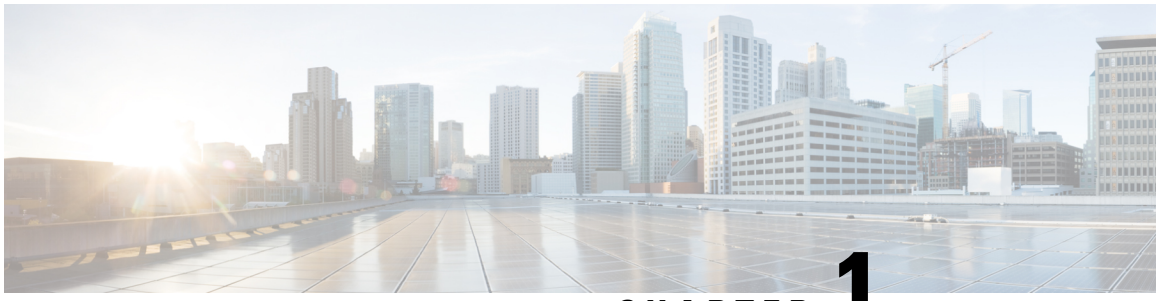
Data Models Configuration Guide for Cisco NCS 1004, Cisco IOS XR Releases

First Published: 2019-11-30

Last Modified: 2024-04-19

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 modeling is standard based approach to model configuration and operational data in networking devices. Using data models, customers can automate and simplify network wide visibility and configuration.

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.

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 key 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

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
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.

Supported YANG Models in NCS 1004

Table 1: Feature History

Feature Name	Release Information	Feature Description
New Unified Model for Enhanced IKEv2 Encryption Support	Cisco IOS XR Release 24.1.1	The new model Cisco-IOS-XR-um-ikev2-cfg introduced in this release enhances the IKEv2 encryption. Now IKEv2 encryption complies with RFC 8784, which describes about using postquantum preshared keys (PPK) for IKEv2 encryption. The PPKs are generated with the help of the Cisco Secure Key Integration Protocol (SKIP) which makes the IKEv2 encryption resilient to quantum attacks.

The supported config and oper YANG models for NCS 1004 are listed below:

Config Yang Models	Oper Yang Models
Cisco-IOS-XR-osa-cfg.yang	Cisco-IOS-XR-osa-oper.yang
Cisco-IOS-XR-controller-optics-cfg.yang	Cisco-IOS-XR-controller-optics-oper.yang
Cisco-IOS-XR-pmengine-cfg.yang	Cisco-IOS-XR-pmengine-oper.yang
Cisco-IOS-XR-ethernet-lldp-cfg.yang	Cisco-IOS-XR-ethernet-lldp-oper.yang
Cisco-IOS-XR-ifmgr-cfg.yang	Cisco-IOS-XR-telemetry-model-driven-oper.yang
Cisco-IOS-XR-telemetry-model-driven-cfg.yang	Cisco-IOS-XR-fpd-infra-oper.yang
Cisco-IOS-XR-fpd-infra-cfg.yang	Cisco-IOS-XR-ikev2-oper.yang
Cisco-IOS-XR-ikev2-cfg.yang	Cisco-IOS-XR-otnsec-oper.yang
Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg	
Cisco-IOS-XR-um-ikev2-cfg	

The supported versions of Open Config model are listed below:

- openconfig-platform.yang
- openconfig-platform-transceiver.yang
- openconfig-terminal-device.yang
- openconfig-interfaces.yang

- openconfig-system.yang

Note openconfig-platform-transceiver.yang model is the augmented model of openconfig-platform.yang model.

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

gNOI for BERT

Table 2: Feature History

Feature Name	Release Information	Description
gNOI for BERT	Cisco IOS XR Release 7.3.1	EMS gNOI supports Bit Error Rate Testing (BERT) operations on NCS 1004 for the following remote procedure calls (RPCs): • StartBERT • StopBERT • GetBERTResults gNOI for BERT is a vendor agnostic open configuration method of enabling and testing network links through the Pseudo Random Binary Sequence (PRBS) feature.

gRPC Network Operations Interface (gNOI) defines a set of gRPC-based microservices for executing operational commands on network devices. Extensible Manageability Services (EMS) gNOI is the Cisco IOS XR implementation of gNOI.

gNOI uses gRPC as the transport protocol and the configuration is same as that of gRPC.

From R7.3.1 onwards, EMS gNOI supports Bit Error Rate Testing (BERT) operations on NCS 1004 for the following remote procedure calls (RPCs):

- StartBERT
- StopBERT
- GetBERTResults

Start a New BERT Session

StartBERT

Starts a new BERT operation for a set of ports. Each BERT operation is uniquely identified by an ID, which is given by the caller. The caller can then use this ID (as well as the list of the ports) either to stop the BERT operation or get the BERT results, or can perform both BERT operations.

```
rpc StartBERT(StartBERTRequest) returns(StartBERTResponse) {}
```

Request and response messages

```
message StartBERTRequest {
  // Per port BERT start requests.
  message PerPortRequest {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;  // required
    // The selected PRBS generating polynomial for BERT.
    PrbsPolynomial prbs_polynomial = 2;  // required
    // BERT duration in seconds. Must be a positive number.
    uint32 test_duration_in_secs = 3;  // required
  }
  // Unique BERT operation ID specified by the client. Multiple BERTs run on
  // different ports can have the same BERT operation ID. This ID will be used
  // later to stop the operation and/or get its results.
  // TODO: Investigate whether we can use numerical IDs instead.
  string bert_operation_id = 1;
  // All the per-port BERTs that are considered one BERT operation and have the
  // same BERT operation ID.
  repeated PerPortRequest per_port_requests = 2;
}

message StartBERTResponse {
  // Per-port BERT start responses.
  message PerPortResponse {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;
    // BERT start status for this port.
    BertStatus status = 2;
  }
  // The same BERT operation ID given by the request.
  string bert_operation_id = 1;
  // Captures the results of starting BERT on a per-port basis.
  repeated PerPortResponse per_port_responses = 2;
}
```

The supported values for **prbs_polynomial** on NCS1004 are PRBS7, PRBS13, PRBS23, and PRBS31.

- The **StartBERT** RPC can return an error status in any one of the following scenarios:
  - When BERT operation is supported on none of the ports specified by the request.
  - When BERT is already in progress on any port specified by the request.
  - In case of any low-level hardware or software internal errors.

  The RPC returns an **OK** status when there is no error situation encountered.

Stop and Delete an Existing BERT Session from the Device

Stops an already in-progress BERT operation on a set of ports. The caller uses the BERT operation ID it previously used when starting the operation to stop it.

StopBERT

```
rpc StopBERT(StopBERTRequest) returns(StopBERTResponse) {}
```

Request and response messages

```
message StopBERTRequest {
  // Per-port BERT stop requests.
  message PerPortRequest {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;
  }
  // The same BERT operation ID given when BERT operation was started.
  string bert_operation_id = 1;
  // All the per-port BERTs that need to be stopped. Must be part of the BERT
  // operation specified by the `bert_operation_id` above.
  repeated PerPortRequest per_port_requests = 2;
}

message StopBERTResponse {
  // Per-port BERT stop responses.
  message PerPortResponse {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;
    // BERT stop status for this port.
    BertStatus status = 2;
  }
  // The same BERT operation ID given by the request.
  string bert_operation_id = 1;
  // Captures the results of stopping BERT on a per-port basis.
  repeated PerPortResponse per_port_responses = 2;
}
```

When the **PerPortRequest** field is not configured, then the device stops and deletes BERT sessions on all the ports associated with the BERT ID.

The RPC is expected to return an error status in any one of the following situations:

- When there is at least one BERT operation in progress on a port which cannot be stopped in the middle of the operation (either due to lack of support or internal problems).
- When no BERT operation, which matches the given BERT operation ID, is in progress or completed on any of the ports specified by the request.

The **StopBERT** RPC returns to an **OK** status when there is no error situation is encountered.

Note The BERT operation is considered completed if the device has a record or history of it. Also note that you might receive a stop request for a port which has completed BERT, as long as the recorded BERT operation ID matches the one specified by the request.

Get BERT Statistics for an Existing Session

Gets BERT results during the BERT operation or after the operation completes. The caller uses the BERT operation ID that it previously used when starting the operation to query it. The device stores results for the last BERT based on the required period of time.

GetBERTResults

```
rpc GetBERTResult(GetBERTResultRequest) returns(GetBERTResultResponse) {}
```

Request and response messages

```
message GetBERTResultRequest {
  // Per-port BERT get result requests.
  message PerPortRequest {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;
  }
  // The same BERT operation ID given when BERT operation was started.
  string bert_operation_id = 1;
  // All the per-port BERTs result of which we want to query. Must be part of
  // the BERT operation specified by the `bert_operation_id` above.
  repeated PerPortRequest per_port_requests = 2;
  // If set to true, the results for all the per-port BERTs will be returned.
  // `bert_operation_id` and `per_port_requests` will be ignored will be
  // ignored in that case.
  bool result_from_all_ports = 3;
}

message GetBERTResultResponse {
  // Per-port BERT results/status.
  message PerPortResponse {
    // Path to the interface corresponding to the port.
    types.Path interface = 1;
    // BERT result get status for this port. Only if the status is
    // BERT_STATUS_OK are the rest of the fields meaningful.
    BertStatus status = 2;
    // The ID of the BERT operation running on this port. Since the caller
    // can query the BERT results for all the ports, ID can potentially be
    // different for different ports.
    string bert_operation_id = 3;
    // The selected PRBS generating polynomial for BERT on this port.
    PrbsPolynomial prbs_polynomial = 4;
    // The last time BERT started on this port.
    uint64 last_bert_start_timestamp = 5;
    // The last time BERT results were read for this port.
    uint64 last_bert_get_result_timestamp = 6;
    // Indicate whether BERT peer lock has was established. If false,
    // `bert_lock_lost`, `error_count_per_minute`, and `total_errors` will not
    // be meaningful.
    bool peer_lock_established = 7;
    // Indicate whether BERT peer lock was lost after being established
    // once.
    bool peer_lock_lost = 8;
    // Sequence of bit errors per min since lock was established.
    repeated uint32 error_count_per_minute = 9;
    // Total number of bit errors accumulated since lock was established.
    uint64 total_errors = 10;
  }
  // Captures the BERT results on a per-port basis.
  repeated PerPortResponse per_port_responses = 1;
}
```

When the **per_port_requests** is ignored, then the device returns results and status for all the ports associated with the BERT ID.

The following table lists the descriptions of BERT results and status.

Table 3: BERT Results and Status

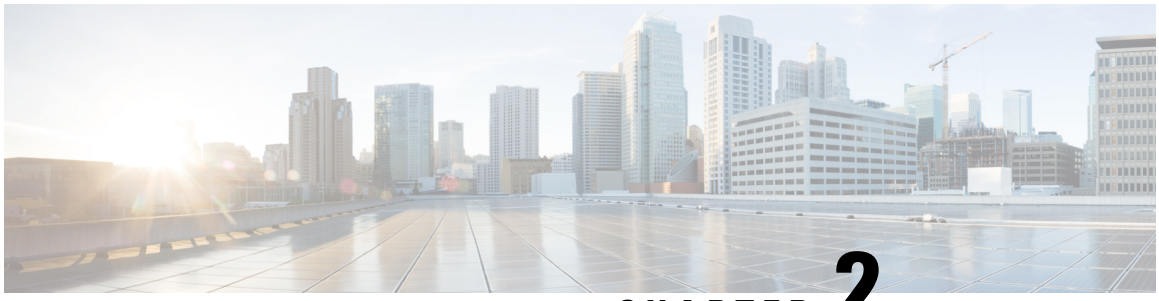
Field	Description
interface	Port in **types.Path** format representing a path in the open configuration interface model.
status	• **BERT_STATUS_OK** denotes that the BERT session is active. • **BERT_STATUS_PORT_NOT_RUNNING_BERT** denotes that BERT is not running as the duration has expired.
bert_operation_id	BERT operation ID that the port is associated with.
prbs_polynomial	The PRBS polynomial value that is configured.
last_bert_start_timestamp	Start operation timestamp in form of a 64-bit value UNIX time, which is the number of seconds elapsed since January 1, 1970 UTC.
repeated last_bert_get_results_timestamp	Timestamp of the last GetBERTResults operation in form of a 64-bit value UNIX time, which is the number of seconds elapsed since January 1, 1970 UTC.
peer_lock_established	Current status of peer lock. Note that there could be a 10-second delay in updating this field.
peer_lock_lost	Indicates if the peer lock is lost anytime after a peer lock is established. This field is only meaningful if **peer_lock_established** field is set.
error_count_per_minute	A list of one-minute historical PM buckets containing bit error counts. Historical buckets are maintained since the StartBERT operation started.
total_errors	Cumulative count of bit errors of the StartBERT operation.

The GetBERTResults RPC can return error status in any one of the following scenarios:

- When no BERT operation, which matches the given BERT operation ID, is in-progress or completed on any of the ports specified by the request.
- When the BERT operation ID does not match the in progress or completed BERT operation on any of the ports specified by the request.

The RPC returns an **OK** status when none of these situations is encountered.

Note The BERT operation is considered as completed only when the device has a record of it.



CHAPTER 2

Using 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:

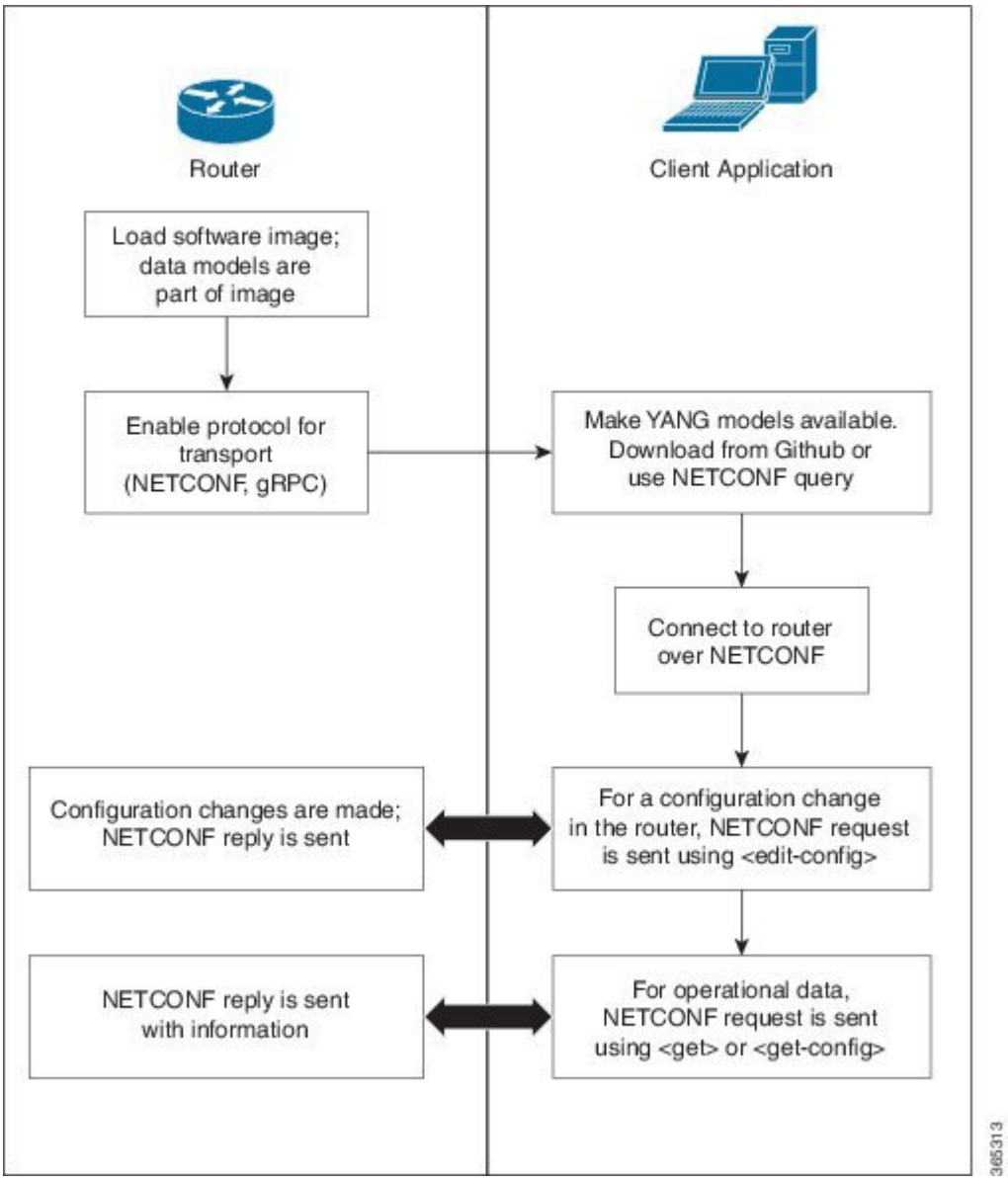
```
https://github.com/YangModels/yang/tree/master/vendor/cisco/xr
```

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

- Install the required packages (k9sec and mgbl)
- 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 in NCS 1004

Supported YANG Models in NCS 1004

Table 4: Feature History

Feature Name	Release Information	Feature Description
New Unified Model for Enhanced IKEv2 Encryption Support	Cisco IOS XR Release 24.1.1	The new model Cisco-IOS-XR-um-ikev2-cfg introduced in this release enhances the IKEv2 encryption. Now IKEv2 encryption complies with RFC 8784, which describes about using postquantum preshared keys (PPK) for IKEv2 encryption. The PPKs are generated with the help of the Cisco Secure Key Integration Protocol (SKIP) which makes the IKEv2 encryption resilient to quantum attacks.

The supported config and oper YANG models for NCS 1004 are listed below:

Config Yang Models	Oper Yang Models
Cisco-IOS-XR-osa-cfg.yang	Cisco-IOS-XR-osa-oper.yang
Cisco-IOS-XR-controller-optics-cfg.yang	Cisco-IOS-XR-controller-optics-oper.yang
Cisco-IOS-XR-pmengine-cfg.yang	Cisco-IOS-XR-pmengine-oper.yang
Cisco-IOS-XR-ethernet-lldp-cfg.yang	Cisco-IOS-XR-ethernet-lldp-oper.yang
Cisco-IOS-XR-ifmgr-cfg.yang	Cisco-IOS-XR-telemetry-model-driven-oper.yang
Cisco-IOS-XR-telemetry-model-driven-cfg.yang	Cisco-IOS-XR-fpd-infra-oper.yang
Cisco-IOS-XR-fpd-infra-cfg.yang	Cisco-IOS-XR-ikev2-oper.yang
Cisco-IOS-XR-ikev2-cfg.yang	Cisco-IOS-XR-otnsec-oper.yang
Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg	
Cisco-IOS-XR-um-ikev2-cfg	

The supported versions of Open Config model are listed below:

- openconfig-platform.yang
- openconfig-platform-transceiver.yang
- openconfig-terminal-device.yang
- openconfig-interfaces.yang

- openconfig-system.yang

Note openconfig-platform-transceiver.yang model is the augmented model of openconfig-platform.yang model.

Configure Slice

Step 1 Use the Cisco-IOS-XR-osa-cfg.yang YANG model for provisioning the slice with traffic on the client and trunk ports.

All the five client ports of the slice need to be configured at the same bitrate except for mixed mode configuration. Both the trunk ports are always set with the same FEC mode. In mixed mode configuration, the client ports are configured at different bitrates.

YANG model	Example
Cisco-IOS-XR-osa-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>0/1</node-name> <mxponder-slices xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-cfg"> <mxponder-slice> <slice-id>0</slice-id> <trunk-rates>six-hundred-gig</trunk-rates> <client-rates>hundred-gig-e</client-rates> </mxponder-slice> </mxponder-slices> </active-node> </active-nodes> </config> </edit-config> </rpc>

YANG model	Example
Cisco-IOS-XR-osa-cfg.yang	<?xml version="1.0"?> <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"> <hardware-module xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-cfg"> <node> <location>0_RP0_CPU0</location> <slice> <values>

YANG model	Example
	<client-rate>ten-and-hundred-gig</client-rate> <trunk-rate>two-hundred-gig</trunk-rate> <fec>sd7</fec> </values> <slice-id>0</slice-id> </slice> </node> </hardware-module> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-osa-oper.yang YANG model to verify the slice configuration.

YANG model	Example
Cisco-IOS-XR-osa-oper.yang	<?xml version="1.0" ?> <rpc message-id="856612" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter> <hw-module xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-oper" > <slice-all> <slice-info> <slice-id>0</slice-id> </slice-info> </slice-all> <slice-all> <slice-info> <slice-id>1</slice-id> </slice-info> </slice-all> <slice-all> <slice-info> <slice-id>2</slice-id> </slice-info> </slice-all> <slice-all> <slice-info> <slice-id>3</slice-id> </slice-info> </slice-all> </hw-module> </filter> </get> </rpc>

Configure Optics Controller

Step 1 Use the Cisco-IOS-XR-ifmgr-cfg.yang YANG model for configuring the optics controller.

