Cisco NCS 1010 Data Models Configuration Guide, IOS XR Release 7.7.x

First Published: 2022-05-29

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS

CHAPTER 1

Data Models

Data Models - Programmatic and Standards-based Configuration

Cisco IOS XR software supports the automation of configuration of multiple routers across the network using Data models. Configuring routers using data models overcomes drawbacks posed by traditional router management techniques.

CLIs are widely used for configuring a router and for obtaining router statistics. Other actions on the router, such as, switch-over, reload, process restart are also CLI-based. Although, CLIs are heavily used, they have many restrictions.

Customer needs are fast evolving. Typically, a network center is a heterogenous mix of various devices at multiple layers of the network. Bulk and automatic configurations need to be accomplished. CLI scraping is not flexible and optimal. Re-writing scripts many times, even for small configuration changes is cumbersome. Bulk configuration changes through CLIs are error-prone and may cause system issues. The solution lies in using data models - a programmatic and standards-based way of writing configurations to any network device, replacing the process of manual configuration. Data models are written in a standard, industry-defined language. Although configurations using CLIs are easier (more human-friendly), automating the configuration using data models results in scalability.

Cisco IOS XR supports the YANG data modeling language. YANG can be used with Network Configuration Protocol (NETCONF) to provide the desired solution of automated and programmable network operations.

YANG model

YANG is a data modeling language used to describe configuration and operational data, remote procedure calls and notifications for network devices. The salient features of YANG are:

- Human-readable format, easy to learn and represent
- Supports definition of operations
- Reusable types and groupings

- Data modularity through modules and submodules
- Supports the definition of operations (RPCs)
- Well-defined versioning rules
- Extensibility through augmentation

For more details of YANG, refer RFC 6020 and 6087.

NETCONF and gRPC (Google Remote Procedute Call) provide a mechanism to exchange configuration and operational data between a client application and a router and the YANG models define a valid structure for the data (that is being exchanged).

Protocol	**Transport**	**Encoding/ Decoding**
NETCONF	SSH	XML
gRPC	HTTP/2	XML, JSON

Each feature has a defined YANG model. Cisco-specific YANG models are referred to as synthesized models. Some of the standard bodies, such as IETF , IEEE and Open Config, are working on providing an industry-wide standard YANG models that are referred to as common models.

Components of Yang model

A module defines a single data model. However, a module can reference definitions in other modules and submodules by using the **import** statement to import external modules or the **include** statement to include one or more submodules. A module can provide augmentations to another module by using the **augment** statement to define the placement of the new nodes in the data model hierarchy and the **when** statement to define the conditions under which the new nodes are valid. **Prefix** is used when referencing definitions in the imported module.

YANG models are available for configuring a feature and to get operational state (similar to show commands)

This is the configuration YANG model for AAA (denoted by - cfg)

```
(snippet)
module Cisco-IOS-XR-aaa-locald-cfg {

  /*** NAMESPACE / PREFIX DEFINITION ***/

  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-cfg";


  prefix "aaa-locald-cfg";

  /*** LINKAGE (IMPORTS / INCLUDES) ***/

  import Cisco-IOS-XR-types { prefix "xr"; }

  import Cisco-IOS-XR-aaa-lib-cfg { prefix "a1"; }

  /*** META INFORMATION ***/

  organization "Cisco Systems, Inc.";
          .........................
          ......................... (truncated)
```

This is the operational YANG model for AAA (denoted by -oper)

```
(snippet)
module Cisco-IOS-XR-aaa-locald-oper {

  /*** NAMESPACE / PREFIX DEFINITION ***/

  namespace "http://cisco.com/ns/yang/Cisco-IOS-XR-aaa-locald-oper";


  prefix "aaa-locald-oper";

  /*** LINKAGE (IMPORTS / INCLUDES) ***/

  import Cisco-IOS-XR-types { prefix "xr"; }

  include Cisco-IOS-XR-aaa-locald-oper-sub1 {
    revision-date 2015-01-07;
  }

  /*** META INFORMATION ***/

  organization "Cisco Systems, Inc.";
    ........................
    ........................ (truncated)
```

Note A module may include any number of sub-modules, but each sub-module may belong to only one module. The names of all standard modules and sub-modules must be unique.

Data types

YANG defines data types for leaf values. These data types help the user in understanding the relevant input for a leaf.

Name	Description
binary	Any binary data
bits	A set of bits or flags
boolean	"true" or "false"
decimal64	64-bit signed decimal number
empty	A leaf that does not have any value
enumeration	Enumerated strings
identityref	A reference to an abstract identity
instance-identifier	References a data tree node
int (integer-defined values)	8-bit, 16-bit, 32-bit, 64-bit signed integers
leafref	A reference to a leaf instance
uint	8-bit, 16-bit, 32-bit, 64-bit unsigned intergers

Name	Description
string	Human-readable string
union	Choice of member types

Data Model and CLI Comparison

Each feature has a defined YANG model that is synthesized from the schemas. A model in a tree format includes:

- Top level nodes and their subtrees
- Subtrees that augment nodes in other yang models
- Custom RPCs

The options available using the CLI are defined as leaf-nodes in data models. The defined data types, indicated corresponding to each leaf-node, help the user to understand the required inputs.

gRPC

gRPC is a language-neutral, open source, RPC (Remote Procedute Call) system developed by Google. By default, it uses protocol buffers as the binary serialization protocol. It can be used with other serialization protocols as well such as JSON, XML etc. The user needs to define the structure by defining protocol buffer message types in*.proto* files. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs.

gRPC encodes requests and responses in binary. Although Protobufs was the only format supported in the initial release, gRPC is extensible to other content types. The Protobuf binary data object in gRPC is transported using HTTP/2 (RFC 7540). HTTP/2 is a replacement for HTTP that has been optimized for high performance. HTTP/2 provides many powerful capabilities including bidirectional streaming, flow control, header compression and multi-plexing. gRPC builds on those features, adding libraries for application-layer flow-control, load-balancing and call-cancellation.

gRPC supports distributed applications and services between a client and server. gRPC provides the infrastructure to build a device management service to exchange configuration and operational data between a client and a server in which the structure of the data is defined by YANG models.

Cisco gRPC IDL

The protocol buffers interface definition language (IDL) is used to define service methods, and define parameters and return types as protocol buffer message types.

gRPC requests can be encoded and sent across to the router using JSON. gRPC IDL also supports the exchange of CLI.

For gRPC transport, gRPC IDL is defined in .proto format. Clients can invoke the RPC calls defined in the IDL to program XR. The supported operations are - Get, Merge, Delete, Replace. The gRPC JSON arguments are defined in the IDL.

```
syntax = "proto3";

package IOSXRExtensibleManagabilityService;
```

```
service gRPCConfigOper {

    rpc GetConfig(ConfigGetArgs) returns(stream ConfigGetReply) {};

    rpc MergeConfig(ConfigArgs) returns(ConfigReply) {};

    rpc DeleteConfig(ConfigArgs) returns(ConfigReply) {};

    rpc ReplaceConfig(ConfigArgs) returns(ConfigReply) {};

    rpc CliConfig(CliConfigArgs) returns(CliConfigReply) {};

}
```

gRPC Operations

- oper get-config—Retrieves a configuration
- oper merge-config— Appends to an existing configuration
- oper delete-config—Deletes a configuration
- oper replace-config—Modifies a part of an existing configuration
- oper get-oper—Gets operational data using JSON
- oper cli-config—Performs a configuration
- oper showcmdtextoutput

