Model-Driven Telemetry

Model-Driven Telemetry

Model-driven telemetry provides a mechanism to stream YANG-modelled data to a data collector. This module describes model-driven telemetry and provides sample telemetry RPCs.

Prerequisites for Model-Driven Telemetry

- Knowledge of YANG is needed to understand and define the data that is required when using telemetry.
- Knowledge of XML, XML namespaces, and XML XPath.
- Knowledge of standards and principles defined by the IETF telemetry specifications.
- The *urn:ietf:params:netconf:capability:notification:1.1* capability must be listed in hello messages. This capability is advertised only on devices that support IETF telemetry.
- NETCONF-YANG must be configured and running on the device.

Note NETCONF-YANG must be configured for telemetry to work, even if NETCONF is not used. For more information on configuring NETCONF-YANG, see the *NETCONF Protocol* module.

Verify that the following processes are running, by using the **show platform software yang-management process** command:

```
Device# show platform software yang-management process

confd : Running
nesd : Running
syncfd : Running
ncsshd : Running
dmiauthd : Running
nginx : Running
ndbmand : Running
pubd : Running
```

```
gnmib : Running
```

Note The process *pubd* is the model-driven telemetry process, and if it is not running, model-driven telemetry will not work.

The following table provides details about each of the Device Management Interface (DMI) processes.

Table 1: Field Descriptions

Device Management Interface Process Name	Primary Role
confd	Configuration daemon.
nesd	Network element synchronizer daemon.
syncfd	Sync daemon (maintains synchronization between the running state and corresponding models).
ncsshd	NETCONF Secure Shell (SSH) daemon.
dmiauthd	DMI authentication daemon.
nginx	NGINX web server. Acts as a web server for RESTCONF.
ndbmand	NETCONF database manager.
pubd	Publication manager and publisher used for model-driven telemetry.
gnmib	GNMI protocol server.

NETCONF-Specific Prerequisites

- Knowledge of NETCONF and how to use it, including:
  - Establishing a NETCONF session.
  - Sending and receiving hello and capabilities messages.
  - Sending and receiving YANG XML RPCs over the established NETCONF session. For more information, see the *Configure NETCONF/YANG and Validate Example for Cisco IOS XE 16.x Platforms* document.

Enabling and Validating NETCONF

The NETCONF functionality can be verified by creating an SSH connection to the device using a valid username and password and receiving a hello message, which contains the capability of the device:

```
Device:~ USER1$ ssh -s cisco1@172.16.167.175 -p 830 netconf
cisco1@172.16.167.175's password: cisco1
```

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<capabilities>
<capability>urn:ietf:params:netconf:base:1.0</capability>
<capability>urn:ietf:params:netconf:base:1.1</capability>
<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability
.
.
.
</capabilities>
<session-id>2870</session-id></hello>]]>]]>

Use < ^C > to exit
```

NETCONF is ready to use, when a successful reply is received in response to your hello message.

RESTCONF-Specific Prerequisites

- Knowledge of RESTCONF and how to use it (when creating a subscription using RESTCONF).
- RESTCONF must be configured on the device.
- RESTCONF must send correctly-formed Uniform Resource Identifiers (URIs) that adhere to RESTCONF *RFC 8040*.

Enabling and Validating RESTCONF

Validate RESTCONF using appropriate credentials and the following URI:

```
Operation: GET
Headers:
" Accept: application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
" Content-Type: application/yang-data+json
Returned Output (omitted for breverity):
{
    "ietf-restconf:data": {
        "ietf-yang-library:modules-state": {
            "module": [
                {
                    "name": "ATM-FORUM-TC-MIB",
                    "revision": "",
                    "schema":
"https://10.85.116.28:443/restconf/tailf/modules/ATM-FORUM-TC-MIB",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-FORUM-TC-MIB"
                },
                {
                    "name": "ATM-MIB",
                    "revision": "1998-10-19",
                    "schema":
"https://10.85.116.28:443/restconf/tailf/modules/ATM-MIB/1998-10-19",
                    "namespace": "urn:ietf:params:xml:ns:yang:smiv2:ATM-MIB"
                },
                {
                    "name": "ATM-TC-MIB",
                    "revision": "1998-10-19",
                    "schema": "https://10.85.116.28:443/restconf/tailf/
..
```

