Open ROADM MSA Device White Paper

Open ROADM MSA

2024/04/04

Device Model Whitepaper

Provides documentation for YANG device models defined for Open ROADM MSA.

1 Document Revision History

Date	Revision	Description
November 5, 2020	7.1.0	Document updated to have 7.1.0 model support (v1.2)
March 7, 2024	13.1	Document updated to have 13.1 model support (minor updates pending)

Contents

List of Figures

List of Tables

List of YANG sub-trees/Examples

2 INTRODUCTION

2.1 Introduction to Open ROADM

The goals of the Open ROADM Multi-Source Agreement are (1) the disaggregation and opening up of traditionally proprietary ROADM systems and (2) the SDN-enablement of traditionally fixed ROADMs. The Open ROADM team's goal has been simplicity from the beginning, concentrating on lower performance metro systems to make it work in the first release.

There are many ways to disaggregate ROADM systems, e.g. hardware disaggregation (e.g. defining a common shelf) or functional disaggregation (less about hardware, more about function). Due to the complexity of common shelves, the Open ROADM MSA chose the functional disaggregation. We defined three optical functions: pluggable optics, transponder and ROADM (the optical switch part with amplifiers, couplers, WSS, etc.). Common shelves can be introduced at a later point for some functions, like transponders, if they make sense.

All of the three disaggregated functions (pluggable optics, transponder and ROADM) are controllable through an open standards-based API (written in the data modeling language YANG) that can be accessed through an SDN Controller using NETCONF. Open ROADM uses the NETCONF interface for provisioning, as well as PM, alarms, etc.

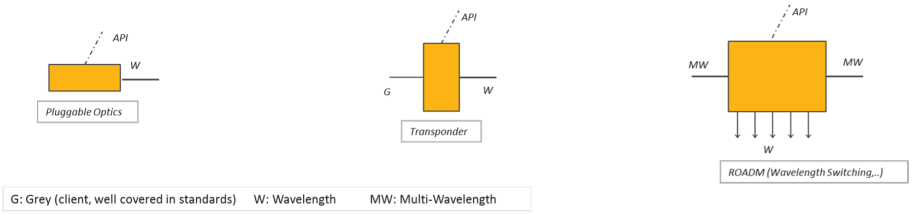

Figure 1: The three open ROADM disaggregated functions

There are two optical specifications, one called Single-Wave (or W) to define how pluggable optics or transponders interoperate and the other called Multi-Wave (or MW) to define how ROADMs interoperate.

The W specification for the initial release was only the 100G with Cortina HD-FEC, 200G and 400G with better performance were later introduced in Release 3.

The MW specification abstracts a lot of the previously proprietary ROADM to ROADM control loops into the SDN controller to simplify the interaction.
The OSC is a Gigabit Ethernet channel. In the first release, the specification was for fixed grid 50 GHz, 96 wavelengths for simplicity, with flexgrid modeling support added in the second release (2.x).

For release 2.x, we added two new functions, an OTN Xponder/switch for lower rate services, like 1GigE and 10GigE, as well as an in-line amplifier for reach extension up to 1000km for 100G HD-FEC.

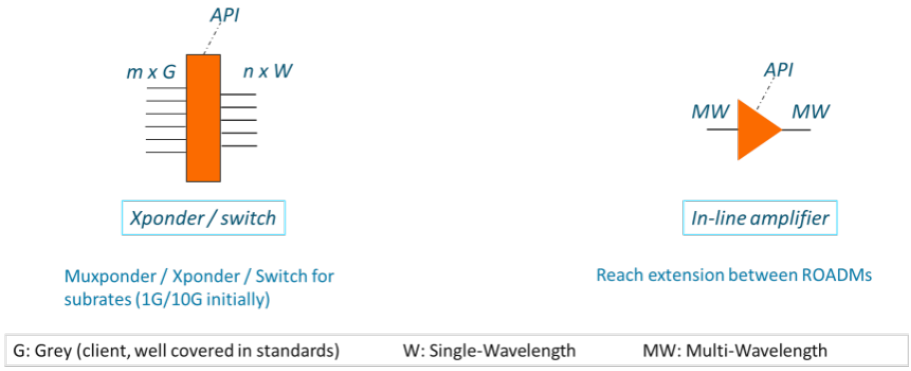

Figure 2: Switchponder and ILA

The YANG data models provided contain a network abstraction, as well as Service Remote Procedure calls (RPCs) and Device template. Every network provider can choose their own network abstraction, we just included one as an example on how device and network layer tie together, the same goes for the Service RPC calls. The Device template would be filled in by every hardware supplier with their device- specific information.

So for the overall architecture the diagram is below on how to glue it all together.

Since the whole network is supposed to be SDN-controlled, the assumption is that the ROADMs are flexible, software-controlled ROADMs with colorless-directionless (CD) or colorless-directionless- contentionless (CDC). The models are built to support both CD and CDC ROADMs.

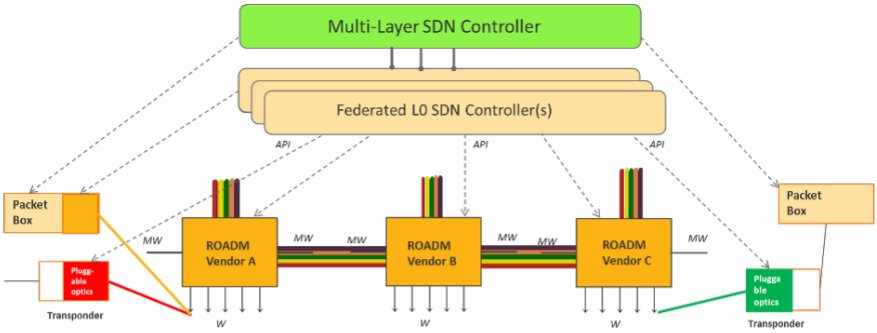

Figure 3: Open ROADM overall architecture

2.2 YANG Data Model

The Open ROADM MSA YANG data models are based on YANG v1.1 per RFC 7950 [1] and consists of:

- Config/Operational data defines the management objects which can be queried by the controller. Some management objects can be read/write (config), while others are read-only (operational). Examples include shelf commissioning data, wavelength connections, etc.
- Notifications for the purposes of reporting autonomous events to the controller. Examples include alarms, inventory changes, re-start, etc.
- Remote Procedure Calls (RPC) that do not effect a change in the device configuration data, e.g. restarts, LED control, debug information retrieval (tech info), get connection port trail to determine the route of a service through a device, disable automatic laser shutdown, start OTDR scan, and set current date and time.

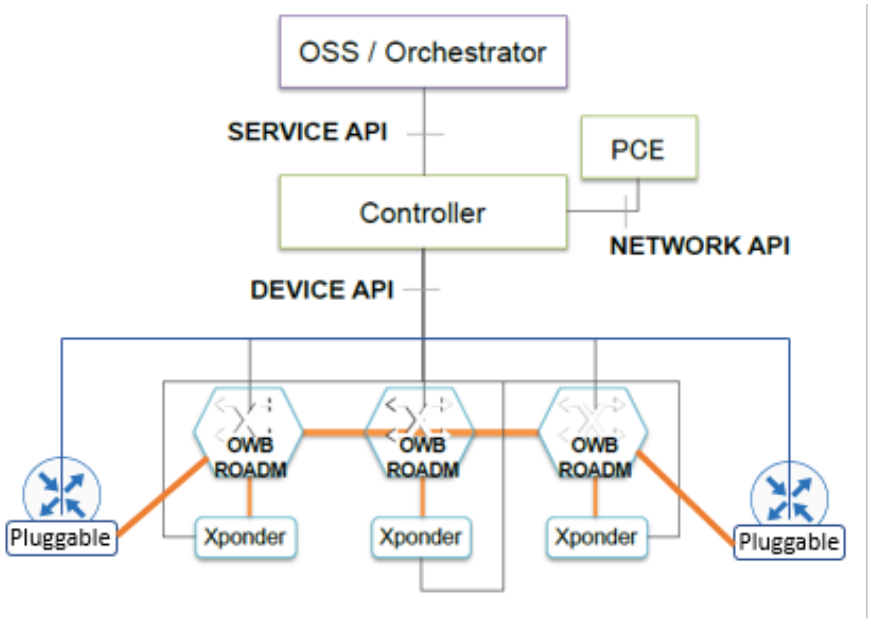

Figure 4: Open ROADM YANG Data Model Organization.

A device also supports the common models that provide the following objects and RPCs:

- Active alarm list including issuing alarm notifications (push model)
- Performance monitoring including both a current list and a historical list. The data tree view of the historical PM is optional for vendors to implement. Vendors must implement the retrieval of historical PM data via a file using the collect-historical-pm-file RPC
- Potential TCA list provides a means to view the potential TCAs on a node and to set the threshold levels. TCA notifications are also supported
- Software manifest YANG module. Note that this is intended for the device data that is provided out-of-band to a controller to guide the controller on software download and database operations.
- Other RPC operations including: clear PM register and change password

The Open ROADM MSA YANG data model is organized into the following functional models (refer to Figure 4: Open ROADM YANG Data Model Organization):

- **Service Model** is used for the creation, modification, and deletion of services.
- **Network Model** is used to abstract the network model for routing.
- **Device Model** discussed in this white paper is the interface to Open ROADM MSA devices and the basis for the inventory database. External pluggable optics are also supported

2.3 Feature Sets Of Each Release

Release number	Release date	Description
1.2.1	2017-02-06	Metro reach up to 500km, 100G line without DCMs, all-ROADM, 100G clients only, fixed grid
2.2	2017-12-15	Regional reach up to 1000km with ILA support, FlexGrid ready, OTN Crossponder for 1G and 10G clients
3.1	2018-05-30	200G/300G/400G wavelength with strong FEC, OTUCn/FlexO/OTSiG, asynchronous Service notifications, Service BER check RPC, Service resiliency, backup path topology, and routing metrics added in Service model
4.1	2018-11-30	400G Client, low noise amplifier for long reach, Streaming Telemetry from OpenConfig
5.1	2019-05-31	streaming telemetry from OpenConfig and bookended transponders
6.1	2019-11-29	B100G/OTUCn/ODU improvements, Service routing across planned/live/idealized network
7.0	2020-03-27	Unification of FlexO and OTSi interfaces, introduce profile-based capability announcements
7.1	2020-05-29	Bug-fixes, Device model enhancement, equipment based service diversity
8.0	2020-09-25	Enhance xponder container for service diversity, Service model enhancements
8.1	2020-12-11	Bug-fixes, Release device white-paper (7.1), Release 400G W Specs
9.0	2021-03-26	FOIC 1.1 identity support, Order-index for OTSI list for split-lambda, Add prov-max-add-drop-ports in SRG container, bug-fixes, Service model enhancements (split-lambda) to allow topology to display multiple routes
9.1	2021-05-28	TCA profile support, Modeling support for Edge ROADM and directional support, Support for 3rd party optics, Service model updates to support split-lambda service requests.
10.0	2021-09-24	Support for pluggable/operational mode, Support for foreign SRLGs, Service model enhancements to support service roll, addition of 3dB and 10dB spectral parameters.
10.1	2021-12-10	Service model enhancements: co-existence of co-routing and diverse containers, RPC for populating the standard operational modes.
11.0	2022-03-25	Equipment protection, opticalPowerOutputDrift PM type, network model node capability for optical operating modes.
11.1	2022-05-27	Additional PHY codes, device model top level containers no longer mandatory; service model RPCs and notifications for alien support, added explicit routing and ordered include list flags.
12.0	2022-09-30	Refactor of the IPv4/IPv6 models including removal of the IP address provisioning from the info container, removed IP and PPP interface types; added service model support for link-restriction-type. Corrected errors in the amp structure of the catalog and JSON files.
12.1	2022-12-09	Support for database recovery mode, manual laser disable, ROADM modeling of internal monitor ports and reporting port trail PM data, support pluggable lane alarms and PM to be reported against the port entity, and operational state reporting per direction. Network model added support for regenerator capabilities, service model support of restriction scope.
13.0	2023-03-31	Deprecated support for the opticalLineFail alarm; network model support for turn back function.
13.1	2023-05-26	Support for pcs-dp-qam16 modulation format, added openroadm-version to the manifest files, light touch provisioning templates for the remote transponder use case, support for FlexO-xe (ethernet optimized mapping), support for alarm/pm capable indication in the device and service models. Specification addressed JSON format errors.

Table 1: Summary of high-level feature sets of each release

3 HIGH LEVEL DEVICE DESCRIPTION

3.1 PHYSICAL DESIGN

The physical design for Open ROADM devices is not specified in the MSA. The form factor (width, depth, shelf), power supply (AC/DC) or standards met (such as NEBS3) are up to the vendor or network operator. The MSA only covers the functional aspects, as well as optical interoperability of the single- and multi-wave interfaces. Since no function of a craft-interface terminal has been specified, a manual factory-reset mechanism (such as a physical button) is desired to get the network element back to the factory reset state (same state as before power was switched on the first time). It is recommended, such reset mechanism should be hidden and clearly labeled so no accidental reset is triggered.

3.2 FUNCTIONAL DESIGN

3.2.1 Open ROADM Controller

The Open ROADM MSA architecture assumes the existence of an Open ROADM controller[1] that controls the Open ROADM MSA devices and provides device (inventory), network and service APIs to northbound OSS systems. The Open ROADM MSA does not specify the requirements, implementation, or operation of an Open ROADM controller. However, it is expected the specification of the Open ROADM MSA device, network, and service models would be used at the controller level. Some controller functions and/or procedures may be provided within this document. These items should be taken as guidance or examples of a possible controller implementation to guide the development of Open ROADM MSA devices.

3.2.2 Reconfigurable Add-Drop Multiplexer (ROADM)

The Open ROADM MSA defines a ROADM device capable of providing colorless and directionless (or colorless, directionless and contention less) add/drop functionality. This means that a ROADM site can add/drop any wavelength at any port and connect that wavelength to any direction in a ROADM device[2] from the local transponder. The Open ROADM MSA does not define implementation details such as the number of degrees, form-factor, etc. The Open ROADM MSA defines (refer to Figure 5: ROADM Interfaces):

- **API** using NETCONF interface with a YANG-based data model abstracts the management, control and provisioning of multi-vendor ROADM devices
- **Multi-wave (MW) interface** defines the optical specifications for the multi-wave DWDM interface between line degrees of the ROADM devices.
  - The MW interface includes an Optical Supervisory Channel (OSC) that provides LLDP-based topology information, supports laser safety, and provides MCN/DCN reach-through to remote ROADM nodes.
  - Aside from the mandatory safety automatic power shutdown control loop, no control loops are running between two adjacent ROADMs. All control loops are abstracted into the ROADM controller. There are, however, a few local link control loops running on the device, such as transient control. Otherwise, there is no ROADM to ROADM communication and optical equalization is handled off-box by the controller.
- **Single-wave (Wr) interface** defines the optical specifications for the add/drop ports of the ROADM devices.

Refer to the Open ROADM MSA specification found at [2],[3] for the C/D ROADM specifications:

- Overall Open ROADM Architecture in the “Architecture” tab
- General specs in the “Common” tab
- Physical specs defined in the “Physical spec” tab
- Line degree to line degree optical specs defined in the “MW-MW” tab
- Line degree to drop port and add port to line degree optical specs defined in the “MW-Wr” tab
- Local control specs defined in the “Local Control” tab
- Supported alarms defined in the “Alarms” tab
- Supported PMs defined in the “PMs” tab
- Ethernet OSC functionality defined in the “OSC Overview” tab
- Ethernet OSC optical specs defined in the “OSC-Optical Line Port” tab
- Automatic line shutoff functionality defined in the “Laser Safety” tab
- OAMP functions including protocol stack and specs in the “OAMP Port” tab

[1] The term “controller” used throughout the document refers to an Open ROADM controller
[2] The term “device”, “node” and “NE” are used interchangeably within this document.

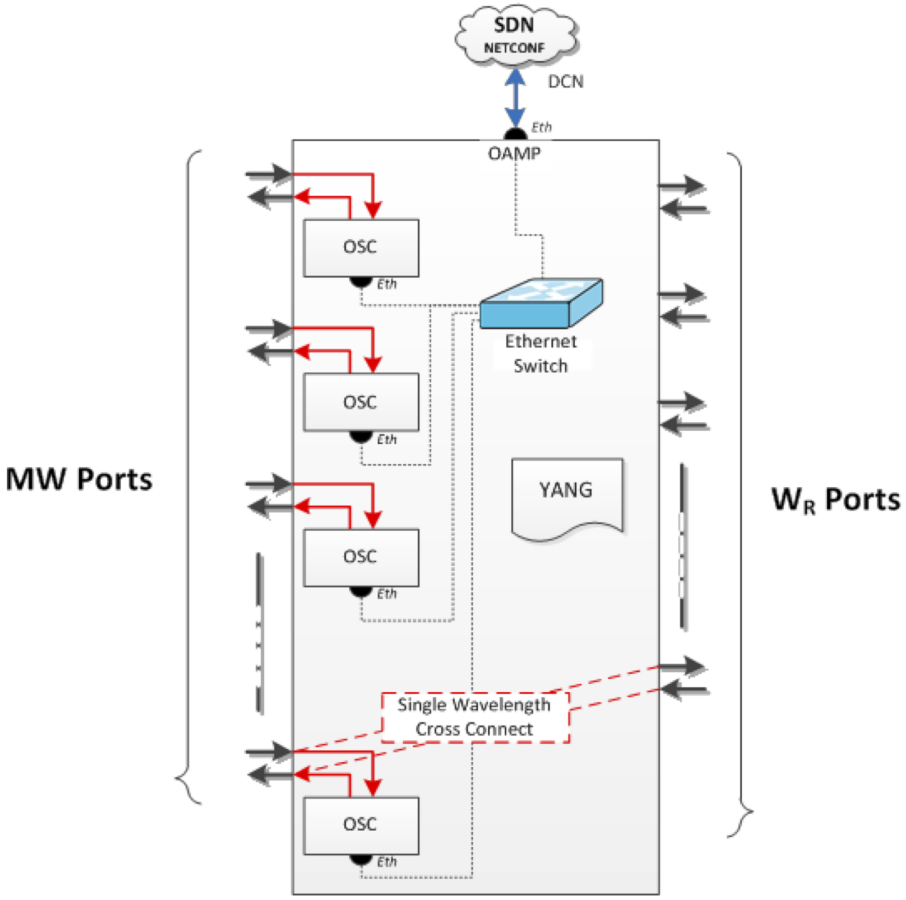

Figure 5: ROADM interfaces.

- OpenDaylight mounting requirements in the “Device Mounting” tab
- OTDR functionality defined in the “OTDR” tab

3.2.3 In-Line Amplifier (ILA)

The Open ROADM MSA defines an In-Line Amplifier (ILA) device capable of amplifying the different channels of the WDM multiplex. An ILA site amplifies different wavelengths of the multiplex in both directions between 2×N degrees. The MSA does not define implementation details such as the number of degrees, form-factor, etc.. A logical amplifier entity can include one or several amplifiers that can be implemented through one or several circuit-packs. The MSA defines (refer to Figure 6: In-line Amplifier Interfaces):

- **API** using NETCONF interface with a YANG-based data model abstracts the management, control and provisioning of multi-vendor ILA devices.
- **Multi-wave ILA (MWi) interface** defines the optical specifications for the multi-wave interface between ILAs or between ILA and line degrees of the ROADM devices.
  - The MWi interface includes an OSC that provides LLDP-based topology information, supports laser safety, and provides MCN/DCN reach-through to remote ILA nodes. Otherwise, there is no ILA node to ILA node or no ROADM node to ILA node communication. Optical equalization (in the limit of the specs) is handled off-box by the controller by setting target-tilt.

Please refer to the Open ROADM MSA specification found at [2],[3] for the ILA specifications:

- General specs in the “Common” tab
- Optical specs defined in the “MWi (standard)” and “MWi (low noise)” tabs
- Local control specs defined in the “Local Control” tab
- Supported PMs defined in the “PMs” tab (MWi column)
- Supported alarms defined in the “Alarms” tab
- Ethernet OSC functionality defined in the “OSC Overview” tab

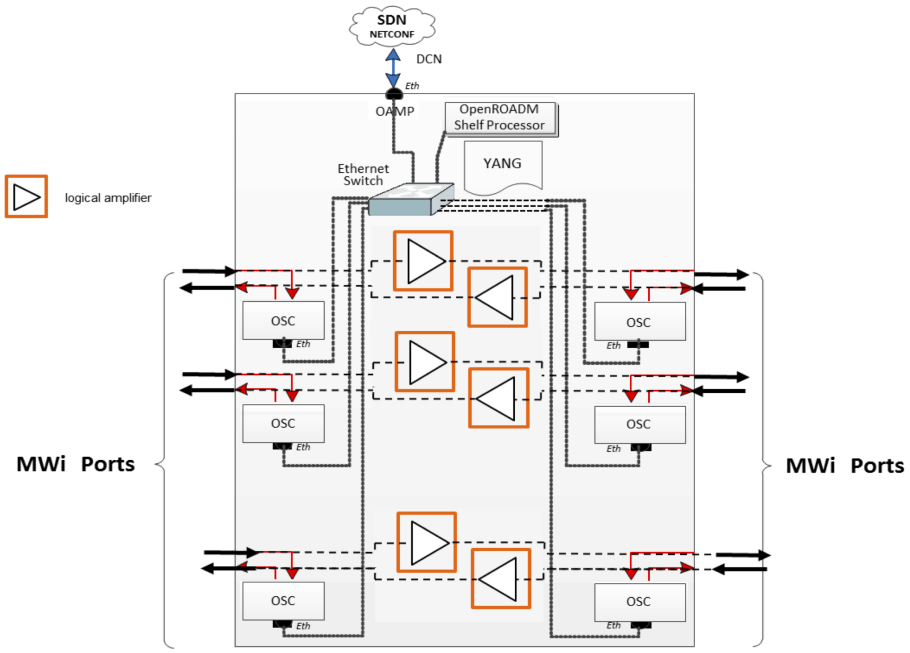

Figure 6: In-line Amplifier Interfaces.

- Ethernet OSC optical specs defined in the “OSC-Optical Line Port” tab
- Automatic line shutoff functionality defined in the “Laser Safety” tab
- OAMP functions including protocol stack and specs in the OAMP Port tab
- OpenDaylight mounting requirements in the “Device mounting” tab
- OTDR functionality defined in the “OTDR” tab

The Open ROADM MSA includes support for both a “standard” amplifier and a low noise amplifier. The specs of ILAs are in line with the specs of the ROADMs.

3.2.4 Xponder

Xponders is a general term for equipment supported by the network element with line-side interfaces having the same optical specification as a W port described in Section 3.3.1 (refer to Figure 7: Xponder Interfaces). Xponders include transponders, muxponders, switchponders, and regenerators. Each of these xponder types have different characteristics, as described in the following sub-sections.

Please refer to the Open ROADM MSA specification found at [2],[3] for the Xponder specifications:

- General specs in the “Common” tab
- Optical specs defined in the “W 100G Optical Spec” and “W 200G-400G Optical Spec” tabs [3]
- Support PMs defined in the “W PM Spec” tab
- Support alarms defined in the “W ALM Spec” tab
- Physical specs defined in the “Physical spec” tab
- OAMP functions including protocol stack and specs in the “OAMP Port” tab
- OpenDaylight mounting requirements in the “Device mounting” tab

3.2.4.1 Transponder

The Open ROADM MSA release 1.x defined a Transponder device capable of mapping a 100GE or OTU4 client signal into a 100G OTU4 DWDM signal for transport across an Open ROADM infrastructure. Support for the additional network rates of 200G, 300G, and 400G OTUCN were added in Open ROADM MSA release 3.x.

The MSA does not define implementation details such as the number of client/line ports, form-factor, etc.. The MSA defines:

- **API** using NETCONF interface with a YANG-based data model abstracts the management, control and provisioning of multi-vendor Transponder devices

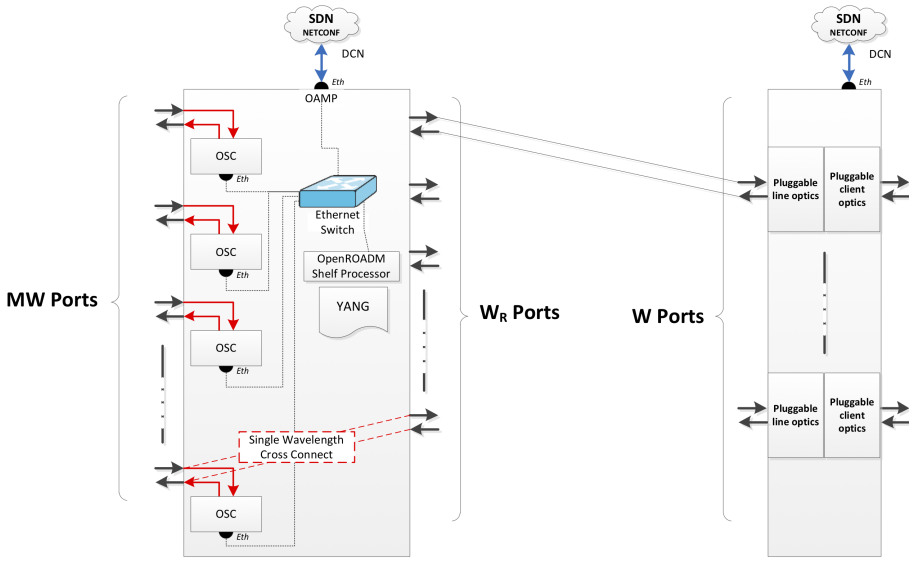

Figure 7: Xponder interfaces.

- **Line side, the single-wave (W) interface** defines the optical specifications for the full C-band tunable DWDM optical line interface of the Transponder that connects to a Wr add/drop port on the ROADM device
  - 100G Line-side pluggable
  - 200G-400G Line-side pluggable. For OpenROADM optical specifications [3]
- **Client-side**
  - 100G client interfaces/ports support:
    - 100GBASE-R mapped into OPU4 using PCS codeword transparent Ethernet mapping, PT=07
    - OTU4
  - 400G client interfaces/ports support:
    - 400GBASE-R could be mapped into OPU4flex(CBR), PT=32 (G.709 17.13.2)
    - 400GBASE-R could be mapped into OPU4flex(IMP), PT=1D (G.709 17.11)

Please refer to the Open ROADM MSA specification found at [2],[3]for the Transponder functional specifications (“W TRPN functional” tab).

3.2.4.2 Muxponder/Switchponder

The Open ROADM MSA defines a general Optical Transport Network (OTN) switching model which allows for the support of ODUk/ODUj level cross-connects between interfaces. **The MSA does not define implementation details such as the number of client/line ports, form-factor, etc.** The MSA (refer to [2]) defines:

- **API** using NETCONF interface with a YANG-based data model abstracts the management, control and provisioning of multi-vendor switch devices
- **Line side, the single-wave (W) interface** have the same specifications and requirements as those defined for a Transponder with the Single-wave (W) interface defining the optical specifications for the full C-band tunable DWDM optical line interface that connects to a Wr add/drop port on the ROADM device
  - 100G Line-side pluggable
  - 200G-400G Line-side pluggable For OpenROADM optical specifications [3]
- **Client-side**
  - 1GE client interfaces/ports support:
    - 1000BASE-X mapped into ODU0 using TTT and GMP, ITU-T G.709 [4] Clause 17.7.1.1 1000BASE-X Transcoding, PT=07
  - 10G client interfaces/ports support:
    - 10GBASE-R mapped into OPU2 using GFP-F, ITU-T G.7041 [5] Clause 7.1 Ethernet MAC Payload, PT=05

    - 10GBASE-R mapped into OPU2 using GFP-F, ITU-T G.7041 [5] Clause 7.9 Transporting Ethernet 10GBASE-R payloads with preamble transparency and ordered set information, PT=09
    - 10GBASE-R mapped into OPU2e using Bit-synchronous mapping (CBR10G3), PT=03
    - OTU2/2e
  - 100G client interfaces/ports support:
    - 100GBASE-R mapped into OPU4 using PCS codeword transparent Ethernet mapping, PT=07
    - OTU4
    - Definition of the ODU multiplexing hierarchy including the types of Low Order ODU supported in the hierarchy.
    - Definition of cross connect restrictions such that a single model can define the restricted case of a muxponder to the general case of an any to any non-blocking switch through the definition of switching pools
    - Definition of port and slot capabilities to define what interface types are supported by the device
    - Definition of port group restrictions to define possible dependencies on how ports can be configured
    - Definition of provisioned port groups to define which port groups are supported
    - Definition of ODU protection groups to define supported protection switching schemes and how they can be configured
    - Definition of Tandem Connection Monitoring (TCM) to define end-to-end path monitoring of arbitrary sub-network connections, as well as, the levels of tandem connection monitors supported.

