

# Enhancements to Data Models

This section provides an overview of the enhancements made to data models.


- Improved YANG Input Validator and Get Requests, on page 2
- NETCONF Accounting Logs, on page 4
- Enhancements to Sensor Paths, on page 6
- YANG-Push, on page 8
- OpenConfig Data Model Enhancements, on page 12
- Install Label in oc-platform Data Model, on page 13
- OAM for MPLS and SR-MPLS in mpls-ping and mpls-traceroute Data Models, on page 15
- OpenConfig YANG Model:SR-TE Policies, on page 20
- Aggregate Prefix SID Counters for OpenConfig SR YANG Module, on page 21
- OpenConfig YANG Model:MACsec, on page 21
- OpenConfig YANG Model:AFT, on page 27
- Automatic Resynchronization of OpenConfig Configuration, on page 31

# Improved YANG Input Validator and Get Requests

*Table 1: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Improved YANG Input Validator and Get Requests | Release 7.10.1 | Introduced in this release on: NCS 5500 fixed port routers; NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards; NCS 5700 line cards [Mode: Compatibility; Native])<br><br>The OpenConfig data models provide a structure for managing networks via YANG protocols. With this release, enhancements to the configuration architecture improve input validations and ensure that the Get requests made through gNMI or NETCONF protocols return only explicitly configured OpenConfig leaves.<br><br>Previously, Get requests returned all the items in the Cisco native data models that the system could convert into OpenConfig items, regardless of whether they were initially configured via OpenConfig. We have added a new legacy mode option for a limited number of releases which helps you preserve this behaviour. |

In IOS XR Software Release 7.10.1, the following are the enhancements to improve YANG Input Validator and Get Requests:

- Get requests made via NETCONF or gNMI now return only OpenConfig leaves that were configured using OpenConfig models.

  Use the legacy mode as follows:

  NETCONF: Add a legacy mode attribute to the **get-config** request tag,

  Example: **get-config xmlns:xr-md=”http://cisco.com/ns/yang/cisco-xr-metadata” xr-md:mode="legacy"**

  gNMI: Set the origin to **openconfig-legacy**.

- Improved input validation for OpenConfig configurations to provide a more consistent experience across the schema.

The new validation includes enhanced error reporting, though some errors may include references to XR configuration schema paths and item values in the message string.

- OpenConfig leaves now return default values consistently.

  Get requests use the **Explicit Basic Mode** (refer RFC6243) to return only the OpenConfig leaves that were explicitly configured.

### Usage Guidelines and Limitations

In this release, the following usage guidelines and limitations apply based on the following functionalities:

- Upgrades to Cisco IOS XR Software Release 7.10.1 and later will not show OpenConfig leaves in Get requests until OpenConfig has been successfully committed.
- Similarly, downgrading from Release 7.10.1 to an earlier version and then upgrading back to Release 7.10.1 will not show OpenConfig leaves in Get requests until OpenConfig has been successfully committed.
- Each feature must be fully configured using OpenConfig or Cisco native data model or CLI.

  If configuration items applied to a feature via OpenConfig are overridden by configuring those items directly via Cisco native data model, this will not be reflected in the system view of currently configured OpenConfig items.

  Use the Cisco native data model to configure features not supported by OpenConfig data model.
- Use either gNMI or NETCONF to manage configuration via OpenConfig. We recommend not to use both the management agents on the same device simultaneously.

  Once a successful commit has been made using gNMI or NETCONF, that management agent is considered the **active agent**.

  OpenConfig items cannot be configured by the non-active agent. However, the non-active agent can configure Cisco native data model items and perform Get requests on any configuration items.

  All OpenConfig leaves must first be removed by the active agent before a different agent can be used.
- During the commit process (which can take many minutes for large changesets), Get requests can be made on the running datastore.

  Other request types like, Edit request, Commit request from other clients, and Get request on the candidate datastore of another client are rejected.
- When ACLs are configured via OpenConfig, CLI actions such as resequencing ACLs and copying ACLs will not be reflected in the system view of the current OpenConfig configuration.
- Configuration modifications made by Config Scripts to features configured through OpenConfig will not be reflected in the system view of the current OpenConfig configuration which is returned from Get-config operations.
- Configuration removal from the system may occur as a result of some events, such as install operations and startup configuration failures during line card insertion.

  OpenConfig items currently configured do not reflect this change. In such cases, a syslog will be generated to remind the user to manually apply OpenConfig configurations to the system.
- All OpenConfig will be removed from the system when a **Commit Replace** operation is performed using the CLI.

- By using the **show running-config | (xml | json) openconfig** command, you can still view the running OpenConfig. However, you cannot filter the view using XR CLI configuration keywords.

**Note** Starting from Cisco IOS XR Release 24.4.1, the **show running-config | (xml | json) openconfig**command is not supported.

- The **load rollback changes** and **load commit changes** commands are not supported for rollback or commit that include OpenConfig leaves.

# NETCONF Accounting Logs

***Table 2: Feature History Table***

| Feature Name | Release Information | Description |
|---|---|---|
| Accounting Records for NETCONF Operations | Release 7.6.1 | Depending on the accounting configuration command you use, every NETCONF operation that the router performs is reported to the local server as syslog messages or remote AAA servers like TACACS+ as accounting messages, or both. |

With this feature, you can view the accounting logs of all NETCONF operations such as `edit-config`, `get-config`, `get` operations that are performed on the router. The logs include the following data:

- RPC name
- Commit ID
- Session ID
- Message ID
- XPath

For more information, see *Implementing System Logging* chapter in the *System Monitoring Configuration Guide for Cisco NCS 5500 Series Routers*.

To enable NETCONF accounting logs, do the following steps:

### Procedure

**Step 1** Enter the configuration mode.

**Example:**

```
Router#conf t
```

**Step 2** Create a method list for accounting.

**Example:**

```
Router(config)#aaa accounting commands default start-stop group tacacs+ local
```

Use one or both of the method list value to enable system accounting.

- **TACACS+**—The logs are stored on the TACACS+ server.
- **Local**—The logs are stored in a user-specified file on the router. The maximum file size is 2047 MB.

**Step 3** Commit the configuration.

**Example:**

```
Router(config)#commit
```

**Note**
Syslog message about start and end of the session with details such as session ID, user, and remote address information is displayed for NETCONF operations only when both the EXEC accounting and local command accounting is enabled.

```
Router(config)#aaa accounting exec default start-stop group tacacs+
Router(config)#aaa accounting commands default start-stop local
```

---

**Example**

NETCONF Accounting Logs

With the RPC commit operation, the configuration changes are reported in the form of CLI commands. In this example, the `edit-config` operation is converted into its equivalent CLI `aaa accounting system default start-stop none` command in the logs; the user ID and session ID details are logged.

```
RP/0/RP0/CPU0:Mar 15 17:04:34.950 UTC: locald_DLRSC[233]: %SECURITY-LOCALD-6-LOCAL_CMD_ACCT
 :
 RPC CMD: "aaa accounting system default start-stop none" by <user> from TTY
netconf-3745105668
10.0.0.1 rpc_name commit rpc_commitid 808464433 rpc_sessid 3745105668
rpc_msgid 6ed74d71-1eda-4757-a4d6-8223b6fca588
```