```
<snip>
..
}
```

RESTCONF is validated successfully when you receive the above reply with all device capabilities.

gRPC-Specific Prerequisites

- Set up a gRPC collector that understands key-value Google Protocol Buffers (GPB) encoding.

Restrictions for Model-Driven Telemetry

- Automatic hierarchy in selections is not supported for on-change subscriptions when using the *yang-push* stream. This means that when selecting a list, child lists of the list are not automatically included. For example, the subscriber must manually create a subscription for each child list.

  This restriction also applies to periodic subscriptions, if subscribed to the elements in the list below:

  - Cisco-IOS-XE-wireless-access-point-oper
  - Cisco-IOS-XE-wireless-ap-global-oper
  - Cisco-IOS-XE-wireless-awips-oper
  - Cisco-IOS-XE-wireless-client-global-oper
  - Cisco-IOS-XE-wireless-client-oper
  - Cisco-IOS-XE-wireless-general-cfg
  - Cisco-IOS-XE-wireless-general-oper
  - Cisco-IOS-XE-wireless-mesh-cfg
  - Cisco-IOS-XE-wireless-mesh-oper
  - Cisco-IOS-XE-wireless-mobility-oper
  - Cisco-IOS-XE-wireless-rfid-oper
  - Cisco-IOS-XE-wireless-rrm-emul-oper
  - Cisco-IOS-XE-wireless-rrm-global-oper
  - Cisco-IOS-XE-wireless-rrm-oper
  - Cisco-IOS-XE-wireless-site-cfg
  - bootcamp-test-autonomous
  - openconfig-access-points
  - openconfig-ap-manager
  - openconfig-lacp
  - openconfig-platform-psu

- Checking the authorization of data access is not supported. All the data requested by a subscriber is sent.

- Subtree filters are not supported. If subtree filters are specified, the subscription is marked as invalid.
- Defining multiple receivers within subscription parameters is not supported; only the first receiver destination is attempted. Other defined receivers are ignored.

gRPC-Specific Restrictions

- Transport Layer Security-based (TLS-based) authentication between a device and receiver is not supported.

  TLS-based authentication is supported in Cisco IOS XE Amsterdam 17.1.1 and later releases.

yang-push-Specific Restriction

- Subscription quality of service (QoS) is not supported.

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.

Telemetry Roles

In systems that use telemetry, there are different roles involved. This document uses the following telemetry roles:

- Publisher: Network element that sends the telemetry data.
- Receiver: Network element that receives the telemetry data. Also called the collector.
- Controller: Network element that creates subscriptions but does not receive the telemetry data. The telemetry data associated with the subscriptions it creates goes to receivers. Also called the management agent or management entity.
- Subscriber: Network element that creates subscriptions. While it technically does not have to also be the receiver, for the purposes of this document, both are the same.

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.

Dial-In and Dial-Out Model-Driven Telemetry

There are two flavors of model-driven telemetry: dial-in and dial-out.

Table 2: Dial-in and Dial-Out Model-Driven Telemetry

Dial-in (Dynamic)	Dial-out (Static or Configured)
Telemetry updates are sent to the initiator/subscriber.	Telemetry updates are sent to the specified receiver/collector.
Life of the subscription is tied to the connection (session) that created it, and over which telemetry updates are sent. No change in the running configuration is observed.	Subscription is created as part of the running configuration; it remains the device configuration till the configuration is removed.
Dial-in subscriptions need to be re-initiated after a reload, because established connections or sessions are killed during stateful switchover.	Dial-out subscriptions are created as part of the device configuration, and they automatically reconnect to the receiver after a stateful switchover.
Subscription ID is dynamically generated upon successful establishment of a subscription.	Subscription ID is fixed and configured on the device as part of the configuration.

Data Source Specifications

Sources of telemetry data in a subscription are specified by the use of a stream and a filter. The term stream refers to a related set of events. RFC 5277 defines an event stream as a set of event notifications matching some forwarding criteria.

Normally, the set of events from a stream are filtered. Different filter types are used for different stream types.

Cisco IOS XE supports two streams: *yang-push* and *yang-notif-native*.

Update Notifications

As part of a subscription, you can specify when the data is required; however this is stream-dependent. Some streams support making data available only when there is a change or an event within the stream. Other streams make data available when there is a change or at a defined time period.

The result of the *when* specification is a series of update notifications that carry the telemetry data of interest. How the data is sent is dependent on the protocol used for the connection between the publisher and the receiver.

Subscription Identifiers

Subscriptions are identified by a 32-bit positive integer value. The subscription ID for configured subscriptions is set by the controller, and for dynamic subscriptions is set by the publisher.

Controllers must limit the values they use for configured subscriptions in the range 0 to 2147483647, to avoid collisions with dynamic subscriptions created on the publisher. The dynamic subscription ID space is global, meaning that the subscription IDs for independently-created dynamic subscriptions do not overlap.

Subscription Management

Any form of management operation can be used to create, delete, and modify configured subscriptions. This includes both CLIs and network protocol management operations.

All subscriptions, both configured and dynamic, can be displayed using **show** commands and network protocol management operations.

The following table describes the supported streams and encodings along with the combinations that are supported. While streams-as-inputs is intended to be independent of the protocols-as-outputs, not all combinations are supported.

Table 3: Supported Combination of Protocols

Transport Protocol	**NETCONF**		**gRPC**		**gNMI**	
	Dial-In	Dial-Out	Dial-In	Dial-Out	Dial-In	Dial-Out
Stream						
yang-push	Yes	No	No	Yes	Yes	No
yang-notif-native	Yes	No	No	No	No	No
Encodings	XML	No	No	Key-value Google Protocol Buffers (kvGPB)	JSON_IETF	No

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: • 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.

Dynamic Subscription Control

This section describes how to create and delete dynamic subscriptions

Creating Dynamic Subscriptions

Dynamic subscriptions are created by subscribers that connect to the publisher and call for subscription creation using a mechanism within that connection, usually, a remote procedure call (RPC). The lifetime of the subscription is limited to the lifetime of the connection between the subscriber and the publisher, and telemetry data is sent only to that subscriber. These subscriptions do not persist if either the publisher or the subscriber is rebooted. You can create dynamic subscriptions by using the in-band <establish-subscription> RPC. The <establish-subscription> RPC is sent from an IETF telemetry subscriber to the network device. The stream, xpath-filter, and period fields in the RPC are mandatory.

Periodic Dynamic Subscriptions

The following is a sample periodic subscription:

