



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](#)
- [OpenConfig Data Model Enhancements, on page 4](#)
- [Define Power State of Line Card Using Data Model, on page 5](#)
- [Install Label in oc-platform Data Model, on page 6](#)
- [OpenConfig YANG Model:SR-TE Policies, on page 8](#)
- [Aggregate Prefix SID Counters for OpenConfig SR YANG Module, on page 9](#)
- [OpenConfig YANG Model:MACsec, on page 10](#)
- [OpenConfig YANG Model:dscp-set, on page 16](#)
- [OpenConfig YANG Model:procmon, on page 19](#)
- [Automatic Resynchronization of OpenConfig Configuration, on page 20](#)
- [LLDP global and interface-level configuration enhancement, on page 24](#)
- [Configure LLDP global and interface-level configuration enhancement, on page 26](#)

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	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. 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.

OpenConfig Data Model Enhancements

Table 2: Feature History Table

Feature Name	Release Information	Description
LACP OpenConfig Model	Release 7.5.3	Use the <code>openconfig-lacp.yang</code> 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. 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 <pre>lacp/interfaces/interface[name]/members/member[interface]/state/:</pre> • <code>last-change</code> • <code>counters/lacp-timeout-transitions</code> 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: • <code>openconfig-mpls</code> • <code>openconfig-mpls-te</code> • <code>openconfig-mpls-rsvp</code> • <code>openconfig-mpls-igp</code> • <code>openconfig-mpls-types</code> • <code>openconfig-mpls-sr</code> You can access this data model from the Github repository.

Define Power State of Line Card Using Data Model

Table 3: Feature History Table

Feature Name	Release Information	Description
Control Line Card Power Using YANG Data Model	Release 7.5.1	The <code>oc-platform.yang</code> YANG data model enables or disables power to the line card and identifies its slot or chassis. You can access this data model from the Github repository.

This feature adds the following component paths to the model to configure and fetch the power state of the line card, enable/disable the power state, and slot ID of line cards:

- `/components/component/linecard/config/power-admin-state`
- `/components/component/linecard/state/power-admin-state`
- `/components/component/linecard/state/slot-id`

```

module: openconfig-platform-linecard
augment /oc-platform:components/oc-platform:component:
  +--rw linecard
    +--rw config
      | +--rw power-admin-state?   oc-platform-types:component-power-type
    +--ro state
      +--ro power-admin-state?   oc-platform-types:component-power-type
      +--ro slot-id?             string

```

The following example shows the configuration to enable the line card in location "0/0" to power up:

```

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <components xmlns="http://openconfig.net/yang/platform">
        <component>
          <name>0/0</name>
          <linecard xmlns="http://openconfig.net/yang/platform/linecard">
            <config>
              <power-admin-state>POWER_ENABLED</power-admin-state>
            </config>
          </linecard>
        </component>
      </components>
    </config>
  </edit-config>
</rpc>

```

To disable the line card, use `POWER_DISABLED` in the state field.

In the following example, an RPC request is sent to retrieve the power state of all line cards:

```

<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <get>

```

```

<filter>
<components xmlns="http://openconfig.net/yang/platform">
  <component>
    <linecard xmlns="http://openconfig.net/yang/platform/linecard">
      <state/>
    </linecard>
  </component>
</components>
</filter>
</get>
</rpc>

```

The following example shows the RPC response to the request:

```

<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <components xmlns="http://openconfig.net/yang/platform">
      <component>
        <name>0/0</name>
        <linecard xmlns="http://openconfig.net/yang/platform/linecard">
          <state>
            <power-admin-state>POWER_ENABLED</power-admin-state>
            <slot-id>0/0</slot-id>
          </state>
        </linecard>
      </component>
    </components>
  </data>
</rpc-reply>

```

Install Label in oc-platform Data Model

Table 4: 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: • software version • golden ISO (GISO) label • committed IOS XR packages 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.

OpenConfig YANG Model:SR-TE Policies

Table 5: 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. 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 is 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 6: 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: • The aggregate-sid-counters container in the sr-mpls-top group to aggregate the prefix segment identifier (SID) counters across the router interfaces. • 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. 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 7: 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. 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
- mka/state/counter

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-XPB-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>

```

```

    </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>
          </key-chain>
        </key-chains>
      </mka>
    </macsec>
  </data>
</rpc-reply>

```

```

    <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>
  </interface>
  <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>
  </interface>
  <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:dscp-set

Table 8: Feature History Table

Feature Name	Release Information	Description
OpenConfig YANG Model:dscp-set	Release 7.5.2	This model allows you to configure a minimum and maximum Differentiated Services Code Point (DSCP) value in the dscp-set leaf-list. When you send these values in your request to the NETCONF agent, it filters the traffic by matching the values in the list with the incoming packet header. This ensures that your network is not vulnerable to unwanted traffic. You can access the OC data model from the Github repository.