YANG model	Example
Cisco-IOS-XR-ifmgr-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>Optics0/1/0/2</interface-name> <shutdown xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="create"/> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-controller-optics-cfg.yang YANG model for configuring the wavelength on the trunk port.

YANG model	Example
Cisco-IOS-XR-controller-optics-cfg.yang	<?xml version="1.0"?> <rpc message-id="102" 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>Optics0/0/0/2</interface-name> <optics xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-optics-cfg"> <optics-dwdm-carrier> <grid-type>50g-hz-grid</grid-type> <param-type>itu-ch</param-type> <param-value>1</param-value> </optics-dwdm-carrier> </optics> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 3 Use the Cisco-IOS-XR-controller-optics-oper.yang YANG model to verify the wavelength and channel mapping for trunk optics controllers.

YANG model	Example
Cisco-IOS-XR-controller-optics-oper.yang	<?xml version="1.0" ?> <rpc message-id="8566" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter type="subtree"> <optics-oper xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-optics-oper"> <optics-ports> <optics-port> <name>Optics0/0/0/13</name> <optics-dwdm-carrrier-channel-map> </optics-dwdm-carrrier-channel-map> </optics-port> </optics-ports> </optics-oper> </filter> </get> </rpc>

Configure Ethernet and Coherent DSP Controllers

Step 1 Use the Cisco-IOS-XR-ifmgr-cfg.yang YANG model to configure the Ethernet controller.

YANG model	Example
Cisco-IOS-XR-ifmgr-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>HundredGigECtrlr0/1/0/2</interface-name> <shutdown xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="create"/> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-ifmgr-cfg.yang YANG model to configure the Coherent DSP controller.

YANG model	Example
Cisco-IOS-XR-ifmgr-cfg.yang	<?xml version="1.0"?> <rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <edit-config>

YANG model	Example
	<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>CoherentDSP0/0/0/6</interface-name> <shutdown xc:operation="delete" /> </interface-configuration> <interface-configuration> <active>act</active> <interface-name>CoherentDSP0/0/0/13</interface-name> <shutdown></shutdown> </interface-configuration> <interface-configuration> <active>act</active> <interface-name>CoherentDSP0/0/0/20</interface-name> <shutdown></shutdown> </interface-configuration> <interface-configuration> <active>act</active> <interface-name>CoherentDSP0/0/0/27</interface-name> <shutdown></shutdown> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 3 Use the Cisco-IOS-XR-pfi-im-cmd-ctrlr-oper.yang YANG model to display the name, status, and port description of the Ethernet controller.

YANG model	Example
Cisco-IOS-XR-pfi-im-cmd-ctrlr-oper.yang	<?xml version="1.0" ?> <rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter> <controllers xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-pfi-im-cmd-ctrlr-oper"> <controllers> <controller> <interafce-name>HundredGigECtrlr0/0/0/8 </interafce-name> </controller> </controllers> </controllers> </filter> </get> </rpc>

Configure the GCC Interface

Use the Cisco-IOS-XR-controller-odu-cfg.yang YANG model to configure GCC interface.

YANG model	Example
Cisco-IOS-XR-controller-odu-cfg	<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 xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="create">act</active> <interface-name>ODU40/0/0/0/2</interface-name> <odu xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-odu-cfg"> <gcc-modes> <gcc-mode> <type>gcc2-mode</type> <mode>enable</mode> </gcc-mode> </gcc-modes> </odu> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:34d98974-474a-4396-ad1a-6dd4ddfa20bc"> <ok/> </rpc-reply>

Configure idle insertion

Use the Cisco-IOS-XR-drivers-icpe-ethernet-cfg.yang YANG model to configure idle insertion.

YANG model	Example
Cisco-IOS-XR-drivers-icpe-ethernet-cfg	<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 <interface-configuration> <active>act</active>

YANG model	Example
	<interface-name>HundredGigECtrlr0/1/0/9</interface-name> <holdoff-time xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-drivers-icpe-ethernet-cfg">3 </interface-configuration> </interface-configurations> </config> </edit-config> </rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:34d98974-474a-4396-ad1a-6dd4ddfa20bc"> <ok/> </rpc-reply>

Configure Loopback

Step 1 Use the Cisco-IOS-XR-ifmgr-cfg.yang and Cisco-IOS-XR-controller-otu-cfg YANG models for configuring Loopback.

YANG model	Example
Cisco-IOS-XR-ifmgr-cfg.yang Cisco-IOS-XR-controller-otu-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>CoherentDSP0/1/0/0</interface-name> <otu xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-controller-otu-cfg"> <otn-send-tti> <string-type>send-tti-full-ascii/full-ascii</string-type> <full-ascii-string>test1234</full-ascii-string> </otn-send-tti> <otn-expected-tti> <string-type>exp-tti-full-ascii/full-ascii</string-type> <full-ascii-string>test1234</full-ascii-string> </otn-expected-tti> </otu> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-ifmgr-cfg.yang and Cisco-IOS-XR-drivers-media-eth-cfg.yang YANG models for configuring the maintenance mode and loopback on an Ethernet controller.

YANG model	Example
Cisco-IOS-XR-ifmgr-cfg.yang Cisco-IOS-XR-drivers-media-eth-cfg.yang	<rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:34d98974-474a-4396-ad1a-6dd4ddfa20bc"> <ok/> </rpc-reply>

Configure Laser Squelch

Use the Cisco-IOS-XR-ifmgr-cfg.yang YANG model to configure laser squelch.

YANG model	Example
Cisco-IOS-XR-ifmgr-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>HundredGigECtrlr0/1/0/9</interface-name> <laser-squelch xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-drivers-icpe-ethernet-cfg"/> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc><rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:34d98974-474a-4396-ad1a-6dd4ddfa20bc"> <ok/> </rpc-reply>

Configure OTNsec on ODU4 Controllers

Step 1 Use the Cisco-IOS-XR-otnsec-cfg.yang YANG model to configure OTNsec on ODU4 controllers.

YANG model	Example
Cisco-IOS-XR-otnsec-cfg	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <edit-config>

YANG model	Example
	<target> <candidate/> </target> <config> <interface-configurations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ifmgr-cfg"> <interface-configuration> <active xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="create">act</active> <interface-name>ODU40/0/0/0/2</interface-name> <odu-otnsec xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-otnsec-cfg"> <ipv4> <session-id>1</session-id> <destination-address>10.1.1.1</destination-address> <source-address>10.1.1.2</source-address> </ipv4> <ik-ev2-profile>IP1</ik-ev2-profile> <policy>OP1</policy> </odu-otnsec> </interface-configuration> </interface-configurations> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-otnsec-cfg.yang YANG model to configure OTNsec on ODU4 controllers.

YANG model	Example
Cisco-IOS-XR-otnsec-cfg.yang	<rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:34d98974-474a-4396-ad1a-6dd4ddfa20bc"> <ok/> </rpc-reply>

Configure get keyring

Use the Cisco-IOS-XR-keyring-oper.yang YANG model for getting the configured keyring.

YANG model	Example
Cisco-IOS-XR-keyring-oper.yang	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get> <filter> <keyrings xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-keyring-oper"/> </filter> </get> </rpc><?xml version="1.0"?> <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <keyrings xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-keyring-oper">

YANG model	Example
	<keyring> <name>KR1</name> <keyring-name>KR1</keyring-name> <total-peers>8</total-peers> <peer> <peer-name>SITE-A-1</peer-name> <ip-address>10.1.1.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-2</peer-name> <ip-address>10.1.2.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-3</peer-name> <ip-address>10.1.3.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-4</peer-name> <ip-address>10.1.4.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-5</peer-name> <ip-address>10.1.5.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-6</peer-name> <ip-address>10.1.6.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-7</peer-name> <ip-address>10.1.7.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> <peer> <peer-name>SITE-A-8</peer-name> <ip-address>10.1.8.1</ip-address> <subnet>255.255.255.255</subnet> <local-psk>Configured</local-psk> <remote-psk>Configured</remote-psk> </peer> </keyring>

YANG model	Example
	</keyrings> </data> </rpc-reply>

Configure get ikev2 proposal/policy/profile

Use the Cisco-IOS-XR-osa-cfg.yang YANG model for getting the ikev2 proposal/policy/profile.

YANG model	Example
Cisco-IOS-XR-osa-cfg.yang	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get> <filter> <ikev2 xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ikev2-oper"/> </filter> </get> </rpc><rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get> <filter> <ikev2 xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ikev2-oper"/> </filter> </get> </rpc> .... Response - Wed Jul 31 2019 14:16:35 <?xml version="1.0"?> <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <ikev2 xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ikev2-oper"> <nodes> <node> <node-name>0/RP0/CPU0</node-name> <stats> <ike-sa-init-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>508</tx-res> <rx-req>508</rx-req> </ike-sa-init-cnt> <ike-auth-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>417</tx-res> <rx-req>417</rx-req> </ike-auth-cnt> <create-child-sa-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>60219</tx-res> <rx-req>60219</rx-req>

YANG model	Example

```
      </create-child-sa-cnt>
      <create-child-sa-ipsec-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>1149</tx-res>
       <rx-req>1149</rx-req>
      </create-child-sa-ipsec-cnt>
      <create-child-sa-ipsec-rekey-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>37445</tx-res>
       <rx-req>37445</rx-req>
      </create-child-sa-ipsec-rekey-cnt>
      <create-child-sa-ike-rekey-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>21625</tx-res>
       <rx-req>21625</rx-req>
      </create-child-sa-ike-rekey-cnt>
      <gsk-auth-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </gsk-auth-cnt>
      <gsk-reg-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </gsk-reg-cnt>
      <gsk-rekey-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </gsk-rekey-cnt>
      <gsk-rekey-ack-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </gsk-rekey-ack-cnt>
      <informational-cnt>
       <tx-req>119863</tx-req>
       <rx-res>117976</rx-res>
       <tx-res>177059</tx-res>
       <rx-req>177298</rx-req>
      </informational-cnt>
      <unsupported-critical-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </unsupported-critical-cnt>
      <invalid-ike-spi-cnt>
       <tx-req>0</tx-req>
       <rx-res>0</rx-res>
       <tx-res>0</tx-res>
       <rx-req>0</rx-req>
      </invalid-ike-spi-cnt>
      <invalid-major-version-ikev2-cnt>
```

YANG model	Example

```
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </invalid-major-version-ikev2-cnt>
      <invalid-syntax-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </invalid-syntax-cnt>
      <invalid-msg-id-ikev2-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </invalid-msg-id-ikev2-cnt>
      <invalid-spi-ikev2-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </invalid-spi-ikev2-cnt>
      <no-proposal-chosen-ikev2-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </no-proposal-chosen-ikev2-cnt>
      <invalid-ke-pyld-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </invalid-ke-pyld-cnt>
      <auth-failed-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </auth-failed-cnt>
      <single-pair-required-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </single-pair-required-cnt>
      <no-additional-sas-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </no-additional-sas-cnt>
      <internal-addr-failure-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
      </internal-addr-failure-cnt>
      <failed-cp-required-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
```

YANG model	Example
	<tx-res>0</tx-res> <rx-req>0</rx-req> </failed-cp-required-cnt> <ts-unacceptable-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </ts-unacceptable-cnt> <invalid-selectors-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </invalid-selectors-cnt> <initial-contact-ikev2-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>417</rx-req> </initial-contact-ikev2-cnt> <set-window-size-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>22042</tx-res> <rx-req>22042</rx-req> </set-window-size-cnt> <additional-ts-poss-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </additional-ts-poss-cnt> <ipcomp-supported-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </ipcomp-supported-cnt> <nat-detection-src-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </nat-detection-src-cnt> <nat-detection-dst-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </nat-detection-dst-cnt> <cookie-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </cookie-cnt> <use-transport-mode-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req>

YANG model	Example
	</use-transport-mode-cnt> <http-cert-lookup-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </http-cert-lookup-cnt> <rekey-sa-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>37445</rx-req> </rekey-sa-cnt> <esp-tfc-padding-not-supported-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </esp-tfc-padding-not-supported-cnt> <delete-reason-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </delete-reason-cnt> <custom-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </custom-cnt> <redirect-supported-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </redirect-supported-cnt> <redirect-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </redirect-cnt> <redirected-from-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </redirected-from-cnt> <ikev2-dpd-cnt> <tx-req>119699</tx-req> <rx-res>117972</rx-res> <tx-res>117989</tx-res> <rx-req>117989</rx-req> </ikev2-dpd-cnt> <cfg-request-cnt> <tx-req>0</tx-req> <rx-res>0</rx-res> <tx-res>0</tx-res> <rx-req>0</rx-req> </cfg-request-cnt> <cfg-reply-cnt>

YANG model	Example

```
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>417</rx-req>
       </cfg-reply-cnt>
       <cfg-set-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
       </cfg-set-cnt>
       <cfg-ack-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
       </cfg-ack-cnt>
       <nat-inside-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
       </nat-inside-cnt>
       <nat-outside-cnt>
        <tx-req>0</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
       </nat-outside-cnt>
       <no-nat-cnt>
        <tx-req>508</tx-req>
        <rx-res>0</rx-res>
        <tx-res>0</tx-res>
        <rx-req>0</rx-req>
       </no-nat-cnt>
       <tx-succ>360612</tx-succ>
       <rx-succ>356689</rx-succ>
       <tx-write-fail>0</tx-write-fail>
       <tx-send-fail>0</tx-send-fail>
       <tx-set-qos-fail>0</tx-set-qos-fail>
       <tx-socket-not-conn>0</tx-socket-not-conn>
       <tx-unknown-src-port>0</tx-unknown-src-port>
       <rx-get-if-fail>0</rx-get-if-fail>
       <rx-null-ifhndl>0</rx-null-ifhndl>
       <rx-get-ntwrk-offset>0</rx-get-ntwrk-offset>
       <rx-read-ip-head>0</rx-read-ip-head>
       <rx-get-transport-offset>0</rx-get-transport-offset>
       <rx-read-upd-head>0</rx-read-upd-head>
       <rx-unexpected-marker>0</rx-unexpected-marker>
       <rx-get-vrf-id>0</rx-get-vrf-id>
       <rx-read-pyld>0</rx-read-pyld>
      </stats>

      <summary>
       <total-sa>0</total-sa>
       <total-sa-active>0</total-sa-active>
       <total-sa-negotiating>0</total-sa-negotiating>
       <total-outgoing-sa>0</total-outgoing-sa>
       <outgoing-sa-active>0</outgoing-sa-active>
       <outgoing-sa-negotiating>0</outgoing-sa-negotiating>
       <total-incoming-sa>0</total-incoming-sa>
       <incoming-sa-active>0</incoming-sa-active>
```

YANG model	Example

```
      <incoming-sa-negotiating>0</incoming-sa-negotiating>
     </summary>

     <policies>
      <policy>
       <name>default</name>
       <policy-name>default</policy-name>
       <total-local-addr>1</total-local-addr>
       <addr>0.0.0.0</addr>
       <total-proposal>1</total-proposal>
       <proposal>default</proposal>
      </policy>

     </policies>
     <proposals>
      <proposal>
       <name>default</name>
       <proposal-name>default</proposal-name>
       <total-enc-alg>1</total-enc-alg>
       <encryption-alg>CBC-AES-256</encryption-alg>
       <total-hash-alg>2</total-hash-alg>
       <hash-alg>SHA 512</hash-alg>
       <hash-alg>SHA 384</hash-alg>
       <total-prf-alg>2</total-prf-alg>
       <prf-alg>SHA 512</prf-alg>
       <prf-alg>SHA 384</prf-alg>
       <total-group-alg>3</total-group-alg>
       <group-alg>Group 19</group-alg>
       <group-alg>Group 20</group-alg>
       <group-alg>Group 21</group-alg>
       <status>Complete</status>
      </proposal>

     </proposals>
     <profiles>
      <profile>
       <name>IP1</name>
       <profile-name>IP1</profile-name>
       <keyring-name>KR1</keyring-name>
       <match-any>false</match-any>
       <total-match-remote-peers>8</total-match-remote-peers>
       <addr>10.1.1.1</addr>
       <addr>10.1.2.1</addr>
       <addr>10.1.3.1</addr>
       <addr>10.1.4.1</addr>
       <addr>10.1.5.1</addr>
       <addr>10.1.6.1</addr>
       <addr>10.1.7.1</addr>
       <addr>10.1.8.1</addr>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <mask>255.255.255.255</mask>
       <lifetime-in-sec>120</lifetime-in-sec>
       <dpd-interval-in-sec>10</dpd-interval-in-sec>
       <dpd-retry-in-sec>2</dpd-retry-in-sec>
      </profile>
```

YANG model	Example
	</profiles> </node> </nodes> </ikev2> </data> </rpc-reply>

Configure Performance Monitoring

Step 1 Use the Cisco-IOS-XR-ifmgr-cfg.yang and Cisco-IOS-XR-pmengine-cfg.yang YANG models for configuring the performance monitoring parameters for the Optics, Ethernet, and coherentDSP controllers.

Step 2 Use the Cisco-IOS-XR-pmengine-oper.yang YANG models to view the performance monitoring parameters for the Optics, Ethernet, and coherentDSP controllers.

The table below shows an example that displays all the PM parameters for the optics controller. You can use specific filters for the required the output.

YANG model	Example
Cisco-IOS-XR-pmengine-oper.yang	<?xml version="1.0" ?> <rpc message-id="856612" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter type="subtree"> <performance-management xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-pmengine-oper"> <optics> <optics-ports> <optics-port>Optics0/0/0/1</optics-port> </optics-ports> </optics> </performance-management> </filter> </get> </rpc>

The table below shows an example that displays current 15 minute FEC PM for the Coherent DSP controller.

YANG model	Example
Cisco-IOS-XR-pmengine-oper.yang	<?xml version="1.0" ?> <rpc message-id="856612" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter type="subtree"> <performance-management xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-pmengine-oper"> <otu> <otu-ports> <otu-port> <name>CoherentDSP0/0/0/12</name> <otu-current>

YANG model	Example
	<otu-minute15> <otu-minute15fecs/> </otu-minute15> </otu-current> </otu-port> </otu-ports> </otu> </performance-management> </filter> </get> </rpc>

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
|  +--Lock
|  +--UnLock
|  +--Close-Session
|  +--Kill-Session
```

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	Retrieve specific interface configuration details from running configuration using filter option `<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <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>TenGigE0/0/0/2/0</interface-name> </interface-configuration> </interface-configurations> </filter> </get-config> </rpc>`
<get>	Retrieves running configuration and device state information	Retrieve all acl configuration and device state information. `Request: <get> <filter> <ipv4-acl-and-prefix-list xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-acl-oper"/> </filter> </get>`

NETCONF Operation	Description	Example
<edit-config>	Loads all or part of a specified configuration to the specified target configuration	Configure ACL configs using **Merge** operation `<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"> <ipv4-acl-and-prefix-list xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ipv4-acl-cfg" xc:operation=”merge”> <accesses> <access> <access-list-name>aclv4-1</access-list-name> <access-list-entries> <access-list-entry> <sequence-number>10</sequence-number> <remark>GUEST</remark> </access-list-entry> <access-list-entry> <sequence-number>20</sequence-number> <grant>permit</grant> <source-network> <source-address>172.0.0.0</source-address> <source-wild-card-bits>0.0.255.255</source-wild-card-bits> </source-network> </access-list-entry> </access-list-entries> </access> </accesses> </ipv4-acl-and-prefix-list> </config> </edit-config> </rpc>`  `Commit:` `<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <commit/> </rpc>`

<lock>	Allows the client to lock the entire configuration datastore system of a device	Lock 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	Example

<Unlock>	Releases a previously locked configuration.
	An <unlock> operation will not succeed if either of the following conditions is true:
	• The specified lock is not currently active.
	• The session issuing the <unlock> operation is not the same session that obtained the lock.
		Lock and unlock 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 the session. The server releases any locks and resources associated with the session and closes any associated connections.	Close 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>`

<kill-session>	Cancels operations currently in process, releases locks and resources associated with the session, and closes any associated connections.	Cancel a session if the ID is other session ID.
		`Request:`
		`<rpc message-id="101"`
		`xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		`<kill-session>`
		`<session-id>4</session-id>`
		`</kill-session>`
		`</rpc>`
		
		`Response:`
		`<rpc-reply message-id="101"`
		`xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">`
		`<ok/>`
		`</rpc-reply>`

NETCONF Operation to Get Configuration

This example shows how a NETCONF <get-config> request works for CDP feature.

The client initiates a message to get the current configuration of CDP running on the router. The router responds with the current CDP configuration.

Netconf Request (Client to Router)	Netconf Response (Router to Client)
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get-config> <source><running/></source> <filter> <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg"/> </filter> </get-config> </rpc>	<?xml version="1.0"?> <rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <data> <cdp xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cdp-cfg"> <timer>10</timer> <enable>true</enable> <log-adjacency></log-adjacency> <hold-time>200</hold-time> <advertise-v1-only></advertise-v1-only> </cdp> #22 </data> </rpc-reply>

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`.

CLI Over NETCONF

A new yang model, Cisco-IOS-XR-cli-cfg.yang is defined, which consists of a leaf node called 'cli'. The leaf node can be used to either send or receive the CLI configurations.

Limitations:

- Process restart and sysadmin mode is not supported .
- Rollback of configuration changes is not supported.
- Copying of images and logs to and from the box is not supported.

Edit Configuration Request

Edit-Config request with the sample CLI configurations is as follows. It must be followed by a commit rpc request for the configurations to be applied on the router.

Note The operation attribute, default operation parameter in an Edit-Config request can only be "Merge". Other operation parameters are not supported.