```
<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>
```

On-Change Dynamic Subscription

The following is a sample on-change dynamic subscription over NETCONF:

```
<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>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</yp:xpath-filter>

    <yp:dampening-period>0</yp:dampening-period>
</establish-subscription>
```

Deleting Dynamic Subscriptions

You can delete dynamic subscriptions by using in-band <delete subscription> RPC, and disconnecting the transport session.

The <delete-subscription> RPC can be issued only by the subscriber, and it deletes only the subscriptions owned by that subscriber.

A subscription is also deleted when the parent NETCONF session is torn down or disconnected. If the network connection is interrupted, it may take some time for the SSH/NETCONF session to timeout, and subsequent subscriptions to be removed.

RPCs used to create and delete dynamic subscriptions using NETCONF are defined in *Custom Subscription to Event Notifications draft-ietf-netconf-subscribed-notifications-03* and *Subscribing to YANG datastore push updates draft-ietf-netconf-yang-push-07*.

Deleting Subscriptions using NETCONF <delete-Subscription> RPC

The following example shows how to delete a subscription using NETCONF:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <delete-subscription xmlns="urn:ietf:params:xml:ns:yang:ietf-event-notifications"
    xmlns:netconf="urn:ietf:params:xml:ns:netconf:base:1.0">
        <subscription-id>2147483650</subscription-id>
    </delete-subscription>
</rpc>
```

