# Telemetry Configuration Guide for Cisco NCS 6000 Series Routers, IOS XR Release 6.4.x

**First Published:** 2018-03-30

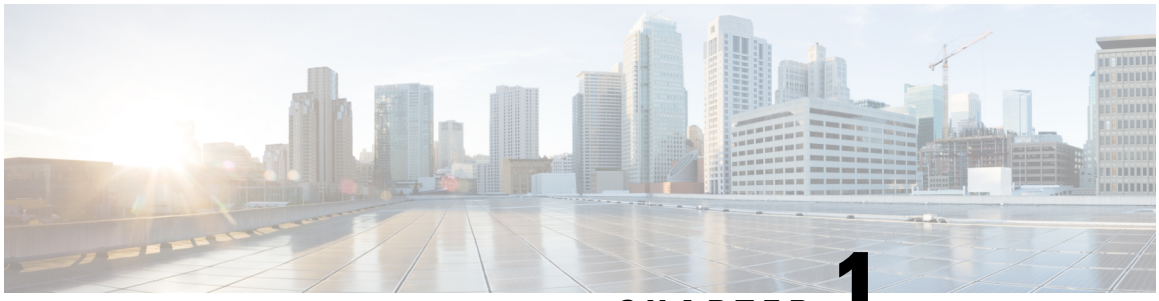
**Americas Headquarters**
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions and services, visit Cisco Marketplace.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

#### Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a web-based tool that acts as a gateway to the Cisco bug tracking system that maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. BST provides you with detailed defect information about your products and software.

# CONTENTS



CHAPTER 1

# New and Changed Feature Information

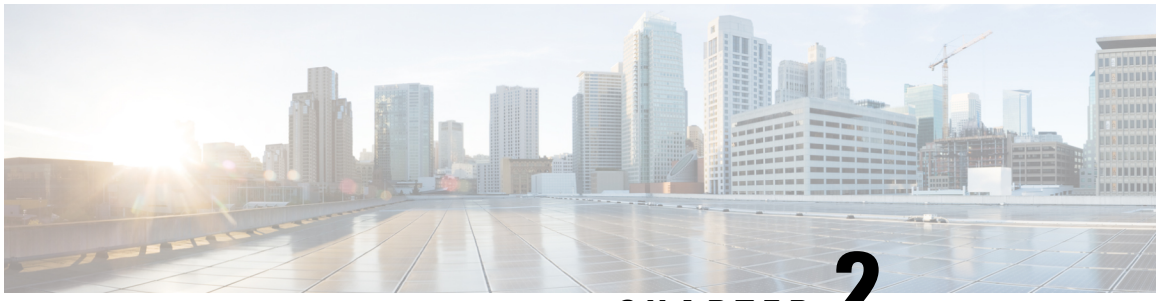
This section lists all the new and changed features for the *Telemetry Configuration Guide for Cisco NCS 6000 Series Routers*.

• New and Changed Telemetry Features, on page 1

## New and Changed Telemetry Features

**Note** *This software release has reached end-of-life status. For more information, see the* End-of-Life and End-of-Sale Notices.

| Feature | Description | Changed in Release | Where Documented |
|---|---|---|---|
| Support for telemetry dark bandwidth | Model-driven telemetry supports MPLS dark bandwidth with the polling interval reduced to 10 seconds. The dark bandwidth is the total interface bandwidth utilization in the data plane, excluding the RSVP-TE bandwidth. | Release 6.4.1 | *Configuring Automatic Bandwidth* topic in *Implementing MPLS Traffic Engineering chapter in MPLS Configuration Guide* |
| Support for Event-driven telemetry through NETCONF notification for Link Layer Discovery Protocol (LLDP) events | NETCONF client is configured to receive event notifications from a NETCONF server through a subscription. | Release 6.4.1 | *Configure Model-driven Telemetry* chapter - Example: Configure Event-driven Telemetry for LLDP, on page 7 |



CHAPTER 2

# Stream Telemetry Data

This document will help you understand the process of streaming telemetry data and its core components.



