



# Model-Driven Telemetry

- [Model-Driven Telemetry](#), on page 1

## Model-Driven Telemetry

Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers.

Applications can subscribe to specific data items they need, by using standard-based YANG data models over NETCONF-YANG.

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

## Prerequisites for Model-Driven Telemetry

- Knowledge of NETCONF-YANG and how to use it, including:
  - Establishing a NETCONF session.
  - Sending/receiving hello and capabilities messages.
  - Sending/receiving YANG XML remote procedure calls (RPCs) over the established NETCONF session. For more information, see the [Configuration Example for NETCONF-YANG](#).

For more information on NETCONF-YANG, see the *Datamodels* chapter.

- Knowledge of XML, XML namespaces, and XML [XPath](#).
- Knowledge of standards and principles defined by the IETF telemetry specification.
- NETCONF-YANG must be configured and running on the device. 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
```

```

vtyserverutild : Running
opdatamgrd     : Running
nginx          : Running
ndbmand        : Running
pubd           : Running

```



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

- The *urn:ietf:params:netconf:capability:notification:1.1* capability must be listed in the hello message. This capability is advertised only on devices that support IETF telemetry.

## 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 receiving equipment for monitoring. Model-driven telemetry provides a mechanism to stream data from a model-driven telemetry-capable device to a destination.

Telemetry uses a subscription model to identify information sources and destinations. Model-driven telemetry replaces the need for the periodic polling of network elements; instead, a continuous request for information to be delivered to a subscriber is established upon the network element. Then, either periodically, or as objects change, a subscribed set of YANG objects are streamed to that subscriber.

The data to be streamed is driven through subscriptions. Subscriptions allow applications to subscribe to updates (automatic and continuous updates) from a YANG datastore, and this enables the publisher to push and in effect stream those updates.



**Note** Dynamic subscriptions are only supported.

### 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 the 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 northbound programmable interfaces (such as RESTCONF or gNMI) are supported to configure subscriptions.

Two types of subscriptions are 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) 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

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 than what is needed when specifying data for the ‘yang-push’ streams, so the following limitations are placed on the XPath expression:

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. The union operator only works for NETCONF transport and not for gRPC.

## Periodic Publication

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.

## RPC Support in Telemetry

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: <ul style="list-style-type: none"> <li>• There are too many subscriptions.</li> <li>• The amount of data requested is considered too large.</li> <li>• The interval for a periodic subscription is too small.</li> </ul>
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 router. 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 subscriptions after a switchover.

## Sample Model-Driven Telemetry RPCs

### Creating a Subscription

Subscriptions are created using XML RPCs over an established NETCONF session. The `<establish-subscription>` RPC is sent from an IETF telemetry client or collector to the network device. The stream, xpath-filter, and period fields in the RPC are mandatory.

The following is a sample subscription to the operational database subscriptions table:

```
<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-subscriptions</yp:xpath-filter>
    <yp:period>1000</yp:period>
  </establish-subscription>
</rpc>
```

### 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:

```
<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 a 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>
      </mdt-subscriptions>
    </mdt-oper-data>
  </data>
</rpc-reply>

```

```

        <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:

```

## Deleting a Subscription

You can delete a telemetry subscription in two ways. One is by sending a `<delete-subscription>` RPC with the subscription ID in the subscription-id tag, which only a subscriber can do. Also, a subscription is 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.

```

<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>

```

## Additional References for Model-Driven Telemetry

### Related Documents

Related Topic	Document Title
YANG Explorer	<a href="https://github.com/CiscoDevNet/yang-explorer">https://github.com/CiscoDevNet/yang-explorer</a>

### Standards and RFCs

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

### Technical Assistance

Description	Link
<p>The Cisco Support website provides extensive online resources, including documentation and tools for troubleshooting and resolving technical issues with Cisco products and technologies.</p> <p>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.</p> <p>Access to most tools on the Cisco Support website requires a Cisco.com user ID and password.</p>	<a href="http://www.cisco.com/support">http://www.cisco.com/support</a>



## 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](http://www.cisco.com/go/cfn). An account on Cisco.com is not required.

Table 1: Feature Information for Model-Driven Telemetry

Feature Name	Release	Feature Information
Model-Driven Telemetry NETCONF Dial-In	Cisco IOS XE Everest 16.6.1	<p>Model-driven telemetry allows network devices to continuously stream real time configuration and operating state information to subscribers.</p> <ul style="list-style-type: none"> <li>• Cisco Catalyst 3650 Series Switches</li> <li>• Cisco Catalyst 3850 Series Switches</li> <li>• Cisco Catalyst 9300 Series Switches</li> <li>• Cisco Catalyst 9500 Series Switches</li> </ul>
	Cisco IOS XE Everest 16.6.2	<ul style="list-style-type: none"> <li>• Cisco Catalyst 9400 Series Switches</li> </ul>
	Cisco IOS XE Fuji 16.7.1	<ul style="list-style-type: none"> <li>• Cisco 4000 Series Integrated Services Routers</li> <li>• Cisco ASR 1000 Series Aggregation Services Routers (ASR1001-HX, ASR1001-X, ASR1002-HX, ASR1002-X)</li> </ul>
	Cisco IOS XE Fuji 16.8.1	<ul style="list-style-type: none"> <li>• Cisco 1000 Series Integrated Services Routers</li> <li>• Cisco ASR 1000 RP2 and RP3 Series Aggregation Services Routers</li> </ul>
	Cisco IOS XE Fuji 16.8.1a	<ul style="list-style-type: none"> <li>• Cisco Catalyst 9500-High Performance Series Switches</li> </ul>
	Cisco IOS XE Fuji 16.9.1	<ul style="list-style-type: none"> <li>• Cisco ASR 900 Series Aggregation Services Routers</li> <li>• Cisco ASR 920 Series Aggregation Services Router</li> <li>• Cisco cBR-8 Converged Broadband Router</li> <li>• Cisco Network Convergence System 4200 Series</li> </ul>

Feature Name	Release	Feature Information
	Cisco IOS XE Gibraltar 16.9.2	<ul style="list-style-type: none"><li>• Cisco Catalyst 9200 and 9200L Series Switches</li><li>• Cisco Catalyst 9300L SKUs</li></ul>