Configured Subscription Management

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

Creating Configured Subscriptions

Configured subscriptions are created by management operations on the publisher by controllers, and explicitly include the specification of the receiver of the telemetry data defined by the subscription. These subscriptions persist across reboots of the publisher.

Configured subscriptions can be configured with multiple receivers, however; only the first valid receiver is used. Connection to other receivers are not attempted, if a receiver is already connected or in the process of being connected. If that receiver is deleted, another receiver is connected.

This section displays sample RPCs to create configured subscriptions.

Periodic Subscription

The following sample RPC creates a periodic subscription using NETCONF that sends telemetry updates to the receiver every 60 seconds:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <period>6000</period>
     <xpath>/memory-ios-xe-oper:memory-statistics/memory-statistic</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
</rpc>
```

The following sample RPC creates a periodic subscription using RESTCONF:

```
URI:https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                    "period": "6000",
   "xpath": "/memory-ios-xe-oper:memory-statistics/memory-statistic"
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

On-change Subscription

The following sample RPC creates an on-change subscription using NETCONF that sends updates only when there is a change in the target database:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><edit-config>
 <target>
  <running/>
 </target>
 <config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
   <mdt-subscription>
    <subscription-id>200</subscription-id>
    <base>
     <stream>yang-push</stream>
     <encoding>encode-kvgpb</encoding>
     <no-synch-on-start>false</no-synch-on-start>
     <xpath>/cdp-ios-xe-oper:cdp-neighbor-details/cdp-neighbor-detail</xpath>
    </base>
    <mdt-receivers>
     <address>10.22.23.48</address>
     <port>57555</port>
     <protocol>grpc-tcp</protocol>
    </mdt-receivers>
   </mdt-subscription>
  </mdt-config-data>
 </config>
</edit-config>
</rpc>
```

The following sample RPC creates an on-change subscription using RESTCONF:

```
URI:
https://10.85.116.28:443/restconf/data/Cisco-IOS-XE-mdt-cfg:mdt-config-data
Headers:
application/yang-data.collection+json, application/yang-data+json,
application/yang-data.errors+json
Content-Type:
application/yang-data+json
BODY:
{
"mdt-config-data": {
 "mdt-subscription":[
 {
  "subscription-id": "102",
  "base": {
   "stream": "yang-push",
   "encoding": "encode-kvgpb",
                     "dampening period": "0",
   "xpath": "/cdp-ios-xe-oper:cdp-neighbor-details/cdp
                          -neighbor-detail "
  }
        "mdt-receivers": {
            "address": "10.22.23.48"
            "port": "57555"
        }
 }
 ]
}
}
```

Modifying Configured Subscriptions

There are two ways to modify configured subscriptions:

- Management protocol configuration operations, such as NETCONF <edit-config> RPC
- CLI (same process as creating a subscription)

Subscription receivers are identified by the address and port number. Receivers cannot be modified. To change the characteristics (protocol, profile, and so on) of a receiver, it must be deleted first and a new receiver created.

If a valid receiver configuration on a valid subscription is in the disconnected state, and the management wants to force a new attempt at setting up the connection to the receiver, it must rewrite the receiver with the exact same characteristics.

Deleting Configured Subscriptions

You can use the CLI or management operation to delete configured subscriptions. The **no telemetry ietf subscription** command removes the configured subscriptions. Configured subscriptions cannot be deleted using RPCs. These subscriptions are deleted through the configuration interface.

Deleting Subscriptions using the CLI