Please refer to the Open ROADM MSA specification found at [2],[3] for the Muxponder functional specifications (“W MUXP functional” tab) identifying the supported Muxponder types, Switchponder functional specifications (“W SWITCH functional” tab) identifying the supported Switchponder types and Switchponder specific PMs and Alarms (“W PM Spec - SWITCH” tab and “W ALM-Spec - SWITCH” tab respectively).

3.2.4.3 Regenerator

A regenerator is used if the signal quality falls below a certain threshold requiring 3R regeneration (Re-amplification, Re-shaping, Re-timing). There are two types of regenerator setups, bi-directional (both directions regenerated in one logical regenerator) or uni-directional (a logical regenerator required for each transmission direction). A bi-directional regenerator (refer to Section 4.8.7.1) is the preferred regenerator implementation, but based on service provider demands (e.g. generation of optical regenerator equipment used, administrative and organizational), vendors can also offer the Uni-directional type (refer to Section 4.8.7.2). Should a wavelength change be needed, the advantage of a bi-directional regenerator is that recoloring is supported natively. While uni-directional regenerators could be built to support recoloring as well, bi-directional regenerators also support other features natively, such as OTU BDI (Backward Detect Indication) and GCC0 (General Communications Channel). Back-to-back transponders (refer to Section 4.8.7.3) can also be used as bi-directional regenerators. Regenerators are supported in the same optical nodes as transponders, muxponders, and switchponders.

The MSA does not define implementation details such as the number of circuit packs, form-factor, etc.. The MSA defines:

- **API** using NETCONF interface with a YANG-based data model that abstracts the management, control and provisioning of multi-vendor regenerator devices
- **Line side, the single-wave (W) interface** have the same specifications and requirements as those defined for a Transponder with the Single-wave (W) interface defining the optical specifications for the full C-band tunable DWDM optical line interface that connects to a Wr add/drop port on the ROADM device
  - 100G Line-side pluggable
  - 200G-400G Line-side pluggable. For OpenROADM optical specifications [3].

3.2.5 Pluggable Optics

The Open ROADM MSA defines the use of standards-based pluggable optics for Xponder devices. Xponder Line interfaces must use standard CFP-DCO, CFP2-DCO or CFP2-ACO pluggable optics that conform to the “W” optical specification. Xponder Client interfaces on the Transponder must use standard QSFP28 or QSFP56-DD pluggable optics, while client interfaces on the Muxponder and Switchponder must use standard SFP or SFP+ or QSFP28 or QSFP56-DD pluggable optics[3].

[3] the Open ROADM MSA does not dictate nor preclude the use of pluggable optics to perform other functions within the ROADM or Xponder device

3.2.5.1 Third Party Pluggable Optics

Third party pluggable refer to pluggable from one vendor being used within another vendor's equipment (host-CP). Third party pluggables can be configured and used within Open ROADM equipment, but cannot be guaranteed to provide all services supported by the pluggable.

3.2.5.2 External Pluggable Optics

External pluggable optics refer to pluggable optics hosted directly on the end user equipment, such as routers. In this case, the routers would be directly connected to the ROADM with no intermediate xponders.

Some management aspects may require further investigations. Foreseen limitations are described in Section 4.9.

3.3 OPEN INTERFACES

3.3.1 W (Single-Wavelength Interface)

The Single-Wavelength interface "W" specifies optical specifications for the full C-band tunable DWDM optical line interface of the Xponder that connects to a Wr add/drop port on the ROADM device and interoperability between xponders. The Optical specification outlines a minimum number of parameters for framing and bit ordering to enable the interoperability between different hardware vendors (see W tabs in the Open ROADM MSA Specification spreadsheet). Figure 7: Xponder Interfaces illustrates a functional representation.

3.3.2 Wr (Single-Wavelength Interface for ROADM Add/Drop Ports)

The Single-Wavelength port on the ROADM (also known as add/drop port) is called "Wr". Here, a Single-Wavelength output "W" from an Xponder is plugged into the ROADM (see MW-Wr tab in the Open ROADM MSA Specification spreadsheet). Figure 5: ROADM Interfaces illustrates a functional representation.

3.3.3 MW (Multi-Wavelength Interface)

The interoperability of different vendor's ROADMs is guaranteed by defining the Multi-Wavelength interface "MW". The Optical Specification sheet from OpenROADM.org contains the minimum interoperability specifications with some performance metrics of the ROADM (see MW-MW and MW-Wr tabs in the Open ROADM MSA Specification spreadsheet). The MW interface includes an Optical Supervisory Channel (OSC) that provides LLDP-based topology information, supports laser safety, and provides MCN/DCN reach-through to remote ROADM nodes (refer Section 3.3.5 for more details on the OSC). Figure 5: ROADM Interfaces illustrates a functional representation.

3.3.4 MWi (Multi-Wavelength Interface for ILAs)

The interoperability of different vendor's ILAs and ILAs to ROADMs is guaranteed by defining the Multi-Wavelength interface "MWi". The Optical Specification sheet from OpenROADM.org contains the minimum interoperability specifications with some performance metrics of the ILA (see MWi tabs in the Open ROADM MSA Specification spreadsheet). Figure 6: In-line Amplifier Interfaces illustrates a functional representation.

3.3.5 OSC (Optical Supervisory Channel)

The optical supervisory channel is part of the MW interoperability. The MSA has defined 1000BASE-LX and 100M (long spans) optical specifications and a simple Ethernet wayside channel (see OSC tabs in Open ROADM MSA Specification spreadsheet). Aside from the mandatory safety automatic power shutdown control loop (see Laser safety tab in the Open ROADM MSA Specification spreadsheet), no control loops are running between two adjacent ROADMs. All control loops are abstracted into the centralized controller. There are, however, a few local link control loops running on the device, such as transient control (see Local Control tab in the Open ROADM MSA Specification spreadsheet). Figure 5: ROADM Interfaces and Figure 6: In-line Amplifier Interfaces illustrate a functional representation.

4 DEVICE OBJECT MODEL HIGH LEVEL DESCRIPTION

The Open ROADM MSA Device Object Model is shown in Figure 8.

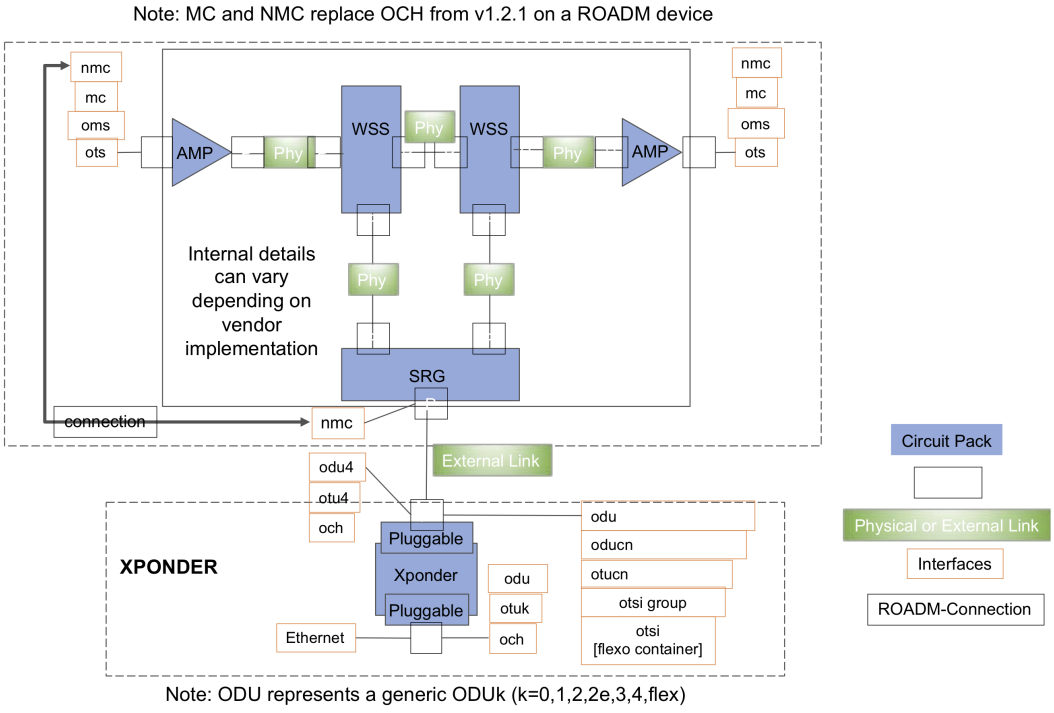

Figure 8: Open ROADM Device Object Model.

The specific objects of the Device Model to abstract the implementation of the ROADM, ILA, Xponder and external pluggable devices are shown in Figure 9. Each of these abstractions is described in the following subsections.

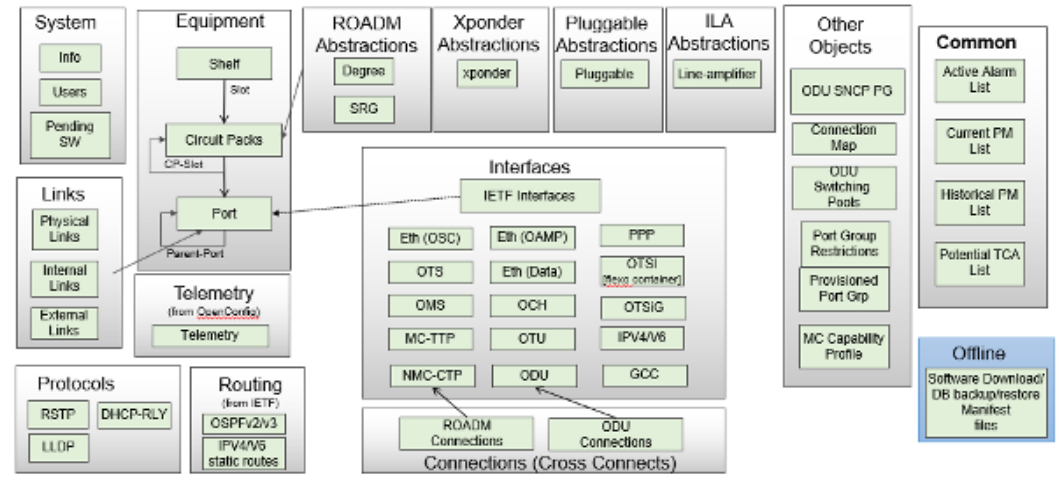

Figure 9: Open ROADM Device Object Model Abstractions.

4.1 System Model

Most system level attributes are contained in the info container (Tree 1)[4]. That includes several identification and planning attributes for the NE as the node id, the node number, the node type and sub-type, the office location (CLLI), vendor/model/serial-id (particularly relevant during ZTP), the mac address, the software version and build, the Open ROADM MSA version supported by the NE, the template in use in this node, the current time in the node, and the geolocation information. The Info container also describes some of the capabilities related to the maximum number of Degrees and SRGs supported, and the maximum number of historical PM bins for 15min and 24hr granularity.

[4]In earlier versions of the device model, IP data (IPv4 and IPv6) was included here in the info group. IP data is now located in dedicated IP groups within the interfaces section and in the routing section of the model

```
+--rw info
|  +--rw node-id?
|  +--rw node-number?
|  +--rw node-type?
|  +--rw node-subtype?
|  +--rw clli?
|  +--ro vendor
|  +--ro model
|  +--ro serial-id
|  +--ro macAddress?
|  +--ro softwareVersion?
|  +--ro software-build?
|  +--ro openroadm-version?
|  +--rw template?
|  +--ro current-datetime?
|  +--rw lifecycle-state?
|  +--ro recovery-mode?
|  +--rw geoLocation
|  |  +--rw latitude?
|  |  +--rw longitude?
|  +--rw netconf
|  |  +--rw idle-timeout?
|  +--rw ssh
|  |  +--rw client-alive-interval?
|  |  +--rw client-alive-max-count?
|  +--ro max-degrees?
|  +--ro max-srgs?
|  +--ro max-num-bin-15min-historical-pm?
|  +--ro max-num-bin-24hour-historical-pm?
```

YANG sub-tree 1: Info sub-tree (MSA version 13.1)

The pending-sw container (Tree 2) includes a software version, the validation timer and the activation-date-time. This is all read only information in use during the software update of the node.

```
+--ro pending-sw
|  +--ro sw-version?
|  +--ro sw-validation-timer?
|  +--ro activation-date-time?
```

YANG sub-tree 2: pending-sw sub-tree (MSA version 13.1)

The database info container (Tree 3) includes the last restore time, the rollback timer and an activation time. This is all read only information.

```
+--ro database-info
|  +--ro last-restored-time?
|  +--ro rollback-timer?
|  +--ro activation-date-time?
```

YANG sub-tree 3: database-info sub-tree (MSA version 13.1)

4.2 Equipment Model

The equipment model (Figure 10: Equipment Model) includes information about shelves (with associated slots) and circuit packs (with associated ports and CP-slots). Shelf are located in a RU or shelf position in a rack/bay, note however that rack are not part of the equipment model. While circuit packs can be located in a slot of a shelf or in a CP-slot of another circuit pack.

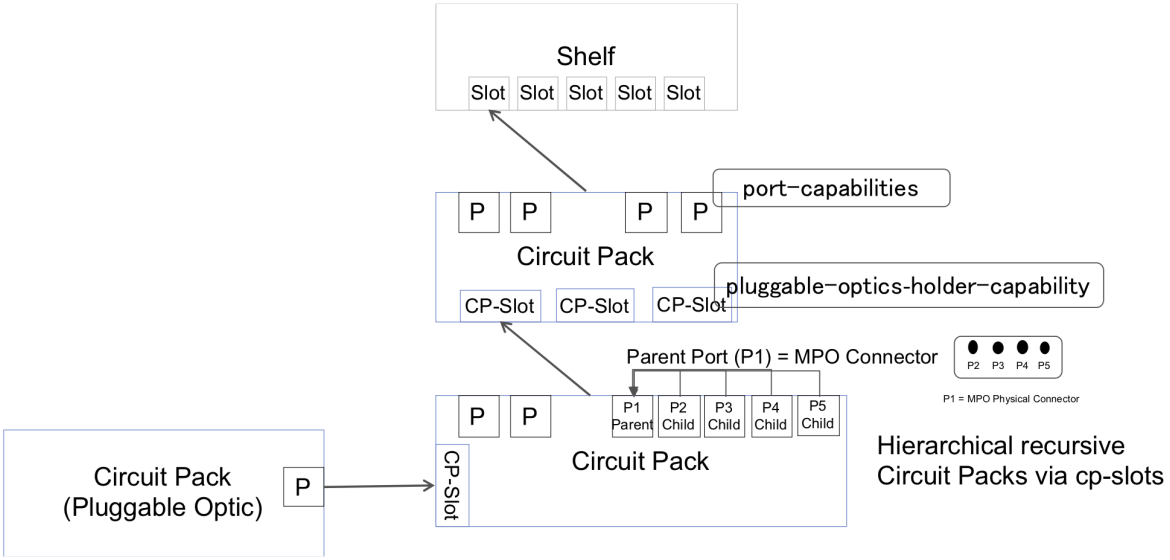

Figure 10: Equipment Model.

4.2.1 Shelf Model

Shelves attributes are contained in the Shelf containers (Tree 4: Shelf sub-tree). Shelves containers are organized in a list: the device model will include a shelf container for each shelf in the node.

```
+--rw shelves* [shelf-name]
|  +--rw shelf-name
|  +--rw shelf-type
|  +--rw rack?
|  +--rw shelf-position?
|  +--rw lifecycle-state?
|  +--rw administrative-state
|  +--ro vendor
|  +--ro model
|  +--ro serial-id
|  +--ro type?
|  +--ro product-code?
|  +--ro manufacture-date?
|  +--ro clei?
|  +--ro hardware-version?
|  +--ro operational-state
|  +--rw equipment-state?
|  +--ro is-physical
|  +--ro is-passive
|  +--ro faceplate-label
|  +--rw user-description?
|  +--rw due-date?
|  +--ro slots* [slot-name]
|     +--ro slot-name
|     +--ro label?
|     +--ro provisioned-circuit-pack?
|     +--ro slot-status?
```

YANG sub-tree 4: Shelf sub-tree (MSA version 13.1)

4.2.2 Circuit-Pack Model

The Shelf container includes a shelf name used as unique identifier for each shelf.
It also includes the shelf-type, a vendor-defined unique string, and the location of the shelf expressed as rack/bay identification and the shelf position in the rack/bay. This container includes several planing attributes including: administrative, lifecycle, operational and equipment states, as well as the 'is-physical' and the 'is-passive' flags, and the due-date.
Identification attributes for the shelf are also here: shelf vendor, model, serial id, type, product code, manufacture date,

```
+--rw circuit-packs* [circuit-pack-name]
|  +--rw circuit-pack-type
|  +--rw circuit-pack-product-code?
|  +--rw third-party-pluggable?
|  +--rw circuit-pack-name
|  +--rw lifecycle-state?
|  +--rw administrative-state
|  +--ro vendor
|  +--ro model
|  +--ro serial-id
|  +--ro type?
|  +--ro product-code?
|  +--ro manufacture-date?
|  +--ro clei?
|  +--ro hardware-version?
|  +--ro operational-state
|  +--ro circuit-pack-category
|  |  +--ro type
|  |  +--ro extension?
|  +--rw equipment-state?
|  +--rw circuit-pack-mode?
|  +--rw shelf
|  +--rw slot
|  +--rw subSlot?
|  +--ro is-pluggable-optics
|  +--ro is-physical
|  +--ro is-passive
|  +--ro faceplate-label
|  +--rw user-description?
|  +--rw due-date?
|  +--rw parent-circuit-pack
|  |  +--rw circuit-pack-name?
|  |  +--rw cp-slot-name?
|  +--ro cp-slots* [slot-name]
|     +    ...
|  +--ro software-load-version?
|  +--ro circuit-pack-features* []
|  |  +--ro feature
|  |     +--ro description
|  |     +--ro boot?
|  |     +--ro activated
|  |     +--ro is-service-affecting
|  |     +--ro activate-timeout
|  +--ro circuit-pack-components* []
|  |  +--ro component
|  |     +--ro name
|  |     +--ro boot?
|  |     +--ro current-version
|  |     +--ro version-to-apply?
|  |     +--ro is-service-affecting
|  |     +--ro activate-timeout
|  +--rw ports* [port-name]
|     +    ...
```

YANG sub-tree 5: Circuit-Pack sub-tree (MSA version 13.1)

CLEI, HW version, faceplate label and the user description of the shelf.
In the shelf container is also reported a list of the shelf's slots, the model will include a slot container for each slot in the shelf.
Each slot is uniquely identified by a slot name. An optional label can also be defined, and, when a circuit-pack is provisioned in the slot, the circuit-pack name is reported in the slot container. The slot container also report the status of the slot that is

used to identify if the slot is empty or installed and if the provisioned CP is a match with the installed CP. (Tree 4)
Circuit-Pack attributes are contained in the Circuit-Pack containers (Tree 5). Circuit packs containers are organized in a list: the device model will include a Circuit-Pack container for each Circuit-Pack in the node.
The Circuit-Pack container includes a circuit pack name used as unique identifier for each CP.
, It also includes the circuit-pack-product-code, the circuit-pack-type a vendor-defined unique string, and the location of the CP expressed as shelf identification, the CP position in the Shelf and, optionally, for CP slotted inside other CP, the sub-slot; for CP slotted in other CP, the model define a dedicated sub-container called 'parent-circuit-pack' including the parent CP identifier, as 'circuit-pack-name' and the 'cp-slot-name'
. This container includes several planing attributes including: administrative, lifecycle, operational and equipment states, as well as the 'is-pluggable-optics', the 'third-party-pluggable', the 'is-physical' and the 'is-passive' flags, and the due-date; the circuit-pack-category, a sub container containing the CP type and extension; the circuit-pack-mode
Identification attributes for the shelf are also here: vendor, model, serial-id, type, product code, manufacture date, CLEI, HW version, faceplate label and the user description of the CP. The CP attribute software-load-version, and the lists 'circuit-pack-features' and 'circuit-pack-components' are used to read and manage software load and firmware upgrade on the CP.
In the CP container are also reported a list of the CP slots, the model will include a cp-slot container for each slot in the CP, and a list of the CP ports, the model will include a port container for each port in the CP.
The CP-slots and the Ports containers are sub containers of the Circuit Pack container, however to are reported later in dedicated section (Ports are in Section 4.2.3, CP-slots are in Tree 28).

4.2.2.1 Third-party-pluggable attribute

The third-party-pluggable attribute is set only for pluggable from one vendor being used within another device vendor's equipment(host). Auto-provisioning of third party pluggable circuit-packs is allowed but may not be supported by a device vendor. Equipment vendors which supply OSC specific pluggables for both OSC applications do not need to support third party pluggables for OSC applications. However, If an equipment vendor does not supply pluggables for all the required OSC network applications, third party support for OSC pluggables is expected, as well as Auto-provisioning of third-party OSC pluggables and the circuit-packs/circuit-pack-type and circuit-packs/circuit-pack-product-code (optional) attributes will need generic values to be filled in by the device, which are not specific to either OSC network application.
DWDM pluggables are not required to be supported as third party pluggables. A request to either set the circuit-packs/third-party-pluggable attribute for a DWDM pluggable or auto-provision a third party DWDM pluggable can be denied by the device.
When the circuit-packs/third-party-pluggable attribute is present:

- This indicates that the physical pluggable circuit-pack and its parent-circuit-is pack are not from the device same vendor
- Any pluggable which can be determined to be compatible with the provisioned part number will not result in a mismatch condition

When the circuit-packs/third-party-pluggable attribute is absent:

- This indicates that the physical pluggable circuit-pack is from the device vendor
- Normal mismatch detection and configuration rules will apply

4.2.3 CP ports

CP-Ports attributes are contained in the Port containers inside the CP containers (Tree 6, Tree 7). The device model will include a port container for each port in the CP. The port container includes a port-name used as unique identifier for each port.

```
+--rw ports* [port-name]
   +--rw port-name
   +--rw supporting-port-list* [index]
   |  +--rw index
   |  +--rw circuit-pack-name
   |  +--rw port-list*
   +--rw port-type?
   +--rw port-qual?
   +--ro port-wavelength-type?
   +--ro port-direction
   +--ro compliance-codes*
   ...
```

YANG sub-tree 6: Ports sub-tree (1/2 - continues below)

```
+--rw ports* [port-name]
   ...
   +--ro is-physical
   +--ro pm-capable
   +--ro alarm-capable
   +--ro faceplate-label   +--rw user-description?
   +--rw circuit-id?
   +--rw lifecycle-state?
   +--rw administrative-state?
   +--ro operational-state
   +--ro operational-state-rx?
   +--ro operational-state-tx?
   +--rw logical-connection-point?   +--ro partner-port
   |  +--ro circuit-pack-name?
   |  +--ro port-name?
   +--ro parent-port
   |  +--ro circuit-pack-name?
   |  +--ro port-name?
   +--ro interfaces* []
   |  +--ro interface-name?
   +--ro mc-capability-profile-name*
   +--rw interconnect-loss-alarm
   |  +--rw user-threshold?
   |  +--ro threshold-default?
   +--rw roadm-port
   |  +--ro port-power-capability-min-rx?
   |  +--ro port-power-capability-min-tx?
   |  +--ro port-power-capability-max-rx?
   |  +--ro port-power-capability-max-tx?
   +--rw transponder-port
   |  +--ro port-power-capability-min-rx?
   |  +--ro port-power-capability-min-tx?
   |  +--ro port-power-capability-max-rx?
   |  +--ro port-power-capability-max-tx?
   +--rw otdr-port
   |  +--rw launch-cable-length?
   |  +--rw port-direction?
   +--rw ila-port
   |  +--ro port-power-capability-min-rx?
   |  +--ro port-power-capability-min-tx?
   |  +--ro port-power-capability-max-rx?
   |  +--ro port-power-capability-max-tx?
   +--rw roadm-monitor-port
   |  +--ro center-frequency?
   |  +--ro roadm-connection*
   +--ro port-capabilities
      +    ...
```

YANG sub-tree 7: Ports sub-tree (2/2)

4.2.3.1 MPO Connector Model

Referring to Figure 10: Equipment Model, the model for a physical MPO connector includes a Parent port (P1) and associated Child sub-ports (P2. . .P5) for each fiber/electrical lane of the MPO cable.

4.2.4 Circuit-Packs and Ports names

A circuit pack's name is unique within the node, including circuit-packs contained in other circuit-packs. However, the port's port-name under the circuit-pack are unique only within the context of the circuit pack's circuit-pack-name. Some management systems require a node-wide unique name for all entities (e.g., for alarming purposes). The circuit-pack-name and port-name can be concatenated with the "##" sequence in the management systems:

- circuit-pack-name = "cpA"
- port-name = "portB"

- Concatenate name for the port would be “cpA##portB”

Note: the circuit-pack-name and the port-name should not contain the “##” pattern to allow a management system to separate the circuit-pack-name and port-name from the concatenated node-wide unique port-name.

4.3 Links Model

Links represent connections between circuit pack ports: each link container define a source port and a destination port. Ports are identified in the links in term of node (only external-link type requires to identify the node), CP identifier, and port identifier.
Links containers are organized in lists: the device model will include a link container for each link in the node, and each link container includes a link name used as unique identifier for the link. Referring to (Figure 11: Links Model), three types of links are modeled in a device:

- **Physical Links** they represent fibers or electrical cables connecting two ports in circuit packs belonging to the same node. They are provisioned during ZTP. Tree 8.
- **External Links** are the fibers between circuit packs/ports on one node and circuit pack/port on another node. They correspond to ROADM to ROADM link, Transponder to ROADM nodes, or External Pluggable to ROADM nodes. Tree 9.
- **Internal Link** reflect the connectivity within each circuit pack or group of circuit packs connected via a backplane. Internal links are read only object that are generated automatically by the device. Tree 10

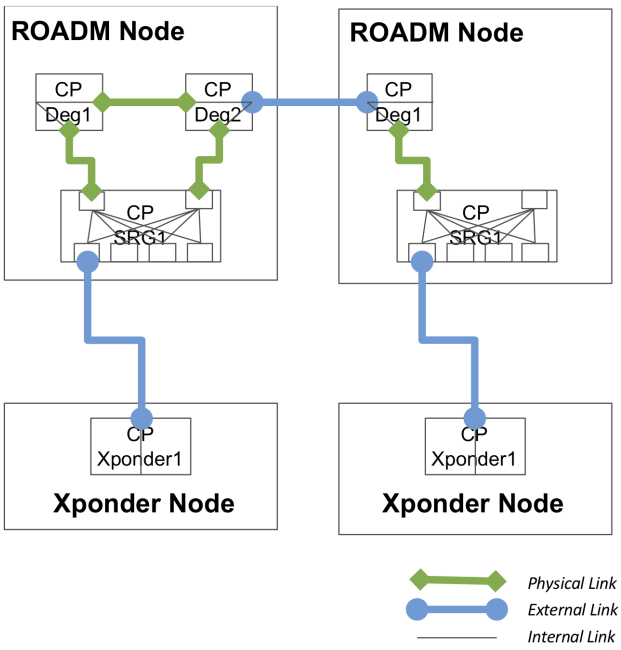

Figure 11: Links model.

```
+--rw physical-link* [physical-link-name]
|  +--rw physical-link-name
|  +--rw is-physical?
|  +--rw user-description?
|  +--rw source
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  +--rw destination
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  +--rw lifecycle-state?
```

YANG sub-tree 8: Physical Link

```
+--rw external-link* [external-link-name]
|  +--rw external-link-name
|  +--rw source
|  |  +--rw node-id
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  +--rw destination
|     +--rw node-id
|     +--rw circuit-pack-name
|     +--rw port-name
```

YANG sub-tree 9: External Link

```
+--ro internal-link* [internal-link-name]
|  +--ro internal-link-name
|  +--ro source
|  |  +--ro circuit-pack-name
|  |  +--ro port-name
|  +--ro destination
|     +--ro circuit-pack-name
|     +--ro port-name
```

YANG sub-tree 10: Internal Link

4.4 Interfaces Model

Interfaces represent the facilities associated with port objects on the device. Refer to

- ROADM Interfaces: Figure 12
- ILA Interfaces: Figure 13
- Xponder Interfaces: Figure 14
- Other interfaces are present in the Open ROADM model, including IPv4,IPv6, GCC, and PPP.

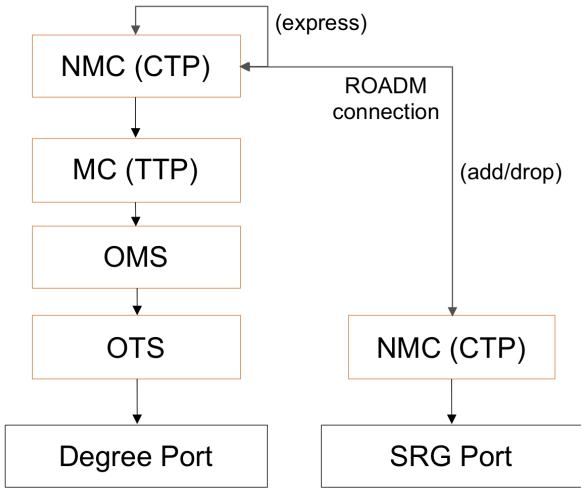

Figure 12: ROADM interfaces.