## Video: Telemetry in Cisco IOS XR

## Scope

Streaming telemetry lets users direct data to a configured receiver. This data can be used for analysis and troubleshooting purposes to maintain the health of the network. This is achieved by leveraging the capabilities of machine-to-machine communication.

The data is used by development and operations (DevOps) personnel who plan to optimize networks by collecting analytics of the network in real-time, locate where problems occur, and investigate issues in a collaborative manner.

## Need

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

IOS XR 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.

Network state indicators, network statistics, and critical infrastructure information are exposed to the application layer, where they are used to enhance operational performance and to reduce troubleshooting time. A push

model uses this capability to continuously stream data out of the network and notify the client. Telemetry enables the push model, which provides near-real-time access to monitoring data.

Streaming telemetry provides a mechanism to select data of interest from IOS XR routers and to transmit it in a structured format to remote management stations for monitoring. This mechanism enables automatic tuning of the network based on real-time data, which is crucial for its seamless operation. The finer granularity and higher frequency of data available through telemetry enables better performance monitoring and therefore, better troubleshooting. It helps a more service-efficient bandwidth utilization, link utilization, risk assessment and control, remote monitoring and scalability. Streaming telemetry, thus, converts the monitoring process into a Big Data proposition that enables the rapid extraction and analysis of massive data sets to improve decision-making.

# Benefits

Streamed real-time telemetry data is useful in:

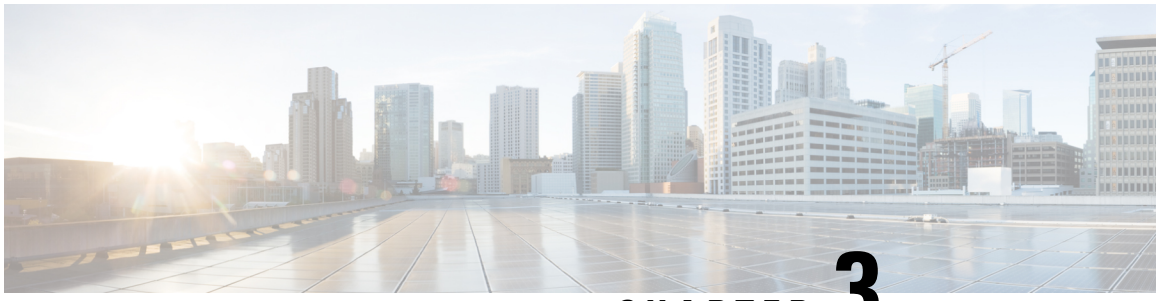
- **Traffic optimization:** When link utilization and packet drops in a network are monitored frequently, it is easier to add or remove links, re-direct traffic, modify policing, and so on. With technologies like fast reroute, the network can switch to a new path and re-route faster than the SNMP poll interval mechanism. Streaming telemetry data helps in providing quick response time for faster traffic.
- **Preventive troubleshooting:** Helps to quickly detect and avert failure situations that result after a problematic condition exists for a certain duration.

# Methods of Telemetry

Telemetry data can be streamed using these methods:

- **Model-driven telemetry:** provides a mechanism to stream data from an MDT-capable device to a destination. The data to be streamed is driven through subscription. There are two methods of configuration:
  - **Cadence-based telemetry:** Cadence-based Telemetry (CDT) continuously streams data (operational statistics and state transitions) at a configured cadence. The streamed data helps users closely identify patterns in the networks. For example, streaming data about interface counters and so on.
- **Policy-based telemetry:** streams telemetry data to a destination using a policy file. A policy file defines the data to be streamed and the frequency at which the data is to be streamed.

**Note** Model-driven telemetry supersedes policy-based telemetry.



CHAPTER 3

# Configure Model-based Telemetry

Streaming model-based telemetry data to the intended receiver involves:

- Configure Dial-out Mode, on page 5

# Configure Dial-out Mode

In a dial-out mode, the router initiates a session to the destinations based on the subscription.