```
Device# configure terminal
Device(config)# no telemetry ietf subscription 101
Device(config)# end
```

Deleting Subscriptions using NETCONF

The following sample RPC shows how to delete a configured subscription:

```
<edit-config>
   <target>
     <running/>
   </target>
   <config>
     <mdt-config-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-cfg">
       <mdt-subscription  operation="delete">
       <subscription-id>102</subscription-id>
      </mdt-subscription>
     </mdt-config-data>
   </config>
 </edit-config>
```

Subscription Monitoring

Subscriptions of all types can be monitored by using CLIs and management protocol operations.

CLI

Use the **show telemetry ietf subscription** command to display information about telemetry subscriptions. The following is sample output from the 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:
```

NETCONF

The following is a NETCONF message that displays information about telemetry subscriptions:

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-subscriptions/>
</mdt-oper-data>
</filter>
</get>


* Enter a NETCONF operation, end with an empty line
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-subscriptions>
        <subscription-id>101</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-kvgpb</encoding>
          <source-vrf>RED</source-vrf>
          <period>10000</period>
          <xpath>/ios:native/interface/Loopback[name="1"]</xpath>
        </base>
        <type>sub-type-static</type>
        <state>sub-state-valid</state>
        <comments/>
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>57500</port>
          <protocol>grpc-tcp</protocol>
          <state>rcvr-state-connecting</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
      <mdt-subscriptions>
        <subscription-id>2147483648</subscription-id>
        <base>
          <stream>yang-push</stream>
          <encoding>encode-xml</encoding>
          <source-vrf/>
          <period>1000</period>
         <xpath>/if:interfaces-state/interface[name="GigabitEthernet0/0"]/oper-status</xpath>

        </base>
        <type>sub-type-dynamic</type>
        <state>sub-state-valid</state>
        <comments/>
```

```
        <mdt-receivers>
          <address>5.22.22.45</address>
          <port>51259</port>
          <protocol>netconf</protocol>
          <state>rcvr-state-connected</state>
          <comments/>
          <profile/>
          <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
        </mdt-receivers>
        <last-state-change-time>1970-01-01T00:00:00+00:00</last-state-change-time>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

Streams

A stream defines a set of events that can be subscribed to, and this set of events can be almost anything. However, as per the definition of each stream, all possible events are related in some way. This section describes the supported streams.

To view the set of streams that are supported use management protocol operations to retrieve the *streams* table from the Cisco-IOS-XE-mdt-oper module (from the YANG model Cisco-IOS-XE-mdt-oper.yang) in the *mdt-streams* container.

The following example shows how to use NETCONF to retrieve supported streams:

```
<get>
<filter>
<mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
<mdt-streams/>
</mdt-oper-data>
</filter>
</get>

* Enter a NETCONF operation, end with an empty line

<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
  <data>
    <mdt-oper-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-mdt-oper">
      <mdt-streams>
        <stream>native</stream>
          <stream>yang-notif-native</stream>
          <stream>yang-push</stream>
      </mdt-streams>
    </mdt-oper-data>
  </data>
</rpc-reply>
```

The example shows that three streams are supported: *native*, *yang-notif-native*, and *yang-push*. The stream *native* is not available for general use and can be ignored.

Note Currently there are no CLIs to return the list of supported streams.

yang-push Stream

The *yang-push* stream is the data in configuration and operational databases that is described by a supported YANG model. This stream supports an XPath filter to specify what data is of interest within the stream, and where the XPath expression is based on the YANG model that defines the data of interest.

Update notifications for this stream may be sent either when data changes or at fixed periods, but not for both, for a given subscription. Subscriptions for data that does not currently exist are permitted, and these run as normal subscriptions.

The only target database supported is *running*.

Determining On-Change Capability

Currently, there is *NO* indication within YANG models about the type of data that can be subscribed to, by using an on-change subscription. Attempts to subscribe to data that cannot be subscribed to by using on-change subscription results in a failure (dynamic) or an invalid subscription (configured).

