transported to remote https server. The files are transported through https. • `event-history` – Maintains and configures historical event logs • `event-monitor` – Consolidates real-time event data streams
Step 12	**destination-group** *dgrp_id* **Example:** `switch(conf-tm-sensor)# destination-group 100`	Create a destination group and enter destination group configuration mode. Alphanumeric ID values are supported.
Step 13	(Optional) **ip address** *ip_address* port *port* **protocol** *procedural-protocol* **encoding** *encoding-protocol* **Example:** `switch(conf-tm-sensor)# ip address 171.70.55.69 port 50001 protocol gRPC encoding GPB` `switch(conf-tm-sensor)# ip address 171.70.55.69 port 50007 protocol HTTP encoding JSON`	Specify an IPv4 IP address and port to receive encoded telemetry data. **Note** gRPC is the default transport protocol. GPB is the default encoding.
Step 14	(Optional) **ipv6 address** *ipv6_address port* **port protocol** *procedural-protocol* **encoding** *encoding-protocol* **Example:** `switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8000 protocol gRPC encoding GPB` `switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8001 protocol HTTP encoding JSON` `switch(conf-tm-sensor)# ipv6 address 10:10::1 port 8002 protocol UDP encoding JSON`	Specify an IPv6 IP address and port to receive encoded telemetry data. **Note** gRPC is the default transport protocol. GPB is the default encoding.
Step 15	(Optional) **source-interface** *interface* **Example:**	Specify the telemetry source interface for a destination IP address.

	Command or Action	Purpose
	`switch(conf-tm-sensor)# source-interface vlan1500`	• Only one source interface will be supported per destination ip:port. For a destination ip:port, last configured source-interface will be used. • The source-interface configuration must be accompanied with respective vrf of that interface using the use-vrf CLI under the destination group. If use-vrf is not specified, management vrf will be used.
Step 16	*ip_version* **address ip_address port** *portnum* **Example:** For IPv4: `switch(conf-tm-dest)# ip address 1.2.3.4 port 50003` For IPv6: `switch(conf-tm-dest)# ipv6 address 10:10::1 port 8000`	Create a destination profile for the outgoing data, where ip_version is either ip (for IPv4) or ipv6 (for IPv6). When the destination group is linked to a subscription, telemetry data is sent to the IP address and port that is specified by this profile.
Step 17	(Optional) **use-chunking size** *chunking_size* **Example:** `switch(conf-tm-dest)# use-chunking size 64`	Enable gRPC chunking and set the chunking size, between 64-4096 bytes. See the section "Support for gRPC Chunking" for more information.
Step 18	**subscription sub_id** **Example:** `switch(conf-tm-dest)# subscription 100` `switch(conf-tm-sub)#`	Create a subscription node with ID and enter the subscription configuration mode. Currently sub_id only supports numeric ID values. **Note** When subscribing to a DN, check whether the DN is supported by DME using REST to ensure that events will stream.
Step 19	**snsr-grp** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000`	Link the sensor group with ID sgrp_id to this subscription and set the data sampling interval in milliseconds. An interval value of 0 creates an event-based subscription, in which telemetry data is sent only upon changes under the specified MO. An interval value greater than 0 creates a frequency-based subscription, in which telemetry data is sent periodically at the specified interval. For example, an interval value of 15000 results in the sending of telemetry data every 15 seconds.
Step 20	**dst-grp** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 100`	Link the destination group with ID dgrp_id to this subscription.

Configuring Cadence for YANG Paths

The cadence for YANG paths must be greater than the total streaming time. If the total streaming time and cadence are incorrectly configured, gathering telemetry data can take longer than the streaming interval. In this situation, you can see:

- Queues that incrementally fill because telemetry data is accumulating faster than it is streaming to the receiver.
- Stale telemetry data which is not from the current interval.

Configure the cadence to a value greater than the total streaming time.

SUMMARY STEPS

1. **show telemetry control database sensor-groups**
2. **sensor group** *number*
3. **subscription** *number*
4. **snsr-grp** *number* **sample-interval** *milliseconds*
5. **show system resources**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**show telemetry control database sensor-groups** **Example:** `switch# show telemetry control database sensor-groups` `Sensor Group Database size = 2` `---------------------------------------------------------------------------` `Row ID Sensor Group ID Sensor Group type` `Sampling interval(ms) Linked subscriptions SubID` `---------------------------------------------------------------------------` `1 2 Timer /YANG 5000` `/Running 1 1` `Collection Time in ms (Cur/Min/Max): 2444/2294/2460` `Encoding Time in ms (Cur/Min/Max): 56/55/57` `Transport Time in ms (Cur/Min/Max): 0/0/1` `Streaming Time in ms (Cur/Min/Max): 2515/2356/28403` `Collection Statistics:` `collection_id_dropped = 0` `last_collection_id_dropped = 0` `drop_count = 0` `2 1 Timer /YANG 5000` `/Running 1 1` `Collection Time in ms (Cur/Min/Max): 144/142/1471` `Encoding Time in ms (Cur/Min/Max): 0/0/1` `Transport Time in ms (Cur/Min/Max): 0/0/0` `Streaming Time in ms (Cur/Min/Max): 149/147/23548`	Calculate the total streaming time. The total streaming time is the sum of the individual current streaming times of each sensor group. Individual streaming times are displayed in Streaming time in ms (Cur) . In this example, total streaming time is 2.664 seconds (2515 milliseconds plus 149 milliseconds). Compare the configured cadence to the total streaming time for the sensor group. The cadence is displayed in sample-interval . In this example, the cadence is correctly configured because the total streaming time (2.664 seconds) is less than the cadence (5.000 seconds, which is the default).

	Command or Action	Purpose
	`Collection Statistics:` `  collection_id_dropped      = 0` `  last_collection_id_dropped = 0` `  drop_count                = 0` `switch#` `telemetry` `  destination-group 1` `    ip address 192.0.2.1 port 9000 protocol HTTP encoding JSON` `  sensor-group 1` `    data-source YANG` `    path /Cisco-NX-OS-device:System/procsys-items depth unbounded` `  sensor-group 2` `    data-source YANG` `    path /Cisco-NX-OS-device:System/intf-items/phys-items depth unbounded` `  subscription 1` `    dst-grp 1` `    snsr-grp 1 sample-interval 5000` `    snsr-grp 2 sample-interval 5000`	
Step 2	**sensor group** *number* **Example:** `switch(config-telemetry)# sensor group1`	If the total streaming time is not less than the cadence, enter the sensor group for which you want to set the interval.
Step 3	**subscription** *number* **Example:** `switch(conf-tm-sensor)# subscription 100`	Edit the subscription for the sensor group.
Step 4	**snsr-grp** *number* **sample-interval** *milliseconds* **Example:** `switch(conf-tm-sub)# snsr-grp number sample-interval 5000`	For the appropriate sensor group, set the sample interval to a value greater than the total streaming time. In this example, the sample interval is set to 5.000 seconds, which is valid because it is larger than the total streaming time of 2.664 seconds.
Step 5	**show system resources** **Example:** `switch# show system resources` `Load average:   1 minute: 0.38   5 minutes: 0.43  15 minutes: 0.43` `Processes:   555 total, 3 running` `CPU states  :   24.17% user,   4.32% kernel, 71.50% idle` `       CPU0 states:   0.00% user,   2.12% kernel,   97.87% idle` `       CPU1 states:   86.00% user,   11.00% kernel,   3.00% idle` `       CPU2 states:   8.08% user,   3.03% kernel,   88.88% idle` `       CPU3 states:   0.00% user,   1.02% kernel,   98.97% idle` `Memory usage:   16400084K total,   5861652K used,`	Check the CPU usage. If the CPU user state shows high usage, as shown in this example, your cadence and streaming value are not configured correctly. Repeat this procedure to properly configure the cadence.

	Command or Action	Purpose
	10538432K free Current memory status: OK	

Configure HEC Protocol

Procedure

The following is the command to include new types and properties for the HEC protocol. The solution involves updating the DME (Data Management Engine) model.

```
destination-group <id>
  host <ip|FQDN> port <port> protocol hec encoding json
    token <0 | 7> <token-id>
    index-name <index-name>
    endpoint <endpoint_url>
```

```
telemetry
  destination-group 1
    host 10.30.217.80 port 29092 protocol hec encoding json
      token 0 c3f63de6-9c0d-49ac-a607-a6f89a1e470e
      index main
      endpoint services/collector
  sensor-group 1
    path sys/procsys/syscpusummary/syscpuhistory-last60seconds
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
```

Configuration Examples for Telemetry Using the CLI

Configure a single telemetry DME stream

with a ten second cadence with GPB encoding.

```
switch# configure terminal
switch(config)# feature telemetry
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(config-tm-dest)# ip address 171.70.59.62 port 50051 protocol gRPC encoding GPB
switch(config-tm-dest)# source-interface vlan1500
switch(config-tm-dest)# exit
switch(config-telemetry)# sensor group sg1
switch(config-tm-sensor)# data-source DME
switch(config-tm-dest)# path interface depth unbounded query-condition keep-data-type
switch(config-tm-dest)# subscription 1
switch(config-tm-dest)# dst-grp 1
switch(config-tm-dest)# snsr grp 1 sample interval 10000
```

Show running configuration

```
switch# show running-config telemetry
telemetry
destination-group 1
ip address 171.70.59.62 port 50051 protocol gRPC encoding GPB
source-interface Vlan1500
sensor-group sg1
data-source DME
subscription 1
dst-grp 1
snsr-grp sg1 sample-interval 10000
```

Subscribe sys/bgp root MO

every 5 seconds to the destination IP 1.2.3.4 port 50003.

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

Subscribe sys/intf MO

- every 5 seconds to destination IP 1.2.3.4 port 50003
- encrypts the stream using GPB encoding verified using the test.pem.

```
switch(config)# telemetry
switch(config-telemetry)# certificate /bootflash/test.pem foo.test.google.fr
switch(conf-tm-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
switch(config-dest)# sensor-group 100
switch(conf-tm-sensor)# path sys/bgp depth 0
switch(conf-tm-sensor)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 5000
switch(conf-tm-sub)# dst-grp 100
```

Subscribe sys/cdp MO

every 15 seconds to destination IP 1.2.3.4 port 50004

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 15000
switch(conf-tm-sub)# dst-grp 100
```

Subscribe cadence-based collection of show command

every 750 seconds.

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 1
switch(conf-tm-dest)# ip address 172.27.247.72 port 60001 protocol gRPC encoding GPB
switch(conf-tm-dest)# sensor-group 1
```

```
switch(conf-tm-sensor# data-source NX-API
switch(conf-tm-sensor)# path "show system resources" depth 0
switch(conf-tm-sensor)# path "show version" depth 0
switch(conf-tm-sensor)# path "show environment power" depth 0
switch(conf-tm-sensor)# path "show environment fan" depth 0
switch(conf-tm-sensor)# path "show environment temperature" depth 0
switch(conf-tm-sensor)# path "show process cpu" depth 0
switch(conf-tm-sensor)# path "show nve peers" depth 0
switch(conf-tm-sensor)# path "show nve vni" depth 0
switch(conf-tm-sensor)# path "show nve vni 4002 counters" depth 0
switch(conf-tm-sensor)# path "show int nve 1 counters" depth 0
switch(conf-tm-sensor)# path "show policy-map vlan" depth 0
switch(conf-tm-sensor)# path "show ip access-list test" depth 0
switch(conf-tm-sensor)# path "show system internal access-list resource utilization" depth
 0
switch(conf-tm-sensor)# subscription 1
switch(conf-tm-sub)# dst-grp 1
switch(conf-tm-dest)# snsr-grp 1 sample-interval 750000
```

Subscribe event-based subscription for sys/fm

streamed only if there is a change under the sys/fm MO

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 0
switch(conf-tm-sub)# dst-grp 100
```

During operation, you can change a sensor group from frequency-based to event-based, and change event-based to frequency-based by changing the sample-interval. This example changes the sensor-group from the previous example to frequency-based. After the following commands, the telemetry application will begin streaming the sys/fm data to the destination every 7 seconds.