```
<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" >
```

```
<cli xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cli-cfg">
interface MgmtEth0/RP0/CPU0/0
no shutdown
ipv4 address dhcp
router static
address-family ipv4 unicast
0.0.0.0/0 MgmtEth0/RP0/CPU0/0 192.168.122.1
ssh server v2
ssh server netconf
netconf-yang agent ssh
</cli>
</config>
</edit-config>
</rpc>
```

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit/>
</rpc>
```

Copy Configuration Request

Copy-Config request with a sample CLI configuration is as follows. It must be followed by a commit rpc request for the configurations to be applied on the router.

Note A Copy-Config request replaces the configurations on the router with the configurations sent in the request. So, any reachability configuration (related to netconf, ssh, management ip) must be sent in the Copy-Config request.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<copy-config>
<target>
<candidate />
</target>
<source>
<config>
<cli xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cli-cfg">
interface MgmtEth0/RP0/CPU0/0
no shutdown
ipv4 address dhcp
router static
address-family ipv4 unicast
0.0.0.0/0 MgmtEth0/RP0/CPU0/0 192.168.122.1
ssh server v2
ssh server netconf
netconf-yang agent ssh
</cli>
</config>
</source>
</copy-config>
</rpc>
```

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<commit/>
</rpc>
```

Get Configurations Request

Get-Config request to retrieve the configurations on the router is as follows.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source>
```

```
<running/>
</source>
<filter type="subtree">
<cli xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cli-cfg">
</cli>
</filter>
</get-config>
</rpc>
```

Get request to retrieve the configurations on the router.

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get>
<filter>
<cli xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cli-cfg"/>
</filter>
</get>
</rpc>
```

Note Get requests always return the running configurations of the router and show cli is not supported in NETCONF clear text.

CLIDIFF Over NETCONF

A new RPC is implemented in the IOS-XR NETCONF agent, which can be used to retrieve the difference in the configuration changes before and after committing. The output is similar to the output of the command **show commit changes diff**. It shows the configurations added or removed after the commit is done. The added configurations will have a "+" sign and the removed configurations will have a "-" sign.

The sample output of **show commit changes diff** command is as follows:

```
RP/0/RSP0/CPU0:ASR9001-1(config)#sh commit changes diff
Fri Sep 23 08:03:07.485 UTC
Building configuration...
!! IOS XR Configuration 5.3.3
- interface Loopback1000
- description test
- ipv4 address 10.10.0.1 255.255.255.255
!
+ multicast-routing
+     address-family ipv4
+         interface Loopback0
+             enable
              !
          !
        !
+ multicast-routing
!
end
```

RPC Request

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-cli-config-diff xmlns= "http://cisco.com/ns/yang/Cisco-IOS-XR-cli-diff-act"/>
</rpc>
```

RPC Reply

```
<?xml version="1.0"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-cli-diff-act">Building
configuration...
!! IOS XR Configuration 6.5.1
+  vrf RED
+   address-family ipv6 unicast
     import route-target
+     65172:1
     !
     export route-target
+     65172:1
     !
    !
   !
end

</response>
</rpc-reply>
```

Note The above RPC reply is seen after adding the following CLI configuration:

```
vrf RED
 address-family ipv6 unicast
  import route-target
   65172:1
  !
  export route-target
   65172:1
  !
 !
!
```

Configure LLDP Drop

Step 1 Use the Cisco-IOS-XR-osa-cfg.yang YANG model to configure LLDP drop.

YANG model	Example

Cisco-IOS-XR-osa-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>0/1</node-name>`
	`  <mxponder-slices xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-cfg">`
	`  <mxponder-slice>`
	`  <slice-id>0</slice-id>`
	`  <lldp xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" nc:operation="merge">true</lldp>`
	`  </mxponder-slice>`

YANG model	Example
	</mxponder-slices> </active-node> </active-nodes> </config> </edit-config> </rpc>

Step 2 Use the Cisco-IOS-XR-osa-cfg.yang YANG model to delete LLDP drop configuration.

YANG model	Example
Cisco-IOS-XR-osa-cfg.yang	<?xml version="1.0"?> <rpc message-id="102" 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"> <hardware-module xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-cfg"> <node> <location>0_RP0_CPU0</location> <slice> <slice-id>0</slice-id> <lldp>false</lldp> </slice> </node> </hardware-module> </config> </edit-config> </rpc>

Step 3 Use the Cisco-IOS-XR-osa-cfg.yang YANG model to retrieve operational data for LLDP drop.

YANG model	Example
Cisco-IOS-XR-osa-cfg.yang	<?xml version="1.0"?> <rpc message-id="856615" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"> <get> <filter> <lldp-snoop-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-osa-oper.yang"/> </filter> </get> </rpc>

IPv4 PING Over NETCONF

Use the Cisco-IOS-XR-ping-act YANG model to do the ping test to the destination IPv4 addresses.The following example shows the RPC request and RPC response messages for a successful ping test. The destination host is reachable and the success rate is 100%.

YANG Model	Example
Cisco-IOS-XR-ping-act.yang	<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:28170002-365f-45be-a8e1-e1f54d8b64b5"><ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <destination> <destination>10.127.60.1</destination> </destination> </ping> </nc:rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:28170002-365f-45be-a8e1-e1f54d8b64b5"> <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <ipv4> <destination>10.127.60.1</destination> <data-size>100</data-size> <timeout>2</timeout> <pattern>abcd</pattern> <rotate-pattern>false</rotate-pattern> <replies> <reply> <reply-index>1</reply-index> <result>!</result> </reply> <reply> <reply-index>2</reply-index> <result>!</result> </reply> <reply> <reply-index>3</reply-index> <result>!</result> </reply> <reply> <reply-index>4</reply-index> <result>!</result>

YANG Model	Example
	</reply> <reply> <reply-index>5</reply-index> <result>!</result> </reply> </replies> <hits>5</hits> <total>5</total> <success-rate>100</success-rate> <rtt-min>1</rtt-min> <rtt-avg>1</rtt-avg> <rtt-max>2</rtt-max> </ipv4> </ping-response> </rpc-reply>

The following example shows the RPC request and RPC response messages for a failure ping test. The destination host is not reachable and the success rate is 0%.

YANG model	Example
Cisco-IOS-XR-ping-act.yang	<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:28170002-365f-45be-a8e1-e1f54d8b64b5"><ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <destination> <destination>10.127.60.1</destination> </destination> </ping> </nc:rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:02800209-6ebf-4955-8588-f6cdfd6f2750"> <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <ipv4> <destination>10.127.60.171</destination> <data-size>100</data-size> <timeout>2</timeout>

YANG model	Example
	<pattern>abcd</pattern> <rotate-pattern>false</rotate-pattern> <replies> <reply> <reply-index>1</reply-index> <result>.</result> </reply> <reply> <reply-index>2</reply-index> <result>.</result> </reply> <reply> <reply-index>3</reply-index> <result>.</result> </reply> <reply> <reply-index>4</reply-index> <result>.</result> </reply> <reply> <reply-index>5</reply-index> <result>.</result> </reply> </replies> <hits>0</hits> <total>5</total> <success-rate>0</success-rate> </ipv4> </ping-response> </rpc-reply>

IPv6 PING Over NETCONF

Table 5: Feature History

Feature Name	Release	Description
NETCONF Support for READ, WRITE, and Execute or Administrative Commands.	Cisco IOS XR Release 7.3.1	Support for IPv4 and IPv6 Ping test using the Cisco-IOS-XR-ping-act YANG model, instead of using CLI commands, is available. RPC (Remote Procedure Call) Request and Response messages are used to do the ping test, which is automated using scripts. This enables you to perform the ping test in a less time-consuming manner and to enhance network scalability.

Use the Cisco-IOS-XR-ping-act YANG model to do the ping test to the destination IPv6 addresses. The following example shows the RPC request and RPC response messages for a successful ping test. The destination host is reachable and the success rate is 100%.

YANG model	Example
Cisco-IOS-XR-ping-act.yang	<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:28170002-365f-45be-a8e1-e1f54d8b64b5"><ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <destination> <destination>2001:420:5446:2014::281:178</destination> </destination> </ping> </nc:rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:15798adc-f9f9-41b2-9aa5-a1c88dd788e8"> <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <ipv6> <destination>2001:420:5446:2014::281:178</destination> <repeat-count>50</repeat-count> <data-size>100</data-size> <timeout>2</timeout> <pattern>abcd</pattern>

YANG model	Example
	<rotate-pattern>false</rotate-pattern> <replies> <reply> <reply-index>1</reply-index> <result>!</result> </reply> <reply> <reply-index>2</reply-index> <result>!</result> </reply> <reply> <reply-index>3</reply-index> <result>!</result> </reply> <reply> <reply-index>4</reply-index> <result>!</result> </reply> <reply> <reply-index>5</reply-index> <result>!</result> </reply> </replies> <hits>5</hits> <total>5</total> <success-rate>100</success-rate> <rtt-min>1</rtt-min> <rtt-avg>1</rtt-avg> <rtt-max>2</rtt-max> </ipv6> </ping-response> </rpc-reply>

The following example shows the RPC request and RPC response messages for a failure ping test. The destination host is not reachable and the success rate is 0%.

YANG model	Example
Cisco-IOS-XR-ping-act.yang	<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:28170002-365f-45be-a8e1-e1f54d8b64b5"><ping xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <destination> <destination>2001:420:5446:2014::281:178</destination> </destination> </ping> </nc:rpc> <rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="urn:uuid:02800209-6ebf-4955-8588-f6cdfd6f2750"> <ping-response xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-ping-act"> <ipv6> <destination>2001:420:5446:2014::281:178</destination> <data-size>100</data-size> <timeout>2</timeout> <pattern>abcd</pattern> <replies> <reply> <reply-index>1</reply-index> <result>.</result> </reply> <reply> <reply-index>2</reply-index> <result>.</result> </reply> <reply> <reply-index>3</reply-index> <result>.</result> </reply> <reply> <reply-index>4</reply-index> <result>.</result>

YANG model	Example
	</reply> <reply> <reply-index>5</reply-index> <result>.</result> </reply> </replies> <hits>0</hits> <total>5</total> <success-rate>0</success-rate> </ipv6> </ping-response> </rpc-reply>

Configure the Line Card in Regen Mode

Table 6: Feature History

Feature Name	Release	Description
OC (Open Configuration) Support for Regen option	Cisco IOS XR Release 7.3.1	The OC support for configuring the Line Card in Regen mode is available. This enables you to perform the configuration using scripts, which is less time-consuming. Also, the Open Configuration model supports the use of vendor-neutral data models to configure and manage the network.

Procedure

	Command or Action	Purpose

Step 1	You can configure Regen mode for the Line Card on slot 1 using the following scripts:	{
		
		"openconfig-terminal-device:terminal-device": {
		
		"logical-channels":
		{

	Command or Action	Purpose

		"channel": [{
		"index": 10000,
		"config": {
		"index": 10000,
		"admin-state": "ENABLED",
		"description": "Coherent Logical Channel",
		"logical-channel-type": "openconfig-transport-types:PROT_OTN"
		},
		"logical-channel-assignments": {
		"assignment": [{
		"index": 1,
		"config": {
		"index": 1,
		"allocation": 300,
		"assignment-type": "OPTICAL_CHANNEL",
		"description":

	Command or Action	Purpose

		"Coherent to optical assignemnt",
		
		"optical-channel": "0/1-OpticalChannel0/1/0/0"
		
		}
		
		},
		
		{
		
		"index": 2,
		
		"config": {
		
		"index": 2,
		
		"allocation": 300,
		
		"assignment-type": "LOGICAL_CHANNEL",
		
		"description": "Coherent to optical assignemnt index junk",
		
		"logical-channel": 10001
		
		}
		
		}
		
		]

	Command or Action	Purpose

		}
		},
		{
		"index": 10001,
		"config": {
		"index": 10001,
		"admin-state": "ENABLED",
		"description": "Coherent Logical Channel",
		"logical-channel-type": "openconfig-transport-types:PROT_OTN"
		},
		"logical-channel-assignments": {
		"assignment": [{
		"index": 1,
		"config": {
		"index": 1,
		"allocation": 300,

	Command or Action	Purpose

		"assignment-type": "OPTICAL_CHANNEL",
		"description": "Coherent to optical assignemnt",
		"optical-channel": "0/1-OpticalChannel0/1/0/1"
		}
		},
		{
		"index": 2,
		"config": {
		"index": 2,
		"allocation": 300,
		"assignment-type": "LOGICAL_CHANNEL",
		"description": "Coherent to optical assignemnt",
		"logical-channel": 10000
		}

	Command or Action	Purpose

		}
		]
		}
		}
		]
		}
		},
		"openconfig-platform:components": {
		"component": [{
		"name": "0/1-OpticalChannel0/1/0/0",
		"openconfig-terminal-device:optical-channel": {
		"config": {
		"line-port": "0/1-Optics0/1/0/0"
		}
		}
		},
		{
		"name": "0/1-OpticalChannel0/1/0/1",
		"openconfig-terminal-device:optical-channel": {
		"config": {
		"line-port": "0/1-Optics0/1/0/1"

	Command or Action	Purpose
		} } }] } }

Configure Subsea Parameters

Table 7: Feature History

Feature Name	Release Information	Feature Description
OC (Open Configuration) Support for Subsea Parameters	Cisco IOS XR Release 7.3.2	The OC (Open Configuration) support for subsea parameters is introduced. This enables you to configure the subsea parameters using OC data models. This was defined under OC-platform as part of the vendor augmented data model.

The open configuration for subsea parameters is as follows:

```
{
 "openconfig-platform:components": {
 "component": [{
 "name": "0/0-OpticalChannel0/0/0/0",
 "openconfig-terminal-device:optical-channel": {
 "config": {
 "line-port": "0/0-Optics0/0/0/0"
   },
 "Cisco-IOS-XR-openconfig-terminal-device-ext:extended": {
  "config": {
  "optics-cd-max": 250000,
  "enh-colorless-mode": 3,
  "enh-sop-tol-mode": 3,
  "nleq-compensation": 2,
  "cross-pol-gain-mode": 10,
  "cross-pol-weight-mode": 4,
  "cpr-ext-win-mode": 8,
  "rx-voa-fixed-ratio": 1700,
  "filter-roll-off-factor": "0.074"
  }
  }
  }
  },

 {
"name": "0/0-OpticalChannel0/0/0/1",
```

```
"openconfig-terminal-device:optical-channel": {
"config": {
"line-port": "0/0-Optics0/0/0/1"
 },
"Cisco-IOS-XR-openconfig-terminal-device-ext:extended": {
"config": {
"optics-cd-max": 250000,
"enh-colorless-mode": 3,
"enh-sop-tol-mode": 3,
"nleq-compensation": 2,
"cross-pol-gain-mode": 4,
"cross-pol-weight-mode": 4,
"cpr-ext-win-mode": 8,
"rx-voa-fixed-ratio": 1700,
"filter-roll-off-factor": "0.074"
}
}
}
}
]
}
}
```


Examples Using gRPC

Example—Verify the Slice Configuration Using gRPC

Set-up:

- Client—client_v3
- Client IP address and configured grpc port—198.51.100.1:57500

```
./client_v3 -server 198.51.100.1:57500 -oper show-cmd-text -cli_input_file show-hw-module
```

The slice configuration is displayed.


```
{
  "Response": "{\"ResReqId\":753690684504425618,\"output\":\"\\n---------------------------
 show hw-module slice all ---------------------------\\nSlice ID:              1\\nStatus:
                  Provisioned\\nClient Bitrate:          100\\nTrunk Bitrate:
100\\nDP FPGA Version:         H201 (NEED UPG)\\n\\nClient Port -   Trunk Port\\t
CoherentDSP0/0/0/12\\t CoherentDSP0/0/0/13\\nTraffic Split
Percentage\\n\\nHundredGigECtrlr0/0/0/7  \\t                100
0\\nHundredGigECtrlr0/0/0/11 \\t                  0                100\\n\\n\\n\"}",
  "FatalErrors": ""
}
```


Example—View the Optics Controller Configuration Using gRPC and Yang

Set-up:

- Client—client_v3
- Client IP address and configured grpc port—198.51.100.1:57500
- Yang model—Cisco-IOS-XR-ifmgr-cfg

```
./client -server_addr=198.51.100.1:57500 -username=root -password=lab -oper=get-config
-yang_path='{"Cisco-IOS-XR-ifmgr-cfg:interface-configurations": [null]}'
```

The optics controller configuration is displayed.


```
{
 "Cisco-IOS-XR-ifmgr-cfg:interface-configurations": {
  "interface-configuration": [
   {
    "active": "act",
    "interface-name": "Optics0/0/0/5",
    "shutdown": [null]
   },
   {
    "active": "act",
    "interface-name": "Optics0/0/0/6",
    "Cisco-IOS-XR-controller-optics-cfg:optics": {
     "optics-dwdm-carrier": {
      "grid-type": "100mhz-grid",
      "param-type": "frequency",
      "param-value": 1927000
     }
    },
    "secondary-admin-state": "maintenance"
   },
   {
    "active": "act",
    "interface-name": "Optics0/0/0/12",
    "shutdown": [
     null
    ]
   },
   {
    "active": "act",
    "interface-name": "Optics0/0/0/13",
    "Cisco-IOS-XR-controller-optics-cfg:optics": {
     "optics-dwdm-carrier": {
      "grid-type": "100mhz-grid",
      "param-type": "frequency",
      "param-value": 1927000
     }
    },
    "secondary-admin-state": "maintenance"
   },
   {
    "active": "act",
    "interface-name": "Optics0/0/0/14",
    "Cisco-IOS-XR-controller-optics-cfg:optics": {
     "rx-thresholds": {
      "rx-threshold": [
       {
        "rx-threshold-type": "low",
        "rx-threshold": -120
       },
       {
        "rx-threshold-type": "high",
        "rx-threshold": 49
       }]}}}
    ,
   {
    "active": "act",
    "interface-name": "Optics0/0/0/18",
    "Cisco-IOS-XR-controller-optics-cfg:optics": {
     "rx-thresholds": {
      "rx-threshold": [
```

```
    {
     "rx-threshold-type": "low",
     "rx-threshold": -120
    },
    {
     "rx-threshold-type": "high",
     "rx-threshold": 49
    }]}}}
   ,
{
 "active": "act",
 "interface-name": "Optics0/0/0/19",
 "shutdown": [
  null
 ],
 "Cisco-IOS-XR-controller-optics-cfg:optics": {
  "optics-dwdm-carrier": {
   "grid-type": "50g-hz-grid",
   "param-type": "frequency",
   "param-value": 19270
  }}}
,
{
 "active": "act",
 "interface-name": "Optics0/0/0/20",
 "Cisco-IOS-XR-controller-optics-cfg:optics": {
  "optics-dwdm-carrier": {
   "grid-type": "50g-hz-grid",
   "param-type": "frequency",
   "param-value": 19270
  },
  "rx-thresholds": {
   "rx-threshold": [
    {
     "rx-threshold-type": "low",
     "rx-threshold": -120
    },
    {
     "rx-threshold-type": "high",
     "rx-threshold": 49
    }]}}}
   ],
{
 "active": "act",
 "interface-name": "Optics0/0/0/26",
 "shutdown": [
  null
 ]
},
{
 "active": "act",
 "interface-name": "Optics0/0/0/27",
 "shutdown": [
  null
 ]
},
{
 "active": "act",
 "interface-name": "MgmtEth0/RP0/CPU0/0",
 "Cisco-IOS-XR-ipv4-io-cfg:ipv4-network": {
  "addresses": {
   "primary": {
    "address": "10.77.132.165",
    "netmask": "255.255.255.0"
```

```
    }}}}
   ,
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/0/1",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/0/2",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/0/3",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
```

```
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/0/4",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/11/1",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
```

```
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/11/2",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/11/3",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
      {
       "ether-threshold": "thresh-fcs-err",
       "ether-threshold-value": 1000
      }
     ]
    }
   }
  }
 }
},
{
 "active": "act",
 "interface-name": "TenGigECtrlr0/0/0/11/4",
 "Cisco-IOS-XR-pmengine-cfg:performance-management": {
  "ethernet-minute15": {
   "minute15-ether": {
    "minute15-ether-reports": {
     "minute15-ether-report": [
      {
       "ether-report": "report-fcs-err"
      }
     ]
    },
    "minute15-ether-thresholds": {
     "minute15-ether-threshold": [
```