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
|  +--get
|  +--lock
|  +--unLock
|  +--close-session
|  +--kill-session
```

These NETCONF operations are described in the following table:

Table 1:

NETCONF Operation	Description	ExampleDescription
<get-config>	Retrieves specific controller configuration details from running configuration using filter option.	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get-config> <source> <running/> </source> <filter> <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg"> <interface-configuration> <active>act</active> <interface-name>Ots0/0/0/0</interface-name> </interface-configuration> </interface-configurations> </filter> </get-config> </rpc>
<get>	Retrieves all OTS controllers state information.	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get> <filter> <ots-oper xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-ots-oper"/> </filter> </get> </rpc>

NETCONF Operation	Description	ExampleDescription
<edit-config>	Configure OTS controller configurations using Merge operation.	<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>Ots0/0/0/0</interface-name> <ots xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-ots-cfg"> <ots-egress-amplifier-gain>132</ots-egress-amplifier-gain> <ots-egress-amplifier-tilt>0</ots-egress-amplifier-tilt> <ots-egress-amplifier-gain-range>normal</ots-egress-amplifier-gain-range> </ots> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc> Commit: <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="102"> <commit/> </rpc>
<lock>	Locks the running configuration.	Request: <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <lock> <target> <running/> </target> </lock> </rpc> Response: <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>

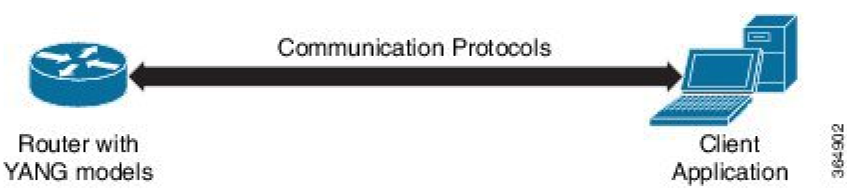
NETCONF Operation	Description	ExampleDescription
<unlock>	Unlocks the running configuration from the same session:	Request: rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <unlock> <target> <running/> </target> </unlock> </rpc> Response <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>
<close session>	Closes a NETCONF session.	Request: <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <close-session/> </rpc> Response: <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <ok/> </rpc-reply>

CHAPTER 2

Use Data Models

Use Data Models

Figure 1: Workflow for using Data models



The above illustration gives a quick snap shot of how YANG can be used with Netconf in configuring a network device using a client application.

The tasks that help the user to implement Data model configuration are listed here.

1. Load the software image ; the YANG models are a part of the software image. Alternatively, the YANG models can also be downloaded from:

   ```
   https://github.com/YangModels/yang/tree/master/vendor/cisco/xr
   ```

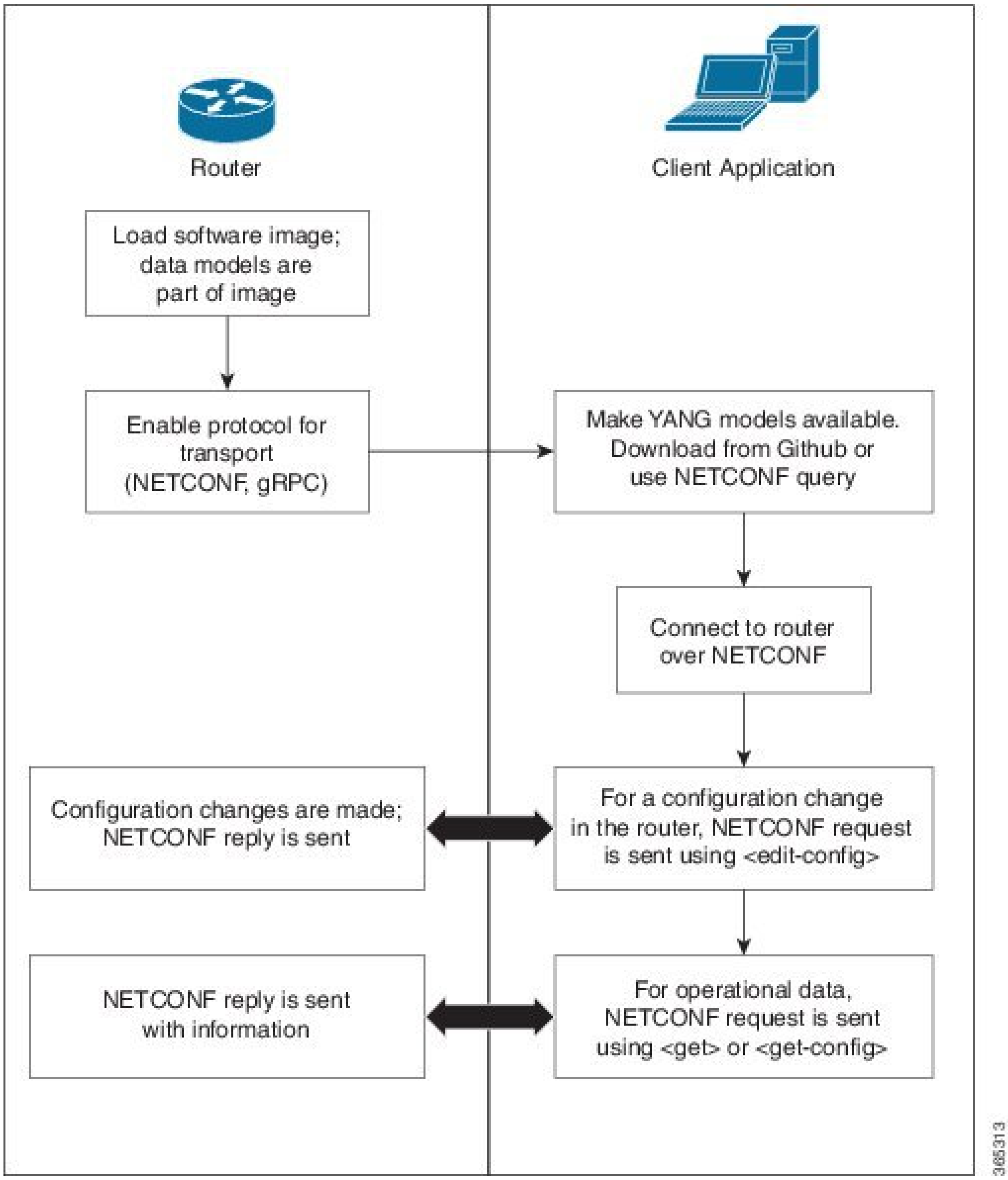
   Users can also query using NETCONF to get the list of models.

   ```
   <?xml version="1.0" encoding="utf-8"?>
   <rpc message-id="100" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
       <get>
           <filter type="subtree">
               <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">

                   <schemas/>
               </netconf-state>
           </filter>
       </get>
   </rpc>
   ```

2. Communication between the router and the application happens by Netconf over SSH. Enable Netconf on the router on a suitable port.
3. From the client application, connect to the router using Netconf over SSH. Run Netconf operations to make configuration changes or get operational data.

Figure 2: Lane Diagram to show the router and client application operations



Enabling Netconf

This task enables Netconf over SSH.

Before you begin

Generate relevant crypto keys.

Step 1 **netconf-yang agent ssh**

Enables the Netconf agent process.

Step 2 **ssh server netconf**

Enables Netconf.

Step 3 **ssh server v2**

Enables SSH on the device and enables Netconf on port 22 if the Netconf agent process is enabled.

What to do next

The **netconf-yang agent session** command enables the user to set session parameters.

```
netconf-yang agent session {limit value |  absolute-timeout value |  idle-timeout value}
```

where,

- **limit** *value*- sets the maximum count for concurrent netconf-yang sessions. Range is 1 to 1024. The default value is 50.
- **absolute-timeout** *value*- sets the absolute session lifetime. Range is 1 to 1440 (in minutes).
- **idle-timeout** *value*- sets the idle session lifetime. Range is 1 to 1440 (in minutes).

Enabling gRPC

Use the following procedure to enable gRPC over HTTPS/2. gRPC supports both, the IPv4 and IPv6 address families (default is IPv4).

Step 1 Install the GO client. For more details on installing the GO client, see https://golang.org/doc/install.

Step 2 Configure the gRPC port, using the **grpc port** command.

```
RP/0/RP0/CPU0:ios(config)#grpc
RP/0/RP0/CPU0:ios(config)#port 57400
RP/0/RP0/CPU0:ios(config)#tls
RP/0/RP0/CPU0:ios(config)#commit
```

Port can range from 57344 to 57999. If a port is unavailable, an error is displayed.

CHAPTER 3

Supported Yang Models

Supported Yang Models

The following is the list of supported config, and oper YANG models for NCS 1010:

Config Models	Oper Models
Cisco-IOS-XR-osa-linesystem-cfg.yang	Cisco-IOS-XR-osa-hwmod-linesys-oper.yang
Cisco-IOS-XR-controller-ots-cfg.yang	Cisco-IOS-XR-controller-ots-oper.yang
Cisco-IOS-XR-ots-och-cfg.yang	Cisco-IOS-XR-controller-ots-och-oper.yang
Cisco-IOS-XR-controller-oms-cfg	Cisco-IOS-XR-controller-oms-oper.yang
Cisco-IOS-XR-controller-och-cfg	Cisco-IOS-XR-controller-och-oper.yang

Config Models	Oper Models
Cisco-IOS-XR-controller-osc-cfg.yang	Cisco-IOS-XR-controller-osc-oper.yang
Cisco-IOS-XR-controller-dfb-cfg.yang	Cisco-IOS-XR-controller-dfb-oper.yang
Cisco-IOS-XR-pmengine-cfg.yang	Cisco-IOS-XR-pmengine-oper.yang
Cisco-IOS-XR-olc-cfg.yang	Cisco-IOS-XR-olc-oper.yang
Cisco-IOS-XR-fpd-infra-cfg	Cisco-IOS-XR-show-fpd-loc-ng-oper
Cisco-IOS-XR-osa-ct-cfg	Cisco-IOS-XR-alarmgr-server-oper.yang
	Cisco-IOS-XR-platform-oper

The following is the list of supported Open Config models:

openconfig-optical-amplifier@2019-12-06
openconfig-optical-attenuator@2019-07-19

Structure of Yang Models

YANG data models can be represented in a hierarchical, tree-based structure with nodes, which makes them more easily understandable. YANG defines four nodes types. Each node has a name, and depending on the node type, the node might either define a value or contain a set of child nodes. The nodes types (for data modeling) are:

- leaf node—Contains a single value of a specific type
- list node—Contains a sequence of list entries, each of which is uniquely identified by one or more keys leafs
- leaf-list node—Contains a sequence of leaf nodes
- container node—Contains a grouping of related nodes containing only child nodes, which can be any of the four node types

The following is the tree structure of the openconfig-optical-amplifier model.

Note Cisco NCS 1010 supports only the leaves that are highlighted as bold in the following open configuration models.