```
switch(config)# telemetry
switch(config-telemetry)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000
```

Subscribe to multiple sensor and destination groups

Multiple sensor groups and destinations can be linked to a single subscription. The subscription in this example streams the data for Ethernet port 1/1 to four different destinations every 10 seconds.

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 1.2.3.4 port 50005
switch(conf-tm-sensor)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001 protocol HTTP encoding JSON
switch(conf-tm-dest)# ip address 1.4.8.2 port 60003
switch(conf-tm-dest)# subscription 100
switch(conf-tm-sub)# snsr-grp 100 sample-interval 10000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200
```

A sensor group can contain multiple paths, a destination group can contain multiple destination profiles, and a subscription can be linked to multiple sensor groups and destination groups, as shown in this example.

```
switch(config)# telemetry
switch(config-telemetry)# sensor-group 100
switch(conf-tm-sensor)# path sys/intf/phys-[eth1/1] depth 0
switch(conf-tm-sensor)# path sys/epId-1 depth 0
switch(conf-tm-sensor)# path sys/bgp/inst/dom-default depth 0

switch(config-telemetry)# sensor-group 200
switch(conf-tm-sensor)# path sys/cdp depth 0
switch(conf-tm-sensor)# path sys/ipv4 depth 0

switch(config-telemetry)# sensor-group 300
switch(conf-tm-sensor)# path sys/fm depth 0
switch(conf-tm-sensor)# path sys/bgp depth 0

switch(conf-tm-sensor)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# ip address 4.3.2.5 port 50005

switch(conf-tm-dest)# destination-group 200
switch(conf-tm-dest)# ip address 5.6.7.8 port 50001

switch(conf-tm-dest)# destination-group 300
switch(conf-tm-dest)# ip address 1.2.3.4 port 60003

switch(conf-tm-dest)# subscription 600
switch(conf-tm-sub)# snsr-grp 100 sample-interval 7000
switch(conf-tm-sub)# snsr-grp 200 sample-interval 20000
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 200

switch(conf-tm-dest)# subscription 900
switch(conf-tm-sub)# snsr-grp 200 sample-interval 7000
switch(conf-tm-sub)# snsr-grp 300 sample-interval 0
switch(conf-tm-sub)# dst-grp 100
switch(conf-tm-sub)# dst-grp 300
```

Subscribe to UDP transport

Use the following command to configure the UDP transport to stream data using a datagram socket either in JSON or GPB:

```
destination-group num
    ip address xxx.xxx.xxx.xxx port xxxx protocol UDP encoding {JSON | GPB }
Example for an IPv6 destination:
destination-group 100 ipv6 address 10:10::1 port 8000 protocol gRPC encoding GPB
```

The UDP telemetry is with the following header:

```
typedef enum tm_encode_ {
TM_ENCODE_DUMMY,
TM_ENCODE_GPB,
TM_ENCODE_JSON,
TM_ENCODE_XML,
TM_ENCODE_MAX, } tm_encode_type_t;
typedef struct tm_pak_hdr_ {
uint8_t version; /* 1 */ uint8_t encoding; uint16_t msg_size; uint8_t secure; uint8_t
padding;
}__attribute__ ((packed, aligned (1))) tm_pak_hdr_t;
```

Use the first 6 bytes in the payload to process telemetry data using UDP, using one of the following methods:

- Read the information in the header to determine which decoder to use to decode the data, JSON or GPB, if the receiver is meant to receive different types of data from multiple endpoints.

- Remove the header if you are expecting one decoder (JSON or GPB) but not the other.

Verify telemetry configuration

You can verify the telemetry configuration using the show running-config telemetry command, as shown in this example.

```
switch(config)# telemetry
switch(config-telemetry)# destination-group 100
switch(conf-tm-dest)# ip address 1.2.3.4 port 50003
switch(conf-tm-dest)# ip address 1.2.3.4 port 50004
switch(conf-tm-dest)# end
switch# show run telemetry

!Command: show running-config telemetry
!Time: Thu Oct 13 21:10:12 2016

version 7.0(3)I5(1)
feature telemetry

telemetry
destination-group 100
ip address 1.2.3.4 port 50003 protocol gRPC encoding GPB
ip address 1.2.3.4 port 50004 protocol gRPC encoding GPB
```

Configuration Example for Telemetry Merge subscriptions

Following is an example to configure merge-subscription

```
Telemetry
  merge-subscriptions
  destination-group 1
    ip address 192.186.1.2 port 1 protocol HTTP encoding JSON
  Destination-group 2
    ip address 192.168.1.3 port 2 protocol gRPC encoding GPB
  sensor-group 1
    path sys/fm
  subscription 1
    dst-grp 1
    snsr-grp 1 sample-interval 10000
  subscription 2
    dst-grp 2
    snsr-grp 1 sample-interval 25000
```

The data for sensor group 1 is collected every 10 seconds. Collected data is sent to destinations of subscription 1 every 10 seconds and destinations of subscription 2 every 20 seconds.

In the given example for snsr-grp 1 in subscription 2, the min_sample interval is 10s and the cur_sample_interval is 25s. Therefore, the data for it will be sent every 20s.

```
Sample_interval = 10*floor(25/10)
            = 10*2
                          = 20s
```

Verifying configuration

Use the following command to verify the configuration.

```
show telemetry control database sensor-groups
Sensor Group Database size = 1
Row ID     Sensor Group ID  Sensor Group type  Sampling interval(ms)  Linked subscriptions
```

```
  SubID
1          1               Timer   /DME            10000/Running      2
   1
Collection Time in ms (Cur/Min/Max): 1/1/2
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 10003/10003/10003
Streaming Time in ms (Cur/Min/Max): 10004/10004/10006
Collection Statistics:
collection_id_dropped      = 0
last_collection_id_dropped = 0
drop_count                 = 0
Configuration method: CONFIG_DME-ADMIN
2          1               Timer   /DME            20000*/Running      2
    2
Collection Time in ms (Cur/Min/Max): 1/1/1
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 4004/4004/4004
Streaming Time in ms (Cur/Min/Max): 4005/4005/4005
Collection Statistics:
collection_id_dropped      = 0
last_collection_id_dropped = 0
drop_count                 = 0
Configuration method: CONFIG_DME-ADMIN
*Calculated sample interval for merge-subscriptions
```

Displaying Telemetry Configuration and Statistics

Use the following NX-OS CLI **show** commands to display telemetry configuration, statistics, errors, and session information.

show telemetry yang direct-path cisco-nxos-device

This command displays YANG paths that are directly encoded to perform better than other paths.

```
switch# show telemetry yang direct-path cisco-nxos-device
) Cisco-NX-OS-device:System/lldp-items
2) Cisco-NX-OS-device:System/acl-items
3) Cisco-NX-OS-device:System/mac-items
4) Cisco-NX-OS-device:System/intf-items
5) Cisco-NX-OS-device:System/procsys-items/sysload-items
6) Cisco-NX-OS-device:System/ospf-items
7) Cisco-NX-OS-device:System/procsys-items
8) Cisco-NX-OS-device:System/ipqos-items/queuing-items/policy-items/out-items
9) Cisco-NX-OS-device:System/mac-items/static-items
10) Cisco-NX-OS-device:System/ch-items
11) Cisco-NX-OS-device:System/cdp-items
12) Cisco-NX-OS-device:System/bd-items
13) Cisco-NX-OS-device:System/eps-items
14) Cisco-NX-OS-device:System/ipv6-items
```

show telemetry control database

This command displays YANG paths that are directly encoded to perform better than other paths.

```
switch# show telemetry control database ?
  <CR>
  >                   Redirect it to a file
  >>                  Redirect it to a file in append mode
  destination-groups  Show destination-groups
  destinations        Show destinations
  sensor-groups       Show sensor-groups
```

```
  sensor-paths        Show sensor-paths
  subscriptions       Show subscriptions
  |                   Pipe command output to filter

switch# show telemetry control database

Subscription Database size = 1

------------------------------------------------------------------------------
Subscription ID      Data Collector Type
------------------------------------------------------------------------------
100                  DME NX-API

Sensor Group Database size = 1

------------------------------------------------------------------------------
Sensor Group ID  Sensor Group type  Sampling interval(ms)  Linked subscriptions
------------------------------------------------------------------------------
100              Timer              10000(Running)         1

Sensor Path Database size = 1

------------------------------------------------------------------------------
Subscribed Query Filter  Linked Groups  Sec Groups  Retrieve level  Sensor Path
------------------------------------------------------------------------------
No                       1              0           Full            sys/fm

Destination group Database size = 2

------------------------------------------------------------------------------
Destination Group ID  Refcount
------------------------------------------------------------------------------
100                   1

Destination Database size = 2

------------------------------------------------------------------------------
Dst IP Addr     Dst Port   Encoding   Transport  Count
------------------------------------------------------------------------------
192.168.20.111       12345      JSON       HTTP       1
192.168.20.123 50001      GPB        gRPC       1
```

show telemetry control database sensor-paths

This command displays sensor path details for telemetry configuration, including counters for encoding, collection, transport, and streaming.

```
switch(conf-tm-sub)# show telemetry control database sensor-paths
Sensor Path Database size = 4
------------------------------------------------------------------------------------------------------

 Row ID Subscribed Linked Groups Sec Groups Retrieve level Path(GroupId) : Query :
Filter
------------------------------------------------------------------------------------------------------
1 No 1 0 Full sys/cdp(1) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 65785/65785/65785
Collection Time in ms (Cur/Min/Max): 10/10/55
Encoding Time in ms (Cur/Min/Max): 8/8/9
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 18/18/65
2 No 1 0 Self show module(2) : NA : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 1107/1106/1107
```

```
Collection Time in ms (Cur/Min/Max): 603/603/802
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/1
Streaming Time in ms (Cur/Min/Max): 605/605/803
3 No 1 0 Full
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 0/0/44
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1/1/44 sys/bgp(1) : NA : NA
4 No 1 0 Self
GPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
JSON Encoded Data size in bytes (Cur/Min/Max): 2442/2441/2442
Collection Time in ms (Cur/Min/Max): 1703/1703/1903
Encoding Time in ms (Cur/Min/Max): 0/0/0
Transport Time in ms (Cur/Min/Max): 0/0/0
Streaming Time in ms (Cur/Min/Max): 1703/1703/1904
switch(conf-tm-sub)#
 show version(2) : NA : NA
```

show telemetry control stats

This command displays the statistics about the internal databases about configuration of telemetry.

```
switch# show telemetry control stats
show telemetry control stats entered

------------------------------------------------------------------------------
Error Description                                            Error Count
------------------------------------------------------------------------------
Chunk allocation failures                                    0
Sensor path Database chunk creation failures                 0
Sensor Group Database chunk creation failures                0
Destination Database chunk creation failures                 0
Destination Group Database chunk creation failures           0
Subscription Database chunk creation failures                0
Sensor path Database creation failures                       0
Sensor Group Database creation failures                      0
Destination Database creation failures                       0
Destination Group Database creation failures                 0
Subscription Database creation failures                      0
Sensor path Database insert failures                         0
Sensor Group Database insert failures                        0
Destination Database insert failures                         0
Destination Group Database insert failures                   0
Subscription insert to Subscription Database failures        0
Sensor path Database delete failures                         0
Sensor Group Database delete failures                        0
Destination Database delete failures                         0
Destination Group Database delete failures                   0
Delete Subscription from Subscription Database failures      0
Sensor path delete in use                                    0
Sensor Group delete in use                                   0
Destination delete in use                                    0
Destination Group delete in use                              0
Delete destination(in use) failure count                     0
Failed to get encode callback                                0
Sensor path Sensor Group list creation failures              0
Sensor path prop list creation failures                      0
Sensor path sec Sensor path list creation failures           0
Sensor path sec Sensor Group list creation failures          0
Sensor Group Sensor path list creation failures              0
Sensor Group Sensor subs list creation failures              0
```

```
Destination Group subs list creation failures                0
Destination Group Destinations list creation failures        0
Destination Destination Groups list creation failures        0
Subscription Sensor Group list creation failures             0
Subscription Destination Groups list creation failures       0
Sensor Group Sensor path list delete failures                0
Sensor Group Subscriptions list delete failures              0
Destination Group Subscriptions list delete failures         0
Destination Group Destinations list delete failures          0
Subscription Sensor Groups list delete failures              0
Subscription Destination Groups list delete failures         0
Destination Destination Groups list delete failures          0
Failed to delete Destination from Destination Group          0
Failed to delete Destination Group from Subscription         0
Failed to delete Sensor Group from Subscription              0
Failed to delete Sensor path from Sensor Group               0
Failed to get encode callback                                0
Failed to get transport callback                             0
switch#  Destination Database size = 1

-------------------------------------------------------------------------------
Dst IP Addr     Dst Port   Encoding   Transport  Count
-------------------------------------------------------------------------------
192.168.20.123 50001      GPB        gRPC       1
```

show telemetry data collector brief

This command displays the brief statistics about the data collection.

```
switch# show telemetry data collector brief
------------------------------------------------------------------------
 Collector Type Successful Collections Failed Collections
------------------------------------------------------------------------
 DME 143 0
```

show telemetry data collector details

This command displays detailed statistics about the data collection which includes breakdown of all sensor paths.