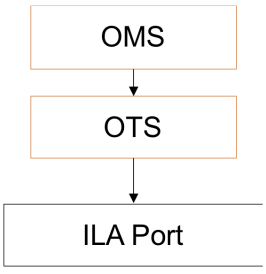

Figure 13: ILA interfaces.

Interface containers are organized in lists: the device model will include an interface container for each interface in the node.
Interface containers have a part common to all interface type (Tree 11) that includes an interface name used as unique identifier for the interface, an optional description string, the identification of the interface type, and an optional a circuit ID that can be defined for interfaces serving exclusively a single service..
The interface container also includes the life cycle state, the administrative state, and the operational state (a single operational state for bidirectional interfaces or a RX / TX state for unidirectional interfaces).
Then, each interface is either supported by a port or by another interface, the supporting port or the supporting interface is identified in the interface container.

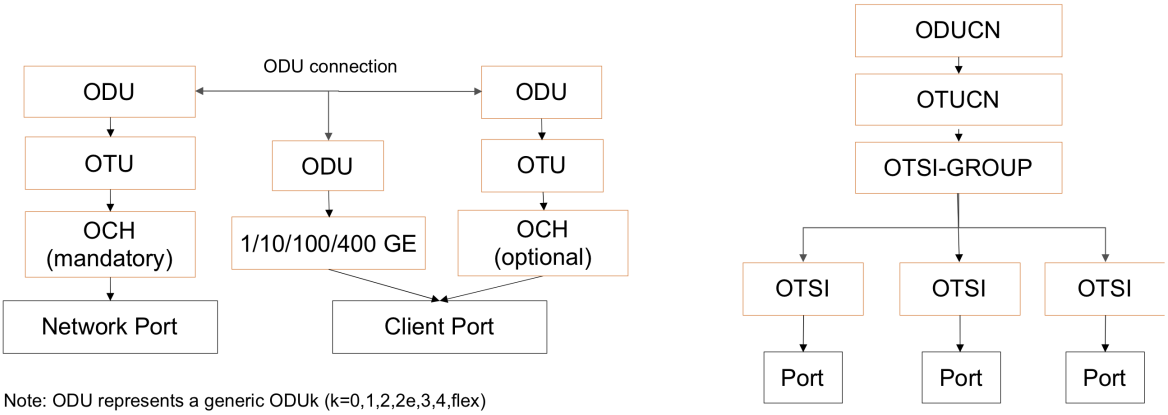

Figure 14: Xponder interfaces.

```
+--rw interface* [name]
|  +--rw name
|  +--rw description?
|  +--rw type
|  +--rw lifecycle-state?
|  +--rw administrative-state
|  +--ro operational-state
|  +--ro operational-state-rx?
|  +--ro operational-state-tx?
|  +--rw circuit-id?
|  +--rw supporting-circuit-pack-name?
|  +--rw supporting-port?
|  +--rw supporting-interface-list*
|  +--rw common-functions
|  |  +--ro protection-profile-name*
 ...
```

YANG sub-tree 11: Interface common part sub-tree

For some interface type, the container will include an extension specific for the interface type. Not all interface types have an extension

4.4.1 Ethernet Interfaces Model

Ethernet Interfaces containers have the 'ethernet' sub-container (Tree 12)

```
+--rw interface* [name]
 ...
|  +--rw ethernet
|  |  +--rw speed?
|  |  +--rw eth-function?
|  |  +--rw fec?
|  |  +--rw egress-consequent-action?
|  |  +--rw duplex?
|  |  +--rw auto-negotiation?
|  |  +--rw enable-laser?
|  |  +--ro laser-status?
|  |  +--ro curr-speed?
|  |  +--ro curr-duplex?
|  |  +--ro max-frame-size?
|  |  +--rw subrate-eth-sla!
|  |  |  +--rw committed-info-rate
|  |  |  +--rw committed-burst-size
|  |  +--ro no-oam-function?
|  |  +--ro no-maint-testsignal-function?
|  |  +--rw parent-flexo-allocation!
|  |  |  +--rw trib-port-number
|  |  |  +--rw iid*
|  |  +--rw maint-testsignal
|  |  |  +---x clear-diagnostics
|  |  |  |  +---w input
|  |  |  |  |  +---w type?
|  |  |  |  +--ro output
|  |  |  |     +--ro status
|  |  |  |     +--ro status-message?
|  |  |  +--rw enabled?
|  |  |  +--rw testPattern
|  |  |  +--rw type?
|  |  |  +--ro inSync?
|  |  |  +--ro seconds
|  |  |  +--ro bitErrors?
|  |  |  +--ro bitErrorRate?
|  |  +--rw maint-loopback
|  |     +--rw enabled?
|  |     +--rw type?
      ...
```

YANG sub-tree 12: Eth Interface specific sub container

4.4.2 GCC FCC Interfaces Model

FCC and GCC Interfaces containers have respectively the 'fcc' or the 'gcc' sub-container (Tree 13)

```
+--rw interface* [name]
 ...
|  +--rw fcc
|  |  +--rw fcc-channel-type?
|  +--rw gcc
|  |  +--rw gcc-channel-type?
 ...
```

YANG sub-tree 13: FCC and GCC Interface specific sub container

4.4.3 IP Interfaces Model

IpV4 Interfaces containers have the 'ipv4' and the 'ipv4-state' sub-container (Tree 14)

```
+--rw interface* [name]
 ...
|  +--rw ipv4!
|  |  +--rw enabled?
|  |  +--rw forwarding?
|  |  +--rw mtu?
|  |  +--rw dhcpc-enabled?
|  |  +--rw (address-type)?
|  |  |  +--:(numbered-address)
|  |  |  |  +--rw address* [ip]
|  |  |  |     +--rw ip
|  |  |  |     +--rw (subnet)
|  |  |  |        +--:(prefix-length)
|  |  |  |        |  +--rw prefix-length?
|  |  |  |        +--:(netmask)
|  |  |  |           +--rw netmask?
|  |  |  +--:(unnumbered-address)
|  |  |     +--rw address-src?
|  |  +--rw proxy-arp!
|  |     +--rw enabled?
|  |     +--rw proxy-subnet* [ipv4-subnet]
|  |        +--rw ipv4-subnet
|  +--ro ipv4-state!
|  |  +--ro forwarding?
|  |  +--ro mtu?
|  |  +--ro address* [ip]
|  |     +--ro ip
|  |     +--ro (subnet)
|  |     |  +--:(prefix-length)
|  |     |  |  +--ro prefix-length?
|  |     |  +--:(netmask)
|  |     |     +--ro netmask?
|  |     +--ro origin?
 ...
```

YANG sub-tree 14: IPv4 and IPV4-status specific sub container

IPV6 Interfaces containers have the 'ipv6' and the 'ipv6-state' sub-container (Tree 15)

```
+--rw interface* [name]
 ...
|  +--rw ipv6!
|  |  +--rw enabled?
|  |  +--rw forwarding?
|  |  +--rw mtu?
|  |  +--rw dhcpc-enabled?
|  |  +--rw (address-type)?
|  |  |  +--:(numbered-address)
|  |  |  |  +--rw address* [ip]
|  |  |  |     +--rw ip
|  |  |  |     +--rw prefix-length
|  |  |  +--:(unnumbered-address)
|  |  |     +--rw address-src?
|  |  +--rw proxy-ndp!
|  |     +--rw enabled?
|  |     +--rw proxy-subnet* [ipv6-subnet]
|  |        +--rw ipv6-subnet
|  +--ro ipv6-state!
|  |  +--ro forwarding?
|  |  +--ro mtu?
|  |  +--ro address* [ip]
|  |  |  +--ro ip
|  |  |  +--ro prefix-length
|  |  +--ro origin?
|  |  +--ro status?
 ...
```

YANG sub-tree 15: IPv6 and IPV6-status specific sub container

4.4.4 OTS Interfaces Model

OTS Interfaces containers have the 'ots' sub-container (Tree 16)

```
+--rw interface* [name]
 ...
|  +--rw ots
|  |  +--rw fiber-type?
|  |  +--rw span-loss-receive?
|  |  +--rw span-loss-transmit?
|  |  +--rw ingress-span-loss-aging-margin?
|  |  +--rw eol-max-load-pIn?
 ...
```

YANG sub-tree 16: OTS Interface specific sub container

4.4.5 ROADM Interfaces Model

MC and NMC sub-containers are reported below in the descriptions of the ROADM model.

4.4.6 Xponders Interfaces Model

OCh, ODU, and OTU, as well as OTSI, and OTSI-Group sub-containers are reported below in the descriptions of the Xponder model.

4.5 Connectivity Matrices

4.5.1 Connection Map

Connection Maps containers model the connectivity for ROADMs, transponders, and regenerators.
Connection Maps are used to indicate all possible port-to-port connectivity between external ports of a ROADM or Xponder.
Connection Maps represent uni-directional connectivity between a source port and a list of destination ports.

Within a ROADM device, a Connection Map describes what Degrees and SRG external ports can be connected by the creation of a roadm-connection. This includes add, drop, express and turn back roadm connections.
For transponders and regenerators, the connectivity is fixed and no cross-connects are required.

```
+--ro connection-map* [connection-map-number]
| +--ro connection-map-number
| +--ro source
| | +--ro circuit-pack-name
| | +--ro port-name
| +--ro destination* [circuit-pack-name port-name]
| | +--ro circuit-pack-name
| | +--ro port-name
```

YANG sub-tree 17: Connection-map

4.5.2 Switching Pools

Switching Pools describe the connectivity associated within the ODU layer by providing the ODU connectivity between external ports of muxponders, and switchponders. Switching Pools represent bi-directional connectivity between ports. Non-blocking as well as blocking configurations are represented. Each set of ports within a non-blocking-list are not blocking. Multiple sets of non-blocking-list entries are blocking and provide an interconnect bandwidth constraint.

```
+--ro odu-switching-pools* [switching-pool-number]
|  +--ro switching-pool-number
|  +--ro switching-pool-type?
|  +--ro odu-connection-direction-capabilities?
|  +--ro non-blocking-list* [nbl-number]
|     +--ro nbl-number
|     +--ro interconnect-bandwidth-unit?
|     +--ro interconnect-bandwidth?
|     +--ro port-list* [circuit-pack-name port-name]
|     |  +--ro circuit-pack-name
|     |  +--ro port-name
|     +--ro pluggable-optics-holder-list*
|        +--ro circuit-pack-name
|        +--ro slot-name
+--ro eth-switching-pools* [switching-pool-number]
|  +--ro switching-pool-number
|  +--ro switching-pool-type?
|  +--ro non-blocking-list* [nbl-number]
|     +--ro nbl-number
|     +--ro interconnect-bandwidth-unit?
|     +--ro interconnect-bandwidth?
|     +--ro port-list* [circuit-pack-name port-name]
|     |  +--ro circuit-pack-name
|     |  +--ro port-name
|     +--ro pluggable-optics-holder-list* [circuit-pack-name slot-name]
|        +--ro circuit-pack-name
|        +--ro slot-name
```

YANG sub-tree 18: ODU switching-pool

Note: the *odu-connection-direction-capabilities* attribute (connection-direction_bi, connection-direction

_bi_and_uni) is used to advertise the additional support for uni-directional cross-connects (optional); bi-directional cross-connects (connection-direction_bi) must be supported by an Open ROADM MSA device.

A non-blocking *switching-pool-type* implies the presence of a single non-blocking-list and any port-name in a defined *port-list* or any *slot-name* in a defined *pluggable-optics-holder-list* may be freely cross connected without restriction up to the interface bandwidth. Each set of ports within a non-blocking-list are not blocking. Multiple sets of non-blocking-list entries are blocking and provide an interconnect bandwidth constraint. The non-blocking list can contain pluggable optic ports or slots that hold pluggable optics. An example of a non-blocking application is illustrated in Figure 15. In this example, the device supports the provisioning of cross connects between any of the ports illustrated. The device will indicate this through the *odu-switching-pools* where the defined pool will be of a non-blocking type and the *non-blocking-list* would have all ports $(P_1 - P_n)$ in it.

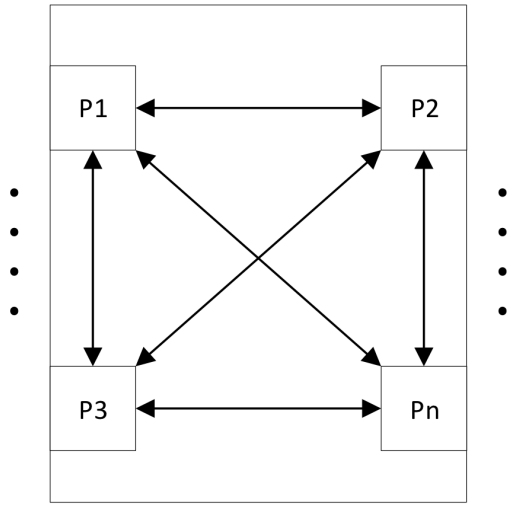

Figure 15: Non-Blocking Muxponder.

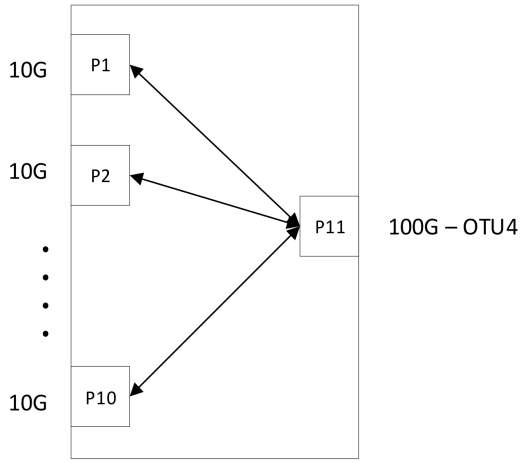

Figure 16: Blocking Muxponder.

The definition of a blocking switching pool implies there is some restriction as to what ports may be cross-connected. A blocking switching-pool-types has more than one non-blocking-list present in its definition. A simple example is that of a muxponder (refer to Figure 16) where the client ports may only be cross-connected to the line port. For the example, the Open ROADM device model for a 10x10G muxponder with an OTU4 line side interface would indicate a blocking switchingpool- type comprised of 10 non-blocking-lists with each such list having one unique client port and a common line port. For example $(P_1, P_{11}), (P_2, P_{11}), (P_3, P_{11}) \ldots (P_{10}, P_{11})$.

A further restriction of a muxponder may be to what tributary port number and tributary slot a given client port can be cross connected into the line interface. If such restrictions are present, they are indicated by the Open ROADM MSA device model as a part of the muxponder capabilities profile (*muxp-profile*) and identified by the *muxp-profile-name* within the *mpdr-client-restriction* container.

Within a given switching pool, the device may allow for cross connects to be made between two or more *non-blocking-list*. If such a capability is present, it may be constrained by the amount of bandwidth which can be connected between such *non-blocking-list*. Let's consider the example of a two-card set which provides a line side OTU4 interface and 10×10G client interfaces on each card. However, as opposed to the simple muxponder, each card is capable of non-blocking switching between all of its ports supporting hairpin connections. In addition, traffic can be cross connected between line and client-side ports on each card, but up to a limit of 100G of bi-directional traffic. This example is illustrated in Figure 17.

In this case, the device may present a blocking *switching-pool-type* comprised of two *non-blocking-lists* where each list contains all the ports on a given card. The bandwidth constraint between the two non-blocking- list is modeled by the *interconnect-bandwidth* and *interconnect-bandwidth-unit*. For example, this interconnect could represent an internal ODU4 link comprised of 80 OPU4 based tributary slots. In such a case, the total interconnect-bandwidth would be represented by $80 \times 1,301,683,217$ bp/s interconnect-bandwidth-units. The interconnect-bandwidth-unit in this case is the minimum ODTU4.ts rate. In a different case, the *interconnect-bandwidth* may not be quantized as in the example above. In such a case it can be appropriate to present the *interconnect-bandwidth* as a single unit of say 105,000,000,000 bps.

Switching pool BW is not modified by the device due to the provisioning of cross connects. It is the responsibility of the controller to maintain the usage of any interconnect bandwidth. Note that the usage of interconnect bandwidth can be derived from cross connect provisioning information.

The device is responsible for updating switching pool information as cards are added/deleted from the system or card provisioning updates the port information. Switching pools reflect the capabilities of the device to cross connect services between the currently configured port mode of operation. For example, the ports available may be altered through port group

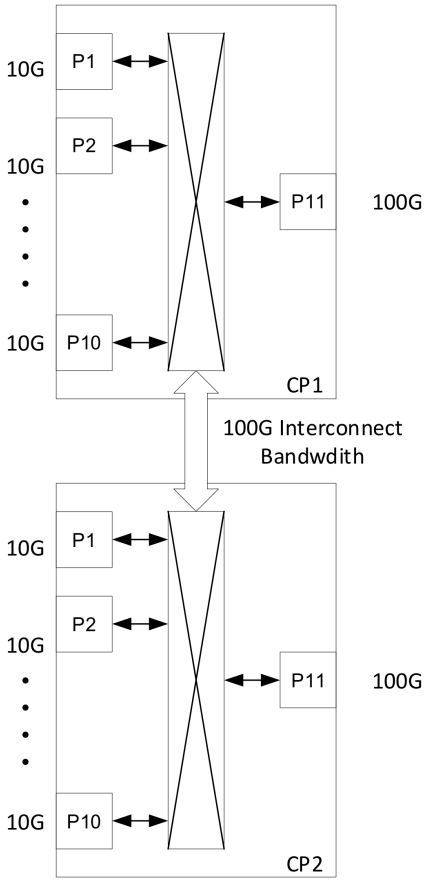

Figure 17: Switching Pool with Interconnect BW.

configuration. Switching pools reflect the capability to perform cross connection between both fixed ports and pluggable optics holders. For the case of the pluggable optics holder, the pluggable optics themselves do not need to be installed for the switching pool to identify the capability.

4.6 ROADM Model

The ROADM model includes abstractions for the Degree and Shared Risk Group (SRG) of the ROADM.
The descriptions of ROADM specific interfaces like the MC and NMC are reported in this section as they are relevant for the ROADM model.

4.6.1 Degree Abstraction

Degree defines the grouping of circuit packs that form a line degree. The Degree abstraction includes the ROADM degree number that uniquely identify each degree, the lifecycle state (planned, deployed, maintenance etc.), and an optional list of optical-operational mode profiles supported by the degree.
This container also identifies the list of all circuit packs that make up the degree, it indicates line side ports, and OTDR circuit pack port, as well as the flexgrid capabilities of the degree.

```
+--rw degree* [degree-number]
| +--rw degree-number
| +--rw lifecycle-state?
| +--ro max-wavelengths
| +--ro optical-operational-mode-profile-name*
| +--rw circuit-packs* [index]
| | +--rw index
| | +--rw circuit-pack-name
| +--rw connection-ports* [index]
| | +--rw index
| | +--rw circuit-pack-name
| | +--rw port-name
| +--rw otdr-port
| | +--rw circuit-pack-name?
| | +--rw port-name?
| +--ro mc-capability-profile-name*
```

YANG sub-tree 19: Degree capabilities

4.6.2 SRG Abstraction

Shared Risk Groups (SRG) define the grouping of circuit packs that form a colorless/directionless (CD) or colorless/directionless/contentionless (CDC) add/drop bank.
The SRG abstraction includes the SRG degree number that uniquely identify each SRG, the max number of add/drop ports for the CD or CDC architecture (SRG is a contention group for CD where a wavelength can only be dropped once with the SRG), lifecycle state (planned, deployed, maintenance etc.), maximum and currently provisioned number of the SRG add/drop ports, a wavelength duplication indication of CD or CDC, circuit packs list identifying all circuit packs that make up the SRG, and the flexgrid capabilities.

```
+--rw shared-risk-group* [srg-number]
| +--ro max-add-drop-ports
| +--ro current-provisioned-add-drop-ports
|  +--rw prov-max-add-drop-ports?
| +--rw srg-number
| +--ro optical-operational-mode-profile-name*
| +--rw lifecycle-state?
| +--ro wavelength-duplication
| +--rw circuit-packs* [index]
| | +--rw index
| | +--rw circuit-pack-name
| +--ro mc-capability-profile-name*
```

YANG sub-tree 20: MC capabilities of the SRG

4.6.3 Flexible Grid Capabilities

The Open ROADM MSA model has been extended to support flexible grid capabilities using Media Channel (MC) and Network Media Channel (NMC) constructs to allow for:

- One or more NMC per MC
- Mix of fixed grid and flexible grid capable ROADM hardware
- Mix of media channel widths

Note: The conversion from the fixed grid model to a flexgrid model occurred in Open ROADM MSA v2. Flexgrid capabilities are profile-based via an MC capabilities profile (*mc-capability-profile*) that is identified by the profile name (*profile-name*). An mc-capability-profile is defined as in the Tree 21.

```
+--ro mc-capability-profile* [profile-name]
| +--ro profile-name
| +--ro center-freq-granularity?
| +--ro min-edge-freq?
| +--ro max-edge-freq?
| +--ro slot-width-granularity?
| +--ro min-slots?
| +--ro max-slots?
```

YANG sub-tree 21: MC capability profile

MC capability profiles can be advertised against any port (*ports**), degree (*degree**), and SRG (*shared-risk-group**) (Tree 22). MC capability profiles are identified by the profile name (*mc-capability-profile-name*).

```
+--ro ports* [port-name]
| +--ro port-name
| +--
| +--ro mc-capability-profile-name*
....
+--ro degree* [degree-number]
| +--ro degree-number
| +--
| +--ro mc-capability-profile-name*
....
+--ro shared-risk-group* [srg-number]
| +--ro srg-number
| +--
| +--ro mc-capability-profile-name*
```

YANG sub-tree 22: Ports, Degree, SRG structure with mc-capability-profile

MC capabilities samples for fixed grid and flexible grid capable ROADM nodes are shown in Table 2: Sample MC Capabilities for ROADM Devices.

Node Type	slot-width-granularity	center-freq-granularity	min-slots	max-slots
Fixed grid	50 GHz	50 GHz	1	1
Flexible grid	12.5 GHz	6.25 GHz	3	384
Flexible grid	6.25 GHz	3.125 GHz	6	768

Table 2: MC Capabilities for ROADM Devices

The figures in Table 2: Sample MC Capabilities for ROADM Devices consider a total frequency range of 4800 GHz from 191.325 THz to 196.125 THz, corresponding to 96 channels for the 50GHz Fixed Grid case. These parameters are used by the controller to determine the wavelength provisioning flexibility that is supported by each port, degree, and SRG within the ROADM node.

4.6.3.1 Media Channel (MC) Interfaces (mc-ttp)

Media channels are terminated at every degree of a ROADM with a trail termination point, MC-TTP, which allows access to the NMC(s) contained within the MC. Media channels are not terminated/modeled on line amplifier nodes. MC-TTP are provisioned on the OMS interface (refer to Figure 12: ROADM Interfaces). An OMS interface can have one or more MC-TTPs and the frequency ranges of MC on a degree port cannot overlap. For ease of operations, the Open ROADM MSA model is restricted to the same MC size over a WDM span. MC-TTP interfaces are defined as in Tree 23.

```
+--rw interface* [name]
 ...
|  +--rw :mc-ttp
|  |  +--rw :min-freq?
|  |  +--rw :max-freq?
|  |  +--ro :center-freq?
|  |  +--ro :slot-width?
 ...
```

YANG sub-tree 23: MC-TTP interfaces

Media channels interfaces are created by provisioning the min/max frequency (THz). These are determine based on the MC capabilities provided by the ROADM. The MC must include the bandwidth of the NMC(s) and guard bands (dead bands) at the extremities (4 GHz).

$$\text{min-freq (THz)} = 193.1 + \frac{(\text{center-freq-granularity} \times n - \text{slot-width-granularity} \times m/2)}{1000} \quad (1)$$

$$\text{max-freq (THz)} = 193.1 + \frac{(\text{center-freq-granularity} \times n + \text{slot-width-granularity} \times m/2)}{1000} \quad (2)$$

where: *min-slots* $\leq m \leq$ *max-slots* and n is a positive or negative integer including 0. The value of n should be selected in order to guarantee for each media channel that the minimum frequency is $\geq$ 191.325 THz and the maximum frequency $\leq$ 196.125 THz. In particular for the Fixed Grid case $-35 \leq n \leq 60$; while for the Flex Grid cases n depends on the number and frequency width of the provisioned MCs. Referring to Figure 18, Media Channel (MC), slot width and center frequency being read only data are calculated according to these formulas:

$$\text{slot-width (GHz)} = \text{slot-width-granularity} \times m \quad (3)$$

$$\text{center-freq (THz)} = 193.1 + \frac{\text{center-freq-granularity}}{1000} \times n \quad (4)$$

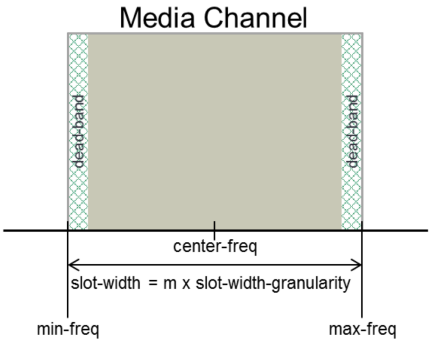

Figure 18: Media channel (MC)

When upgrading from Open ROADM MSA v1 fixed grid model to Open ROADM MSA v2 (or greater) flexgrid model, 100G fixed grid entities would need to convert from the OCH model in MSA v1 to the MC and NMC model in Open ROADM MSA v2. For the 100G fixed grid services upgraded from v1 to v2 or created in v2, the mc-ttp attributes would be set as follows:

- min-freq (THz) = 193.1 + (50 $\times n$ - 25) / 1000
- max-freq (THz) = 193.1 + (50 $\times n$ + 25) / 1000
- center-freq (THz) = 193.1 + 50 / 1000 $\times n$
- slot-width = 50 GHz
- where: $-35 \leq n \leq 60$.

4.6.3.2 Network Media Channel (NMC) Interfaces (nmc-ctp)

ROADM cross connections are done between NMC connection termination points, NMC-CTP. NMC-CTP interfaces are defined as follows:

```
+--rw interface* [name]
 ...
|  +--rw nmc-ctp
|  |  +--rw frequency?
|  |  +--rw width?
|  |  +--rw full-bandwidth-at-3dB?
|  |  +--rw full-bandwidth-at-10dB?
 ...
```

YANG sub-tree 24: NMC-CTP interface

Referring to Figure 19: Network Media Channel (NMC):

$$\text{frequency (THz)} = 193.1 + \frac{\text{center-freq-granularity}}{1000} \times n, \quad (5)$$

where n is a positive or negative integer including 0

$$\text{width (GHz)} = \text{slot-width-granularity} \times m\text{–}4 \times 2 \quad (6)$$

where min-slots $\leq m \leq$ max-slots and 4 GHz are considered for the dead bands. NMC-CTPs are a child of a MC-TTP or a child of a SRG port. An SRG port can have only one NMC-CTP while an MC-TTP can have one or more NMC-CTP. NMCs within an "MC" must not overlap and no dead-bands are needed between adjacent NMCs (refer to Figure 20: MC-TTP and MC-CTP Relationship). Figure 21: MC-TTP and NMC-CTP Relationship shows the relationship of the MC-TTP and NMC-

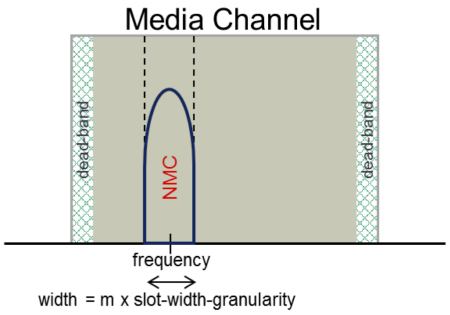

Figure 19: Network Media Channel (NMC)

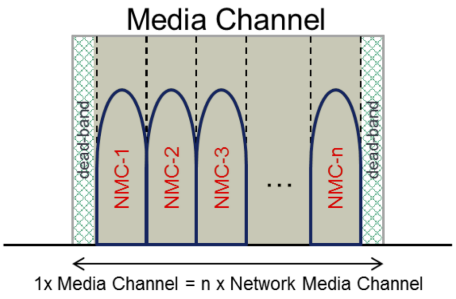

Figure 20: MC-TTP and MC-CTP Relationship

CTP to each other and how they relate to the ports of a degree and SRG. NMC-CTP are only used by the ROADM device;

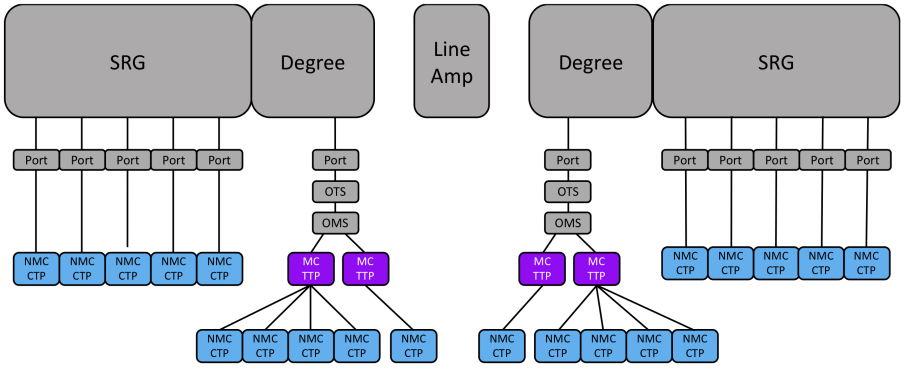

Figure 21: MC-TTP and NMC-CTP Relationship

xponder devices will continue to use OCh (100G) and/or OTSi (B100G) interfaces. The OCh/OTSi center frequency and signal bandwidth are not regulated by the slot-width-granularity or center-freq-granularity. The OCh/OTSi width (i.e. signal bandwidth + modulation guardbands) must be $\leq$ the NMC-width. When upgrading from MSA v1 to MSA v2 or greater, 100G fixed grid entities would need to convert from the OCh model in MSA v1 to the MC and NMC model in MSA v2. For the 100G fixed grid services upgraded from v1 to v2 or created in v2, the nmc-ctp attributes would be set as follows:

- frequency (THz) = 193.1 + $\frac{50}{1000} \times n$
- width = 40 GHz (Note: it was decided to use 40 GHz for the width instead of 42 GHz that the formula listed above would suggest for the fixed grid channels from v1.2.1)
- where: $-35 \leq n \leq 60$.