```
 +--rw optical-amplifier
+--rw amplifiers
| +--rw amplifier* [name]
| +--rw name -> ../config/name
| +--rw config
| | +--rw name? string
| | +--rw type? identityref
| | +--rw target-gain? decimal64
| | +--rw min-gain? decimal64
| | +--rw max-gain? decimal64
```

```
| | +--rw target-gain-tilt? decimal64
| | +--rw gain-range? identityref
| | +--rw amp-mode? identityref
| | +--rw target-output-power? decimal64
| | +--rw max-output-power? decimal64
| | +--rw enabled? boolean
| | +--rw fiber-type-profile? identityref
| +--ro state
| +--ro name? string
| +--ro type? identityref
| +--ro target-gain? decimal64
| +--ro min-gain? decimal64
| +--ro max-gain? decimal64
| +--ro target-gain-tilt? decimal64
| +--ro gain-range? identityref
| +--ro amp-mode? identityref
| +--ro target-output-power? decimal64
| +--ro max-output-power? decimal64
| +--ro enabled? boolean
| +--ro fiber-type-profile? identityref
| +--ro component? -> /oc-platform:components/component/name
| +--ro ingress-port? -> /oc-platform:components/component/name
| +--ro egress-port? -> /oc-platform:components/component/name
| +--ro actual-gain
| +--ro actual-gain-tilt
| +--ro input-power-total
| +--ro input-power-c-band
| +--ro input-power-l-band
| +--ro output-power-total
| +--ro output-power-c-band
| +--ro output-power-l-band
| +--ro laser-bias-current
| +--ro optical-return-loss
+--rw supervisory-channels
+--rw supervisory-channel* [interface]
+--rw interface -> ../config/interface
```

The following is a tree structure of the openconfig-optical-attenuator model.

```
+--rw optical-attenuator
+--rw attenuators
+--rw attenuator* [name]
+--rw name -> ../config/name
+--rw config
| +--rw name? string
| +--rw attenuation-mode?
| +--rw target-output-power? decimal64
| +--rw attenuation? decimal64
| +--rw enabled? boolean
+--ro state
+--ro name? string
+--ro attenuation-mode? identityref
+--ro target-output-power? decimal64
+--ro attenuation? decimal64
+--ro enabled? boolean
+--ro component? -> /oc-platform:components/component/name
+--ro ingress-port? -> /oc-platform:components/component/name
+--ro egress-port? -> /oc-platform:components/component/name
+--ro actual-attenuation
+--ro output-power-total
+--ro optical-return-loss
```

The following is a sample tree structure of Cisco-IOS-XR-controller-ots-oper model.

```
+--ro ots-oper
       +--ro ots-ports
          +--ro ots-port* [name]
             +--ro ots-info
             |  +--ro raman-tx-power
             |  |  +--ro raman-tx-power*
             |  |     +--ro raman-tx-power-instance?   uint32
             |  |     +--ro raman-tx-power-value?      uint32
             |  |     +--ro raman-tx-wavelength?       uint32
             |  +--ro transmit-n-power
             |  |  +--ro transmit-power*
             |  |     +--ro instance?   uint32
             |  |     +--ro value?      int32
             |  +--ro receive-n-power
             |  |  +--ro receive-power*
             |  |     +--ro instance?   uint32
             |  |     +--ro value?      int32
             |  +--ro ingress-channel-slice-attenuation
             |  |  +--ro ingress-channel-slice*
             |  |     +--ro ingress-channel-slice?               uint32
             |  |     +--ro ingress-channel-slice-attenuation?   int32
             |  +--ro egress-channel-slice-attenuation
             |  |  +--ro egress-channel-slice*
             |  |     +--ro egress-channel-slice?               uint32
             |  |     +--ro egress-channel-slice-attenuation?   int32
             |  +--ro raman-tx-power-config
             |  |  +--ro raman-tx-power*
             |  |     +--ro raman-tx-power-instance?   uint32
             |  |     +--ro raman-tx-power-value?      uint32
             |  +--ro ingress-channel-slice-attenuation-configured
             |  |  +--ro ingress-channel-slice*
             |  |     +--ro ingress-channel-slice?               uint32
             |  |     +--ro ingress-channel-slice-attenuation?   int32
             |  +--ro egress-channel-slice-attenuation-configured
             |  |  +--ro egress-channel-slice*
             |  |     +--ro egress-channel-slice?               uint32
             |  |     +--ro egress-channel-slice-attenuation?   int32
             |  +--ro channel-attenuation-info
             |  |  +--ro total-channel-attenuation-slice-count?        uint32
             |  |  +--ro channel-attenuation-slice-spacing?            uint32
             |  |  +--ro channel-attenuation-first-slice-wavelength?   uint32
             |  |  +--ro channel-attenuation-first-slice-frequency?    uint32
             |  |  +--ro ingress-channel-attenuation-info*
             |  |  |  +--ro slice-num?             uint32
             |  |  |  +--ro ingress-attenuation?   uint32
             |  |  +--ro egress-channel-attenuation-info*
             |  |     +--ro slice-num?            uint32
             |  |     +--ro egress-attenuation?   uint32
             |  +--ro otdr-info-rx
             |  |  +--ro scan-status?       Otdr-scan-status
             |  |  +--ro tracepoint-file?   string
             |  |  +--ro total-events?      uint32
             |  |  +--ro scan-timestamp?    string
             |  |  +--ro event-info*
             |  |     +--ro event-number?     uint32
             |  |     +--ro detected-event?   uint32
             |  |     +--ro location?         int64
             |  |     +--ro accuracy?         int64
             |  |     +--ro magnitude?        int64
             |  |     +--ro attenuation?      int64
             |  +--ro otdr-info-tx
             |  |  +--ro scan-status?       Otdr-scan-status
             |  |  +--ro tracepoint-file?   string
             |  |  +--ro total-events?      uint32
```

```
|  |  +--ro scan-timestamp?    string
|  |  +--ro event-info*
|  |     +--ro event-number?     uint32
|  |     +--ro detected-event?   uint32
|  |     +--ro location?         int64
|  |     +--ro accuracy?         int64
|  |     +--ro magnitude?        int64
|  |     +--ro attenuation?      int64
|  +--ro rx-los-p
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro rx-loc
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro tx-power-fail-low
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro ingress-auto-laser-shut
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro ingress-auto-pow-red
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro ingress-ampli-gain-low
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro ingress-ampli-gain-high
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro egress-auto-laser-shut
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro egress-auto-pow-red
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro egress-ampli-gain-low
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro egress-ampli-gain-high
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro high-tx-br-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro high-rx-br-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro span-too-short-tx
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro span-too-short-rx
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman-auto-pow-red
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman1-low-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman2-low-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman3-low-pwr
|  |  +--ro is-detected?   boolean
```

```
|  |  +--ro counter?       uint32
|  +--ro raman4-low-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman5-low-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman1-high-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman2-high-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman3-high-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman4-high-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro raman5-high-pwr
|  |  +--ro is-detected?   boolean
|  |  +--ro counter?       uint32
|  +--ro ots-och-alamr-info
|  |  +--ro rx-los-p
|  |  |  +--ro is-detected?   boolean
|  |  |  +--ro counter?       uint32
|  |  +--ro tx-power-fail-low
|  |     +--ro is-detected?   boolean
|  |     +--ro counter?       uint32
|  +--ro ots-tone-info
|  |  +--ro tone-freq?          string
|  |  +--ro tone-rate?          uint32
|  |  +--ro pattern?            string
|  |  +--ro pattern-expected?   string
|  |  +--ro dectected-oob?      uint32
|  |  +--ro state?              Conn-verfcn-state
|  |  +--ro pattern-received?   string
|  +--ro transport-admin-state?                       Ots-tas
|  +--ro rx-pow-low-threshold?                        int32
|  +--ro rx-pow-high-threshold?                       int32
|  +--ro tx-pow-low-threshold?                        int32
|  +--ro tx-pow-high-threshold?                       int32
|  +--ro pm-enable?                                   uint32
|  +--ro controller-state?                            Ots-controller-state
|  +--ro rx-voa-attenuation?                          int32
|  +--ro tx-voa-attenuation?                          int32
|  +--ro channel-width?                               uint32
|  +--ro central-frequncy?                            uint32
|  +--ro add-drop-channel?                            string
|  +--ro line-channel?                                string
|  +--ro ingress-ampli-gain?                          int32
|  +--ro ingress-ampli-tilt?                          int32
|  +--ro ingress-amp-gain-deg-thres-low?              uint32
|  +--ro ingress-amp-gain-deg-thres-high?             uint32
|  +--ro ingress-ampli-gain-range?                    Ots-amplifier-gain-range

|  +--ro egress-ampli-gain?                           int32
|  +--ro egress-ampli-tilt?                           int32
|  +--ro egress-amp-gain-deg-thres-low?               uint32
|  +--ro egress-amp-gain-deg-thres-high?              uint32
|  +--ro egress-ampli-gain-range?                     Ots-amplifier-gain-range

|  +--ro composite-raman-power?                       uint32
|  +--ro wavelength?                                  uint32
```