- **Successful** – Reflects the number of completed collections. The successful count includes only the number of successful collections. It does not include the number of attempts. A successful collection does not necessarily mean data is returned. Failure is used for cases like invalid path, OC model not installed, errors in collecting data, and so on. No data is not considered to be a failure.
- **Payload** – Reflects the number of payloads per collection.
- **Failed** – Reflects the number of failed collections or payloads.
- **Skipped** – Reflects the number of skipped collections. Collections may skip due to an aggressive sample interval and/or a lack of system resources.
- **Dropped** – Reflects the number of dropped payloads. Payloads may get dropped due to system resources or transport state.
- **Sensor Path** – Shows the sensor path.

```
switch# show telemetry data collector details
------------------------------------------------------------------------------------------------------
Row ID         Successful     Payloads   Failed      Skipped        Dropped        Sensor
```

```
Path(GroupId)
------------------------------------------------------------------------------------------------------------
1              3              6120       0           0              0
Cisco-NX-OS-device:System/intf-items(1)
```

show telemetry event collector errors

This command displays the errors statistic about the event collection.

```
switch# show telemetry event collector errors
------------------------------------------------------------------------------
 Error Description Error Count
------------------------------------------------------------------------------
APIC-Cookie Generation Failures - 0
Authentication Failures - 0
Authentication Refresh Failures - 0
Authentication Refresh Timer Start Failures - 0
Connection Timer Start Failures - 0
Connection Attempts - 3
Dme Event Subscription Init Failures - 0
Event Data Enqueue Failures - 0
Event Subscription Failures - 0
Event Subscription Refresh Failures - 0
Pending Subscription List Create Failures - 0
Subscription Hash Table Create Failures - 0
Subscription Hash Table Destroy Failures - 0
Subscription Hash Table Insert Failures - 0
Subscription Hash Table Remove Failures - 0
Subscription Refresh Timer Start Failures - 0
Websocket Connect Failures - 0
```

show telemetry event collector stats

This command displays the statistics about the event collection which includes breakdown of all sensor paths.

```
switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Collection Count Latest Collection Time Sensor Path
------------------------------------------------------------------------------
```

show telemetry control pipeline stats

This command displays the statistics for the telemetry pipeline.

```
switch# show telemetry pipeline stats
Main Statistics:
    Timers:
        Errors:
            Start Fail        =     0

    Data Collector:
        Errors:
            Node Create Fail  =     0

    Event Collector:
        Errors:
            Node Create Fail  =     0    Node Add Fail     =     0
            Invalid Data      =     0



Queue Statistics:
    Request Queue:
```

```
        High Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0

        Low Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0

    Data Queue:
        High Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0

        Low Priority Queue:
            Info:
                Actual Size       =    50    Current Size      =     0
                Max Size          =     0    Full Count        =     0

            Errors:
                Enqueue Error     =     0    Dequeue Error     =     0
```

show telemetry transport

This command displays all configured transport sessions.

```
switch# show telemetry transport

Session Id      IP Address      Port       Encoding   Transport  Status
--------------------------------------------------------------------------------
0               192.168.20.123   50001      GPB        gRPC       Connected
```

Table 2: Syntax Description for show telemetry transport

Syntax	Description
show	Shows running system information
telemetry	Shows telemetry information
transport	Shows telemetry transport information
session_id	(Optional) Session id
stats	(Optional) Shows all telemetry statistics informatio
errors	(Optional) Show all telemetry error information
readonly	(Optional)
TABLE_transport_info	(Optional) Transport information
session_idx	(Optional) Session Id

ip_address	(Optional) Transport IP address
port	(Optional) Transport port
dest_info	(Optional) Destination information
encoding_type	(Optional) Encoding type
transport_type	(Optional) Transport type
transport_status	(Optional) Transport status
transport_security_cert_fname	(Optional) Transport security file name
transport_last_connected	(Optional) Transport last connected
transport_last_disconnected	(Optional) Last time this destination configuration was
transport_errors_count	(Optional) Transport errors count
transport_last_tx_error	(Optional) Transport last tx error
transport_statistics	(Optional) Transport statistics
t_session_id	(Optional) Transport Session id
connect_statistics	(Optional) Connection statistics
connect_count	(Optional) Connection count
last_connected	(Optional) Last connected timestamp
disconnect_count	(Optional) Disconnect count
last_disconnected	(Optional) Last time this destination configuration was
trans_statistics	(Optional) Transport statistics
compression	(Optional) Compression status
source_interface_name	(Optional) Source interface name
source_interface_ip	(Optional) Source interface IP
transmit_count	(Optional) Transmission count
last_tx_time	(Optional) Last Transmission time
min_tx_time	(Optional) Minimum transmission time
max_tx_time	(Optional) Maximum transmission time
avg_tx_time	(Optional) Average transmission time
cur_tx_time	(Optional) Current transmission time
transport_errors	(Optional) Transport errors
connect_errors	(Optional) Connection errors
connect_errors_count	(Optional) Connection error count
trans_errors	(Optional) Transport errors
trans_errors_count	(Optional) Transport error count

last_tx_error	(Optional) Last transport error
last_tx_return_code	(Optional) Last transport return code
transport_retry_stats	(Optional) Retry Statistics
ts_event_retry_bytes	(Optional) Event Retry buffer size
ts_timer_retry_bytes	(Optional) Timer Retry buffer size
ts_event_retry_size	(Optional) Event Retry number of messages
ts_timer_retry_size	(Optional) Timer Retry number of messages
ts_retries_sent	(Optional) Number of retries sent
ts_retries_dropped	(Optional) Number of retries dropped
event_retry_bytes	(Optional) Event Retry buffer size
timer_retry_bytes	(Optional) Timer Retry buffer size
retries_sent	(Optional) Number of retries sent
retries_dropped	(Optional) Number of retries dropped
retry_buffer_size	(Optional) Retry buffer size

show telemetry transport <session-id>

This command displays detailed session information for a specific transport session.

```
switch# show telemetry transport 0

Session Id:          0
IP Address:Port      192.168.20.123:50001
Encoding:            GPB
Transport:           gRPC
Status:              Disconnected
Last Connected:      Fri Sep 02 11:45:57.505 UTC

Tx Error Count:      224
Last Tx Error:       Fri Sep 02 12:23:49.555 UTC

switch# show telemetry transport 1

Session Id:          1
IP Address:Port      10.30.218.56:51235
Encoding:            JSON
Transport:           HTTP
Status:              Disconnected
Last Connected:      Never

Tx Error Count:      3
Last Tx Error:       Wed Apr 19 15:56:51.617 PDT
The following example shows output from an IPv6 entry.
switch# show telemetry transport 0
Session Id: 0
IP Address:Port [10:10::1]:8000
Transport: GRPC
Status: Idle
Last Connected: Never
Last Disconnected: Never
Tx Error Count: 0
```

```
Last Tx Error: None
Event Retry Queue Bytes: 0
Event Retry Queue Size: 0
Timer Retry Queue Bytes: 0
Timer Retry Queue Size: 0
Sent Retry Messages: 0
Dropped Retry Messages: 0
```

show telemetry transport <session-id> stats

This command displays details of a specific transport session.

```
switch# show telemetry transport 0 stats

Session Id:          0
IP Address:Port      192.168.20.123:50001
Encoding:            GPB
Transport:           GRPC
Status:              Connected
Last Connected:      Mon May 01 11:29:46.912 PST
Last Disconnected:   Never
Tx Error Count:      0
Last Tx Error:       None
```

show telemetry transport <session-id> errors

This command displays detailed error statistics for a specific transport session.

```
switch# show telemetry transport 0 errors

Session Id:                    0
Connection Stats
   Connection Count            1
   Last Connected:             Mon May 01 11:29:46.912 PST
   Disconnect Count            0
   Last Disconnected:          Never
Transmission Stats
   Transmit Count:             1225
   Last TX time:               Tue May 02 11:40:03.531 PST
   Min Tx Time:                7                   ms
   Max Tx Time:                1760                ms
   Avg Tx Time:                500                 ms
```

show telemetry control databases sensor-paths

These following configuration steps result in the **show telemetry control databases sensor-paths** command output

```
below. feature telemetry
telemetry
destination-group 1
ip address 172.25.238.13 port 50600 protocol gRPC encoding GPB
sensor-group 1
path sys/cdp depth unbounded path sys/intf depth unbounded path sys/mac depth 0
subscription 1
dst-grp 1 snsr-grp 1 sample-interval 1000 Command output. switch# show telemetry control
databases sensor-paths
Sensor Path Database size = 3
------------------------------------------------------------------------------
--------------------
Row ID Subscribed Linked Groups Sec Groups Retrieve level Path(GroupId) : Query : Filter
------------------------------------------------------------------------------
--------------------
```

```
1 No 1 0 Full sys/cdp(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 30489/30489/30489
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 6/5/54
Encoding Time in ms (Cur/Min/Max): 5/5/6
Transport Time in ms (Cur/Min/Max): 1027/55/1045
Streaming Time in ms (Cur/Min/Max): 48402/5/48402
2 No 1 0 Full sys/intf(1) : N
A : NA
GPB Encoded Data size in bytes (Cur/Min/Max): 539466/539466/539466
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 66/64/114
Encoding Time in ms (Cur/Min/Max): 91/90/92
Transport Time in ms (Cur/Min/Max): 4065/4014/5334
Streaming Time in ms (Cur/Min/Max): 48365/64/48365
 3 No 1 0 Self sys/mac(1) : NA
: NA
GPB Encoded Data size in bytes (Cur/Min/Max): 247/247/247
JSON Encoded Data size in bytes (Cur/Min/Max): 0/0/0
CGPB Encoded Data size in bytes (Cur/Min/Max): 0/0/0
Collection Time in ms (Cur/Min/Max): 1/1/47
Encoding Time in ms (Cur/Min/Max): 1/1/1
Transport Time in ms (Cur/Min/Max): 4/1/6
Streaming Time in ms (Cur/Min/Max): 47369/1/47369
```

show telemetry transport sessions

The following commands loop through all the transport sessions and prints the information in one command:

```
switch# show telemetry transport sessions
switch# show telemetry transport stats
switch# show telemetry transport errors
switch# show telemetry transport all
The following is an example for telemetry transport session:
switch# show telemetry transport sessions
Session Id:          0
IP Address:Port      172.27.254.13:50004
Transport:           GRPC
Status:              Transmit Error
SSL Certificate:     trustpoint1
Last Connected:      Never
Last Disconnected:   Never
Tx Error Count:      2
Last Tx Error:       Wed Aug 19 23:32:21.749 UTC
…
Session Id:          4
IP Address:Port      172.27.2
```

Telemetry Ephemeral Event

To support ephemeral event, a new sensor path query-condition is added. To enable accounting log ephermeral event streaming, use the following query condition:

```
sensor-group 1 path sys/accounting/log query-condition
query-target=subtree&complete-mo=yes&notify-interval=1

The following are the other sensor paths that support ephemeral event:
sys/pim/inst/routedb-route, sys/pim/pimifdb-adj, sys/pim/pimifdb-prop
sys/igmp/igmpifdb-prop, sys/igmp/inst/routedb, sys/igmpsnoop/inst/dom/db-exptrack,
```

```
sys/igmpsnoop/inst/dom/db-group, sys/igmpsnoop/inst/dom/db-mrouter
sys/igmpsnoop/inst/dom/db-querier, sys/igmpsnoop/inst/dom/db-snoop
```