Note that for the OCh settings on transponders, the OCh frequency and width would have the same values as the nmc-ctp (e.g., OCh width = 40 GHz).

4.6.4 ROADM Cross-Connections

ROADM cross-connections (*roadm-connections*) are uni-directional cross-connects at the NMC (CTP) level. The flexgrid model uses NMC-CTP to NMC-CTP connections to create ROADM cross connects (refer to Figure 22: Flex-Grid NMC-CTP to NMC-CTP Cross-Connections). When cross connecting two NMC-CTPs, the *frequency* and *width* of the NMC-CTPs must match.

The *opticalControlMode* and the *target-output-power* for the channel are controlled by the ROADM cross-connections.

The following ROADM cross-connections types are supported:

- Add: Connects an NMC interface on SRG add port to an NMC interface on OMS TX interface. Applicable control modes are Off, Power with *target-output-power*, and gainLoss
- Drop: Connects an NMC interface on OMS RX interface to an NMC interface on SRG drop port. Applicable control modes are Off and Power (without *target-output-power* setting, the device to select the target power)
- Express: Connects an NMC interface on OMS RX to an NMC interface on OMS TX interface. Applicable control modes are Off, Power with *target-output-power*, and gainLoss
- Turn Back: Connects an NMC interface on SRG add port to an NMC interface on SRG drop port. Applicable control modes are Off and Power (without *target-output-power* setting, the device to determine the target power)

```
+--rw roadm-connections* [connection-name]
| +--rw connection-name
| +--rw opticalControlMode?
| +--rw target-output-power?
| +--rw source
| | +--rw src-if
| +--rw destination
| | +--rw dst-if
|  +--rw circuit-id?
```

YANG sub-tree 25: ROADM connections

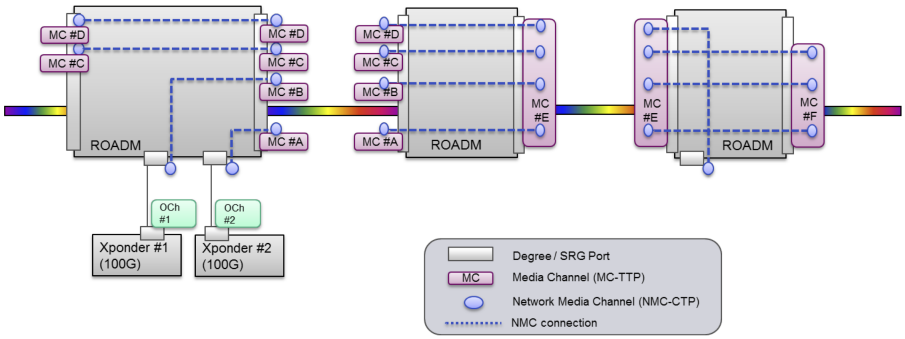

Figure 22: Flex-Grid NMC-CTP to NMC-CTP Cross-Connections

The ROADM device does not model NMC groups (NMCG). The controller is responsible for maintaining NMC groups and ensure all NMC's within the group are routed together.

4.6.5 Turn Back

The turn back functionality allows for traffic signals to be looped back on the SRG client side without transporting the signal through the ROADM transport network. This is shown in Figure 23. The traffic arrives on one SRG port and is looped back to another SRG port within the ROADM. The two SRG ports can be on the same SRG or on different SRGs.

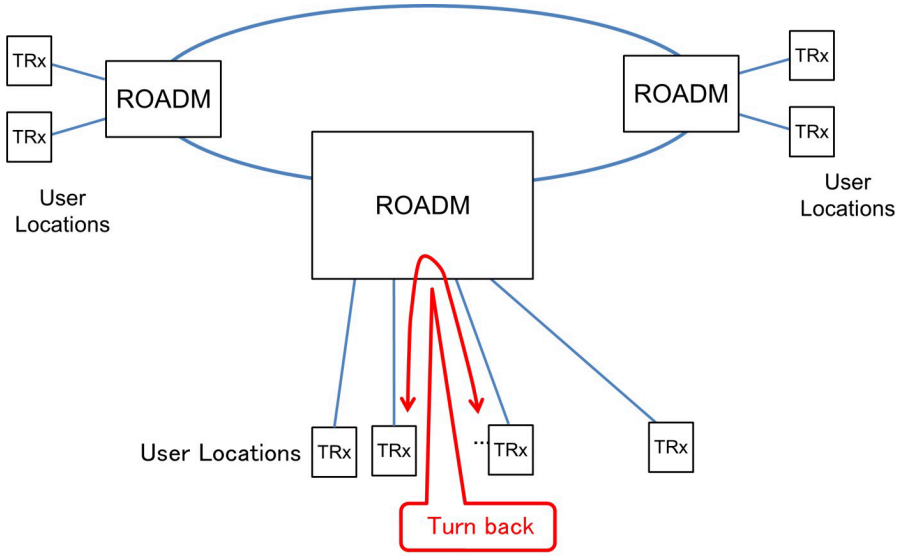

Figure 23: Turn back functionality in ROADM

The application for the turn back feature is to connect two client sites that are both homed to the same ROADM device. Examples of turn back applications are shown in Figure 24.
The turn back feature was developed in support of the Open All-Photonic Network Architecture from the Innovative Optical Wireless Network (IOWN) Global Forum [6].

4.6.5.1 Turn Back Implementations

There are two implementation use cases for the turn back functionality.

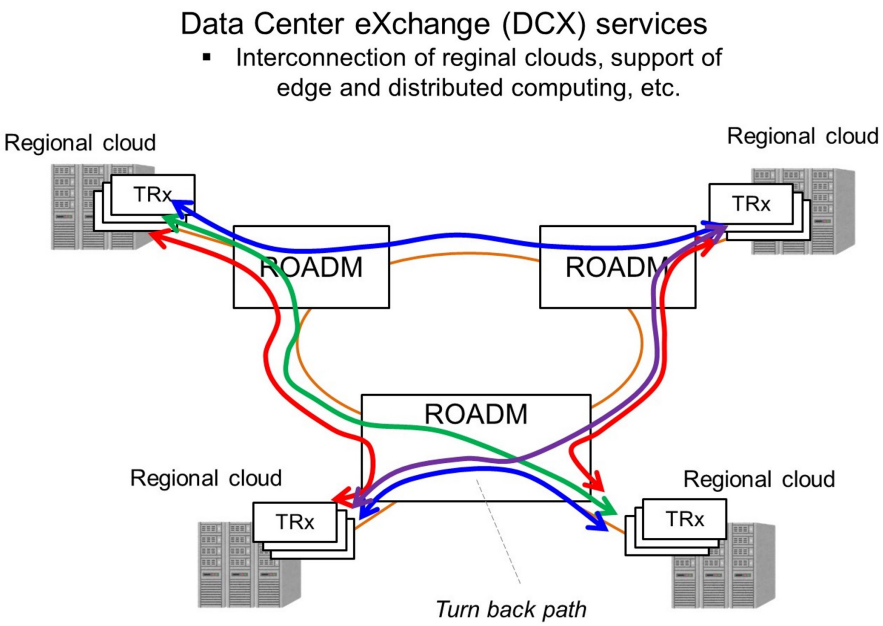

Figure 24: Applications for the Turn Back functionality

Implementation Use Case 1: Turn Back Modules The turn back functionality is realized by dedicated hardware modules. Some of the SRG network-side (internal) ports are connected to the turn back module. When a turn back connection is provisioned, the device programs the turn back module as required to loop the signal back to the appropriate SRGs. Figure 25 shows an example of a ROADM with a turn back module.

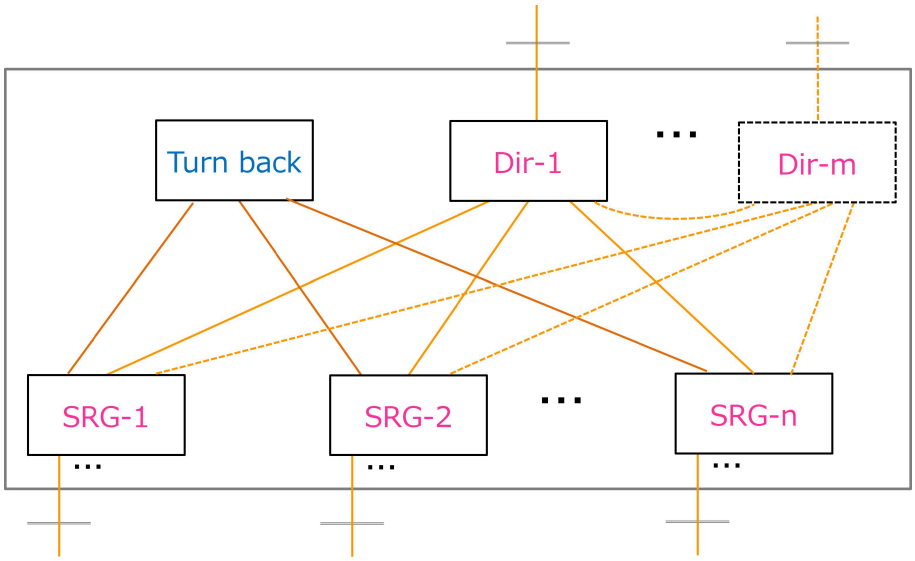

Figure 25: Turn back functionality implemented by dedicated hardware modules

To support the ability to turn back a signal to the same SRG module, each SRG would need to connect two SRG network-side ports to the turn-back module (e.g., to connect to the left-side and right-side of the turn backmodules). This is shown in Figure 26. Note: it is possible to implement the turn back modules by connecting two degree modules back-to-back.
Implementation Use Case 2: Fiber connections between two SRG network-side ports The turn back functionality is realized by connecting optical fibers between two SRG network-side ports. In this case, no special hardware for the turn back functionality is required. An example of this is shown in Figure 27.

4.6.5.2 Turn Back Provisioning

The provisioning for the turn back functionality would involve creating a ROADM connection between a source SRG client port and a destination SRG client port. The SRG client ports may be on the same SRG or on different SRGs. Two cross connects would be established for bi-directional services, one in each direction.
The device is responsible for the configuration of the turn back modules and/or SRG modules as neededto support the turn back connection.

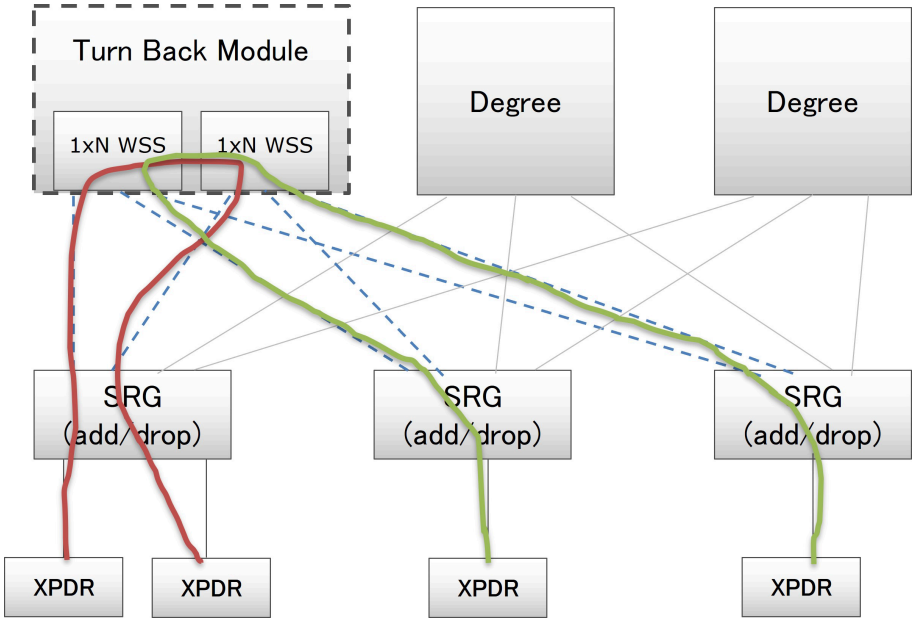

Figure 26: Turn back module supporting signal loop back to the same SRG

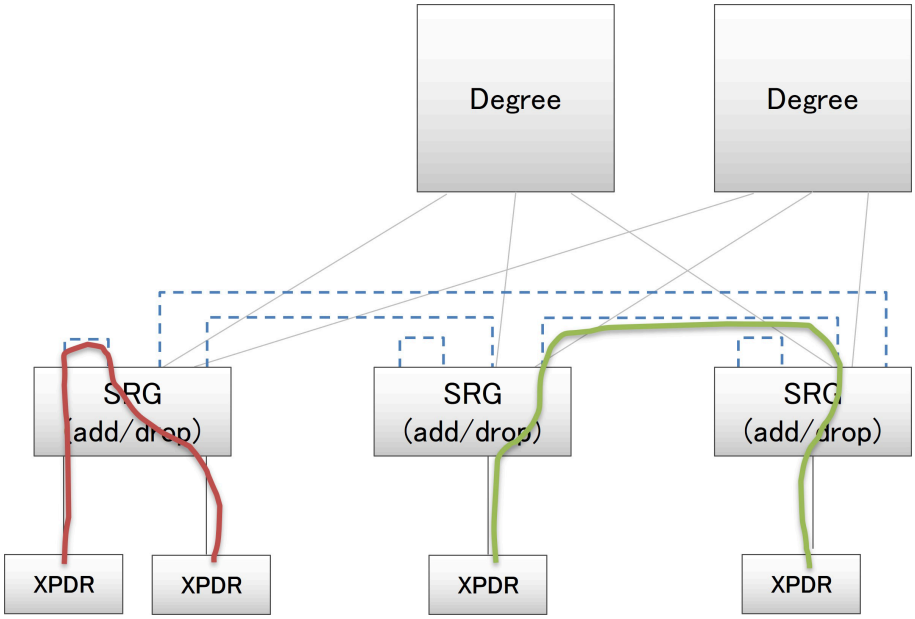

Figure 27: Turn back functionality implemented by optical fibers between SRG ports

For turn back connections, the optical control mode should be set to power and the target power would not be specified. The vendor is responsible for setting the correct power level.

4.7 ILA Model

In-Line Amplifier describes the directions of an in-line amplifier node. The ILA model abstraction includes control (provided by control mode, target gain, target tilt, and egress average channel power), capabilities (provided by amp type and amp gain range), read-only value for the attenuator, lifecycle state (planned, deployed, maintenance etc.), circuit packs list identifying all circuit packs that make up the ILA, line ports, OSC pluggable circuit pack and port, and the OTDR circuit pack and port.

```
+--rw line-amplifier* [amp-number]
|  +--rw amp-number
|  +--ro amp-type
|  +--ro optical-operational-mode-profile-name*
|  +--rw dge-deployed?
|  +--rw control-mode?
|  +--ro amp-gain-range?
|  +--rw target-gain?
|  +--rw target-tilt?
|  +--rw egress-average-channel-power?
|  +--ro out-voa-att?
|  +--ro partner-amp?
|  +--rw ila-direction-label?
|  +--rw lifecycle-state?
|  +--rw spectrum
|  |  +--rw channel* [index]
|  |     +--rw index
|  |     +--rw nmc-ctp-if?
|  |     +--rw connection-name?
|  |     +--rw circuit-id?
|  |     +--rw target-output-power?
|  |     +--ro actual-output-power?
|  |     +--ro actual-dge-attenuation?
|  +--rw circuit-pack* [index]
|  |  +--rw index
|  |  +--rw circuit-pack-name
|  +--rw line-port* [port-direction]
|  |  +--rw port-direction
|  |  +--rw tx-instance-port-direction-label?
|  |  +--rw rx-instance-port-direction-label?
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  +--rw osc-port* [port-direction]
|  |  +--rw port-direction
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  +--rw otdr-port* [otdr-direction]
|     +--rw otdr-direction
|     +--rw circuit-pack-name
|     +--rw port-name
```

YANG sub-tree 26: ILA model

4.8 Xponder Model

4.8.1 Xponder Abstractions

The Xponder model (Tree 27) defines the logical grouping of ports that make up a transponder, muxponder, switchponder, or regenerator. The Xponder model abstraction includes a logical xponder identifier, the xponder type (transponder, muxponder, switchponder, regenerator), lifecycle state (planned, deployed, maintenance etc.), recolor ability (regenerators only), port list of ports associated with the xponder and the associated SRG associated with the port.

The Open ROADM MSA supports the various xponder types differently; xponders can be placed into two modeling categories: transponders and uni-directional regenerators in one category and muxponders, switchponders, and bi-directional regenerators in the other category (see Table 3: Xponder Model for details).

xponder	xpdr-type	port-qualifier	connectivity announce-ment	explicit cross-connect?	section reference
Transponder	tpdr	xpdr-client/network	connectivity-map	no	4.8.5
Muxponder	mpdr	switch-client/network	switching pool	yes	4.8.6
OTN Switch/Switch-ponder	switch	switch-client/network	switching pool	yes	4.8.6
Regenerator	regen	switch-client/network	switching pool	yes	4.8.7.1
	regen-uni	xpdr-client/network	connectivity-map	no	4.8.7.2

Table 3: Xponder Model

```
+--rw xponder* [xpdr-number]
|  +--rw xpdr-number
|  +--rw xpdr-type
|  +--rw lifecycle-state?
|  +--ro recolor?
|  +--rw xpdr-port* [index]
|  |  +--rw index
|  |  +--rw circuit-pack-name
|  |  +--rw port-name
|  |  +--rw eqpt-srg-id?
|  +--rw xpdr-pluggable-optics-holder* [index]
|     +--rw index
|     +--rw circuit-pack-name
|     +--rw cp-slot-name
|     +--rw eqpt-srg-id?
```

YANG sub-tree 27: Xponder model

4.8.2 Slot and Port Capabilities Announcement

A series of capabilities are defined against the slot and port. Circuit-pack slots (cp-slots), pluggable optic capable slots (pluggable-optics-holder-capability), and circuit pack port capabilities (port-capabilities). Both the slot and port capabilities define the supported interface hierarchy and advertised via the if-cap-type. OTN specific capabilities announcements are supported for protection, delay measurement, TCM, muxponder mapping restrictions, and ODU muxing hierarchy.
Note: the *logical-port* attribute is expected to be populated under *port-capability*. It is optional to populate the *logical-port* attribute under the *pluggable-optics-holder-capability*.

```
+--ro cp-slots* [slot-name]
|  +--ro slot-name
|  +--ro label?
|  +--ro provisioned-circuit-pack?
|  +--ro slot-status?
|  +--ro slot-type?
|  +--ro pluggable-optics-holder-capability
|     +--ro supported-circuit-pack* [supported-circuit-pack-type
↪  supported-pluggable-id-type]
|        +--ro supported-circuit-pack-type
|        +--ro supported-pluggable-id-type
|        +--ro ports* [port-name]
|           +--ro port-name
|           +--ro port-capabilities
|                 ...
```

YANG sub-tree 28: CP-slots and ports

```
+--ro port-capabilities
   +--ro supported-interface-capability* [if-cap-type]
      +--ro if-cap-type
      +--ro split-lambda-profile-name*
      +--ro otsigroup-capability-profile-name*
      +--ro otsi-profile-name*
      +--ro optical-operational-mode-profile-name*
      +--ro otn-capability
      |  +--ro otn-capability-profile-name?
      |  +--ro mpdr-client-restriction* []
      |  |  +--ro network-ho-odu-circuit-pack-name
      |  |  +--ro network-ho-odu-port-name
      |  |  +--ro muxp-profile-name*
      |  +--ro otn-odu-mux-hierarchy-profile-name?
      +--ro logical-port
      |  +--ro circuit-pack-name?
      |  +--ro port-name?
      +--ro eth-capability
         +--ro mpdr-client-restriction* []
            +--ro network-eth-circuit-pack-name
            +--ro network-eth-port-name
            +--ro muxp-eth-profile-name*
```

YANG sub-tree 29: Port Capabilities

In the Open ROADM MSA model, a port can be in reference to a circuit pack ,where the circuit pack represents a pluggable optics module. Such a circuit pack nominally has a single port and all such circuit packs can have a common port-name, such as 'P1', given there is a unique circuit-pack-name. As such, both fixed ports and pluggable ports in the Open ROADM MSA model are defined. A fixed port is in reference to an interface which is a part of a circuit pack such as a line module (e.g. network interface of an xponder). A pluggable port is reference to an optical module/circuit pack which might be plugged into an SFP or QSFP28 cage on the line module(e.g. client interface of an xponder). Such a circuit pack should have the Boolean "is-pluggable-optics" set as "True".

In addition to ports, the model defines cp-slots. These are in reference to the SFP, QSFP28, CFP cages themselves. Note that *cp-slots* can also refer to slots on circuit-packs that are not pluggable cages (e.g.slots to hold other circuit packs). An enumeration within a *cp-slot* defines if the *cp-slot* is a pluggable cage (*slot-type=pluggable-optics-holder*).

The *port-capabilities* container is used in both the definition of the CP's ports and in the definition of the ports that is part of the CP's cp-slot. It allows the device to advertise to the controller, via profiles containing the functionality offered by the device. The MSA allows flexibility in the offered set of services by the device subject to the qualification of such device by the end user.

The fundamental set of capabilities announced is provided by the list of *supported-interface-capability*. This list identifies port configuration options which may be used to determine a set of service types offered by the interface. For example, a cp-slot representing an SFP/SFP+ cage might indicate the ability to offer service types ranging from 10GE, OTU2, OTU2e, 1GE, etc. When a pluggable optics module is installed into such a slot, that port may too have a set of capabilities that are subset of those offered by the *cp-slot*. For example, in the example above the pluggable optics may only indicate a capability to provide a 1GE service type. In such a case, the *cp-slot* capabilities remain constant while the installed pluggable optics may filter the supported list. It is the responsibility of the controller to understand the intersection of the capabilities to ensure the desired service type can be offered.

Within the port-capabilities, there can exist further information associated with OTN interfaces defined by the *if-cap-type*, *split-lambda-profile-name*, *otsigroup-capability-profile-name*, *optical-operational-mode-profile-name*, *otn-capability*, and *logical-port* (i.e. logical port on an existing *circuit-pack-name*).

The *if-cap-type* defines the interface type, hierarchy, and rate supported on the port.

For a B100G port, *split-lambda-profile* defines the split lambda mode capabilities, *otsigroup-capability-profile* defines the OTSi Group capabilities, and *otn-odu-hierarchy-profile* defines the OTN ODU hierarchy capabilities.

The *otn-capability* includes an *otn-capability-profile* that defines a set of functions associated with the OTN interface, including the OTN ODU hierarchy capabilities and the muxponder client restriction rules for how LO ODUs are multiplexed into HO ODUs and the relation with the mapped client ports.

The OTN ODU hierarchy capabilities is profile-based (*otn-odu-hierarchy-profile*) and identified by the *otn-odu-hierarchy-profile-name*. The *otn-odu-hierarchy-profile* is defined as follows:

```
+--ro otn-odu-mux-hierarchy-profile* [profile-name]
|  +--ro profile-name
|  +--ro mux-capability* [stage-number ho-odu-type ho-odu-payload-type]
|     +--ro stage-number
|     +--ro ho-odu-type
|     +--ro ho-odu-payload-type
|     +--ro supported-lo-odu-type*
|     +--ro lo-odu-proactive-DMp?
|     +--ro lo-odu-tcm-capable?
|     +--ro lo-odu-proactive-DMt?
|     +--ro lo-odu-tcm-direction-capability?
```

YANG sub-tree 30: OTN-ODU profile hierarchy

The *mpdr-client-restriction* identifies restrictions to the general cross-connect model that may be present on tributary port number and tributary slot usage for client services being cross connected into what would nominally be considered a network side interface. The *mpdr-client-restriction* would be present on the client ports being mapped into LO ODU and then cross-connected to a network-side interface. Such restrictions are common to many muxponder implementations. For example, a 10×10G muxponder may have the restriction that client port 1 must be mapped into trib port 1, trib slots 1-8, client port 2 must be into trib port 2, trib slots 9-16, etc... The *mpdr-client-restrictions* allow for a common provisioning model for muxponder applications. The muxponder capabilities is profile-based (*muxp-profile*) and identified by the *muxp-profile-name*. The is defined *muxp-profile* as given in Tree 31.

```
+--ro muxp-profile* [profile-name]
|  +--ro profile-name
|  +--ro odtu-type
|  +--ro network-odu-rate
|  +--ro network-oducn-n-rate?
|  +--ro network-ho-odu-trib-port-number
|  +--ro network-ho-odu-trib-slots*
|  +--ro network-ho-odu-opucn-trib-slots*
```

YANG sub-tree 31: Muxp-profile

An example of a circuit pack with three pluggable slots is illustrated in Figure 28 and Table 4. The first slot accepts SFP/SFP+ type pluggable optics while the second supports QSFP/QSFP28 pluggable optics. In each case, the *supported-interface-capabilities* provides a list of service types supported along with the further capabilities of that interface type.
A further example is illustrated in Table 5 and Figure 29. This example is provided to illustrate the intended extensibility of the model to support multiple interfaces from a single optical module as might happen in a pig-tail application.

