



## Enhancements to Data Models

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

- [Install Label in oc-platform Data Model](#), on page 1
- [OAM for MPLS and SR-MPLS in mpls-ping and mpls-traceroute Data Models](#), on page 3
- [OpenConfig YANG Model:dscp-set](#), on page 8
- [OpenConfig YANG Model:procmon](#), on page 11

## Install Label in oc-platform Data Model

**Table 1: Feature History Table**

Feature Name	Release Information	Description
Enhancements to openconfig-platform YANG Data Model	Release 7.3.2	<p>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:</p> <ul style="list-style-type: none"> <li>• software version</li> <li>• golden ISO (GISO) label</li> <li>• committed IOS XR packages</li> </ul> <p>You can access this data model from the <a href="#">Github</a> repository.</p>

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 2: Feature History Table

Feature Name	Release Information	Description
YANG Data Models for MPLS OAM RPCs	Release 7.3.2	This feature introduces the <code>Cisco-IOS-XR-mpls-ping-act</code> and <code>Cisco-IOS-XR-mpls-traceroute-act</code> 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 <a href="#">Github</a> 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>
          <dsmapi-label-stack>
            <dsmapi-label>
              <label>16003</label>
            </dsmapi-label>
          </dsmapi-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>
          <dsmapi-label-stack>
            <dsmapi-label>
              <label>3</label>
            </dsmapi-label>
          </dsmapi-label-stack>
          <return-code>8</return-code>
        </hop>
      </hops>
    </path>
  </paths>

```

```

    <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:dscp-set

**Table 3: Feature History Table**

Feature Name	Release Information	Description
OpenConfig YANG Model:dscp-set	Release 7.5.2	<p>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.</p> <p>You can access the OC data model from the <a href="#">Github</a> repository.</p>

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>
              </acl-entry>
            </acl-entries>
          </acl-set>
        </acl-sets>
        <ipv6>
          <config>
            <dscp-set>22</dscp-set>
            <dscp-set>55</dscp-set>
          </config>
        </ipv6>
      </acl-entry>
    </acl-entries>
  </acl-set>
</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 4: Feature History Table

Feature Name	Release Information	Description
OpenConfig YANG Model:procmon	Release 7.5.2	<p>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.</p> <p>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.</p> <p>You can access the OC data model from the <a href="#">Github</a> repository.</p>

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
```