```
             |  +--ro transmit-power?                                int32
             |  +--ro receive-power?                                 int32
             |  +--ro total-cl-tx-power?                             int32
             |  +--ro total-cl-rx-power?                             int32
             |  +--ro receive-signal-power?                          int32
             |  +--ro transmit-signal-power?                         int32
             |  +--ro ingress-ampli-osri?                            boolean
             |  +--ro egress-ampli-osri?                             boolean
             |  +--ro ingress-ampli-force-apr?                       boolean
             |  +--ro egress-ampli-force-apr?                        boolean
             |  +--ro ingress-ampli-safety-control-mode?
Ots-amplifier-safety-control-mode
             |  +--ro egress-ampli-safety-control-mode?
Ots-amplifier-safety-control-mode
             |  +--ro ingress-ampli-safety-control-mode-configured?
Ots-amplifier-safety-control-mode
             |  +--ro egress-ampli-safety-control-mode-configured?
Ots-amplifier-safety-control-mode
             |  +--ro ingress-ampli-osri-configured?                 boolean
             |  +--ro egress-ampli-osri-configured?                  boolean
             |  +--ro ingress-ampli-force-apr-configured?            boolean
             |  +--ro egress-ampli-force-apr-configured?             boolean
             |  +--ro raman-safety-control-mode?
Ots-amplifier-safety-control-mode
             |  +--ro raman-safety-control-mode-configured?
Ots-amplifier-safety-control-mode
             |  +--ro raman-osri?                                    boolean
             |  +--ro raman-force-apr?                               boolean
             |  +--ro raman-osri-configured?                         boolean
             |  +--ro raman-force-apr-configured?                    boolean
             |  +--ro rx-pow-low-warning-threshold?                  int32
             |  +--ro rx-pow-high-warning-threshold?                 int32
             |  +--ro tx-pow-low-warning-threshold?                  int32
             |  +--ro tx-pow-high-warning-threshold?                 int32
             |  +--ro description?                                   string
             |  +--ro channel-attenuation?                           int32
             |  +--ro rx-voa-attenuation-config-val?                 int32
             |  +--ro tx-voa-attenuation-config-val?                 int32
             |  +--ro ampli-control-mode-config-val?
Ots-amplifier-control-mode
             |  +--ro rx-low-th-psd-config-val?                      int32
             |  +--ro total-rx-power?                                int32
             |  +--ro total-tx-power?                                int32
           | +--ro ingress-ampli-gain-range-config-val?          Ots-amplifier-gain-range

             |  +--ro ingress-ampli-gain-config?                     uint32
             |  +--ro ingress-ampli-tilt-config?                     int32
             |  +--ro ingress-ampli-thr-deg-low-config?              uint32
             |  +--ro ingress-ampli-thr-deg-high-config?             uint32
           | +--ro egress-ampli-gain-range-config-val?           Ots-amplifier-gain-range

             |  +--ro egress-ampli-gain-config?                      uint32
             |  +--ro egress-ampli-tilt-config?                      int32
             |  +--ro egress-ampli-gain-thr-deg-low-config?          uint32
             |  +--ro egress-ampli-gain-thr-deg-high-config?         uint32
             |  +--ro channel-attenuation-configured?                int32
             |  +--ro br-power?                                      int32
             |  +--ro raman-br-power?                                int32
             |  +--ro led-state?                                     Led-state
             +--ro ots-spectrum-info
             |  +--ro spectrum-info
             |     +--ro total-spectrum-slice-count?   uint32
             |     +--ro spectrum-slice-spacing?       uint32
             |     +--ro first-slice-wavelength?       uint32
```

```
|     +--ro first-slice-frequency?        uint32
|     +--ro spectrum-slice-power-info*
|        +--ro slice-num?   uint32
|        +--ro rx-power?    int16
|        +--ro tx-power?    int16
+--ro name                 xr:Interface-name
```

The following is a sample tree structure of Cisco-IOS-XR-controller-ots-cfg model.

```
augment /a1:interface-configurations/a1:interface-configuration:
    +--rw ots
       +--rw ingress-channel-slice-attns
       |  +--rw ingress-channel-slice-attn* [ingress-channel-slice-attn]
       |     +--rw ingress-channel-slice-attn         uint32
       |     +--rw ingress-channel-slice-attnvalue    uint32
       +--rw raman-tx-power-disables
       |  +--rw raman-tx-power-disable* [raman-tx-power-disable-instance]
       |     +--rw raman-tx-power-disable-instance    uint32
       +--rw raman-tx-powers
       |  +--rw raman-tx-power* [raman-tx-power-instance]
       |     +--rw raman-tx-power-instance    uint32
       |     +--rw raman-tx-power-value       uint32
       +--rw ots-otdr
       |  +--rw ots-otdr-rx
       |  |  +--rw ots-otdr-rx-expert
       |  |  |  +--rw ots-otdr-rx-capture-start?   uint32
       |  |  |  +--rw ots-otdr-rx-scan-duration?   uint32
       |  |  |  +--rw ots-otdr-rx-pulse-width?     uint32
       |  |  |  +--rw ots-otdr-rx-capture-end?     uint32
       |  |  +--rw ots-otdr-rx-auto
       |  |  |  +--rw ots-otdr-rx-excess-reflection-threshold?   int32
       |  |  |  +--rw ots-otdr-rx-splice-loss-threshold?         uint32
       |  |  |  +--rw ots-otdr-rx-raman-setpoint?                uint32
       |  |  |  +--rw ots-otdr-rx-reflectance-threshold?         int32
       |  |  +--rw ots-otdr-rx-back-scattering?    int32
       |  |  +--rw ots-otdr-rx-refractive-index?   uint32
       |  +--rw ots-otdr-scan-mode
       |  |  +--rw ots-otdr-scan-mode-expert?   empty
       |  +--rw ots-otdr-tx
       |     +--rw ots-otdr-tx-expert
       |     |  +--rw ots-otdr-tx-capture-end?     uint32
       |     |  +--rw ots-otdr-tx-scan-duration?   uint32
       |     |  +--rw ots-otdr-tx-capture-start?   uint32
       |     |  +--rw ots-otdr-tx-pulse-width?     uint32
       |     +--rw ots-otdr-tx-auto
       |     |  +--rw ots-otdr-tx-splice-loss-threshold?         uint32
       |     |  +--rw ots-otdr-tx-excess-reflection-threshold?   int32
       |     |  +--rw ots-otdr-tx-raman-setpoint?                uint32
       |     |  +--rw ots-otdr-tx-reflectance-threshold?         int32
       |     +--rw ots-otdr-tx-refractive-index?   uint32
       |     +--rw ots-otdr-tx-back-scattering?    int32
       +--rw egress-channel-slice-attns
       |  +--rw egress-channel-slice-attn* [egress-channel-slice-attn]
       |     +--rw egress-channel-slice-attn         uint32
       |     +--rw egress-channel-slice-attnvalue    uint32
       +--rw ots-egress-safety-control-mode?                      Ots-safety-control-mode
       +--rw ots-ingress-amplifier-gain?                          uint32
       +--rw ots-tone-pattern-expected?                           string
       +--rw ots-ingress-osri?                                    boolean
       +--rw ots-ingress-amplifier-gain-degrade-high-threshold?   uint32
       +--rw ots-tx-voa-attenuation?                              uint32
       +--rw ots-ingress-safety-control-mode?                     Ots-safety-control-mode
       +--rw ots-tone-detect-oob?                                 empty
       +--rw ots-ingress-force-apr?                               boolean
       +--rw ots-raman-force-apr?                                 boolean
```