Configuring Telemetry Using the NX-API

Telemetry Model in the DME

The telemetry application is modeled in the DME with the following structure:

```
model
|----package [name:telemetry]
    |  @name:telemetry
    |----objects
         |----mo [name:Entity]
              |    @name:Entity
              |      @label:Telemetry System
              |--property
              |    @name:adminSt
              |      @type:AdminState
              |
              |----mo [name:SensorGroup]
              |    |    @name:SensorGroup
              |    |      @label:Sensor Group
              |    |--property
              |    |    @name:id [key]
              |    |      @type:string:Basic
              |    |
              |    |----mo [name:SensorPath]
              |    |         |   @name:SensorPath
              |    |         |     @label:Sensor Path
              |    |         |--property
              |    |         |    @name:path [key]
              |    |         |      @type:string:Basic
              |    |         |    @name:filterCondition
              |    |         |      @type:string:Basic
              |    |         |    @name:excludeFilter
              |    |         |      @type:string:Basic
              |    |         |    @name:depth
              |    |         |      @type:RetrieveDepth
              |
              |----mo [name:DestGroup]
              |    |    @name:DestGroup
              |    |      @label:Destination Group
              |    |--property
              |    |    @name:id
              |    |      @type:string:Basic
              |    |
              |    |----mo [name:Dest]
              |    |         |   @name:Dest
              |    |         |     @label:Destination
              |    |         |--property
              |    |         |    @name:addr [key]
              |    |         |      @type:address:Ip
              |    |         |    @name:port [key]
              |    |         |      @type:scalar:Uint16
              |    |         |    @name:proto
              |    |         |      @type:Protocol
              |    |         |    @name:enc
              |    |         |      @type:Encoding
              |
```

```
|----mo [name:Subscription]
     |    @name:Subscription
     |      @label:Subscription
     |--property
     |    @name:id
     |      @type:scalar:Uint64
     |----reldef
     |    |  @name:SensorGroupRel
     |    |    @to:SensorGroup
     |    |    @cardinality:ntom
     |    |    @label:Link to sensorGroup entry
     |    |--property
     |         @name:sampleIntvl
     |           @type:scalar:Uint64
     |
     |----reldef
          |  @name:DestGroupRel
          |    @to:DestGroup
          |    @cardinality:ntom
          |    @label:Link to destGroup entry
```

Configuring with NX-API

In the object model of the switch DME, the configuration of the telemetry feature is defined in a hierarchical structure of objects as shown in the section "Telemetry Model in the DME." Following are the main objects to be configured:

- **fmEntity** — Contains the NX-API and Telemetry feature states.
- **fmNxapi** — Contains the NX-API state.
- **fmTelemetry** — Contains the Telemetry feature state.
- **telemetryEntity** — Contains the telemetry feature configuration.
- **telemetrySensorGroup** — Contains the definitions of one or more sensor paths or nodes to be monitored for telemetry. The telemetry entity can contain one or more sensor groups.
- **telemetryRtSensorGroupRel** — Associates the sensor group with a telemetry subscription.
- **telemetrySensorPath** — A path to be monitored. The sensor group can contain multiple objects of this type.
- **telemetryDestGroup** — Contains the definitions of one or more destinations to receive telemetry data. The telemetry entity can contain one or more destination groups.
- **telemetryRtDestGroupRel** — Associates the destination group with a telemetry subscription.
- **telemetryDest** — A destination address. The destination group can contain multiple objects of this type.
- **telemetrySubscription** — Specifies how and when the telemetry data from one or more sensor groups is sent to one or more destination groups.
- **telemetryRsDestGroupRel** — Associates the telemetry subscription with a destination group.
- **telemetryRsSensorGroupRel** — Associates the telemetry subscription with a sensor group.
- **telemetryCertificate** — Associates the telemetry subscription with a certificate and hostname.

To configure the telemetry feature using the NX-API, you must construct a JSON representation of the telemetry object structure and push it to the DME with an HTTP or HTTPS POST operation.



Note For detailed instructions on using the NX-API, see the *Cisco Nexus 3000 and 9000 Series NX-API REST SDK User Guide and API Reference*

Before you begin

Your switch must be configured to run the NX-API from the CLI:

```
switch(config)# feature nxapi
```

```
nxapi use-vrf vrf_name nxapi http port port_number
```

SUMMARY STEPS

1. Enable the telemetry feature.
2. Create the root level of the JSON payload to describe the telemetry configuration.
3. Create a sensor group to contain the defined sensor paths.
4. (Optional) Add an SSL/TLS certificate and a host.
5. Define a telemetry destination group.
6. Define a telemetry destination profile.
7. Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent.
8. Enable gRPC chunking and set the chunking size, between 64 and 4096 bytes.
9. Create a telemetry subscription to configure the telemetry behavior
10. Add the sensor group object as a child object to the telemetrySubscription element under the root element (telemetryEntity).
11. Create a relation object as a child object of the subscription to associate the subscription to the telemetry sensor group and to specify the data sampling behavior.
12. Define one or more sensor paths or nodes to be monitored for telemetry.
13. Add sensor paths as child objects to the sensor group object (telemetrySensorGroup).
14. Add destinations as child objects to the destination group object (telemetryDestGroup).
15. Add the destination group object as a child object to the root element (telemetryEntity).
16. Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription.
17. Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription.
18. Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group.
19. Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration.

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	Enable the telemetry feature. **Example:** `{ "fmEntity" : { "children" : [{ "fmTelemetry" : { "attributes" : { "adminSt" : "enabled" } } } } ] } }`	The root element is fmTelemetry and the base path for this element is `sys/fm`. Configure the adminSt attribute as `enabled`.
Step 2	Create the root level of the JSON payload to describe the telemetry configuration. **Example:** `{ "telemetryEntity": { "attributes": { "dn": "sys/tm" }, } }`	The root element is telemetryEntity and the base path for this element is sys/tm. Configure the dn attribute as sys/tm.
Step 3	Create a sensor group to contain the defined sensor paths. **Example:** `"telemetrySensorGroup": { "attributes": { "id": "10", "rn": "sensor-10" }, "children": [{ }] }`	A telemetry sensor group is defined in an object of class telemetrySensorGroup . Configure the following attributes of the object: • id — An identifier for the sensor group. Currently only numeric ID values are supported. • rn — The relative name of the sensor group object in the format: sensor-id . • dataSrc — Selects the data source from DEFAULT , DME , YANG , or NX-API . Children of the sensor group object include sensor paths and one or more relation objects (telemetryRtSensorGroupRel) to associate the sensor group with a telemetry subscription.
Step 4	(Optional) Add an SSL/TLS certificate and a host. **Example:** `{ "telemetryCertificate": { "attributes": {`	The telemetryCertificate defines the location of the SSL/TLS certificate with the telemetry subscription/destination.

	Command or Action	Purpose
	<pre>"filename": "root.pem" "hostname": "c.com" } } }</pre>	
Step 5	Define a telemetry destination group. **Example:** <pre>{ "telemetryDestGroup": { "attributes": { "id": "20" } } }</pre>	A telemetry destination group is defined in telemetryEntity . Configure the id attribute.
Step 6	Define a telemetry destination profile. **Example:** <pre>{ "telemetryDestProfile": { "attributes": { "adminSt": "enabled" }, "children": [{ "telemetryDestOptSourceInterface": { "attributes": { "name": "lo0" } } }] } }</pre>	A telemetry destination profile is defined in telemetryDestProfile . • Configure the adminSt attribute as `enabled`. Under telemetryDestOptSourceInterface , configure the name attribute with an interface name to stream data from the configured interface to a destination with the source IP address.
Step 7	Define one or more telemetry destinations, consisting of an IP address and port number to which telemetry data will be sent. **Example:** <pre>{ "telemetryDest": { "attributes": { "addr": "1.2.3.4", "enc": "GPB", "port": "50001", "proto": "gRPC", "rn": "addr-[1.2.3.4]-port-50001" } } }</pre>	A telemetry destination is defined in an object of class telemetryDest . Configure the following attributes of the object: • addr — The IP address of the destination. • port — The port number of the destination. • rn — The relative name of the destination object in the format: path-[path] . • enc — The encoding type of the telemetry data to be sent. NX-OS supports: • Google protocol buffers (GPB) for gRPC. • JSON for C. • proto — The transport protocol type of the telemetry data to be sent. NX-OS supports: • gRPC

	Command or Action	Purpose
		• HTTP • Supported encoded types are: • HTTP/JSON YES • HTTP/Form-data YES Only supported for Bin Logging. • GRPC/GPB-Compact YES Native Data Source Only. • GRPC/GPB YES • UDP/GPB YES UDP/JSON YES
Step 8	Enable gRPC chunking and set the chunking size, between 64 and 4096 bytes. **Example:** `{ "telemetryDestGrpOptChunking": { "attributes": { "chunkSize": "2048", "dn": "sys/tm/dest-1/chunking" } } }`	See gRPC Chunking section for more information.
Step 9	Create a telemetry subscription to configure the telemetry behavior **Example:** `"telemetrySubscription": { "attributes": { "id": "30", "rn": "subs-30" }, "children": [{ }] }`	A telemetry subscription is defined in an object of class telemetrySubscription . Configure the following attributes of the object: • id — An identifier for the subscription. Currently only numeric ID values are supported. • rn — The relative name of the subscription object in the format: subs-id . Children of the subscription object include relation objects for sensor groups (telemetryRsSensorGroupRel) and destination groups (telemetryRsDestGroupRel).
Step 10	Add the sensor group object as a child object to the telemetrySubscription element under the root element (telemetryEntity). **Example:** `{ "telemetrySubscription": { "attributes": { "id": "30" }`	

	Command or Action	Purpose
	<pre>"children": [{ "telemetryRsSensorGroupRel": { "attributes": { "sampleIntvl": "5000", "tDn": "sys/tm/sensor-10" } } }] } }</pre>	
Step 11	Create a relation object as a child object of the subscription to associate the subscription to the telemetry sensor group and to specify the data sampling behavior. **Example:** <pre>"telemetryRsSensorGroupRel": { "attributes": { "rType": "mo", "rn": "rssensorGroupRel-[sys/tm/sensor-10]", "sampleIntvl": "5000", "tCl": "telemetrySensorGroup", "tDn": "sys/tm/sensor-10", "tType": "mo" } }</pre>	The relation object is of class telemetryRsSensorGroupRel and is a child object of telemetrySubscription . Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rssensorGroupRel-[sys/tm/sensor-group-id] . • sampleIntvl — The data sampling period in milliseconds. An interval value of 0 creates an event-based subscription, in which telemetry data is sent only upon changes under the specified MO. An interval value greater than 0 creates a frequency-based subscription, in which telemetry data is sent periodically at the specified interval. For example, an interval value of 15000 results in the sending of telemetry data every 15 seconds. • tCl — The class of the target (sensor group) object, which is telemetrySensorGroup . • tDn — The distinguished name of the target (sensor group) object, which is sys/tm/sensor-group-id . • rType — The relation type, which is mo for managed object. tType — The target type, which is mo for managed object.
Step 12	Define one or more sensor paths or nodes to be monitored for telemetry. **Example:** Single sensor path <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp",</pre>	A sensor path is defined in an object of class telemetrySensorPath . Configure the following attributes of the object: • path — The path to be monitored. • rn — The relative name of the path object in the format: path-[path] • depth — The retrieval level for the sensor path. A depth setting of 0 retrieves only the root MO properties. filterCondition — (Optional) Creates a specific filter for event-based subscriptions. The DME provides the filter

	Command or Action	Purpose
	<pre> "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre> **Example:** Multiple sensor paths <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }, { "telemetrySensorPath": { "attributes": { "excludeFilter": "", "filterCondition": "", "path": "sys/fm/dhcp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre> **Example:** Single sensor path filtering for BGP disable events: <pre>{ "telemetrySensorPath": { "attributes": { "path": "sys/cdp", "rn": "path-[sys/cdp]", "excludeFilter": "", "filterCondition": "eq(fmBgp.operSt.\"disabled\")", "path": "sys/fm/bgp", "secondaryGroup": "0", "secondaryPath": "", "depth": "0" } } }</pre>	expressions. For more information about filtering, see the Cisco APIC REST API Usage Guidelines on composing queries. You can find it at the following Cisco APIC documents landing page:

	Command or Action	Purpose
Step 13	Add sensor paths as child objects to the sensor group object (telemetrySensorGroup).	
Step 14	Add destinations as child objects to the destination group object (telemetryDestGroup).	
Step 15	Add the destination group object as a child object to the root element (telemetryEntity).	
Step 16	Create a relation object as a child object of the telemetry sensor group to associate the sensor group to the subscription. **Example:** `"telemetryRtSensorGroupRel": { "attributes": { "rn": "rtsensorGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } }`	The relation object is of class telemetryRtSensorGroupRel and is a child object of telemetrySensorGroup . Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rtsensorGroupRel-[sys/tm/subscription-id] . • tCl — The target class of the subscription object, which is telemetrySubscription . tDn — The target distinguished name of the subscription object, which is sys/tm/subscription-id .
Step 17	Create a relation object as a child object of the telemetry destination group to associate the destination group to the subscription. **Example:** `"telemetryRtDestGroupRel": { "attributes": { "rn": "rtdestGroupRel-[sys/tm/subs-30]", "tCl": "telemetrySubscription", "tDn": "sys/tm/subs-30" } }`	The relation object is of class telemetryRtDestGroupRel and is a child object of telemetryDestGroup . Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rtdestGroupRel-[sys/tm/subscription-id] . • tCl — The target class of the subscription object, which is telemetrySubscription . tDn — The target distinguished name of the subscription object, which is sys/tm/subscription-id .
Step 18	Create a relation object as a child object of the subscription to associate the subscription to the telemetry destination group. **Example:** `"telemetryRsDestGroupRel": { "attributes": { "rType": "mo", "rn": "rsdestGroupRel-[sys/tm/dest-20]", "tCl": "telemetryDestGroup", "tDn": "sys/tm/dest-20", "tType": "mo" } }`	The relation object is of class telemetryRsDestGroupRel and is a child object of telemetrySubscription . Configure the following attributes of the relation object: • rn — The relative name of the relation object in the format: rsdestGroupRel-[sys/tm/destination-group-id] . • tCl — The class of the target (destination group) object, which is telemetryDestGroup . • tDn — The distinguished name of the target (destination group) object, which is sys/tm/destination-group-id . • rType — The relation type, which is mo for managed object. tType — The target type, which is mo for managed object.

