



Data Models Configuration Guide for Cisco NCS 1014, IOS XR Releases 7.11.x, 24.1.x, 24.2.x, and 24.3.x

First Published: 2023-11-30

Last Modified: 2024-09-04

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	1
	Data Models - Programmatic and Standards-based Configuration	1
	YANG model	1
	Components of Yang model	2
	Structure of Yang models	3
	Data types	3
	Data Model and CLI Comparison	4
	gRPC	4

CHAPTER 2	Using Data Models	7
	Use Data Models	7
	Enabling Netconf	8
	Enabling gRPC	9

CHAPTER 3	Supported YANG Models in NCS 1014	11
	Supported Yang Models	11
	Extending Cisco Native Models for OpenConfig Support	13
	OpenConfig Support for FEC Data	19

CHAPTER 4	OpenConfig Support for NCS1K14-2.4T-K9 Card	23
	Overview	23
	Supported Operational modes, Optics, and OpenConfig Models	23
	Extended Terminal Device Configuration for Baud Rate	25
	Extended Transceiver Model	27
	Client Configuration Details	28
	Sample Configurations	29



CHAPTER 1

Data Models

- [Data Models - Programmatic and Standards-based Configuration, on page 1](#)
- [YANG model, on page 1](#)
- [gRPC, on page 4](#)

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 Procedure 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 "al"; }

  /*** 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

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

Data Model and CLI Comparison

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

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

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

gRPC

gRPC is a language-neutral, open source, RPC (Remote Procedure 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 showcmandtextoutput



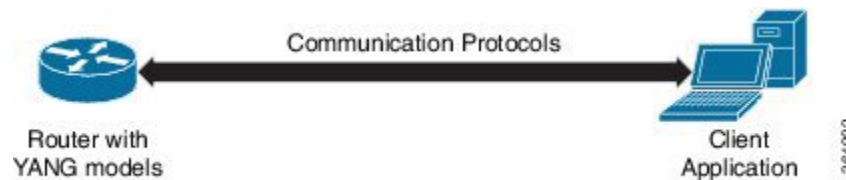
CHAPTER 2

Using Data Models

- [Use Data Models, on page 7](#)
- [Enabling Netconf, on page 8](#)
- [Enabling gRPC, on page 9](#)

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:

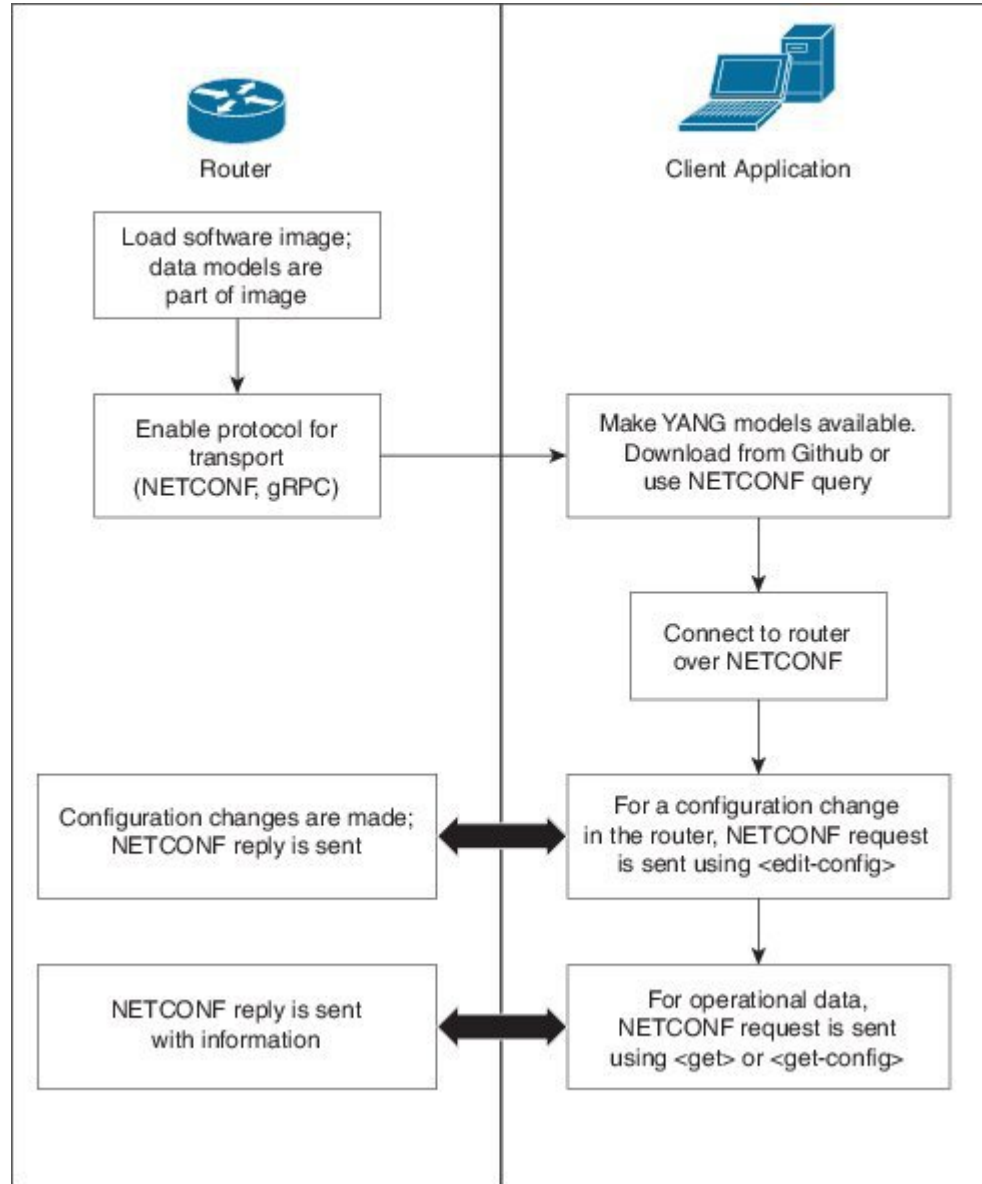
```
https://github.com/YangModels/yang/tree/master/vendor/cisco/xr
```