For other RPCs, the data is reported in the form of XPaths. In this example, the NETCONF operation does not report equivalent CLI command. The RPC name is recorded in the logs.For syslogs with length greater than 400 characters, the log is split into two entries. Here, the XPathis split for brevity

```
RP/0/RP0/CPU0:Mar 15 30 18:39:45.412 UTC: locald_DLRSC[418]: %SECURITY-LOCALD-6-LOCAL_CMD_ACCT
 :
RPC CMD: rpc_name get by <user> from TTY netconf-921603460 10.0.0.1 rpc_sessid 921603460
rpc_msgid
101 xpath
Cisco-IOS-XR-wdsysmon-fd-proc-oper:process-monitoring/nodes/node[node-name=0/RP0/CPU0]/
process-name/proc-cpu-utilizations/proc-cpu-utilization[process-name=packet]Cisco-IOS-XR-pmengine-oper:
performance management/ethernet/ethernet-ports/ethernet-port/ethernet-current/ethernet-secon


RP/0/RP0/CPU0:Mar 15 18:39:45.412 UTC: locald_DLRSC[418]: %SECURITY-LOCALD-6-LOCAL_CMD_ACCT
 :
RPC CMD: d30/second30-ethersCisco-IOS-XR-pmengine-oper:performance-management/otu/otu-ports/
otu-port/otu-current/otu-minute15/otu-minute15fecsCisco-IOS-XR-wdsysmon-fd-proc-oper:process-monitoring/
nodes/node[node-name=0/RP0/CPU0]/process-name/proc-cpu-utilizations/proc-cpu-utilization[process-name=raw_ip]
```

**TACACS+ Logs:** The following example shows the logs from a TACACS+ server:

```
Commit changes:

Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=834
service=shell priv-lvl=0 commit_start=2021/10/11 22:56:19.882 commit_id=1000000022 rpc_
sessid=29961779 rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=835
service=shell priv-lvl=0 cmd=interface GigabitEthernet0/0/0/2 <cr> commit_id=1000000022
rpc_sessid=29961779 rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=836
service=shell priv-lvl=0 cmd= description test <cr> commit_id=1000000022 rpc_sessid=29961779

rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=837
service=shell priv-lvl=0 cmd= mtu 1600 <cr> commit_id=1000000022 rpc_sessid=29961779
rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=838
service=shell priv-lvl=0 cmd= ipv4 address 5.6.7.8 255.255.255.0 route-tag 100 <cr>
commit_id=1000000022
rpc_sessid=29961779 rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:24 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=839
service=shell priv-lvl=0 cmd= shutdown <cr> commit_id=1000000022 rpc_sessid=29961779
rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:25 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=840
service=shell priv-lvl=0 cmd=! <cr> commit_id=1000000022 rpc_sessid=29961779
rpc_msgid=101 rpc_name=commit
Tue Mar 15 15:56:25 2022 192.0.2.254 root netconf-29961779 192.0.2.1 stop timezone=UTC
task_id=841
service=shell priv-lvl=0 commit_end=2021/10/11 22:56:20.471 commit_id=1000000022
rpc_sessid=29961779 rpc_msgid=101 rpc_name=commit

Get-config:

Tue Mar 15 15:05:47 2022 192.0.2.254 root netconf-1616743444 192.0.2.1 stop timezone=UTC
task_id=519
service=shell priv-lvl=0 rpc_sessid=1616743444 rpc_msgid=101 rpc_name=get-config
rpc_xpath= /Cisco-IOS-XR-ifmgr-cfg:interface-configurations
```

# Enhancements to Sensor Paths

This section provides an overview about the sensor paths introduced or enhanced across Cisco IOS XR releases.

*Table 3: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Telemetry Support for OpenConfig Interfaces, IPv4 and IPv6 Addresses and State | Release 7.4.2 | This feature provides telemetry GNMI and GRPC support for the following `openconfig-if-ip.yang` sensor paths. Previously, only NETCONF edit-config, get-config and get operations were supported. With this new feature, telemetry polling at a cadence or on-change can be retrieved for IPv4 and IPv6 data.<br>• /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface/ipv6/<br>• addresses/address[ip]/state/ip<br>• addresses/address[ip]/state/prefix-length<br>• addresses/address[ip]/state/origin<br>• state/enabled<br>• state/mtu<br>• state/dup-addr-detect-transmits<br>• state/counters/in-pkts<br>• state/counters/in-octets<br>• state/counters/out-pkts<br>• state/counters/out-octets<br>• state/openconfig-if-ip-ext:autoconf/create-global-addresses<br>• /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface/ipv4/<br>• addresses/address[ip]/state/ip<br>• addresses/address[ip]/state/prefix-length<br>• addresses/address[ip]/state/origin<br>• state/mtu<br>• state/dhcp-client<br>• state/in-pkts<br>• state/in-octets<br>• state/out-pkts<br>• state/out-octets<br>You can access this data model from the Github repository. |

# YANG-Push

YANG-Push is a telemetry mechanism that:

- enables the application to subscribe to updates from a YANG datastore
- allows the server to push continuous, real-time updates to the subscriber, and
- reduces the need for inefficient polling to monitor operational state.

***Table 4: Feature History Table***

| Feature Name | Release Information | Feature Description |
|---|---|---|
| YANG-Push | Release 25.3.1 | Introduced in this release on: NCS 5500 fixed port routers; NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards; NCS 5700 line cards [Mode: Native])<br><br>YANG-Push provides a real-time telemetry solution by allowing applications to subscribe to specific YANG datastore updates. This feature enables efficient, low-latency streaming of operational state data to subscribed receivers. By reducing the reliance on traditional polling methods, YANG-Push enhances network observability, accelerates troubleshooting, and optimizes data collection for modern network automation and assurance workflows. |

### Integration of IETF telemetry

YANG-Push notifications bring the IETF version of telemetry (as specified in RFC 8641) into Cisco IOS-XR. This approach is not an entirely new way of doing telemetry in IOS-XR. Instead, the functionality of YANG-Push is very similar to the existing telemetry features already available in IOS-XR. The main difference is that YANG-Push standardizes telemetry operations according to IETF specifications, facilitating broader interoperability and integration with third-party and multi-vendor tools. The subscription and push model for delivering operational state data remains consistent with the established IOS-XR telemetry mechanism.

# Guidelines for YANG push configuration

- Only UDP-Notification transport is supported.

# Configure YANG-Push subscriptions

### Before you begin

- Verify that the receiver's IP address and port are reachable from the device.

### Procedure

**Step 1** Define a receiver for YANG-Push notifications.

**Example:**

```
Router(config)# yang-push receiver London1
Router(config-yangpush-receiver)# address 192.168.1.0
Router(config-yangpush-receiver)# port 44444
Router(config-yangpush-receiver)# transport udp-notif [max-segment-size <size>]
```

**Step 2** Create a YANG-Push subscription.

Some of these are optional: module, dscp, purpose and source-interface.

When you specify the `module` for a subscription, the device makes sure the first part of your filter path comes from that module. It ignores the rest of the path for module checking.

**Example:**

```
Router(config)# yang-push subscription 12345
Router(config-yangpush-subs)# receiver example1
Router(config-yangpush-subs)# filter xpath /interfaces/interface
Router(config-yangpush-subs)# module ietf-interfaces
Router(config-yangpush-subs-module)# revision 2018-02-20
Router(config-yangpush-subs)# encoding json
Router(config-yangpush-subs)# update-policy on-change sync-on-start
Router(config-yangpush-subs)# dscp 32
Router(config-yangpush-subs)# purpose "Monitor interface status"
Router(config-yangpush-subs)# source-interface GigabitEthernet0/0
```

**Step 3** Verify yang-push subscription

Use the **show yang-push subscriptions** commands to verify subscriptions. If no subscription ID is specified, then the status for all subscriptions is shown.

