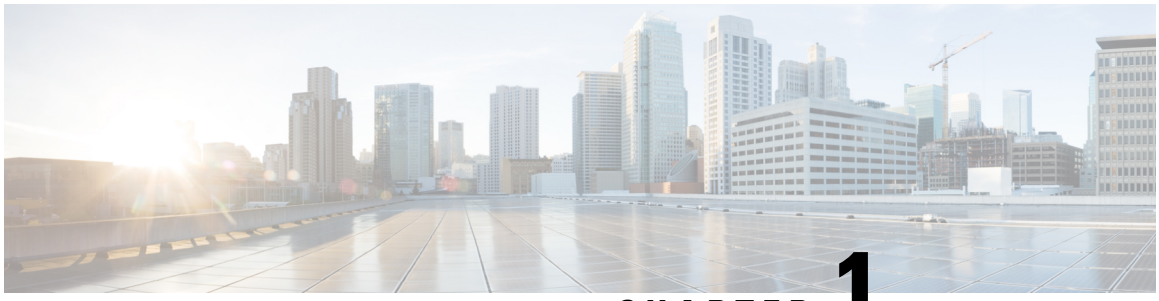
Cisco Catalyst 9800 Series Wireless Controller Programmability Guide

First Published: 2018-11-20

Last Modified: 2019-01-09

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

CONTENTS



CHAPTER 1

Cisco Catalyst 9800 Series Wireless Controller Programmability Guide

Information About Data Models

Introduction to Data Models - Programmatic and Standards-Based Configuration

The traditional way of managing network devices is by using Command Line Interfaces (CLIs) for configurational (configuration commands) and operational data (show commands). For network management, Simple Network Management Protocol (SNMP) is widely used, especially for exchanging management information between various network devices. Although CLIs and SNMP are heavily used, they have several restrictions. CLIs are highly proprietary, and human intervention is required to understand and interpret their text-based specification. SNMP does not distinguish between configurational and operational data.

The solution lies in adopting a programmatic and standards-based way of writing configurations to any network device, replacing the process of manual configuration. Network devices running on Cisco IOS XE support the automation of configuration for multiple devices across the network using data models. Data models are developed in a standard, industry-defined language, that can define configuration and state information of a network.

Cisco IOS XE supports the Yet Another Next Generation (YANG) data modeling language. YANG can be used with the Network Configuration Protocol (NETCONF) to provide the desired solution of automated and programmable network operations. NETCONF (RFC 6241) is an XML-based protocol that client applications use to request information from and make configuration changes to the device. YANG is primarily used to model the configuration and state data used by NETCONF operations.

In Cisco IOS XE, model-based interfaces interoperate with existing device CLI, Syslog, and SNMP interfaces. These interfaces are optionally exposed northbound from network devices. YANG is used to model each protocol based on RFC 6020.

Note To access Cisco YANG models in a developer-friendly way, clone the *GitHub repository*, and navigate to the *vendor/cisco* subdirectory. Models for various releases of IOS-XE, IOS-XR, and NX-OS platforms are available here.

Yang Data Models: Native Vs Open

Native Data Models

IOS XE native data models specific to IOS XE are not interoperable with other platforms. They closely mirror the structure of the IOS XE Command-line Interface (CLI), which makes them more familiar to experienced users of the IOS XE. The key benefit of native data models is the breadth of feature coverage.

The YANG modules for IOS XE are posted to the GitHub site at: https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

Modules from multiple versions, products, and vendors are also stored at this location.

Open Data Models

Open data models provide a common interface across multiple platforms. IOS XE supports a number of open data models from both the IETF and Open Config standards bodies.

Some of the supported open data models are listed below:

- IETF
- OpenConfig

The OpenConfig YANG models are are posted to the GitHub site at: https://github.com/openconfig/public/tree/master/release/models/system

NETCONF

NETCONF provides a mechanism to install, manipulate, and delete the configuration of network devices.

It uses an Extensible Markup Language (XML)-based data encoding for the configuration data as well as the protocol messages.

NETCONF uses a simple Remote Procedure Call (RPC) based mechanism to facilitate communication between a client and a server. The client can be a script or application running as part of a network manager. The server is typically a network device (switch or router). It uses Secure Shell (SSH) as the transport layer across network devices. It uses SSH port number 830 as the default port. The port number is a configurable option.