4.8.2.1 Port Group Restrictions

The concept of port group restrictions is based on the ability of a device to offer a set of software programmable service types on a group of interfaces. However, within that group of interfaces there may be restrictions on what can simultaneously be provisioned. For example, consider the case of a 10G muxponder which provides 8 × SFP based ports. It may be that these ports are software configurable to support a range of services. For example, 8 GbE or 4 × OC-48/STM16. The device, however, may present restrictions on what can be configured on which interfaces at one time.
The device model provides the ability to document such restrictions through a *port-group-restriction-grp* grouping. Nominally the group models a table with a set of ports which can include a *port-list* for fixed ports and a *pluggable-optics-holder-list* for pluggable optics. This group is referenced by a *port-sharing-id*to identify ports which have dependency on configuration and/or bandwidth usage. Within such a group, there is modeled a list of *possible-port-config*. Each entry in this list is identified by a *config-id* and identifies the possible service types that can be configured as indicated by the *supported-if-capability*.
An example of port group restrictions is shown in Figure 30. In this example the group is comprised of two sets of four interfaces where bandwidth and configuration are restricted between interfaces 1-4 and 5-8. Interfaces 1-7 are pluggable-optics-holder, while port 8 is fixed port residing on a common circuit pack, CP1.
Within ports 1-4 and 5-8 there is a restriction the total BW of the configured ports should not exceed 5Gbps. Within each group, interface types can be configured for GE, OC3/STM1, OC12/STM14, OC48/STM16, or OTU1. However, all such types cannot be satisfied on any port at any time in any combination. The possible-port-config identifies the supported combinations as illustrated in Table 6 and Tree 32.
In Table 6, one can identify that config-id 1 restricts the interface type of FC-400 to only be supported on port 1, config-id 2 supports OTU1 and/or OC48/STM16 on ports 1 and 3, config-id 3 allows port 1 to be either OTU1 or OC48/STM16 while ports 3 and 4 can be any combination of GE, OC3/STM1, OC12/STM4, etc ...
The active or configured port group is advertised via *port-group-restriction/possible-port-config*, referenced by a *port-*

```
+--ro port-group-restriction
|  +--ro port-bandwidth-sharing* [port-sharing-id]
|     +--ro port-sharing-id
|     +--ro port-list* [circuit-pack-name port-name]
|     |  +--ro circuit-pack-name
|     |  +--ro port-name
|     +--ro pluggable-optics-holder-list* [circuit-pack-name slot-name]
|     |  +--ro circuit-pack-name
|     |  +--ro slot-name
|     +--ro shared-bandwidth?
|     +--ro possible-port-config* [config-id]
|        +--ro config-id
|        +--ro port-if-type-config* [circuit-pack-name port-name]
|        |  +--ro circuit-pack-name
|        |  +--ro port-name
|        |  +--ro port-if-type*
|        |  +--ro otsi-rate?
|        +--ro slot-if-type-config* [circuit-pack-name slot-name port-name]
|           +--ro circuit-pack-name
|           +--ro slot-name
|           +--ro port-name
|           +--ro port-if-type*
|           +--ro port-module-type*
|           +--ro otsi-rate?
```

YANG sub-tree 32: Port Group Restriction

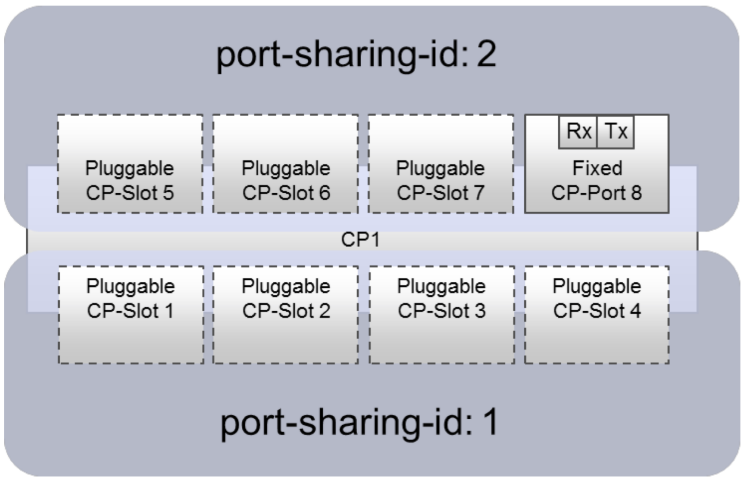

Figure 30: Port Group Restriction Interfaces

circuit-pack-name	cp-slot-name	circuit-pack-type	port-name	port-capabilities
1	1	SFP	1	supported-interface-capability* = {if-och-otu1, if-1gbe} **For if-och-otu1** {if-protection-capability, tcm-direction, ho-odu-index, ...} **For if-1gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ... }
		SFP+	1	supported-interface-capability* = {if-och-otu1, if-1gbe, if-och-otu2, if-10gbe} **For if-och-otu1** {if-protection-capability, tcm-direction, ho-odu-index, ... } **For if-1gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ... } **For if-och-otu2** {if-protection-capability, tcm-direction, ho-odu-index, ... } **For if-10gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ... }
	2	QSFP+ 40G	2	supported-interface-capability* = {if-och-otu3, if-40gbe} **For if-och-otu3** {if-protection-capability, tcm-direction, ho-odu-index, ...} **For if-40gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ...}
		QSFP28 100G	2	supported-interface-capability* = {if-och-otu3, if-och-otu4, if-100gbe} **For if-och-otu3** {if-protection-capability, tcm-direction, ho-odu-index, ... } **For if-och-otu4** {if-protection-capability, tcm-direction, ho-odu-index, ... } **For if-100gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ... }

Table 4: Port capabilities

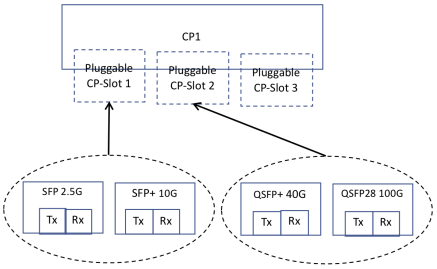

Figure 28: Port capabilities

```
+--rw provisioned-port-grp
+--rw port-bandwidth-sharing* [port-sharing-id]
+--rw port-sharing-id
+--rw provisioned-port-config?
```

YANG sub-tree 33: Provisioned port group configuration

sharing-id and *config-id*.

4.8.2.2 Provisioned Port Group Configuration

The supported port groups are specified via the *provisioned-port-grp/provisioned-port-config* container, referenced by a *port-sharing-id*.

4.8.3 MC Capabilities

Like ROADMs (see Section 4.6.3), xponders also use an MC capabilities profile (*mc-capability-profile*). Xponders use a MC capabilities profile to specify the frequency provisioning range and granularity used by its network ports. An *mc-capability-profile* for xponder use is defined as follows:

```
+--ro mc-capability-profile* [profile-name]
   | +--ro profile-name
   | +--ro center-freq-granularity? (frequency-GHz)
   | +--ro min-edge-freq? (frequency-THz)
   | +--ro max-edge-freq? (frequency-THz)

[duplicated]
```

YANG sub-tree 34: MC capability profile

Note: *slot-width-granularity*, *min-slots*, and *max-slots* are not used for xponders. MC capabilities for transponders are only

circuit-pack-name	cp-slot-name	circuit-pack-type	port-name	port-capabilities
1	3	QSFP28 100G	1	supported-interface-capability*= {if-och-otu4, if-100gbe} **For if-och-otu4** {if-protection-capability, tcm-direction, ho-odu-index, ... } **For if-100gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ...}
		QSFP28 4x25G	3-1	supported-interface-capability* = {if-och-otu2, if-10gbe, if-25gbe} **For if-och-otu2** {if-protection-capability, tcm-direction, ho-odu-index, ...} **For if-10gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ...} **For if-25gbe** {client-mapping-odu-type, otu-type, network-ho-odu-trib-port-number, ...}
			3-2	supported-interface-capability* = {if-och-otu2, if-10gbe, if-25gbe} ...
			3-3	supported-interface-capability* = {if-och-otu2, if-10gbe, if-25gbe} ...
			3-4	supported-interface-capability* = {if-och-otu2, if-10gbe, if-25gbe} ...

Table 5: Port capabilities-Multi Phy

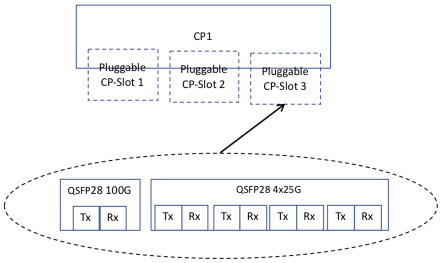

Figure 29: Port capabilities-Multi Phy

required for the transponder network port. This information is used by the controller to set the transponder frequency on the OCH or OTSI interface.

4.8.4 ODU Interface

4.8.4.1 ODU Termination Points

The Open ROADM MSA model defines the function of the ODUk interface by the odu/odu-function attribute, which can be one of three types of termination points, allowing the provisioning of a multiplex hierarchy and facilitating cross connects:

- **ODU-TTP** – An ODU TTP (trail termination point) is associated with the context of a HO ODU. It is a PM layer terminated ODU which faces the line side or client side under an OTU interface. For example, an ODU must be terminated in order to multiplex ODUj into it. An ODU-TTP always has a payload type of PT-20/21/22 indicating its tributary slot size and its ability to be muxed into it. It essentially has an MSI table which originates here.
- **ODU-CTP** – An ODU CTP (connection termination point) which can be part of a cross connect is considered a LO ODU. It is a non-terminated ODU which faces the line side or client side under an OTUCn/k or ODU-TTP. An ODU-CTP defines the endpoints of a cross connect.
- **ODU-TTP-CTP** – An ODU TTP CTP is both terminated and cross connected. An ODU-TTP-CTP is utilized in the context of mapping client interfaces into ODU which may then be cross connected.

For context, refer to the example shown in Figure 31.

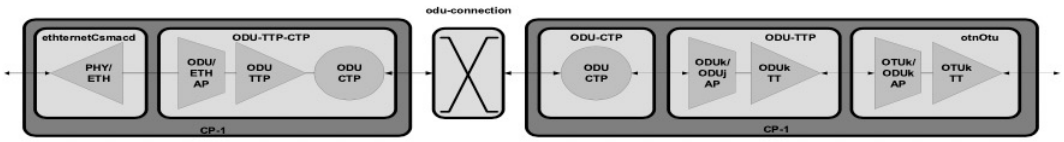

Figure 31: ODU Termination Points

The Open ROADM MSA model defines the monitoring mode of the ODU interface by the odu/monitoring-mode attribute, which states whether the ODU Overhead is terminated, not terminated, or monitored:

- **terminated**: the ODU interface has detection and generation enabled
  - For TCMs, the overhead is erased (replaced with all zeros) towards the downstream
- **non-terminated**: the ODU interface has no detection or generation and the overhead is transparently passed through the interface in both directions
- **monitored**: the ODU interface has detection enabled in the receive direction and the overhead is transparently passed through the interface in both directions

Note: “receive direction” refers to “from the faceplate” for an OTU-TTP function and “from the cross-connection” for an ODU-TTP-CTP function.

4.8.4.2 ODU Function

Depending on the xponder type, ODU interfaces can be provisioned on the network and/or client ports with the odu-function set to the applicable ODU termination points (refer to Table 7: ODU Function).
Transponder - 100GE/400GE Client:

port-sharing-id	circuit-pack-name	port-name	cp-slot-name	shared-bandwidth	config-id	circuit-pack-name	port-name	cp-slot-name	port-if-type*
1	CP1		1	5G	1	CP1	1	1	FC-400
	CP1		2		2	CP1	1	1	OTU1
	CP1		3						OC-48
	CP1		4			CP1	3	3	OTU1
									OC-48
					3	CP1	1	1	OTU1
									OC-48
						CP1	3	3	GE
									OC3
									OC12
						CP1	4	4	GE
									OC3
									OC12
					4	CP1	1	1	GE
									OC3
									OC12
						CP1	2	2	GE
									OC3
									OC12
						CP1	3	3	OTU1
									OC48
					5	CP1	1	1	GE
									OC3
									OC12
						CP1	2	2	GE
									OC3
									OC12
						CP1	3	3	GE
									OC3
									OC12
						CP1	4	4	GE
									OC3
									OC12

Table 6: Port Group Restrictions Definition

Xponder	Port Provisioning?	Cross-Connects?	odu-function
Transponder (100GE/400GE client)	network	N/A	*ODU-TTP-CTP*
Transponder (OTU4 client)	network and client	no ODU cross-connects between client and network ODU interfaces	*ODU-CTP*
Muxponder Switchponder	network and client	explicit ODU cross-connects established between network and client ODU interfaces	*ODU-TTP* for the network-side HO-ODU interface *ODU-CTP* for the network-side LO-ODU interface *ODU-CTP* for client-side LO-ODU interfaces that are OTU2 interfaces (assumes ODU2 level cross-connects *ODU-TTP-CTP* for client-side ODU interfaces that are Ethernet interfaces
Regenerator (bi-directional)	network and client	explicit ODU cross-connects established between the network and client ODU interfaces	*ODU-CTP*
Regenerator (uni-directional)	network and client	no ODU cross-connects between client and network ODU interfaces	*ODU-CTP*

Table 7: ODU function

- ODU interface provisioned on the network port
- odu-function=ODU-TTP-CTP

Transponder - OTU4:
- ODU interface provisioned on both the network and client ports
- No ODU cross-connects established between the network and client ODU interfaces
- odu-function=ODU-CTP

Switchponder/Muxponder:
- ODU interfaces provisioned on both the network and client ports
- Explicit ODU cross-connects established between the network and client ODU interfaces
- odu-function=
  1. ODU-TTP for the network-side HO-ODU interface
  2. ODU-CTP for the network-side LO-ODU interface
  3. ODU-CTP for client-side LO-ODU interfaces that are OTU2 interfaces (assumes cross-connect at the ODU2 level)
  4. ODU-TTP-CTP for client-side ODU interfaces that are Ethernet interfaces

Regenerator (bi-directional):
- ODU interfaces provisioned on both the network and client ports
- Explicit ODU cross-connects established between the network and client ODU interfaces
- odu-function=ODU-CTP

Regenerator (uni-directional):
- ODU interfaces provisioned on both the network and client ports
- No ODU cross-connects established between the network and client ODU interfaces
- odu-function=ODU-CTP

4.8.5 Transponder Model

The transponder was the first xponder device modeled in Open ROADM with MSA v1. A transponder device is defined as an xponder device with the same rate on the client side and the network side. A transponder device does not multiplex multiple client signals into the same network interface (e.g., multiplexing of multiple clients would be classified as a switchponder or muxponder instead). For example, a 100GE client mapped into an OTU4/OCH network interface would be classified as a transponder device.
An example of a transponder is shown in Figure 32. The transponder would be modeled with a port qualifier of xpdr-client or xpdr-network. The Connection Map entity is used to show the mapping between the client and network ports for the transponder. Also, there are no explicit cross connects provisioned for transponders.

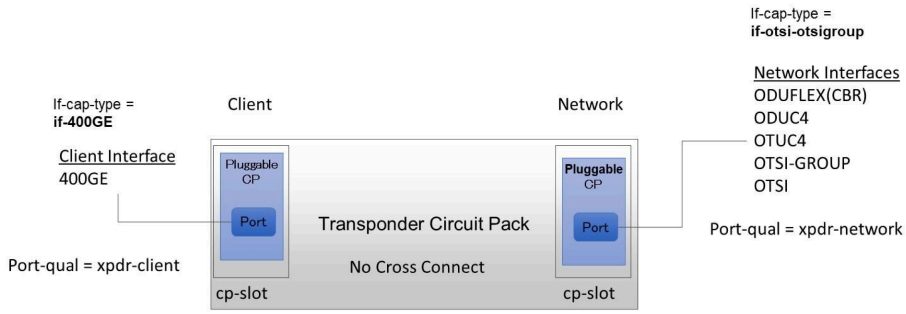

Figure 32: Example of a 400GE transponder

4.8.5.1 Connection Map

Like ROADMs and uni-directional regenerators, connection maps (see Section 4.5.1) are used to indicate the port-to-port connectivity between external ports of a transponder device. Connection Maps represent uni-directional connectivity between the ports. For transponders, the connectivity is fixed and no cross-connects are required. There should be two entries in the connection map for a transponder entity: a unidirectional entry from the client port to the network port, and a unidirectional entry from the network port to the client port.

4.8.5.2 Client/Network Mapping

As described in Section 4.15, B100G clients are mapped into a OPUCn 5G Tributary Slot (TS) #A.B per ITU-T G.709 clause 20.1 [7], where A=1. . .n and B=1. . .20 (5G TS). The mapping table below (refer to Table 8 shows the corresponding Tributary Port Number (TPN) and Tributary Slot (TS) for a 400GE mapped to a 400G network interface. Refer to the OpenROADM optical specifications [3]

Client Port	Network ODU Mapping
1	5G TS Logical Channel #1 TPN 1 TS 1.1 - 1.20 TS 2.1 - 2.20 TS 3.1 - 3.20 TS 4.1 - 4.20

Table 8: Transponder 400GE to ODUflex(400GE) to OTUC4 Mapping

4.8.6 Muxponder/Switchponder Model

4.8.6.1 ODU Cross-Connections

ODU cross-connections (odu-connection) are applicable to muxponders, switchponders, and bi-directional regenerators (see Section 4.8.7.1). ODU cross-connections connect ODU interfaces on the client and/or network ports (client-to-client, client-to-network, network-to-network). To support ODU cross connects, ODU interfaces will be created on the network side and the client side with an odu-connection(s) connecting the two interfaces.
An ODU cross-connect is made by an odu-connection which has a src-if and dest-if. The source interface and destination interface are required to be an otnODU interface with a function of either ODU-CTP or ODU-TTP-CTP. Cross-connects can either be bi-directional or uni-directional as defined by the direction attribute where bi-directional cross-connects are the default.

```
+--rw odu-connection* [connection-name]
|  +--rw connection-name
|  +--rw direction?
|  +--rw source
|  |  +--rw src-if
|  +--rw destination
|  |  +--rw dst-if
|  +--rw circuit-id?
```

YANG sub-tree 35: ODU connection

It is mandatory for implementations to support bi-directional cross-connects, while support for uni-directional cross-connects is optional. An attribute in the switching pool container (see Section 4.8.6.4) is used to announce the types of cross-connects a vendor's device supports. Vendors can set the *odu-connection-direction-capabilities* to either "*connection-direction-bi*" if they only support bi-directional cross-connects or "*connection-direction-bi-and-uni*" if they support both bi-directional and uni-directional cross-connects.

4.8.6.2 ETH Cross-Connections

Ethernet cross-connects will be used to model the Flexo-xe muxponders/switchponders connectivity. More details about the characteristic information to be added in the next version of the white-paper.

```
+--rw eth-connections* [connection-name]
|  +--rw connection-name
|  +--rw direction?
|  +--rw source
|  |  +--rw src-if
|  +--rw destination
|  |  +--rw dst-if
|  +--rw circuit-id?
```

YANG sub-tree 36: ETH connection

4.8.6.3 Client/Network Mapping

The OTN muxponder is modeled as an OTN switch, but with the additional constraint that the mapping between the client-side and network-side is fixed. There is no flexibility in the hardware to support a variable mapping, so a routing function in

the controller needs to understand the fixed mapping constraint with regards to tributary slots and tributary port number. In addition, only certain client mappings into ODU payloads are permitted. Advertising muxponder connectivity is achieved via the switching pool (see Section 4.5.2) supporting the provisioning of explicit cross-connects, allowing for the optional advertisement of "fake" ODU interfaces (if the hardware only supports one ODU interface), and the capabilities advertisement specific to muxponders.

Up to 100Gb/s Client/Network Mappings:

The Open ROADM MSA muxponder supports 1GE and 10GE/OTU2 clients and network mappings to enable interoperability:

- 8 × 1GE clients mapping into a 10G network (refer to Table 9)
- 10 × 10GE/OTU2 clients mapping into a 100G network (refer to Table 10). Mapping of the 10GE into an ODU2 and ODU2e are both supported.

Client Port	Network ODU Mapping
1	TPN 1, TS 1
2	TPN 2, TS 2
3	TPN 3, TS 3
4	TPN 4, TS 4
5	TPN 5, TS 5
6	TPN 6, TS 6
7	TPN 7, TS 7
8	TPN 8, TS 8

Table 9: Muxponder 8 × 1GE to 10G Mapping (OTU2)

Client Port	Network ODU Mapping
1	TPN 1, TS 1-8
2	TPN 1, TS 9-16
3	TPN 3, TS 17-24
4	TPN 4, TS 25-32
5	TPN 5, TS 33-40
6	TPN 6, TS 41-48
7	TPN 7, TS 49-56
8	TPN 8, TS 57-64
9	TPN 9, TS 65-72
10	TPN 10, TS 73-80

Table 10: Muxponder 10 × 10GE/OTU2 to 100G Mapping (OTU4)

Beyond 100Gb/s (B100G) Client/Network Mappings:

As described in Section 4.15, B100G (200G-400G) are mapped into a OPUCn 5G Tributary Slot (TS) #A.B per ITU-T G.709 clause 20.1 [7], where A=1...n and B=1...20 (5G TS). The mapping tables below show the corresponding Tributary Port Number (TPN) and Tributary Slot (TS). Refer to the OpenROADM optical specifications [3]

- 2 × 100GE/OTU4 mapped into a 200G network (refer to Table 11)
- 3 × 100GE/OTU4 mapped into a 300G network (refer to Table 12)
- 4 × 100GE/OTU4 mapped into a 400G network (refer to Table 13)

Client Port	Network ODU Mapping
1	5G TS Logical Channel #1 TPN 1, TS 1.1 - 1.20
2	5G TS Logical Channel #2 TPN 2, TS 2.1 - 2.20

Table 11: Muxponder 2×100GE/OTU4 to 200G Mapping (OTUC2)

4.8.6.4 Switching Pool Advertisement

Similar to OTN switches, the muxponder will advertise potential connectivity between the client-side and network-side based on switching pools (see below). The Open ROADM MSA supports muxponders that can map between the client and network ports, but not between the client ports themselves.

Client Port	Network ODU Mapping
1	5G TS Logical Channel #1 TPN 1, TS 1.1 - 1.20
2	5G TS Logical Channel #2 TPN 2, TS 2.1 - 2.20
3	5G TS Logical Channel #3 TPN 3, TS 3.1 - 3.20

Table 12: Muxponder 3×100GE/OTU4 to 300G Mapping (OTUC3)

Client Port	Network ODU Mapping
1	5G TS Logical Channel #1 TPN 1, TS 1.1 - 1.20
2	5G TS Logical Channel #2 TPN 2, TS 2.1 - 2.20
3	5G TS Logical Channel #3 TPN 3, TS 3.1 - 3.20
4	5G TS Logical Channel #4 TPN 4, TS 4.1 - 4.20

Table 13: Muxponder 4×100GE/OTU4 to 400G Mapping (OTUC3)

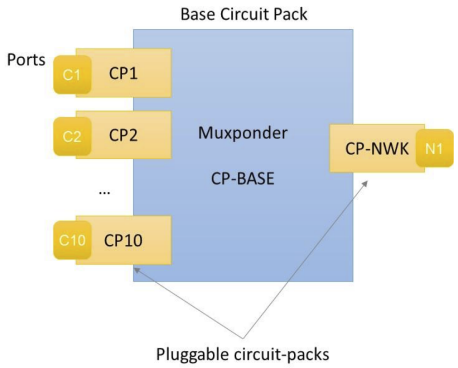

Figure 33: 10×10GE to 100G Muxponder

```
odu-switching-pools
switching-pool-number 1
switching-pool-type blocking
non-blocking-list
{nbl-number 1
interconnect-bandwidth-unit 0
interconnect-bandwidth 0
port-list [{CP1, C1}, {CP-NWK, N1}]
},
{nbl-number 2
interconnect-bandwidth-unit 0
interconnect-bandwidth 0
port-list [{CP2, C2}, {CP-NWK, N1}]
},
...,
{nbl-number 10
interconnect-bandwidth-unit 0
interconnect-bandwidth 0
port-list [{CP10, C10}, {CP-NWK, N1}]
}
```

YANG sub-tree 37: ODU switching pool example

Figure 33: 10x10GE to 100G Muxponder shows an example muxponder that supports ten 10GE pluggable client ports and one 100G OTU4 network port. The switching pool advertisement for this muxponder would be as follows in Example 37:

The non-blocking list elements, specifically the port list, indicate that each client port and network port are not blocked. But the interconnect-bandwidth of “0” indicates that the client ports cannot be connected to each other.

4.8.6.5 Explicit ODU Cross Connects

ODU services that traverse over an OTN muxponder are established with an explicit ODU cross connect (odu-connection), even if the hardware has no ODU grooming flexibility.
For muxponders, implementations must support one bi-directional cross-connect from each client port to the network port (refer to Figure 34), and may additionally support the ability to provision this using two uni-directional cross connects.

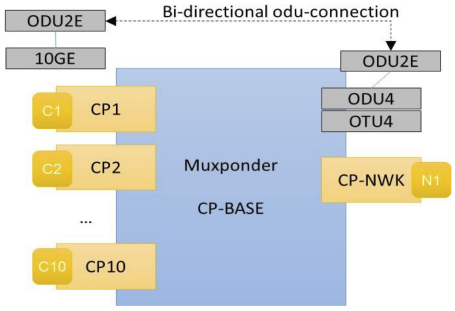

Figure 34: ODU Connection

4.8.6.6 Advertisement of ODU Capabilities Based on Hardware Limitations

Some implementations of muxponders may only support one ODU interface at the physical hardware level. But to support the muxponder’s odu-connections, the model will have two ODU interfaces (client and network side). Thus, one of the two ODU facilities will be “virtual” such that there is no functionality in the hardware.
The *no-oam-function* attribute was added to the ODU interface to indicate this case to the controller. Presence of the *no-oam-function* attribute indicates that the ODU facility cannot support OAM functions such as alarms, PMs, TCAs, TCMs, delay measurements, maintenance test signals, etc. The *no-oam-function* attribute can present on the network-side or client-side interface, but not both.
An implementation may have a restriction regarding generation of maintenance test signals. Presence of the *no-maint-testsignal-function* attribute indicates that the ODU facility cannot support the maintenance test signal function. The *no-maint-testsignal-function* attribute can be present on the network-side or client-side interface, but not both. Note that the *no-oam-function attribute* implicitly includes the *no-maint-testsignal-function* indication.

4.8.6.7 Muxponder Specific Capabilities Announcements

The port-capabilities provides the mapping restrictions between the client and network side. This capability is advertised against the client port and provides information about how it maps to the network side:

- *network-ho-odu-circuit-pack-name* and *network-ho-odu-port-name*: identifies the port that hosts the HO-ODU interface (note: the capability advertisement may exist before the HO-ODU is created)
- *odtu-type*, *network-ho-odu-trib-port-number*, and *network-ho-odu-trib-slots*: identifies the type of HO-ODU, as well as, the mapping of the LO-ODU.

4.8.7 Regenerator Model

There is no significant change to the YANG xponder modeling for regenerator equipment, including the circuit-pack or port models.

- Vendors may define a specific *circuit-pack-type*, *circuit-pack-product-code* and/or *circuit-pack-mode* to support regenerators.
- It is recommended to use the transponder-port container for the *port-power-capability* advertisement. Within the xponder container, each regenerator is treated as a separate xponder entity. For example, a bi-directional regenerator is assigned the *xpdr-type* of *regen*. The xponder container enumerates the network ports associated with the regenerators (the port on the network pluggable) and indicates if the regenerator supports wavelength recoloring.

Note: B100G regenerators are slightly different with respect to MS and RS sections, but B100G regenerators have not been discussed in the general Open ROADM MSA forum. In addition, ODUCn and ODUk transparency have not been discussed either.

```
xpdr-number=1
xpdr-type=regen
recolor= "true" or "false" to indicate whether recolor is supported
xpdr-ports=
{index=1, circuit-pack-name=A, port-name=WEST}
{index=2, circuit-pack-name=B, port-name=EAST}
```

YANG sub-tree 38: Bi-directional regenerator xponder entry

4.8.7.1 Bi-Directional Regenerator Model

The baseline bidirectional model example shown in Figure 35 assumes dual CFP2 pluggable modules supporting the regenerator function for the East and West side. The pluggable modules are considered sub-units (pluggable circuit-packs) on this circuit pack type and plugged into it (usually from the front). All ports and interfaces are modeled as bi-directional with the orange arrows representing one direction and the purple arrows representing the opposite direction. The Switching Pool (see Section 4.5.2) is used to represent the connectivity between the network ports, as well as, the ODU cross-connect (switch-network) functionality. Figure 35: Bi-directional Regenerator Model Example illustrates one example of a bi-directional regenerator implementation; vendors can offer different technical solutions, including multiple regenerators on the same circuit board.

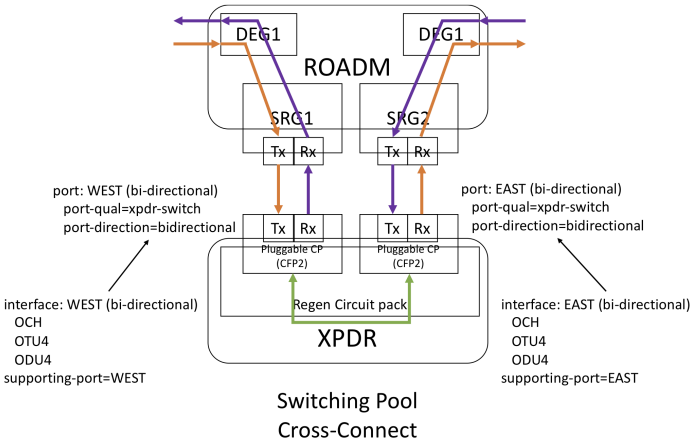

Figure 35: Bi-directional regenerator model example

There is no change required to the OCh/OTU/ODU interface and switching pool models in the Open ROADM MSA YANG model.

- The bi-directional interfaces are already supported today
- Two bi-directional OCH interfaces (EAST and WEST) wavelengths need to be specified independently

Bi-directional regenerators are modeled in a similar fashion as switchponders and muxponders (see Section 4.8.6). The network ports use the *port-qual* of *switch-network* and a switching pool (see Section 4.5.2) is used to represent the connectivity between the network ports; explicit ODU cross-connects (see Section 4.8.6.5) are used to connect the network ports.
An example for a bi-directional regenerator xponder entry would be:

4.8.7.2 Uni-Directional Regenerator Model

With the uni-directional regenerator model, each logical regenerator operates in one direction. If service is required in both directions, two logical regenerators must be used. With this model, two sets of unidirectional ports and interfaces are needed to support bidirectional regeneration. The Connectivity matrix advertisement is used to handle uni-directional ports and connectivity; ODU cross-connects are not used for uni-directional regenerators. Typically, the uni-directional regenerator type does not support recoloring, OTU BDI, or GCC0.
Figure 36 illustrates one example of a uni-directional regenerator implementation; vendors can offer different technical solutions, including multiple regenerators on the same circuit board. In this example, all ports and interfaces are modeled as uni-directional with the orange arrows representing one direction and the purple arrows representing the opposite direction. Each dotted box reflects one uni-directional regenerator circuit pack.
The Open ROADM MSA models uni-directional regenerators in a similar fashion as a transponder (see Section 4.8.5). Like a transponder, a a connection map is used to show the connectivity from the network receive port to the network transmit port and no cross-connects are required (connectivity is fixed).