	Command or Action	Purpose
Step 19	Send the resulting JSON structure as an HTTP/HTTPS POST payload to the NX-API endpoint for telemetry configuration. **Example:**	The base path for the telemetry entity is `sys/tm` and the NX-API endpoint is: `{{URL}}/api/node/mo/sys/tm.json`

Example

The following is an example of all the previous steps that are collected into one POST payload (note that some attributes may not match):

```
{
    "telemetryEntity": {
        "children": [{
            "telemetrySensorGroup": {
                "attributes": {
                    "id": "10"
                   }
                   "children": [{
                    "telemetrySensorPath": { "attributes": {

                    "excludeFilter": "",
                    "filterCondition": "", "path": "sys/fm/bgp",
                    "secondaryGroup": "0",
                    "secondaryPath": "",
                                                        "depth": "0"
                   }
                  }
                }
                ]
             }
            },
            {
                "telemetryDestGroup": {
                    "attributes": {
                       "id": "20"
                     }
                     "children": [{
                        "telemetryDest": {
                                                                "attributes": {
                            "addr": "10.30.217.80",
                                                                  "port": "50051",
                                                                  "enc": "GPB",
                                                                  "proto": "gRPC"

                          }
                        }
                      }
                     ]
                   }
                },
                {
                   "telemetrySubscription": {
                       "attributes": {
                           "id": "30"
                        }
                        "children": [{
                          "telemetryRsSensorGroupRel": {
```

```
                                                        "attributes": {
                                "sampleIntvl": "5000",
                                "tDn": "sys/tm/sensor-10"
                              }
                            }
                          },
                          {
                             "telemetryRsDestGroupRel": {
                                                          "attributes": {
                                  "tDn": "sys/tm/dest-20"
                              }
                            }
                          }
                          ]
                        }
                    }
                    ]
        }
}
```

Configuration Example for Telemetry Using the NX-API

Streaming Paths to a Destination

This example creates a subscription that streams paths sys/cdp and sys/ipv4 to a destination 1.2.3.4 port 50001 every five seconds.

```
POST https://192.168.20.123/api/node/mo/sys/tm.json
Payload:
{
    "telemetryEntity": {
        "attributes": {
            "dn": "sys/tm"
        },
        "children": [{
            "telemetrySensorGroup": {
                "attributes": {
                    "id": "10",
                    "rn": "sensor-10"
                },  "children": [{
                    "telemetryRtSensorGroupRel": {
                        "attributes": {
                            "rn": "rtsensorGroupRel-[sys/tm/subs-30]",
                            "tCl": "telemetrySubscription",
                            "tDn": "sys/tm/subs-30"
                        }
                    }
                }, {
                    "telemetrySensorPath": {
                        "attributes": {
                            "path": "sys/cdp",
                            "rn": "path-[sys/cdp]",
                            "excludeFilter": "",
                            "filterCondition": "",
                            "secondaryGroup": "0",
                            "secondaryPath": "",
                            "depth": "0"
                        }
                    }
```

```
            }, {
                "telemetrySensorPath": {
                    "attributes": {
                        "path": "sys/ipv4",
                        "rn": "path-[sys/ipv4]",
                        "excludeFilter": "",
                        "filterCondition": "",
                        "secondaryGroup": "0",
                        "secondaryPath": "",
                        "depth": "0"
                    }
                }
            }]
        }
    }, {
        "telemetryDestGroup": {
            "attributes": {
                "id": "20",
                "rn": "dest-20"
            },
            "children": [{
                "telemetryRtDestGroupRel": {
                    "attributes": {
                        "rn": "rtdestGroupRel-[sys/tm/subs-30]",
                        "tCl": "telemetrySubscription",
                        "tDn": "sys/tm/subs-30"
                    }
                }
            }, {
                "telemetryDest": {
                    "attributes": {
                        "addr": "1.2.3.4",
                        "enc": "GPB",
                        "port": "50001",
                        "proto": "gRPC",
                        "rn": "addr-[1.2.3.4]-port-50001"
                    }
                }
            }]
        }
    }, {
        "telemetrySubscription": {
            "attributes": {
                "id": "30",
                "rn": "subs-30"
            },
            "children": [{
                "telemetryRsDestGroupRel": {
                    "attributes": {
                        "rType": "mo",
                        "rn": "rsdestGroupRel-[sys/tm/dest-20]",
                        "tCl": "telemetryDestGroup",
                        "tDn": "sys/tm/dest-20",
                        "tType": "mo"
                    }
                }
            }, {
                "telemetryRsSensorGroupRel": {
                    "attributes": {
                        "rType": "mo",
                        "rn": "rssensorGroupRel-[sys/tm/sensor-10]",
                        "sampleIntvl": "5000",
                        "tCl": "telemetrySensorGroup",
                        "tDn": "sys/tm/sensor-10",
```

```
                            "tType": "mo"
                        }
                    }
                }]
            }
        }]
    }
}
```

Filter Conditions on BGP Notifications

The following example payload enables notifications that trigger when the BFP feature is disabled as per the `filterCondition` attribute in the `telemetrySensorPath` MO. The data is streamed to`10.30.217.80 port 50055`.

```
POST https://192.168.20.123/api/node/mo/sys/tm.json
Payload:
{
  "telemetryEntity": {
    "children": [{
      "telemetrySensorGroup": {
        "attributes": {
          "id": "10"
        }
        "children": [{
          "telemetrySensorPath": {
            "attributes": {
              "excludeFilter": "",
              "filterCondition": "eq(fmBgp.operSt,\"disabled\")",
              "path": "sys/fm/bgp",
              "secondaryGroup": "0",
              "secondaryPath": "",
              "depth": "0"
            }
          }
        }
        ]
      }
    },
    {
      "telemetryDestGroup": {
        "attributes": {
          "id": "20"
        }
        "children": [{
          "telemetryDest": {
            "attributes": {
              "addr": "10.30.217.80",
              "port": "50055",
              "enc": "GPB",
              "proto": "gRPC"
            }
          }
        }
        ]
      }
    },
    {
      "telemetrySubscription": {
        "attributes": {
          "id": "30"
        }
        "children": [{
```

```
            "telemetryRsSensorGroupRel": {
              "attributes": {
                "sampleIntvl": "0",
                "tDn": "sys/tm/sensor-10"
              }
            }
          },
          {
            "telemetryRsDestGroupRel": {
              "attributes": {
                "tDn": "sys/tm/dest-20"
              }
            }
          }
          ]
        }
      }
      ]
    }
  }
```

Using Postman Collection for Telemetry Configuration

See https://github.com/CiscoDevNet/nx-telemetry-proto/tree/master/postman_collections for an example to start configuring the telemetry feature, and can run all telemetry CLI equivalents in a single payload. Modify the file in the preceding link using your preferred text editor to update the payload to your needs, then open the collection in Postman and run the collection.

Multicast Flow Path Visibility

This feature provides you a means to export all the necessary multicast states available on Nexus 3548-XL switch. The export ensures you to have a complete and reliable traceability of the path that each flow takes starting from the source to each of the receivers.

This feature targets to publish all the appropriate information in DME and makes it accessible to any consumer/controller either through push model (Software Telemetry) or pull model (DME REST queries).

The following are the benefits of this feature:

- Flow Path Visualization
- Flow statistics or states export for failure detection
- Root cause analysis by allowing users to run appropriate debug commands on the switches along the flow path

MFDM is a Multicast FIB distribution management which consumes the information from the upper-level component, builds an intelligence for each multicast feature, and then propagates the information to the consumer. This is the core component where the feature is implemented along with DME. It is responsible for publishing all the multicast states to DME, based on the information provided by MRIB and the statistics collected by MFIB.

DME is used to store all the information that needs to be made available to the consumer/controller. It will also be responsible of generating the appropriate notifications to telemetry whenever an object is created or deleted or modified to support event-based notifications.

Telemetry process is responsible for streaming out all the data stored in DME to the consumers and format the data in proper form.

CLIs for Multicast Flow Path Visibility

The following are the CLIs that are introduced to verify the accurate functionality of the Multicast Flow Path Visibility:

- A configuration command to enable the export of information to DME. This CLI enables the feature for every route present in the system.

```
switch(config)# multicast flow-path export

        switch(config)# sh system internal dme run all dn sys/mca/config
```

- A consistency checker show command to perform consistency checks between states present in MFDM and DME. This command allows you to catch inconsistencies quickly, especially on high scale setups.

```
switch# show forwarding distribution internal multicast
        consistency-checker flow-path route

Starting flow-path DME consistency-check for VRF:
        default

(0.0.0.0/0, 230.0.0.1/32). Result:
      PASS

(10.0.0.10/32, 230.0.0.1/32). Result:
      PASS

(0.0.0.0/0, 232.0.0.0/8). Result: PASS
```

- A global show command is used to check if the feature is enabled in the system or not.

```
switch(config)# show forwarding distribution internal
        multicast global_state

**** MFDM Flow PATH VISIBILITY INFO
      ****

Multicast flow-path info export enabled:
      Y

BE DME Handler: 0x117c3e6c

PE DME Handler: 0x117b955c

switch(config)# show forwarding distribution internal
        multicast fpv CC

PASS/FAIL (In case of fail, it will highlight the
        inconsistencies)
```

Cloud Scale Software Telemetry

About Cloud Scale Software Telemetry

Cisco NX-OS supports Cisco Nexus Cloud Scale switches that use the Tahoe ASIC. In such platform, supported Cloud Scale switches host a TCP/IP server that is tightly integrated with the ASICs, which expedites reporting telemetry data from the switch. The server runs on TCP port 7891, and telemetry clients can connect to the server on this port to retrieve hardware-counter data in a maximum of 10 milliseconds.

Cloud Scale software telemetry offers the users the flexibility of creating your own client programs or using the default client program that is bundled into NX-OS release 9.3.1 and later. The user can write client programs in any programming language that supports TCP/IP, such as Python 2.7 or higher, C, or PHP. Client programs must be constructed with the correct message formatting.

Beginning with NX-OS release 9.3(1), the Cloud Scale software telemetry feature is available in NX-OS. The feature is enabled by default, so supported switches running NX-OS 9.3(1) or later can use this feature.

Cloud Scale Software Telemetry Message Formats

Cloud Scale telemetry begins with a handshake between the client and TCP/IP server on the switch, during which the client initiates the connection over the TCP socket. The client message is a 32-bit integer set to zero. The switch responds with a message that contains the counter data in a specific format.

In NX-OS release 9.3(1), the following message format is supported. If you create your own client programs, make sure that the messages that your clients initiate conform to this format

Length	Specifies
4 bytes	The number of ports, *N*
56 bytes	The data for each port, for a total of 56 * *N* bytes. Each 56-byte chunk of data consists of the following: • 24 bytes of interface name • 8 bytes of the transmitted (TX) packets • 8 bytes of transmitted (TX) bytes • 8 bytes of received (RX) packets 8 bytes of received (RX) bytes

Guidelines and Limitations for Cloud Scale Software Telemetry

The following are the guidelines and limitations for the Cloud Scale software telemetry feature:

- For information about supported platforms for Cisco NX-OS prior to release 9.3(x), see the section for https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/platform/platform.html in that guide. Starting with Cisco NX-OS release 9.3(x) for information about supported platforms, see https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/platform/platform.html.
- For custom client telemetry programs, one message format is supported. Your client programs must comply with this format.
- Beginning with Cisco NX-OS Release 10.3(1)F, software telemetry is supported on the Cisco Nexus 9800 platform switches.

Telemetry Path Labels

About Telemetry Path Labels

Beginning with NX-OS release 9.3(1), model-driven telemetry supports path labels. Path labels provide an easy way to gather telemetry data from multiple sources at once. With this feature, you specify the type of telemetry data you want collected, and the telemetry feature gathers that data from multiple paths. The feature then returns the information to one consolidated place, the path label. This feature simplifies using telemetry because you no longer must:

- Have a deep and comprehensive knowledge of the Cisco DME model.
- Create multiple queries and add multiple paths to the subscription, while balancing the number of collected events and the cadence.
- Collect multiple chunks of telemetry information from the switch, which simplifies serviceability.