**Example:**

```
Router# show yang-push subscriptions
Subscription 12345
------------------
 Status:      Valid
 Purpose:     purpose-string
 Receivers:
   Name: London1
     Status:    Active
     Address:   1.1.1.1
     Port:      44444
     Transport: UDP-Notif (MSS: 1514)
 XPath:       Cisco-IOS-XR-pfi-im-cmd-oper:interfaces
 Modules:     Cisco-IOS-XR-pfi-im-cmd-oper@2021-08-27
 Encoding:    JSON
 DSCP:        12
 Source Intf: Loopback28
```

```
 Update:      Periodic, every 30.00s from 2024-08-03T12:34:56.789Z

Subscription 35153
------------------
 Status:      Valid
 Receivers:
   Name: NewYork7
     Status:    Active
     Address:   192.168.1.0
     Port:      12345
     Transport: UDP-Notif (MSS: 1514)
  Name: Bombay3 (Mumbai3)
     Status:    Connecting
     Address:   2.3.4.5
     Port:      2345
     Transport: UDP-Notif (MSS: 1514)
 XPath:       Cisco-IOS-XR-infra-syslog-oper:syslog/messages/message
 Encoding:    JSON
 Update:      On-change, with sync on subscription start

Subscription 45678
------------------
 Status:      Invalid (Invalid Module Version or Revision)
              The revisions of the following installed Yang models do not
              match the configured revisions:
                Cisco-IOS-XR-infra-syslog-oper@2020-02-08
              No data will be sent until the configured revisions are updated.
 Receivers:
   Name: NewYork7
     Status:    Disconnected
     Address:   192.168.1.0
     Port:      12345
     Transport: UDP-Notif (MSS: 1514)
 XPath:       Cisco-IOS-XR-infra-syslog-oper:syslog/messages/message
 Revisions:   2019-04-05
 Encoding:    JSON
 Update:      On-change
Subscription 55555
------------------
 Status:      Valid
              Warning: One or more receivers for this subscription have not been
              fully configured.
 Receivers:
   Name: AWS-South (Missing)
   Name: NewYork3
     Status:    Disconnected
                This receiver is not fully configured.
                No data will be sent until all missing fields are configured.
     Address:   <missing>
     Port:      <missing>
     Transport: <missing>
 XPath:       Cisco-IOS-XR-pfi-im-cmd-oper:interfaces
 Encoding:    JSON
 Update:      Periodic, every 42949672.95s

Subscription 65432
------------------
 Status:      Invalid (Missing Configuration)
              This subscription is not fully configured.
              No data will be sent until all missing fields are configured.
 Receivers:   <missing>
 XPath:       <missing>
 Encoding:    <missing>
 Transport:   <missing>
```

```
 Update:      <missing>
```

**Step 4** Show a brief summary of all subscriptions

Use the **show yang-push subscriptions brief** command to display a summary of all subscriptions.

**Example:**

```
Router#show yang-push subscriptions brief
Status (S): V - Valid, I - Invalid
Receiver Status (RS): A - Active, C - Connecting, D - Disconnected
Sub  | S | Path                                                | Upd.          | Receivers
 | RS
------+---+-----------------------------------------------------+--------------+----------------------+----
12345 | V | Cisco-IOS-XR-pfi-im-cmd-oper:interfaces           | 30.00s       | London1
  | A
35153 | V | ...S-XR-infra-syslog-oper:syslog/messages/message | On-chg       | NewYork7
  | A
     |   |                                                   |              | Mumbai3
 | C
45678 | I | ...S-XR-infra-syslog-oper:syslog/messages/message | On-chg       | NewYork7
  | D
55555 | V | Cisco-IOS-XR-pfi-im-cmd-oper:interfaces           | 42949672.95s | AWS-South
  | D
     |   |                                                   |              | NewYork3
 | D
65432 | I | -                                                 | -            | -
  | -
```

**Step 5** Verify YANG-Push receiver status.

Use the **show yang-push receivers** command to display the current status of YANG-Push receivers. If no receiver name is specified, the status for all receivers is shown.

**Example:**

```
Router#show yang-push receivers

Receiver London1
----------------
 Address:     1.1.1.1
 Port:        44444
 Transport:   UDP-Notif (MSS: 1514)
 Subscriptions: 12345 (Active)
```

The device begins streaming YANG-Push updates based on your configuration.

# OpenConfig Data Model Enhancements

*Table 5: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| LACP OpenConfig Model | Release 7.5.3 | Use the `openconfig-lacp.yang` data model to manage Link Aggregation Control Protocol ( LACP) aggregate interfaces by monitoring the number of LACP timeouts and the time since the last timeout.<br><br>With this release, the data model is revised from version 1.1.0 to 1.2.0 to introduce the following sensor paths for the operational state of the bundle member interface `lacp/interfaces/interface[name]/members/member[interface]/state/`:<br><br>• `last-change`<br><br>• `counters/lacp-timeout-transitions`<br><br>You can stream Event-driven telemetry data for the time since the last change of a timeout, and Model-driven telemetry data for the number of times the state has transitioned with a timeout. The state change is monitored since the time the device restarted or the interface was brought up, whichever is most recent. |
| Revised OpenConfig MPLS Model to Version 3.0.1 for Streaming Telemetry | Release 7.3.3 | The OpenConfig MPLS data model provides data definitions for Multiprotocol Label Switching (MPLS) configuration and associated signaling and traffic engineering protocols. In this release, the following data models are revised for streaming telemetry from OpenConfig version 2.3.0 to version 3.0.1:<br><br>• openconfig-mpls<br><br>• openconfig-mpls-te<br><br>• openconfig-mpls-rsvp<br><br>• openconfig-mpls-igp<br><br>• openconfig-mpls-types<br><br>• openconfig-mpls-sr<br><br>You can access this data model from the Github repository. |

# Install Label in oc-platform Data Model

*Table 6: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Enhancements to openconfig-platform YANG Data Model | Release 7.3.2 | The openconfig-platform YANG data model provides a structure for querying hardware and software router components via the NETCONF protocol. This release delivers an enhanced openconfig-platform YANG data model to provide information about:<br>• software version<br>• golden ISO (GISO) label<br>• committed IOS XR packages<br>You can access this data model from the Github repository. |

The openconfig-platform (oc-platform.yang) data model is enhanced to provide the following data:

- IOS XR software version (optionally with GISO label)
- Type, description, operational status of the component. For example, a CPU component reports its utilization, temperature or other physical properties.
- List of the committed IOS XR packages

To retrieve oc-platform information from a router via NETCONF, ensure you configured the router with the SH server and management interface:

```
Router#show run
Building configuration...
!! IOS XR Configuration version = 7.3.2
!! Last configuration change at Tue Sep  7 16:18:14 2016 by USER1
!
......
......
netconf-yang agent ssh
ssh server netconf vrf default
interface MgmtEth 0/RP0/CPU0/0
    no shut
    ipv4 address dhcp
```

The following example shows the enhanced `OPERATING_SYSTEM` node component (line card or route processor) of the oc-platform data model:

```
<component>
<name>IOSXR-NODE 0/RP0/CPU0</name>
<config>
<name>0/RP0/CPU0</name>
```

```
</config>
<state>
<name>0/RP0/CPU0</name>
<type xmlns:idx="http://openconfig.net/yang/platform-types">idx:OPERATING_SYSTEM</type>
<location>0/RP0/CPU0</location>
<description>IOS XR Operating System</description>
<software-version>7.3.2</software-version> ----------------------> Label Info
<removable>true</removable>
<oper-status xmlns:idx="http://openconfig.net/yang/platform-types">idx:ACTIVE</oper-status>
</state>
<subcomponents>
 <subcomponent>
  <name><platform>-af-ea-7.3.2v1.0.0.1</name>
  <config>
   <name><platform>-af-ea-7.3.2v1.0.0.1</name>
  </config>
  <state>
   <name><platform>-af-ea-7.3.2v1.0.0.1</name>
  </state>
 </subcomponent>
...
```