IETF Draft Compliance

Telemetry using the *yang-push* stream is based on the IETF NETCONF working group's early drafts for telemetry. These are the following:

- *Custom Subscription to Event Notifications, Version 03*
- *Subscribing to YANG datastore push updates, Version 07*

The following features described in the drafts are not supported:

- Subtree filters
- Out-of-band notifications
- Any subscription parameter not explicitly stated as supported

X-Path Filter for yang-push

The dataset within the *yang-push* stream to be subscribed to should be specified by the use of an XPath filter. The following guidelines apply to the XPath expression:

- XPath expressions can have keys to specify a single entry in a list or container. The supported key specification syntax is

  ```
  [{key name}={key value}]
  ```

  The following is an example of an XPath expression:

  ```
  filter xpath
  /rt:routing-state/routing-instance[name="default"]/ribs/rib[name="ipv4-default"]/routes/route

  # VALID!
  ```

  Compound keys are supported by the use of multiple key specifications. Key names and values must be exact; no ranges or wildcard values are supported.

- In XPath expressions, select multiple keys using [] between the keys, and encapsulate the string with “. The following is an example of an XPath expression:

```
filter xpath
/environment-ios-xe-oper:environment-sensors/environment-sensor[location=\"Switch\ 1\"]
[name=\"Inlet\ Temp\ Sens\"]/current-reading
```

- XPath expressions support the use of the union operator (|) to allow a single subscription to support multiple objects. The union operator only works for NETCONF transport and not for gRPC.

Periodic Publication for yang-push

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.

The 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 that can be configured is 100, or one second. There is no default value. This value must be explicitly set in the <establish-subscription> RPC for dynamic subscriptions and in the configuration for configured subscriptions.

Periodic updates contain a full copy of the subscribed data element or table for all supported transport protocols.

When subscribing for empty data using a periodic subscription, empty update notifications are sent at the requested period. If the data comes into existence, its values at the next period are sent as a normal update notification.

On-Change Publication for yang-push

When creating an on-change subscription, the dampening period must be set to 0 to indicate that there is no dampening period; no other value is supported.

With on-change subscriptions, the first push update is the entire set of subscribed to data (the initial sychronization as defined in the IETF documents). This is not controllable. Subsequent updates are sent when the data changes, and consist of only the changed data. However, the minimum data resolution for a change is a row. So, if an on-change subscription is to a leaf within a row, if any item in that row changes, an update notification is sent. The exact contents of the update notification depend on the transport protocol.

In addition, on-change subscriptions are not hierarchical. That is, when subscribing to a container that has child containers, changes in the child container are not seen by the subscription.

Subscriptions for data that does not currently exist are permitted and run as normal subscriptions. The initial synchronization update notification is empty and there are no further updates until data is available.

XPath expressions must specify a single object. That object can be a container, a leaf, a leaf list or a list.

The yang-notif-native Stream

The *yang-notif-native* stream is any YANG notification in the publisher where the underlying source of events for the notification uses Cisco IOS XE native technology. This stream also supports an XPath filter that specifies which notifications are of interest. Update notifications for this stream are sent only when events that the notifications are for occur.

Since this stream supports only on-change subscriptions, the dampening interval must be specified with a value of 0.

XPath Filter for yang-notif-native

The dataset within the *yang-notif-native* stream to be subscribed to is specified by the use of an XPath filter. The following guideline applies to the XPath expression:

- XPath expressions must specify an entire YANG notification; attribute filtering is not supported.
- The union operator (|) is not supported.

Transport Protocol

The protocol that is used for the connection between a publisher and a receiver decides how the data is sent. This protocol is referred to as the transport protocol, and is independent of the management protocol for configured subscriptions. The transport protocol affects both the encoding of the data (for example XML, Google Protocol Buffers [GPB]) and the format of the update notification itself.

Note The stream chosen may also affect the format of the update notification.

Supported transport protocols are NETCONF and gRPC.