```
          {
           "ether-threshold": "thresh-fcs-err",
           "ether-threshold-value": 1000
          }
         ]
        }
       }
      }
     }
    }
   ]
  }
}
emsGetConfig: ReqId 1, byteRecv: 7455

----------------- gRPC Summary ----------------------

Operation: get-config
Number of iterations: 1
Total bytes transferred: 7455
Number of bytes per second: 124482
Ave elapsed time in seconds: 0.059888
Min elapsed time in seconds: 0.059888
Max elapsed time in seconds: 0.059888

--------------- End gRPC Summary ---------------------
```

Unified YANG Models

Table 8: Feature History

Feature Name	Release Information	Feature Description
Unified YANG Models	Cisco IOS XR Release 7.5.1	CLI-based Yang data models, also known as Unified YANG models, are introduced in R7.5.1. The Unified YANG models provide a complete coverage of the router functionality, and serve as an abstraction for YANG and CLI commands.

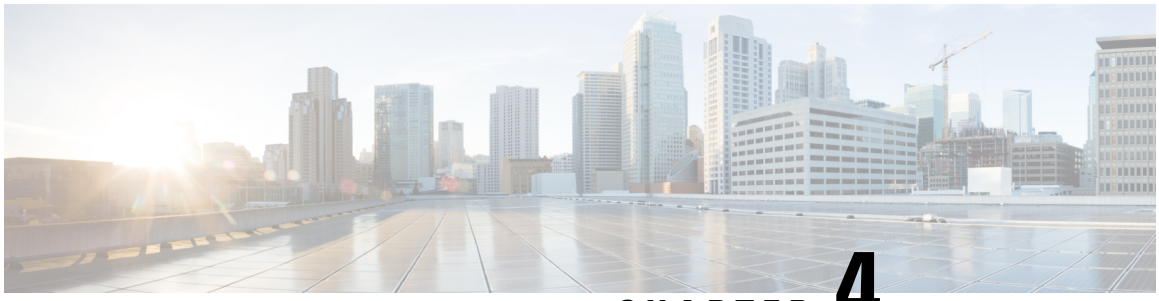
CLI-based YANG data models, also known as Unified YANG models, are introduced in R7.5.1. The Unified YANG models provide a complete coverage of the router functionality, and serve as an abstraction for YANG and CLI commands. Unified YANG models are generated from the CLI and replace the native schema-based models. The Unified YANG models are available in the location: pkg/yang. The term **um** in a model name indicates that the YANG model is a Unified model. For example, Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg model.

Use the Cisco-IOS-XR-um-location-cfg and Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg unified YANG models to configure the slice with traffic on both the client and trunk ports.

YANG model	Example
Cisco-IOS-XR-um-location-cfg Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg	<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101"> <get-config> <source> <running/> </source> <filter> <locations xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-um-location-cfg"> <location> <location-name>0/1</location-name> <hw-module xmlns="http://cisco.com/ns/yang/ Cisco-IOS-XR-um-ncs1004-hw-module-osa-cfg"> <mxponder> <client-rate>100GE</client-rate> <trunk-rate>300G</trunk-rate> </mxponder> </hw-module> </location> </locations> </filter> </get-config> </rpc>

The above example of the YANG model is equivalent to the following CLI:

```
hw-module location 0/1 mxponder
 client-rate 100GE
 trunk-rate 300G
```



CHAPTER 4

Structure of YANG Models

Structure of YANG Models

YANG data models can be represented in a hierarchical, tree-based structure with nodes. This representation makes the models easy to understand.

There are two Terminal-device models for Cisco NCS 1004, such as:

- OpenConfig Terminal model
- OpenConfig Platform model

For more details on supported versions, see Supported YANG Models in NCS 1004, on page 5.

The following is the tree structure of the OpenConfig Terminal model:

```
module: openconfig-terminal-device
   +--rw terminal-device
      +--ro state
      +--rw logical-channels
      |  +--rw channel* [index]
      |     +--rw index
      |     +--rw config
      |     |  +--rw index?                  uint32
      |     |  +--rw description?            string
      |     |  +--rw admin-state?            oc-opt-types:admin-state-type
      |     |  +--rw rate-class?             identityref
      |     |  +--rw trib-protocol?          identityref
      |     |  +--rw logical-channel-type?   identityref
      |     |  +--rw loopback-mode?          oc-opt-types:loopback-mode-type
      |     +--ro state
```

Structure of YANG Models

```
|      |  +--ro index?                  uint32
|      |  +--ro description?            string
|      |  +--ro admin-state?            oc-opt-types:admin-state-type
|      |  +--ro rate-class?             identityref
|      |  +--ro trib-protocol?          identityref
|      |  +--ro logical-channel-type?   identityref
|      |  +--ro loopback-mode?          oc-opt-types:loopback-mode-type
|      |  +--ro link-state?             enumeration
|      +--rw otn
|      |  +--rw config
|      |  |  +--rw tti-msg-transmit?   string
|      |  |  +--rw tti-msg-expected?   string
|      |  |  +--rw tti-msg-auto?       boolean
|      |  +--ro state
|      |     +--ro tti-msg-transmit?           string
|      |     +--ro tti-msg-expected?           string
|      |     +--ro tti-msg-recv?               string
|      |     +--ro errored-seconds?            yang:counter64
|      |     +--ro severely-errored-seconds?   yang:counter64
|      |     +--ro unavailable-seconds?        yang:counter64
|      |     +--ro fec-corrected-bits?         yang:counter64
|      |     +--ro background-block-errors?    yang:counter64
|      |     +--ro fec-uncorrectable-words
|      |     +--ro pre-fec-ber
|      |     |  +--ro instant?   decimal64
|      |     |  +--ro avg?       decimal64
|      |     |  +--ro min?       decimal64
|      |     |  +--ro max?       decimal64
|      |     +--ro post-fec-ber
|      |     |  +--ro instant?   decimal64
|      |     |  +--ro avg?       decimal64
|      |     |  +--ro min?       decimal64
|      |     |  +--ro max?       decimal64
|      +--rw ethernet
|      |  +--rw config
|      |  +--ro state
|      |     +--ro in-mac-pause-frames?      yang:counter64
|      |     +--ro in-oversize-frames?       yang:counter64
|      |     +--ro in-jabber-frames?         yang:counter64
|      |     +--ro in-fragment-frames?       yang:counter64
|      |     +--ro in-crc-errors?            yang:counter64
|      |     +--ro out-mac-pause-frames?     yang:counter64
|      +--rw ingress
|      |  +--rw config
|      |  |  +--rw transceiver?        -> /oc-platform:components/component/name
|      |  |  +--rw physical-channel*   ->
/oc-platform:components/component/oc-transceiver:transceiver/physical-channels/channel/index

|      |  +--ro state
|      |     +--ro transceiver?        -> /oc-platform:components/component/name
|      |     +--ro physical-channel*   ->
/oc-platform:components/component/oc-transceiver:transceiver/physical-channels/channel/index

|      +--rw logical-channel-assignments
|         +--rw assignment* [index]
|            +--rw index     -> ../config/index
|            +--rw config
|            |  +--rw index?             uint32
|            |  +--rw description?       string
|            |  +--rw assignment-type?   enumeration
|            |  +--rw logical-channel?   ->
/terminal-device/logical-channels/channel/index
|            |  +--rw optical-channel?   -> /oc-platform:components/component/name
|            |  +--rw allocation?        decimal64
```

```
      |           +--ro state
      |              +--ro index?             uint32
      |              +--ro description?       string
      |              +--ro assignment-type?   enumeration
      |              +--ro logical-channel?   ->
/terminal-device/logical-channels/channel/index
      |              +--ro optical-channel?   -> /oc-platform:components/component/name
      |              +--ro allocation?        decimal64
      +--rw operational-modes
         +--ro mode* [mode-id]
            +--ro mode-id    -> ../state/mode-id
            +--ro config
            +--ro state
               +--ro mode-id?       uint16
               +--ro description?   string
               +--ro vendor-id?     string
```

The following is the tree structure of the OpenConfig Platform model:

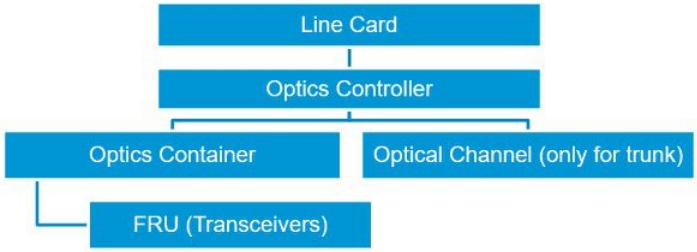
```
module: openconfig-platform
   +--rw components
      +--rw component* [name]
         +--rw name                              -> ../config/name
         +--rw config
         |  +--rw name?   string
         +--ro state
         |  +--ro name?          string
         |  +--ro type?          union
         |  +--ro id?            string
         |  +--ro description?   string
         |  +--ro mfg-name?      string
         |  +--ro version?       string
         |  +--ro serial-no?     string
         |  +--ro part-no?       string

         +--rw oc-transceiver:transceiver
         |  +--ro oc-transceiver:state
         |  |  +--ro oc-transceiver:form-factor?                identityref
         |  |  +--ro oc-transceiver:present?                    enumeration
         |  |  +--ro oc-transceiver:connector-type?             identityref
         |  |  +--ro oc-transceiver:internal-temp?              int16
         |  |  +--ro oc-transceiver:vendor?                     string
         |  |  +--ro oc-transceiver:vendor-part?                string
         |  |  +--ro oc-transceiver:vendor-rev?                 string
         |  |  +--ro oc-transceiver:ethernet-compliance-code?   identityref
         |  |  +--ro oc-transceiver:sonet-sdh-compliance-code?  identityref
         |  |  +--ro oc-transceiver:otn-compliance-code?        identityref
         |  |  +--ro oc-transceiver:serial-no?                  string
         |  |  +--ro oc-transceiver:date-code?                  yang:date-and-time
         |  |  +--ro oc-transceiver:fault-condition?            boolean
         |  +--rw oc-transceiver:physical-channels
         |     +--rw oc-transceiver:channel* [index]
         |        +--rw oc-transceiver:index     -> ../config/index
         |          +--ro oc-transceiver:output-frequency?     oc-opt-types:frequency-type

         |           +--ro oc-transceiver:output-power
         |           |  +--ro oc-transceiver:instant?   decimal64
         |           |  +--ro oc-transceiver:avg?       decimal64
         |           |  +--ro oc-transceiver:min?       decimal64
         |           |  +--ro oc-transceiver:max?       decimal64
         |           +--ro oc-transceiver:input-power
         |           |  +--ro oc-transceiver:instant?   decimal64
         |           |  +--ro oc-transceiver:avg?       decimal64
         |           |  +--ro oc-transceiver:min?       decimal64
         |           |  +--ro oc-transceiver:max?       decimal64
```

```
          |           +--ro oc-transceiver:laser-bias-current
          |              +--ro oc-transceiver:instant?   decimal64
          |              +--ro oc-transceiver:avg?       decimal64
          |              +--ro oc-transceiver:min?       decimal64
          |              +--ro oc-transceiver:max?       decimal64
          +--rw oc-opt-term:optical-channel
             +--rw oc-opt-term:config
             |  +--rw oc-opt-term:frequency?             oc-opt-types:frequency-type
             |  +--rw oc-opt-term:target-output-power?   decimal64
             |  +--rw oc-opt-term:operational-mode?      uint16
             |  +--rw oc-opt-term:line-port?             ->
/oc-platform:components/component/name
             +--ro oc-opt-term:state
                +--ro oc-opt-term:frequency?
oc-opt-types:frequency-type
                +--ro oc-opt-term:target-output-power?                          decimal64
                +--ro oc-opt-term:operational-mode?                             uint16
                +--ro oc-opt-term:line-port?                                    ->
/oc-platform:components/component/name
                +--ro oc-opt-term:chromatic-dispersion
                |  +--ro oc-opt-term:instant?   decimal64
                |  +--ro oc-opt-term:avg?       decimal64
                |  +--ro oc-opt-term:min?       decimal64
                |  +--ro oc-opt-term:max?       decimal64
                +--ro oc-opt-term:second-order-polarization-mode-dispersion
                |  +--ro oc-opt-term:instant?   decimal64
                |  +--ro oc-opt-term:avg?       decimal64
                |  +--ro oc-opt-term:min?       decimal64
                |  +--ro oc-opt-term:max?       decimal64
                +--ro oc-opt-term:polarization-dependent-loss
                   +--ro oc-opt-term:instant?   decimal64
                   +--ro oc-opt-term:avg?       decimal64
                   +--ro oc-opt-term:min?       decimal64
                   +--ro oc-opt-term:max?       decimal64
```

Inventory Details of Terminal-device Model

The hierarchy of Cisco NCS 1004 inventory is shown below:

Figure 3: Hierarchy of Cisco NCS 1004 Inventory



The inventory details and the naming convention of the components used in the Cisco NCS 1004 Terminal-device model are as follows:

Table 9: Inventory Details

Components	Naming Convention
Optics Controller	R/S-OpticsCtlrR/S/I/P
Optics Container	R/S-OpticsContainerR/S/I/P
Transceivers	R/S-OpticsR/S/I/P
Optical Channel Module	R/S-OpticalChannelR/S/I/P

The following table lists all the valid transceivers and optical channels that can be used for configuring Cisco NCS 1004 using Terminal-device model:

Table 10: Transceiver and Optical Channel Details

Components	Applicable Channels
Transceivers	0/0-Optics0/0/0/0 to 0/0-Optics0/0/0/13 0/1-Optics0/1/0/0 to 0/1-Optics0/1/0/13 0/2-Optics0/2/0/0 to 0/2-Optics0/2/0/13 0/3-Optics0/3/0/0 to 0/3-Optics0/3/0/13
Optical Channels	• 0/0-OpticalChannel0/0/0/0 • 0/0-OpticalChannel0/0/0/1 • 0/1-OpticalChannel0/1/0/0 • 0/1-OpticalChannel0/1/0/1 • 0/2-OpticalChannel0/2/0/0 • 0/2-OpticalChannel0/2/0/1 • 0/3-OpticalChannel0/3/0/0 • 0/3-OpticalChannel0/3/0/1

Note Only the optical channels of trunk ports must be mapped to the line ports. For more information about the port details, see Slice and Port Numbering.

Configuring Cisco NCS1004 Using Terminal-device Model

The following configurations are supported on the 1.2 Tbps line card. Client port operate at 100GE and OTU4 and map to trunk ports operating at 200G, 300G, 400G, 500G, or 600G.

You can configure the client port to OTU4 only in the muxponder configuration. LLDP drop, L1 encryption, and AINS are not supported on the OTU4 configuration.

The following table displays the client and trunk ports that are enabled for the muxponder configuration.

Trunk Data Rate	Client Data Rate (100GE/OTU4)	Trunk Ports	Client Ports
200	100GE/OTU4	0, 1	2,3, 4, 5
300	100GE/OTU4	0, 1	2, 3, 4, 5, 6, 7
400	100GE	0, 1	2, 3, 4, 5, 6, 7, 8, 9
500	100GE	0, 1	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
600	100GE	0,1	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13

The following table displays the client and trunk ports that are enabled for the muxponder slice 0 configuration.

Trunk Data Rate	Client Data Rate	Trunk Ports	Client Ports
200	100	0	2, 3
300	100	0	2, 3, 4
400	100	0	2, 3, 4, 5
500	100	0	2, 3, 4, 5, 6
600	100	0	2, 3, 4, 5, 6, 7

The following table displays the client and trunk ports that are enabled for the muxponder slice 1 configuration.

Trunk Data Rate	Client Data Rate	Trunk Ports	Client Ports
200	100	1	8, 9
300	100	1	8, 9, 10
400	100	1	8, 9, 10, 11
500	100	1	8, 9, 10, 11, 12
600	100	1	8, 9, 10, 11, 12, 13

All configurations can be accomplished using appropriate values for client bitrate and trunk bitrate parameters of the **hw-module** command.

The following table displays the trunk parameter ranges.

Trunk Payload	FEC	Min BPS	Max BPS	Min GBd	Max GBd
200G	27%	2	4.40625	31.51	69.43
300G	27%	2.8984375	6	34.7175497	71.8681352
400G	27%	3.8671875	6	46.2900663	71.8197392

Trunk Payload	FEC	Min BPS	Max BPS	Min GBd	Max GBd
500G	27%	4.8281250	6	57.8625828	71.9068991
600G	15%	5.2578125	-	-	71.9552971

Migrating CLI to Terminal-device Configuration

Cisco NCS 1004 supports migration from CLI to OC configuration only, vice-versa is not supported. The transition from CLI to terminal-device must be done via merge-config operation in gRPC.

To migrate from CLI configuration to the terminal-device configuration, perform the following:

Step 1 Enable the transition from CLI configuration to the terminal-device configuration, using the following command:

terminal-device transition cli-to-yang enable

Step 2 You must configure a slice using the CLI configuration command. For more details, see Configure the Slice.

Note Do not use all keyword to configure all slices, instead you must configure each slice individually.

Note Configure the trunk port frequencies with 100MHz spacing as after the migration to OC Models only 100MHz spacing is supported.

Note Ignore this step if you are migrating a configured slice. You cannot change the slice configuration while performing migration. For example, if you have configured 100G to 200 G traffic on a slice using CLI, then you can perform OC configuration for the same 100G to 200G slice configuration.

Step 3 Apply OC configuration using Netconf or gRPC. For more details, see Configuring Cisco NCS1004 Using Terminal-device Model, on page 71.

Step 4 Remove the slice configuration for the migrated slice. This configuration does not impact the traffic as OC configuration is already applied.

To disable the transition from CLI configuration to the terminal-device configuration, use the following command:

terminal-device transition cli-to-yang disable

OpenConfig Terminal Device

OC-terminal MDT data for Trunk Controller is as follows:

```
"openconfig": {
  "terminal-device": {
    "logical-channels": {
      "channel": {
        "30000": {
          "logical-channel-assignments": {
            "assignment": {
              "1": {
```

```
      "state": {
        "allocation": 500,
        "assignment-type": "OPTICAL_CHANNEL",
        "description": "Coherent to optical assignemnt",
        "index": 1,
        "optical-channel": "0_0-OpticalChannel0_0_0_0"
      }
    }
  }
},
"otn": {
  "state": {
    "background-block-errors": 0,
    "errored-seconds": 0,
    "esnr": {
      "avg": 0,
      "instant": 0,
      "interval": 30000000000,
      "max": 0,
      "max-time": 1573537980083123944,
      "min": 0,
      "min-time": 1573537980083123944
    },
    "fec-corrected-bits": 0,
    "fec-uncorrectable-words": 0,
    "post-fec-ber": {
      "avg": 0,
      "instant": 0,
      "interval": 30000000000,
      "max": 0,
      "max-time": 0,
      "min": 0,
      "min-time": 0
    },
    "pre-fec-ber": {
      "avg": 0,
      "instant": 0,
      "interval": 30000000000,
      "max": 0,
      "max-time": 0,
      "min": 0,
      "min-time": 0
    },
    "q-value": {
      "avg": 0,
      "instant": 0,
      "interval": 30000000000,
      "max": 0,
      "max-time": 0,
      "min": 0,
      "min-time": 0
    },
    "severely-errored-seconds": 0,
    "unavailable-seconds": 0
  }
},
"state": {
  "admin-state": "DISABLED",
  "description": "Coherent Logical Channel",
  "index": 30000,
  "link-state": "DOWN",
  "logical-channel-type": "openconfig-transport-types:PROT_OTN",
  "loopback-mode": "NONE"
```

```
        }
      },
```

OC-terminal-device client MDT data is as follows:

```
      "30002": {
        "ethernet": {
          "state": {
            "in-crc-errors": 0,
            "in-fragment-frames": 0,
            "in-jabber-frames": 0,
            "in-mac-pause-frames": 0,
            "in-oversize-frames": 0,
            "in-pcs-bip-errors": 0,
            "in-pcs-errored-seconds": 0,
            "in-pcs-severely-errored-seconds": 0,
            "in-pcs-unavailable-seconds": 0,
            "out-mac-pause-frames": 0
          }
        },
        "ingress": {
          "state": {
            "transceiver": "Optics0_0_0_2"
          }
        },
        "logical-channel-assignments": {
          "assignment": {
            "1": {
              "state": {
                "allocation": 100,
                "assignment-type": "LOGICAL_CHANNEL",
                "description": "ETH to ODU4 assignemnt",
                "index": 1,
                "logical-channel": 30020
              }
            }
          }
        },
        "state": {
          "admin-state": "ENABLED",
          "description": "ETH Logical Channel",
          "index": 30002,
          "link-state": "UP",
          "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
          "loopback-mode": "NONE",
          "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
          "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
        }
      },
```

OC-transceiver data for trunk is as follows:

```
"openconfig-platform": {
  "components": {
    "component": {
      "0/0-Optics0/0/0/0": {
        "openconfig-platform-transceiver:transceiver": {
          "physical-channels": {
            "channel": {
              "1": {
                "state": {
                  "index": 1,
                  "input-power": {
                    "avg": -1.68,
                    "instant": -1.69,
```