Path labels span across multiple instances of the same object type in the model, then gather and return counters or events. Path labels support the following telemetry groups:

- **Environment**, which monitors chassis information, including fan, temperature, power, storage, supervisors, and line cards.
- **Interface**, which monitors all the interface counters and status changes.

This label supports predefined keyword filters that can refine the returned data by using the **query-condition** command.

- **Resources**, which monitors system resources such as CPU utilization and memory utilization.
- **VXLAN,** which monitors VXLAN EVPNs including VXLAN peers, VXLAN counters, VLAN counters, and BGP Peer data.

Polling for Data or Receiving Events

The sample interval for a sensor group determines how and when telemetry data is transmitted to a path label. The sample interval can be configured either to periodically poll for telemetry data or gather telemetry data when events occur.

- When the sample interval for telemetry is configured as a non-zero value, telemetry periodically sends the data for the environment, interfaces, resources, and vxlan labels during each sample interval.
- When the sample interval is set to zero, telemetry sends event notifications when the environment, interfaces, resources, and vxlan labels experience operational state updates, as well as creation and deletion of MOs.

Polling for data or receiving events are mutually exclusive. You can configure polling or event-driven telemetry for each path label.

Guidelines and Limitations for Path Labels

The telemetry path labels feature has the following guidelines and limitations:

- The feature supports only Cisco DME data source only.
- You cannot mix and match usability paths with regular DME paths in the same sensor group. For example, you cannot configure `sys/intf` and `interface` in the same sensor group. Also, you cannot configure the same sensor group with `sys/intf` and `interface`. If this situation occurs, NX-OS rejects the configuration.
- User filter keywords, such as `oper-speed` and `counters=[detailed]`, are supported only for the `interface` path.
- The feature does not support other sensor path options, such as `depth` or `filter-condition`.
- The telemetry path labels has the following restrictions in using path labels:
    - Must start with prefix **show** in lowercase, as it is case sensitive.

For example: **show version** is allowed. However, **SHOW version** or version is not allowed.

- Cannot include following characters:
    - ;
    - |
- " " or ' '
    - Cannot include following words:
- telemetry
- conf t

configure

Configuring the Interface Path to Poll for Data or Events

The interface path label monitors all the interface counters and status changes. It supports the following interface types:

- Physical
- Subinterface
- Management
- Loopback
- VLAN
- Port Channel

You can configure the interface path label to either periodically poll for data or receive events. See Polling for Data or Receiving Events, on page 45.



Note The model does not support counters for subinterface, loopback, or VLAN, so they are not streamed out.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**path interface** **Example:** `switch(conf-tm-sensor)# path interface` `switch(conf-tm-sensor)#`	Configure the interface path label, which enables sending one telemetry data query for multiple individual interfaces. The label consolidates the queries for multiple interfaces into one. Telemetry then telemetry gathers the data and returns it to the label. Depending on how the polling interval is configured, interface data is sent based on a periodic basis or whenever the interface state changes.

	Command or Action	Purpose
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path for Non-Zero Counters

You can configure the interface path label with a pre-defined keyword filter that returns only counters that have non-zero values. The filter is `counters=[detailed]`.

By using this filter, the interface path gathers all the available interface counters, filters the collected data, then forwards the results to the receiver. The filter is optional, and if you do not use it, all counters, including zero-value counters, are displayed for the interface path.

Note Using the filter is conceptually similar to issuing show interface mgmt0 counters detailed

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface query-condition counters=[detailed]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*

7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**path interface query-condition counters=[detailed]** **Example:** `switch(conf-tm-sensor)# path interface query-condition counters=[detailed]` `switch(conf-tm-sensor)#`	Configure the interface path label and query for only the non-zero counters from all interfaces.
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:**	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling

	Command or Action	Purpose
	`switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path for Operational Speeds

You can configure the interface path label with a pre-defined keyword filter that returns counters for interfaces of specified operational speeds. The filter is `oper-speed=[]`. The following operational speeds are supported: auto, 10M, 100M, 1G, 10G, 40G, 200G, and 400G.

By using this filter, the interface path gathers the telemetry data for interfaces of the specified speed, then forwards the results to the receiver. The filter is optional. If you do not use it, counters for all interfaces are displayed, regardless of their operational speed.

The filter can accept multiple speeds as a comma-separated list, for example `oper-speed=[1G,10G]` to retrieve counters for interfaces that operate at 1 and 10 Gbps. Do not use a blank space as a delimiter.

Note Interface types subinterface, loopback, and VLAN do not have operational speed properties, so the filter does not support these interface types.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **snsr-group** *sgrp_id* **sample-interval** *interval*
4. **path interface query-condition oper-speed=[speed]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:**	Enter configuration mode.

	Command or Action	Purpose
	`switch# configure terminal` `switch(config)#`	
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 4	**path interface query-condition oper-speed=[speed]** **Example:** `switch(conf-tm-sensor)# path interface query-condition oper-speed=[1G,40G]` `switch(conf-tm-sensor)#`	Configure the interface path label and query for counters from interfaces running the specified speed, which in this example, is 1 and 40 Gbps only
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Interface Path with Multiple Queries

You can configure multiple filters for the same query condition in the interface path label. When you do so, the individual filters you use are ANDed.

Separate each filter in the query condition by using a comma. You can specify any number of filters for the query-condition, but be aware that the more filters you add, the more focused the results become.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path interface query-condition counters=[detailed],oper-speed=[1G,40G]**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**path interface query-condition counters=[detailed],oper-speed=[1G,40G]** **Example:** `switch(conf-tm-sensor)# path interface query-condition counters=[detailed],oper-speed=[1G,40G]` `switch(conf-tm-sensor)#`	Configures multiple conditions in the same query. In this example, the query does both of the following: • Gathers and returns non-zero counters on interfaces running at 1 Gbps. Gathers and returns non-zero counters on interfaces running at 40 Gbps.

	Command or Action	Purpose
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Environment Path to Poll for Data or Events

The environment path label monitors chassis information, including fan, temperature, power, storage, supervisors, and line cards. You can configure the environment path to either periodically poll for telemetry data or get the data when events occur. For information, see Polling for Data or Receiving Events, on page 45.

You can set the resources path to return system resource information through either periodic polling or based on events. This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path environment**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**path environment** **Example:** `switch(conf-tm-sensor)# path environment` `switch(conf-tm-sensor)#`	Configures the environment path label, which enables telemetry data for multiple individual environment objects to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the environment data is either streaming based on the polling interval, or sent when events occur.
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when environment events occur.

	Command or Action	Purpose
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Enabling Power Usage Tracking Functionality

Starting from NX-OS Release 10.4.1(F), the **power usage-history** command is supported on Cisco Nexus 9336C-FX2 and 9332D-GX2B switches to track power consumption. By default this feature is disabled.

Beginning with Cisco NX-OS Release 10.4(2)F, the **power usage-history** command is supported on Cisco Nexus 9000 Series platform switches.

Follow the steps to enable the feature.

SUMMARY STEPS

1. **configure terminal**
2. **[no] power usage-history**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enters configuration mode.
Step 2	**[no] power usage-history** **Example:** `switch# power usage-history` `switch(config)#`	Enables power usage tracking feature. Use the **no** form of this command to disable this feature.

Displaying Power Consumption History

Power Usage Tracking Show Command

See Enabling Power Usage Tracking Functionality, on page 58 to enable power usage tracking feature. After enabling, use **show environment power history** to display power usage statistics for various targets.

Command	Shows
show environment power history { peak | target 1min | target 1hr | target 14 days | target 14days day _day_no }	Power usage information for various targets.

Command Examples

The following shows an example of the **show environment power history peak** command.

```
switch# show environment power history peak
 Power                                    Output Power       Peak  Time           Input
Power      Peak  Time
Supply    Model              Status         (peak)         (Output Power)          (peak)
       (Input Power)
-------------------------------------------------------------------------------------------------------

1      N9K-PAC-3000W-B       Ok             334.30 W      06/07/2023 10:54:29     362.45 W
 06/07/2023 10:54:59
2      N9K-PAC-3000W-B       Ok             362.45 W      06/07/2023 10:53:29     425.80 W
 06/07/2023 10:51:44
switch#


Last 1min usage data would contain average usage in last 15secs, 30secs and 60 secs.

module-4#  show environment power history target 1min

Power                                    Output/Input             Output/Input
 Output/Input
Supply    Model              Status          (15 sec)                (30 sec)
   (60 sec)
-------------------------------------------------------------------------------------------------------

1      N9K-PAC-3000W-B       Ok          330.78 W / 362.45  W     330.00 W / 362.00  W
330.00 W / 362.00  W
2      N9K-PAC-3000W-B       Ok          358.94 W / 415.24  W     358.00 W / 415.00  W
358.00 W / 415.00  W
switch#
```

The following shows the output of the **show environment power history target 1hr** command.

```
switch# show environment power history target 1hr
   1 min avg data for 1 Hr for
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-----------------------------------------------------
    331.00 W          362.00 W           06/07/2023 11:34:44
    330.00 W          362.00 W           06/07/2023 11:33:44
    333.00 W          362.00 W           06/07/2023 11:32:44
    333.00 W          362.00 W           06/07/2023 11:31:44
    331.00 W          362.00 W           06/07/2023 11:30:44


 1 min avg data for 1 Hr for
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
-----------------------------------------------------
    358.00 W          417.00 W           06/07/2023 11:34:44
    358.00 W          417.00 W           06/07/2023 11:33:44
    358.00 W          417.00 W           06/07/2023 11:32:44
    358.00 W          417.00 W           06/07/2023 11:31:44
    357.00 W          415.00 W           06/07/2023 11:30:44
```

The following shows an example of the **show environment power history target 24hr** command.

```
switch# show environment power history target 24hr
   1HR  avg data for 24 Hr for
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power             Time
```

```
-----------------------------------------------------
    332.15 W          363.56 W           06/07/2023 12:50:44
    332.13 W          363.66 W           06/07/2023 11:50:44

 1HR  avg data for 24 Hr for
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
   Output Power      Input Power            Time
-----------------------------------------------------
    358.23 W          416.68 W           06/07/2023 12:50:44
    358.35 W          417.05 W           06/07/2023 11:50:44
switch#
```

The following shows an example of the **show environment power history target 14days** command.

```
switch# show environment power history target 14days
   1 Day avg data  over a period of 14 days
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
Day   Output Power        Input Power            Date
------------------------------------------------------------
  1      332.17 W          363.61 W             06/07/23

 1 Day avg data  over a period of 14 days
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
Day   Output Power        Input Power            Date
------------------------------------------------------------
  1      358.23 W          416.81 W             06/07/23
switch#

This CLI displays the average usage throughout the day for each day in last 14days. For
each PSU 14 days average usage is displayed. A detailed per hour usage for each day is
displayed when day number is given. Output for that is given in next slide.
```

The following shows an example of the **show environment power history target 14days day 1** command.

```
switch# show environment power history target 14days day 1
 1 HR  avg data for 1 Day
 slot: 1 Product Name: N9K-PAC-3000W-B status: Ok
Day 1
   Output Power      Input Power            Time
-----------------------------------------------------
    332.23 W          363.61 W           06/07/2023 13:50:44
    332.15 W          363.56 W           06/07/2023 12:50:44
    332.13 W          363.66 W           06/07/2023 11:50:44

 1 HR  avg data for 1 Day
 slot: 2 Product Name: N9K-PAC-3000W-B status: Ok
Day 1
   Output Power      Input Power            Time
-----------------------------------------------------
    358.11 W          416.71 W           06/07/2023 13:50:44
    358.23 W          416.68 W           06/07/2023 12:50:44
    358.35 W          417.05 W           06/07/2023 11:50:44
switch#
switch#
```

Configuring the Resources Path to Poll for Events or Data

The resources path monitors system resources such as CPU utilization and memory utilization. You can configure this path to either periodically gather telemetry data, or when events occur. See Polling for Data or Receiving Events, on page 45.

This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **path resources**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**path resources** **Example:** `switch(conf-tm-sensor)# path resources` `switch(conf-tm-sensor)#`	Configure the resources path label, which enables telemetry data for multiple individual system resources to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the resource data is either streaming based on the polling interval, or sent when system memory changes to Not OK.
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:**	Configure the telemetry data for the subscription to stream to the specified IP address and port.