NETCONF also supports capability discovery and model downloads. Supported models are discovered using the *ietf-netconf-monitoring* model. Revision dates for each model are shown in the capabilities response. Data models are available for optional download from a device using the *get-schema* RPC. You can use these YANG models to understand or export the data model. For more details on NETCONF, see *RFC 6241*.

In releases prior to Cisco IOS XE Fuji 16.8.1, an operational data manager (based on polling) was enabled separately. In Cisco IOS XE Fuji 16.8.1 and later releases, operational data works on platforms running NETCONF (similar to how configuration data works), and is enabled by default. For more information on the components that are enabled for operational data queries or streaming, see the GitHub respository, to view **-oper* in the naming convention.

How to Configure Data Models

Configuring NETCONF

Follow the procedure given below to enable NETCONF-YANG on the device.

SUMMARY STEPS

1. **configure terminal**
2. **netconf-yang**
3. **exit**
4. **show platform software yang-management process**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**netconf-yang** **Example:** `Device(config)# netconf-yang`	Enables the NETCONF interface on the network device.
Step 3	**exit** **Example:** `Device(config)# exit`	Exits the global configuration mode.

Step 4	**show platform software yang-management process**
	Example:
	`Device(config)# show platform software yang-management process`
	
	`confd : Running`
	`nesd : Running`
	`syncfd : Running`
	`ncsshd : Running`
	`dmiauthd : Running`
	`nginx : Running`
	`ndbmand : Running`
	`pubd : Running`
		Verifies whether all the processes are running.
		If netconf-yang is running, all the processes except gnmib should be UP state.

	Command or Action	Purpose
	gnmib : Not Running	

NETCONF Operations

Table 1: NETCONF Operations

Operation	Description
<get>	Retrieves the running configuration information and device state.
<get-config>	Retrieves all or part of a specified configuration datastore.
<edit-config>	Edits a configuration datastore by creating, deleting, merging or replacing content.
<copy-config>	Copies an entire configuration datastore to another configuration one.
<delete-config>	Deletes a configuration datastore.
<lock>	Locks an entire configuration datastore of a device.
<unlock>	Releases a configuration datastore lock previously locked with the <lock> operation.
<close-session>	Graceful termination of a NETCONF session.
<kill-session>	Forcefully terminates a NETCONF session.

NETCONF <get> Request

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <get>
       <filter>
          <config-format-text-cmd>
             <text-filter-spec> | include interface </text-filter-spec>
          </config-format-text-cmd>
          <oper-data-format-text-block>
             <exec>show interfaces</exec>
             <exec>show arp</exec>
          </oper-data-format-text-block>
       </filter>
    </get>
 </rpc>]]>]]>
```

NETCONF <get> Response

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
       <cli-config-data>
<cmd>interface Loopback0</cmd>
<cmd>interface GigabitEthernet0/1</cmd>
<cmd>interface GigabitEthernet0/2</cmd>
```

```
</cli-config-data>
<cli-oper-data-block>
          <item>
             <exec>show interfaces</exec>
             <response>
                <!-- output of "show interfaces" ------>
             </response>
          </item>
          <item>
             <exec>show arp</exec>
             <response>
                <!-- output of "show arp" ------>
             </response>
          </item>
       </cli-oper-data-block>
    </data>
</rpc-reply>]]>]]>
```

NETCONF <get-config> Request

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
      <source>
         <running/>
      </source>
      <filter>
         <config-format-text-cmd>
            <text-filter-spec> | inc interface </text-filter-spec>
         </config-format-text-cmd>
</filter>
   </get-config>
</rpc>]]>]]>
```

NETCONF <get-config> Response

```
<?xml version="1.0" encoding=\"UTF-8\"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
      <cli-config-data>
         <cmd>interface FastEthernet0/1</cmd>
         <cmd>interface FastEthernet0/2</cmd>
      </cli-config-data>
   </data>
</rpc-reply>]]>]]>
```

NETCONF <edit-config> Request

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <edit-config>
      <target>
         <running/>
      </target>
      <config>
         <cli-config-data>
<cmd>hostname test</cmd>
            <cmd>interface fastEthernet0/1</cmd>
            <cmd>ip address 192.168.1.1 255.255.255.0</cmd>
</cli-config-data>
      </config>
   </edit-config>
</rpc>]]>]]>
```

