Enhancements to Data Models

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

- Define Power State of Line Card Using Data Model, on page 1
- Install Label in oc-platform Data Model, on page 3
- OpenConfig YANG Model:MACsec, on page 5
- OpenConfig YANG Model:dscp-set, on page 11
- OpenConfig YANG Model:procmon, on page 14

Define Power State of Line Card Using Data Model

Table 1: Feature History Table

Feature Name	Release Information	Description
Control Line Card Power Using YANG Data Model	Release 7.5.1	The `oc-platform.yang` 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 2: 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:MACsec

Table 3: 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

Limitation

- The current implementation of Cisco IOS XR supports only the local time zone configuration in the YYYY-MM-DDTHH:MM:SS format for the following paths:
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/config/valid-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/config/expiration-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/state/valid-date-time
  - /macsec/mka/key-chains/key-chain/mka-keys/mka-key/state/expiration-date-time

- Under the MACsec policy, you can disable the delay-protection and include-icv-indicator leaves only by using the delete operation. You cannot modify the configuration by updating the default field value, from true to false. This codeblock shows a sample delete operation:

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

Running Configuration

```
RP/0/0/CPU0:ios#show running-config
Tue Apr 19 21:36:08.882 IST
Building configuration...
!! IOS XR Configuration 0.0.0
!! Last configuration change at Thu Apr 14 16:25:17 2022 by UNKNOWN
key chain kc
 macsec
  key 1234
   key-string password
00554155500E5D5157701E1D5D4C53404A5A5E577E7E727F6B647040534355560E080A00005B554F4E080A0407070303530A54540C0252445E550958525A771B16
 cryptographic-algorithm aes-256-cmac
   lifetime 00:01:01 january 01 2021 infinite
  netconf-yang agent
 ssh
interface GigabitEthernet0/0/0/0
 shutdown
interface GigabitEthernet0/0/0/1
 macsec psk-keychain kc
interface GigabitEthernet0/0/0/2
 macsec psk-keychain kc policy mp
interface GigabitEthernet0/0/0/3
 shutdown
interface GigabitEthernet0/0/0/4
 shutdown
macsec-policy mp
 cipher-suite GCM-AES-XPN-256
 key-server-priority 4
ssh server v2
end
```

RPC Request for get-config

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

RPC Response for get-config

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <macsec xmlns="http://openconfig.net/yang/macsec">
   <mka>
    <policies>
     <policy>
```

```
      <name>mp</name>
      <config>
       <name>mp</name>
       <macsec-cipher-suite>gcm-aes-xpn-256</macsec-cipher-suite>
       <key-server-priority>4</key-server-priority>
      </config>
     </policy>
    </policies>
    <key-chains>
     <key-chain>
      <name>kc</name>
      <config>
       <name>kc</name>
      </config>
      <mka-keys>
       <mka-key>
        <id>1234</id>
        <config>
         <id>1234</id>
         <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
         <valid-date-time>2021-01-01T00:01:01</valid-date-time>
         <expiration-date-time>NO_EXPIRATION</expiration-date-time>
        </config>
       </mka-key>
      </mka-keys>
     </key-chain>
    </key-chains>
   </mka>
   <interfaces>
    <interface>
     <name>GigabitEthernet0/0/0/1</name>
     <config>
      <name>GigabitEthernet0/0/0/1</name>
     </config>
     <mka>
      <config>
       <key-chain>kc</key-chain>
      </config>
     </mka>
    </interface>
    <interface>
     <name>GigabitEthernet0/0/0/2</name>
     <config>
      <name>GigabitEthernet0/0/0/2</name>
     </config>
     <mka>
      <config>
       <key-chain>kc</key-chain>
       <mka-policy>mp</mka-policy>
      </config>
     </mka>
    </interface>
   </interfaces>
  </macsec>
 </data>
</rpc-reply>
```

RPC Request for get

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

