Enhancements to Data Models

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

NETCONF Accounting Logs

Table 1: 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 ASR 9000 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 2: 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. • /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface/ipv6/ • addresses/address[ip]/state/ip • addresses/address[ip]/state/prefix-length • addresses/address[ip]/state/origin • state/enabled • state/mtu • state/dup-addr-detect-transmits • state/counters/in-pkts • state/counters/in-octets • state/counters/out-pkts • state/counters/out-octets • state/openconfig-if-ip-ext:autoconf/create-global-addresses • /oc-if:interfaces/oc-if:interface/oc-if:subinterfaces/oc-if:subinterface/ipv4/ • addresses/address[ip]/state/ip • addresses/address[ip]/state/prefix-length • addresses/address[ip]/state/origin • state/mtu • state/dhcp-client • state/in-pkts • state/in-octets • state/out-pkts • state/out-octets You can access this data model from the Github repository.

OpenConfig Data Model Enhancements

Table 3: 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. 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/`: • `last-change` • `counters/lacp-timeout-transitions` 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: • openconfig-mpls • openconfig-mpls-te • openconfig-mpls-rsvp • openconfig-mpls-igp • openconfig-mpls-types • openconfig-mpls-sr You can access this data model from the Github repository.

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.

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

Table 5: 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. 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 6: 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 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 7: 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: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-entries>
```

```
      <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-entries>
```

```
    <acl-entry>
      <sequence-id>10</sequence-id>
       <config>
          <sequence-id>10</sequence-id>
       </config>
      <actions>
      <config>
        <forwarding-action>ACCEPT</forwarding-action>
     </config>
    </actions>
 <ipv6>
  <config>
    <dscp-set>22</dscp-set>
    <dscp-set>55</dscp-set>
  </config>
 </ipv6>
</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 200000001, flag 8, len 0
```


OpenConfig YANG Model:AFT

Table 10: 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. 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": "11529216420459399 38"
}
}" > >

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": "11529217794848539 82"
}
}" > >
```

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": "11530623795342370 25"
}
}" > >
```

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 11: 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, `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. 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 12: 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.

Event Name Displayed in the System Log	Description
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 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.