The following example shows the enhanced `OPERATING_SYSTEM_UPDATE` package component (RPMs) of the oc-platform data model:

```
<component>
<name>IOSXR-PKG/1 <platform>-isis-2.1.0.0-r732</name>
<config>
<name><platform>-isis-2.1.0.0-r732</name>
</config>
<state>
<name><platform>-isis-2.1.0.0-r732</name>
<type xmlns:idx="http://openconfig.net/yang/platform-types">idx:OPERATING_SYSTEM_UPDATE</type>
<description>IOS XR Operating System Update</description>
<software-version>7.3.2</software-version>----------------------> Label Info
<removable>true</removable>
<oper-status xmlns:idx="http://openconfig.net/yang/platform-types">idx:ACTIVE</oper-status>
</state>
</component>
```

### Associated Commands

- **show install committed**—Shows the committed IOS XR packages.
- **show install committed summary**—Shows a summary of the committed packages along with the committed IOS XR version that is displayed as a label.

# OAM for MPLS and SR-MPLS in mpls-ping and mpls-traceroute Data Models

*Table 7: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| YANG Data Models for MPLS OAM RPCs | Release 7.3.2 | This feature introduces the `Cisco-IOS-XR-mpls-ping-act` and `Cisco-IOS-XR-mpls-traceroute-act` YANG data models to accommodate operations, administration and maintenance (OAM) RPCs for MPLS and SR-MPLS.<br>You can access these Cisco IOS XR native data models from the Github repository. |

The Cisco-IOS-XR-mpls-ping-act and Cisco-IOS-XR-mpls-traceroute-act YANG data models are introduced to provide the following options:

- Ping for MPLS:
  - MPLS IPv4 address
  - MPLS TE
  - FEC-129 Pseudowire
  - FEC-128 Pseudowire
  - Multisegment Pseudowire
- Ping for SR-MPLS:
  - SR policy name or BSID with LSP end-point
  - SR MPLS IPv4 address
  - SR Nil-FEC labels
  - SR Flexible Algorithm
- Traceroute for MPLS:
  - MPLS IPv4 address
  - MPLS TE
- Traceroute for SR-MPLS:
  - SR policy name or BSID with LSP end-point

- SR MPLS IPv4 address
- SR Nil-FEC labels
- SR Flexible Algorithm

The following example shows the ping operation for an SR policy and LSP end-point:

```
<mpls-ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-ping-act">
  <sr-mpls>
  <policy>
    <name>srte_c_10_ep_10.10.10.1</name>
      <lsp-endpoint>10.10.10.4</lsp-endpoint>
  </policy>
  </sr-mpls>
  <request-options-parameters>
    <brief>true</brief>
  </request-options-parameters>
</mpls-ping>
```

**Response:**

```
<?xml version="1.0"?>
 <mpls-ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-ping-act">
  <request-options-parameters>
   <exp>0</exp>
   <fec>false</fec>
   <interval>0</interval>
   <ddmap>false</ddmap>
   <force-explicit-null>false</force-explicit-null>
   <packet-output>
    <interface-name>None</interface-name>
    <next-hop>0.0.0.0</next-hop>
   </packet-output>
   <pad>abcd</pad>
   <repeat>5</repeat>
   <reply>
    <dscp>255</dscp>
    <reply-mode>default</reply-mode>
    <pad-tlv>false</pad-tlv>
   </reply>
   <size>100</size>
   <source>0.0.0.0</source>
   <destination>127.0.0.1</destination>
   <sweep>
    <minimum>100</minimum>
    <maximum>100</maximum>
    <increment>1</increment>
   </sweep>
   <brief>true</brief>
   <timeout>2</timeout>
   <ttl>255</ttl>
  </request-options-parameters>
  <replies>
   <reply>
    <reply-index>1</reply-index>
    <return-code>3</return-code>
    <return-char>!</return-char>
    <reply-addr>14.14.14.3</reply-addr>
    <size>100</size>
   </reply>
   <reply>
    <reply-index>2</reply-index>
```

```
    <return-code>3</return-code>
    <return-char>!</return-char>
    <reply-addr>14.14.14.3</reply-addr>
    <size>100</size>
   </reply>
   <reply>
    <reply-index>3</reply-index>
    <return-code>3</return-code>
    <return-char>!</return-char>
    <reply-addr>14.14.14.3</reply-addr>
    <size>100</size>
   </reply>
   <reply>
    <reply-index>4</reply-index>
    <return-code>3</return-code>
    <return-char>!</return-char>
    <reply-addr>14.14.14.3</reply-addr>
    <size>100</size>
   </reply>
   <reply>
    <reply-index>5</reply-index>
    <return-code>3</return-code>
    <return-char>!</return-char>
    <reply-addr>14.14.14.3</reply-addr>
    <size>100</size>
   </reply>
  </replies>
 </mpls-ping-response>
```

The following example shows the ping operation for an SR policy BSID and LSP end-point:

```
<mpls-ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-ping-act">
<sr-mpls>
<policy>
    <bsid>1000</bsid>
    <lsp-endpoint>10.10.10.4</lsp-endpoint>
</policy>
</sr-mpls>
<request-options-parameters>
    <brief>true</brief>
</request-options-parameters>
</mpls-ping>
```

**Response:**

```
<?xml version="1.0"?>
 <mpls-ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-ping-act">
  <request-options-parameters>
   <exp>0</exp>
   <fec>false</fec>
   <interval>0</interval>
   <ddmap>false</ddmap>
   <force-explicit-null>false</force-explicit-null>
   <packet-output>
    <interface-name>None</interface-name>
    <next-hop>0.0.0.0</next-hop>
   </packet-output>
   <pad>abcd</pad>
   <repeat>5</repeat>
   <reply>
    <dscp>255</dscp>
    <reply-mode>default</reply-mode>
    <pad-tlv>false</pad-tlv>
```

```
  </reply>
  <size>100</size>
  <source>0.0.0.0</source>
  <destination>127.0.0.1</destination>
  <sweep>
   <minimum>100</minimum>
   <maximum>100</maximum>
   <increment>1</increment>
  </sweep>
  <brief>true</brief>
  <timeout>2</timeout>
  <ttl>255</ttl>
 </request-options-parameters>
 <replies>
  <reply>
   <reply-index>1</reply-index>
   <return-code>3</return-code>
   <return-char>!</return-char>
   <reply-addr>14.14.14.3</reply-addr>
   <size>100</size>
  </reply>
  <reply>
   <reply-index>2</reply-index>
   <return-code>3</return-code>
   <return-char>!</return-char>
   <reply-addr>14.14.14.3</reply-addr>
   <size>100</size>
  </reply>
  <reply>
   <reply-index>3</reply-index>
   <return-code>3</return-code>
   <return-char>!</return-char>
   <reply-addr>14.14.14.3</reply-addr>
   <size>100</size>
  </reply>
  <reply>
   <reply-index>4</reply-index>
   <return-code>3</return-code>
   <return-char>!</return-char>
   <reply-addr>14.14.14.3</reply-addr>
   <size>100</size>
  </reply>
  <reply>
   <reply-index>5</reply-index>
   <return-code>3</return-code>
   <return-char>!</return-char>
   <reply-addr>14.14.14.3</reply-addr>
   <size>100</size>
  </reply>
 </replies>
</mpls-ping-response>
```