```
        +--rw ots-egress-amplifier-gain-degrade-low-threshold?     uint32
        +--rw ots-ingress-amplifier-gain-degrade-low-threshold?    uint32
        +--rw ots-egress-amplifier-tilt?                           int32
        +--rw ots-raman-safety-control-mode?                       Ots-safety-control-mode
        +--rw ots-tone-frequency?                                  string
        +--rw ots-egress-amplifier-gain?                           uint32
        +--rw ots-tone-pattern?                                    string
        +--rw ots-egress-amplifier-gain-degrade-high-threshold?    uint32
        +--rw ots-raman-osri?                                      boolean
        +--rw ots-egress-osri?                                     boolean
        +--rw ots-egress-amplifier-gain-range?
Ots-ingress-egress-ampli-gain-range
        +--rw ots-ingress-amplifier-gain-range?
Ots-ingress-egress-ampli-gain-range
        +--rw ots-ingress-amplifier-tilt?                          int32
        +--rw ots-tone-rate?                                       uint32
        +--rw ots-egress-force-apr?                                boolean
```

Configure Flex Grid in OLT

Step 1 Use the Cisco-IOS-XR-osa-linesystem-cfg.yang Yang model to configure flex grid channel in the OLT card.

Yang Model	Example
Cisco-IOS-XR-osa-linesystem-cfg.yang	`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <edit-config> <target> <candidate/> </target> <config> <active-nodes xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-config-mda-cfg"> <active-node> <node-name>0/0/NXR0</node-name> <terminal-amplifier xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-linesystem-cfg"> <olt-grid-mode> <olt-channel-identifiers> <olt-channel-identifier> <channel-number>1</channel-number> <centre-frequency>191.375</centre-frequency> <channel-width>75</channel-width> </olt-channel-identifier> </olt-channel-identifiers> </olt-grid-mode> </terminal-amplifier> </active-node> </active-nodes> </config> </edit-config> </rpc>`

Step 2 Use the Cisco-IOS-XR-osa-hwmod-linesys-oper.yang Yang model to get the operational data of the flex grid channel configured on the OLT card.

Yang Model	Example
Cisco-IOS-XR-osa-hwmod-linesys-oper.yang	<?xml version="1.0" ?> <rpc-reply message-id="urn:uuid:0b9fd0cf-b58c-4af2-8503-7b20f933145d" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <osa xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-hwmod-linesys-oper"> <node-ids> <node-id> <node-name>0/0/NXR0</node-name> <terminal-ampli> <status>slice-config-complete</status> <flexi-grid-info> <channel-number>1</channel-number> <centre-frequency-thz>191.375000</centre-frequency-thz> <channel-width-ghz>75.000</channel-width-ghz> </flexi-grid-info> </terminal-ampli> </node-id> </node-ids> </osa> </data> </rpc-reply>

Configure Flex Grid in ILA card

Step 1 Use the Cisco-IOS-XR-osa-linesystem-cfg.yang Yang model to configure the flex grid channel in the ILA card.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-osa-linesystem-cfg.yang</td><td>

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <active-nodes
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-config-mda-cfg">
        <active-node>
          <node-name>0/0/NXR0</node-name>
          <inline-amplifier
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-linesystem-cfg">
            <ila-grid-mode>
              <ila-channel-identifiers>
                <ila-channel-identifier>
                  <channel-number>1</channel-number>
                  <centre-frequency>191.375</centre-frequency>
                  <channel-width>75</channel-width>
                </ila-channel-identifier>
              </ila-channel-identifiers>
            </ila-grid-mode>
          </inline-amplifier>
        </active-node>
      </active-nodes>
    </config>
  </edit-config>
</rpc>
```

</td></tr>
</table>

Step 2 Use the Cisco-IOS-XR-osa-hwmod-linesys-oper.yang Yang model to get the operational data for the flex grid channel configured on the ILA card.

Yang Model	Example
Cisco-IOS-XR-osa-hwmod-linesys-oper.yang	`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">` `  <edit-config>` `    <target>` `      <candidate/>` `    </target>` `    <config>` `      <active-nodes xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-config-mda-cfg">` `        <active-node>` `          <node-name>0/0/NXR0</node-name>` `          <inline-amplifier xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-linesystem-cfg">` `            <ila-grid-mode>` `              <ila-channel-identifiers>` `                <ila-channel-identifier>` `                  <channel-number>1</channel-number>` `                  <centre-frequency>191.375</centre-frequency>` `                  <channel-width>75</channel-width>` `                </ila-channel-identifier>` `              </ila-channel-identifiers>` `            </ila-grid-mode>` `          </inline-amplifier>` `        </active-node>` `      </active-nodes>` `    </config>` `  </edit-config>` `</rpc>`

Configure OTS Controller

Step 1 Use the Cisco-IOS-XR-controller-ots-cfg.yang Yang model to configure the OTS controller.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-controller-ots-cfg.yang</td><td><pre>
<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>Ots0/0/0/0</interface-name>
          <ots
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-ots-cfg">

<ots-egress-safety-control-mode>auto</ots-egress-safety-control-mode>

<ots-ingress-amplifier-gain>160</ots-ingress-amplifier-gain>
            <ots-ingress-osri>true</ots-ingress-osri>
            <ots-tx-voa-attenuation>200</ots-tx-voa-attenuation>
            <ots-ingress-force-apr>false</ots-ingress-force-apr>

<ots-egress-amplifier-tilt>-40</ots-egress-amplifier-tilt>

<ots-egress-amplifier-gain>180</ots-egress-amplifier-gain>
            <ots-egress-osri>false</ots-egress-osri>

<ots-ingress-amplifier-gain-range>normal</ots-ingress-amplifier-gain-range>

<ots-ingress-amplifier-tilt>50</ots-ingress-amplifier-tilt>
            <ots-egress-force-apr>true</ots-egress-force-apr>
          </ots>
        </interface-configuration>
      </interface-configurations>
    </config>
  </edit-config>
</rpc>
</pre></td></tr>
</table>

Step 2 Use the Cisco-IOS-XR-controller-ots-oper.yang Yang model to view the parameters of the OTS controller.

Note In the current release, all the controller models are mapped to the OTS controller model. Hence the operational data of all the controllers display "ots-state-up" as the controller state, and "ots-tas-ui-is" as transport-admin-sate, irrespective of the functionality.

Yang Model	Example
Cisco-IOS-XR-controller-ots-oper.yang	

Yang Model	Example

```
<?xml version="1.0" ?>
<rpc-reply
message-id="urn:uuid:1ecef265-e94d-4b42-ad53-adb137a58efc"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <data>
       <ots-oper
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-ots-oper">

          <ots-ports>
              <ots-port>
                <name>Ots0/0/0/0</name>
                   <ots-info>

<transport-admin-state>ots-tas-ui-is</transport-admin-state>

<controller-state>ots-state-up</controller-state>
                 <tx-voa-attenuation>200</tx-voa-attenuation>

<ingress-ampli-gain>160</ingress-ampli-gain>

<ingress-ampli-tilt>50</ingress-ampli-tilt>

<ingress-ampli-gain-range>ots-amplifier-gain-range-normal</ingress-ampli-gain-range>

                    <egress-ampli-gain>180</egress-ampli-gain>
                       <egress-ampli-tilt>-40</egress-ampli-tilt>
                     <total-cl-tx-power>2000</total-cl-tx-power>
                      <total-cl-rx-power>-1000</total-cl-rx-power>


<receive-signal-power>2000</receive-signal-power>

<transmit-signal-power>2000</transmit-signal-power>
                    <ingress-ampli-osri>true</ingress-ampli-osri>
                      <egress-ampli-osri>false</egress-ampli-osri>

             .................................................
             .................................................
             ..................................................
<tx-power>-105</tx-power>
                             </spectrum-slice-power-info>
                          <spectrum-slice-power-info>
                              <slice-num>1546</slice-num>
                             <rx-power>-105</rx-power>
                               <tx-power>-105</tx-power>
                              </spectrum-slice-power-info>
                            <spectrum-slice-power-info>
                               <slice-num>1547</slice-num>
                                    <rx-power>-105</rx-power>
                                <tx-power>-105</tx-power>
                      </spectrum-slice-power-info>
                          <spectrum-slice-power-info>
                           <slice-num>1548</slice-num>
                               <rx-power>-105</rx-power>
                        <tx-power>-105</tx-power>
              </spectrum-slice-power-info>
            </spectrum-info>
          </ots-spectrum-info>
        </ots-port>
      </ots-ports>
    </ots-oper>
```

Yang Model	Example
	`  </data>` `</rpc-reply>`

Configure OCH Controller

Step 1 Use the Cisco-IOS-XR-controller-och-cfg.yang Yang model to configure the OCH controller.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-controller-och-cfg.yang</td><td>

```
<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>Och0/3/0/31</interface-name>
          <och
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-och-cfg">

<och-tone-pattern-expected>1234abcd</och-tone-pattern-expected>
            <och-tone-rate>20</och-tone-rate>
          </och>
        </interface-configuration>
      </interface-configurations>
    </config>
  </edit-config>
</rpc>
```

</td></tr>
</table>

Step 2 Use Cisco-IOS-XR-controller-och-oper.yang Yang model to view the OCH controller parameters.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-controller-och-oper.yang</td><td><pre>
<?xml version="1.0" ?>
<rpc-reply
message-id="urn:uuid:50be1a71-e729-442d-aec7-14f486cd6028"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <och-oper
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-och-oper">

      <och-ports>
        <och-port>
          <name>Och0/3/0/31</name>
          <och-info>
            <rx-power>0</rx-power>
            <tx-power>-5000</tx-power>
            <channel-frequency>191375</channel-frequency>
            <channel-width>1500</channel-width>
            <channel-wavelength>156652</channel-wavelength>
            <controller-state>ots-state-up</controller-state>

            <led-state>off</led-state>
            <rx-los-p>
              <is-detected>false</is-detected>
              <counter>0</counter>
            </rx-los-p>
            <tx-power-fail-low>
              <is-detected>false</is-detected>
              <counter>0</counter>
            </tx-power-fail-low>
            <och-tone-info>
              <tone-rate>20</tone-rate>
              <pattern-expected>1234abcd</pattern-expected>

              <dectected-oob>0</dectected-oob>
              <state>conn-vrfcn-state-not-running</state>
            </och-tone-info>

<transport-admin-state>ots-tas-ui-is</transport-admin-state>
          </och-info>
        </och-port>
      </och-ports>
    </och-oper>
  </data>
  </rpc-reply>
</pre></td></tr>
</table>

Configure Optical Cross-Connect

Step 1 Use the Cisco-IOS-XR-Ots-Och-cfg.yang Yang model to configure an optical cross-connect (OTS-OCH controller).

