Use NETCONF Protocol to Define Network Operations with Data Models

XR devices ship with the YANG files that define the data models they support. Using a management protocol such as NETCONF, you can programmatically query a device for the list of models it supports and retrieve the model files.

Network Configuration Protocol (NETCONF) is a standard transport protocol that communicates with network devices. NETCONF provides mechanisms to edit configuration data and retrieve operational data from network devices. The configuration data represents the way interfaces, routing protocols and other network features are provisioned. The operational data represents the interface statistics, memory utilization, errors, and so on.

NETCONF uses an Extensible Markup Language (XML)-based data encoding for the configuration data, as well as protocol messages. It uses a simple RPC-based (Remote Procedure Call) mechanism to facilitate communication between a client and a server. The client can be a script or application that runs as part of a network manager. The server is a network device such as a router. NETCONF defines how to communicate with the devices, but does not handle what data is exchanged between the client and the server.

To enable NETCONF, run the following commands:

```
RP/0/RP0:ios#configure
Sun Jun 21 22:32:45.159 IST
RP/0/RP0:ios(config)#netconf agent tty
RP/0/RP0:ios(config-netconf-tty)#netconf-yang agent
RP/0/RP0:ios(config-ncy-agent)#ssh server netconf vrf default
RP/0/RP0:ios(config)#commit
Sun Jun 21 22:33:07.995 IST
RP/0/RP0:ios(config)#end
```

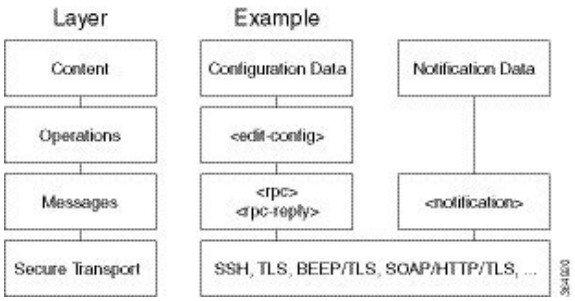
NETCONF Session

A NETCONF session is the logical connection between a network configuration application (client) and a network device (router). The configuration attributes can be changed during any authorized session; the effects are visible in all sessions. NETCONF is connection-oriented, with SSH as the underlying transport. NETCONF sessions are established with a "hello" message, where features and capabilities are announced. Sessions are terminated using *close* or *kill* messages.

NETCONF Layers

NETCONF protocol can be partitioned into four layers:

Figure 1: NETCONF Layers



- **Content layer:** includes configuration and notification data
- **Operations layer:** defines a set of base protocol operations invoked as RPC methods with XML-encoded parameters
- **Messages layer:** provides a simple, transport-independent framing mechanism for encoding RPCs and notifications
- **Secure Transport layer:** provides a communication path between the client and the server

For more information about NETCONF, refer RFC 6241.

This article describes, with a use case to configure the local time on a router, how data models help in a faster programmatic configuration as comapared to CLI.

NETCONF Operations

NETCONF defines one or more configuration datastores and allows configuration operations on the datastores. A configuration datastore is a complete set of configuration data that is required to get a device from its initial default state into a desired operational state. The configuration datastore does not include state data or executive commands.

The base protocol includes the following NETCONF operations:

```
|  +--get-config
|  +--edit-Config
|     +--merge
|     +--replace
|     +--create
|     +--delete
|     +--remove
|     +--default-operations
|        +--merge
|        +--replace
|        +--none
|  +--get
```

These NETCONF operations are described in the following table:

NETCONF Operation	Description	Example

<get-config>	Retrieves all or part of a specified configuration from a named data store	Retrieves the nsf configuration using Cisco-IOS-XR-clns-isis-cfg.
		`<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:0459351e-d60f-4c6d-a195-ffee7efb3921">`
		`<get-config>`
		`<source>`
		`<running/>`
		`</source>`
		`<filter>`
		`<isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg">`
		`<instances>`
		`<instance>`
		`<nsf/>`
		`</instance>`
		`</instances>`
		`</isis>`
		`</filter>`
		`</get-config>`
		`</rpc>`