NETCONF <edit-config> Response

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:netconf:base:1.0">
   <ok/>
</rpc-reply>]]>]]>
```

NETCONF <delete-config> Request

```
<?xml version="1.0"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <delete-config>
 <target>
 <running/>
 </target>
 </delete-config>
</rpc>]]>]]>
```

NETCONF <close-session> Request

```
<?xml version="1.0"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session/>
</rpc>]]>]]>
```

NETCONF <kill-session> Request

```
<?xml version="1.0"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <kill-session>
 <session-id>31990</session-id>
 </kill-session>
</rpc>]]>]]>
```

Configuring Cisco Catalyst 9800 Series Wireless Controller Using NETCONF-YANG

Table 2: YANG Models and Their Objects

Wireless Configuration Model	Objects	Description
Cisco-IOS-XE-wireless-ap-cfg.yang	• ap tag	Configuration of AP tags.
Cisco-IOS-XE-wireless-apf-cfg.yang	• apf	Configuration of AP functionalities.
Cisco-IOS-XE-wireless-cts-sxp-cfg.yang		
Cisco-IOS-XE-wireless-dot11-cfg.yang	• dot11ac-mcs-entries • dot11-entries	Configuration related to 802.11 feature.
Cisco-IOS-XE-wireless-fabric-cfg.yang	• fabric-profiles • fabric • fabric-controlplane-names	Configuration of SDA wireless.

Wireless Configuration Model	Objects	Description
Cisco-IOS-XE-wireless-flex-cfg.yang	• flex-policy-entries	Configuration of site.
Cisco-IOS-XE-wireless-fqdn-cfg.yang	• fqdn-configs	Configuration of fqdn url-filter list.
Cisco-IOS-XE-wireless-general-cfg.yang	• exclusion-list-file-records • na-server-cert • sim-l3-interface-cache-data • bcast-cfg	Configuration of general wireless parameters.
Cisco-IOS-XE-wireless-location-cfg.yang	• nmsp-config • location	Configuration of location parameters.
Cisco-IOS-XE-wireless-mesh-cfg.yang	• mesh • mesh-profiles • mesh-psks	Configuration of mesh.
Cisco-IOS-XE-wireless-mobility-cfg.yang	• mobility-config • mobility-groups • mode	Configuration of wireless mobility. **Note** When you configure the "wireless mobility peer ip" without public-ip, the public-ip is set to zeroes (0.0.0.0). This is due to a limitation in YANG model. However, you can configure public-ip by explicitly setting the public-ip parameter.
Cisco-IOS-XE-wireless-mstream-cfg.yang	• mstream-groups • mstream-glob	Configuration of wireless multicast.
Cisco-IOS-XE-wireless-rf-cfg.yang	• atf-policies • rf-tags • l2roam-rf-params • rf-profile • rf-profile-default-entries	Configuration of rf-profile.

Wireless Configuration Model	Objects	Description
Cisco-IOS-XE-wireless-rfid-cfg.yang	• rfid	Configuration of the notify threshold timer.
Cisco-IOS-XE-wireless-rogue-cfg.yang	• rogue-global • rogue-schedules • rogue-ap-cfg-entries • rogue-client-cfg-entries • rogue-ignore-data-entries • rule-data-entries	Configuration of rogue data.
Cisco-IOS-XE-wireless-rrm-cfg.yang	• rrms • rrm-mgr-cfg-entries	Configuration of RRM feature.
Cisco-IOS-XE-wireless-security-cfg.yang	• lsc-prov • lsc-provision-entries	Configuration of wireless security parameters.
Cisco-IOS-XE-wireless-site-cfg.yang	• ap-cfg-profiles • ap-packet-capture-profiles • site-tag-configs	Configuration of site.
Cisco-IOS-XE-wireless-wlan-cfg.yang	• wlan-cfg-entries • wlan-policies • policy-list-entries	Configuration of WLAN parameters and policies.

Information About Operational Data Parser Polling

Operational Data Overview

You can use YANG data models to read operational state data from a device. The operational data allows you to determine the current state and behavior of a device, similar to IOS **show** commands.

You can perform NETCONF GET operations to retrieve read-only operational state data from a system. You must enable NETCONF, activate data parsers (where applicable), and then retrieve the data through an appropriate YANG model.

The *How to Configure Operational Data* section provides information on configuring operational data through a programmable interface and the CLI.

Operational Data Parsers and Corresponding YANG Models