NETCONF Protocol

The NETCONF protocol is available for the transport of dynamic subscriptions only, and can be used with *yang-push* and *yang-notif-native* streams.

Three update notification formats are used when using NETCONF as the transport protocol:

- When the subscription uses the *yang-push* stream, and if it is periodic or when the initial synchronization update notification is sent on an on-change subscription.
- When the subscription uses the *yang-push* stream and it is an on-change subscription, other than the initial synchronization update notification.
- When the subscription uses the *yang-notif-native* stream.

yang-push Format

This format defines two formats of the update notifications, when the *yang-push* stream is sent over NETCONF as a transport with XML encoding is as defined in *draft-ietf-netconf-yang-push-07*. For more information, see section 3.7 of the IETF draft.

yang-notif-native Format

When the source stream is *yang-notif-native*, the format of the update notification when encoded in XML over NETCONF is as defined by *RFC 7950*. For more information, see section 7.16.2 of the RFC.

Unlike the formats for the *yang-push* stream, the subscription ID is not found in the update notification.

gRPC Protocol

The gRPC protocol is available only for the transport of configured subscriptions, and can be used with *yang-push* and *yang-notif-native* streams. Only kvGPB encoding is supported with gRPC transport protocol.

Receiver connection retries based on gRPC protocol (exponential back-off) are supported.

For telemetry messages defined in .proto files, see: *mdt_grpc_dialout.proto* and telemetry.proto.

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

Managing Configured Subscriptions

Use the **show platform software ndbman switch** {*switch-number* | **active** | **standby**} **models** command to display the list of YANG models that support on-change subscription.

Note Currently, you can only use the gRPC protocol for managing configured subscriptions.

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic**} *period*
7. **encoding encode-kvgpb**
8. **source-vrf** *vrf-id*
9. **source-address** *source-address*
10. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
11. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode.

	Command or Action	Purpose
	Example: `Device> enable`	• Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 101`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**stream yang-push** **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /memory-ios-xe-oper:memory-statistics/memory-statistic`	Specifies the XPath filter for the subscription.
Step 6	**update-policy** {**on-change** | **periodic**} *period* **Example:** `Device(config-mdt-subs)# update-policy periodic 6000`	Configures a periodic update policy for the subscription.
Step 7	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies kvGPB encoding.
Step 8	**source-vrf** *vrf-id* **Example:** `Device(config-mdt-subs)# source-address Mgmt-intf`	Configures the source VRF instance.
Step 9	**source-address** *source-address* **Example:** `Device(config-mdt-subs)# source-vrf 192.0.2.1`	Configures the source address.
Step 10	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* **Example:** `Device(config-mdt-subs)# receiver ip address 10.28.35.45 57555 protocol grpc-tcp`	Configures the receiver IP address, protocol, and profile for notifications.
Step 11	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

Configuring On-Change gRPC Subscriptions

SUMMARY STEPS

1. **enable**
2. **configure terminal**
3. **telemetry ietf subscription** *id*
4. **stream yang-push**
5. **filter xpath** *path*
6. **update-policy** {**on-change** | **periodic** *period*}
7. **encoding encode-kvgpb**
8. **receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name*
9. **end**

DETAILED STEPS

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device> enable`	Enables privileged EXEC mode. • Enter your password if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 8`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**stream yang-push** **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /iosxe-oper:ios-oper-db/hwidb-table`	Specifies the XPath filter for the subscription.
Step 6	**update-policy** {**on-change** | **periodic** *period*} **Example:** `Device(config-mdt-subs)# update-policy on-change`	Configures an on-change update policy for the subscription.
Step 7	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies kvGPB encoding.