	Command or Action	Purpose
	`switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when resource events occur.
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the VXLAN Path to Poll for Events or Data

The vxlan path label provides information about the switch's Virtual Extensible LAN EVPNs, including VXLAN peers, VXLAN counters, VLAN counters, and BGP Peer data. You can configure this path label to gather telemetry information either periodically, or when events occur. See Telemetry Data Streamed for Native Data Source Paths, on page 67.

This path does not support filtering.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **vxlan environment**
5. **destination-group** *grp_id*
6. **ip address** *ip_addr* **port** *port*
7. **subscription** *sub_id*
8. **snsr-group** *sgrp_id* **sample-interval** *interval*
9. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(config-telemetry)# sensor-group 6` `switch(conf-tm-sensor)#`	Create a sensor group for telemetry data.
Step 4	**vxlan environment** **Example:** `switch(conf-tm-sensor)# vxlan environment` `switch(conf-tm-sensor)#`	Configure the vxlan path label, which enables telemetry data for multiple individual VXLAN objects to be sent to the label. The label consolidates the multiple data inputs into one output. Depending on the sample interval, the VXLAN data is either streaming based on the polling interval, or sent when events occur.
Step 5	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 6	**ip address** *ip_addr* **port** *port* **Example:** `switch(conf-tm-dest)# ip address 1.2.3.4 port 50004` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port.
Step 7	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 8	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when VXLAN events occur.

	Command or Action	Purpose
Step 9	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Verifying the Path Label Configuration

At any time, you can verify that path labels are configured, and check their values by displaying the running telemetry configuration.

SUMMARY STEPS

1. **show running-config-telemetry**

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**show running-config-telemetry** **Example:** `switch(conf-tm-sensor)# show running-config telemetry` `!Command: show running-config telemetry` `!Running configuration last done at: Mon Jun 10 08:10:17 2019` `!Time: Mon Jun 10 08:10:17 2019` `version 9.3(1) Bios:version` `feature telemetry` `telemetry` `  destination-profile` `    use-nodeid tester` `    sensor-group 4` `      path interface query-condition and(counters=[detailed],oper-speed=[1G,10G])` `    sensor-group 6` `      path interface query-condition oper-speed=[1G,40G]` `    subscription 6` `      snsr-grp 6 sample-interval 6000` ` nxosv2(conf-tm-sensor)#`	Displays the current running config for telemetry, In this example, sensor group 4 is configured to gather non-zero counters from interfaces running at 1 and 10 Gbps. Sensor group 6 is configured to gather all counters from interfaces running at 1 and 40 Gbps.

Displaying Path Label Information

Path Label Show Commands

Through the **show telemetry usability** commands, you can display the individual paths that the path label walks when you issue a query.

Command	Shows
show telemetry usability {all | environment | interface | resources | vxlan}	Either all telemetry paths for all path l specified path label. Also, the output sh data based on periodic polling or even For the interfaces path label, also any k configured.
show running-config telemetry	The running configuration for telemetr

Command Examples



Note The show telemetry usability all command is a concatenation of all the individual commands that are shown in this section.

The following shows an example of the **show telemetry usability environment** command.

```
switch# show telemetry usability environment
1) label_name : environment
path_name : sys/ch query_type : poll query_condition :
rsp-subtree=full&query-target=subtree&target-subtree-class=eqptPsuSlot,eqptFtSlot,eqptSupCSlot,eqptPsu,eqptFt,eqptSensor,eqptLCSlot
2) label_name : environment
path_name : sys/ch query_type : event query_condition :
[illegible]
 switch#
```

The following shows the output of the **show telemetry usability interface** command.

```
switch# show telemetry usability interface
1) label_name : interface
path_name : sys/intf query_type : poll query_condition :
query-target=children&query-target-filter=eq(l1PhysIf.adminSt,"up")&rsp-subtree=children&rsp-subtree-class=rmonEtherStats,rmonIfIn,rmonIfOut,rmonIfHCIn,rmonIfHCOut
2) label_name : interface
path_name : sys/mgmt-[mgmt0] query_type : poll query_condition :
query-target=subtree&query-target-filter=eq(mgmtMgmtIf.adminSt,"up")&rsp-subtree=full&rsp-subtree-class=rmonEtherStats,rmonIfIn,rmonIfOut,rmonIfHCIn,rmonIfHCOut
3) label_name : interface
path_name : sys/intf query_type : event query_condition :
[illegible]
ethpmEncRtdIf.operSt,"down")),and(updated(ethpmEncRtdIf.operSt),eq(ethpmEncRtdIf.operSt,"up"))))
4) label_name : interface
path_name : sys/mgmt-[mgmt0] query_type : event query_condition :
query-target=subtree&query-target-filter=or(or(deleted(),created()),or(and(updated(mgmtIf.operSt),eq(mgmtIf.operSt,"down")),and(updated(mgmtIf.operSt),eq(mgmtIf.operSt,"up"))))
 switch#
```

The following shows an example of the **show telemetry usability resources** command.

```
switch# show telemetry usability resources
  1) label_name          : resources
```

```
     path_name           : sys/proc
     query_type          : poll
     query_condition     : rsp-subtree=full&rsp-foreign-subtree=ephemeral

  2) label_name          : resources

     path_name           : sys/procsys
     query_type          : poll
     query_condition     :
query-target=subtree&target-subtree-class=procSystem,procSysCpuStats,procSysLoad,procSysProcTable,procSysProcEntry,procSysSched,procSysSchedEntry,procSysStat,procSysMem,procSysMemStats,procSysMemUsage,procSysMemFree

  3) label_name          : resources

     path_name           : sys/procsys/sysmem
     query_type          : event
     query_condition     :
query-target-filter=and(updated(procSysMem.memstatus),ne(procSysMem.memstatus,"OK"))

switch#
```

The following shows an example of the **show telemetry usability vxlan** command.

```
switch# show telemetry usability vxlan
  1) label_name          : vxlan

     path_name           : sys/bd
     query_type          : poll
     query_condition     : query-target=subtree&target-subtree-class=l2VlanStats

  2) label_name          : vxlan

     path_name           : sys/eps
     query_type          : poll
     query_condition     : rsp-subtree=full&rsp-foreign-subtree=ephemeral

  3) label_name          : vxlan

     path_name           : sys/eps
     query_type          : event
     query_condition     : query-target=subtree&target-subtree-class=nvoDyPeer

  4) label_name          : vxlan

     path_name           : sys/bgp
     query_type          : event
    query_condition     : query-target=subtree&query-target-filter=or(deleted(),created())

  5) label_name          : vxlan

     path_name           : sys/bgp
     query_type          : event
     query_condition     :
query-target=subtree&target-subtree-class=bgpDom,bgpPeer,bgpPeerAf,bgpDomAf,bgpPeerAfEntry,bgpOperRtctrlL3,bgpOperRtP,bgpOperRtEntry,bgpOperAfCtrl

switch#
```

Native Data Source Paths

About Native Data Source Paths

NX-OS Telemetry supports the native data source, which is a neutral data source that is not restricted to a specific infrastructure or database. Instead, the native data source enables components or applications to hook into and inject relevant information into the outgoing telemetry stream. This feature provides flexibility because the path for the native data source does not belong to any infrastructure, so any native applications can interact with NX-OS Telemetry.

The native data source path enables you to subscribe to specific sensor paths to receive selected telemetry data. The feature works with the NX-SDK to support streaming telemetry data from the following paths:

- RIB path, which sends telemetry data for the IP routes.
- MAC path, which sends telemetry data for static and dynamic MAC entries.
- Adjacency path, which sends telemetry data for IPv4 and IPv6 adjacencies.

When you create a subscription, all telemetry data for the selected path streams to the receiver as a baseline. After the baseline, only event notifications stream to the receiver.

Streaming of native data source paths supports the following encoding types:

- Google Protobuf (GPB)
- JavaScript Object Notation (JSON)
- Compact Google Protobuf (compact GPB)

Telemetry Data Streamed for Native Data Source Paths

For each source path, the following table shows the information that is streamed when the subscription is first created (the baseline) and when event notifications occur.

Path Type	Subscription Baseline	Event Notif

RIB	Sends all routes	Sends event not delete events. T through telemet • Next-hop r • Addre • Outgo • VRF r • Owne • Prefer • Metric • Tag fo • Segme • Tunne • Encap • Bitwis Type • For Layer- • VRF r • Route • Mask • Numb • Event Next hops
MAC	Executes a `GETALL` from DME for static and dynamic MAC entries	Sends event noti events. The foll telemetry for the • MAC addr • MAC addr • VLAN nur • Interface n • Event type Both static and event notificatio

Adjacency	Sends the IPv4 and IPv6 adjacencies	Sends event events. The telemetry fo • IP addr • MAC a • Interfac • Physica • VRF na • Prefere • Source • Addres Adjacency e

For additional information, refer to Github https://github.com/CiscoDevNet/nx-telemetry-proto.

Guidelines and Limitations

The native data source path feature has the following guidelines and limitations:

- For streaming from the RIB, MAC, and Adjacency native data source paths, sensor-path property updates do not support custom criteria like **depth**, **query-condition**, or **filter-condition**.
- Beginning with Cisco NX-OS Release 10.4(3)F, new query conditions are introduced to support sample-based subscription or updates-only support for RIB native path.

Configuring the Native Data Source Path for Routing Information

You can configure the native data source path for routing information, which sends information about all routes that are contained in the URIB. When you subscribe, the baseline sends all the route information. After the baseline, notifications are sent for route update and delete operations for the routing protocols that the switch supports. For the data sent in the RIB notifications, see Telemetry Data Streamed for Native Data Source Paths, on page 61.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path rib query-condition** [**data=ephemeral** | **updates_only**]

6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. subscription sub_id
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(conf-tm-sub)# sensor-grp 6` `switch(conf-tm-sub)#`	Create a sensor group.
Step 4	**data-source native** **Example:** `switch(conf-tm-sensor)# data-source native` `switch(conf-tm-sensor)#`	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	**path rib query-condition** [**data=ephemeral** | **updates_only**] **Example:** `nxosv2(conf-tm-sensor)# path rib` `nxosv2(conf-tm-sensor)#` **Example:** `nxosv2(conf-tm-sensor)# path rib query condition data=ephemeral` `nxosv2(conf-tm-sensor)#` **Example:** `nxosv2(conf-tm-sensor)# path rib query condition updates_only` `nxosv2(conf-tm-sensor)#`	Configure the RIB path which streams routes and route update information. **query condition data=ephemeral** (optional) - You can configure sample interval 0 or other than 0. This sample interval will determine how frequently the route information is sent to the destination periodically (at the configured sample interval). **query condition updates-only** (optional) - Supported only for sample interval 0. With this query condition, the initial snapshot data will not be sent, only the route information updates will be sent to the destination.
Step 6	**destination-group** *grp_id* **Example:**	Enter telemetry destination group submode and configure the destination group.

	Command or Action	Purpose
	`switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	
Step 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }** **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json` `switch(conf-tm-dest)#` `Example:` `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb` `switch(conf-tm-dest)#` `Example:` `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	subscription sub_id **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when rib events occur. **Note** Depending on the sample interval, the rib sensor path streams based on the polling interval.
Step 10	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Source Path for MAC Information

You can configure the native data source path for MAC information, which sends information about all entries in the MAC table. When you subscribe, the baseline sends all the MAC information. After the baseline,notifications are sent for add, update, and delete MAC address operations. For the data sent in the MAC notifications, see Telemetry Data Streamed for Native Data Source Paths, on page 67.

Note For update or delete events, MAC notifications are sent only for the MAC addresses that have IP adjacencies.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source native**
5. **path mac**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(conf-tm-sub)# sensor-grp 6` `switch(conf-tm-sub)#`	Create a sensor group.
Step 4	**data-source native** **Example:** `switch(conf-tm-sensor)# data-source native` `switch(conf-tm-sensor)#`	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	**path mac** **Example:** `nxosv2(conf-tm-sensor)# path mac` `nxosv2(conf-tm-sensor)#`	Configure the MAC path which streams information about MAC entries and MAC notifications.

	Command or Action	Purpose
Step 6	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }** **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Source Path for all MAC Information

You can configure the native data source path for MAC information, which sends information about all entries in the MAC table from Layer 3 and Layer 2. When you subscribe, the baseline sends all the MAC information.

After the baseline, notifications are sent for add, update, and delete MAC address operations. For the data sent in the MAC notifications, see Telemetry Data Streamed for Native Data Source Paths, on page 67.