There are two types of operational data parsers; one that is always on, and the other that must be configured to poll operational data at regular intervals. For the first type of operational data parser, no configuration is required. Data is always fetched from the device during a NETCONF GET request. These data parsers do not have a polling-interval, and operational data is updated as soon as a change occurs.

The second type of operational data parsers must be activated either via the CLI or a NETCONF message (For more information, see the How to Enable Operational Data Parser Polling section.). The operational data for these types of parsers is polled at regular polling intervals and this information is retrieved during a NETCONF GET request.

The following table lists the data parsers that must be activated, and the corresponding YANG model where the operational data is stored.

Table 3: Operational Data Parsers to be Activated and Corresponding Yang Models

YANG Model to Access Operational Data	Objects	Description
Cisco-IOS-XE-wireless-access-point-oper.yang	• wireless-interface-data • qos-global-stats	Access point operator.
Cisco-IOS-XE-wireless-client-oper.yang	• latency-stats • global-stats • dot11-stats • dot1x-global-stats • sm-web-auth-stats	Wireless client operator.
Cisco-IOS-XE-wireless-fqdn-oper.yang		FQDN operator.
Cisco-IOS-XE-wireless-lisp-agent-oper.yang	• latency-stats • global-stats • dot11-stats • dot1x-global-stats • sm-web-auth-stats	LISP agent operator.
Cisco-IOS-XE-wireless-mcast-oper.yang	• mcast-config-info • bcast-cfg-op	Wireless multicast operator.
Cisco-IOS-XE-wireless-mesh-oper.yang	• mesh-oper-data	Mesh operator.
Cisco-IOS-XE-wireless-mobility-oper.yang	• global-stats	Global mobility operator.

YANG Model to Access Operational Data	Objects	Description
Cisco-IOS-XE-wireless-nmsp-oper.yang	• service-subscriptions • statistics	Network mobility services protocol operator.
Cisco-IOS-XE-wireless-rf-profile-oper.yang		RF profile operator.
Cisco-IOS-XE-wireless-rfid-oper.yang	• rfid-stats	Wireless RFID operator.
Cisco-IOS-XE-wireless-rogue-oper.yang	• rogue-stats • rldp-stats	Rogue operational data model operator.
Cisco-IOS-XE-wireless-rrm-oper.yang	• rrm-oper-data	Radio resource management operator.

How to Enable Operational Data Parser Polling

Enabling Operational Data Parser Polling Through a Programmable Interface

Perform this task to enable operational data parser polling through a programmable interface:

1. After enabling NETCONF-YANG, send an <edit-config> remote procedure call (RPC) using cisco-odm.yang (available in the GitHub Repository) to enable operational data polling. When the polling is enabled, all operational data parsers are activated by default. The default polling-interval of each parser is 120 seconds (120000 milliseconds). The polling interval decides the frequency at which the parser obtains the operational data and updates the corresponding YANG model in the datastore.
2. After operational data polling is enabled, send a <get> RPC to obtain the operational data. Use the parser-to-YANG model mapping to determine which operational YANG model should be used to retrieve the operational data. The following RPC reply fetches access control list (ACL) operational data using Cisco-IOS-XE-acl-oper.yang:

```
     CORRESPONDING RPC REPLY:
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
    <data>
        <access-lists xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-acl-oper">
            <access-list>
                <access-control-list-name>TEST</access-control-list-name>
                <access-list-entries>
                    <access-list-entry>
                        <rule-name>10</rule-name>
                        <access-list-entries-oper-data>
                        <match-counter>100</match-counter>
                    </access-list-entry>
                    <access-list-entry>
                        <rule-name>20</rule-name>
                        <access-list-entries-oper-data>
                        <match-counter>122</match-counter>
                    </access-list-entry>
```