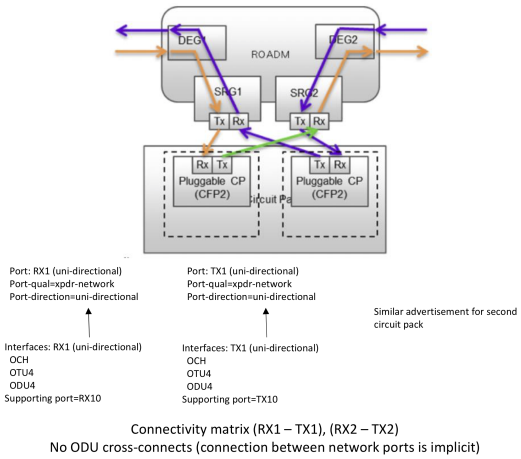

Figure 36: Uni-directional Regenerator Model Example

```
xpdr-number=1
xpdr-type=regen-uni
recolor= "false"
xpdr-ports=
{index=1, circuit-pack-name=A, port-name=RX1},
{index=2, circuit-pack-name=B, port-name=TX1}
xpdr-number=2
xpdr-type=regen-uni
recolor= "false"
xpdr-ports=
{index=1, circuit-pack-name=A, port-name=RX2},
{index=2, circuit-pack-name=B, port-name=TX2}
```

YANG sub-tree 39: Example for a uni-directional regenerator

Note: if a bi-directional regenerator is implemented with two uni-directional regenerators, two independent xponders are identified in the xponder container for each uni-directional flow. An example for a uni-directional regenerator xponder entry would be given in Example 39:

4.8.7.3 Bi-directional Regeneration Using Back-to-Back Transponders Model

Some service providers may elect to use back-to-back transponders as bi-directional regenerators, using fiber jumpers to connect the client sides. This approach could help reduce sparing and certification costs. Back-to-back transponders used as regenerators are treated in the same way as baseline bi-directional regenerators (refer to Section 4.8.7.1). The Vendor configuration template will dictate the client QSFP28 pluggable optics and loopback fibers (physical links, fiber jumpers).
If tunable transponders are used, recoloring is not needed as an explicit feature, since tunable transponders have the ability to convert (or recolor) any wavelength coming in from the client-side to a desired wavelength at the line-side. Figure 37 illustrates an example of a possible vendor implementation. In this case one transponder is used in each direction for bi-directional service. The orange arrows show one direction of traffic and the purple arrows show the other direction. The client sides are connected via fiber jumpers (thin orange and purple arrows in the bottom). Each dotted box reflects one transponder circuit pack.
For this implementation, the network ports shall be advertised as switch-network and odu-connections supported between the two network ports. Implementations should not expect cross connects between the network and client ports as the client ports are considered internal to the regenerator if implemented as back-to-back transponders.
Attributes specific to a bi-directional regenerator implemented as back-to-back transponders is modeled similar to bi-directional regenerators. Note: Only network ports are specified, as client ports are considered to be internal connections and therefore not enumerated in the xponder container.

4.8.8 Bookended Xponders Model

The bookended xponder concept is to allow for the bookending of vendor xponders on both sides of the end-to-end service in order to improve performances by using optical settings not standardized by the Open ROADM MSA. This includes

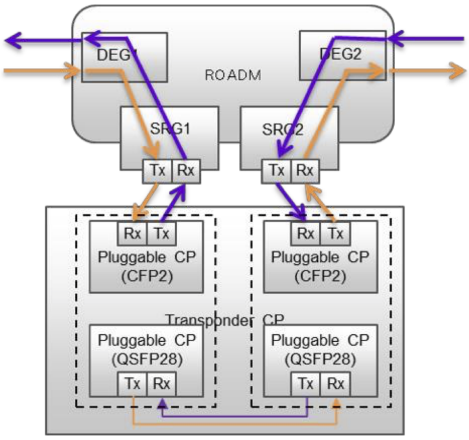

Figure 37: Bi-directional Regeneration Using Back-to-Back Transponders Example

```
xpdr-number=1
xpdr-type=regen
recolor= "true" or "false" to indicate whether recolor is supported
xpdr-ports=
{index=1, circuit-pack-name=A, port-name=WEST}
{index=2, circuit-pack-name=D, port-name=EAST}
```

YANG sub-tree 40: Example for Bi-directional regeneration with back-to-back transponders

vendor proprietary FEC and modulation format settings.
The Open ROADM MSA device model allows for vendor proprietary line-side optical specifications (e.g. FEC, modulation) and pre-standard interface hierarchies with deployment configurations such as an Open Line System (OLS) configuration with bookended xponders (i.e. same vendor proprietary xponders at both ends). The interface hierarchy configurations supported by the Open ROADM MSA device model are shown in Figure 38.

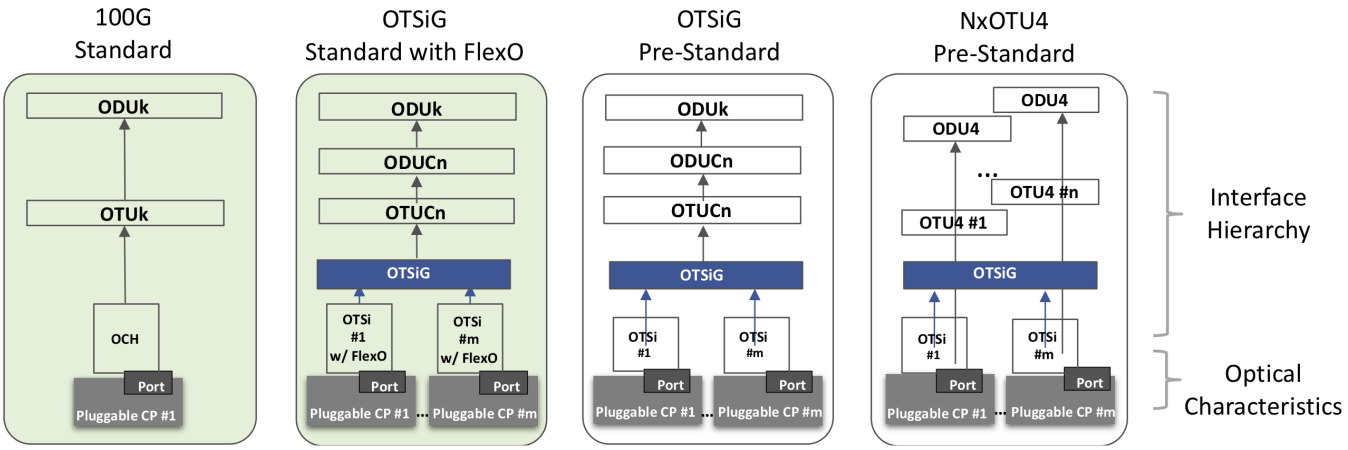

Figure 38: Bookended Xponder Configurations

Bookended xponders are required when the xponder has a vendor proprietary line-side optical specification and/or a pre-standard interface hierarchy. Referring to Figure 38, the two configurations on the left are the Open ROADM MSA standard configurations for 100G and B100G interfaces, while the two configurations on the right are considered pre-standard B100G configurations. The Open ROADM MSA device model has defined a provisioning model and interface hierarchy that can accommodate all of these configurations with bookended xponders.
The vendor proprietary line-side optical specifications (e.g. rate, modulation format, FEC) for OCh and OTSi interfaces are specified by a vendor provided optical profile (*provision-mode=profile* indicates the optical specification is profile-based) and provisioned using one of the operational mode profiles (*optical-operational-mode*identifies the profile name) advertised in the capabilities.
Similar to the interface hierarchy model used by the Open ROADM standard OTSiG (with FlexO) configuration for B100G, the interface hierarchy model can accommodate pre-standard OTSiG (without FlexO) configurations with the addition of *otsi/otsi-member-id* attributes and pre-standard N×OTU4 configurations with the addition of *otu4/otu4-member-id* attributes

to uniquely identify each member of a group and the order of each member in that group. The *otsi-group-capabilities-profile* is used to advertise the group capabilities.
Note: the electrical hierarchy has been decoupled from the optical profile with the supported optical profile modes published via the port-capabilities announcement (refer to Section 4.8.2).

4.9 External Pluggable Model

External pluggable was introduced into the OR YANG Model since release 2. However the model was not validated, and some attributes were missing to make it usable. In release 10.0, we introduce an external-pluggable list when the device node-type is “extplug” which replicates the structure used for Xponder.

```
+--rw ext-pluggable* [extplug-number]
|  +--rw extplug-number
|  +--rw lifecycle-state?
|  +--rw extplug-port* [index]
|     +--rw index
|     +--rw circuit-pack-name
|     +--rw port-name
```

YANG sub-tree 41: Pluggable capabilities

The proposed model applies to pluggable hosted in routers or in any other devices that may not support Open ROADM API models for all the hosted circuit-packs. These devices however would need to support the Open ROADM common and the device models to describe and configure the network side of the pluggable, whether this a native support or through a mediation function.

The pluggable is a simple circuit-pack and has no slot, no hosted circuit-pack/daughter board. Its optical specifications may comply with Open ROADM or not. In the first case, the pluggable performances are handled through standard openroadm-operational-modes described in the catalog. In the later case, it shall be handled through one or several specific-operational-mode(s) as for the bookended transponder solution. The network ports can be configured in the same way as regular Xponder network ports.

The pluggable and the device (hosting equipment) shall be managed through the same single Network Controller, whereby this controller can also handle the device through other APIs. It is incumbent to the controller to ascertain that there is no conflicting configuration of the device and the pluggable in order to ensure its consistent provisioning.

Considering the use case of Routers hosting external pluggable, depending on the architecture selected for the control, the end-to-end service provisioning will be performed through :

- a higher order controller (H-SDNC) if an IP-SDNC is used to control the device as well as the external pluggable and a RNC is used to control the OpenROADM infrastructure,
- a multilayer controller that handles both the optical domain (RNC Function) and the IP domain (IP-SDNC function)

The solution where a hosting router, could receive some request from two different autonomous controllers, one being dedicated to the IP layer control, and the other to the ROADM infrastructure has been dismissed since it is in contradiction with the SDN concept where a controller shall be considered as a single source of truth: having two different domain controllers controlling the same device is hardly conceivable.

4.10 Transmission Protection Groups and Protection Switching Model

The Open ROADM MSA models transmission protection switching in several YANG containers:

- Protection group container *client-sncp-pg* (see YANG sub-tree 42) to support Client SNC/I protection/Y-Cable protection according to ITU-T G.873.1 (03/2020) App. II.3 [8]. See section 4.10.8 for details.
- Protection group container *odu-sncp-pg* (see YANG sub-tree 43) to support SNC/Ne, SNC/Ns and SNC/S protection according to ITU-T G.873.1 (03/2020) Table 8.1 [8]
- Protection profile container *protection-profile* (see YANG sub-tree 44) to announce transmission protection capabilities to the controller. See section 4.10.6 for details.

```
+--rw protection-grps
 ...
|  +--rw client-sncp-pg* [name]
|     +--rw name
|     +--rw switching-direction?
|     +--rw revertive?
|     +--rw sd-enable?
|     +--rw wait-to-restore?
|     +--rw holdoff-timer
|     |  +--rw holdoff?
|     |  +--rw holdoff-multiplier?
|     +--rw working-if
|     +--rw pg-interfaces*
|     +--ro active-if?
|     +--ro request-state?
```

YANG sub-tree 42: Client Subnetwork connection protection (org-openroadm-prot-otn-linear-aps)

```
+--rw protection-grps
 ...
|  +--rw odu-sncp-pg* [name]
|     +--rw name
|     +--rw level
|     +--rw prot-type?
|     +--rw mode
|     +--rw tcm-layer
|     +--rw switching-direction?
|     +--rw revertive?
|     +--rw sd-enable?
|     +--rw wait-to-restore?
|     +--rw holdoff-timer
|     |  +--rw holdoff?
|     |  +--rw holdoff-multiplier?
|     +--rw working-if
|     +--rw pg-interfaces*
|     +--ro active-if?
|     +--ro request-state?
  ...
```

YANG sub-tree 43: OTN connection protection (org-openroadm-prot-otn-linear-aps)

```
+--ro protection-profiles
|  +--ro protection-profile* [profile-name]
|     +--ro profile-name
|     +--ro level
|     +--ro prot-type
|     +--ro switching-direction
|     +--ro mode-list*
|     +--ro xc-capabilities-list*
|     +--ro pg-capabilities-list*
```

YANG sub-tree 44: Transmission protection profiles group

ITU-T G.873.1 [8] defines protection groups of the linear protection mechanisms for the optical transport network at the Optical Data Unit k (ODUk) level. A protection group consists of a head-end, tail-end, the working transport entity, and the protect transport entity (refer to Figure 39). Whenever the quality of the optical signal falls below a pre-configured threshold, the traffic is switched from the working channels to the protection channels.

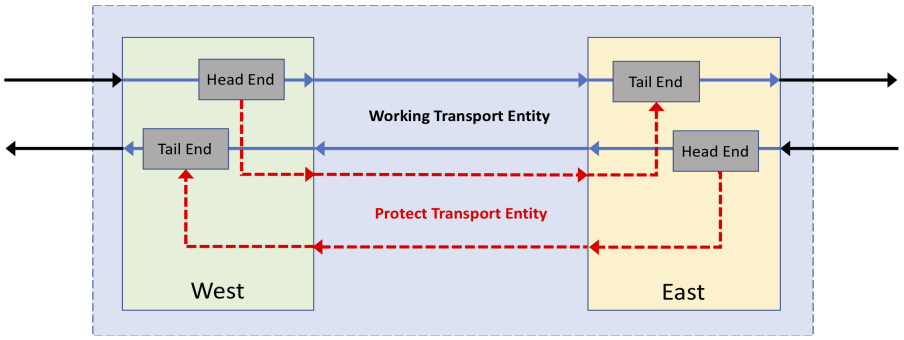

Figure 39: Protection Group

Open ROADM MSA protection groups support 1+1 line and path protection. Both uni-directional (*switching-direction* = *unidirectional-switching*) and bi-directional (*switching-direction* = *bidirectional-switching*) protection switching is supported. For both line and path protection the default is uni-directional switching. ITU-T G.873.1 [8] defines the automatic protection switching (APS) protocol and protection switching operation of the linear protection mechanisms for the optical transport network at the Optical Data Unit k (ODUk) level. These mechanisms are based on the generic protection specification, ITU-T G.808.1 [9].

The protection types supported by the Open ROADM MSA are as follows:

- 1+1 uni-directional w/o APS
- 1+1 bi-directional w/ APS

4.10.1 Uni-Directional Protection Switching

With uni-directional switching (*odu-sncp-pg/switching-direction=unidirectional-switching*), the selectors operate independently at each end of the protection group (*odu-sncp-pg*). The priority is determined by the near-end only. Uni-directional switching can protect two uni-directional failures in opposite directions on different entities. When a failure in one direction is detected, only traffic in this direction is switched and traffic in the other direction is still received from the original channel. Switching in each direction is independent and has no impact on switching in the other direction. The default provisioning for the *switching-direction* is uni-directional (*unidirectional-switching*).

4.10.2 Bi-Directional Protection Switching

With bi-directional switching (*odu-sncp-pg/switching-direction=bidirectional-switching*), an attempt is made to coordinate the selector at each end so they have the same bridge and selector settings. When a failure in one direction is detected, the traffic in both directions is switched regardless of whether the traffic in the other direction is faulty. Bi-directional switching requires an Automatic Protection Switching (APS) protocol and/or a protection communication channel (PCC) to coordinate the two endpoints. APS is enabled when the *switching-direction* is bi-directional. The APS/PCC is transmitted over the protect transport entity. Although, it may also be transmitted identically on working transport entities, receivers should not assume so and should have the capability to ignore the information on the working transport entities.

4.10.3 Protection Switch Commands

The end-to-end switch commands (*odu-sncp-protection-switch/switch-command*) supported by the Open ROADM MSA are shown in table 14.

SNCP has two operating modes: revertive (switches back to the working path after the switch reason has cleared or a Force-Switch command is initiated away from the protect path) and non-revertive (only switches back if the signal of the working path is better-quality than the protect path or a *Force-Switch* command or *Manual-Switch* command is initiated). The default provisioning for 1+1 protection is non-revertive (*odu-sncp-pg/revertive=false*), as the protection mechanism is fully dedicated (avoids a second glitch to the traffic). The choice of revertive/non-revertive should be the same at both endpoints of the protection group.

To prevent frequent operation of the selector (chattering) as a result of intermittent failures, the revertive operating mode (*odu-sncp-pg/revertive=true*) uses a wait-to-restore (WTR) timer (*odu-sncp-pg/wait-to-restore*). The protection group remains in a WTR state until the timer expires or any other event or *switch-command*, at which point the traffic is selected from the working transport entity. Note: a *Force-Switch* request from working to protect will not have traffic selected from the working transport entity; but it would clear the WTR state.

Each protection group has a provisionable hold-off-timer (*odu-sncp-pg/holdoff-timer*) as defined by ITU-T G.873.1 [8], to allow either a layered protection switch to have a chance to correct the problem before switching at the client layer, or to allow an upstream ring to switch before the downstream ring in a dual ring interconnect configuration (ensures the switch occurs in the same ring as the failure). After the hold-off timer expires, a check is made to determine whether a defect still exists on the trail that started the timer (it doesn't have to be the same defect that started the timer). If a defect is still present, the defect will be reported and traffic will be switched.

Command (switch-command)	Description
Lock-Out-Protect (LoP)	Prevents switching from the working path to the protect path, effectively disabling the protection group. If the Normal Traffic Signal is currently on the protect path, an automatic switch to the working path will occur.
Force-Switch (FS)	Forces Normal Traffic Signal to be switched from one path to the other path. Note: a forced switch is prevented if a Lock-Out Protect or another Force Switch request is active.
Manual-Switch (MS)	In the absence of a failure of a working or protect path, the Normal Traffic Signal is switched from the protect path to the working path or from the working path to the protect path. If the path being switched to is failed, if there is a *Lock-Out-Protect* request active, or if there is a *Force-Switch* request active, the *Manual-Switch* is prevented.
Release (CLEAR)	Releases the active near-end *Lock-Out-Protect, Force-Switch, Manual-Switch* commands or wait-to-restore (WTR) state
Exercise	Repeat the currently active near-end command to test the APS protocol without triggering any switching

Table 14: End-to-End Protection Switch Commands

4.10.4 Protection Switch Alarms

The following table 15 shows all alarms that can be raised against a protection group. Together with the underlying protection switch states, they are defined by ITU-T G.873.1 [8].

Probable Cause	Protection Switch State	Description
forcedSwitchAwayFromProtect	W, FS	Forced Switch to Working
forcedSwitchAwayFromWorking	P, FS	Forced Switch to Protection
lockoutOfProtection	W, LOP	Lockout of Protection
manualSwitchAwayFromProtect	W, MS	Manual Switch to Working
manualSwitchAwayFromWorking	P, MS	Manual Switch to Protection
automaticSwitchAwayFromProtectDueToSF	W, SF or W, SD	Signal Fail or Degrade, W selected
automaticSwitchAwayFromWorkingDueToSF	P, SF or P, SD	Signal Fail or Degrade, P selected
automaticSwitchDueToWTR	P, WTR	Wait-to-Restore
<none>	W, NR	Traffic is on Working
<none>	P, DNR	Traffic is on Protection
apsProtocolError	n/a	Failure of Protocol - No or Unexpected Response
apsConfigurationMismatch	n/a	Failure of Protocol - Configuration Mismatch

Table 15: Alarms of a Protection Group

4.10.5 Linear OTN Protection Modes

The Open ROADM MSA supports the linear OTN protection modes (*odu-sncp-pg/mode*) listed in Table 16 per ITU-T G.873.1 [8].

Protection Modes (mode)	Description	Server Layer Protected Entity	Protection Switched Entity	Switch Trigger Criteria	Protection Type Supported		
					Line	Path	Client
SNC/Ne	SNC with Non-intrusive end-to-end OH monitoring	One or more HO ODUk and/or OTUk	ODUkP	ODU TSF/TSD (*ODUP-SSF-SSD*)	Y	Y	N
SNC/Ns	SNC with Non-intrusive sub-layer OH monitoring	One or more HO ODUk and/or OTUk	ODUkT	ODU TSF/TSD (*ODUT1, ODUT2, ODUT3, ODUT5, ODUT6*)	Y	Y	N
SNC/S	SNC with sub-layer monitoring	One or more HO ODUkP	ODUkT or ODUkP	ODUkT SSF/SSD (*ODUT1, ODUT2, ODUT3, ODUT5, ODUT6*)	Y	Y	N
SNC/I	SNC with Inherent monitoring	One HO ODUk or One OTUk	ODUkP or ODUkT	ODU SSF/SSD (*OTUk/ODUkA-SSF-SSD*)	Y	N	Y

Table 16: Linear OTN Protection Modes

The relationship between Line and Path level protection and the associated linear OTN protection modes and protection switch entities is shown in Figure 40:

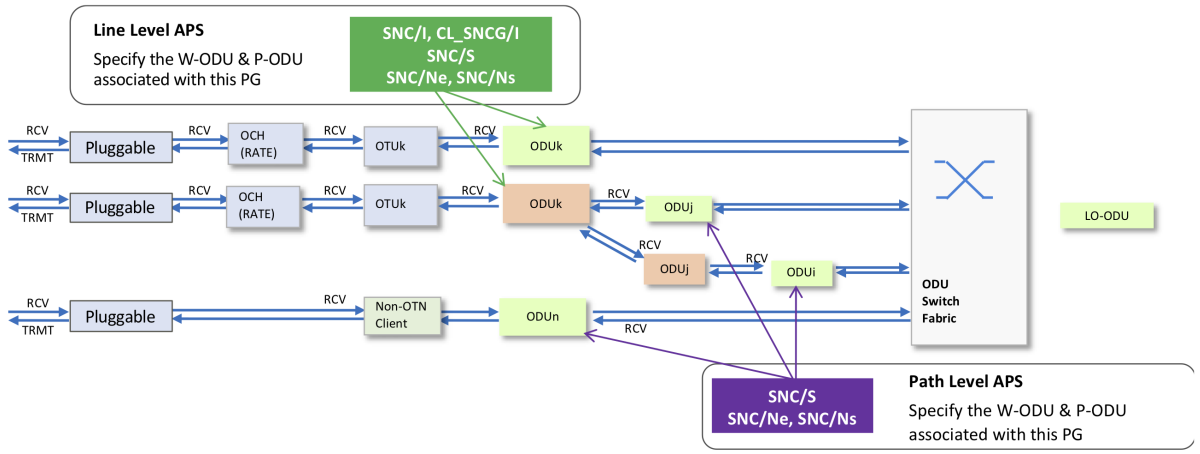

Figure 40: Linear OTN Protection Modes and Associated Protection Switched Entities

It should be noted that only the SNC/Ns and SNC/S linear protection modes are applicable to Tandem Connection Monitoring (TCM) assignment (refer to Section 4.13 for details on TCM). The OTN Protection mode SNC/Ns cannot support layer adjacency discovery of working and protect links (non-intrusive monitor has no ability to insert discovery information). Protection switching is triggered by signal fail (SF) or signal degrade (SD) detected at the ODUkT sublayer trail (TCM) for Linear OTN Protection modes SNC/S and SNC/Ns.

4.10.6 Device Capabilities for Transmission Protection

An OpenROADM device announces its transmission protection capabilities to the OpenROADM controller by means of *protection-profile* objects. For each available protection capability, the device creates a (read/only) *protection-profile* in the *org-openroadm-device/protection-profiles* container.

Interfaces of type *ethernetCsmacd* or *otnOdu* that support some kind of transmission protection announce this capability by pointing to one or more protection profiles using the *interface/common-functions/protection-profile-name* attribute. More than one interface object can point to the same protection profile. Only interfaces refering to the same protection profile (same *profile-name*) can be part of the same protection group.

The tree view of the protection profile object is as follows:

```
+--ro protection-profiles
|  +--ro protection-profile* [profile-name]
|     +--ro profile-name
|     +--ro level
|     +--ro prot-type
|     +--ro switching-direction
|     +--ro mode-list*
|     +--ro xc-capabilities-list*
|     +--ro pg-capabilities-list*
```

Each profile provides the following information attributes:

profile-name: is the unique name to identify the protection profile. Profile names are assigned by the device.

level: indicates whether the announcing entity supports *line*, *path* or *client* protection. *line* and *path* protection is implemented by the *odu-sncp-pg* protection group, *client* protection by the *client-sncp-pg* protection group.

prot-type: shows the protection type supported by the announcing interface. Currently only *odu-one-plus-one* is defined.

switching-direction: either *unidirectional-switching* or *bidirectional-switching*.

mode-list: used for line and path level profiles only, shows the list of supported protection modes as a subset of { SNC/Ne, SNC/Ns, SNC/S, SNC/I }. For client level profiles, the SNC/I mode is implicit.

xc-capabilities-list: list of cross-connection types supported by the CPs of a corresponding protection group. Possible values: { *unidirectional-xc*, *bidirectional-xc*, *select-and-bridge-xc*, *drop-and-continue-xc* }.

pg-capabilities-list: list optional feautures supported by a corresponding protection group. *sd-detection* indicates support for signal degrade (SD) detection according to ITU-T G.873.1 section 6.1. *revertive* indicates support for revertive switching as defined in ITU-T G.873.1 section 8.3.

4.10.7 1+1 Line Protection Switching

1+1 Line Protection provides protection of a single link, ensuring a break in the working line is automatically switched to the protect line (refer to Figure 41).

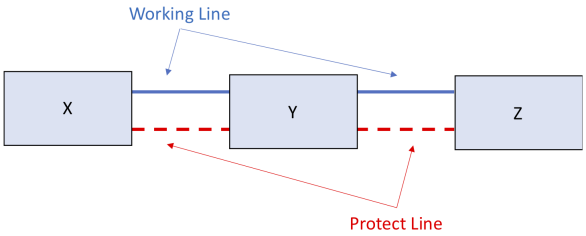

Figure 41: 1+1 Line Protection

Within a node, the 1+1 line protection group working interface is assigned to interface A and the protection group interfaces are assigned to interfaces A,B (refer to Figure 42). All LO ODUj (*ODU-CTP*) instances are assigned to the A interface (*ODU-TTP*).

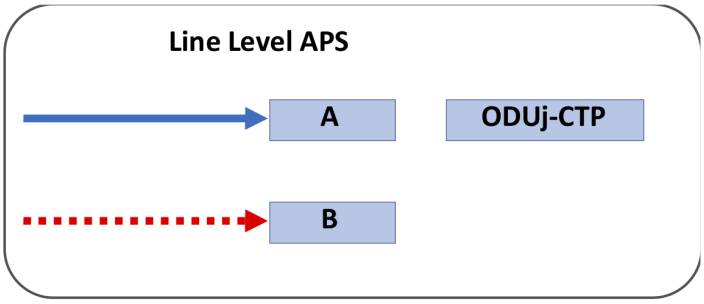

Figure 42: Line Level Automatic Protection Switching (APS) Model

A single operation is used to create a 1+1 line protection group that identifies the working and protect interfaces. (Note: these examples are to show high level concepts and are not intended to be the exact attributes that would be required. Please refer to section 4.10.9 for the full list of mandatory and optional attributes).

```
<edit-config>

create Line PG (odu-sncp-pg):
name=<assigned by controller>
level=line
prot-type=odu-one-plus-one
mode=SNC/S or SNC/I (as per protection-profile)
working-if=A
pg-interfaces={A,B}

Commit
```

4.10.8 Client SNC/I Protection/Y-Cable Protection

SNC/I Protection/Y-Cable Protection is defined in ITU-T G.873.1 (03/2020) App II.3 [8]. To protect a client signal, it is split between two different ports in the client-to-network direction. This split signal is carried across the OTN as two separate, unprotected signals. At the far end, the two signals are terminated and the client signals are recovered. One or the other client signal is transmitted, based on monitoring of the ODUk overhead (including OPU-CSF). OpenROADM only supports the Y-cable variant (option (a) in G.873.1 App II.3) and a control process that monitors the ODUkP trail termination functions is used to determine which one provides the better signal and to control the client termination function such that only one of the two transmitters is active.

Figure 43 shows the involved functional entities.