Yang Model	Example
Cisco-IOS-XR-Ots-Och-cfg.yang	`<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>Ots-Och0/0/0/0/1</interface-name>` `          <ots-och xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-Ots-Och-cfg">` `            <add-drop-channel>Ots-Och0/0/0/2/1</add-drop-channel>` `          </ots-och>` `        </interface-configuration>` `      </interface-configurations>` `    </config>` `  </edit-config>`

Step 2 Use the Cisco-IOS-XR-controller-ots-och-oper.yang Yang model to view the parameters of the OTS-OCH controller.

Yang Model	Example
Cisco-IOS-XR-controller-ots-och-oper.yang	`<?xml version="1.0" ?>` `<rpc-reply message-id="urn:uuid:71601b7f-caee-4e65-9627-b5043e66436d"` ` xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">` `    <data>` `       <ots-och-oper xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-ots-och-oper">` `      <ots-och-ports>` `        <ots-och-port>` `          <name>Ots-Och0/0/0/0/1</name>` `          <ots-och-info>` `<transport-admin-state>ots-tas-ui-is</transport-admin-state>` `            <controller-state>ots-state-up</controller-state>` `            <add-drop-channel>Ots-Och0/0/0/2/1</add-drop-channel>` `            <total-rx-power>-1050</total-rx-power>` `            <total-tx-power>-1050</total-tx-power>` `          </ots-och-info>` `        </ots-och-port>` `      </ots-och-ports>` `    </ots-och-oper>` `  </data>` `</rpc-reply>`

Configure OMS Controller

Step 1 Use the Cisco-IOS-XR-controller-oms-cfg.yang Yang model to configure the OMS controller.

Yang Model	Example
Cisco-IOS-XR-controller-oms-cfg.yang	`<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>Oms0/3/0/32</interface-name>` `          <oms xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-oms-cfg">` `            <oms-tone-rate>20</oms-tone-rate>` `<oms-tone-pattern-expected>abcd1234</oms-tone-pattern-expected>` `            <oms-tone-detect-oob/>` `          </oms>` `        </interface-configuration>` `      </interface-configurations>` `    </config>` `  </edit-config>` `</rpc>`

Step 2 Use the Cisco-IOS-XR-controller-oms-oper.yang Yang model to view the parameters of the OMS controller.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-controller-oms-oper</td><td>


```
<?xml version="1.0" ?>
<rpc-reply
message-id="urn:uuid:ba7b0faf-3762-4a8e-b9fe-e8d190a2dbe7"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
    <oms-oper
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-oms-oper">

      <oms-ports>
        <oms-port>
          <name>Oms0/3/0/32</name>
          <oms-info>
            <rx-power>0</rx-power>
            <tx-power>0</tx-power>
            <controller-state>ots-state-up</controller-state>

            <led-state>off</led-state>
            <rx-los-p>
              <is-detected>false</is-detected>
              <counter>0</counter>
            </rx-los-p>
            <tx-power-fail-low>
              <is-detected>false</is-detected>
              <counter>0</counter>
            </tx-power-fail-low>
                    <oms-tone-info>
                            <tone-rate>20</tone-rate>
              <pattern-expected>abcd1234</pattern-expected>

              <dectected-oob>1</dectected-oob>
              <state>conn-vrfcn-state-not-running</state>
            </oms-tone-info>

<transport-admin-state>ots-tas-ui-is</transport-admin-state>
          </oms-info>
        </oms-port>
      </oms-ports>
    </oms-oper>
  </data>
 </rpc-reply>
```


</td></tr>
</table>

Configure DFB Controller

Step 1 Use the Cisco-IOS-XR-controller-dfb-cfg.yang Yang model to configure the DFB controller.

Yang Model	Example
Cisco-IOS-XR-controller-dfb-cfg.yang	<code><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>Dfb0/0/0/0</interface-name> <dfb xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-dfb-cfg"> <dfb-tx-voa-attenuation>150</dfb-tx-voa-attenuation> </dfb> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc></code>

Step 2 Use the Cisco-IOS-XR-controller-dfb-oper.yang Yang model to view the DFB controller parameters.

Yang Model	Example
Cisco-IOS-XR-controller-dfb-oper.yang	`<?xml version="1.0" ?>` `<rpc-reply` `message-id="urn:uuid:41205dcf-f92f-4b73-bdf3-ba64438d15ac"` `xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"` `xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">` `  <data>` `    <dfb-oper` `xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-dfb-oper">` `      <dfb-ports>` `        <dfb-port>` `          <name>Dfb0/0/0/0</name>` `          <dfb-info>` `            <laser-state>on</laser-state>` `<controller-state>ots-state-up</controller-state>` `<transport-admin-state>ots-tas-ui-is</transport-admin-state>` `            <total-rx-power>1000</total-rx-power>` `            <total-tx-power>2000</total-tx-power>` `            <tx-voa-attenuation>150</tx-voa-attenuation>` `<tx-voa-attenuation-config-val>150</tx-voa-attenuation-config-val>` `            <rx-los-p>` `              <is-detected>false</is-detected>` `              <counter>0</counter>` `            </rx-los-p>` `            <tx-power-fail-low>` `              <is-detected>false</is-detected>` `              <counter>0</counter>` `            </tx-power-fail-low>` `          </dfb-info>` `        </dfb-port>` `      </dfb-ports>` `    </dfb-oper>` `  </data>` `</rpc-reply>`

Configure OSC Controller

Step 1 Use the Cisco-IOS-XR-controller-osc-cfg.yang Yang model to configure the OSC controller.

Yang Model	Example
Cisco-IOS-XR-controller-osc-cfg.yang	`<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>Osc0/0/0/0</interface-name>` `          <osc xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-osc-cfg">` `            <osc-transmit-power>20</osc-transmit-power>` `            <osc-transmit-shutdown>false</osc-transmit-shutdown>` `          </osc>` `        </interface-configuration>` `      </interface-configurations>` `    </config>` `  </edit-config>` `</rpc>`

Step 2 Use Cisco-IOS-XR-controller-osc-oper.yang Yang model to view the OSC controller parameters.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-controller-osc-oper.yang</td><td><pre>
<?xml version="1.0" ?>
<rpc-reply
message-id="urn:uuid:57794a6c-fe5b-425e-8df7-7c09a789b757"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
      <osc-oper
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-osc-oper">

            <osc-ports>
            <osc-port>
                <name>Osc0/0/0/0</name>
                  <osc-info>
                       <laser-state>off</laser-state>

<controller-state>ots-state-up</controller-state>

<transport-admin-state>ots-tas-ui-is</transport-admin-state>
                            <total-rx-power>-5000</total-rx-power>

                            <total-tx-power>-5000</total-tx-power>

                              <rx-los-p>
                                  <is-detected>false</is-detected>

                                   <counter>0</counter>
                             </rx-los-p>
                           <tx-power-fail-low>
              <is-detected>false</is-detected>
              <counter>0</counter>
            </tx-power-fail-low>
          </osc-info>
        </osc-port>
        </osc-ports>
    </osc-oper>
    </data>
  </rpc-reply>
</pre></td></tr>
</table>

Configure FPD Package

Step 1 Use the Cisco-IOS-XR-fpd-infra-cfg.yang Yang model to configure FPD package.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-fpd-infra-cfg.yang</td><td><pre><rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <fpd
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-fpd-infra-cfg">
        <auto-upgrade>enable</auto-upgrade>
      </fpd>
    </config>
  </edit-config>
</rpc></pre></td></tr>
</table>

Step 2 Use Cisco-IOS-XR-show-fpd-loc-ng-oper.yang Yang model to view the operational data for FPD package details

Yang Model	Example
Cisco-IOS-XR-show-fpd-loc-ng-oper.yang	`<?xml version="1.0" ?>` `<rpc-reply` `message-id="urn:uuid:a69ef4eb-f4c8-461e-8858-4d70169df583"` `xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"` `xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">` `  <data>` `    <show-fpd` `xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-show-fpd-loc-ng-oper">` `      <locations>` `        <location>` `          <location-name>0-RP0-CPU0</location-name>` `          <fpds>` `            <fpd>` `              <fpd-name>fpd_list</fpd-name>` `              <upgrade-status>No upgrade in` `progress</upgrade-status>` `              <fpd-info-detaile>` `                <location>0/RP0/CPU0</location>` `                <card-name>NCS1010-CNTLR-K9</card-name>` `                <fpd-name>ADMConfig</fpd-name>` `                <hw-version>0.1 </hw-version>` `                <secure-boot-attr>   </secure-boot-attr>` `                <status>NEED UPGD</status>` `                <running-version>  7.01 </running-version>` `                <programd-version>  7.01 </programd-version>` `                <reload-location>0/RP0</reload-location>` `              </fpd-info-detaile>` `              <fpd-info-detaile>` `               <location>0/RP0/CPU0</location>` `                <card-name>NCS1010-CNTLR-K9</card-name>` `                <fpd-name>BIOS</fpd-name>` `                 <hw-version>0.1 </hw-version>` `.` `.` `.` `<set-timestamp>1654756033</set-timestamp>` `              <clear-time>-</clear-time>` `              <clear-timestamp>0</clear-timestamp>` `              <description>Ots0/0/0/0 - APC blocked</description>` `            </alarm-info>` `          </active>` `        </brief-system>` `      </brief>` `    </alarms>` `  </data>` `  </rpc-reply>`

View NCS 1010 Platform Details

Step 1 Use the Cisco-IOS-XR-platform-oper.yang Yang model to view the platform details of the NCS 1010 node.