All 64-bit IOS XR platforms (except for NCS 6000 series routers) support gRPC and TCP protocols. All 32-bit IOS XR platforms support only TCP.

For more information about the dial-out mode, see Dial-out Mode, on page 11.

The process to configure a dial-out mode involves:

## Create a Destination Group

The destination group specifies the destination address, port, encoding and transport that the router uses to send out telemetry data.

1. Identify the destination address, port, transport, and encoding format.
2. Create a destination group.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group <group-name>

Router(config-model-driven-dest)#address family ipv4 <IP-address> port <port-number>
Router(config-model-driven-dest-addr)#encoding <encoding-format>
Router(config-model-driven-dest-addr)#protocol <transport>
Router(config-model-driven-dest-addr)#commit
```

### Example: Destination Group for TCP Dial-out

The following example shows a destination group `DGroup1` created for TCP dial-out configuration with key-value Google Protocol Buffers (also called self-describing-gpb) encoding:

```
Router(config)#telemetry model-driven
Router(config-model-driven)#destination-group DGroup1
Router(config-model-driven-dest)#address family ipv4 172.0.0.0 port 5432
```

```
Router(config-model-driven-dest-addr)#encoding self-describing-gpb
Router(config-model-driven-dest-addr)#protocol tcp
Router(config-model-driven-dest-addr)#commit
```

# Create a Sensor Group

The sensor-group specifies a list of YANG models that are to be streamed.

1. Identify the sensor path for XR YANG model.
2. Create a sensor group.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group <group-name>
Router(config-model-driven-snsr-grp)# sensor-path <XR YANG model>
Router(config-model-driven-snsr-grp)# commit
```

**Example: Sensor Group for Dial-out**

**Note** gRPC is supported in only 64-bit platforms.

The following example shows a sensor group `SGroup1` created for dial-out configuration with the YANG model for interface statistics:

```
Router(config)#telemetry model-driven
Router(config-model-driven)#sensor-group SGroup1
Router(config-model-driven-snsr-grp)# sensor-path
Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/latest/generic-counters
Router(config-model-driven-snsr-grp)# commit
```

**What to Do Next:**

Create a subscription.

# Create a Subscription

The subscription associates a destination-group with a sensor-group and sets the streaming method.

A source interface in the subscription group specifies the interface that will be used for establishing the session to stream data to the destination. If both VRF and source interface are configured, the source interface must be in the same VRF as the one specified under destination group for the session to be established.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription <subscription-name>
Router(config-model-driven-subs)#sensor-group-id <sensor-group> sample-interval <interval>

Router(config-model-driven-subs)#destination-id <destination-group>
Router(config-model-driven-subs)#source-interface <source-interface>
Router(config-mdt-subscription)#commit
```

### Example: Subscription for Cadence-based Dial-out Configuration

The following example shows a subscription `Sub1` that is created to associate the sensor-group and destination-group, and configure an interval of 30 seconds to stream data:

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription Sub1
Router(config-model-driven-subs)#sensor-group-id SGroup1 sample-interval 30000
Router(config-model-driven-subs)#destination-id DGroup1
Router(config-mdt-subscription)# commit
```

# Validate Dial-out Configuration

Use the following command to verify that you have correctly configured the router for dial-out.

```
Router#show telemetry model-driven subscription <subscription-group-name>
```

### Example: Validation for TCP Dial-out

```
Router#show telemetry model-driven subscription Sub1
Thu Jul 21 15:42:27.751 UTC
Subscription:  Sub1                    State: ACTIVE
-------------
  Sensor groups:
  Id                Interval(ms)        State
  SGroup1           30000               Resolved

  Destination Groups:
  Id                Encoding            Transport   State   Port    IP
  DGroup1           self-describing-gpb tcp         Active  5432    172.0.0.0
```

## Example: Configure Event-driven Telemetry for LLDP

Telemetry supports NETCONF event notifications where the NETCONF client is configured to receive event notifications from a NETCONF server through a subscription. The NETCONF client must subscribe using a `create-subscription` request. Currently, only the events from Link Layer Discovery Protocol (LLDP) is supported. These event notifications are sent until either the NETCONF session or the subscription is terminated.