<get>	Retrieves running configuration and device state information	Retrieves the optics configuration on particular interface.
		`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">`
		`<get>`
		`<filter>`
		`<optics-oper xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-optics-oper">`
		`<optics-ports>`
		`<optics-port>`
		`<name>Optics0/2/0/11</name>`
		`<optics-db-info/>`
		`</optics-port>`
		`</optics-ports>`
		`</optics-oper>`
		`</filter>`
		`</get>`
		`</rpc>`

NETCONF Operation	Description	Example

<edit-config>	Loads all or part of a specified configuration to the specified target configuration	Edits the nsf configuration using Cisco-IOS-XR-clns-isis-cfg
		`<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:68baa1ad-d574-4530-ac65-fd83c8cbf2ea">`
		`<edit-config>`
		`<target>`
		`<candidate/>`
		`</target>`
		`<config>`
		`<isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg">`
		`<instances>`
		`<instance>`
		`<instance-name>100</instance-name>`
		`<nsf>`
		`<flavor>cisco-proprietary-nsf</flavor>`
		`</nsf>`
		`</instance>`
		`</instances>`
		`</isis>`
		`</config>`
		`</edit-config>`
		`</rpc>`
		`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102">`
		`<commit/>`
		`</rpc>`

NETCONF Operation to Get Configuration

The <rpc> element in the request and response messages enclose a NETCONF request sent between the client and the router. The `message-id` attribute in the <rpc> element is mandatory. This attribute is a string chosen by the sender and encodes an integer. The receiver of the <rpc> element does not decode or interpret this string but simply saves it to be used in the <rpc-reply> message. The sender must ensure that the `message-id` value is normalized. When the client receives information from the server, the <rpc-reply> message contains the same `message-id`.

This example shows how a NETCONF <get-config> and <edit-config> request works for the ISIS configuration.

To retrieve the nsf configuration using Cisco-IOS-XR-clns-isis-cfg:

Netconf Request (Client to Router)	Netconf Response (Router to Client)
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:0459351e-d60f-4c6d-a195-ffee7efb3921"> <get-config> <source> <running/> </source> <filter> <isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg"> <instances> <instance> <nsf/> </instance> </instances> </isis> </filter> </get-config> </rpc>	<rpc-reply message-id="urn:uuid:0459351e-d60f-4c6d-a195-ffee7efb3921" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg"> <instances> <instance> <instance-name>100</instance-name> <nsf> <**flavor>ietf-standard-nsf</flavor>** </nsf> </instance> </instances> </isis> </data> </rpc-reply>

The following displays the command output from the router:

```
RP/0/RP0:ios#show run router isis
Fri Jul  3 22:22:22.150 IST
router isis 100
is-type level-2-only
net 49.0000.0000.0000.0001.00
nsr
nsf ietf
address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback1
```

To edit the nsf configuration, using Cisco-IOS-XR-clns-isis-cfg:

Netconf Request (Client to Router)	Netconf Response (Router to Client)
<rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:68baa1ad-d574-4530-ac65-fd83c8cbf2ea"> <edit-config> <target> <candidate/> </target> <config> <isis xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-clns-isis-cfg"> <instances> <instance> <instance-name>100</instance-name> <nsf> <flavor>cisco-proprietary-nsf</flavor> </nsf> </instance> </instances> </isis> </config> </edit-config> </rpc> <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102"> <commit/> </rpc>	<?xml version="1.0" ?> <rpc-reply message-id="urn:uuid:68baa1ad-d574-4530-ac65-fd83c8cbf2ea" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>

After a successful edit config operation, the router configuration is changed to:

```
RP/0/RP0:ios#show run router isis
Fri Jul  3 22:25:33.793 IST
router isis 100
is-type level-2-only
net 49.0000.0000.0000.0001.00
nsr
nsf cisco
address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback1
```