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



305313

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 1014

- [Supported Yang Models, on page 11](#)
- [Extending Cisco Native Models for OpenConfig Support, on page 13](#)
- [OpenConfig Support for FEC Data, on page 19](#)

Supported Yang Models

The following is the list of supported config, oper, and act YANG models for NCS 1014:

Table 1: Native Models

Config Models	Oper Models
Cisco-IOS-XR-osa-linesystem-cfg.yang	Cisco-IOS-XR-osa-hwmod-linesys-oper.yang
Cisco-IOS-XR-controller-ots-cfg.yang	Cisco-IOS-XR-controller-ots-oper.yang
Cisco-IOS-XR-ots-och-cfg.yang	Cisco-IOS-XR-controller-ots-och-oper.yang
Cisco-IOS-XR-controller-oms-cfg	Cisco-IOS-XR-controller-oms-oper.yang
Cisco-IOS-XR-controller-och-cfg	Cisco-IOS-XR-controller-och-oper.yang
Cisco-IOS-XR-controller-osc-cfg.yang	Cisco-IOS-XR-controller-osc-oper.yang
Cisco-IOS-XR-controller-dfb-cfg.yang	Cisco-IOS-XR-controller-dfb-oper.yang
Cisco-IOS-XR-pmengine-cfg.yang	Cisco-IOS-XR-pmengine-oper.yang
Cisco-IOS-XR-olc-cfg.yang	Cisco-IOS-XR-olc-oper.yang
Cisco-IOS-XR-fpd-infra-cfg.yang	
Cisco-IOS-XR-osa-ct-cfg.yang	Cisco-IOS-XR-osa-controller-optics-oper.yang
Cisco-IOS-XR-osa-sp-cfg.yang	Cisco-IOS-XR-osa-hwmod-linesys-oper.yang
Cisco-IOS-XR-osa-cfg.yang	Cisco-IOS-XR-osa-oper.yang
Cisco-IOS-XR-ikev2-cfg.yang	Cisco-IOS-XR-ikev2-oper.yang
Cisco-IOS-XR-controller-optics-cfg.yang	Cisco-IOS-XR-controller-optics-oper.yang
Cisco-IOS-XR-ethernet-lldp-cfg.yang	Cisco-IOS-XR-ethernet-lldp-oper.yang

Config Models	Oper Models
Cisco-IOS-XR-telemetry-model-driven-cfg.yang	Cisco-IOS-XR-telemetry-model-driven-oper.yang
Cisco-IOS-XR-ifmgr-cfg.yang	Cisco-IOS-XR-envmon-oper.yang
Cisco-IOS-XR-fpd-infra-cfg.yang	Cisco-IOS-XR-show-fpd-loc-ng-oper.yang
Cisco-IOS-XR-invproxy-hwmodule-cfg.yang	Cisco-IOS-XR-procfind-oper.yang
Cisco-IOS-XR-syncc-controller-cfg.yang	Cisco-IOS-XR-procthreadname-oper.yang
Cisco-IOS-XR-drivers-media-eth-gl-pfc-wd-cfg.yang	Cisco-IOS-XR-invmgr-oper.yang
Cisco-IOS-XR-drivers-media-eth-cfg.yang	Cisco-IOS-XR-plat-chas-invmgr-ng-oper.yang
Cisco-IOS-XR-drivers-icpe-ethernet-cfg.yang	Cisco-IOS-XR-platform-oper.yang
Cisco-IOS-XR-drivers-media-eth-cfg.yang	Cisco-IOS-XR-invmgr-diag-oper.yang
Cisco-IOS-XR-drivers-media-eth-gl-pfc-wd-cfg.yang	Cisco-IOS-XR-nto-misc-oper.yang
Cisco-IOS-XR-sysmgr-cfg.yang	Cisco-IOS-XR-wd-oper.yang
Cisco-IOS-XR-um-ncs-hw-module-osa-cfg.yang	Cisco-IOS-XR-ledmgr-oper.yang
	Cisco-IOS-XR-shellutil-filesystem-oper.yang
	Cisco-IOS-XR-shellutil-oper.yang
	Cisco-IOS-XR-drivers-media-eth-oper.yang
	Cisco-IOS-XR-mediasvr-linux-oper.yang
	Cisco-IOS-XR-drivers-media-eth-oper.yang
	Cisco-IOS-XR-sysmgr-oper.yang
	Cisco-IOS-XR-procmem-oper.yang
	Cisco-IOS-XR-procfind-oper.yang
Act Models	
Cisco-IOS-XR-upgrade-fpd-ng-act.yang	
Cisco-IOS-XR-shellutil-delete-act.yang	
Cisco-IOS-XR-drivers-media-eth-act.yang	
Cisco-IOS-XR-shellutil-copy-act.yang	
Cisco-IOS-XR-shellutil-copy-act.yang	
Cisco-IOS-XR-drivers-media-eth-act.yang	
Cisco-IOS-XR-sysmgr-act.yang	
Cisco-IOS-XR-system-reboot-act	