**Note** Configuring a sensor group and a subscription is not required for receiving NETCONF notifications. While sensor path and subscription configurations are required for receiving telemetry events, NETCONF `create-subscription` is required for receiving NETCONF notifications.

To generate NETCONF notifications:

1. Enable NETCONF agent and SSH sub system.

   ```
   ssh server netconf
   netconf-yang agent ssh
   ```

2. Enable model-driven telemetry.

   ```
   telemetry model-driven
   ```

3. Enable LLDP.

```
lldp
```

This example shows event-driven telemetry fo LLDP configuration data.

1. Create a destination group.

```
grpc
port 56782
address-family ipv4
!
telemetry model-driven
destination-group <destination-udp>
  address-family ipv4 <client-ip>1 port <udp port num>
   encoding self-describing-gpb
   protocol udp
  !
!
destination-group <destination-tcp>
  address-family ipv4 <client-ip> port <tcp port num>
   encoding gpb
   protocol tcp
  !
destination-group <destination-grpc>
  address-family ipv4 <grpc client ip>port <grpc port num>
   encoding self-describing-gpb
   protocol grpc no-tls
```

2. Create a sensor group.

```
sensor-group <sensor-group-name>
  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/global-lldp/lldp-info
  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/interfaces/interface
  sensor-path Cisco-IOS-XR-ethernet-lldp-oper:lldp/nodes/node/neighbors/details/detail

 !
```

3. Create a subscription.

```
subscription udp-out
  sensor-group-id <sensor-group-name> sample-interval 0
  destination-id <destination-udp>
!

subscription <subscription-name>
  sensor-group-id <sensor-group-name> sample-interval 0
  destination-id <destination-tcp>

subscription <subscription-name>
  sensor-group-id <sensor-group-name> sample-interval 0
!
netconf-yang agent
ssh
!
```

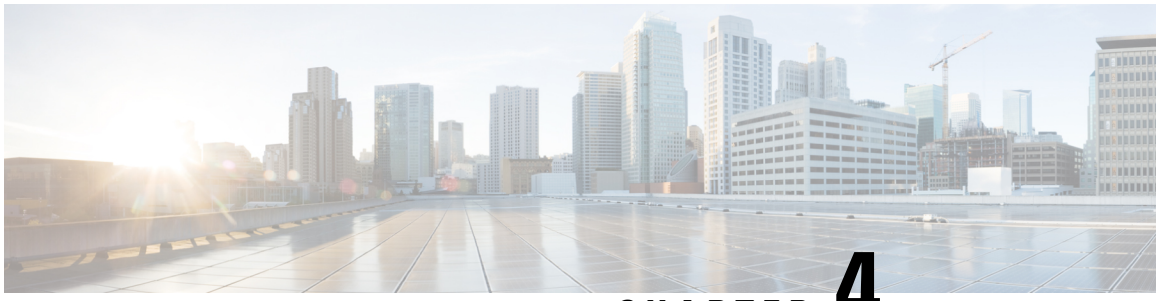
4. Set the notification to stream data when an event occurs.

```
Router(config-lldp)#timer 12
Router(config-lldp)#commit

Router(config-lldp)#holdtime 150
```

```
Router (config-lldp)#commit

Router (config-lldp)#exit
#506
<?xml version="1.0"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>Date-and-Time</eventTime>
 <lldp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ethernet-lldp-oper">
  <global-lldp>
   <lldp-info>
    <chassis-id>000b.1bc9.e700</chassis-id>
    <chassis-id-sub-type>4</chassis-id-sub-type>
    <system-name>ios</system-name>
    <timer>12</timer>
    <hold-time>120</hold-time>
    <re-init>2</re-init>
   </lldp-info>
  </global-lldp>
 </lldp>
</notification>
Ready to send a request.
Paste your request or enter 'get', 'get-config', 'create-sub', or 'bye' to quit):
```