```
                "interval": 30000000000,
                "max": -1.64,
                "max-time": 1574223676071500740,
                "min": -1.71,
                "min-time": 1574223669071573554
              },
              "laser-bias-current": {
                "avg": 0,
                "instant": 0,
                "interval": 30000000000,
                "max": 0,
                "max-time": 1574223660072966799,
                "min": 0,
                "min-time": 1574223660072966799
              },
              "output-frequency": 193100000,
              "output-power": {
                "avg": -1.49,
                "instant": -1.5,
                "interval": 30000000000,
                "max": -1.45,
                "max-time": 1574223664071514654,
                "min": -1.54,
                "min-time": 1574223686071572161
              }
            }
          }
        }
      },
      "state": {
        "fault-condition": false,
        "fec-corrected-bits": 3886,
        "fec-uncorrectable-words": 0,
        "present": "PRESENT"
      }
    }
```

OC-transceiver data for client port is as follows:

```
"0/0-Optics0/0/0/2": {
  "openconfig-platform-transceiver:transceiver": {
    "physical-channels": {
      "channel": {
        "1": {
          "state": {
            "index": 1,
            "input-power": {
              "avg": -40,
              "instant": -40,
              "interval": 30000000000,
              "max": -40,
              "max-time": 1574223660072988517,
              "min": -40,
              "min-time": 1574223660072988517
            },
            "laser-bias-current": {
              "avg": 42.01,
              "instant": 42.04,
              "interval": 30000000000,
              "max": 42.05,
              "max-time": 1574223662071529895,
              "min": 41.96,
              "min-time": 1574223677071538621
            },
            "output-frequency": 231399800,
```

```
      "output-power": {
        "avg": 2.89,
        "instant": 2.91,
        "interval": 30000000000,
        "max": 2.92,
        "max-time": 1574223665071507015,
        "min": 2.87,
        "min-time": 1574223667071599738
      }
    }
  },
  "2": {
    "state": {
      "index": 2,
      "input-power": {
        "avg": -40,
        "instant": -40,
        "interval": 30000000000,
        "max": -40,
        "max-time": 1574223660072993271,
        "min": -40,
        "min-time": 1574223660072993271
      },
      "laser-bias-current": {
        "avg": 41.73,
        "instant": 41.74,
        "interval": 30000000000,
        "max": 41.76,
        "max-time": 1574223661071554206,
        "min": 41.66,
        "min-time": 1574223666071512088
      },
      "output-frequency": 230598900,
      "output-power": {
        "avg": 2.94,
        "instant": 2.94,
        "interval": 30000000000,
        "max": 2.97,
        "max-time": 1574223670071513829,
        "min": 2.92,
        "min-time": 1574223663071555469
      }
    }
  },
  "3": {
    "state": {
      "index": 3,
      "input-power": {
        "avg": -40,
        "instant": -40,
        "interval": 30000000000,
        "max": -40,
        "max-time": 1574223660072997439,
        "min": -40,
        "min-time": 1574223660072997439
      },
      "laser-bias-current": {
        "avg": 42.07,
        "instant": 42.07,
        "interval": 30000000000,
        "max": 42.14,
        "max-time": 1574223689071519875,
        "min": 42.02,
        "min-time": 1574223680071519773
```

```
            },
            "output-frequency": 229798200,
            "output-power": {
              "avg": 3.11,
              "instant": 3.12,
              "interval": 30000000000,
              "max": 3.14,
              "max-time": 1574223671071530318,
              "min": 3.08,
              "min-time": 1574223677071549246
            }
          }
        },
        "4": {
          "state": {
            "index": 4,
            "input-power": {
              "avg": -40,
              "instant": -40,
              "interval": 30000000000,
              "max": -40,
              "max-time": 1574223660073001092,
              "min": -40,
              "min-time": 1574223660073001092
            },
            "laser-bias-current": {
              "avg": 41.88,
              "instant": 41.86,
              "interval": 30000000000,
              "max": 41.95,
              "max-time": 1574223669071611563,
              "min": 41.83,
              "min-time": 1574223664071551968
            },
            "output-frequency": 230255300,
            "output-power": {
              "avg": 3.32,
              "instant": 3.29,
              "interval": 30000000000,
              "max": 3.35,
              "max-time": 1574223687071561519,
              "min": 3.3,
              "min-time": 1574223664071551968
            }
          }
        }
      },
      "state": {
        "connector-type": "openconfig-transport-types:LC_CONNECTOR",
        "date-code": "190807",
        "fault-condition": false,
        "form-factor": "openconfig-transport-types:QSFP28",
        "otn-compliance-code": "openconfig-transport-types:OTN_NOT_SET",
        "present": "PRESENT",
        "serial-no": "FNS23320KEK",
        "sonet-sdh-compliance-code": "openconfig-transport-types:SONET_NOT_SET",
        "vendor": "CISCO-FINISAR",
        "vendor-part": "FTLC1151SDPL-C1",
        "vendor-rev": "B"
      }
    }
  },
```

Note Github Link for OpenConfig-terminal-device :
https://github.com/openconfig/public/blob/master/release/models/optical-transport/openconfig-terminal-device.yang

Github link for OpenConfig-transceiver:
https://github.com/openconfig/public/blob/master/release/models/platform/openconfig-platform-transceiver.yang


```
{
  "Calaeum.Caluem_@123^": {
    "openconfig": {
      "terminal-device": {
        "logical-channels": {
          "channel": {
            "30000": {
              "logical-channel-assignments": {
                "assignment": {
                  "1": {
                    "state": {
                      "allocation": 500,
                      "assignment-type": "OPTICAL_CHANNEL",
                      "description": "Coherent to optical assignemnt",
                      "index": 1,
                      "optical-channel": "0_0-OpticalChannel0_0_0_0"
                    }
                  }
                }
              },
              "otn": {
                "state": {
                  "background-block-errors": 0,
                  "errored-seconds": 0,
                  "esnr": {
                    "avg": 0,
                    "instant": 0,
                    "interval": 30000000000,
                    "max": 0,
                    "max-time": 1573537980083123944,
                    "min": 0,
                    "min-time": 1573537980083123944
                  },
                  "fec-corrected-bits": 0,
                  "fec-uncorrectable-words": 0,
                  "post-fec-ber": {
                    "avg": 0,
                    "instant": 0,
                    "interval": 30000000000,
                    "max": 0,
                    "max-time": 0,
                    "min": 0,
                    "min-time": 0
                  },
                  "pre-fec-ber": {
                    "avg": 0,
                    "instant": 0,
                    "interval": 30000000000,
                    "max": 0,
                    "max-time": 0,
                    "min": 0,
                    "min-time": 0
                  },
                  "q-value": {
```

```
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
          "max-time": 0,
          "min": 0,
          "min-time": 0
        },
        "severely-errored-seconds": 0,
        "unavailable-seconds": 0
      }
    },
    "state": {
      "admin-state": "DISABLED",
      "description": "Coherent Logical Channel",
      "index": 30000,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE"
    }
  },
  "30001": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 500,
            "assignment-type": "OPTICAL_CHANNEL",
            "description": "Coherent to optical assignemnt",
            "index": 1,
            "optical-channel": "0_0-OpticalChannel0_0_0_1"
          }
        }
      }
    },
    "otn": {
      "state": {
        "background-block-errors": 0,
        "errored-seconds": 0,
        "esnr": {
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
          "max-time": 1573537980083239722,
          "min": 0,
          "min-time": 1573537980083239722
        },
        "fec-corrected-bits": 0,
        "fec-uncorrectable-words": 0,
        "post-fec-ber": {
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
          "max-time": 1573537980083377735,
          "min": 0,
          "min-time": 1573537980083377735
        },
        "pre-fec-ber": {
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
```

```
          "max-time": 1573537980083377735,
          "min": 0,
          "min-time": 1573537980083377735
        },
        "q-value": {
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
          "max-time": 1573537980083377735,
          "min": 0,
          "min-time": 1573537980083377735
        },
        "severely-errored-seconds": 0,
        "unavailable-seconds": 30
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "Coherent Logical Channel",
      "index": 30001,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE"
    }
  },
  "30002": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 0,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_2"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30020
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30002,
      "link-state": "UP",
```

```
    "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
    "loopback-mode": "NONE",
    "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
    "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
  }
},
"30003": {
  "ethernet": {
    "state": {
      "in-crc-errors": 0,
      "in-fragment-frames": 0,
      "in-jabber-frames": 0,
      "in-mac-pause-frames": 0,
      "in-oversize-frames": 0,
      "in-pcs-bip-errors": 0,
      "in-pcs-errored-seconds": 0,
      "in-pcs-severely-errored-seconds": 0,
      "in-pcs-unavailable-seconds": 30,
      "out-mac-pause-frames": 0
    }
  },
  "ingress": {
    "state": {
      "transceiver": "Optics0_0_0_3"
    }
  },
  "logical-channel-assignments": {
    "assignment": {
      "1": {
        "state": {
          "allocation": 100,
          "assignment-type": "LOGICAL_CHANNEL",
          "description": "ETH to ODU4 assignemnt",
          "index": 1,
          "logical-channel": 30021
        }
      }
    }
  },
  "state": {
    "admin-state": "ENABLED",
    "description": "ETH Logical Channel",
    "index": 30003,
    "link-state": "DOWN",
    "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
    "loopback-mode": "NONE",
    "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
    "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
  }
},
"30004": {
  "ethernet": {
    "state": {
      "in-crc-errors": 0,
      "in-fragment-frames": 0,
      "in-jabber-frames": 0,
      "in-mac-pause-frames": 0,
      "in-oversize-frames": 0,
      "in-pcs-bip-errors": 0,
      "in-pcs-errored-seconds": 0,
      "in-pcs-severely-errored-seconds": 0,
      "in-pcs-unavailable-seconds": 30,
      "out-mac-pause-frames": 0
    }
```

```
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_4"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30022
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30004,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30005": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_5"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30023
          }
        }
      }
    },
    "state": {
```

```
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30005,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30006": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_6"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30024
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30006,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30007": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
```

```
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_8"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30025
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30007,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30008": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_9"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30026
          }
```

```
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30008,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30009": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_10"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30027
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30009,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30010": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
```

```
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 30,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_11"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30028
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30010,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30011": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 0,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_0_0_12"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
```

```
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 30029
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 30011,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "30020": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30000
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30020,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30021": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30000
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30021,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
```

```
    }
  },
  "30022": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30000
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30022,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30023": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30000
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30023,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30024": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30000
          }
        }
      }
    },
```

```
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30024,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30025": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30001
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30025,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30026": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "Assignment: ODU4 to Coherent",
            "index": 1,
            "logical-channel": 30001
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ODU4 logical channel",
      "index": 30026,
      "link-state": "DOWN",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4"
    }
  },
  "30027": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
```

```
          "assignment-type": "LOGICAL_CHANNEL",
          "description": "Assignment: ODU4 to Coherent",
          "index": 1,
          "logical-channel": 30001
        }
      }
    }
  },
  "state": {
    "admin-state": "ENABLED",
    "description": "ODU4 logical channel",
    "index": 30027,
    "link-state": "DOWN",
    "logical-channel-type": "openconfig-transport-types:PROT_OTN",
    "loopback-mode": "NONE",
    "trib-protocol": "openconfig-transport-types:PROT_ODU4"
  }
},
"30028": {
  "logical-channel-assignments": {
    "assignment": {
      "1": {
        "state": {
          "allocation": 100,
          "assignment-type": "LOGICAL_CHANNEL",
          "description": "Assignment: ODU4 to Coherent",
          "index": 1,
          "logical-channel": 30001
        }
      }
    }
  },
  "state": {
    "admin-state": "ENABLED",
    "description": "ODU4 logical channel",
    "index": 30028,
    "link-state": "DOWN",
    "logical-channel-type": "openconfig-transport-types:PROT_OTN",
    "loopback-mode": "NONE",
    "trib-protocol": "openconfig-transport-types:PROT_ODU4"
  }
},
"30029": {
  "logical-channel-assignments": {
    "assignment": {
      "1": {
        "state": {
          "allocation": 100,
          "assignment-type": "LOGICAL_CHANNEL",
          "description": "Assignment: ODU4 to Coherent",
          "index": 1,
          "logical-channel": 30001
        }
      }
    }
  },
  "state": {
    "admin-state": "ENABLED",
    "description": "ODU4 logical channel",
    "index": 30029,
    "link-state": "DOWN",
    "logical-channel-type": "openconfig-transport-types:PROT_OTN",
    "loopback-mode": "NONE",
    "trib-protocol": "openconfig-transport-types:PROT_ODU4"
```

```
    }
  },
  "40000": {
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 600,
            "assignment-type": "OPTICAL_CHANNEL",
            "description": "Coherent to optical assignemnt",
            "index": 1,
            "optical-channel": "0_1-OpticalChannel0_1_0_0"
          }
        }
      }
    },
    "otn": {
      "state": {
        "background-block-errors": 0,
        "errored-seconds": 0,
        "esnr": {
          "avg": 20.25,
          "instant": 20.3,
          "interval": 30000000000,
          "max": 20.3,
          "max-time": 1573537980480579044,
          "min": 20.1,
          "min-time": 1573537987470421480
        },
        "fec-corrected-bits": 17436506376,
        "fec-uncorrectable-words": 0,
        "post-fec-ber": {
          "avg": 0,
          "instant": 0,
          "interval": 30000000000,
          "max": 0,
          "max-time": 1573537980480852141,
          "min": 0,
          "min-time": 1573537980480852141
        },
        "pre-fec-ber": {
          "avg": 0.00671,
          "instant": 0.0068,
          "interval": 30000000000,
          "max": 0.00738,
          "max-time": 1573537994470526801,
          "min": 0.00655,
          "min-time": 1573537995470571723
        },
        "q-value": {
          "avg": 7.8,
          "instant": 0.078,
          "interval": 30000000000,
          "max": 7.8,
          "max-time": 1573537980480852141,
          "min": 7.8,
          "min-time": 1573537980480852141
        },
        "severely-errored-seconds": 0,
        "unavailable-seconds": 0
      }
    },
    "state": {
      "admin-state": "ENABLED",
```

```
    "description": "Coherent Logical Channel",
    "index": 40000,
    "link-state": "UP",
    "logical-channel-type": "openconfig-transport-types:PROT_OTN",
    "loopback-mode": "NONE"
  }
},
"40001": {
  "logical-channel-assignments": {
    "assignment": {
      "1": {
        "state": {
          "allocation": 600,
          "assignment-type": "OPTICAL_CHANNEL",
          "description": "Coherent to optical assignemnt",
          "index": 1,
          "optical-channel": "0_1-OpticalChannel0_1_0_1"
        }
      }
    }
  },
  "otn": {
    "state": {
      "background-block-errors": 0,
      "errored-seconds": 0,
      "esnr": {
        "avg": 20.02,
        "instant": 20.1,
        "interval": 30000000000,
        "max": 20.1,
        "max-time": 1573537980480723522,
        "min": 19.9,
        "min-time": 1573537984470478041
      },
      "fec-corrected-bits": 50377421964,
      "fec-uncorrectable-words": 0,
      "post-fec-ber": {
        "avg": 0,
        "instant": 0,
        "interval": 30000000000,
        "max": 0,
        "max-time": 1573537980481194932,
        "min": 0,
        "min-time": 1573537980481194932
      },
      "pre-fec-ber": {
        "avg": 0.00825,
        "instant": 0.0085,
        "interval": 30000000000,
        "max": 0.00897,
        "max-time": 1573537994470555826,
        "min": 0.008,
        "min-time": 1573538007470595227
      },
      "q-value": {
        "avg": 7.52,
        "instant": 0.075,
        "interval": 30000000000,
        "max": 7.6,
        "max-time": 1573537980481194932,
        "min": 7.5,
        "min-time": 1573537980481194932
      },
      "severely-errored-seconds": 0,
```

```
        "unavailable-seconds": 0
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "Coherent Logical Channel",
      "index": 40001,
      "link-state": "UP",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN",
      "loopback-mode": "NONE"
    }
  },
  "40002": {
    "ethernet": {
      "state": {
        "in-crc-errors": 0,
        "in-fragment-frames": 0,
        "in-jabber-frames": 0,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 0,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_1_0_2"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 40020
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 40002,
      "link-state": "UP",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "40003": {
    "ethernet": {
      "state": {
        "in-crc-errors": 18446744073709551580,
        "in-fragment-frames": 0,
        "in-jabber-frames": 3,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
```

```
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 0,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_1_0_3"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
            "logical-channel": 40021
          }
        }
      }
    },
    "state": {
      "admin-state": "ENABLED",
      "description": "ETH Logical Channel",
      "index": 40003,
      "link-state": "UP",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
      "loopback-mode": "NONE",
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
    }
  },
  "40004": {
    "ethernet": {
      "state": {
        "in-crc-errors": 18446744073709551531,
        "in-fragment-frames": 0,
        "in-jabber-frames": 7,
        "in-mac-pause-frames": 0,
        "in-oversize-frames": 0,
        "in-pcs-bip-errors": 0,
        "in-pcs-errored-seconds": 0,
        "in-pcs-severely-errored-seconds": 0,
        "in-pcs-unavailable-seconds": 0,
        "out-mac-pause-frames": 0
      }
    },
    "ingress": {
      "state": {
        "transceiver": "Optics0_1_0_4"
      }
    },
    "logical-channel-assignments": {
      "assignment": {
        "1": {
          "state": {
            "allocation": 100,
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "ETH to ODU4 assignemnt",
            "index": 1,
```

```
          "logical-channel": 40022
        }
      }
    }
  },
  "state": {
    "admin-state": "ENABLED",
    "description": "ETH Logical Channel",
    "index": 40004,
    "link-state": "UP",
    "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET",
    "loopback-mode": "NONE",
    "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
    "trib-protocol": "openconfig-transport-types:PROT_100G_MLG"
  }
},
"40005": {
  "ethernet": {
    "state": {
      "in-crc-errors": 18446744073709551581,
      "in-fragment-frames": 0,
      "in-jabber-frames": 3,
      "in-mac-pause-frames": 0,
      "in-oversize-frames": 0,
      "in-pcs-bip-errors": 0,
      "in-pcs-errored-seconds": 0,
      "in-pcs-severely-errored-seconds": 0,
      "in-pcs-unavailable-seconds": 0,
      "out-mac-pause-frames": 0
    }
  },
  "ingress": {
    "state": {
      "transceiver": "Optics0_1_0_5"
    }
  },
  "logical-channel-assignments": {
    "assignment": {
      "1": {
        "state": {
          "allocation": 100,
          "assignment-type": "LOGICAL_CHANNEL",
          "description": "ETH to ODU4 assignemnt",
          "index": 1,
          "logical-channel": 40023
        }
      }
    }
  },
```


Ethernet Stats Addition for OpenConfig

The PCS performance monitoring counter details are mentioned in the following table.

PCS Counter	Influencing Alarm	Influencing Counter
PCS-ES	None	BIP>0 or FRM-ERR>0 or BAD-SH>0 And does not meet SES condition.
PCS-SES	SIGLOSS or SYNCLOSS, or LF	BIP>15% (Based on G.8201)
PCS-UAS	10 seconds of consecutive SES	10 seconds of consecutive SES
PCS-ES-FE	None	None
PCS-SES-FE	RF	None
PCS-UAS-FE	10 seconds of consecutive SES	10 seconds of consecutive SES

The MDT output is as follows:

```
"20002": {
              "ethernet": {
                "state": {
                  "in-crc-errors": 0,
                  "in-fragment-frames": 0,
                  "in-jabber-frames": 0,
                  "in-mac-pause-frames": 0,
                  "in-oversize-frames": 0,
                  "in-pcs-bip-errors": 0,
                  "in-pcs-errored-seconds": 0,
                  "in-pcs-severely-errored-seconds": 0,
                  "in-pcs-unavailable-seconds": 0,
                  "out-mac-pause-frames": 0
```

CLI for Ethernet PCS Stats

```
P/0/RP0/CPU0:BH1_P2A4#show controllers hundredGigECtrlr 0/1/0/2 pm current 15-min pcs
Thu Jan 30 11:28:16.370 UTC

Ethernet PCS in the current interval [11:15:00 - 11:28:16 Thu Jan 30 2020]

Ethernet PCS current bucket type : Valid
BIP[00]                    : 0                          Threshold : 0
   TCA(enable) : NO
 BIP[01]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[02]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[03]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[04]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[05]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[06]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[07]                   : 0                           Threshold : 0
    TCA(enable) : NO
 BIP[08]                   : 0                           Threshold : 0
```

```
   TCA(enable) : NO
BIP[09]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[10]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[11]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[12]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[13]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[14]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[15]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[16]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[17]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[18]                    : 0                          Threshold : 0
   TCA(enable) : NO
BIP[19]                    : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[00]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[01]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[02]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[03]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[04]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[05]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[06]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[07]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[08]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[09]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[10]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[11]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[12]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[13]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[14]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[15]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[16]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[17]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[18]                : 0                          Threshold : 0
   TCA(enable) : NO
FRM-ERR[19]                : 0                          Threshold : 0
   TCA(enable) : NO
BAD-SH[00]                 : 0                          Threshold : 0
```

```
   TCA(enable) : NO
BAD-SH[01]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[02]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[03]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[04]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[05]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[06]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[07]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[08]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[09]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[10]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[11]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[12]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[13]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[14]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[15]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[16]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[17]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[18]                : 0                       Threshold : 0
   TCA(enable) : NO
BAD-SH[19]                : 0                       Threshold : 0
   TCA(enable) : NO
ES                        : 0                       Threshold : 0
   TCA(enable) : NO
SES                       : 0                       Threshold : 0
   TCA(enable) : NO
UAS                       : 796                     Threshold : 0
   TCA(enable) : NO
ES-FE                     : 0                       Threshold : 0
   TCA(enable) : NO
SES-FE                    : 0                       Threshold : 0
   TCA(enable) : NO
UAS-FE                    : 0                       Threshold : 0
   TCA(enable) : NO
```

Configure LLDP on Management Port

Table 11: Feature History

Feature Name	Release	Description
OC (Open Configuration) Support for LLDP (Link Layer Discovery Protocol) on Management Port	Cisco IOS XR Release 7.3.1	The OC support for configuring LLDP on a management port is available. This feature enables you to perform the configuration using scripts, which is less time-consuming. Also, the Open Configuration model supports the use of vendor-neutral data models to configure and manage the network.

Step 1 You can use the following script to configure LLDP on management port.