Config Models	Oper Models
Cisco-IOS-XR-install-act.yang	
Cisco-IOS-XR-install-augmented-act	

The following is the list of supported Open Config models:

Table 2: OpenConfig Models

openconfig-platform.yang
openconfig-platform-transceiver.yang
openconfig-terminal-device.yang
openconfig-interfaces.yang
openconfig-system.yang
openconfig-network-instance
openconfig-procmon.yang

See <https://cfng.cisco.com/ios-xr/yang-explorer/view-data-model> for the list of Yang models supported by NCS 1014.

Extending Cisco Native Models for OpenConfig Support

Table 3: Feature History

Feature Name	Release Information	Feature Description
Extending Cisco Native Models for OpenConfig Support	Cisco IOS XR Release 24.3.1	The OpenConfig model is completely supported on NCS 1014 chassis, by extending the existing Cisco native model configuration. It supports the Q-margin and Enhanced Q-margin parameters as part of the extended OpenConfig model.

From R24.3.1 onwards, OpenConfig augmentation support is available for NCS 1014. For any configuration parameters that exist in the Cisco native model but are not included in the OpenConfig model, then such parameters can be added to the OpenConfig model by extending the existing Cisco native model.

Benefits of Using OpenConfig Augmentation

OpenConfig augmentation allows NCS 1014 chassis to be fully managed using OpenConfig models.

Purpose of OpenConfig Augmentation for Cisco Native Models

The purpose is to identify errors in signal transmission by including Q margin and Enhanced Q margin values in the OpenConfig model, which requires adding them as extended leaves within the existing open terminal model configuration.

Enabling OpenConfig Augmentation Support for Cisco Native Models

To enable OpenConfig Augmentation support for Cisco native models in the terminal device refer to the given example.

The entry highlighted in bold shows the newly added OpenConfig model as an extension to the existing Cisco native model.

```

module: openconfig-terminal-device
  +--rw terminal-device
    +--rw config
      +--ro state
        +--rw logical-channels
          | +--rw channel* [index]
          | | +--rw index -> ../config/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
          | | | +--rw test-signal? boolean
          | | +--ro state
          | | | +--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 test-signal? boolean
          | | | +--ro link-state? enumeration
          | +--rw otn
          | | +--rw config
          | | | +--rw tti-msg-transmit? string
          | | | +--rw tti-msg-expected? string
          | | | +--rw tti-msg-auto? boolean
          | | | +--rw tributary-slot-granularity? identityref
          | | | +--ro state
          | | | | +--ro tti-msg-transmit? string
          | | | | +--ro tti-msg-expected? string
          | | | | +--ro tti-msg-auto? boolean
          | | | | +--ro tributary-slot-granularity? identityref
          | | | | +--ro tti-msg-recv? string
          | | | | +--ro rdi-msg? string
          | | | | +--ro errored-seconds? yang:counter64
          | | | | +--ro severely-errored-seconds? yang:counter64
          | | | | +--ro unavailable-seconds? yang:counter64
          | | | | +--ro code-violations? yang:counter64
          | | | | +--ro errored-blocks? yang:counter64
          | | | | +--ro fec-uncorrectable-blocks? yang:counter64
          | | | | +--ro fec-uncorrectable-words? yang:counter64
          | | | | +--ro fec-corrected-bytes? yang:counter64
          | | | | +--ro fec-corrected-bits? yang:counter64
          | | | | +--ro background-block-errors? yang:counter64
          | | | | +--ro pre-fec-ber
          | | | | | +--ro instant? decimal64
          | | | | | +--ro avg? decimal64
          | | | | | +--ro min? decimal64
          | | | | | +--ro max? decimal64
          | | | | | +--ro interval? oc-types:stat-interval
          | | | | | +--ro min-time? oc-types:timeticks64
  