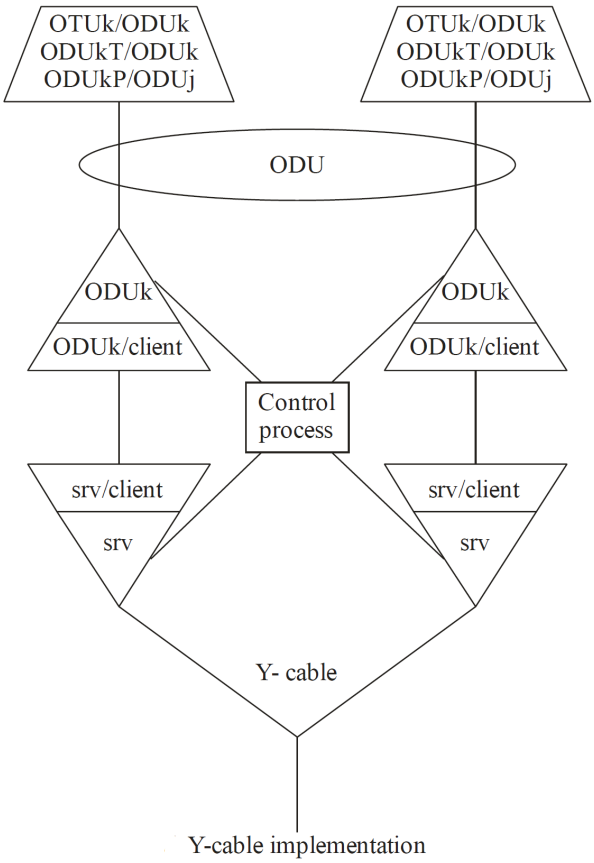

Figure 43: Client SNC/I Protection Model for Y-Cable

Client SNC/I Protection (Y-cable) is enabled in the OpenROADM model by creating a *client-sncp-pg* protection group. This protection group connects the two Ethernet interfaces that protect each other (*interface/type = ethernetCsmacd*). The *client-sncp-pg/pg-interfaces* leaf points to both of the interfaces, *client-sncp-pg/working-if* points to the working one. The other leaves of the protection group are described in section 4.10.9.

Figure 44 shows the overall architecture of a transmission system using Client SNC/I Protection with Y-Cable:

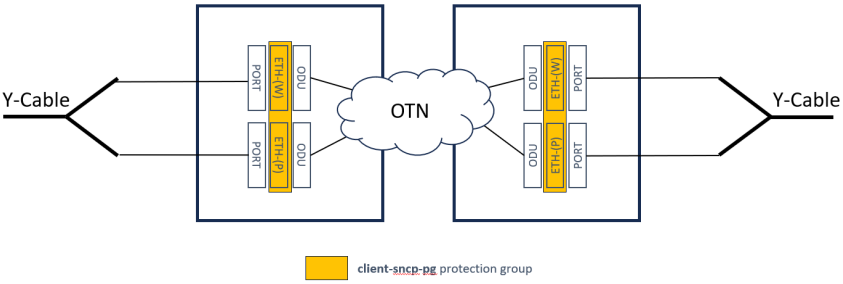

Figure 44: Client SNC/I Protection Architecture with Y-Cable

A single operation is used to create a client protection group that identifies the working and protect interfaces. (Note: these examples are to show high level concepts and are not intended to be the exact attributes that would be required. Please refer to section 4.10.9 for the full list of mandatory and optional attributes).

<edit-config>

create Client PG (client-sncp-pg):
name=<assigned by controller>
working-if=ETH(W)
pg-interfaces={ETH(W),ETH(P)}

Commit

4.10.9 Attributes of a Transmission Protection Group

The OpenROADM MSA defines two versions of a transmission protection group:

- *client-sncp-pg* is used for Client SNC/I protection
- *odu-sncp-pg* is used for Line and Path protection

The following YANG tree shows their definitions:

```
+--rw protection-grps
 ...
|  +--rw client-sncp-pg* [name]
|     +--rw name
|     +--rw switching-direction?
|     +--rw revertive?
|     +--rw sd-enable?
|     +--rw wait-to-restore?
|     +--rw holdoff-timer
|     |  +--rw holdoff?
|     |  +--rw holdoff-multiplier?
|     +--rw working-if
|     +--rw pg-interfaces*
|     +--ro active-if?
|     +--ro request-state?
|
|  +--rw odu-sncp-pg* [name]
|     +--rw name
|     +--rw level
|     +--rw prot-type?
|     +--rw mode
|     +--rw tcm-layer
|     +--rw switching-direction?
|     +--rw revertive?
|     +--rw sd-enable?
|     +--rw wait-to-restore?
|     +--rw holdoff-timer
|     |  +--rw holdoff?
|     |  +--rw holdoff-multiplier?
|     +--rw working-if
|     +--rw pg-interfaces*
|     +--ro active-if?
|     +--ro request-state?
  ...
```

Most of the attributes are shared between the two protection group versions. The following definitions indicate when an attribute is specific to one of the protection group types:

- *name* is the unique name to identify the protection group. Protection group names are assigned by the controller.
- *level* (odu-sncp-pg only) designates the protection group as either *line* or *path*.
- *prot-type* (odu-sncp-pg only) indicates the protection type and defaults to *odu-one-plus-one*.
- *mode* (odu-sncp-only) sets the protection mode and can be one of SNC/Ne, SNC/Ns, SNC/S or SNC/I.
- *tcm-layer* (odu-sncp-only) defines the TCM layer for SNC/S and SNC/Ns protection (ODUT1..6). For other protection modes, this attribute must be absent.
- *switching-direction* sets the direction of protection switching to either *unidirectional-switching* (default) or *bidirectional-switching* (APS signaling enabled).
  For details see ITU-T G.873.1 Table 7-1 and 8-1 [8].
- *revertive* is a boolean attribute to set the reversion behavior (default=false).
  For details see ITU-T G.873.1 section 8.3 [8].
- *sd-enable* is a boolean attribute to enable SD detection. Default is *true*, if the PG does support SD detection. If SD detection is not supported, this attribute must be absent or be set to *false*.
  For details see ITU-T G.873.1 section 6.1 [8].
- *wait-to-restore* specifies the wait-to-restore time in minutes when *revertive=true*. Range is 1..12 minutes with a default value of 5 minutes.
- *holdoff-timer*: The switch-holdoff time is specified in msec by the formula *holdoff* * *holdoff-multiplier*. *holdoff* can be 0 (default), 20 or 100 msec. *holdoff-multiplier* supports the range of 1 (default) .. 100.
- *working-if* points to the designated ”working” ODU or ETH interface object.
- *pg-interfaces* points to all interfaces that are part of this protection group, including the working interface.
- *active-if* is a state variable and points to the currently active interface.
- *request-state* shows the current request state of the protection group, with a value of:
  LOP-NE, LOP-FE, SF-NE, SF-FE, FS-NE, FS-FE, SD-NE, SD-FE, MS-NE, MS-FE,
  WTR-NE, WTR-FE, EXER-NE, EXER-FE, DNR-NE, NR-NE.
  In case of uni-directional switching only the ”-NE” states apply.
  For details see ITU-T G.873.1 Table 9-2 and 9-3 [8].

4.10.10 1+1 Path Protection Switching

1+1 Path Protection provides end-to-end protection, ensuring a failure occurring at any point along the path will cause the end nodes to switch the traffic to the protection path. The Open ROADM MSA defines Sub-Network Connection Protection (SNCP) as the dedicated 1+1 path protection mechanism (*protection-grps/odu-sncp-pg*).

SNCP's functional equivalent in SONET is Uni-directional Path Switched Ring (UPSR). Like UPSR, SNCP transmits the data signal along two different paths (working path and protect path). The same data signal is transmitted on both paths; working traffic flows along one path and protection traffic flows in the the other path. Switching occurs at the end of the path and is triggered by defects or alarms along the path. The head-end node in the protection protocol performs a bridge operation, while the tail-end node receives two copies of the electrical signal at the path layer, compares them, and performs a selector operation, by selecting the better-quality signal (refer to Figure 45). Note: SNCP does not require that the two paths be completely diverse.

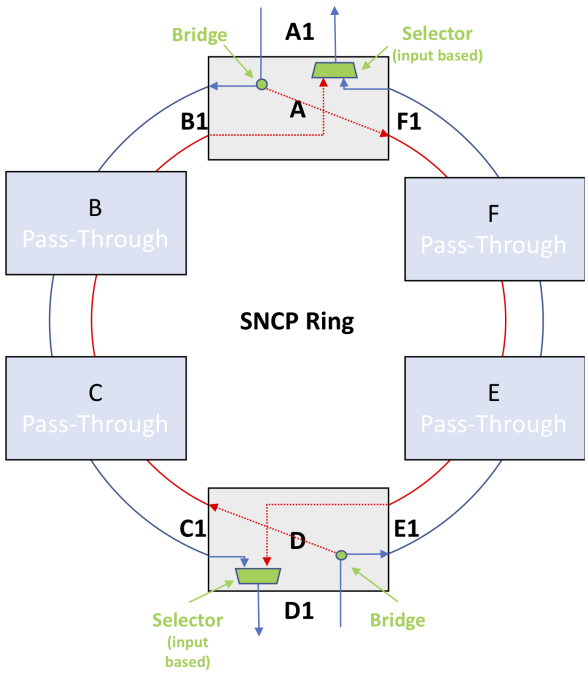

Figure 45: 1+1 Path Protection - SNCP Ring Configuration

With 1+1 path protection, a protection group must contain a working path and a protection path. In addition, the paths with the same drop interface must be attached to the same protection group (refer to Figure 46). Before the protect path crossconnect can be accepted, the associated protection group must exist. The Open ROADM MSA model requires support for bi-directional cross-connects; uni-directional cross-connect support is deemed optional (see Section 4.8.6.1).

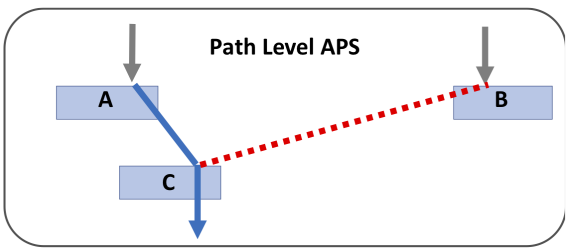

Figure 46: Path Level Automatic Protection Switching (APS) Model

4.10.11 1+1 Path Protection Group Creation, Modification and Deletion

Path protection groups can be part of many different service management scenarios. The following list shows the most typical examples:

- Creation of a protected service from scratch (section 4.10.12)
- Deletion of a protected service (section 4.10.13)
- Replace working and protecting path (section 4.10.14)
- Add protection to a service (section 4.10.15)
- Remove protection from a service (section 4.10.16)

To implement these (and other) scenarios, an OpenROADM device supports a minimum set of configurations related to path protection groups. Figure 47 below shows these 7 configurations. All transitions between any two of these configurations are possible with a single *edit-config* or NETCONF *commit*.

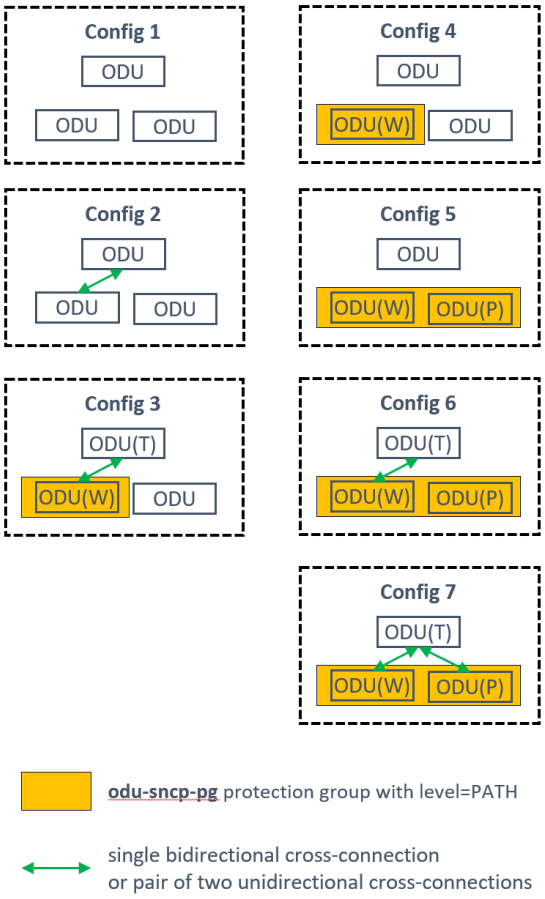

Figure 47: Path Protection Switching Configurations

4.10.12 Example 1: Creation of a Protected Service from Scratch

To create a protected service in a switchponder from scratch, ”Config 7” in Figure 47 must be reached. This can be done in a single step, or individually using a sequence like:
Config 1 → Config 5 → Config 6 → Config 7

The most efficient procedure is to create the working and protect drop cross-connections and the associated protection group in a single operation (see example below).

<edit-config>
[Config 1]

create Path PG (odu-sncp-pg):
level=path
working-if=ODU(W)
pg-interfaces={ODU(W), ODU(P)}

create XCON ODU(W)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(W) dst-if=ODU(T)

create XCON ODU(P)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(P) dst-if=ODU(T)

Commit

[Config 7]

4.10.13 Example 2: Deletion of a Protected Service

The deletion of a path protection group and of the two related cross-connections is shown by the example below.

<edit-config>
[Config 7]

delete XCON ODU(P)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(P) dst-if=ODU(T)

delete XCON ODU(W)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(W) dst-if=ODU(T)

delete Path PG (odu-sncp-pg):
level=path
working-if=ODU(W)
pg-interfaces={ODU(W), ODU(P)}

Commit

[Config 1]

4.10.14 Example 3: Replace Working and Protecting Path

This example (Figure 48) shows the following reconfiguration of a protection group:

- Step 1: Make the protecting path the new working path of the PG. This is a reconfiguration within ”Config 7”.
- Step 2: Delete the old working path. Note that this operation will switch to the remaining path of the PG automatically. This is a reconfiguration from ”Config 7” to ”Config 3”.
- Step 3: Add a new protecting path to the PG. This is a reconfiguration from ”Config 3” back to ”Config 7”.

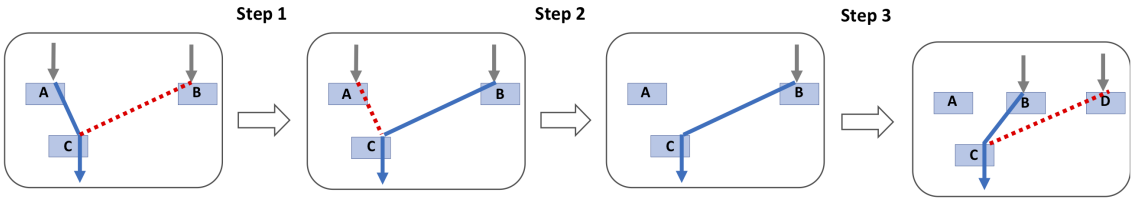

Figure 48: Existing Path Level APS Protection Group Change Migration

Step 1: Make the protecting path B the new working path

<edit-config>
[Config 7]

edit Path PG (odu-sncp-pg):
level=path
working-if=B
pg-interfaces={A, B}

Commit

[Config 7]

Step 2: Delete the old working path A

<edit-config>
[Config 7]

delete XCON A-C (odu-connection):
direction=bi-directional
src-if=A dst-if=C

edit Path PG (odu-sncp-pg):
level=path
working-if=B
pg-interfaces={B}

Commit

[Config 3]

Step 3: Add new protecting path D

<edit-config>
[Config 3]

edit Path PG (odu-sncp-pg):
level=path
working-if=B
pg-interfaces={B, D}

add XCON D-C (odu-connection):
direction=bi-directional
src-if=D dst-if=C

Commit

[Config 7]

4.10.15 Example 4: Add Protection to a Service

This example adds a protecting path to an existing service. This scenario is an extension of a simple cross-connection (”Config 2”) by a path protection group and a protecting path, ending up at ”Config 7”.

<edit-config>
[Config 2]

add Path PG (odu-sncp-pg):
level=path

working-if=ODU(W)
pg-interfaces={ODU(W), ODU(P)}

add XCON ODU(P)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(P) dst-if=ODU(T)

Commit

[Config 7]

4.10.16 Example 5: Remove Protection from a Service

This example removes protection, i. e. the protection path from a service. It is a reconfiguration from a ”Config 7” to a ”Config 2”.

<edit-config>
[Config 7]

delete XCON ODU(P)-ODU(T) (odu-connection):
direction=bi-directional
src-if=ODU(P) dst-if=ODU(T)

delete Path PG (odu-sncp-pg):
level=path
working-if=ODU(W)
pg-interfaces={ODU(W), ODU(P)}

Commit

[Config 2]

4.10.17 SNCP Back-to-Back Ring Topologies (Single Node)

Connecting SNCP rings in a back-to-back ring topology provides an extra level of path protection between rings by eliminating single points of failure. There are various ways to configure interconnected SNCP rings. One example is to use a single node to interconnect SNCP rings, as shown in Figure 49, where the traffic signal is broadcasted at the interconnecting node, resulting in duplicate traffic signals transmitted between the rings at the interconnecting node.

The protection group and cross-connect setup at the interconnecting node with uni-directional cross-connects would be as follows:

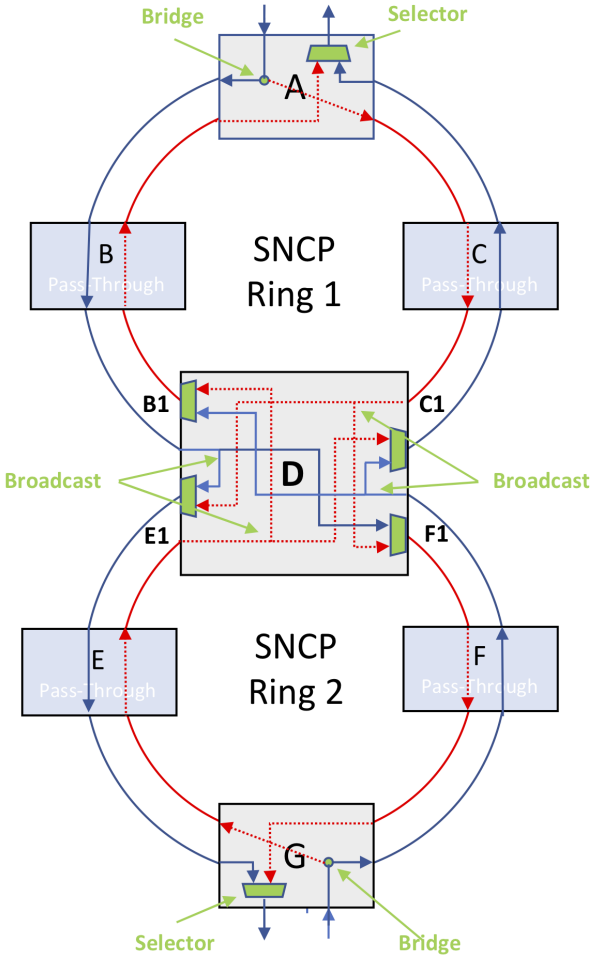

Figure 49: SNCP Single Node Dual Ring Interconnect

The topology shown in Figure 49 can support both, **dual-ring** interconnect and **segmented** protection. In the dual-ring case, ports B1 and C1 are part of ring 1, ports E1 and F1 are part of ring 2. In the case of segmented protection, ports B1 and C1 terminate segment 1, ports E1 and F1 terminate segment 2.

The configuration of protection groups and cross-connections of node D (Figure 49) is shown in Figure 50 below. All transitions between any two of these configurations are possible with a single *edit-config* or NETCONF *commit*. All ODU interfaces in the configurations must have a protection profile supporting *unidirectional* and *select-and-bridge* cross-connection capabilities.

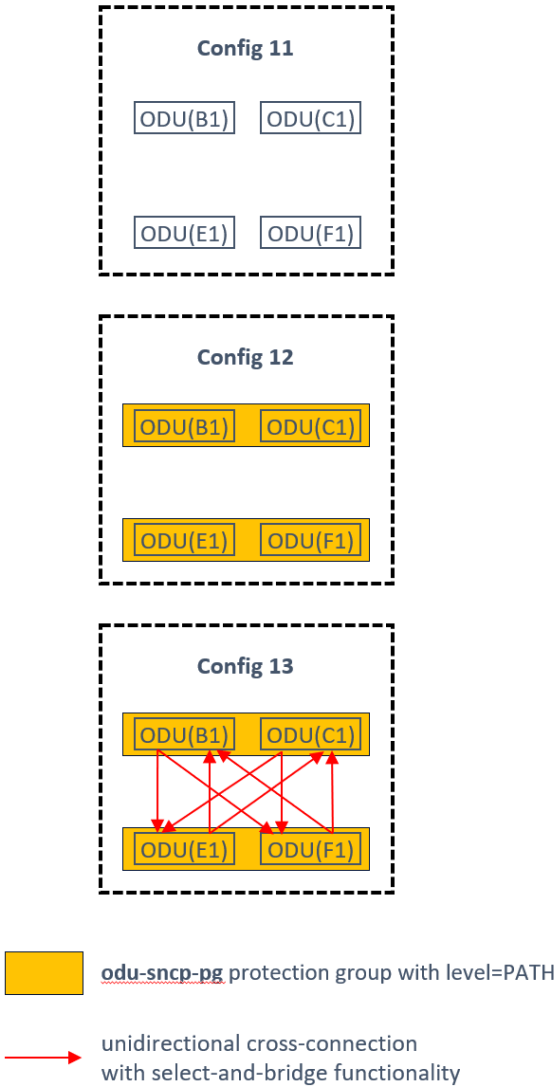

Figure 50: Single Node Dual-Ring Interconnect and Segmented SNC Configurations

The operations to configure the protection setup in Figure 50 are shown in the following example:

<edit-config>
[Config 11]

create Path PG1 (odu-sncp-pg):
level=path
working-if=B1
pg-interfaces={B1,C1}

create Path PG2 (odu-sncp-pg):
level=path
working-if=F1
pg-interfaces={F1,E1}

create XCON B1-F1 (odu-connection):
direction=uni-directional
src-if=B1 dst-if=F1

create XCON B1-E1 (odu-connection):
direction=uni-directional
src-if=B1 dst-if=E1

create XCON C1-F1 (odu-connection):
direction=uni-directional
src-if=C1 dst-if=F1

create XCON C1-E1 (odu-connection):
direction=uni-directional
src-if=C1 dst-if=E1

create XCON F1-B1 (odu-connection):
direction=uni-directional
src-if=F1 dst-if=B1

create XCON F1-C1 (odu-connection):
direction=uni-directional
src-if=F1 dst-if=C1

create XCON E1-B1 (odu-connection):
direction=uni-directional
src-if=E1 dst-if=B1

create XCON E1-C1 (odu-connection):
direction=uni-directional
src-if=E1 dst-if=C1

Commit

[Config 13]

4.10.18 SNCP Back-to-Back Ring Topologies (Four Nodes)

Using four nodes to interconnect SNCP rings is an example described in ITU-T G.842 [10] (see Figure 51). However, in order to support this configuration, uni-directional cross-connect support is required. In this example, the data signal is dropped and continued at the interconnected nodes, resulting in duplicate data signals transmitted between the rings at the interconnected nodes.

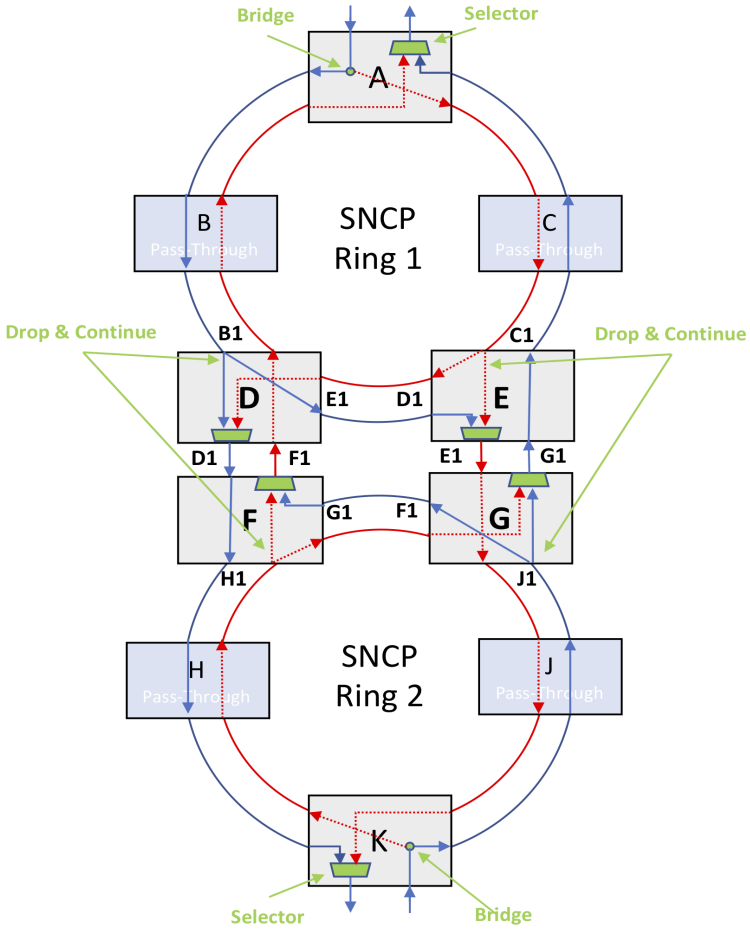

Figure 51: SNCP Four Node Dual Ring Interconnect (ITU-T G.842 [10])

The protection group and cross-connect setup at the interconnecting nodes D, E, F and G is symmetrical and is shown in Figure 52 below using node D as example. All transitions between any two of these configurations are possible with a single *edit-config* or NETCONF *commit*. All ODU interfaces of this configuration must have a protection profile supporting the *unidirectional* cross-connection capability. In addition, ODU interface ”B1” in the example below must support the *drop-and-continue* capability.