Note For update or delete events, MAC notifications are sent only for the MAC addresses that have IP adjacencies.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source** *native*
5. **path mac-all**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(conf-tm-sub)# sensor-grp 6` `switch(conf-tm-sub)#`	Create a sensor group.
Step 4	**data-source** *native* **Example:** `switch(conf-tm-sensor)# data-source native` `switch(conf-tm-sensor)#`	Set the data source to native so that any native application can use the streamed data without requiring a specific model or database.
Step 5	**path mac-all** **Example:** `nxosv2(conf-tm-sensor)# path mac-all` `nxosv2(conf-tm-sensor)#`	Configure the MAC path which streams information about all MAC entries and MAC notifications.

	Command or Action	Purpose
Step 6	**destination-group** *grp_id* **Example:** `switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	Enter telemetry destination group submode and configure the destination group.
Step 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }** **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Configuring the Native Data Path for IP Adjacencies

You can configure the native data source path for IP adjacency information, which sends information about all IPv4 and IPv6 adjacencies for the switch. When you subscribe, the baseline sends all the adjacencies. Afterthe baseline, notifications are sent for add, update, and delete adjacency operations. For the data sent in the adjacency notifications, see Telemetry Data Streamed for Native Data Source Paths, on page 61.

Before you begin

If you have not enabled the telemetry feature, enable it now (**feature telemetry**).

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data-source** *native*
5. **path adjacency**
6. **destination-group** *grp_id*
7. **ip address ip_addr port port protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub_id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(conf-tm-sub)# sensor-grp 6` `switch(conf-tm-sub)#`	Create a sensor group.
Step 4	**data-source** *native* **Example:** `switch(conf-tm-sensor)# data-source native` `switch(conf-tm-sensor)#`	Set the data source to native so that any native application can use the streamed data.
Step 5	**path adjacency** **Example:** `nxosv2(conf-tm-sensor)# path adjacency` `nxosv2(conf-tm-sensor)#`	Configure the Adjacency path which streams information about the IPv4 and IPv6 adjacencies.
Step 6	**destination-group** *grp_id* **Example:**	Enter telemetry destination group submode and configure the destination group.

	Command or Action	Purpose
	`switch(conf-tm-sensor)# destination-group 33` `switch(conf-tm-dest)#`	
Step 7	**ip address ip_addr port port protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }** **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb` `switch(conf-tm-dest)#` **Example:** `switch(conf-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb-compact` `switch(conf-tm-dest)#`	Configure the telemetry data for the subscription to stream to the specified IP address and port and set the protocol and encoding for the data stream.
Step 8	**subscription** *sub_id* **Example:** `switch(conf-tm-dest)# subscription 33` `switch(conf-tm-sub)#`	Enter telemetry subscription submode, and configure the telemetry subscription.
Step 9	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(conf-tm-sub)# snsr-grp 6 sample-interval 5000` `switch(conf-tm-sub)#`	Link the sensor group to the current subscription and set the data sampling interval in milliseconds. The sampling interval determines whether the switch sends telemetry data periodically, or when interface events occur.
Step 10	**dst-group** *dgrp_id* **Example:** `switch(conf-tm-sub)# dst-grp 33` `switch(conf-tm-sub)#`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Displaying Native Data Source Path Information

Use the NX-OS **show telemetry event collector** commands to display statistics and counters, or errors for the native data source path.

Displaying Statistics

You can issue **show telemetry event collector stats** command to display the statistics and counters for each native data source path.

An example of statistics for the RIB path:

```
switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
```

```
1 4 Mon Jul 01 13:53:42.384 PST rib(1) switch#
An example of the statistics for the MAC path:
switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
1 3 Mon Jul 01 14:01:32.161 PST mac(1) switch#
An example of the statistics for the Adjacency path:
 switch# show telemetry event collector stats
------------------------------------------------------------------------------
 Row ID Collection Count Latest Collection Time Sensor Path(GroupId)
------------------------------------------------------------------------------
 1 7 Mon Jul 01 14:47:32.260 PST adjacency(1)
switch#
```

Displaying Error Counters

You can use the **show telemetry event collector stats** command to display the error totals for all the native data source paths.

```
switch# show telemetry event collector errors
------------------------------------------------------------------------------
 Error Description Error Count
------------------------------------------------------------------------------
 Dme Event Subscription Init Failures - 0
 Event Data Enqueue Failures - 0
 Event Subscription Failures - 0
 Pending Subscription List Create Failures - 0
 Subscription Hash Table Create Failures - 0
 Subscription Hash Table Destroy Failures - 0
 Subscription Hash Table Insert Failures - 0
 Subscription Hash Table Remove Failures - 0
switch#
```

Streaming Syslog

About Streaming Syslog for Telemetry

Beginning with Cisco NX-OS release 9.3(3), model-driven telemetry supports streaming of syslogs using YANG as a data source. When you create a subscription, all the syslogs are streamed to the receiver as a baseline. This feature works with the NX-SDK to support streaming syslog data from the following syslog paths:

- Cisco-NX-OS-Syslog-oper:syslog
- Cisco-NX-OS-Syslog-oper:syslog/messages

After the baseline, only syslog event notifications stream to the receiver. Streaming of syslog paths supports the following encoding types:

- Google Protobuf (GPB)
- JavaScript Object Notation (JSON)

Configuring the YANG Data Source Path for Syslog Information

You can configure the syslog path for syslogs, which sends information about all syslogs that are generated on the switch. When you subscribe, the baseline sends all the existing syslog information. After the baseline, notifications are sent for only for new syslogs that are generated on the switch.

Before you begin

If you have not enabled the telemetry feature, enable it now with the **feature telemetry** command.

SUMMARY STEPS

1. **configure terminal**
2. **telemetry**
3. **sensor-group** *sgrp_id*
4. **data source** *data-source-type*
5. **path Cisco-NX-OS-Syslog-oper:syslog/messages**
6. **destination-group** *grp_id*
7. **ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }**
8. **subscription** *sub-id*
9. **snsr-group** *sgrp_id* **sample-interval** *interval*
10. **dst-group** *dgrp_id*

DETAILED STEPS

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `switch# configure terminal` `switch(config)#`	Enter configuration mode.
Step 2	**telemetry** **Example:** `switch(config)# telemetry` `switch(config-telemetry)#`	Enter configuration mode for the telemetry features.
Step 3	**sensor-group** *sgrp_id* **Example:** `switch(conf-tm-sub)# sensor-grp 6` `switch(conf-tm-sub)#`	Create a sensor group.
Step 4	**data source** *data-source-type* **Example:** `switch(config-tm-sensor)# data source YANG`	Set the data source to YANG, so that it uses the native YANG streaming model to stream syslogs

	Command or Action	Purpose
Step 5	**path Cisco-NX-OS-Syslog-oper:syslog/messages** **Example:** `switch(config-tm-sensor)# path Cisco-NX-OS-Syslog-oper:syslog/messages`	Configure the syslog path which streams syslog generated on the switch.
Step 6	**destination-group** *grp_id* **Example:** `switch(config-tm-sensor)# destination-group 33`	Enter telemetry destination group sub-mode and configure the destination group.
Step 7	**ip address** *ip_addr* **port** *port* **protocol { HTTP | gRPC } encoding { JSON | GPB | GPB-compact }** **Example:** `switch(config-tm-dest)# ip address 192.0.2.11 port 50001 protocol http encoding json` **Example:** `switch(config-tm-dest)# ip address 192.0.2.11 port 50001 protocol grpc encoding gpb`	Configure the telemetry data for the subscription to stream to the specified IP address and port, and set the protocol and encoding for the data stream.
Step 8	**subscription** *sub-id* **Example:** `switch(config-tm-dest)# subscription 33`	Enter telemetry subscription submode and configure the telemetry subscription.
Step 9	**snsr-group** *sgrp_id* **sample-interval** *interval* **Example:** `switch(config-tm-sub)# snsr-group 6 sample-interval 0`	Link the sensor group to the current subscription and set the data sampling to 0 so that the switch sends telemetry data when syslog events occur. For interval , 0 is the only acceptable value.
Step 10	**dst-group** *dgrp_id* **Example:** `switch(config-tm-sub)# dst-grp 33`	Link the destination group to the current subscription. The destination group that you specify must match the destination group that you configured in the destination-group command.

Telemetry Data Streamed for Syslog Path

For each source path, the following table shows the information that is streamed when the subscription is first created "the baseline" and when event notifications occur.

Path	Subscription Baseline	Event Notificat

Cisco-NX-OS-Syslog-oper:syslog/messages	Stream all the existing syslogs from the switch.	Sends event switch: • messag • node-na • time-sta • time-of • time-zo • categor • messag • severity • text

Displaying Syslog Path Information

Use the Cisco NX-OS **show telemetry event collector** commands to display statistics and counters, or errors for the syslog path.

Displaying Statistics

You can enter the **show telemetry event collector stats** command to display the statistics and counters for each syslog path.

The following is an example of statistics for the syslog path:

```
------------------------------------------------------------------------------
Row ID         Collection Count  Latest Collection Time     Sensor Path(GroupId)
------------------------------------------------------------------------------
1            138             Tue Dec 03 11:20:08.200 PST Cisco-NX-OS-Syslog-oper:syslog(1)

2              138               Tue Dec 03 11:20:08.200 PST
Cisco-NX-OS-Syslog-oper:syslog/messages(1)
```

Displaying Error Counters

You can use the **show telemetry event collector errors** command to display the error totals for all the syslog paths.

```
switch(config-if)# show telemetry event collector errors

------------------------------------------------------------------------------
Error Description                                Error Count
------------------------------------------------------------------------------
Dme Event Subscription Init Failures                    - 0
Event Data Enqueue Failures                             - 0
Event Subscription Failures                             - 0
Pending Subscription List Create Failures               - 0
Subscription Hash Table Create Failures                 - 0
Subscription Hash Table Destroy Failures                - 0
Subscription Hash Table Insert Failures                 - 0
Subscription Hash Table Remove Failures                 - 0
```

Sample JSON Output

The following is a sample of JSON output:

```
172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
172.19.216.13 - - [03/Dec/2019 19:38:50] "POST
/network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages HTTP/1.0" 200 -
>>> URL            : /network/Cisco-NX-OS-Syslog-oper%3Asyslog%2Fmessages
>>> TM-HTTP-VER    : 1.0.0
>>> TM-HTTP-CNT    : 1
>>> Content-Type   : application/json
>>> Content-Length : 578
    Path => Cisco-NX-OS-Syslog-oper:syslog/messages
            node_id_str   : task-n9k-1
            collection_id : 40
            data_source   : YANG
            data          :
[
  [
    {
      "message-id": 420
    },
    {
      "category": "ETHPORT",
      "group": "ETHPORT",
      "message-name": "IF_UP",
      "node-name": "task-n9k-1",
      "severity": 5,
      "text": "Interface loopback10 is up ",
      "time-of-day": "Dec 3 2019 11:38:51",
      "time-stamp": "1575401931000",
      "time-zone": ""
    }
  ]
]
```

Sample KVGPB Output

The following is a sample KVGPB output

```
---Telemetry msg received @ 18:22:04 UTC


All the fragments:1 read successfully total size read:339

node_id_str: "task-n9k-1"
subscription_id_str: "1"
collection_id: 374
data_gpbkv {
  fields {
    name: "keys"
    fields {
      name: "message-id"
      uint32_value: 374
    }
  }

  fields {
    name: "content"
    fields {
      fields {
        name: "node-name"
        string_value: "task-n9k-1"
```

```
      }

      fields {
        name: "time-of-day"
        string_value: "Jun 26 2019 18:20:21"
      }

      fields {
        name: "time-stamp"
        uint64_value: 1574293838000
      }

      fields {
        name: "time-zone"
        string_value: "UTC"
      }

      fields {
        name: "process-name"
        string_value: ""
      }

      fields {
        name: "category"
        string_value: "VSHD"
      }

      fields {
        name: "group"
        string_value: "VSHD"
      }

      fields {
        name: "message-name"
        string_value: "VSHD_SYSLOG_CONFIG_I"
      }

      fields {
        name: "severity"
        uint32_value: 5
      }

      fields {
        name: "text"
        string_value: "Configured from vty by admin on console0"
      }
    }
  }
}
```

Troubleshooting Telemetry

-

Displaying Telemetry Log and Trace Information

Use the following NX-OS CLI commands to display the log and trace information.

show tech-support telemetry

This NX-OS CLI command collects the telemetry log contents from the tech-support log. In this example, the command output is redirected into a file in bootflash.

```
switch# show tech-support telemetry > bootflash:tmst.log
```

Additional References

Related Documents

Related Topic	Document Title
Example configurations of telemetry deployment for VXLAN EVPN.	*Telemetry Deployment for VXLAN EVPN*