```

```

| | | | +--ro max-time? oc-types:timeticks64
| | | | +--ro post-fec-ber
| | | | +--ro instant? decimal64
| | | | +--ro avg? decimal64
| | | | +--ro min? decimal64
| | | | +--ro max? decimal64
| | | | +--ro interval? oc-types:stat-interval
| | | | +--ro min-time? oc-types:timeticks64
| | | | +--ro max-time? oc-types:timeticks64
| | | | +--ro q-value
| | | | +--ro instant? decimal64
| | | | +--ro avg? decimal64
| | | | +--ro min? decimal64
| | | | +--ro max? decimal64
| | | | +--ro interval? oc-types:stat-interval
| | | | +--ro min-time? oc-types:timeticks64
| | | | +--ro max-time? oc-types:timeticks64
| | | +--ro esnr
| | | | +--ro instant? decimal64
| | | | +--ro avg? decimal64
| | | | +--ro min? decimal64
| | | | +--ro max? decimal64
| | | | +--ro interval? oc-types:stat-interval
| | | | +--ro min-time? oc-types:timeticks64
| | | | +--ro max-time? oc-types:timeticks64
| | +--ro oc-opt-ext:extended
| | | +--ro oc-opt-ext:state
| | | | +--ro oc-opt-ext:enhanced-q-margin? decimal64
| | | | +--ro oc-opt-ext:q-margin
| | | | | +--ro oc-opt-ext:instant? decimal64
| | | | | +--ro oc-opt-ext:avg? decimal64
| | | | | +--ro oc-opt-ext:min? decimal64
| | | | | +--ro oc-opt-ext:max? decimal64
| | | | | +--ro oc-opt-ext:interval? Decimal64
| | | | | +--ro oc-opt-ext:min-time? decimal64
| | | | | +--ro oc-opt-ext:max-time? decimal64
| +--rw ethernet
| | +--rw config
| | | +--rw client-als? enumeration
.....
.....

augment /oc-platform:components/oc-platform:component:
+--rw optical-channel
  +--rw config
    | +--rw frequency? oc-opt-types:frequency-type
    | +--rw target-output-power? decimal64
    | +--rw operational-mode? uint16
    | +--rw line-port? -> /oc-platform:components/component/name
  +--ro state
    | +--ro frequency? oc-opt-types:frequency-type
    | +--ro target-output-power? decimal64
    | +--ro operational-mode? uint16
    | +--ro line-port? ->
/oc-platform:components/component/name
| +--ro group-id? uint32
.....
.....

```

Augmented Model

This is the augmented model configuration.

The entry highlighted in bold shows the newly added OpenConfig model as an extension to the existing Cisco native model.

```

module Cisco-IOS-XR-openconfig-terminal-device-ext{
    namespace "http://cisco.com/ns/yang/"+
        "Cisco-IOS-XR-openconfig-terminal-device-ext";

    prefix oc-opt-ext;

    import openconfig-platform {
        prefix oc-platform;
    }
    import openconfig-terminal-device {
        prefix oc-opt-term;
    }

    organization "Cisco Systems, Inc.";

    contact
        "Cisco Systems, Inc.
        Customer Service

        Postal: 170 West Tasman Drive
        San Jose, CA 95134

        Tel: +1 800 553-NETS

        E-mail: cs-yang@cisco.com";

    description
        "This module is an extension of optical terminal device model
        and contains the definition of extended parameters for Optical
        Channels in order to optimize the AC1200 settings to get the highest
        performance and spectral efficiency.

        This module contains definitions for the following management objects:
        General Parameters
        Submarine Parameters

        Copyright (c) 2013-2023 by Cisco Systems, Inc.
        All rights reserved.";

    revision 2023-05-08 {
        description
            "Addition of baud-rate configuration";
    }

    revision 2023-02-28 {
        description
            "Addition of input power lower and upper thresholds parameters and
            Fastpoll enable configuration";
        reference "7.8.1";
    }
    revision 2020-08-30 {
        description
            "IOS XR 6.0 revision";
        reference "7.3.1";
    }

    grouping terminal-device-optical-channel-ext-info {
        description "Submarine parameters for optical channel";
        leaf optics-cd-min {
            type int32 {

```

```

        range "-350000..350000";
    }
    description
        "Select min chromatic dispersion (in units of
        ps/nm)";
}
leaf optics-cd-max {
    type int32 {
        range "-350000..350000";
    }
    description
        "Select max chromatic dispersion (in units of
        ps/nm)";
.....
.....

augment "/oc-platform:components/oc-platform:component/oc-opt-term:optical-channel" {
    container extended {
        description
            "Enclosing container for the list of Subsea parameters";

        container config {
            description
                "Extended Configuration parameters";
            leaf rx-voa-target-power {
                type int32 {
                    range "-190..30";
                }
                description "Receive Target Power in increments of 0.1 dBm
                    Default value is -5 dBm";
            }
            leaf rx-voa-fixed-ratio {
                type int32 {
                    range "100..1700";
                }
                description "Receive Ratio of Optical Attenuation in
                    increments of 0.01 dB Default value is 15 dB";
            }
            leaf fastpoll-sop-enable {
                type boolean {
                }
                description "Receive Fastpoll status if it is enabled or
                    disabled";
            }
            uses terminal-device-optical-channel-ext-info;
        }

        container state {

            config false;

            description
                "Extended Operational parameters";
            .....
        }
    }

augment
"/oc-opt-term:terminal-device/oc-opt-term:logical-channels/oc-opt-term:channel/oc-opt-term:otn"
{
    container extended {
augment
"/oc-opt-term:terminal-device/oc-opt-term:logical-channels/oc-opt-term:channel/oc-opt-term:otn"
{
    container extended {
        config false;
    }
}