You can configure two Differentiated Services Code Point (DSCP) values in the dscp-set leaf-list. You can enter these values in any order, and they are internally mapped to dscp-min and dscp-max values. The incoming IPv4 or IPv6 packet header contains the DSCP field. This DSCP field is matched with the range of values that exist between the specified minimum (dscp-min) and maximum (dscp-max) values. When the DSCP field contains one of the values specified in the list, the incoming packet is allowed access to your network. You can add or delete the dscp-set leaf-list in the IPv4 and IPv6 OpenConfig YANG model by sending a NETCONF request.



Note When you delete one of the values from the dscp-set, the model applies the remaining value for both dscp-min and dscp-max fields.

Adding the dscp-set in the IPv4 OC YANG Model

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
    <candidate/>
  </target>
  <config type="subtree" xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <acl xmlns="http://openconfig.net/yang/acl">
      <acl-sets>
        <acl-set>
          <name>test-dscp-set</name>
          <type>ACL_IPV4</type>
          <config>
            <name>test-dscp-set</name>
            <type>ACL_IPV4</type>
          </config>
        </acl-set>
      </acl-sets>
    </acl>
  </config>
</edit-config>
</rpc>
```

```

    <acl-entry>
      <sequence-id>10</sequence-id>
      <config>
        <sequence-id>10</sequence-id>
      </config>
      <actions>
        <config>
          <forwarding-action>ACCEPT</forwarding-action>
        </config>
      </actions>
      <ipv4>
        <config>
          <dscp-set>12</dscp-set>
          <dscp-set>15</dscp-set>
        </config>
      </ipv4>
    </acl-entry>
  </acl-entries>
</acl-set>
</acl-sets>
</acl>
</config>
</edit-config>
</rpc>

```

Deleting the dscp-set in the IPv4 OC YANG Model

```

<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config type="subtree" xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <acl xmlns="http://openconfig.net/yang/acl">
        <acl-sets>
          <acl-set xc:operation="delete">
            <name> test-dscp-set</name>
            <type>ACL_IPV4</type>
          </acl-set>
        </acl-sets>
      </acl>
    </config>
  </edit-config>
</rpc>

```

Adding the dscp-set in the IPv6 OC YANG Model

```

<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config type="subtree" xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <acl xmlns="http://openconfig.net/yang/acl">
        <acl-sets>
          <acl-set>
            <name>test-dscp-v6-edit</name>
            <type>ACL_IPV6</type>
            <config>
              <name>test-dscp-v6-edit</name>
              <type>ACL_IPV6</type>
            </config>
          </acl-set>
        </acl-sets>
      </acl>
    </config>
  </edit-config>
</rpc>

```

```

    <acl-entry>
      <sequence-id>10</sequence-id>
      <config>
        <sequence-id>10</sequence-id>
      </config>
      <actions>
        <config>
          <forwarding-action>ACCEPT</forwarding-action>
        </config>
      </actions>
    </acl-entry>
  </acl-entries>
</acl-set>
</acl-sets>
</acl>
</config>
</edit-config>
</rpc>

```

Deleting the dscp-set in the IPv6 OC YANG Model

```

<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config type="subtree" xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
      <acl xmlns="http://openconfig.net/yang/acl">
        <acl-sets>
          <acl-set xc:operation="delete">
            <name>test-dscp-v6-edit</name>
            <type>ACL_IPV6</type>
          </acl-set>
        </acl-sets>
      </acl>
    </config>
  </edit-config>
</rpc>

```

OpenConfig YANG Model:procmon

Table 9: Feature History Table

Feature Name	Release Information	Description
OpenConfig YANG Model:procmon	Release 7.5.2	This model provides data definitions to monitor the health of one or more processes running on a system, delivering insights into the performance of critical processes and helping remediate performance bottlenecks. For example, the stress tool that is part of the Linux distribution may be consuming high CPU. The openconfig-procmon model pulls this information and sends it to you when you query the node. As a remediation measure, you can then restart the process. You can access the OC data model from the Github repository.

Subscribe to the following sensor path:

openconfig-system:system/processes/process

Based on a Process ID (PID), you can stream state parameters, such as name, args, start-time, uptime, cpu-usage-user, cpu-usage-system, cpu-utilization, memory usage and memory utilization.

When you send the PID to a MDT-capable device requesting state parameters of a process, the PID of the process acts as a key for the request. If the requested PID is invalid, you will not receive any response.



Note The location of the PID is always assumed to be the Active RP. This model does not have any leaf or field where you can specify the location or node name.

Example

This output shows state parameters that monitor the health of the dhcpd process having PID: 22482 using the XR built-in mdt_exec tool. You can also use telemetry tools, such as gNMI and gRPC.