The following example shows the traceroute operation for an SR policy and LSP end-point:

```
<mpls-traceroute xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-traceroute-act">
<sr-mpls>
<policy>
    <name>srte_c_10_ep_10.10.10.1</name>
    <lsp-endpoint>10.10.10.4</lsp-endpoint>
</policy>
</sr-mpls>
<request-options-parameters>
    <brief>true</brief>
</request-options-parameters>
```

```
</mpls-traceroute>
```

### Response:

```
<?xml version="1.0"?>
 <mpls-traceroute-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-mpls-traceroute-act">

  <request-options-parameters>
   <exp>0</exp>
   <fec>false</fec>
   <ddmap>false</ddmap>
   <force-explicit-null>false</force-explicit-null>
   <packet-output>
    <interface-name>None</interface-name>
    <next-hop>0.0.0.0</next-hop>
   </packet-output>
   <reply>
    <dscp>255</dscp>
    <reply-mode>default</reply-mode>
   </reply>
   <source>0.0.0.0</source>
   <destination>127.0.0.1</destination>
   <brief>true</brief>
   <timeout>2</timeout>
   <ttl>30</ttl>
  </request-options-parameters>
  <paths>
   <path>
    <path-index>0</path-index>
    <hops>
    <hop>
        <hop-index>0</hop-index>
        <hop-origin-ip>11.11.11.1</hop-origin-ip>
        <hop-destination-ip>11.11.11.2</hop-destination-ip>
        <mtu>1500</mtu>
        <dsmap-label-stack>
            <dsmap-label>
                <label>16003</label>
            </dsmap-label>
        </dsmap-label-stack>
        <return-code>0</return-code>
        <return-char> </return-char>
    </hop>
    <hop>
        <hop-index>1</hop-index>
        <hop-origin-ip>11.11.11.2</hop-origin-ip>
        <hop-destination-ip>14.14.14.3</hop-destination-ip>
        <mtu>1500</mtu>
        <dsmap-label-stack>
            <dsmap-label>
                <label>3</label>
            </dsmap-label>
        </dsmap-label-stack>
        <return-code>8</return-code>
        <return-char>L</return-char>
    </hop>
    <hop>
        <hop-index>2</hop-index>
        <hop-origin-ip>14.14.14.3</hop-origin-ip>
        <hop-destination-ip></hop-destination-ip>
        <mtu>0</mtu>
        <dsmap-label-stack/>
        <return-code>3</return-code>
        <return-char>!</return-char>
```

```
  </hop>
  </hops>
 </path>
</paths>
</mpls-traceroute-response>
```

# OpenConfig YANG Model:SR-TE Policies

*Table 8: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| OpenConfig YANG Model:SR-TE Policies | Release 7.3.4 | This release supports the OpenConfig (OC) Segment Routing-Traffic Engineering (SR-TE) YANG data model that provides data definitions for SR-TE policy configuration and associated signaling and traffic engineering protocols. Using the model, you can stream a collection of SR-TE operational statistics, such as color, endpoint, and state.<br>You can access the OC data model from the Github repository. |

The OC SR-TE policies YANG Data Model supports Version 0.22. Subscribe to the following sensor path to send a pull request to the YANG leaf, list, or container:

`openconfig-network-instance:network-instances/network-instance/segment-routing/te-policies`

The response from the router is a collection of SR-TE operational statistics, such as color, endpoint, and state.

### Limitations

- Segment-list ID
  - All locally-configured segment-lists have a unique segment-list ID except for the BGP TE controller. Instead, the BGP TE controller uses the index of the segment-list as the segment-list ID. This ID depends on the local position of the segment-list and can change over time. Therefore for BGP TE controller, you must stream the entire table of the segment-list to ensure that the segment-list ID is always up-to-date.

- Next-hop index
  - The Next-hop container is imported from the `openconfig-aft-common.yang` module where the next-hop index is defined as Uint64. However, the AFT OC in the FIB uses a positional value of the index and does not identify the next-hop entry separately. Similarly, the next-hop container for OC-SRTE ais also implemented as a positional value of the entry in the list. Ensure that you stream the entire table of the next-hop to get a updated index along with the next-hop entry.

# Aggregate Prefix SID Counters for OpenConfig SR YANG Module

*Table 9: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Aggregate Prefix SID Counters for OpenConfig SR YANG Module | Release 7.3.4 | The following components are now available in the OpenConfig (OC) Segment-Routing (SR) YANG model:<br>• The **aggregate-sid-counters** container in the **sr-mpls-top** group to aggregate the prefix segment identifier (SID) counters across the router interfaces.<br>• **The aggregate-sid-counter** and the **mpls-label key** to aggregate counters across all the router interfaces corresponding to traffic forwarded with a particular prefix-SID.<br>You can access the OC data model from the Github repository. |

The OpenConfig SR YANG model supports Version 0.3. Subscribe to the following sensor path:

`openconfig-mpls/mpls/signaling-protocols/segment-routing/aggregate-sid-counters/aggregate-sid-counter/mpls-label/state`

When a receiver subscribes to the sensor path, the router periodically streams the statistics to telemetry for each SR-label. The default collection interval is 30 seconds.

# OpenConfig YANG Model:MACsec

*Table 10: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| OpenConfig YANG Model:MACsec | Release 7.5.2 | You can now use the OpenConfig YANG data model to define the MACsec key chain and policy, and apply MACsec encryption on a router interface.<br>You can access the OC data model from the Github repository. |

With the OpenConfig YANG Model:MACsec, you can also retrieve operational data from the NETCONF agent using gRPC. By automating processes that are repeated across multiple network elements, you can leverage the YANG models for MACsec.

You can use the following operations to stream Telemetry data by sending a request to the NETCONF agent:

- <get>
- <get-config>
- <edit-config>

Subscribe to the following sensor paths to send a pull request to the YANG leaf, list, or container:

- mka/key-chains/key-chain/mka-keys/mka-key
- interfaces/interface/mka
- interfaces/interface
- mka/policies/policy
- interfaces/interface/scsa-rx/scsa-rx
- interfaces/interface/scsa-tx/scsa-tx

### Limitation

- The current implementation of Cisco IOS XR supports only the local time zone configuration in the YYYY-MM-DDTHH:MM:SS format for the following paths:
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/config/valid-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/config/expiration-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/state/valid-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/state/expiration-date-time
- Under the MACsec policy, you can disable the delay-protection and include-icv-indicator leaves only by using the delete operation. You cannot modify the configuration by updating the default field value, from true to false. This codeblock shows a sample delete operation:

```
<config>
<delay-protection nc:operation="delete"/>
<include-icv-indicator nc:operation="delete"/>
</config>
```

### Running Configuration

```
RP/0/0/CPU0:ios#show running-config
Tue Apr 19 21:36:08.882 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Apr 14 16:25:17 2022 by UNKNOWN
key chain kc
 macsec
  key 1234
   key-string password
00554155500E5D5157701E1D5D4C53404A5A5E577E7E727F6B647040534355560E080A00005B554F4E080A0407070303530A54540C0252445E550958525A771B16
```

```
 cryptographic-algorithm aes-256-cmac
   lifetime 00:01:01 january 01 2021 infinite
  netconf-yang agent
 ssh
interface GigabitEthernet0/0/0/0
 shutdown
interface GigabitEthernet0/0/0/1
 macsec psk-keychain kc
interface GigabitEthernet0/0/0/2
 macsec psk-keychain kc policy mp
interface GigabitEthernet0/0/0/3
 shutdown
interface GigabitEthernet0/0/0/4
 shutdown
macsec-policy mp
 cipher-suite GCM-AES-XPN-256
 key-server-priority 4
ssh server v2
end
```