```

```

        container state {
            config false;

            leaf enhanced-q-margin {
                type decimal64 {
                    fraction-digits 2;
                }
                description
                    "Enhanced Instantaneous Q-Margin Value";
            }

            container q-margin {
                description
                    "Q-Margin value in dB with two decimal precision. Values
                    include the instantaneous, average, minimum, and maximum
                    statistics. If avg/min/max statistics are not supported,
                    the target is expected to just supply the instant value";

                leaf instant {
                    type decimal64 {
                        fraction-digits 2;
                    }
                    description
                        "The instantaneous value of the q-margin (in units of dB).";
                }

                leaf avg {
                    type decimal64 {
                        fraction-digits 2;
                    }
                    description
                        "The arithmetic mean value of the q-margin over the time interval of 10 or
                        30 sec. (in units of dB).";
                }

                leaf min {
                    type decimal64 {
                        fraction-digits 2;
                    }
                    description
                        "The minimum value of the q-margin over the time interval of 10 or 30 sec.
                        (in units of dB).";
                }

                leaf max {
                    type decimal64 {
                        fraction-digits 2;
                    }
                    description
                        "The maximum value of the q-margin over the time interval of 10 or 30 sec.
                        (in units of dB).";
                }

                leaf min-time {
                    type decimal64 {
                        fraction-digits 2;
                    }
                    description
                        "The absolute time at which the minimum value occurred.
                        The value is the timestamp in nanoseconds relative to
                        the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
                }
            }
        }

```

```

        leaf max-time {
            type decimal64 {
                fraction-digits 2;
            }
            description
                "The absolute time at which the maximum value occurred.
                The value is the timestamp in nanoseconds relative to
                the Unix Epoch (Jan 1, 1970 00:00:00 UTC).";
        }

        leaf interval {
            type decimal64 {
                fraction-digits 2;
            }
            description
                "If supported by system, this reports the time interval
                over which the min/max/average statistics are computed by the
                system.";
        }
    }
}
description
    "This augment extends the operational data of
    'terminal-otn-protocol-state'";
}
description
    "This augment extends the operational data of
    'terminal-otn-protocol-top'";
}
}
}

```



Note The CLI appearing in bold is the augmented code added to the existing Cisco native model.

OpenConfig Support for FEC Data

Table 4: Feature History

Feature Name	Release Information	Feature Description
OpenConfig Support for FEC Data	Cisco IOS XR Release 24.3.1	OpenConfig model support is added for Forward Error Correction (FEC) data on the NCS 1014 chassis. It helps in avoiding deviations in the pre-fec-ber and post-fec-ber leaves for pluggables.

In the NCS 1014 chassis, Forward Error Correction (FEC) is enabled on all 400G pluggables and in 100G deployments. The OpenConfig model allows monitoring of FEC statistics through NetConf and telemetry for the NCS1K14-2.4T-X-K9 card.

Purpose of OpenConfig Support for FEC Data in NCS 1014 Chassis

In the NCS 1014 chassis, Cisco native models do not support the **pre-fec-ber** and **post-fec-ber** containers which are needed for monitoring FEC statistics on the NCS1K14-2.4T-X-K9 card. The OpenConfig model can be used for this purpose by extending the existing native commands.

Benefits of Providing OpenConfig Support for FEC Data in NCS 1014

Enabling OpenConfig model support for FEC data in NCS 1014 avoids deviations in the **pre-fec-ber** and **post-fec-ber** leaves for transceivers across all routing and optical platforms.

Supported FEC Parameters

The instant data and performance monitoring statistics parameters are enabled for supporting FEC on transceivers are:

- min,
- max,
- avg,
- min-time and
- max-time,
- instant.

Enabling OpenConfig Support for FEC Data

To enable OpenConfig operations and telemetry support for FEC data, refer to the example.

The entry highlighted in bold shows the newly added OpenConfig model telemetry support for FEC data.

```

module: openconfig-platform-transceiver
augment /oc-platform:components/oc-platform:component:
  +---rw transceiver
    +---rw config
      | +---rw enabled? boolean
      | +---rw form-factor-preconf? identityref
      | +---rw ethernet-pmd-preconf? identityref
      | +---rw fee-mode? identityref
    +---ro state
      | +---ro enabled? boolean
      | +---ro form-factor-preconf? identityref
      | +---ro ethernet-pmd-preconf? identityref
      | +---ro fee-mode? identityref
      | +---ro present? enumeration
      | +---ro form-factor? identityref
      | +---ro connector-type? identityref
      | +---ro vendor? string
      | +---ro vendor-part? string
      | +---ro vendor-rev? string
      | +---ro ethernet-pmd? identityref
      | +---ro sonet-sdh-compliance-code? identityref
      | +---ro otn-compliance-code? identityref
      | +---ro serial-no? string
      | +---ro date-code? oc-yang:date-and-time
      | +---ro fault-condition? boolean
      | +---ro fee-status? identityref
      | +---ro fee-uncorrectable-blocks? yang:counter64