5. Validate response received from NETCONF agent.

```
#506
<?xml version="1.0"?>
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
 <eventTime>Date-and-Time</eventTime>
 <lldp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ethernet-lldp-oper">
  <global-lldp>
   <lldp-info>
    <chassis-id>000b.1bc9.e700</chassis-id>
    <chassis-id-sub-type>4</chassis-id-sub-type>
    <system-name>ios</system-name>
    <timer>12</timer>
    <hold-time>150</hold-time>
    <re-init>2</re-init>
   </lldp-info>
  </global-lldp>
 </lldp>
</notification>
```



CHAPTER 4

# Core Components of Model-driven Telemetry Streaming

The core components used in streaming model-driven telemetry data are:



## Session

A telemetry session can be initiated using:

### Dial-out Mode

In a dial-out mode, the router dials out to the receiver. This is the default mode of operation. The router acts as a client and receiver acts as a server. In this mode, sensor-paths and destinations are configured and bound together into one or more subscriptions. The router continually attempts to establish a session with each destination in the subscription, and streams data to the receiver. The dial-out mode of subscriptions is persistent. When a session terminates, the router continually attempts to re-establish a new session with the receiver every 30 seconds.

## Sensor Path

The sensor path describes a YANG path or a subset of data definitions in a YANG model with a container. In a YANG model, the sensor path can be specified to end at any level in the container hierarchy.

An MDT-capable device, such as a router, associates the sensor path to the nearest container path in the model. The router encodes and streams the container path within a single telemetry message. A receiver receives data about all the containers and leaf nodes at and below this container path.

The router streams telemetry data for one or more sensor-paths, at the configured frequency (cadence-based streaming) to one or more receivers through subscribed sessions.

# Subscription

A subscription binds one or more sensor paths and destinations. An MDT-capable device streams data for each sensor path at the configured frequency (cadence-based streaming) to the destination.

The following example shows subscription `SUB1` that associates a sensor-group, sample interval and destination group.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#subscription SUB1
Router(config-model-driven-subs)#sensor-group-id SGROUP1 sample-interval 10000
Router(config-model-driven-subs)#strict-timer
```

**Note** With a `strict-timer` configured for the sample interval, the data collection starts exactly at the configured time interval allowing a more deterministic behavior to stream data.

In 32-bit platforms, `strict-timer` can be configured only under the subscription. Whereas, 64-bit platforms support configurtion at global level in addition to the subscription level. However, configuring at the global level will affect all configured subscriptions.

```
Router(config)#telemetry model-driven
Router(config-model-driven)#strict-timer
```

# Transport and Encoding

The router streams telemetry data using a transport mechanism. The generated data is encapsulated into the desired format using encoders.

Model-Driven Telemetry (MDT) data is streamed through :

- **Transmission Control Protocol (TCP):** used for only dial-out mode.
- **User Datagram Protocol (UDP):** used for only dial-out mode.

The data to be streamed can be encoded into Google Protocol Buffers (GPB) encoding. In GPB, the encoding can either be compact GPB (for optimising the network bandwidth usage) or self-describing GPB. The encodings supported are:

- **GPB encoding:** configuring for GPB encoding requires metadata in the form of compiled .proto files. A .proto file describes the GPB message format, which is used to stream data. The .proto files are available in the Github repository.
  - **Compact GPB encoding:** data is streamed in compressed and non self-describing format. A .proto file corresponding to each sensor-path must be used by the receiver to decode the streamed data.
  - **Key-value (KV-GPB) encoding:** data of each sensor path streamed is in a self-describing formatted ASCII text. A single .proto file `telemetry.proto` is used by the receiver to decode any sensor path data. Because the key names are included in the streamed data, the data on the wire is much larger as compared to compact GPB encoding.