Configure an Interface Port Mode Using Data Model in a NETCONF Session

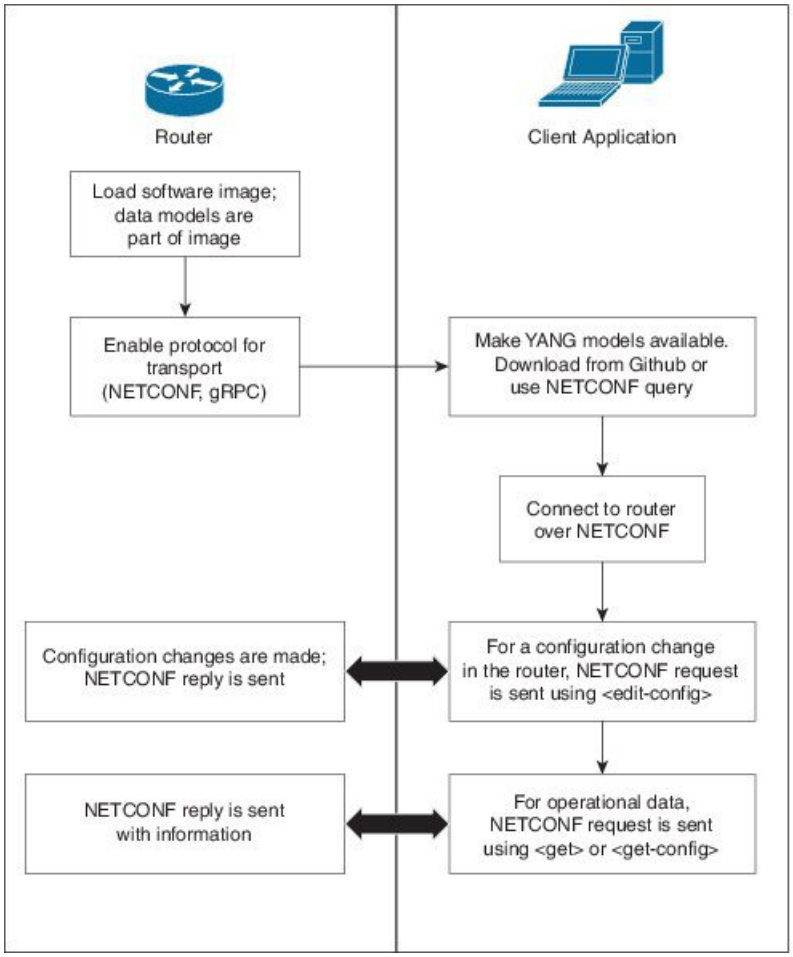
NETCONF is an XML-based protocol used over Secure Shell (SSH) transport to configure a network. The client applications use this protocol to request information from the router, and make configuration changes to the router.

The process for using data models involves:

- Obtain the data models.
- Establish a connection between the router and the client using NETCONF communication protocol.
- Manage the configuration of the router from the client using data models.

The following image shows the tasks involved in using data models.

Figure 2: Process for Using Data Models



In this section, you use the native data model `Cisco-IOS-XR-portmode-cfg.yang` to programmatically configure and verify the port mode of an interface using a NETCONF session.

Procedure

Step 1 Explore the native configuration model for the port mode configuration on the router.

Example:

```
RP/0/RP0:ios#show netconf-yang capabilities | in "Cisco-IOS-XR-portmode-cfg"
Mon Aug 24 15:34:14.483 IST
http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg                    |2020-02-24|
RP/0/RP0:ios#
```

Step 2 Explore the native configuration model on the NETCONF client.

The `Cisco-IOS-XR-portmode-cfg` is present within `Cisco-IOS-XR-ifmgr-cfg`. When you load `Cisco-IOS-XR-portmode-cfg`, the `Cisco-IOS-XR-ifmgr-cfg` automatically loads and you can find the `Cisco-IOS-XR-portmode-cfg` when you expand the `Cisco-IOS-XR-ifmgr-cfg`.

Step 3 Retrieve the current port mode configuration on the router.