```



```

    | +--ro fee-uncorrectable-words? yang:counter64
    | +--ro fee-corrected-bytes? yang:counter64
    | +--ro fee-corrected-bits? yang:counter64
+--ro pre-fec-ber
  | +--ro instant? decimal64
  | +--ro avg? decimal&
  | +--ro min? decimal64
  | +--ro max? decimal&
  | +--ro interval? oc-types:stat-interval
  | +--ro min-time? oc-types:timeticks64
  | +--ro max-time? oc-types:timeticks64
+--ro post-fec-ber
  | +--ro instant? decimal64
  | +--ro avg? decimal64
  | +--ro min? decimal64
  | +--ro max? decimal64
  | +--ro interval? oc-types:stat-interval
  | +--ro min-time? oc-types:timeticks64
  | +--ro max-time? oc-types:timeticks64
+--ro output-power
  | +--ro instant? decimal64
  | +--ro avg? decimal64
  | +--ro max-time? oc-types:timeticks64
+--ro input-power
  | +--ro instant? decimal64
  | +--ro avg? decimal64
  | +--ro min? decimal64
  | +--ro max? decimal64
  | +--ro interval? oc-types:stat-interval
  | +--ro min-time? oc-types:timeticks64
  | +--ro max-time? oc-types:timeticks64
+--ro laser-bias-current
  .....
+--ro laser-bias-current
  +--ro instant? decimal64
  +--ro avg? decimal64
  +--ro min? decimal64
  +--ro max? decimal64
  +--ro interval? oc-types:stat-interval
  +--ro min-time? oc-types:timeticks64
  +--ro max-time? oc-types:timeticks64
augment / oc-if:interfaces/oc-if:interface/oc-if:state:
+--ro transceiver? -> /oc-platform:components/component[oc
platform:name=current()/../oc-port:hardware-port]/ocplatform:subcomponents/subcomponent/name
augment / oc-if:interfaces/oc-if:interface/oc-if:state:
+--ro physical-channel* -> /oc-platform:components/component[oc
platform:name=current()/../oc-transceiver:transceiver]/transceiver/physical▶
channels/channel/index

```

Instant value example:

```

instant value:
RP/0/RP0/CPU0:N112#sh controllers fourHundredGigEctrlr 0/2/0/1
Operational data for interface FourHundredGigEctrlr0/2/0/1:

```

State:

```

Administrative state: enabled
Operational state: Up
LED state: Green On
Maintenance: Disabled
AINS Soak: None
  Total Duration: 0 hour(s) 0 minute(s)
  Remaining Duration: 0 hour(s) 0 minute(s) 0 second(s)
Laser Squelch: Disabled

```

```

Insert Idle Ingress: Disabled
Insert Idle Egress: Disabled

```

Phy:

```

Media type: Not known
Statistics:

```

```

FEC:
  Corrected Codeword Count: 35409983      Valid: True      Start time:
14:45:53 Mon Aug 19 2024
  Uncorrected Codeword Count: 56          Valid: True      Start time:
14:45:53 Mon Aug 19 2024
PCS:
  Total BIP errors: 0                     Valid: True      Start time:
14:45:53 Mon Aug 19 2024
  Total frame errors: 0                   Valid: False     Start time:
14:45:53 Mon Aug 19 2024
  Total Bad SH: 0                         Valid: False     Start time:
14:45:53 Mon Aug 19 2024

```

Autonegotiation disabled.

Operational values:

```

Speed: 400Gbps
Duplex: Full Duplex
Flowcontrol: None
Loopback: None (or external)
Pre FEC BER: 1.5E-09
Post FEC BER: 0.0E+00
BER monitoring:
  Not supported
Forward error correction: Standard (Reed-Solomon)
Holdoff Time: 0ms

```

Output CLI example of pre-fec and post-fec ber parameters:

```

min/max/avg:
RP/0/RP0/CPU0:N112#sh controllers fourHundredGigEctr1r 0/2/0/1 pm current 15-min fec

Ethernet FEC in the current interval [06:00:00 - 06:07:12 Tue Aug 20 2024]

FEC current bucket type : Valid
EC-WORDS : 313148 Threshold : 0 TCA(enable) : NO
UC-WORDS : 0 Threshold : 0 TCA(enable) : NO

MIN AVG MAX Threshold TCA Threshold TCA
(min) (enable) (max) (enable)
PreFEC BER : 8.2E-10 1.7E-09 2.6E-09 0E-15 NO 0E-15 NO
PostFEC BER : 0E-15 0E-15 0E-15 0E-15 NO 0E-15 NO

Last clearing of "show controllers ETHERNET" counters 13:52:01
RP/0/RP0/CPU0:N112#

```



CHAPTER 4

OpenConfig Support for NCS1K14-2.4T-K9 Card

The NCS1K14-2.4T-K9 card is a single slot line card. The card is equipped with six QSFPDD and two CIM-8 ports. This chapter briefs the detail configurations, client and trunk optics, supported OpenConfig models for the NCS1K14-2.4T-K9 card.

- [Overview, on page 23](#)
- [Supported Operational modes, Optics, and OpenConfig Models, on page 23](#)
- [Extended Terminal Device Configuration for Baud Rate, on page 25](#)
- [Extended Transceiver Model, on page 27](#)
- [Client Configuration Details, on page 28](#)
- [Sample Configurations, on page 29](#)

Overview

The NCS1K14-2.4T-K9 card is a single slot line card. The card is equipped with six QSFPDD and two CIM-8 ports. You can configure six QSFPDD ports as client and two CIM-8 as trunk.

The NCS1K14-2.4T-K9 card supports both transponder (TXP) and muxponder(MXP) configuration and they can coexist on the same line card.