### RPC Request for get-config

```
<get-config>
   <source>
      <running/>
   </source>
        <filter>
          <macsec xmlns="http://openconfig.net/yang/macsec">
          </macsec>
        </filter>
</get-config>
```

### RPC Response for get-config

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <macsec xmlns="http://openconfig.net/yang/macsec">
   <mka>
    <policies>
     <policy>
      <name>mp</name>
      <config>
       <name>mp</name>
       <macsec-cipher-suite>gcm-aes-xpn-256</macsec-cipher-suite>
       <key-server-priority>4</key-server-priority>
      </config>
     </policy>
    </policies>
    <key-chains>
     <key-chain>
      <name>kc</name>
      <config>
       <name>kc</name>
      </config>
      <mka-keys>
       <mka-key>
        <id>1234</id>
        <config>
         <id>1234</id>
         <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
         <valid-date-time>2021-01-01T00:01:01</valid-date-time>
         <expiration-date-time>NO_EXPIRATION</expiration-date-time>
```

```
        </config>
       </mka-key>
      </mka-keys>
     </key-chain>
    </key-chains>
   </mka>
   <interfaces>
    <interface>
     <name>GigabitEthernet0/0/0/1</name>
     <config>
      <name>GigabitEthernet0/0/0/1</name>
     </config>
     <mka>
      <config>
       <key-chain>kc</key-chain>
      </config>
     </mka>
    </interface>
    <interface>
     <name>GigabitEthernet0/0/0/2</name>
     <config>
      <name>GigabitEthernet0/0/0/2</name>
     </config>
     <mka>
      <config>
       <key-chain>kc</key-chain>
       <mka-policy>mp</mka-policy>
      </config>
     </mka>
    </interface>
   </interfaces>
  </macsec>
 </data>
</rpc-reply>
```

### RPC Request for get

```
<get>
   <filter>
     <macsec xmlns="http://openconfig.net/yang/macsec">
     </macsec>
     </filter>
</get>
```

### RPC Response for get

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <macsec xmlns="http://openconfig.net/yang/macsec">
   <mka>
    <policies>
     <policy>
      <name>mp</name>
      <config>
       <name>mp</name>
       <macsec-cipher-suite>gcm-aes-xpn-256</macsec-cipher-suite>
       <key-server-priority>4</key-server-priority>
      </config>
      <state>
       <name>mp</name>
       <key-server-priority>4</key-server-priority>
       <macsec-cipher-suite>gcm-aes-xpn256</macsec-cipher-suite>
```

```
    <confidentiality-offset>zero-bytes</confidentiality-offset>
    <delay-protection>false</delay-protection>
    <include-icv-indicator>false</include-icv-indicator>
    <sak-rekey-interval>0</sak-rekey-interval>
   </state>
  </policy>
  <policy>
   <name>DEFAULT-POLICY</name>
   <state>
    <name>DEFAULT-POLICY</name>
    <key-server-priority>16</key-server-priority>
    <macsec-cipher-suite>gcm-aes-xpn256</macsec-cipher-suite>
    <confidentiality-offset>zero-bytes</confidentiality-offset>
    <delay-protection>false</delay-protection>
    <include-icv-indicator>false</include-icv-indicator>
    <sak-rekey-interval>0</sak-rekey-interval>
   </state>
  </policy>
 </policies>
 <key-chains>
  <key-chain>
   <name>kc</name>
   <config>
    <name>kc</name>
   </config>
   <mka-keys>
    <mka-key>
     <id>1234</id>
     <config>
      <id>1234</id>
      <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
      <valid-date-time>2021-01-01T00:01:01</valid-date-time>
      <expiration-date-time>NO_EXPIRATION</expiration-date-time>
     </config>
     <state>
      <id>1234</id>
      <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
      <valid-date-time>2021-01-01T00:01:01</valid-date-time>
      <expiration-date-time>NO_EXPIRATION</expiration-date-time>
     </state>
    </mka-key>
   </mka-keys>
   <state>
    <name>kc</name>
   </state>
  </key-chain>
 </key-chains>
</mka>
<interfaces>
 <interface>
  <name>GigabitEthernet0_0_0_1</name>
  <state>
   <name>GigabitEthernet0_0_0_1</name>
   <counters>
    <tx-untagged-pkts>8</tx-untagged-pkts>
    <rx-untagged-pkts>0</rx-untagged-pkts>
    <rx-badtag-pkts>2</rx-badtag-pkts>
    <rx-unknownsci-pkts>3</rx-unknownsci-pkts>
    <rx-nosci-pkts>4</rx-nosci-pkts>
   </counters>
  </state>
  <mka>
   <state>
    <mka-policy>DEFAULT-POLICY</mka-policy>
```

```
    <key-chain>kc</key-chain>
    <counters>
     <in-mkpdu>0</in-mkpdu>
     <in-sak-mkpdu>0</in-sak-mkpdu>
     <out-mkpdu>225271</out-mkpdu>
     <out-sak-mkpdu>0</out-sak-mkpdu>
    </counters>
   </state>
  </mka>
  <scsa-tx>
   <scsa-tx>
    <sci-tx>024f88a08c9d0001</sci-tx>
    <state>
     <sci-tx>024f88a08c9d0001</sci-tx>
     <counters>
      <sc-encrypted>0</sc-encrypted>
      <sa-encrypted>0</sa-encrypted>
     </counters>
    </state>
   </scsa-tx>
  </scsa-tx>
 </interface>
 <interface>
  <name>GigabitEthernet0_0_0_2</name>
  <state>
   <name>GigabitEthernet0_0_0_2</name>
   <counters>
    <tx-untagged-pkts>8</tx-untagged-pkts>
    <rx-untagged-pkts>0</rx-untagged-pkts>
    <rx-badtag-pkts>2</rx-badtag-pkts>
    <rx-unknownsci-pkts>3</rx-unknownsci-pkts>
    <rx-nosci-pkts>4</rx-nosci-pkts>
   </counters>
  </state>
  <mka>
   <state>
    <mka-policy>mp</mka-policy>
    <key-chain>kc</key-chain>
    <counters>
     <in-mkpdu>0</in-mkpdu>
     <in-sak-mkpdu>0</in-sak-mkpdu>
     <out-mkpdu>225271</out-mkpdu>
     <out-sak-mkpdu>0</out-sak-mkpdu>
    </counters>
   </state>
  </mka>
  <scsa-tx>
   <scsa-tx>
    <sci-tx>0246c822daae0001</sci-tx>
    <state>
     <sci-tx>0246c822daae0001</sci-tx>
     <counters>
      <sc-encrypted>0</sc-encrypted>
      <sa-encrypted>0</sa-encrypted>
     </counters>
    </state>
   </scsa-tx>
  </scsa-tx>
 </interface>
 <interface>
  <name>GigabitEthernet0/0/0/1</name>
  <config>
   <name>GigabitEthernet0/0/0/1</name>
  </config>
```

```
     <mka>
      <config>
       <key-chain>kc</key-chain>
      </config>
     </mka>
    </interface>
    <interface>
     <name>GigabitEthernet0/0/0/2</name>
     <config>
      <name>GigabitEthernet0/0/0/2</name>
     </config>
     <mka>
      <config>
       <key-chain>kc</key-chain>
       <mka-policy>mp</mka-policy>
      </config>
     </mka>
    </interface>
   </interfaces>
  </macsec>
 </data>
</rpc-reply>
```