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:7d7b08bb-a688-4b4a-bf76-b9367bfaa150"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
      <interface-configuration>
        <active>act</active>
        <interface-name>Optics0/3/0/6</interface-name>
        <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg">
          <info>
            <rate>none</rate>
            <mapping>none</mapping>
            <framing>opu3</framing>
            <type>otn</type>
          </info>
        </port-mode>
      </interface-configuration>
    </interface-configurations>
  </data>
</rpc-reply>
```

Step 4 Change the port mode configuration using the `edit-config` option.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
        <interface-configuration>
          <active>act</active>
          <interface-name>Optics0/3/0/6</interface-name>
          <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg">
            <info>
              <rate>none</rate>
              <mapping>none</mapping>
              <framing>opu2</framing>
              <type>otn</type>
            </info>
          </port-mode>
        </interface-configuration>
      </interface-configurations>
    </config>
  </edit-config>
</rpc>
```

Step 5 View the updated port mode configuration of the interface.

```
<?xml version="1.0" ?>
<rpc-reply message-id="urn:uuid:a122c79d-1cb0-4a02-b4e2-3cc5c7efef9a"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
```

```
    <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
      <interface-configuration>
        <active>act</active>
        <interface-name>Optics0/3/0/6</interface-name>
        <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg">
          <info>
            <rate>none</rate>
            <mapping>none</mapping>
            <framing>opu2</framing>
            <type>otn</type>
          </info>
        </port-mode>
      </interface-configuration>
   </interface-configurations>
  </data>
</rpc-reply>
```

Configure Breakouts Using Data Model in a NETCONF Session

In this section, you use the native data model `Cisco-IOS-XR-portmode-cfg.yang` to programmatically configure the breakout for an interface using a NETCONF session.

Procedure

Step 1 Explore the native configuration model for the port mode configuration on the router.

Example:

```
RP/0/RP0:ios#show netconf-yang capabilities | in "Cisco-IOS-XR-portmode-cfg"
Mon Aug 24 15:34:14.483 IST
http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg                              |2020-02-24|
RP/0/RP0:ios#
```

Step 2 Explore the native configuration model on the NETCONF client.

The `Cisco-IOS-XR-portmode-cfg` is present within `Cisco-IOS-XR-ifmgr-cfg`. When you load `Cisco-IOS-XR-portmode-cfg`, the `Cisco-IOS-XR-ifmgr-cfg` automatically loads and you can find the `Cisco-IOS-XR-portmode-cfg` when you expand the `Cisco-IOS-XR-ifmgr-cfg`.

Step 3 To configure the breakout, use the `edit-config` option.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target><candidate/></target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration>
    <active>act</active>
    <interface-name>Optics0/7/0/0</interface-name>
    <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg">
     <lane-numbers>
      <lane-number>
       <lane-no>1</lane-no>
       <rate>10ge</rate>
       <mapping>none</mapping>
```

```
       <framing>packet</framing>
       <type>ethernet</type>
      </lane-number>
     </lane-numbers>
    </port-mode>
   </interface-configuration>
   </interface-configurations>
 </config>
</edit-config>
</rpc>
```

Note When you configure breakouts for ethernet packets through NETCONF, you must explicitly define the rate as "10GE" and the mapping as "None".

Step 4 To view the updated port mode configuration of the interface, use the `get` option.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter>
<interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration>
      <interface-name>Optics0/7/0/0</interface-name>
   </interface-configuration>
    </interface-configurations>
    </filter>
</get>

 </rpc>
```

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration>
    <active>act</active>
    <interface-name>Optics0/7/0/0</interface-name>
    <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg">
     <lane-numbers>
      <lane-number>
       <lane-no>1</lane-no>
       <rate>10ge</rate>
       <mapping>none</mapping>
       <framing>packet</framing>
       <type>ethernet</type>
      </lane-number>
     </lane-numbers>
    </port-mode>
   </interface-configuration>
  </interface-configurations>
#22
 </data>
</rpc-reply>
```

Step 5 (Optional) To delete the breakout, use the `edit-config` option.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
<target><candidate/></target>
<config xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0">
<interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg">
   <interface-configuration>
    <active>act</active>
    <interface-name>Optics0/7/0/0</interface-name>
    <port-mode xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-portmode-cfg"
```

```
xc:operation="delete">
     <lane-numbers>
      <lane-number>
       <lane-no>1</lane-no>
       <rate>10ge</rate>
       <mapping>none</mapping>
       <framing>packet</framing>
       <type>ethernet</type>
      </lane-number>
     </lane-numbers>
    </port-mode>
   </interface-configuration>
   </interface-configurations>
 </config>
</edit-config>
</rpc>
```