```
    </filter>
</get>
```

RPC Response for get

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <macsec xmlns="http://openconfig.net/yang/macsec">
   <mka>
    <policies>
     <policy>
      <name>mp</name>
      <config>
       <name>mp</name>
       <macsec-cipher-suite>gcm-aes-xpn-256</macsec-cipher-suite>
       <key-server-priority>4</key-server-priority>
      </config>
      <state>
       <name>mp</name>
       <key-server-priority>4</key-server-priority>
       <macsec-cipher-suite>gcm-aes-xpn256</macsec-cipher-suite>
       <confidentiality-offset>zero-bytes</confidentiality-offset>
       <delay-protection>false</delay-protection>
       <include-icv-indicator>false</include-icv-indicator>
       <sak-rekey-interval>0</sak-rekey-interval>
      </state>
     </policy>
     <policy>
      <name>DEFAULT-POLICY</name>
      <state>
       <name>DEFAULT-POLICY</name>
       <key-server-priority>16</key-server-priority>
       <macsec-cipher-suite>gcm-aes-xpn256</macsec-cipher-suite>
       <confidentiality-offset>zero-bytes</confidentiality-offset>
       <delay-protection>false</delay-protection>
       <include-icv-indicator>false</include-icv-indicator>
       <sak-rekey-interval>0</sak-rekey-interval>
      </state>
     </policy>
    </policies>
    <key-chains>
     <key-chain>
      <name>kc</name>
      <config>
       <name>kc</name>
      </config>
      <mka-keys>
       <mka-key>
        <id>1234</id>
        <config>
         <id>1234</id>
         <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
         <valid-date-time>2021-01-01T00:01:01</valid-date-time>
         <expiration-date-time>NO_EXPIRATION</expiration-date-time>
        </config>
        <state>
         <id>1234</id>
         <cryptographic-algorithm>AES_256_CMAC</cryptographic-algorithm>
         <valid-date-time>2021-01-01T00:01:01</valid-date-time>
         <expiration-date-time>NO_EXPIRATION</expiration-date-time>
        </state>
       </mka-key>
      </mka-keys>
```

```
    <state>
     <name>kc</name>
    </state>
   </key-chain>
  </key-chains>
 </mka>
 <interfaces>
  <interface>
   <name>GigabitEthernet0_0_0_1</name>
   <state>
    <name>GigabitEthernet0_0_0_1</name>
    <counters>
     <tx-untagged-pkts>8</tx-untagged-pkts>
     <rx-untagged-pkts>0</rx-untagged-pkts>
     <rx-badtag-pkts>2</rx-badtag-pkts>
     <rx-unknownsci-pkts>3</rx-unknownsci-pkts>
     <rx-nosci-pkts>4</rx-nosci-pkts>
    </counters>
   </state>
   <mka>
    <state>
     <mka-policy>DEFAULT-POLICY</mka-policy>
     <key-chain>kc</key-chain>
     <counters>
      <in-mkpdu>0</in-mkpdu>
      <in-sak-mkpdu>0</in-sak-mkpdu>
      <out-mkpdu>225271</out-mkpdu>
      <out-sak-mkpdu>0</out-sak-mkpdu>
     </counters>
    </state>
   </mka>
   <scsa-tx>
    <scsa-tx>
     <sci-tx>024f88a08c9d0001</sci-tx>
     <state>
      <sci-tx>024f88a08c9d0001</sci-tx>
      <counters>
       <sc-encrypted>0</sc-encrypted>
       <sa-encrypted>0</sa-encrypted>
      </counters>
     </state>
    </scsa-tx>
   </scsa-tx>
  </interface>
  <interface>
   <name>GigabitEthernet0_0_0_2</name>
   <state>
    <name>GigabitEthernet0_0_0_2</name>
    <counters>
     <tx-untagged-pkts>8</tx-untagged-pkts>
     <rx-untagged-pkts>0</rx-untagged-pkts>
     <rx-badtag-pkts>2</rx-badtag-pkts>
     <rx-unknownsci-pkts>3</rx-unknownsci-pkts>
     <rx-nosci-pkts>4</rx-nosci-pkts>
    </counters>
   </state>
   <mka>
    <state>
     <mka-policy>mp</mka-policy>
     <key-chain>kc</key-chain>
     <counters>
      <in-mkpdu>0</in-mkpdu>
      <in-sak-mkpdu>0</in-sak-mkpdu>
      <out-mkpdu>225271</out-mkpdu>
```

```
          <out-sak-mkpdu>0</out-sak-mkpdu>
         </counters>
        </state>
       </mka>
       <scsa-tx>
        <scsa-tx>
         <sci-tx>0246c822daae0001</sci-tx>
         <state>
          <sci-tx>0246c822daae0001</sci-tx>
          <counters>
           <sc-encrypted>0</sc-encrypted>
           <sa-encrypted>0</sa-encrypted>
          </counters>
         </state>
        </scsa-tx>
       </scsa-tx>
      </interface>
      <interface>
       <name>GigabitEthernet0/0/0/1</name>
       <config>
        <name>GigabitEthernet0/0/0/1</name>
       </config>
       <mka>
        <config>
         <key-chain>kc</key-chain>
        </config>
       </mka>
      </interface>
      <interface>
       <name>GigabitEthernet0/0/0/2</name>
       <config>
        <name>GigabitEthernet0/0/0/2</name>
       </config>
       <mka>
        <config>
         <key-chain>kc</key-chain>
         <mka-policy>mp</mka-policy>
        </config>
       </mka>
      </interface>
     </interfaces>
    </macsec>
   </data>
  </rpc-reply>
```

OpenConfig YANG Model:dscp-set

Table 4: 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 5: 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
```