# OpenConfig YANG Model:AFT

***Table 11: Feature History Table***

| Feature Name | Release Information | Description |
|---|---|---|
| OpenConfig YANG Model:AFT | Release 7.3.4 | This release supports the OpenConfig Abstract Forwarding Table (AFT) containers, such as IPv4, IPv6, Network Instance, and MPLS. With this support, the AFT sends only essential interface forwarding entries, such as the next-hop, next-hop group, and RSVP-TE for an IP prefix, to the Network Management System (NMS). Since the NMS receives only essential entries, the forwarding process is simplified.<br>You can access the OC data model from the Github repository. |

### Supported Agents

The following agents are supported in the SAMPLE and ON-CHANGE modes:

- gNMI
- IOS-XR proprietary telemetry dial-in and dial-out

### Limitations

- The Netconf agent is not supported on configuration and operation data.
- The ON-CHANGE mode is supported only at the path level as shown below:
  - /network-instances/network-instance/afts/ipv4-unicast/ipv4-entry
  - /network-instances/network-instance/afts/ipv6-unicast/ipv6-entry
  - /network-instances/network-instance/afts/mpls/label-entry
  - /network-instances/network-instance/afts/next-hop-groups/next-hop-group/state
  - /network-instances/network-instance/afts/next-hop-groups/next-hop-group/next-hops/next-hop
  - /network-instances/network-instance/afts/next-hops/next-hop
- The current implementation of the OC-AFT model, version 0.6.0 does not set the atomic flag for atomic updates for gNMI.

### Response

A SubscribeRequest message is sent by a gNMI client to request updates from the router for a specified set of paths. The following SubscriptionResponse messages are sent by the router:

AFT IPv4 unicast

```
SubscribeResponse.update: <
timestamp: 1647978999525525791
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts ipv4-unicast ipv4-entry[prefix=10.0.0.1/32] > < json_ietf_val:"{
"state": {
"prefix": "10.0.0.1/32",
"next-hop-group": "1152921642045939938"
}
}" > >

SubscribeResponse.update: <
timestamp: 1647978999341662576
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts ipv4-unicast ipv4-entry[prefix=10.1.1.1/32] > < json_ietf_val:"{
"state": {
"prefix": "10.1.1.1/32",
"next-hop-group": "1152921779484853982"
}
}" > >
```

AFT IPv6 unicast

```
SubscribeResponse.update: <
timestamp: 1647984444644492536
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
 afts ipv6-unicast ipv6-entry[prefix=50:50:58::331/128] > < json_ietf_val:"{
```

```
"state": {
"prefix": "50:50:58::331/128",
"next-hop-group": "1153062379534237025"
}
}" > >
```

List of MPLS entries within the AFT

```
SubscribeResponse.update: <
timestamp: 1648009876493069763
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts mpls label-entry[label=12000] > < json_ietf_val:"{
"state": {
"label": 12000,
"next-hop-group": "1152921642046007012"
}
}" > >
```

```
SubscribeResponse.update: <
timestamp: 1648011005293000000
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts mpls label-entry[label=12000] > < json_ietf_val:"{
"state": {
"label": 12000,
"packets-forwarded": "0",
"octets-forwarded": "0"
}
}" > >
```

AFT next-hop-group

```
SubscribeResponse.update: <
timestamp: 1648011006899606800
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts next-hop-groups next-hop-group[id=1152921642045939938] >
 < json_ietf_val:"{
"next-hops": {
"next-hop": {
"index": "1152921642045903362",
"state": {
"index": "1152921642045903362",
"weight": "0"
}
}
}
}" > >


>
SubscribeResponse.update: <
timestamp: 1648011006899606800
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts next-hop-groups next-hop-group[id=1152921642045939938] >
< json_ietf_val:"{
```

```
"next-hops": {
"next-hop": {
"index": "1152921642045903355",
"state": {
"index": "1152921642045903355",
"weight": "0"
}
}
}
}" > >

SubscribeResponse.update: <
timestamp: 1648011006899606800
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
 afts next-hop-groups next-hop-group[id=1152921642045939938] >
 < json_ietf_val:"{
"next-hops": {
"next-hop": {
"index": "1152921642045903348",
"state": {
"index": "1152921642045903348",
"weight": "0"
}
}
}
}" > >
```

AFT next-hops next-hop

```
SubscribeResponse.update: <
timestamp: 1648011006713962739
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts next-hops next-hop[index=1152921642045903362] > < json_ietf_val:"{
"state": {
"index": "1152921642045903362",
"ip-address": "13.1.1.1"
},
"interface-ref": {
"state": {
"interface": "tunnel-ip2",
"subinterface": 0
}
}
}" > >

SubscribeResponse.update: <
timestamp: 1648011006713954259
prefix: <
origin: openconfig-network-instance
>
update: < path: < element: network-instances network-instance[name=default]
afts next-hops next-hop[index=1152921642045903355] > < json_ietf_val:"{
"state": {
"index": "1152921642045903355",
"ip-address": "13.1.1.2"
},
"interface-ref": {
"state": {
"interface": "tunnel-ip3",
"subinterface": 0
```

```
}
}
}" > >
```

# Automatic Resynchronization of OpenConfig Configuration

*Table 12: Feature History Table*

| Feature Name | Release Information | Feature Description |
|---|---|---|
| View Inconsistent OpenConfig Configuration | Release 24.1.1 | Introduced in this release on: NCS 5500 fixed port routers; NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards; NCS 5700 line cards [Mode: Compatibility; Native]<br><br>OpenConfig infrastructure now provides an operational data YANG model, `Cisco-IOS-XR-yiny-oper`, which can be queried to view the inconsistent OpenConfig configuration caused due to activities such as interface breakout operations, installation activities or insertion of a new line card.<br><br>See GitHub, YANG Data Models Navigator |

| Feature Name | Release Information | Feature Description |
|---|---|---|
| Automatic Resynchronization of OpenConfig Configuration | Release 7.11.1 | Introduced in this release on: NCS 5500 fixed port routers; NCS 5700 fixed port routers; NCS 5500 modular routers (NCS 5500 line cards; NCS 5700 line cards [Mode: Compatibility; Native])<br><br>OpenConfig infrastructure can now reapply all the OpenConfig configurations automatically if there are any discrepancies in the running configuration.<br><br>With this feature, there is no need for manual replacement of the OpenConfig configuration using Netconf or gNMI.<br><br>The re-sync operation is triggered if the running configurations and the OpenConfig configuration go out of sync after any system event that removes some running configurations from the system. A corresponding system log gets generated to indicate the re-sync status. |

In the earlier releases, when activities such as interface breakout operations, installation activities or insertion of a new line card took place, there was a risk of OpenConfig configuration and the running configuration going out of sync. A full replacement of the OpenConfig configuration was required in order to get the OpenConfig configurations back in sync using Netconf or gNMI.

From the Cisco IOS XR Software Release 7.11.1, if the OpenConfig configurations and running configurations go out of sync, or any activities takes place which may result in the two configurations to go out of sync, the system automatically reapplies all the OpenConfig configurations and resolve the sync issue. If there is a synchronization issue between the running configuration and the OpenConfig configuration, a corresponding system log is generated to indicate it. Similarly, a corresponding system log is generated indicating the status of the re-synchronization attempt.

This feature is enabled by default. This process is completely automated.

From the Cisco IOS XR Software Release 24.1.1, the new `Cisco-IOS-XR-yiny-oper` YANG model displays the OpenConfig configuration which is out of sync with the running configuration, including the error associated with each out of sync configuration.

The `Cisco-IOS-XR-yiny-oper` operational data is a snapshot of the current system status, rather than a record of all past failures. That is, if an item of configuration is out of sync and is later resolved, such as through a resynchronization or another configuration operation, then this configuration is no longer considered out of sync and is removed from the snapshot.