```
"openconfig-lldp:lldp": {
"interfaces": {
  "interface": [
   {
    "name": "MgmtEth0/RP0/CPU0/0",
    "config": {
     "name": "MgmtEth0/RP0/CPU0/0",
     "enabled": true
    }
   },
   {
    "name": "MgmtEth0/RP0/CPU0/1",
    "config": {
     "name": "MgmtEth0/RP0/CPU0/1",
     "enabled": true
    }
   },
   {
    "name": "MgmtEth0/RP0/CPU0/2",
    "config": {
     "name": "MgmtEth0/RP0/CPU0/2",
     "enabled": true
    }
   }
  ]
}
}
```

Step 2 You can get the operational data through GNMI

```
"MgmtEth0/RP0/CPU0/1": {
                          "neighbors": {
                             "neighbor": {
                                 "SW-VEGA-CORBU-C4#Gi1/0/13": {
                                      "capabilities": {
```

```
                                    "capability": {
                                        "openconfig-lldp-types:MAC_BRIDGE": {
                                             "state": {
                                                 "enabled": true,
                                                 "name": "openconfig-lldp-types:MAC_BRIDGE"
                                             }
                                         },
                                         "openconfig-lldp-types:ROUTER": {
                                             "state": {
                                                 "enabled": false,
                                                 "name": "openconfig-lldp-types:ROUTER"
                                             }
                                         }
                                     }
                                },
                                 "custom-tlvs": {
                                     "tlv": {
                                         "32962": {
                                             "1": {
                                                "295": {
                                                     "state": {
                                                         "oui": "32962",
                                                        "oui-subtype": "1",
                                                         "type": 127,
                                                         "value": "Aa8="
                                                     }
                                                 }
                                             }
                                            },
                                            "4623": {
                                                 "1": {
                                                    "295": {
                                                        "state": {
                                                            "oui": "4623",
                                                             "oui-subtype": "1",
                                                             "type": 127,
                                                             "value": "A2wBAB4="
                                                         }
                                                     }
                                                 }
                                             }
                                         }
                                     },
                                     "state": {
                                        "chassis-id": "6899.cd9f.f480",
                                         "chassis-id-type": "MAC_ADDRESS",
                                         "id": "SW-VEGA-CORBU-C4",
                                         "management-address": "4.31.25.25",
                                         "management-address-type": "ipv4",
                                         "port-description": "GigabitEthernet1/0/13",
                                         "port-id": "Gi1/0/13",
                                         "port-id-type": "INTERFACE_NAME",
                                         "system-description": "Cisco IOS Software, Catalyst L3
Switch Software (CAT3K_CAA-UNIVERSALK9-M), Version 15.0(1)EX3, RELEASE SOFTWARE (fc2)\nTechnical
Support: http://www.cisco.com/techsupport\nCopyright (c) 1986-2013 by Cisco Systems, Inc.\nCompiled
 Mon 23-Sep-13 18:24 by prod_r",
                                         "system-name": "SW-VEGA-CORBU-C4"
                                     }
                                 }
                             }
                         },
                         "state": {
                             "enabled": true,
                             "name": "MgmtEth0/RP0/CPU0/1"
```

```
        }
    },
```

OpenConfig Terminal Device Revision

Table 12: Feature History

Feature Name	Release	Description
OC (Open Configuration) Terminal Device Revision	Cisco IOS XR Release 7.3.1	The Open Configuration terminal device revision to 1.7.2 allows you to provide LLDP support on the client optics. This feature allows you to learn LLDP neighbors and the topology of the devices for Operations, Administration, and Maintenance (OAM) purposes.

LLDP Support on Client Optics

The client-side LLDP is enabled by default. The LLDP state data is collected over gNMI telemetry.

Limitations

- There is no support on configuration, since LLDP is enabled by default.
- There is no support for LLDP counters.
- There is no support for leaf age and last update in LLDP neighbor discovery.

Sample gNMI telemetry output for LLDP:

```
{
  "openconfig-terminal-device": {
    "terminal-device": {
      "logical-channels": {
        "channel": {
          "10005": {
            "ethernet": {
              "lldp": {
                "neighbors": {
                  "neighbor": {
                    "nncs5500_node1#HundredGigE0/0/0/30": {
```

```
                    "state": {
                      "chassis-id": "008a.96cd.34df",
                      "chassis-id-type": "MAC_ADDRESS",
                      "id": "nncs5500_node1#HundredGigE0/0/0/30",
                      "management-address": "10.127.60.23",
                      "management-address-type": "ipv4",
                      "port-id": "HundredGigE0/0/0/30",
                      "port-id-type": "INTERFACE_NAME",
                      "system-description": " 7.2.1.36I, NCS-5500",
                      "system-name": "nncs5500_node1"
                    }
                  }
                }
              },
               "state": {
                 "enabled": true,
                 "snooping": true
               }
             }
           }
         }
       }
     }
   }
}
```

Open Configuration Model for Client FEC

Before you begin

Table 13: Feature History

Feature Name	Release	Description
OC (Open Configuration) Model for Client FEC and Laser Squelch	Cisco IOS XR Release 7.3.1	The OC model for configuring client FEC and Laser Squelch is available. This feature enables you to perform the configuration using scripts, which is less time-consuming. Also, the Open Configuration model supports the use of vendor-neutral data models to configure and manage the network.

Step 1 You can enable FEC (Forward Error Correction) on clients using the following scripts:

```
"openconfig-platform:components": {
 "component": [
  {
   "name": "0/0-Optics0/0/0/2",
   "config": {
    "name": "0/0-Optics0/0/0/2"
   },
   "openconfig-platform-transceiver:transceiver": {
    "config": {
     "fec-mode": "openconfig-platform-types:FEC_ENABLED"
    }
   }
  }
```

Step 2 You can get operational data using GNMI.

```
"state": {
    "connector-type": "openconfig-transport-types:LC_CONNECTOR",
    "date-code": "2019-08-05T00:00:00Z+00:00",
    "fault-condition": false,
    "fec-mode": "openconfig-platform-types:FEC_ENABLED",
    "fec-uncorrectable-words": 0,
    "form-factor": "openconfig-transport-types:QSFP28",
    "otn-compliance-code": "openconfig-transport-types:OTN_UNDEFINED",
    "present": "PRESENT",
    "serial-no": "INL23321878",
    "sonet-sdh-compliance-code": "openconfig-transport-types:SONET_UNDEFINED",
    "vendor": "CISCO-INNOLIGHT",
    "vendor-part": "10-3220-02",
    "vendor-rev": "1C"
}
```

Configure Laser Squelch

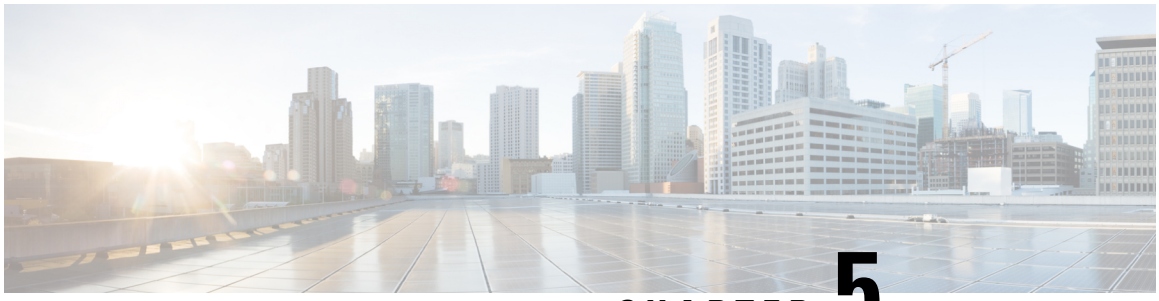
Step 1 You can enable laser squelching using the following scripts:

```
"openconfig-terminal-device:terminal-device": {
 "logical-channels": {
  "channel": [
   {
    "index": 30002,
    "config": {
     "index": 30002,
     "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
     "admin-state": "ENABLED",
     "description": "ETH Logical Channel
                             ",
     "loopback-mode": "NONE",
     "trib-protocol": "openconfig-transport-types:PROT_100G_MLG",
     "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET"
    },
    "ethernet": {
     "config": {
      "als-delay": 3000,
      "client-als": "LASER_SHUTDOWN"
     }
    },
    "ingress": {
     "config": {
      "transceiver": "0/0-Optics0/0/0/2"
     }
    },
    "logical-channel-assignments": {
     "assignment": [
      {
       "index": 1,
       "config": {
        "index": 1,
        "allocation": "100",
        "assignment-type": "LOGICAL_CHANNEL",
        "description": "ETH to ODU4 assignemnt
                                ",
        "logical-channel": 30020
       }
      }
     ]
    }
   },
```

Step 2 You can get operational data using GNMI.

```
"state": {
                                  "als-delay": 5,
                                   "client-als": "LASER_SHUTDOWN",
                                   "in-crc-errors": 18446744073709551473,
                                   "in-fragment-frames": 0,
                                  "in-jabber-frames": 2,
                                   "in-mac-pause-frames": 0,
                                   "in-oversize-frames": 0,
                                   "in-pcs-bip-errors": 0,
                                   "in-pcs-errored-seconds": 0,
                                  "in-pcs-severely-errored-seconds": 0,
```

```
                "in-pcs-unavailable-seconds": 10,
                "out-mac-pause-frames": 0
            }
        },
```



CHAPTER 5

Configure 400G TXP and MXP Data Paths

This chapter describes the required configurations for configuring the 400G TXP and MXP datapaths using CFP2 DCO.

From Release 7.5.1 onwards the OTN-XP card supports Openconfig configuration for Muxponder Mode 4x100G MXP and 400G TXP on ports 12 and 13 for CFP2 DCO.

Table 14: Feature History

Feature Name	Release Information	Feature Description
OC Support for 400G TXP/MXP	Cisco IOS XR Release 7.5.1	This feature allows you to configure the 400G TXP and 400G MXP using CFP2 DCO. On the OTN-XP card, you can configure OC datapath on a single 400GE or 4x100G payload that is received over the client port as a 400G signal over DWDM on the line side. The card improves efficiency, performance, and flexibility for customer networks allowing 400GE or 4x100G client transport over 400G WDM wavelength.

Slices and Port Mapping on 400G TXP/MXP

The 400G MXP and 400G TXP can be configured with the same LC mode. Following are the supported operating modes to configure 400G TXP and 400G MXP datapaths:

- 2x400G-TXP
- 2x4x100G-MXP

- Mix of MXP/TXP

The following tables shows port maping for slices 0 and 1.

Table 15: 400G-TXP: Ports Mapping for Slice 0

Client Port	Port 10
Trunk Port	Port 12
Client Payload	400GE
Trunk Rate	400G (OTUC4)
Client Optics	QDD-400G-DR4-S, QDD-400G-FR-S, QDD-400G-LR8
Trunk Optics	ONS-CFP2D-400G-C

Table 16: 400G-TXP: Ports Mapping for Slice 1

Client Port	Port 8
Trunk Port	Port 13
Client Payload	400GE
Trunk Rate	400G (OTUC4)
Client Optics	QDD-400G-DR4-S, QDD-400G-FR4-S,QDD-400G-LR8
Trunk Optics	ONS-CFP2D-400G-C

Table 17: 4x100G-MXP: Ports Mapping for Slice 0

Client Port	Port 1,6,7 and 10
Trunk Port	Port 12
Client Payload	100GE and OTU4
Trunk Rate	400G (OTUC4)
Client Optics	ONS-QSFP28-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S
Trunk Optics	ONS-CFP2D-400G-C

Table 18: 4x100G-MXP: Ports Mapping for Slice 1

Client Port	Port 0,4,5 and 8

Trunk Port	Port 13
Client Payload	100GE and OTU4
Trunk Rate	400G (OTUC4)
Client Optics	QDD-400G-DR4-S, QDD-400G-ZR-S
Trunk Optics	ONS-QSFP28-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S

Configuring 400G TXP/MXP Clients

The following table explains the different commands used for 100G, OTU4, and 400GE client ports.

Table 19: Configuration Details for 100G, OTU4, and 400GE Client Ports

Client Port	Logical Channel	Optical Channel	ODUCn	Coherent DSP	Optical Channel

100G	"index": 101,
	"rate-class":
	"cisco-otn-type:PHAE100",
	"description":
	"Client
	Logical
	Channel",
	"admin-state":
	"ENABLED",
	"loopback-mode":
	"NONE",
	"trib-protocol":
	"cisco-otn-type:ETH100GC",
	"logical-channel-type":
	"cisco-otn-type:ETHNP"
		"index": 201,
		"rate-class":
		"cisco-otn-type:PHAE100",
		"admin-state":
		"ENABLED",
		"loopback-mode":
		"NONE",
		"description":
		"Client
		Logical
		Channel",
		"trib-protocol":
		"cisco-otn-type:ETH100G",
		"logical-channel-type":
		"cisco-otn-type:RIOTN"
			"index": 112,
			"config": {
			"index": 112,
			"admin-state":
			"ENABLED",
			"description":
			"Trunk-side-ODUCn",
			"rate-class":
			"cisco-otn-type:PHAE400",
			"trib-protocol":
			"cisco-otn-type:ETH100G",
			"logical-channel-type":
			"cisco-otn-type:RIOTN"
				"index": 212,
				"config": {
				"index": 212,
				"admin-state":
				"ENABLED",
				"loopback-mode":
				"NONE",
				"description":
				"Coherent
				DSP",
				"rate-class":
				"cisco-otn-type:PHAE400",
				"logical-channel-type":
				"cisco-otn-type:RIOTN"
					"name":
					"0/1-OpticalChannel0/1/0/12",
					"cisco-ethernet-optical-channel":
					{
					"config": {
					"frequency":
					"193100000",
					"target-output-power":
					-700,
					"operational-mode":
					4178,
					"line-port":
					"0/1-Optics0/1/0/12"

OTU4	"index": 101,
	"rate-class":
	"cisco-otn-type:PHAE100",
	"description":
	"Client
	Logical
	Channel",
	"admin-state":
	"ENABLED",
	"loopback-mode":
	"NONE",
	"trib-protocol":
	"cisco-otn-type:RIOTU4",
	"logical-channel-type":
	"cisco-otn-type:RIOTN"
		"index": 201,
		"rate-class":
		"cisco-otn-type:PHAE100",
		"admin-state":
		"ENABLED",
		"loopback-mode":
		"NONE",
		"description":
		"Client
		Logical
		Channel",
		"trib-protocol":
		"cisco-otn-type:RIOTU4",
		"logical-channel-type":
		"cisco-otn-type:RIOTN"
			"index": 112,
			"config": {
			"index": 112,
			"admin-state":
			"ENABLED",
			"description":
			"Trunk-side-ODUCn",
			"rate-class":
			"cisco-otn-type:PHAE400",
			"trib-protocol":
			"cisco-otn-type:RIOTU4",
			"logical-channel-type":
			"cisco-otn-type:RIOTN"
				"index": 212,
				"config": {
				"index": 212,
				"admin-state":
				"ENABLED",
				"loopback-mode":
				"NONE",
				"description":
				"Coherent
				DSP",
				"rate-class":
				"cisco-otn-type:PHAE400",
				"logical-channel-type":
				"cisco-otn-type:RIOTN"
					"name":
					"0/1-OpticalChannel0/1/0/12",
					"cisco-ethernet-optical-channel":
					{
					"config": {
					"frequency":
					"193100000",
					"target-output-power":
					-700,
					"operational-mode":
					4178,
					"line-port":
					"0/1-Optics0/1/0/12"

Client Port	Logical Channel	Optical Channel	ODUCn	Coherent DSP	Optical Channel

400GE	"index": 101,
	"rate-class":
	"openconfig-transport-types:TRIB_RATE_400G",
	"description":
	"Client
	Logical
	Channel",
	"admin-state":
	"ENABLED",
	"loopback-mode":
	"NONE",
	"trib-protocol":
	
	"openconfig-transport-types:PROT_400GE",
	"logical-channel-type":
	
	"openconfig-transport-types:PROT_HIGIG"
		"index": 201,
		"rate-class":
		"openconfig-transport-types:TRIB_RATE_400G",
		"admin-state":
		"ENABLED",
		"loopback-mode":
		"NONE",
		"description":
		"Client
		Logical
		Channel",
		"trib-protocol":
		
		"openconfig-transport-types:PROT_100GE_R4",
		"logical-channel-type":
		
		"openconfig-transport-types:PROT_OTN"
			"index": 112,
			"config": {
			"index": 112,
			"admin-state":
			"ENABLED",
			"description":
			
			"Trunk-side-ODUCn",
			"rate-class":
			"openconfig-transport-types:TRIB_RATE_400G",
			"trib-protocol":
			
			"openconfig-transport-types:PROT_ODUCN",
			"logical-channel-type":
			
			"openconfig-transport-types:PROT_OTN"
				"index": 212,
				"config": {
				"index": 212,
				"admin-state":
				"ENABLED",
				"loopback-mode":
				"NONE",
				"description":
				"Coherent
				DSP",
				"rate-class":
				"openconfig-transport-types:TRIB_RATE_400G",
				"logical-channel-type":
				
				"openconfig-transport-types:PROT_OTN"
					"name":
					"0/1-OpticalChannel0/1/0/12",
					"openconfig-optical-channel":
					{
					"config": {
					"frequency":
					"193100000",
					"target-output-power":
					-700,
					"operational-mode":
					4178,
					"line-port":
					"0/1-Optics0/1/0/12"

Sample Configuration

Configuring 400G TXP with CFP2 Single Slice

The following is a sample to configure the 400G TXP with CFP2 single slice:

```
{
  "openconfig-terminal-device:terminal-device": {
    "logical-channels": {
      "channel": [
        {
          "index": 112,
          "config": {
            "index": 112,
            "admin-state": "ENABLED",
            "description": "Trunk-side-ODUCn",
            "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
            "trib-protocol": "openconfig-transport-types:PROT_ODUCN",
            "logical-channel-type": "openconfig-transport-types:PROT_OTN"
          },
          "logical-channel-assignments": {
            "assignment": [
              {
                "index": 1,
                "config": {
                  "index": 1,
                  "allocation": "400",
                  "assignment-type": "LOGICAL_CHANNEL",
                  "description": "logical to Logical",
                  "logical-channel": 212
                }
              }
            ]
          }
        },
        {
          "index": 212,
          "config": {
            "index": 212,
            "admin-state": "ENABLED",
            "loopback-mode": "NONE",
```

```
      "description": "Coherent DSP",
      "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN"
    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "400",
            "assignment-type": "OPTICAL_CHANNEL",
            "description": "logical to optical",
            "optical-channel": "0/1-OpticalChannel0/1/0/12"
          }
        }
      ]
    }
  },
  {
    "index": 101,
    "config": {
      "index": 101,
      "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
      "description": "Client Logical Channel",
      "admin-state": "ENABLED",
      "loopback-mode": "NONE",
      "trib-protocol": "openconfig-transport-types:PROT_400GE",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET"
    },
    "ingress": {
      "config": {
        "transceiver": "0/1-Optics0/1/0/10"
      }
    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "400",
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "logical to logical assignemnt",
            "logical-channel": 201
          }
        }
      ]
    }
  },
  {
    "index": 201,
    "config": {
      "index": 201,
      "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
      "admin-state": "ENABLED",
      "loopback-mode": "NONE",
      "description": "Client Logical Channel",
      "trib-protocol": "openconfig-transport-types:PROT_ODUFLEX_CBR",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN"
    },
    "logical-channel-assignments": {
      "assignment": [
        {
```