```
RP/0/RP1/CPU0:SF-D#run mdt_exec -s openconfig-system:system/processes/process[pid=22482]
Enter any key to exit...
  Sub_id 200000001, flag 0, len 0
  Sub_id 200000001, flag 4, len 583
-----
{"node_id_str":"SF-D","subscription_id_str":"app_TEST_200000001",
"encoding_path":"openconfig-system:system/processes/process","collection_id":"13",
"collection_start_time":"1648387172382","msg_timestamp":"1648387172384",
```

```
"data_json": [{"timestamp": "1648387172384", "keys": [{"pid": "22482"}]},
"content": {"state": {"pid": "22482", "name": "dhcpd", "args": ["dhcpd"]},
"start-time": "1648385883000000000", "uptime": "1289384179023", "cpu-usage-user": "270000000",
"cpu-usage-system": "180000000", "cpu-utilization": 0, "memory-usage": "16641952",
"memory-utilization": 0}}], "collection_end_time": "1648387172384"}
-----
Sub_id 2000000001, flag 8, len 0
```

Automatic Resynchronization of OpenConfig Configuration

Table 10: Feature History Table

Feature Name	Release Information	Feature Description
View Inconsistent OpenConfig Configuration	Release 24.1.1	OpenConfig infrastructure now provides an operational data YANG model, <code>Cisco-IOS-XR-yiny-oper</code> , 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. See GitHub , YANG Data Models Navigator
Automatic Resynchronization of OpenConfig Configuration	Release 7.11.1	OpenConfig infrastructure can now reapply all the OpenConfig configurations automatically if there are any discrepancies in the running configuration. With this feature, there is no need for manual replacement of the OpenConfig configuration using Netconf or gNMI. 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 11: 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.
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 be 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.

LLDP global and interface-level configuration enhancement

LLDP (Link Layer Discovery Protocol) global and interface-level configuration enhancement

- prevents LLDP global parameters configuration from automatically enabling LLDP on all interfaces, and
- ensures that LLDP enablement is controlled explicitly at the interface level, maintaining granular control over LLDP behavior.

Table 12: Feature History Table

Feature Name	Release Information	Feature Description
LLDP global and interface-level configuration enhancement	Release 25.1.1	Introduced in this release on: Fixed Systems (8200 [ASIC: Q200, P100], 8700 [ASIC: P100, K100]); Centralized Systems (8600 [ASIC:Q200]); Modular Systems (8800 [LC ASIC: Q100, Q200, P100]) You can now configure LLDP (Link Layer Discovery Protocol) global parameters without automatically enabling LLDP on all interfaces by default. This feature provides a granular operational control of the LLDP interface-level configurations. By aligning with OpenConfig models, this feature introduces a more efficient operational control where LLDP is enabled on an interface only when both global and interface-level LLDP configurations are enabled. Previously, LLDP was enabled if either configuration was present, potentially leading to inconsistent behavior. The feature introduces these changes: CLI: • lldp interface-only YANG Data Model: • <code>Cisco-IOS-XR-um-lldp-cfg</code> (see GitHub, YANG Data Models Navigator)

How does this feature affect the OpenConfig and native YANG model behavior?

When you configure OpenConfig global LLDP parameters, the `lldp interface-only` configuration is automatically added to the native `Cisco-IOS-XR-um-lldp-cfg` YANG model to retain consistent OpenConfig behavior. This ensures that modifying global LLDP parameters through OpenConfig does not unintentionally enable LLDP on all interfaces. This behavior provides seamless alignment between OpenConfig and the native YANG model, ensuring predictable and consistent LLDP behavior across interfaces.

What happens when LLDP is configured globally without using this feature?

Without the **lldp interface-only** option, enabling LLDP globally through OpenConfig adds the LLDP container with global parameters in the native `Cisco-IOS-XR-um-lldp-cfg` YANG model, which in turn activates LLDP on all interfaces. This can lead to inconsistent LLDP behavior, particularly when users expect interface-level configurations to remain in control.

How does the new behavior differ from the old behavior?

Use the table to understand the difference in LLDP operational states between the old and new behaviors.

Table 13: OpenConfig LLDP behavior changes

If OpenConfig global LLDP is...	And OpenConfig interface-level LLDP is...	Then the new LLDP interface state is...	And the old LLDP interface state was...
enabled (default)	container not added state	disabled	enabled.
enabled	enabled (default—after adding the interface container)	enabled	enabled.
enabled	disabled	disabled	enabled.
disabled	enabled	disabled	enabled.
disabled	disabled	disabled	disabled.

What is the effect of global and interface-level LLDP configurations on interface states?

The table provides an overview of how different configurations affect the operational state of the LLDP interfaces.

Table 14: LLDP state based on global and interface-level configuration

If global LLDP is...	And interface-level LLDP configuration is...	Then the interface LLDP state is...
lldp	not configured	enable.
lldp interface-only	not configured	disable.
lldp interface-only	lldp enable	enable.
lldp	lldp enable	enable.

Configure LLDP global and interface-level configuration enhancement

Use this procedure to configure the LLDP interface-only feature on your router, allowing global LLDP parameters to be modified without automatically enabling LLDP on all interfaces.

Procedure

Step 1 Configure the LLDP interface-only parameter.

Example:

```
Router(config)# lldp  
Router(config-lldp)# timer 60  
Router(config-lldp)# interface-only  
Router(config-lldp)# exit
```

Step 2 Verify the LLDP interface-only parameter and global parameters are correctly applied.

Example:

```
Router# show run lldp  
Tue Jan 28 05:43:11.868 UTC  
lldp  
timer 60  
interface-only  
!
```