Supported Operational modes, Optics, and OpenConfig Models

The NCS1K14-2.4T-K9 card supports the following Operational modes, client and trunk optics, and OpenConfig models:

Operational Modes

The following table provides information for Operational modes, config in SliceMode, Slice 0 Client, and Slice 1 Client:

Operational modes	Config in SliceMode	Slice 0 Client	Slice 1 Client
400G	4X100GE	1	4
600G	400GE+2x100GE	1,2	4,5
800G	2x400GE	1,2	4,5

Operational modes	Config in SliceMode	Slice 0 Client	Slice 1 Client
1000G	2x400GE+2x100GE	1,2,3	4,5,6

Client Optics

The following table provides information about PIDs, and related interface, transmit power, transmit wavelength, fiber type, fiber connector, and distance support:

PID	Interface	Transmit power	Transmit wavelength	Fiber type	Fiber connector	Distance support	Description
QDD-400G-FR4-S	400GE	-7.0 to +6.0 dbm per wavelength	1310 nm	Duplex SMF	Duplex LC connector	2km	Only be used as 400GE non-breakout mode
QDD-400G-DR4-S	400GBASE-DR4	-10.1 to +4.0 dbm per wavelength	1310 nm	MPO-12 parallel SMF	12-fiber MPO	500m	Can be used as 4x100GE breakout mode
QDD-400G-AOCxM	400GBASE-AOC	-10.1 to +4.0 dbm per wavelength	850 nm	MMF	AOC	1, 2, 3, 5, 7, 10, 15, 20, 25, and 30 meters	Only be used as 400GE non-breakout mode
QDD-4X100G-LR-S	10Base-LR	-8.2 to +0.5 per wavelength	1310nm	G.652 micron SMF	12-fiber MPO	10km	Can be used in 4x100GE breakout mode as well as 400GE non-breakout mode.

Trunk Optics



Note The transceiver name appears in the new format "Optics rack/slot/instance/port" from release 7.11.1.

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

PID	Payload	Trunk Port Number	Inventory Details
CIM8-C-K9	400G, 600G, 800G, and 1000G	0, and 7	NAME: "Optics0/1/0/0", DESCR: "Cisco CIM8 C K9 Pluggable Optics Module" PID: CIM8-C-K9, VID:VES1,SN:ACA273401DG

OpenConfig Models

The NCS1K14-2.4T-K9 card supports the following OpenConfig models:

Table 5: Supported OC Models

Model	Feature
openconfig-platform.yang	Inventory and LCMODE
openconfig-platform-transceiver.yang	Pluggable Inventory and Operational Data
openconfig-terminal-device.yang	Logical and Optical Channels – Datapath and OperData
openconfig-interface.yang	Optical Interface Enable/Disable (shut/no-shut)
openconfig-system.yang (augmented with openconfig-alarms)	Alarms
openconfig/gnoi/os.proto	Software Upgrade
Openconfig/gnoi/diag.proto	PRBS Testing

Extended Terminal Device Configuration for Baud Rate

The following table provides standard operational-modes for configuring the baud rate:

Table 6: Standard Operational Modes

Mode	FEC	Baud-Rate	Description
4201	SD_15	138.000000	SoftDecision_FEC15:Baud_138.00000000
4202	SD_15	139.000000	SoftDecision_FEC15:Baud_139.00000000
4203	SD_15	140.000000	SoftDecision_FEC15:Baud_140.00000000
4204	SD_15	141.000000	SoftDecision_FEC15:Baud_141.00000000
4205	SD_15	142.000000	SoftDecision_FEC15:Baud_142.00000000
4206	SD_15	100.000000	SoftDecision_FEC15:Baud_100.00000000
4207	SD_15	80.000000	SoftDecision_FEC15:Baud_80.00000000

Mode	FEC	Baud-Rate	Description
4208	SD_15	88.000000	SoftDecision_FEC15:Baud_88.00000000
4209	SD_15	98.000000	SoftDecision_FEC15:Baud_98.00000000
4210	SD_15	108.000000	SoftDecision_FEC15:Baud_108.00000000
4211	SD_15	118.000000	SoftDecision_FEC15:Baud_118.00000000
4212	SD_15	128.000000	SoftDecision_FEC15:Baud_128.00000000
4213	SD_15	110.000000	SoftDecision_FEC15:Baud_110.00000000
4214	SD_15	111.000000	SoftDecision_FEC15:Baud_111.00000000
4215	SD_15	112.000000	SoftDecision_FEC15:Baud_112.00000000
4216	SD_15	113.000000	SoftDecision_FEC15:Baud_113.00000000
4217	SD_15	114.000000	SoftDecision_FEC15:Baud_114.00000000
4218	SD_15	115.000000	SoftDecision_FEC15:Baud_115.00000000

You can use the **extended terminal-device baud rate** to set a new baud rate value compared to the value provided in the **Standard Operational Mode** table.



Note The Optical Channel name appears in the new format "OpticalChannel *rack/slot/instance/port*" from release 7.11.1.

Sample Configuration

```

-----
Edit config baud-rate
-----
<edit-config>
  <target>
    <candidate/>
  </target>
  <config>
    <components xmlns="http://openconfig.net/yang/platform">
      <component>

        <name>OpticalChannel0/0/0/0</name>
        <optical-channel xmlns="http://openconfig.net/yang/terminal-device">
          <extended
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-openconfig-terminal-device-ext">
            <config>
              <baud-rate>15.1234567</baud-rate>
            </config>
          </extended>
        </optical-channel>
      </component>
    </components>
  </config>
</edit-config>

```



Note If both the operating mode and extended baud rate exist, the line card employs the extended baud rate value.

Extended Transceiver Model

The extended transceiver model provides you with the Forward Error Correction (FEC) information for individual physical-channels.