```
                "index": 1,
                "config": {
                  "index": 1,
                  "allocation": "400",
                  "assignment-type": "LOGICAL_CHANNEL",
                  "description": "logical to Logical",
                  "logical-channel": 112
                }
              }
            ]
          }
        }
      ]
    }
  },
  "openconfig-platform:components": {
    "component": [
      {
        "name": "0/1-OpticalChannel0/1/0/12",
        "openconfig-terminal-device:optical-channel": {
          "config": {
            "frequency": "193100000",
            "target-output-power": -700,
            "operational-mode": 4178,
            "line-port": "0/1-Optics0/1/0/12"
          }
        }
      },
      {
        "name": "0/1",
        "properties": {
          "property": [
            {
              "name": "LCMODE",
              "config": {
                "name": "LCMODE",
                "value": "4x100g-MxP-400G-tXp"
              }
            }
          ]
        }
      }
    ]
  }
}
```


400G TXP/MXP Support

This topic lists the OC models, Alarm and Features supported by the 400G TXP/MXP datapaths:

Table 20: Supported OC Models

Model	Feature
openconfig-platform.yang	Inventory and LC Mode
openconfig-platform-transceiver.yang	Pluggable Inventory and Oper Data
openconfig-terminal-device.yang	Logical and Optical Channels – Datapath and OperData

Model	Feature
openconfig-interface.yang	Optical Interface Enable/Disable (shut/no-shut)

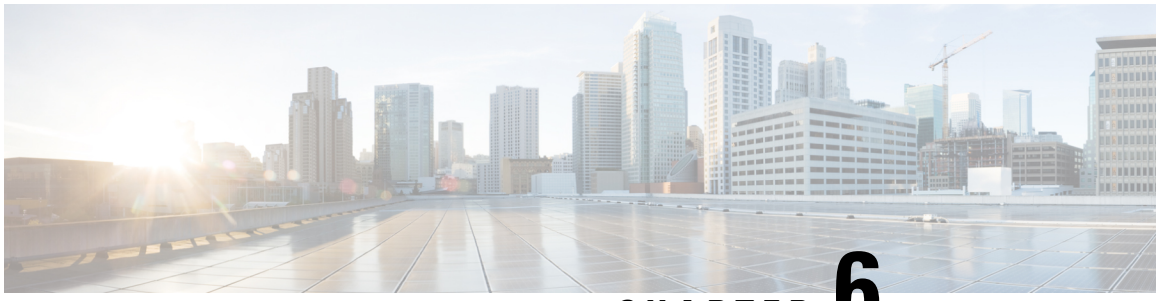
Table 21: Supported Alarm

Alarm
openconfig-system.yang (augmented with Openconfig-alarms)

Supported Features

- Client loopback support for 100G/OTU4/400G logical channel
- Trunk loopback support for Coherent DSP Optical channel
- Admin-state support for all Logical and Optical channel
- Laser Squelch and als-delay for ethernet client
- Client Fec
- LLDP

Note The 400G TXP/MXP does not support Trail Trace Identifier (TTI).



CHAPTER 6

OC Support for TXP and MXP Data Paths Using ZRP

This chapter describes open configuration details for 2x100GE-MXP-DD, 3x100GE-MXP-DD, 400GE-TXP-DD, and 4x100GE-MXP-DD datapaths using QDD ZRP pluggable.

Table 22: Feature History

Feature Name	Release Information	Feature Description
OC Support for TXP and MXP Data Paths using ZRP	Cisco IOS XR Release 7.7.1	From this release onwards, in NCS1K4-OTN-XP line cards, you can configure the 2x100GE-MXP-DD, 3x100GE-MXP-DD, 4x100GE-MXP-DD, and 400GE-TXP-DD card modes using ZR+ pluggable via OpenConfig. This enhancement improves efficiency, performance, and flexibility for customer networks allowing 2x100GE, 3x100GE, 400GE, and 4x100GE client transport over 400GE WDM wavelength.

Slices and Port Mapping

To configure the 400GE-TXP-DD and 4x100GE-MXP-DD lc modes, the following data path configurations are used.

LC Mode	Datapath
400GE-TXP-DD	2x400GE-TXP-DD
4x100GE-MXP-DD	You can take any combination of the following datapaths to configure the 4x100GE-MXP-DD lc mode, based on the requirement: • 2x100GE-MXP-DD • 3x100GE-MXP-DD • 4x100GE-MXP-DD

The following tables shows port maping for slices 0 and 1.

Table 23: 2x100GE-MXP-DD: Ports Mapping for Slice 0

Client Port	Port 7 and 10
Trunk Port	Port 11
Client Payload	100GE
Trunk Rate	200G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S,QSFP28-100G-AOC, QSFP28-100G-PSM4,QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Table 24: 2x100GE-MXP-DD: Ports Mapping for Slice 1

Client Port	Port 4 and 5
Trunk Port	Port 9
Client Payload	100GE
Trunk Rate	200G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S, QSFP28-100G-AOC, QSFP28-100G-PSM4,QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Table 25: 3x100GE-MXP-DD: Ports Mapping for Slice 0

Client Port	Port 1, 7, and 10

Trunk Port	Port 11
Client Payload	100GE
Trunk Rate	300G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S, QSFP28-100G-AOC, QSFP28-100G-PSM4, QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Table 26: 3x100GE-MXP-DD: Ports Mapping for Slice 1

Client Port	Port 8, 4, and 5
Trunk Port	Port 9
Client Payload	100GE
Trunk Rate	300G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S, QSFP28-100G-AOC, QSFP28-100G-PSM4,QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Table 27: 400GE-TXP-DD: Ports Mapping for Slice 0

Client Port	Port 10
Trunk Port	Port 11
Client Payload	400GE
Trunk Rate	400G
Client Optics	QDD-400G-FR4-S ,QDD-400G-DR4-S, and QDD-400G-LR8-S
Trunk Optics	QDD-400G-ZRP-S

Table 28: 400GE-TXP-DD: Ports Mapping for Slice 1

Client Port	Port 8
Trunk Port	Port 9
Client Payload	400GE
Trunk Rate	400G

Client Optics	QDD-400G-FR4-S,QDD-400G-DR4-S, QDD-400G-LR8-S
Trunk Optics	QDD-400G-ZRP-S

Table 29: 4x100GE-MXP-DD: Ports Mapping for Slice 0

Client Port	Port 1,6,7 and 10
Trunk Port	Port 11
Client Payload	100GE
Trunk Rate	400G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S, QSFP28-100G-AOC, QSFP28-100G-PSM4, and QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Table 30: 4x100GE-MXP-DD: Ports Mapping for Slice 1

Client Port	Port 0,4,5 and 8
Trunk Port	Port 9
Client Payload	100GE
Trunk Rate	400G
Client Optics	QSFP-100G-LR4, QSFP-100G-FR-S, QSFP-100G-SR4-S, QSFP-100G-CWDM4-S, QSFP-100G-LR4-S, QSFP28-100G-AOC, QSFP28-100G-PSM4, and QSFP28-100G-DR-S
Trunk Optics	QDD-400G-ZRP-S

Configuring Clients

The commands used for configuring the 100G and 400GE client ports are explained in the following table.

Table 31: Configuration Details for 100G and 400GE Client Ports:

Client Port	Logical Channel	Coherent DSP	Optical Channel	LC Mode
100G	index": 101, "rate-class": "openconfig-transport-types: TRIB_RATE_100G", "description": "Client Logical Channel", "admin-state": "ENABLED", "loopback-mode": "NONE", "trib-protocol": "openconfig-transport-types: PROT_100G_MLG", "logical-channel-type": "openconfig-transport-types: PROT_ETHERNET"	"index": 212, "config": {"index": 212, "admin-state": "ENABLED", "loopback-mode": "NONE", "description": "Coherent DSP", "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "logical-channel-type": "openconfig-transport-types: PROT_OTN"	"name": "0/1-OpticalChannel0/1/0/12", "openconfig-terminal-device: optical-channel": { "config": {"frequency": "193100000", "target-output-power": -700, "operational-mode": 4178, "line-port": "0/1-Optics0/1/0/12"	"name":"0/1", "properties":{ "property":[{ "name":"LCMODE", "config":{ "name":"LCMODE", "value":"4x100GE-MXP-DD"
400GE	"index": 101, "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "description": "Client Logical Channel", "admin-state": "ENABLED", "loopback-mode": "NONE", "trib-protocol": "openconfig-transport-types: PROT_400GE", "logical-channel-type": "openconfig-transport-types: PROT_ETHERNET"	"index": 212, "config": {"index": 212,"admin-state": "ENABLED","loopback-mode": "NONE", "description": "Coherent DSP", "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "logical-channel-type": "openconfig-transport-types: PROT_OTN"	"name": "0/1-OpticalChannel0/1/0/12", "openconfig-terminal-device: optical-channel": {"config": {"frequency": "193100000", "target-output-power": -700, "operational-mode": 4178, "line-port": "0/1-Optics0/1/0/12"	"name":"0/1", "properties":{ "property":[{ "name":"LCMODE", "config":{ "name":"LCMODE", "value":"400GE-TXP-DD"

Sample Configuration for 400GE-TXP-DD

```
{
  "openconfig-terminal-device:terminal-device":{
     "logical-channels":{
        "channel":[
           {
              "index":1011,
              "config":{
                 "index":1011,
                 "admin-state":"ENABLED",
                 "loopback-mode":"NONE",
                 "description":"Coherent Logical Channel",
                 "rate-class":"openconfig-transport-types:TRIB_RATE_400G",
                 "logical-channel-type":"openconfig-transport-types:PROT_OTN"
              },
              "logical-channel-assignments":{
                 "assignment":[
                    {
                       "index":1,
```

```
                        "config":{
                           "index":1,
                           "allocation":"400",
                           "assignment-type":"OPTICAL_CHANNEL",
                           "optical-channel":"0/1-OpticalChannel0/1/0/11"
                        }
                     }
                  ]
               }
            },
            {
               "index":1010,
               "config":{
                  "index":1010,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_400G",
                  "admin-state":"ENABLED",
                  "loopback-mode":"NONE",
                  "description":"Client Logical Channel",
                  "trib-protocol":"openconfig-transport-types:PROT_400GE",
                  "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
               },
               "ingress":{
                  "config":{
                     "transceiver":"0/1-Optics0/1/0/10"
                  }
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
                        "config":{
                           "index":1,
                           "allocation":"400",
                           "assignment-type":"LOGICAL_CHANNEL",
                           "logical-channel":1011
                        }
                     }
                  ]
               }
            }
         ]
      }
   },
   "openconfig-platform:components":{
      "component":[
         {
            "name":"0/1",
            "properties":{
               "property":[
                  {
                     "name":"LCMODE",
                     "config":{
                        "name":"LCMODE",
                        "value":"400GE-TXP-DD"
                     }
                  }
               ]
            }
         },
         {
          "name": "0/1-OpticalChannel0/1/0/11",
          "openconfig-terminal-device:optical-channel": {
            "config": {
              "frequency": "193100000",
```

```
            "target-output-power": -700,
            "operational-mode": 4178,
            "line-port": "0/1-Optics0/0/0/11"
          }
        }
      }
    ]
  }
}
```

Sample Configuration for 4x100GE-MXP-DD

```
{
  "openconfig-terminal-device:terminal-device": {
    "logical-channels": {
      "channel": [
        {
          "index": 2303,
          "config": {
            "index": 2303,
            "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
            "admin-state": "ENABLED",
            "description": "Client Logical Channel",
            "trib-protocol": "openconfig-transport-types:PROT_100G_MLG",
            "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET"
          },
          "ingress": {
            "config": {
              "transceiver": "0/1-Optics0/1/0/5"
            }
          },
          "logical-channel-assignments": {
            "assignment": [
              {
                "index": 1,
                "config": {
                  "index": 1,
                  "allocation": "100",
                  "assignment-type": "LOGICAL_CHANNEL",
                  "logical-channel": 30001
                }
              }
            ]
          }
        },
        {
          "index": 30001,
          "config": {
            "index": 30001,
            "admin-state": "ENABLED",
            "description": "Coherent Logical Channel",
            "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
            "logical-channel-type": "openconfig-transport-types:PROT_OTN"
          },
          "logical-channel-assignments": {
            "assignment": [
              {
                "index": 1,
                "config": {
                  "index": 1,
                  "allocation": "400",
                  "assignment-type": "OPTICAL_CHANNEL",
                  "optical-channel": "0/1-OpticalChannel0/1/0/9"
```

```
                }
              }
            ]
          }
        }
      ]
    }
  },
  "openconfig-platform:components": {
    "component": [
      {
        "name": "0/1-OpticalChannel0/1/0/9",
        "openconfig-terminal-device:optical-channel": {
          "config": {
            "frequency": "193100000",
            "target-output-power": -700,
            "operational-mode": 4178,
            "line-port": "0/1-Optics0/1/0/9"
          }
        }
      },
      {
        "name":"0/1",
        "properties":{
           "property":[
              {
                 "name":"LCMODE",
                 "config":{
                    "name":"LCMODE",
                    "value":"4x100GE-MXP-DD"
                 }
              }
           ]
        }
     }
    ]
  }
}
```

Sample Configuration for 2x100GE-MXP-DD

```
{
"openconfig-terminal-device:terminal-device":
{"logical-channels":
 {"channel": [{"index": 2303,"config":
{"index": 2303,"rate-class":
"openconfig-transport-types:TRIB_RATE_100G",
"admin-state":
"ENABLED","description":
"Client Logical Channel","trib-protocol":
 "openconfig-transport-types:PROT_100G_MLG",
"logical-channel-type":
"openconfig-transport-types:PROT_ETHERNET"},"ingress":
 {"config": {"transceiver": "0/1-Optics0/1/0/7"
}},"logical-channel-assignments":
 {"assignment":
 [{"index": 1,"config": {"index": 1,"allocation":
"100","assignment-type":
 "LOGICAL_CHANNEL","logical-channel":
20001}}]}},
{"index": 20001,"config":
{"index": 20001,"admin-state":
```

```
"ENABLED","description": "Coherent Logical Channel","rate-class":
 "openconfig-transport-types:TRIB_RATE_200G","logical-channel-type":
"openconfig-transport-types:PROT_OTN"},
"logical-channel-assignments":
{"assignment": [{"index": 1,"config": {"index":
1,"allocation": "200","assignment-type":
"OPTICAL_CHANNEL","optical-channel":
 "0/1-OpticalChannel0/1/0/9"}}]}}]}
},"openconfig-platform:components":
 {"component": [{
"name": "0/1-OpticalChannel0/1/0/9",
"openconfig-terminal-device:optical-channel":
 {"config": {"frequency":
 "193100000","target-output-power": -700,"operational-mode":
 4178,"line-port": "0/1-Optics0/1/0/9"}}},{"name":"0/1","properties":
{"property":
[{"name":"LCMODE","config":
{"name":"LCMODE","value":"4x100GE-MXP-DD"}
}]}}]}}
```


Sample Configuration for 3x100GE-MXP-DD


```
{
"openconfig-terminal-device:
terminal-device":
 {"logical-channels":
{"channel": [{"index": 2303,"config":
 {"index": 2303,"rate-class":
"openconfig-transport-types:TRIB_RATE_100G","admin-state":
 "ENABLED","description":
 "Client Logical Channel","trib-protocol":
"openconfig-transport-types:PROT_100G_MLG",
"logical-channel-type":
"openconfig-transport-types:PROT_ETHERNET"},
"ingress": {"config":
 {"transceiver": "0/1-Optics0/1/0/8"}},
"logical-channel-assignments": {"assignment": [{"index": 1,"config": {"index": 1,"allocation":
 "100","assignment-type": "LOGICAL_CHANNEL","logical-channel":
 30001}}]}},{"index": 30001,"config":
 {"index": 30001,"admin-state": "ENABLED","description":
"Coherent Logical Channel","rate-class":
"openconfig-transport-types:
TRIB_RATE_300G","logical-channel-type":
 "openconfig-transport-types:PROT_OTN"},
"logical-channel-assignments":
 {"assignment": [{"index": 1,"config":
 {"index": 1,"allocation":
"300","assignment-type":
"OPTICAL_CHANNEL","optical-channel":
"0/1-OpticalChannel0/1/0/9"}}]}}]}},
"openconfig-platform:components":
 {"component": [{"name":
 "0/1-OpticalChannel0/1/0/9",
"openconfig-terminal-device:optical-channel":

 {"config": {"frequency":
"193100000","target-output-power":
 -700,"operational-mode":
4178,"line-port": "0/1-Optics0/1/0/9"}}},
{"name":"0/1","properties":{"property":
[{"name":"LCMODE","config":
```

```
{"name":"LCMODE","value":"4x100GE-MXP-DD"}
}]}}]}}
```

Supported OC Models and Features

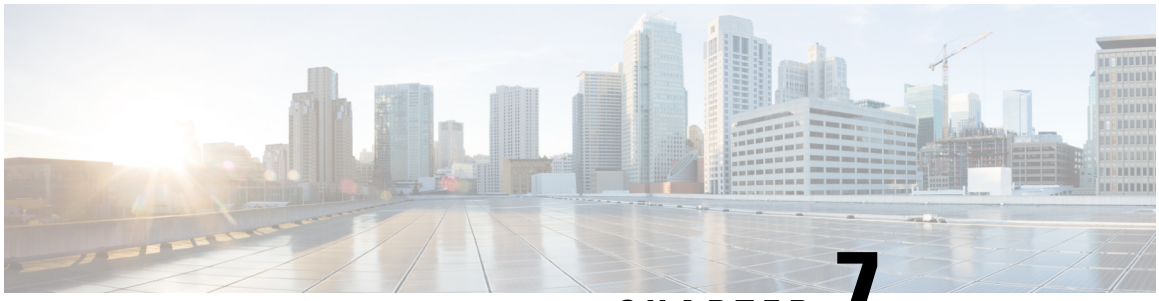
The 400GE-TXP-DD and 4x100GE-MXP-DD support the following OC models and fetures:

Table 32: Supported OC Models

Model	Feature
openconfig-platform.yang	Inventory and LC Mode
openconfig-platform-transceiver.yang	Pluggable Inventory and Oper Data
openconfig-terminal-device.yang	Logical and Optical Channels – Datapath and OperData
openconfig-interface.yang	Optical Interface Enable/Disable (shut/no-shut)

Supported Features

- Client loopback support for 100G and 400G logical channels.
- Trunk loopback support for the coherent DSP Optical channel.
- Admin-state support for all Logical and Optical channels.
- Laser Squelch and als-delay for ethernet client.
- Client Fec.
- LLDP.
- Trunk Fec modes such as Ofec and Cfec modes.
- Target output power and frequency on trunk optics.



CHAPTER 7

OpenConfig Support for NCS1K4-QXP-K9 Card

The NCS1K4-QXP-K9 card is a single slot line card that is equipped with 16 QSFPDD ports. This chapter briefs the detail configurations, client and trunk optics, supported OpenConfig models for the NCS1K4-QXP-K9 card.

Overview

The NCS1K4-QXP-K9 card is a single slot line card and it is equipped with 16 QSFPDD ports. You can configure eight QSFPDD ports as trunk and eight QSFPDD parts as client.

The NCS1K4-QXP-K9 card supports both transponder (TXP) and muxponder(MXP) configuration and they can coexist on the same line card. The NCS1K4-QXP-K9 card is operational by default and there is no need to configure the line card mode for the card.

Supported Operational modes, Optics, and OpenConfig Models

The NCS1K4-QXP-K9 card supports the following Operating modes, client and trunk optics, and OpenConfig models:

Operational Modes

The following table provides information for Operational modess, related modulation, and FEC:

Operational Modes	Modulation	FEC
400GE TXP	16QAM	CFEC and OFEC
4x100GE MXP	16QAM	CFEC and OFEC
3X100GE MXP	8QAM	OFEC
2X100GE MXP	QPSK	OFEC

Operational Modes	Modulation	FEC
100GE TXP	QPSK	OFEC

Client Optics

The following table provides information for PIDs, and its related payloads:

PID	Payload	Description
QDD-400G-FR4-S	400GE	The combination can be of 4x100GE, 3x100GE, and 2x100GE
QDD-400G-DR4-S	400GE and 100GE	
QDD-400G-AOC	400GE	
QDD-4X100G-LR-S	400GE and 100GE	
QSFP28-DR-S, QSFP28-FR-S, QSFP28-LR4, and QSFP28-LR-S	100GE	For 100G TXP

Note The 400G-DR4-S can be interoperable with QSFP28-DR-S and QSFP28 FR-S 100GE optics and the 4x100G-LR-S can be interoperable with QSFP28-DR-S and QSFP28 FR-S 100GE optics.

Trunk Optics

The following table provides information for PIDs, its related payloads, trunk ports, and inventory details:

PID	Payload	Trunk Port Number	Inventory Details
QDD-400G-ZRP-S	400G, 300G, 200G, and 100G	0, 2, 4, 6, 8, and 10	NAME: "0/0-Optics0/0/0/0", DESCR: "Cisco QSFP DD 400G ZRP Pluggable Optics Module" PID: QDD-400G-ZRP-S , VID: V01, SN: ACA25220033

OpenConfig Models

The NCS1K4-QXP-K9 card supports the following OpenConfig models:

Table 33: Supported OC Models

Model	Feature
openconfig-platform.yang	Inventory
openconfig-platform-transceiver.yang	Pluggable Inventory and Operational Data

Model	Feature
openconfig-terminal-device.yang	Logical and Optical Channels – Datapath and OperData
openconfig-interface.yang	Optical Interface Enable/Disable (shut/no-shut)

Client Configuration Details

The following table explains the different commands that are used for 100G and 400GE client ports.

Table 34: Configuration Details for 100G and 400GE Client Ports

Client Port	Logical Channel	Coherent DSP	Optical Channel
100G	"index":101, "rate-class": "openconfig-transport-types: TRIB_RATE_100G", "description": "Client Logical Channel", "admin-state":"ENABLED", "loopback-mode":"NONE", "trib-protocol": "openconfig-transport-types: PROT_100G_MLG", "logical-channel-type": "openconfig-transport-types: PROT_ETHERNET"	"index": 212, "config": { "index": 212, "admin-state": "ENABLED", "loopback-mode": "NONE", "description": "Coherent DSP", "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "logical-channel-type": "openconfig-transport-types: PROT_OTN"	"name": "0/1-OpticalChannel0/1/0/12", "openconfig-terminal-device: optical-channel": {"config": {"frequency": "193100000", "target-output-power": -700,"operational-mode": 4178, "line-port": "0/1-Optics0/1/0/12"
400GE	"index": 101, "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "description": "Client Logical Channel","admin-state": "ENABLED","loopback-mode": "NONE","trib-protocol": "openconfig-transport-types: PROT_400GE", "logical-channel-type": "openconfig-transport-types: PROT_ETHERNET"	"index":212, "config": { "index": 212, "admin-state": "ENABLED", "loopback-mode": "NONE","description": "Coherent DSP", "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "logical-channel-type": "openconfig-transport-types: PROT_OTN"	"name": "0/1-OpticalChannel0/1/0/12", "openconfig-terminal-device: optical-channel": { "config": { "frequency": "193100000", "target-output-power": -700,"operational-mode": 4178,"line-port": "0/1-Optics0/1/0/12"

Sample Configurations

Configuring 400 TXP (Client 1 and Slice 0)