Yang Models	Example
Cisco-IOS-XR-platform-oper.yang	`<?xml version="1.0" ?>` `<rpc-reply message-id="urn:uuid:5323a66f-728c-45a8-a8be-96751fe7081a" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">` `<data>` `<platform xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-platform-oper">` `<racks>` `<rack>` `<rack-name>0</rack-name>` `<slots>` `<slot>` `<slot-name>RP0</slot-name>` `<instances>` `<instance>` `<instance-name>CPU0</instance-name>` `<state>` `<card-type>NCS1010-CNTLR-K9</card-type>` `<card-redundancy-state>active</card-redundancy-state>` `<state>not-applicable</state>` `<admin-state>NSHUT,NMON</admin-state>` `<node-name>0/RP0/CPU0</node-name>` `<oper-state>IOS XR RUN</oper-state>` `</state>` `</instance>` `</instances>` `</slot>` `<slot>` `.` `.` `.` `<card-type>NCS1K-MD-32E-C</card-type>` `<card-redundancy-state>red-state-none</card-redundancy-state>` `<state>not-applicable</state>` `<admin-state>NSHUT,NMON</admin-state>` `<node-name>0/3</node-name>` `<oper-state>OPERATIONAL</oper-state>` `</state>` `</slot>` `</slots>` `</rack>` `</racks>` `</platform>` `</data>` `</rpc-reply>`

Step 2

Step 3

What to do next

Configure Optical Applications

Step 1 Use the Cisco-IOS-XR-olc-cfg.yang Yang model to configure the optical applications.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-olc-cfg.yang</td><td>

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="101">
  <edit-config>
    <target>
      <candidate/>
    </target>
    <config>
      <olc xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-olc-cfg">
        <controller-rsips>
          <controller-rsip>
            <controller>Ots0/0/0/0</controller>
            <apc>
              <psds>
                <psd>
                  <psd-index>1</psd-index>
                  <psd-value>-110</psd-value>
                </psd>
              </psds>
              <apc-cfg-state>disable</apc-cfg-state>
            </apc>
            <raman>
              <raman-tune-target-gain>100</raman-tune-target-gain>

            </raman>
            <span-loss>
              <max-threshold>110</max-threshold>
              <min-threshold>100</min-threshold>
            </span-loss>
          </controller-rsip>
        </controller-rsips>
      </olc>
    </config>
  </edit-config>
</rpc>
```

</td></tr>
</table>

Step 2 Use Cisco-IOS-XR-olc-oper.yang Yang model to view the operational data for the optical applications.

Yang Model	Example
Cisco-IOS-XR-olc-oper.yang	` <rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get> <filter> <olc xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-olc-oper"/> </filter> </get> </rpc> #####Response###### <?xml version="1.0"?> <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <olc xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-olc-oper"> <raman-tuning-ctrlr-tables> <raman-tuning-ctrlr-table> <name>Ots0/0/0/0</name> <status>disabled</status> <tuning-complete-timestamp>N/A</tuning-complete-timestamp> <target-gain>N/A</target-gain> <max-gain>N/A</max-gain> <tuning-complete-gain>N/A</tuning-complete-gain> </raman-tuning-ctrlr-table> </raman-tuning-ctrlr-tables> <apc-status-ctrlr-tables> <apc-status-ctrlr-table> <name>Ots0/0/0/0</name> <status>disabled</status> </apc-status-ctrlr-table> . . . <span-loss-ctrlr-tables> <span-loss-ctrlr-table> <name>Ots0/0/0/0</name> <neighbour-rid>10.1.1.2</neighbour-rid> <apparent-rx-span-loss>-10.0 dB</apparent-rx-span-loss> <apparent-tx-span-loss>20.0 dB</apparent-tx-span-loss> <dfb-rx-span-loss-with-pumps-off>-1310.7 dB</dfb-rx-span-loss-with-pumps-off> <ts-dfb-rx-span-loss-with-pumps-off>2022-06-17 13:11:16</ts-dfb-rx-span-loss-with-pumps-off> <dfb-tx-span-loss-with-pumps-off>0.0 dB</dfb-tx-span-loss-with-pumps-off> <ts-dfb-tx-span-loss-with-pumps-off>2022-06-17 13:11:28</ts-dfb-tx-span-loss-with-pumps-off> <est-rx-span-loss>NA</est-rx-span-loss> <est-tx-span-loss>NA</est-tx-span-loss> </span-loss-ctrlr-table> </span-loss-ctrlr-tables> </olc> </data>`

View Performance Monitoring Parameters

Use Cisco-IOS-XR-pmengine-oper.yang Yang model to view the performance monitoring parameters on the controllers.

Yang Model	Example
Cisco-IOS-XR-pmengine-oper.yang	

Yang Model	Example
	(see code below)

```
rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="101">
  <get>
    <filter>
      <performance-management
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-pmengine-oper">
        <optics>
          <optics-ports>
            <optics-port>
              <name>Ots0/0/0/0</name>
              <optics-current>
                <optics-second30>
                  <optics-second30-optics/>
                </optics-second30>
              </optics-current>
            </optics-port>
          </optics-ports>
        </optics>
      </performance-management>
    </filter>
  </get>
</rpc>
#####Response######
<?xml version="1.0"?>
<rpc-reply message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <data>
  <performance-management
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-pmengine-oper">
   <optics>
    <optics-ports>
     <optics-port>
      <name>Ots0/0/0/0</name>
      <optics-current>
       <optics-second30>
        <optics-second30-optics>
         <optics-second30-optic>
          <number>1</number>
          <index>0</index>
          <valid>true</valid>
         <timestamp>10:00:00 - 10:00:15 Thu Jun 16 2022</timestamp>

          <last-clear30-sec-time>never</last-clear30-sec-time>
          <sec30-support>true</sec30-support>
          <flex-bin-support>true</flex-bin-support>
          <flex-bin-interval>10</flex-bin-interval>
          <opt>
           <valid>true</valid>
           <minimum>20.00</minimum>
           <average>20.00</average>
           <maximum>20.00</maximum>
           <minimum-threshold>-20.00</minimum-threshold>
           <configured-min-thresh>NA</configured-min-thresh>
           <minimum-tca-report>false</minimum-tca-report>
           .
           .
           .
           <valid>true</valid>
           <minimum>0.00</minimum>
           <average>0.00</average>
           <maximum>0.00</maximum>
```

Yang Model	Example
	<minimum-threshold>-5.00</minimum-threshold> <configured-min-thresh>NA</configured-min-thresh> <minimum-tca-report>false</minimum-tca-report> <maximum-threshold>5.00</maximum-threshold> <configured-max-thresh>NA</configured-max-thresh> <maximum-tca-report>false</maximum-tca-report> <min-time>1655373600749422903</min-time> <max-time>1655373600749422903</max-time> </eatl> <iagn> <maximum-tca-report>false</maximum-tca-report> <min-time>1655373600749422903</min-time> <max-time>1655373600749422903</max-time> -thresh> <minimum>40.00</minimum> <average>40.00</average> eport> <maximum-threshold>190.00</maximum-threshold> <configured-max-thresh>NA</configured-max-thresh> <maximum-tca-report>false</maximum-tca-report> <min-time>1655373600749422903</min-time> <max-time>1655373600749422903</max-time> </raman-5> </optics-second30-optic> </optics-second30-optics> </optics-second30> </optics-current> </optics-port> </optics-ports> </optics> </performance-management> </data> </rpc-reply>

Configure Equipment Mismatch Alarm

Use the Cisco-IOS-XR-osa-ct-cfg.yang Yang model to configure the equipment mismatch alarm. For example, when the NCS 1010 node is loaded with the OLT- C card and if you try to configure the node with a different line card configuration, the equipment mismatch alarm rises.

Yang Model	Example
Cisco-IOS-XR-osa-ct-cfg.yang	`<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">` `  <edit-config>` `    <target>` `      <candidate/>` `    </target>` `    <config>` `      <resrv-cli xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-ct-cfg">` `        <slot-info-cli>` `          <lc-slot>0_0_NXR0</lc-slot>` `          <card-type-cli>ncs1k-olt-r-c</card-type-cli>` `        </slot-info-cli>` `      </resrv-cli>` `    </config>` `  </edit-config>` `</rpc>`

View the List of Alarms on the NCS 1010 Node

Use the Cisco-IOS-XR-alarmgr-server-oper.yang Yang model to view the list of alarms generated on the NCS 1010 node.

<table>
<tr><th>Yang Model</th><th>Example</th></tr>
<tr><td>Cisco-IOS-XR-alarmgr-server-oper.yang</td><td>