	Command or Action	Purpose
Step 8	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* **Example:** `Device(config-mdt-subs)# receiver ip address 10.22.22.45 45000 protocol grpc_tls profile secure_profile`	Configures the receiver IP address, protocol, and profile for notifications.
Step 9	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

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": ""
          }
        ]
      }
    ]
  }
```


Additional References for Model-Driven Telemetry

Related Documents

Related Topic	Document Title
YANG Explorer	https://github.com/CiscoDevNet/yang-explorer

Standards and RFCs

Standard/RFC	Title
Custom Subscription to Event Notifications draft-ietf-netconf-subscribed-notifications-03	https://tools.ietf.org/id/draft-ietf-netconf-subscribed-notifications-03.txt
NETCONF Support for Event Notifications	draft-ietf-netconf-netconf-event-notifications-01
RFC 5277	NETCONF Event Notifications
RFC 6241	Network Configuration Protocol (NETCONF)
RFC 7950	The YANG 1.1 Data Modeling Language
RFC 8040	RESTCONF Protocol
Subscribing to Event Notifications	draft-ietf-netconf-rfc5277bis-01
Subscribing to YANG Datastore Push Updates	draft-ietf-netconf-yang-push-04
Subscribing to YANG datastore push updates draft-ietf-netconf-yang-push-07	https://tools.ietf.org/id/draft-ietf-netconf-yang-push-07.txt

Technical Assistance

Description	Link
The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies. To receive security and technical information about your products, you can subscribe to various services, such as the Product Alert Tool (accessed from Field Notices), the Cisco Technical Services Newsletter, and Really Simple Syndication (RSS) Feeds. Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.	http://www.cisco.com/support

Feature Information for Model-Driven Telemetry

The following table provides release information about the feature or features described in this module. This table lists only the software release that introduced support for a given feature in a given software release train. Unless noted otherwise, subsequent releases of that software release train also support that feature.

Use Cisco Feature Navigator to find information about platform support and Cisco software image support. To access Cisco Feature Navigator, go to www.cisco.com/go/cfn. An account on Cisco.com is not required.

Table 4: Feature Information for Model-Driven Telemetry

Feature Name	Release	Feature Information
Model-Driven Telemetry NETCONF Dial-In	Cisco IOS XE Everest 16.6.1	Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers. • Cisco Catalyst 3650 Series Switches • Cisco Catalyst 3850 Series Switches • Cisco Catalyst 9300 Series Switches • Cisco Catalyst 9500 Series Switches
	Cisco IOS XE Everest 16.6.2	• Cisco Catalyst 9400 Series Switches
	Cisco IOS XE Fuji 16.7.1	• Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers (ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X)
	Cisco IOS XE Fuji 16.8.1	• Cisco 1000 Series Integrated Services Routers • Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers
	Cisco IOS XE Fuji 16.8.1a	• Cisco Catalyst 9500-High Performance Series Switches
	Cisco IOS XE Fuji 16.9.1	• Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregation Services Router • Cisco cBR-8 Converged Broadband Router • Cisco Network Convergence System 4200 Series

Feature Name	Release	Feature Information
	Cisco IOS XE Gibraltar 16.9.2	• Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300L SKUs
	Cisco IOS XE Gibraltar 16.10.1	• Cisco Cloud Services Router 1000v • Cisco Network Convergence System 520 Series

Feature Name	Release	Feature Information
Model-Driven Telemetry gRPC Dial-Out	Cisco IOS XE Gibraltar 16.10.1	Configured subscriptions cause the publisher to initiate connections to receivers, and these connections are considered as dial-out. This feature was implemented on the following platforms: • Cisco 1000 Series Integrated Services Routers • Cisco 4000 Series Integrated Services Routers • Cisco ASR 1000 Series Aggregation Services Routers • Cisco ASR 900 Series Aggregation Services Routers • Cisco ASR 920 Series Aggregated Services Router • Cisco Catalyst 9200 and 9200L Series Switches • Cisco Catalyst 9300 and 9300L Series Switches • Cisco Catalyst 9400 Series Switches • Cisco Catalyst 9500 and 9500-High Performance Series Switches • Cisco cBR-8 Converged Broadband Router • Cisco Cloud Services Router 1000V Series • Cisco Network Convergence System 520 Series • Cisco Network Convergence System 4200 Series