### Operations that Remove Running Configuration

Here are three types of operation that can have the effect of removing running configuration from the system. Running configurations are either affected because they directly remove configuration in the system or because they result in configuration failing to be accepted by the system during start-up.

- **Install operations**: Running configuration can be removed during non-reload and reload install operations. During non-reload install, running configuration is removed when it is incompatible with the new software. In this case, it is directly removed by the Install infra. The configuration is removed during reload install operations if the attempt to restore the startup configuration is partially successful.
- **Breakout interfaces configuration**: When breakout interfaces are configured or de-configured, all the existing configuration on interfaces is affected. The affect may be creation or deletion of the parent and child interfaces. This results in an inconsistency between the running configuration and the OpenConfig datastore for any of the removed configurations that was mapped from OpenConfig configuration.

  The automatic restoration of OpenConfig configuration resolves this inconsistency by re-adding that removed configuration.
- **New line card insertion**: On insertion of a new line card into the system, any pre-configuration for that card is verified for the first time and may be rejected, causing it to be removed. This results in an inconsistency between the running configuration and the OpenConfig datastore.

In any of the above scenarios, if there is a sync issue, system logs are generated and the system tries to reapply all the OpenConfig configurations. If the re-sync attempt is successful, the configurations which were removed earlier, are re-applied. If the re-sync attempt fails, this means that some of the OpenConfig configuration is no longer valid.



**Note** The above scenarios are invalid if there are no OpenConfig configuration present in the system.

### System Logs Indicating Out-of-Sync Configuration

System log messages are generated due to the above operations that can lead to discrepancies in configurations on the router. Listed are examples of system log messages raised if any such discrepancies occur.

***Table 13: Examples of system log messages generated due to Out-of-Sync Configurations :***

| Event Name Displayed in the System Log | Description |
|---|---|
| **unexpected commit errors** | When an unexpected commit errors in case of a SysDB server crash. |
| **config rollback (to a commit ID created using a different software version)** | When a configuration rollbacks back to a commit ID created using a different software version. |
| **inconsistent configuration** | This system log is generated when an inconsistency alarm is raised due to failure in restoring the start-up configurations after activities like system reload or insertion of a new line card. Re-synchronization of the configuration is triggered only after the alarm is cleared. |

| Event Name Displayed in the System Log | Description |
|---|---|
| **configuration removal (triggered on 0/2/CPU0 by the last config operation for interface GigabitEthernet0/2/0/0 and 6 other interfaces)** | When interface configuration is removed in response to a change in interface breakout configuration. |
| **configuration removal (to prepare for an install operation)** | Configuration is removed from the system during a non-reload install operation due to incompatibility with the new software. |

### Alarms Related to Out-of-Sync OpenConfig Configuration

- **Inconsistency alarm**: When a there is a failure in restoring the start-up configurations after a system reload or insertion of a new line card, inconsistency alarm is raised. If the inconsistency alarm is raised, you can see an informational system log is generated which indicates that the OpenConfig configuration and running configuration may be out of sync. A re-sync attempt will be made when the configuration inconsistency alarm is cleared. This system log is an early warning that the system is potentially out of sync.

  Inconsistency alarm message:

  ```
  NMI OpenConfig configuration is potentially out of sync with the running configuration
  (details: system configuration become inconsistent during OIR restore on 0/0/CPU0). An
  automatic reapply of the OpenConfig configuration will be performed when the inconsistency
  alarm is cleared.
  ```

- **Missing item in the OpenConfig datastore alarm**: If there are missing items in the configurations which could not be added to the OpenConfig datastore while loading in a snapshot from disk, you can see an error system log is raised which indicates that there are some items which are absent in the running OpenConfig configuration. This scenario occurs when the yang schema is changed from the time the snapshot was created.

  Item missing alarm message:

  ```
  gNMI OpenConfig configuration is potentially out of sync with the running configuration:
  3 failed to be applied to the system (details: snapshot 2 was created with a different
  schema version). The system may contain config items mapped from OC that no longer exist
  in the OC datastore. Automatic attempts to reapply OC will not remove these items, even
  if they otherwise succeed. Config should be replaced manually using a GNMI Replace
  operation.
  ```

### System Logs Generated During Configuration Resynchronization:

When an attempt to re-apply OpenConfig (resynchronization) is complete, the following informational system logs are generated to indicate the user that the OpenConfig and running configuration were out of sync, and whether the attempt to resolve this was successful.

- Successful re-sync:

  ```
  As a result of configuration removal (to prepare for an install operation), the gNMI
  OpenConfig configuration has been successfully reapplied.
  ```

- Unsuccessful re-sync:

```
As a result of configuration removal (to prepare for an install operation), an attempt
to reapply the gNMI OpenConfig configuration was made, but some items remain out of
sync with the running configuration. Out of sync configuration can be viewed using the
Cisco-IOS-XR-yiny-oper model.
```

- Re-sync failure during mapping of OpenConfig configurations to XR configurations:

```
As a result of configuration removal (to prepare for an install operation), the attempt
to reapply the gNMI OpenConfig configuration failed, and the out of sync configuration
could not be updated. gNMI OpenConfig configuration is potentially out of sync with the
running configuration. Configuration should be reapplied manually using a GNMI Replace
operation
```

Re-sync failure during mapping of OpenConfig configurations to XR configurations is a rare scenario. When there is a failure in the re-sync process while mapping the OpenConfig configuration to XR items, it causes the re-sync request to aborted. This scenario is only possible after an install which changes the OpenConfig mappings such that some configuration is no longer supported.

### Resolve Out of Sync Configuration

An automatic resynchronization fails if the out-of-sync scenario is unresolved or the OpenConfig configuration and running XR configuration are out of sync.

Here are the two scenarios with steps to resolve the out-of-sync configuration if an attempt for automatic resynchronization fails.

**Resync Fails Partially**:

1. Query the items of configuration which are out of sync using the `Cisco-IOS-XR-yiny-oper` YANG model
2. For each out-of-sync configuration item:
    - Delete the OpenConfig items that are out of sync.
    - Re-add the deleted OpenConfig items in a separate request.

**Resync Fails Completely**:

Perform a full replace of the OpenConfig configuration using Netconf or gNMI.

By successfully completing these steps, you can now ensure that all configurations are in sync.

### YANG Model Data for Inconsistent Configuration

Each configuration of the `Cisco-IOS-XR-yiny-oper` YANG model has a list entry with the following fields:

- **Path**: The path of the XR configuration, in YPath format.
- **Input paths**: The OpenConfig paths of the items from which the XR configuration is mapped.

  **Activity**: If last occurrence of this failure was:
    - in a user-initiated commit operation.
    - in a system-initiated resynchronization attempt, after an install operation, breakout interfaces being configured, or line card insertion.

- **Operation**: If a configuration being `set` or `delete`:

For a configuration that is out of sync because it failed during a resynchronization attempt, the operation is always `set`, but for a user-initiated commit operation, the operation is whichever the user was attempting during the commit.

- **Latest failure type**: If the latest failure is a `verify` failure or an `apply` failure.

  Only `verify` errors are currently tracked as out of sync and reported in the operational data, but this field is present in the model for potential future usage if `apply` errors are also tracked.

  - For configuration that fails during startup, both `verify` and `apply` failures can make the configurations out of sync.
  - For configuration that fails during a commit operation, only `apply` failures can make the configuration out of sync. This is because configuration is not allowed in the datastore if `verify` failures occur during a commit operation.

- **Latest error**: The latest error message describing the error.