Sample Configuration:

```
"Optics0/1/0/8": {
  "openconfig-platform-transceiver:transceiver": {
    "physical-channels": {
      "channel": {
        "1": {
          "state": {
            "index": 1,
            "input-power": {
              "avg": 1.64,
              "instant": 1.6,
              "interval": 1000000000,
              "max": 1.72,
              "max-time": 1649788692425519767,
              "min": 1.59,
              "min-time": 1649788694425593293
            },
            "laser-bias-current": {
              "avg": 800,
              "instant": 800,
              "interval": 1000000000,
              "max": 800,
              "max-time": 1649788690426089532,
              "min": 800,
              "min-time": 1649788690426089532
            },
            "output-frequency": 228849200,
            "output-power": {
              "avg": 1.62,
              "instant": 1.61,
              "interval": 1000000000,
              "max": 1.62,
              "max-time": 1649788690426089532,
              "min": 1.62,
              "min-time": 1649788690426089532
            }
          },
          "extended": {
            "state": {
              "index": 1
              "fec-mode": "openconfig-platform-types:FEC_ENABLED",
              "fec-uncorrectable-words": 0,
              "fec-corrected-words": 0
            }
          }
        }
      }
    }
  }
}
```

```

}
}

```

Client Configuration Details

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

Table 7: Configuration Details for 100G and 400GE Client Ports

Client Port	Logical Channel	Trunk ODU	Coherent DSP	Optical Channel
100G	<pre> "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" </pre>	<pre> index": 111, "config": { "index": 111, "rate-class": "openconfig-transport-types: TRIB_RATE_100G", "admin-state": "ENABLED", "description": "Trunk-side-ODU", "trib-protocol": "openconfig-transport-types: PROT_ODUFLEX_CBR", "logical-channel-type": "openconfig-transport-types: PROT_OTN" </pre>	<pre> "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" </pre>	<pre> "name": "OpticalChannel0/1/0/0", "openconfig-terminal-device: optical-channel": {"config": {"frequency": "193100000", "target-output-power": -700,"operational-mode": 4178, "line-port": "Optics0/1/0/0" </pre>

Client Port	Logical Channel	Trunk ODU	Coherent DSP	Optical Channel
400GE	<pre>"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"</pre>	<pre>"index": 211, "config": { "index": 211, "rate-class": "openconfig-transport-types: TRIB_RATE_400G", "admin-state": "ENABLED", "description": "Trunk-side-ODU", "trib-protocol": "openconfig-transport-types: PROT_ODUFLEX_CBR", "logical-channel-type": "openconfig-transport-types: PROT_OTN"</pre>	<pre>"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"</pre>	<pre>"name": "OpticalChannel0/1/0/0", "openconfig-terminal-device: optical-channel": { "config": { "frequency": "193100000", "target-output-power": -700, "line-port": "Optics0/1/0/0"</pre>



Note Trunk payload rate determines the Trib rate.

Sample Configurations

Configuring 400 TXP (Client and Slice)

```
{
  "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": "Optics0/1/0/1"
            }
          }
        }
      ]
    }
  }
}
```

```

    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "400",
            "assignment-type": "LOGICAL_CHANNEL",
            "description": "logical to logical assignemnt",
            "logical-channel": 111
          }
        }
      ]
    }
  },
  {
    "index": 111,
    "config": {
      "index": 111,
      "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
      "admin-state": "ENABLED",
      "description": "Trunk-side-ODU",
      "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": 30000
        }
      }
    ]
  }
},
{
  "index": 201,
  "config": {
    "index": 201,
    "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": "Optics0/1/0/2"
    }
  },
  "logical-channel-assignments": {
    "assignment": [
      {
        "index": 1,
        "config": {
          "index": 1,
          "allocation": "400",

```

```

        "assignment-type": "LOGICAL_CHANNEL",
        "description": "logical to logical assignemnt",
        "logical-channel": 211
      }
    ]
  },
  {
    "index": 211,
    "config": {
      "index": 211,
      "rate-class": "openconfig-transport-types:TRIB_RATE_400G",
      "admin-state": "ENABLED",
      "description": "Trunk-side-ODU",
      "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": 30000
          }
        }
      ]
    }
  },
  {
    "index": 30000,
    "config": {
      "index": 30000,
      "admin-state": "ENABLED",
      "description": "Coherent DSP",
      "logical-channel-type": "openconfig-transport-types:PROT_OTN"
    },
    "logical-channel-assignments": {
      "assignment": [
        {
          "index": 1,
          "config": {
            "index": 1,
            "allocation": "800",
            "assignment-type": "OPTICAL_CHANNEL",
            "description": "logical to optical",

            "optical-channel": "OpticalChannel0/1/0/0"
          }
        }
      ]
    }
  }
},
"openconfig-platform:components": {
  "component": [
    {

```

```
    "name": "OpticalChannel0/1/0/0",
    "openconfig-terminal-device:optical-channel": {
      "config": {
        "line-port": "Optics0/1/0/0"
      }
    }
  ],
},
"openconfig-interfaces:interfaces": {
  "interface": [
    {
      "name": "Optics0/1/0/0",
      "config": {
        "name": "Optics0/1/0/0",
        "type": "iana-if-type:opticalChannel",
        "description": "T0",
        "enabled": "true"
      }
    }
  ]
}
}
```