```
                </access-list-entries>
            </access-list>
        </access-lists>
    </data>
</rpc-reply>
```

Note For more information, see the cisco-odm.yang model in the *GitHub repository*.

Information About Model-Driven Telemetry

Model-Driven Telemetry Overview

Telemetry is an automated communications process by which measurements and other data are collected at remote or inaccessible points and transmitted to the receiving equipment for monitoring. Model-driven telemetry provides a mechanism to stream YANG-modeled data to a data collector.

Applications can subscribe to specific data items they need, by using standard-based YANG data models over NETCONF, RESTCONF, or gRPC Network Management Interface (gNMI) protocols. Subscriptions can also be created by using CLIs if it is a configured subscription.

Structured data is published at a defined cadence, or on-change, based upon the subscription criteria and data type.

Subscription Overview

Subscriptions are items that create associations between telemetry roles, and define the data that is sent between them.

Specifically, a subscription is used to define the set of data that is requested as part of the telemetry data; when the data is required, how the data is to be formatted, and when not implicit, who (which receivers) should receive the data.

Even though the maximum number of supported subscriptions is platform-dependent, currently 100 subscriptions are supported. The subscriptions can be either configured or dynamic, and use any combination of transport protocols. If too many subscriptions are operating at the same time to allow all valid configured subscriptions to be active, the removal of an active subscription will cause one of the inactive but valid configured subscriptions to be attempted. Periodic triggered subscriptions (100 centiseconds is the default minimum) and on-change triggered subscriptions are supported.

NETCONF and other north-bound programmable interfaces (such as RESTCONF or gNMI) are supported to configure subscriptions.

There are two types of subscriptions used in telemetry on Cisco IOS XE systems: dynamic and configured subscriptions.

Because dynamic subscriptions are created by clients (the subscriber) that connect into the publisher, they are considered dial-in. Configured subscriptions cause the publisher to initiate connections to receivers, and as a result, they are considered dial-out.

Sample establish-subscription RPC

The following is a sample <establish-subscription> RPC. The stream, xpath-filter, and period fields in the RPC are mandatory.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<establish-subscription
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:yp="urn:ietf:params:xml:ns:yang:ietf-yang-push">
<stream>yp:yang-push</stream>
<yp:xpath-filter>/mdt-oper:mdt-oper-data/mdt-subscriptions</yp:xpath-filter>
<yp:period>1000</yp:period>
</establish-subscription>
</rpc>
```

YANG-Push

RFC 6020: YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF) and RFC 6241: Network Configuration Protocol (NETCONF) and explain YANG-push, which is the subscription and push mechanism for YANG databases. YANG-push subscriptions are defined using a data model. Using YANG-push, subscriber applications can request a continuous, customized stream of updates from YANG databases. The YANG-push encompasses all data in the configuration and operational databases that is described by the YANG model installed on a device. You must provide a filter for data, as subscription to all data is not supported.

Note The yang-push stream must be specified.

XPath Filter Support

Yang-Explorer is a web browser based tool that can access a network device, load the available data models and allow you to explore the data structures. It is available here: https://github.com/CiscoDevNet/yang-explorer

The dataset within the “yang-push” stream to be subscribed to is specified by the use of an XPath filter. However, XPath is far more expressive that is needed when specifying data for ‘yang-push’ streams, so the following limitations are placed on the XPath expression:

PARAMETER: XPATH-FILTER

filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic

The xpath filter is used to specify the information element to subscribe to. The xpath tells the telemetry parser where the data you want to subscribe to is located (within the target YANG data model). To build an xpath to the data you want to subscribe to, click down to the information element using or Yang-Explorer , and copy the value given in the ‘XPath’ section.

1. It must specify a single object. That object can be a container, a leaf, a leaf-list or a list.
2. It may have keys to specify a single entry in a list or container. The supported key specification syntax is “[{key name}={key value}]”. Compound keys are supported by the use of multiple key specifications. The key names and values must be exact; no ranges or wildcard values are supported.
3. The use of the union operator (|) is supported to allow a single subscription to support multiple objects.