```
<?xml version="1.0" ?>
<rpc-reply
message-id="urn:uuid:518e2c10-c837-4b36-9bab-93f935148ce5"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
    <alarms
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-alarmgr-server-oper">

      <brief>
        <brief-system>
          <active>
            <alarm-info>
              <location>0/Rack</location>
              <severity>major</severity>
              <group>fpd-infra</group>
              <set-time>06/09/2022 06:26:48 UTC</set-time>
              <set-timestamp>1654756008</set-timestamp>
              <clear-time>-</clear-time>
              <clear-timestamp>0</clear-timestamp>
              <description>One Or More FPDs Need Upgrade Or Not In
 Current State</description>
            </alarm-info>
            <alarm-info>
              <location>0/RP0/CPU0</location>
              <severity>major</severity>
              <group>fpd-infra</group>
              <set-time>06/09/2022 06:26:49 UTC</set-time>
              <set-timestamp>1654756009</set-timestamp>
              <clear-time>-</clear-time>
              <clear-timestamp>0</clear-timestamp>
              <description>One Or More FPDs Need Upgrade Or Not In
 Current State</description>
            </alarm-info>
            <alarm-info>
              <location>0/0/NXR0</location>
              <severity>major</severity>
              <group>fpd-infra</group>
              <set-time>06/09/2022 06:26:51 UTC</set-time>
              <set-timestamp>1654756011</set-timestamp>
              <clear-time>-</clear-time>
              <clear-timestamp>0</clear-timestamp>
              <description>One Or More FPDs Need Upgrade Or Not In
 Current State</description>
            </alarm-info>
            <alarm-info>
              <location>0/0/NXR0</location>
              <severity>minor</severity>
              <group>software</group>
              <set-time>06/09/2022 06:27:13 UTC</set-time>
              <set-timestamp>1654756033</set-timestamp>
              <clear-time>-</clear-time>
              <clear-timestamp>0</clear-timestamp>
              <description>Ots0/0/0/0 - APC blocked</description>

            </alarm-info>
          </active>
        </brief-system>
      </brief>
     </alarms>
   </data>
  </rpc-reply>
```


</td></tr>
</table>

Configure Optical Amplifier on OLT Line Card Using Open Config Model

The openconfig-optical-amplifier Yang model uses the following naming convention for the preamplifier and the booster amplifier in the OLT line card:

R/S-<AMP TYPE><ID>

- *R*—Rack.
- *S*—Slot.
- *<AMP TYPE>*—AMP-PRE (for preamplifier) or AMP-BST (for booster amplifier).
- *ID*—The value is 0 in openconfig.

For example, the amplifiers are mentioned as 0/0-AMP-PRE0 or 0/0-AMP-BST0 which is a line port ots0/0/0/0 in the IOS-XR.

Step 1 Use the openconfig-optical-amplifier Yang model to configure the amplifier on the OLT line card.

Openconfig Model	Example
openconfig-optical-amplifier	{ "openconfig-optical-amplifier:optical-amplifier": { "amplifiers": { "amplifier": [{ "name": "0/0-AMP-PRE0", "config": { "name": "0/0-AMP-PRE0", "target-gain": "19.00", "gain-range": "MID_GAIN_RANGE", "target-gain-tilt": "3.90", "enabled": true } }, { "name": "0/0-AMP-BST0", "config": { "name": "0/0-AMP-BST0", "target-gain": "19.00", "target-gain-tilt": "-1.5", "enabled": true } }] } } }

Step 2 Get the operational data using GNMI.

```
{
  "openconfig-optical-amplifier": {
    "optical-amplifier": {
      "amplifiers": {
        "amplifier": {
          "0/0-AMP-BST0": {
            "state": {
              "enabled": true,
              "name": "0/0-AMP-BST0",
              "target-gain": 19.00,
              "target-gain-tilt": -1.5
            }
          },
          "0/0-AMP-PRE0": {
            "state": {
              "enabled": true,
              "gain-range": "MID_GAIN_RANGE",
              "name": "0/0-AMP-PRE0",
              "target-gain": 19.00,
              "target-gain-tilt": 3.90
            }
          }
        }
      }
    }
  }
}
```


Configure Optical Amplifier on ILA Line Card Using Open Config Model

The openconfig-optical-amplifier Yang model uses the following naming convention for the two booster amplifiers in the ILA line card:

R/S-<AMP TYPE><ID>

- *R*—Rack.
- *S*—Slot.
- *<AMP TYPE>*—AMP-BST for the booster amplifier.
- *ID*—The value is 0 or 2 in openconfig.

For example, the amplifiers are mentioned as 0/0-AMP-BST0 and 0/0-AMP-BST2 which are the line ports ots0/0/0/0 and ots0/0/0/2 respectively in the IOS-XR.

Step 1 Use the openconfig-optical-amplifier Yang model to configure the amplifier on the ILA line card.

Openconfig model	Example
openconfig-optical-amplifier	`{ "openconfig-optical-amplifier:optical-amplifier": { "amplifiers": { "amplifier": [{ "name": "0/0-AMP-BST0", "config": { "name": "0/0-AMP-BST0", "target-gain": "24.00", "target-gain-tilt": "-3.90", "enabled": false, "gain-range": "HIGH_GAIN_RANGE", } }, { "name": "0/0-AMP-BST2", "config": { "name": "0/0-AMP-BST2", "target-gain": "24.00", "target-gain-tilt": "-3.20", "enabled": false, "gain-range": "HIGH_GAIN_RANGE" } }] } } }`

Step 2 Get the operational data using GNMI.

```
{
  "openconfig-optical-amplifier": {
    "optical-amplifier": {
      "amplifiers": {
        "amplifier": {
          "0/0-AMP-BST0": {
            "state": {
              "enabled": false,
              "gain-range": "HIGH_GAIN_RANGE",
              "name": "0/0-AMP-BST0",
              "target-gain": 24.00,
              "target-gain-tilt": -3.90
            }
          },
          "0/0-AMP-BST2": {
            "state": {
              "enabled": false,
              "gain-range": "HIGH_GAIN_RANGE",
              "name": "0/0-AMP-BST2",
              "target-gain": 24.00,
              "target-gain-tilt": -3.20
            }
          }
        }
      }
    }
  }
}
```

Configure Optical Attenuator on OLT Line Card Using Open Config Model

The openconfig-attenuator Yang model uses the following naming convention for the Variable Optical Attenuator (VOA) on the Line-TX of the OLT line card:

R/S-VOA-BST<*ID*>

- *R*—Rack.
- *S*—Slot.
- *ID*—The value is 0 in openconfig.

For example, the VOA is mentioned as 0/0-VOA-BST0 which is a line port ots0/0/0/0 in the IOS-XR.

Step 1 Use the openconfig-attenuator Yang model to configure the attenuator on the OLT line card.

Openconfig Model	Example
openconfig-optical-attenuator	`{` `"openconfig-optical-attenuator:optical-attenuator": {` `"attenuators": {` `"attenuator": [` `{` `"name": "0/0-VOA-BST0",` `"config": {` `"name": "0/0-VOA-BST0",` `"attenuation": "20.00"` `}` `}` `]` `}` `}` `}`

Step 2 Get the operational data using GNMI.

```
{
  "openconfig-optical-attenuator": {
    "optical-attenuator": {
      "attenuators": {
        "attenuator": {
          "0/0-VOA-BST0": {
            "state": {
              "attenuation": 20.00,
              "enabled": true,
              "name": "0/0-VOA-BST0"
            }
          }
        }
      }
    }
  }
```

```
}
```

Configure Optical Attenuator on ILA Line Card Using Open Config Model

The openconfig-attenuator Yang model uses the following naming convention for the VOAs on the Line-1 TX and Line-2 TX of the ILA line card:

R/S-VOA-BST<*ID*>

- *R*—Rack.
- *S*—Slot.
- *ID*—The value is 0 or 2 in openconfig to identify the line port.

For example, the VOAs are mentioned as 0/0-VOA-BST0 and 0/0-VOA-BST2 which are the line ports ots0/0/0/0 and ots0/0/0/2 respectively, in the IOS-XR.

Step 1 Use the openconfig-attenuator Yang model to configure the attenuator on the ILA line card.

Openconfig Model	Example
openconfig-optical-attenuator	`{ "openconfig-optical-attenuator:optical-attenuator": { "attenuators": { "attenuator": [ { "name": "0/0-VOA-BST0", "config": { "name": "0/0-VOA-BST0", "attenuation": "15.00" } }, { "name": "0/0-VOA-BST2", "config": { "name": "0/0-VOA-BST2", "attenuation": "14.00" } } ] } } }`

Step 2 Get the operational data using GNMI.

```
{
  "openconfig-optical-attenuator": {
    "optical-attenuator": {
      "attenuators": {
        "attenuator": {
```

```
          "0/0-VOA-BST0": {
            "state": {
              "attenuation": 15.00,
              "enabled": true,
              "name": "0/0-VOA-BST0"
            }
          },
          "0/0-VOA-BST2": {
            "state": {
              "attenuation": 14.00,
              "enabled": true,
              "name": "0/0-VOA-BST2"
            }
          }
        }
      }
    }
  }
}
```