```
{
   "openconfig-terminal-device:terminal-device":{
      "logical-channels":{
         "channel":[
            {
               "index":101,
               "config":{
                  "index":101,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_400G",
                  "admin-state":"ENABLED",
                  "description":"Client Logical Channel",
                  "trib-protocol":"openconfig-transport-types:PROT_400GE",
                  "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
               },
               "ingress":{
                  "config":{
                     "transceiver":"0/0-Optics0/0/0/1",
                     "physical-channel":[
                        1
                     ]
                  }
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
                        "config":{
                           "index":1,
                           "allocation":"400",
                           "assignment-type":"LOGICAL_CHANNEL",
                           "description":"logical to logical assignemnt",
                           "logical-channel":110
                        }
                     }
                  ]
               }
            },
            {
               "index":110,
               "config":{
                  "index":110,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_400G",
                  "admin-state":"ENABLED",
                  "description":"Coherent DSP",
                  "logical-channel-type":"openconfig-transport-types:PROT_OTN"
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
                        "config":{
                           "index":1,
                           "allocation":"400",
                           "assignment-type":"OPTICAL_CHANNEL",
                           "description":"logical to optical",
                           "optical-channel":"0/0-OpticalChannel0/0/0/0"
                        }
```

```
                    }
                ]
            }
        }
    ]
  }
 },
 "openconfig-platform:components":{
    "component":[
       {
          "name":"0/0-OpticalChannel0/0/0/0",
          "openconfig-terminal-device:optical-channel":{
             "config":{
                "line-port":"0/0-Optics0/0/0/0"
             }
          }
       }
    ]
 }
}
```

Configuring 4x100G MXP (Client 1 and Slice 0)

```
{
   "openconfig-terminal-device:terminal-device":{
      "logical-channels":{
         "channel":[
            {
               "index":101,
               "config":{
                  "index":101,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
                  "admin-state":"ENABLED",
                  "description":"Client Logical Channel",
                  "trib-protocol":"openconfig-transport-types:PROT_100GE",
                  "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
               },
               "ingress":{
                  "config":{
                     "transceiver":"0/0-Optics0/0/0/1",
                     "physical-channel":[
                        1
                     ]
                  }
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
                        "config":{
                           "index":1,
                           "allocation":"100",
                           "assignment-type":"LOGICAL_CHANNEL",
                           "description":"logical to logical assignemnt",
                           "logical-channel":110
                        }
                     }
                  ]
               }
            },
            {
               "index":102,
               "config":{
                  "index":102,
```

```
            "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
            "admin-state":"ENABLED",
            "description":"Client Logical Channel",
            "trib-protocol":"openconfig-transport-types:PROT_100GE",
            "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
         },
         "ingress":{
            "config":{
               "transceiver":"0/0-Optics0/0/0/1",
               "physical-channel":[
                  2
               ]
            }
         },
         "logical-channel-assignments":{
            "assignment":[
               {
                  "index":1,
                  "config":{
                     "index":1,
                     "allocation":"100",
                     "assignment-type":"LOGICAL_CHANNEL",
                     "description":"logical to logical assignemnt",
                     "logical-channel":110
                  }
               }
            ]
         }
      },
      {
         "index":103,
         "config":{
            "index":103,
            "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
            "admin-state":"ENABLED",
            "description":"Client Logical Channel",
            "trib-protocol":"openconfig-transport-types:PROT_100GE",
            "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
         },
         "ingress":{
            "config":{
               "transceiver":"0/0-Optics0/0/0/1",
               "physical-channel":[
                  3
               ]
            }
         },
         "logical-channel-assignments":{
            "assignment":[
               {
                  "index":1,
                  "config":{
                     "index":1,
                     "allocation":"100",
                     "assignment-type":"LOGICAL_CHANNEL",
                     "description":"logical to logical assignemnt",
                     "logical-channel":110
                  }
               }
            ]
         }
      },
      {
         "index":104,
```

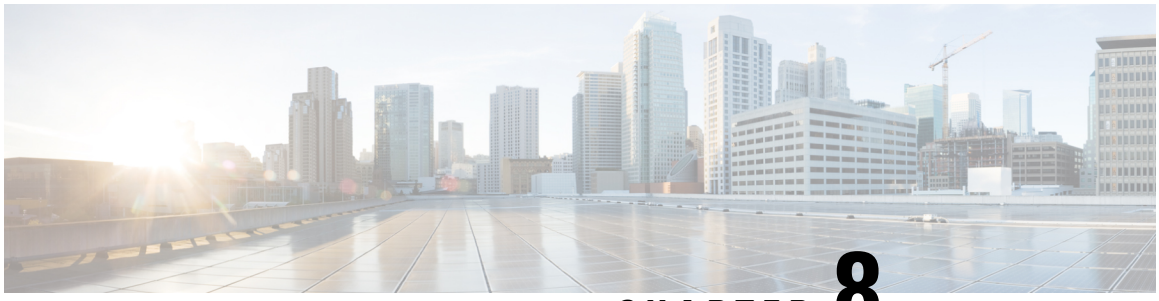
```
                  "config":{
                     "index":104,
                     "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
                     "admin-state":"ENABLED",
                     "description":"Client Logical Channel",
                     "trib-protocol":"openconfig-transport-types:PROT_100GE",
                     "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
                  },
                  "ingress":{
                     "config":{
                        "transceiver":"0/0-Optics0/0/0/1",
                        "physical-channel":[
                           4
                        ]
                     }
                  },
                  "logical-channel-assignments":{
                     "assignment":[
                        {
                           "index":1,
                           "config":{
                              "index":1,
                              "allocation":"100",
                              "assignment-type":"LOGICAL_CHANNEL",
                              "description":"logical to logical assignemnt",
                              "logical-channel":110
                           }
                        }
                     ]
                  }
               },
               {
                  "index":110,
                  "config":{
                     "index":110,
                     "rate-class":"openconfig-transport-types:TRIB_RATE_400G",
                     "admin-state":"ENABLED",
                     "description":"Coherent DSP",
                     "logical-channel-type":"openconfig-transport-types:PROT_OTN"
                  },
                  "logical-channel-assignments":{
                     "assignment":[
                        {
                           "index":1,
                           "config":{
                              "index":1,
                              "allocation":"400",
                              "assignment-type":"OPTICAL_CHANNEL",
                              "description":"logical to optical",
                              "optical-channel":"0/0-OpticalChannel0/0/0/0"
                           }
                        }
                     ]
                  }
               }
            ]
         }
      },
      "openconfig-platform:components":{
         "component":[
            {
               "name":"0/0-OpticalChannel0/0/0/0",
               "openconfig-terminal-device:optical-channel":{
                  "config":{
```

```
                    "line-port":"0/0-Optics0/0/0/0"
                }
            }
        }
    ]
  }
}
```

Configuring 400G TXP (Client 11 and Slice 5)

```
{
   "openconfig-terminal-device:terminal-device":{
      "logical-channels":{
         "channel":[
            {
               "index":1111,
               "config":{
                  "index":1111,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
                  "admin-state":"ENABLED",
                  "loopback-mode":"NONE",
                  "description":"Client Logical Channel",
                  "trib-protocol":"openconfig-transport-types:PROT_100GE",
                  "logical-channel-type":"openconfig-transport-types:PROT_ETHERNET"
               },
               "ingress":{
                  "config":{
                     "transceiver":"0/0-Optics0/0/0/11"
                  }
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
                        "config":{
                           "index":1,
                           "allocation":"100",
                           "assignment-type":"LOGICAL_CHANNEL",
                           "description":"logical to logical assignemnt",
                           "logical-channel":1110
                        }
                     }
                  ]
               },
               "ethernet":{
                  "config":{
                     "client-als":"LASER_SHUTDOWN"
                  }
               }
            },
            {
               "index":1110,
               "config":{
                  "index":1110,
                  "rate-class":"openconfig-transport-types:TRIB_RATE_100G",
                  "admin-state":"ENABLED",
                  "loopback-mode":"NONE",
                  "description":"Coherent DSP",
                  "logical-channel-type":"openconfig-transport-types:PROT_OTN"
               },
               "logical-channel-assignments":{
                  "assignment":[
                     {
                        "index":1,
```

```
                        "config":{
                           "index":1,
                           "allocation":"100",
                           "assignment-type":"OPTICAL_CHANNEL",
                           "description":"logical to optical",
                           "optical-channel":"0/0-OpticalChannel0/0/0/10"
                        }
                     }
                  ]
               }
            }
         ]
      }
   },
   "openconfig-platform:components":{
      "component":[
         {
            "name":"0/0-OpticalChannel0/0/0/10",
            "openconfig-terminal-device:optical-channel":{
               "config":{
                  "line-port":"0/0-Optics0/0/0/10"
               }
            }
         }
      ]
   }
}
```



CHAPTER 8

Configure 10G-Grey-Mux

This Chapter describes the implementation of OC support for 10G Grey Mux data path.

Table 35: Feature History

Feature Name	Release Information	Feature Description
OC Support for 10G Grey Mux	Cisco IOS XR Release 7.5.2	You can now configure a 10x10G muxponder on each slice of the OTN-XP line card. The 10x10G muxponder multiplexes ten 10G clients and map it to a grey OTU4 trunk signal. This feature improves efficiency, performance, and flexibility for customer networks allowing 10x10G client transport over 100G WDM wavelength.

Slices and Port Mapping on 10G Grey Mux

You can configure the 10G Grey Mux with slices 0 and 1. The following tables show port maping for slices 0 and 1.

Table 36: 10G Grey Mux: Ports Mapping for Slice 0

Trunk Port	Port 0
Trunk Payload	OTU4 (Grey)
Trunk PPM	QSFP-100G-LR4
Client Ports	Ports : 2(lanes-3 & 4), 4(lanes-1 to 4), and 5(lanes – 1 to 4)

Supported Client Rates	10GE, OTU2, and OTU2e
Client PPMs	QSFP-4x10-MLR and QSFP-40G-SR4

Table 37: 10G Grey Mux: Ports Mapping for Slice 1

Trunk Port	Port 0
Trunk Payload	OTU4 (Grey)
Trunk PPM	QSFP-100G-LR4
Clients Ports	Ports : 6(lanes-1 to 4),7(lanes-1 to 4), and 11(lanes-3 &4)
Supported Client Rates	10GE, OTU2, and OTU2e
Client PPMs	QSFP-4x10-MLR and QSFP-40G-SR4

Configuring 10G-Grey-Mux Clients

The following table explains the different commands used for 100G, OTU4, and 400GE client ports.

Table 38: Configuration Details for 10G, OTU2, and OTU2E Client Ports

Client Port	Logical Channel	Optical Channel	ODU4	OTU4	Optical Channel	LC Mode
10G	{ "index": 200, "rate-class": "[illegible]", "admin-state": "ENABLED", "loopback-mode": "NONE", "description": "Client Logical Channel", "trib-protocol": "[illegible]", "logical-channel-type": "[illegible]" }, "ingress": { "config": { "transceiver": "0/1-Optics0/1/0/5", "physical-channel": [1] } },	"index": 201, "rate-class": "[illegible]", "admin-state": "ENABLED", "description": "client-odu", "trib-protocol": "[illegible]", "logical-channel-type": "[illegible]"	"index": 100, "config": { "index": 100, "rate-class": "[illegible]", "admin-state": "ENABLED", "description": "ODU4", "trib-protocol": "[illegible]", "logical-channel-type": "[illegible]"	"index": 101, "rate-class": "[illegible]", "admin-state": "[illegible]", "description": "TERMINAL", "trib-protocol": "Coherent", "logical-channel-type": "[illegible]"	"name": "0/1-OpticalChannel0/1/0/0", "[illegible]": { "config": { "line-port": "0/1-Optics0/1/0/0" } }	"name": "0/1", "properties": { "property": [{ "name": "LCMODE", "config": { "name": "LCMODE", "value": "10g-GrEy-MxP" } }] }

Client Port	Logical Channel	Optical Channel	ODU4	OTU4	Optical Channel	LC Mode
OTU2	{ "index": 200, "rate-class": [redacted], "admin-state": "ENABLED", "loopback-mode": "NONE", "description": "Client Logical Channel", "trib-protocol": [redacted], "logical-channel-type": [redacted] }, "ingress": { "config": { "transceiver": "0/1-Optics0/1/0/5", "physical-channel": [1] } },	"index": 201, "rate-class": [redacted], "admin-state": "ENABLED", "description": "client-odu", "trib-protocol": [redacted], "logical-channel-type": [redacted]	"index": 100, "config": { "index": 100, "rate-class": [redacted], "admin-state": "ENABLED", "description": "ODU4", "trib-protocol": [redacted], "logical-channel-type": [redacted]	"index": 101, "rate-class": [redacted], "admin-state": "TERMINAL", "description": "Client", "trib-protocol": [redacted], "logical-channel-type": [redacted]	"name": [redacted], [redacted] { "config": { "line-port": "0/1-Optics0/1/0/0" } }	"name": "0/1", "properties": { "property": [{ "name": "LCMODE", "config": { "name": "LCMODE", "value": "10g-GrEy-MxP" } }] }

Client Port	Logical Channel	Optical Channel	ODU4	OTU4	Optical Channel	LC Mode
OTU2E	{ "index": 200, "rate-class": "admin-state": "ENABLED", "loopback-mode": "NONE", "description": "Client Logical Channel", "trib-protocol": "logical-channel-type": }, "ingress": { "config": { "transceiver": "0/1-Optics0/1/0/5", "physical-channel": [1] } },	"index": 201, "rate-class": "admin-state": "ENABLED", "description": "client-odu", "trib-protocol": "logical-channel-type":	"index": 100, "config": { "index": 100, "rate-class": "admin-state": "ENABLED", "description": "ODU4", "trib-protocol": "logical-channel-type":	"index": 101, "rate-class": "admin-state": "description": "logical-channel-type":	"name": "0/1-OpticalChannel0/0/0", { "config": { "line-port": "0/1-Optics0/1/0/0" } }	"name": "0/1", "properties": { "property": [{ "name": "LCMODE", "config": { "name": "LCMODE", "value": "10g-GrEy-MxP" } }] }

Sample Configuration

The following is a sample to configure the 10G-Grey-Mux single slice:

```
{
  "openconfig-terminal-device:terminal-device": {
    "logical-channels": {
      "channel": [
        {
          "index": 100,
          "config": {
            "index": 100,
            "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
```

```
      "admin-state": "ENABLED",
      "description": "ODU4",
      "trib-protocol": "openconfig-transport-types:PROT_ODU4",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN"
    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "100",
            "assignment-type": "LOGICAL_CHANNEL",
            "logical-channel": 101,
            "description": "Coherent to optical assignemnt"
          }
        }
      ]
    }
  },
  {
    "index": 101,
    "config": {
      "index": 101,
      "rate-class": "openconfig-transport-types:TRIB_RATE_100G",
      "admin-state": "ENABLED",
      "loopback-mode": "TERMINAL",
      "description": "Coherent",
      "trib-protocol": "openconfig-transport-types:PROT_OTU4",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN"
    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "100",
            "assignment-type": "OPTICAL_CHANNEL",
            "optical-channel": "0/1-OpticalChannel0/1/0/0"
          }
        }
      ]
    }
  },
  {
    "index": 200,
    "config": {
      "index": 200,
      "rate-class": "openconfig-transport-types:TRIB_RATE_10G",
      "admin-state": "ENABLED",
      "loopback-mode": "NONE",
      "description": "Client Logical Channel",
      "trib-protocol": "openconfig-transport-types:PROT_10GE_LAN",
      "logical-channel-type": "openconfig-transport-types:PROT_ETHERNET"
    },
    "ingress": {
      "config": {
        "transceiver": "0/1-Optics0/1/0/5",
        "physical-channel": [1]
      }
    },
    "logical-channel-assignments": {
      "assignment": [
```

```
            {
              "index": 1,
              "config": {
                "index": 1,
                "allocation": "10",
                "assignment-type": "LOGICAL_CHANNEL",
                "logical-channel": 201
              }
            }
          ]
        }
      },
      {
        "index": 201,
        "config": {
          "index": 201,
          "rate-class": "openconfig-transport-types:TRIB_RATE_10G",
          "admin-state": "ENABLED",
          "description": "client-odu",
          "trib-protocol": "openconfig-transport-types:PROT_ODU2E",
          "logical-channel-type": "openconfig-transport-types:PROT_OTN"
        },
        "logical-channel-assignments": {
          "assignment": [
            {
              "index": 1,
              "config": {
                "index": 1,
                "allocation": "10",
                "assignment-type": "LOGICAL_CHANNEL",
                "logical-channel": 100
              }
            }
          ]
        }
      }
    ]
  }
},
"openconfig-platform:components": {
  "component": [
    {
      "name": "0/1-OpticalChannel0/1/0/0"
    },
    {
      "name": "0/1-OpticalChannel0/1/0/0",
      "openconfig-terminal-device:optical-channel": {
        "config": {
          "line-port": "0/1-Optics0/1/0/0"
        }
      }
    },
    {
      "name": "0/1",
      "properties": {
        "property": [
          {
            "name": "LCMODE",
            "config": {
              "name": "LCMODE",
              "value": "10g-GrEy-MxP"
            }
          }
        ]
```

```
            }
          }
        ]
      }
}
```

Supported Payloads, OC Models, and Features

This section describes about the supported payloads, OC models, and features by 10G Grey Mux.:

- Payloads
- OC Models
- Features

Supported Payloads

The supported payloads are defined by the client bit rate and ODU rate that can be configured on the BO-LC 10g-grey-mxp mode.

Client Bit Rate

The possible client bit rate can be:

- 10GE (All 10 lanes of a slice set to 10GE rate).
- OTU2 (All 10 lanes of a slice set to OTU2 rate).
- Mix (A mix of 10GE, OTU2, and OTU2e rates configured on a slice.

ODU Rate

The possible ODU rate can be:

- ODU2e– mapper for 10GE.
- ODU2e- client and LO-ODU for OTU2e clients.
- ODU2– client and LO-ODU for OTU2 clients.

Note The ODU rates are not user configurable.

Table 39: Supported OC Models

Model	Feature
openconfig-platform.yang	Inventory and LC Mode
openconfig-platform-transceiver.yang	Pluggable Inventory and Oper Data
openconfig-terminal-device.yang	Logical and Optical Channels – Datapath and OperData
openconfig-interface.yang	Optical Interface Enable/Disable (shut/no-shut)

Supported Features

- Client loopback support for 10G, OTU2, and OTU2E logical channels
- Trunk loopback support for the OTU4 Optical channel.
- Admin-state support for all Logical and Optical channels.
- Laser Squelch and als-delay for ethernet client.
- Client Fec.
- LLDP.

Note The 10G0Grey-Mux does not support Trail Trace Identifier (TTI).