Periodic and On-Change Publications

Periodic Subscriptions

With periodic subscriptions, the first push-update with the subscribed information is sent immediately; but it can be delayed if the device is busy or due to network congestion. Updates are then sent at the expiry of the configured periodic timer. For example, if the period is configured as 10 minutes, the first update is sent immediately after the subscription is created and every 10 minutes thereafter.

Period is time, in centiseconds (1/100 of a second), between periodic push updates. A period of 1000 will result in getting updates to the subscribed information every 10 seconds. The minimum period interval is 100, or one second. There is no default value. This value must be explicitly set in the <establish subscription> RPC.

Subscriptions for data that does not currently exist are permitted and run as normal subscriptions. When subscribing for empty data, empty update notifications are sent at the requested period.

Periodic updates contain a full copy of the subscribed data element or table.

On-Change Subscription

When an on-change subscription is created, the device pushes out the entire contents of the subscribed data to the collector. This initial message is called sync-on-start. It provides the current view of the database to the user.

However, unlike periodic subscriptions (which send updates at a fixed rate), an on-change subscription only sends an update when there is a change in the underlying data we are subscribed to (the ‘delta’). This use case is beneficial when we are subscribed to data that we don’t expect to change frequently. Instead of repeatedly sending the same information over the network, we receive only one update with the change in information; saving bandwidth and avoiding congestion.

RPC Support in Telemetry

You can send and receive YANG XML remote procedure calls (RPCs) in established NETCONF sessions.

The <establish-subscription>, and <delete-subscription> RPCs are supported for telemetry.

When an <establish-subscription> RPC is sent, the RPC reply from a publisher contains an <rpc-reply> message with a <subscription-result> element containing a result string.

The following table displays the response and the reason for the response in an <rpc-reply> message:

Result String	RPC	Cause
ok	<establish-subscription> <delete-subscription>	Success
error-no-such-subscription	<delete-subscription>	The specified subscription does not exist.
error-no-such-option	<establish-subscription>	The requested subscription is not supported.
error-insufficient-resources	<establish-subscription>	A subscription cannot be created because of the following reasons:

Result String	RPC	Cause
		• There are too many subscriptions. • The amount of data requested is too large. • The interval for a periodic subscription is too small.
error-other	<establish-subscription>	Some other error.

NETCONF Sessions in Telemetry

Telemetry subscriptions and updates are transmitted over NETCONF sessions. The NETCONF session that is used to establish a telemetry subscription receives the telemetry updates. If the NETCONF session is torn down or the connection is lost, associated telemetry subscriptions are also torn down.

All sessions are NETCONF sessions and as a result, all session limitations are specific to the NETCONF implementation.

High Availability in Telemetry

Dynamic telemetry connections are established over a NETCONF session via SSH to the active switch or a member in a switch stack, or the active route-processor in an high-availability capable device. After switchover, you must destroy and re-establish all sessions that use Crypto, including NETCONF sessions that carry telemetry subscriptions. You must also recreate all dynamic subscriptions after a switchover.

gRPC dial-out subscriptions are configured on the device as part of the running configuration of the active switch or member of the stack. When switchover occurs, existing connections to the telemetry receivers are torn down and reconnected (as long as there is still a route to the receiver). Subscriptions need not be reconfigured.

Note In the event of a device reload, subscription configurations must be synced to the start-up configuration of the device. This ensures that after the device reboots, subscription configurations remain intact on the device. Once the necessary processes are up and running, the device attempts to connect to the telemetry receiver and resume normal operations.

Sample Model-Driven Telemetry RPCs

Configured Subscription Management

This section describes how to create, modifiy, and delete configured subscriptions.

Receiving a Response Code

When a subscription is successfully created, the device responds with a subscription-result of notif-bis:ok and with a subscription ID. The following is a sample response RPC message for a dynamic subscription:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
<subscription-result xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
xmlns:notif-bis="urn:ietf:params:xml:ns:yang:ietf-event-notifications">notif-bis:
ok</subscription-result>
<subscription-id
xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications">2147484201</subscription-id>
</rpc-reply>
```

Receiving Subscription Push-Updates

Subscription updates pushed from the device are in the form of an XML RPC and are sent over the same NETCONF session on which these are created. The subscribed information element or tree is returned within the *datastore-contents-xml* tag. The following is a sample RPC message that provides the subscribed information:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:51.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483650</subscription-id>
        <datastore-contents-xml>
            <cpu-usage
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-process-cpu-oper"><cpu-utilization>
            <five-minutes>5</five-minutes></cpu-utilization></cpu-usage>
        </datastore-contents-xml>
    </push-update>
</notification>
```

If the information element to which a subscription is made is empty, or if it is dynamic (for example, a named access list) and does not exist, the periodic update will be empty and will have a self-closing *datastore-contents-xml* tag. The following is as sample RPC message in which the periodic update is empty:

```
<notification xmlns="urn:ietf:params:xml:ns:netconf:notification:1.0">
    <eventTime>2017-05-09T21:34:09.74Z</eventTime>
    <push-update xmlns="urn:ietf:params:xml:ns:yang:ietf-yang-push">
        <subscription-id>2147483649</subscription-id>
        <datastore-contents-xml />
    </push-update>
</notification>
```

Retrieving Subscription Details

You can retrieve the list of current subscriptions by sending a <get> RPC to the Cisco-IOS-XE-mdt-oper model. You can also use the **show telemetry ietf subscription** command to display the list of current subscriptions.

The following is a sample <get> RPC message:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get>
    <filter>
      <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
        <mdt-subscriptions/>
      </mdt-oper-data>
    </filter>
  </get>
</rpc>
```

The following is a sample RPC reply:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>2147485164</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <period>100</period>
          <xpath>/ios:native/router/ios-rip:rip/ios-rip:version</xpath>
        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
        <updates-in>0</updates-in>
        <updates-dampened>0</updates-dampened>
        <updates-dropped>0</updates-dropped>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

The following is sample output from the **show telemetry ietf subscription dynamic brief** command:

```
Device# show telemetry ietf subscription dynamic brief

Telemetry subscription brief

  ID               Type        State       Filter type
  ---------------------------------------------------
  2147483667       Dynamic     Valid       xpath
  2147483668       Dynamic     Valid       xpath
  2147483669       Dynamic     Valid       xpath
```

The following is sample output from the **show telemetry ietf subscription** *subscription-ID* **detail** command:

```
Device# show telemetry ietf subscription 2147483667 detail

Telemetry subscription detail:

  Subscription ID: 2147483667
  State: Valid
  Stream: yang-push
  Encoding: encode-xml
```

```
  Filter:
    Filter type: xpath
    XPath: /mdt-oper:mdt-oper-data/mdt-subscriptions
  Update policy:
    Update Trigger: periodic
    Period: 1000
  Notes:
```

The following is sample output from the **show telemetry ietf subscription all detail** command:

```
Device# show telemetry ietf subscription all detail

Telemetry subscription detail:

  Subscription ID: 101
  Type: Configured
  State: Valid
  Stream: yang-push
  Encoding: encode-kvgpb
  Filter:
    Filter type: xpath
    XPath: /iosxe-oper:ios-oper-db/hwidb-table
  Update policy:
    Update Trigger: on-change
    Synch on start: Yes
    Dampening period: 0
  Notes:
```

Retrieving Subscription Details Using RESTCONF

```
Subscription details can also be retrieved through a RESTCONF GET request to the
Cisco-IOS-XE-mdt-oper database:
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-oper: mdt-oper-data/mdt-subscriptions
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
Returned output:
{
  "Cisco-IOS-XE-mdt-oper:mdt-subscriptions": [
    {
      "subscription-id": 101,
      "base": {
        "stream": "yang-push",
        "encoding": "encode-kvgpb",
        "source-vrf": "",
        "no-synch-on-start": false,
        "xpath": "/iosxe-oper:ios-oper-db/hwidb-table"
      },
      "type": "sub-type-static",
      "state": "sub-state-valid",
      "comments": "",
      "updates-in": "0",
      "updates-dampened": "0",
      "updates-dropped": "0",
      "mdt-receivers": [
        {
          "address": "5.28.35.35",
          "port": 57555,
```

```
            "protocol": "grpc-tcp",
            "state": "rcvr-state-connecting",
            "comments": "Connection retries in progress",
            "profile": ""
          }
        ]
      }
    ]
}
```

Dynamic Subscription Control

This section describes how to create and delete dynamic subscriptions