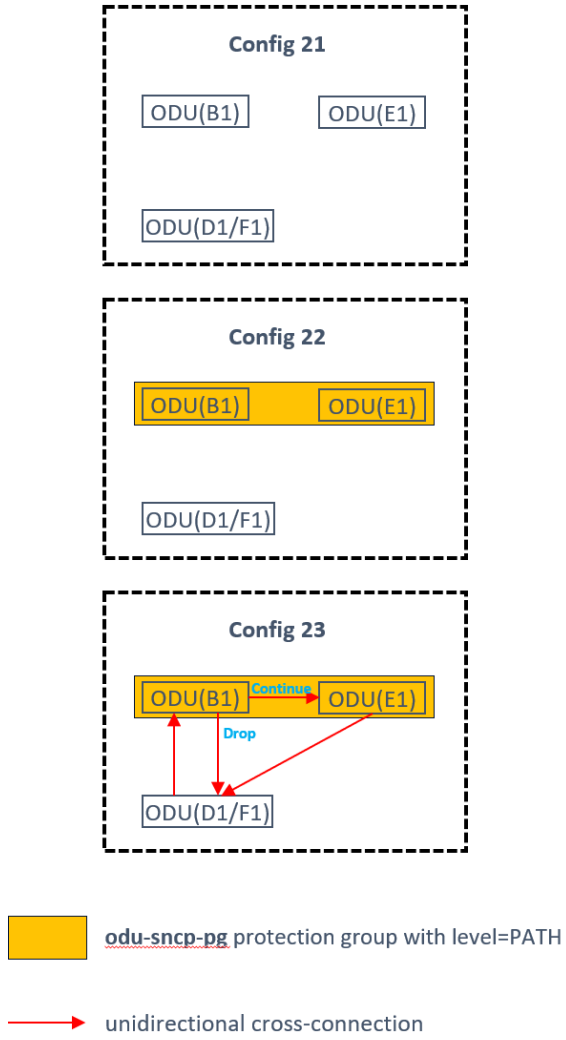

Figure 52: Quad Node Dual-Ring Interconnect

The operations to configure the protection setup in Figure 52 are shown in the following, using node D as example:

<edit-config>
[Config 21]

create Path PG (odu-sncp-pg):
level=path
working-if=B1
pg-interfaces={B1,E1}

create drop XCON B1-D1 (odu-connection):
direction=uni-directional
src-if=B1 dst-if=D1

create drop XCON E1-D1 (odu-connection):
direction=uni-directional
src-if=E1 dst-if=D1

create continue XCON B1-E1 (odu-connection):
direction=uni-directional
src-if=B1 dst-if=E1

create pass-through XCON F1-B1 (odu-connection):
direction=uni-directional
src-if=F1 dst-if=B1

Commit
[Config 23]

4.11 Protocols Model

The Open ROADM MSA YANG model supports LLDP for link discovery over the OSC (WDM layer), RSTP for communication via the L2 DCN, and IPv4/IPv6 DHCP relay for remote transponder connectivity via GCC0.

```
+--rw protocols
|  +--rw lifecycle-state?
|  +--rw ipv4-dhcp-relay
|  |  +--rw ipv4-server-group* [server-group-name]
|  |     +--rw server-group-name
|  |     +--rw interface-name*
|  |     +--rw server-address*
|  +--rw ipv6-dhcp-relay
|  |  +--rw ipv6-server-group*
|  |     +--rw server-group-name
|  |     +--rw interface-name*
|  |     +--rw server-address*
|  +--rw gnmi
|  |  +--rw enabled?
|  |  +--rw certificate-id?
|  |  +--rw port?
```

YANG sub-tree 45: Protocols (ipv4, ipv6, and gnmi)

```
+--rw protocols
    ...
|  +--rw lldp
|  |  +--rw global-config
|  |  |  +--rw adminStatus?
|  |  |  +--rw msgTxInterval?
|  |  |  +--rw msgTxHoldMultiplier?
|  |  +--rw port-config* [ifName]
|  |  |  +--rw ifName
|  |  |  +--rw adminStatus?
|  |  +--ro nbr-list
|  |     +--ro if-name* [ifName]
|  |        +--ro ifName
|  |        +--ro remoteSysName?
|  |        +--ro remoteMgmtAddressSubType?
|  |        +--ro remoteMgmtAddress?
|  |        +--ro remotePortIdSubType?
|  |        +--ro remotePortId?
|  |        +--ro remoteChassisIdSubType?
|  |        +--ro remoteChassisId?
...
```

YANG sub-tree 46: Protocols (LLDP)

```
+--rw protocols
    ...
|  +--rw rstp
|     +--ro max-bridge-instances?
|     +--rw rstp-bridge-instance* [bridge-name]
|        +--rw bridge-name
|        +--rw rstp-config
|        |  +--rw bridge-priority?
|        |  +--rw shutdown?
|        |  +--rw hold-time?
|        |  +--rw hello-time?
|        |  +--rw max-age?
|        |  +--rw forward-delay?
|        |  +--rw transmit-hold-count?
|        |  +--rw rstp-bridge-port-table* [ifname]
|        |     +--rw ifname
|        |     +--rw cost?
|        |     +--rw priority?
|        +--ro rstp-state
|           +--ro rstp-bridge-attr
|           |  +--ro root-bridge-port?
|           |  +--ro root-path-cost?
|           |  +--ro root-bridge-priority?
|           |  +--ro root-bridge-id?
|           |  +--ro root-hold-time?
|           |  +--ro root-hello-time?
|           |  +--ro root-max-age?
|           |  +--ro root-forward-delay?
|           |  +--ro bridge-id?
|           |  +--ro topo-change-count?
|           |  +--ro time-since-topo-change?
|           +--ro rstp-bridge-port-attr
|              +--ro rstp-bridge-port-table* [ifname]
|                 +--ro ifname
|                 +--ro bridge-port-state?
|                 +--ro bridge-port-role?
|                 +--ro bridge-port-id?
|                 +--ro oper-edge-bridge-port?
|                 +--ro designated-bridge-port?
|                 +--ro designated-bridgeid?
```

YANG sub-tree 47: Protocols (RSTP)

4.12 Users Model

The default username/password is openroadm/openroadm.

```
+--rw users
|  +--rw user* [name]
|     +--rw name
|     +--rw password?
|     +--rw group?
```

YANG sub-tree 48: Username and Password

Only one role, that grants full access, is defined on the network element; the Controller is expected to manage user roles.

4.13 Tandem Connection Monitoring (TCM) Model

TCM overhead supports monitoring of arbitrary sub-network connections. Per ITU-T G.709 [4], each independent ODU trail (path) is supported by an end-to-end path monitor, as well as, six levels of tandem connection monitors (TCM1, TCM2, . . . TCM6). The number of active TCM levels along an ODUk trail may vary between 0-6. TCM functions maybe nested or cascaded along any particular ODU trail. TCM functions nested within the same ODU trail must use different TCM levels. Typically, TCM 6 is used to cover the shortest network span and TCM 1 is used to cover the longest network span. In Figure 53 (ITU-T G.709 [4]), monitored connections A1-A2/B1-B2/C1-C2 and A1-A2/B3-B4 are considered nested, while B1-B2/B3-B4 are considered cascaded.

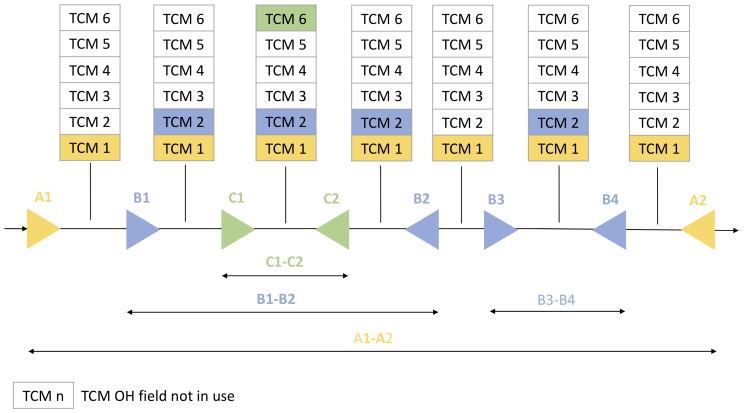

Figure 53: Nested and Cascaded ODUk Monitored Connections (ITU-T G.709 [4]

The Open ROADM MSA supports a variety of attributes that defines a *tcm* layer.

- ***layer***: defines which TCM layer (1. . .6) to use
- ***tcm-direction***: defines the transmit direction of the *tcm* layer, where *up-tcm* is set for TCM terminations facing toward the switch fabric, *down-tcm* is set for TCM terminations facing toward the facility (away from the switch fabric). Refer to Figure 54 for an example. It should be noted that upgrading from Open ROADM MSA 1.1 requires the *tcm-direction* to be set for existing *tcm* layers.
- ***monitoring-mode***: defines whether the *tcm* layer is terminated or monitored
  - ***terminated***: the TCM layer has detection and generation enabled and the overhead is erased (replaced with all zeros) towards the downstream
  - ***monitored***: the TCM layer has detection enabled and the overhead is transparently passed through the interface in both directions
- ***ltc-act-enabled***: defines whether the tcm layer alarm transfer on detection of Loss of Tandem Connection (LTC) is enabled or disabled. The default is disabled.
- ***proactive-delay-measurement-enabled***: defines whether the tcm layer Proactive Delay Measurement is enabled or disabled. The default is disabled.

Note: At the same ODU interface, the number of TCM instances is limited to 12 (6 up, 6 down).

4.14 Trail Trace Identifiers (TTI) Model

Trail Trace Identifiers (TTI) are used to identify a signal from source to destination in an OTN network (similar to the J0 byte of SONET/SDN networks). The TTI is a 64-byte multi-frame signal that occupies one byte of the frame aligned with the OTUk multi-frame and transmitted four times per multi-frame. The TTI includes the Source Access Point Identifier (SAPI) and Destination Access Point Identifier (DAPI), which contain information regarding the country of origin, service provider, and administration details. The TTI byte is carried in the OTUk Section Monitoring (SM) overhead, ODUk Path Monitoring (PM) overhead, and ODUk Tandem Connection Monitoring (TCM) overhead.

For adjacency discovery, the identifier (TTI) that is sent out of the interface (*otn, odu, odu/tcm*) is configurable by the controller and the TTI received from the interface is retrievable by the controller. Once known by the controller, the TTI can be used to discover or confirm network topology.

The operator or controller provisions the 15-character transmitted API strings (*tx-sapi, tx-dapi*) and the 32-character transmitted operator specific string (*tx-operator*). The operator or controller may provision the 15-character expected API strings (*expected-sapi, expected-dapi*) if TTI mismatch detection is desired. Note: implementations shall automatically add the SAPI[0] and DAPI[0] fields (fixed to all-0s) per ITU-T G.709 [7].

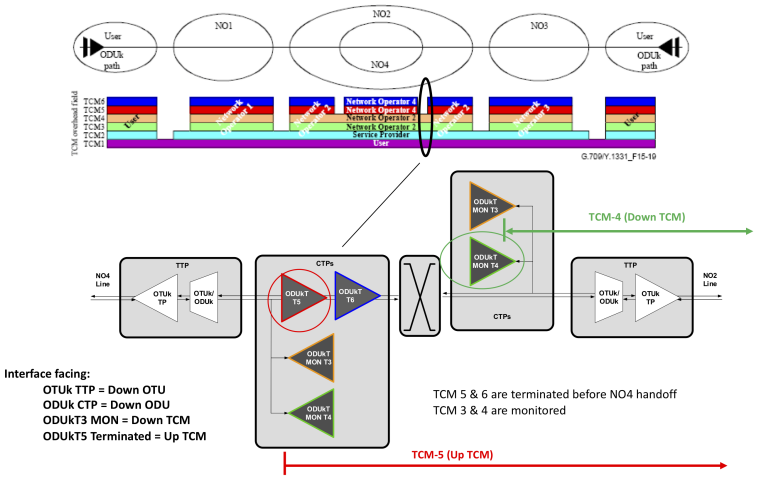

Figure 54: TCM direction

The received API strings (*accepted-sapi, accepted-dapi, accepted-operator*) are retrieved from the received multi-frame TTI field. An *otn-tti-info-change notification* (*resource-type, otn-interface-type, otn-interface-tcm-layer, otn-interface-tcm-direction, accepted-sapi, accepted-dapi, accepted-operato*) is generated when any changes to the received TTI attributes (*accepted-sapi, accepted-dapi, accepted-operator*) occur.

A TTI Mismatch (TIM) alarm (*trailTraceIdentifierMismatch*) occurs when the received string (*accepted-sapi and/or accepted-dapi*) is different than the expected provisioned string (*expected-sapi and/or expected-dapi*). The following TIM detection modes (*tim-detect-mode*) are modeled against a device:

- Disabled: TTI is ignored
- SAPI: the expected-sapi is compared to the accepted-sapi (other TTI fields are ignored)
- DAPI: the expected-dapi is compared to the accepted-dapi (other TTI fields are ignored)
- SAPI-and-DAPI: the expected-sapi and expected-dapi are compared to the accepted-sapi and accepted-dapi (accepted-operator is ignored)

Enabling of the TIM consequent action (e.g. ODU AIS) is achieved via the attribute *tim-act-enabled=true*.

4.15 Beyond 100G (B100G) Model

For rates beyond 100Gb/s (B100G), such as 200Gb/s, 300Gb/s, and 400Gb/s, a new optical transport unit called an OTUCn was defined by ITU-T G.709 [4] (i.e. for rates too large for an OPU4). The OTUCn contains an optical data unit (ODUCn) which contains an optical payload unit (OPUCn). No client signals are directly mapped into an OPUCn. They must first be mapped into ODUk (including ODUflex, refer to Section 4.17.2), which is then mapped or multiplexed into the OPUCn. The OPUCn payload area is divided into Tributary Slots (TS) with each client ODUk occupying an integer number of TS. The optical channel data tributary unit (ODTUCn.ts, ts = 1 to 20n) is directly byte synchronously mapped (BMP) into a set of OPUCn TS numbered 1.1 to n.20 (#A.20). TSs can be arbitrarily selected to prevent bandwidth fragmentation. ODTUC.ts tributary ports are numbered 1 to 10n providing mapping of a maximum number of ODU2/ODU2e. The Open ROADM MSA recommends to minimize mapping ODU0/ODU1 into OPUCn (5G TS).

Since different B100G interface types have different FEC performance capability requirements, the FEC is specified on a per-interface basis as opposed to making the FEC overhead an integral part of the frame structure like the fixed dedicated FEC overhead of the OTUk frame format. As a result, OTUCn signal bit rates do not include a FEC overhead area; the OTUk signal bit rates include the FEC overhead area.

The ODUCn is always a transport layer signal that only carries signals that have been mapped into lower rate ODUk signals (k=0,1,2,2e,3,4,flex). Furthermore, an ODUCn signal is only carried point-to-point between network nodes. In other words, the ODUCn signal is only a Multiplex Section layer entity that cannot be switched; it only exists to carry LO ODUk signals between a pair of nodes, with all switching being done at the ODUk level.

Since switching does not occur, it does not require TCM in the same way that a switched ODUk does. However, for regenerator applications, it is still valuable for determining the performance of the optical signals on each side of the regenerator. Due to the multiple optical segments and the different service providers along the OTUCn path, multiple levels of TCM are still required. As a result, B100G signals use the same TCM levels (TCM1..TCM6) defined for ODUk signals (refer to Section 4.13 OTN Tandem Connection Monitoring (TCM)).

The Open ROADM MSA models the OTUCn rate as *otu:otucn-n-rate*, which specifies the n associated with an OTUCn (Tree 49).

```
+--rw otu
+--rw otucn-n-rate?

[update / review needed]
```

YANG sub-tree 49: OTUCn rate

```
+--rw odu!
...
+--rw oducn-n-rate?
....
+--rw parent-odu-allocation!
+--rw trib-port-number
+--rw trib-slots-choice?
+--:(opu)
+--rw trib-slots*
+--:(opucn)
+--rw opucn-trib-slots*
...

[update / review needed]
```

YANG sub-tree 50: ODUCn trib slots

An ODUCn rate is modeled as *odu:oducn-n-rate* , which specifies the n associated with an ODUC. The corresponding Tributary Port Number (1. . .10n) and Tributary Slots (#A.B, A=1. . .n, B=1. . .20) are defined by *odu-port-number* and *opucn-trib-slots* respectively.

4.15.1 Sub-Rate OTUCn (OTUCn-M)

Sub-rates are not defined for the OPUCn and ODUCn; they are defined in terms of n $\times$ 100 Gb/s signals. An OTUCn-M frame is a type of OTUCn frame that contains n instances of OTUC, ODUC, and OPUC overhead and M of the 5 Gb/s OPUCn TS (20$\times$n tributary slots). OTUCn-M sub-rates can accommodate an OTUCn with an incremental 25Gb/s bit rate, for example:

1. *125G OTUC2-25*
2. *150G OTUC2-30*
3. *200G OTUC2(-40)*
4. *250G OTUC3-50*
5. *300G OTUC3(-60)*
6. *350G OTUC4-70*
7. *400G OTUC4(-80)*

Using sub-rate 250Gb/s (OTUC3-50) as an example, 10 of the TS are not to be used. ITU G.709 [7] leaves it up to the vendor to choose which TS are assigned to carry each of the OTUCn instances. In order to ensure interoperability, the Open ROADM MSA recommends to not use the last TS $(20 \times n - M)$ in the OPUCn for the OTUCn sub-rates (OTUCn-M).

The Open ROADM MSA models the OTUCn-M rate as *otu:otucn-M-rate* which is used to specify the M number of 5Gbs OTUCn TS associated with OTUCn-M.

4.15.2 Optical Tributary Signal (OTSi)

Rather than Optical Channel (Och), standards have adopted the use of Optical Tributary Signal (OTSi) for B100G. OTSi is an optical signal that is placed within a network media channel (NMC) that is transported through a series of media channels across the optical network. OTSi interfaces are modeled as (Tree 52):

FlexO is modeled under OTSi interfaces (refer to Section 4.16 for FlexO details):

An OTSi Group (OTSiG) is a group of optical tributary signals carrying the OTUCn. There is an OTSiG instance per OTUCn rate or sub-rate OTUCn-M. Network ports associated with different OTUCn must announce different OTSiG instances. When *org-openroadm-device:type = openROADM-if:otsi-group*, the *osti-group* container is used:

```
+--rw otu
...
+--rw otucn-n-rate?
+--rw otucn-M-rate?
...

[update / review needed]
```

YANG sub-tree 51: OTUCn-M

```
+--rw otsi
+--rw provision-mode?
+--rw otsi-rate?
+--rw otsi-member-id?
+--rw frequency?
+--ro orgwidth?
+--rw modulation-format?
+--rw transmit-power?
+--rw fec?
+--rw optical-operational-mode?
+--ro operational-mode-params
+--ro spectral-width?
+--ro supported-group-if?
...

[update / review needed]
```

YANG sub-tree 52: OTSi Interface

```
+--rw otsi
...
+--rw flexo!
+--....

[update / review needed]
```

YANG sub-tree 53: FlexO

```
+--rw otsi-group
+--rw provision-group-rate?
+--rw otsi-group-id?

[update / review needed]
```

YANG sub-tree 54: OTSi group

```
+--ro otsigroup-capability-profile* [profile-name]
|  +--ro profile-name
|  +--ro if-cap-type?
|  +--ro flexo-payload-type?
|  +--ro (otu-rate)?
|  |  +--:(otucn)
|  |  |  +--ro otucn-n-rate?
|  |  +--:(nxotu4)
|  |     +--ro supported-n-otu4?
|  +--ro foic-type*
|  +--ro otn-capability-profile-name?
|  +--ro otn-odu-mux-hierarchy-profile-name?
```

YANG sub-tree 55: OTSiG capabilities

It should be noted that the *otsi-group-id* is mandatory for B100G FlexO (refer to Section 4.16 for FlexO details); it is not required for non-FlexO B100G. A series of capabilities are defined against an OTSiG (Tree 55):

4.15.3 Logical Port

The Open ROADM MSA device model has defined a *logical-port* to represent a logical bonding of the physical ports supporting an OTSiG (refer to Figure 55). The *logical-port* can be modeled as a physical hardware port or a true logical port that doesn't exist in the hardware.

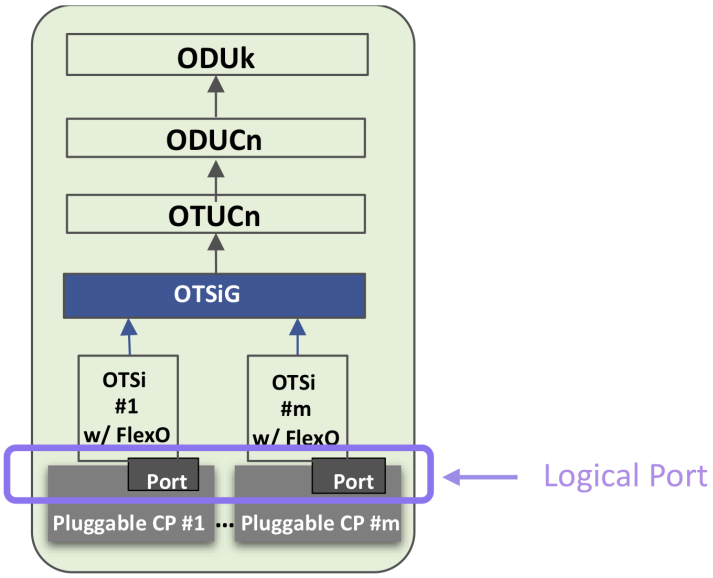

Figure 55: Logical port

The *logical-port* can be driven by the planned template or automatically created by the device when the associated OTSiG is provisioned (see Section 4.15.2 for OTSiG details). The logical-port is advertised via the *port-capabilities* announcement, composed of a physical *circuit-pack-name* and a *logical port-name*.

Figure 56 shows a hypothetical device that has two xponder entities. Each xponder has 4 client-side 100GE interfaces (C1...C4, C5...C8) and 4 line-side interfaces (E1...E4, E5...E8) with the modulated optical signal (OTUC4) transmitted over four OTSi interfaces in an OTSiG. Based on Figure 56 Table 17 shows the B100G interfaces and how to populate the associated attributes.

4.16 FlexO Model for B100G

In order to provide a flexible, modular mechanism to support different line rates with B100G signals, the ITU-T has defined a similar modular interface approach to the OIF FlexE (B100G Ethernet signals) [11] for B100G OTN signals called FlexO (ITU-T G.709.1 [12], ITU-T G.709.3 [9]). FlexO takes advantage of being able to use existing 100GE/OTU4 optical modules for the individual FlexO PHYs. A FlexO modular interface may consist of one of the following bonded together to carry an OTUCn, with each 100Gb/s FlexO frame carrying an OTUC slice:

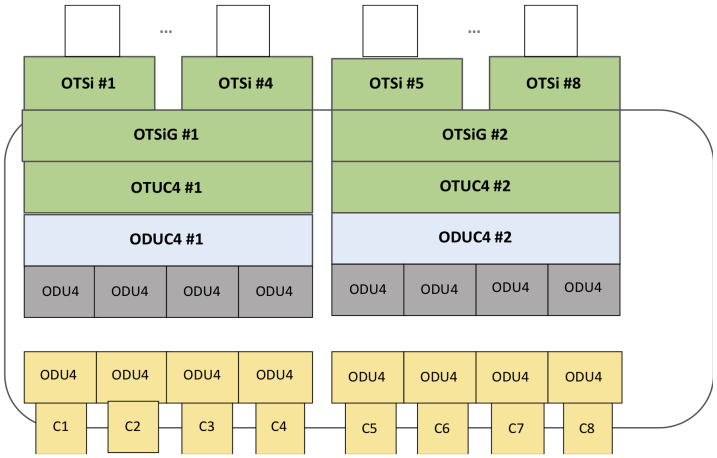

Figure 56: Xponder example

interface	supporting-interface-list	supporting-port	supporting-circuit-pack-name
ODUC4 #1	OTUC4 #1	logical port	circuit-pack
OTUC4 #1	OTSiG #1		
OTUC4 #1	OTSi #1, #2, #3, #4		
OTSi #4	-	E4	Pluggable hosting E4
OTSi #3	-	E3	Pluggable hosting E3
OTSi #2	-	E2	Pluggable hosting E2
OTSi #1	-	E1	Pluggable hosting E1

Table 17: B100G Interface Attributes Example; specified by Controller, generated by device

- a set of 100Gb/s optical PHY streams (n 100Gb/s PHYs)
- a set of 200Gb/s optical PHY streams (m 200Gb/s PHYs, $m = \lceil n/2 \rceil$[5])
- a set of 300Gb/s optical PHY streams (m 300Gb/s PHYs, $m = \lceil n/3 \rceil$) for line-side only
- a set of 400Gb/s optical PHY streams (m 400Gb/s PHYs, $m = \lceil n/4 \rceil$)

The FlexO electrical interface (FOIC) between the circuit-pack and the pluggable optics (e.g. CFP2-DCO) is defined as FOICx.k, where Cx indicates the interface rate and k indicates the number of PHY lanes being used (i.e. FlexO lane distribution FOIC/OTSi). Some 100G FlexO instances may be unequipped, for example, carrying an OTUC3 over a 400G module with one unequipped FlexO instance. Unequipped instances are placed at the end of the 200G or 400G PHY. Referring to Figure 57, FlexO-x-DO (high performance soft-decision FEC implementation) enables the ability to decouple

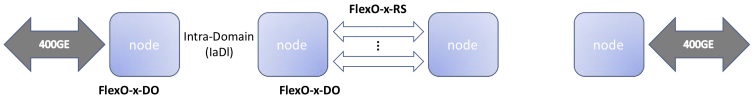

Figure 57: FlexO Line Intra-Domain Interface (IaDI) and FlexO Client Inter-Domain Interface (IrDI)

the line bandwidth from wavelength capacity and FlexO-x-RS (standard Reed Solomon FEC implementation) enables an inter-domain interface that allows a hand-off between two OTN transport networks. Note: FlexO-1-DO/RS refers to the 100G frame structure ,FlexO-2-DO/RS refers to the 200G frame structure, and FlexO-4-DO/RS refers to the 400G frame structure.

The Open ROADM MSA requires support for the FlexO long reach line group (see Section 4.16.1) and optionally, the FlexO short reach group (see Section 4.16.2).

FlexO is modeled under OTSi interfaces (Tree 56):

Note: The accepted-group-id and accepted-iid attributes were incorrectly added to the Open ROADM MSA model (v7.1) as read-write (rw) attributes instead of the intended read-only (ro) attributes. The model above reflects the corrections that will be made in the next release of the Open ROADM MSA model. A device may choose to populate or not populate these attributes. An Open ROADM MSA v7.1 controller should not crash or take any action based on these attributes.

[5] $\lceil x \rceil$ = ceiling(x)

```
+--rw interface* [name]
| +--rw name
| +--rw description?
| +--rw type
| +--rw lifecycle-state?
| +--rw administrative-state
| +--ro operational-state
| +--rw circuit-id?
| +--rw supporting-circuit-pack-name?
| +--rw supporting-port?
| +--rw supporting-interface-list*
| +--rw otsi
| | +--rw flexo!
| | +--rw foic-type
| | +--rw iid*
| | +--ro:accepted-group-id*
| | +--ro accepted-iid*

[update / review needed]w
```

YANG sub-tree 56: FlexO under OTSi

4.16.1 FlexO Long-Reach Line Interfaces (FlexO-x-DO)

The FlexO long-reach (FlexO-x-DO) line interface group supports bonding (i.e. grouping) of multiple of these interfaces such that an OTUCn (n $\geq$ 1) can be transferred via one or more OTSi over one or more physical interfaces (refer to Figure 58). FlexO-x-DO interfaces use FEC types with a higher coding gain than the KP4 FEC defined for FlexO-x-RS interfaces. The Open ROADM MSA, with a focus on service provider applications (metro/LH), agreed to standardize on a high performance soft-decision FEC, called openFEC (oFEC), and align with the terminology in the latest version of the ITU-T G.709.3 specification (FlexO-x-DO). Refer to the W-Port Digital Specification (200G-400G) for further details [3].

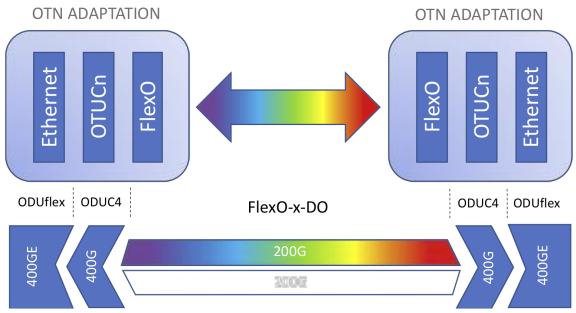

Figure 58: FlexO-x-DO Bonding

In the example shown in Figure 59, a 100G member would be a single OTSi that is contained in an OTSiG. Note: an OTUC4 would be transmitted over four OTSi (FOIC1.4-DO) contained in an OTSiG. The k lane signals are modulated onto one OTSi, which is transported via one media element. A FlexO-x-DO interface is simply a handoff between the electrical and optical domains, where the virtual lane FOIC bits get mapped to some constellation (i.e. QPSK modulation).

The FlexO interface type (*foic-type*) attribute is used to identify the type of FOIC interface (e.g. *foic1.4, foci2.4, foic3.6, foic4.8*, where, 4=QPSK, 6=8QAM, and 8=16QAM modulation scheme) and the FlexO Instance Identifier (*iid*) attribute is used to identify each FlexO frame/instance interleaved into the FlexO-x-DO interface of a group with different *iid* values carried in multiple FlexO instances (refer to ITU G.709.1 [12]) transported over the same media element.

4.16.2 FlexO Short-Reach Client Interfaces (FlexO-x-RS)

Support for the FlexO short-reach interface group (FlexO-x-RS) is considered optional. Referring to Figure 60, FlexO-x-RS is defined for interoperable multivendor applications by providing an interoperable interface for OTUCn transport signals over bonded:

- 100GBASE-R pluggable optics by mapping one 100G FlexO instance over each 100G PHY, or
- 200GBASE-R pluggable optics by interleaving two 100G FlexO instances over each 200G PHY, or

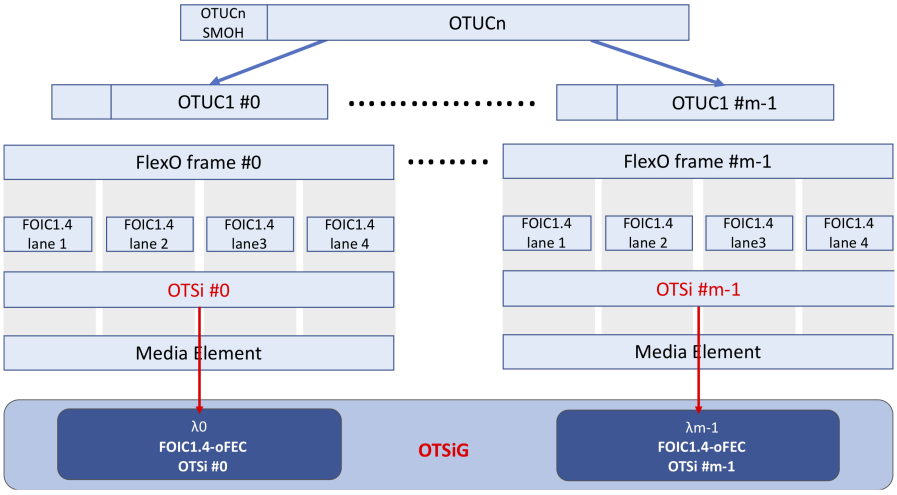

Figure 59: FlexO-x-DO 100G Line Interface Containment Example

- 400GBASE-R pluggable optics by interleaving four 100G FlexO "'instances" over each 400G PHY

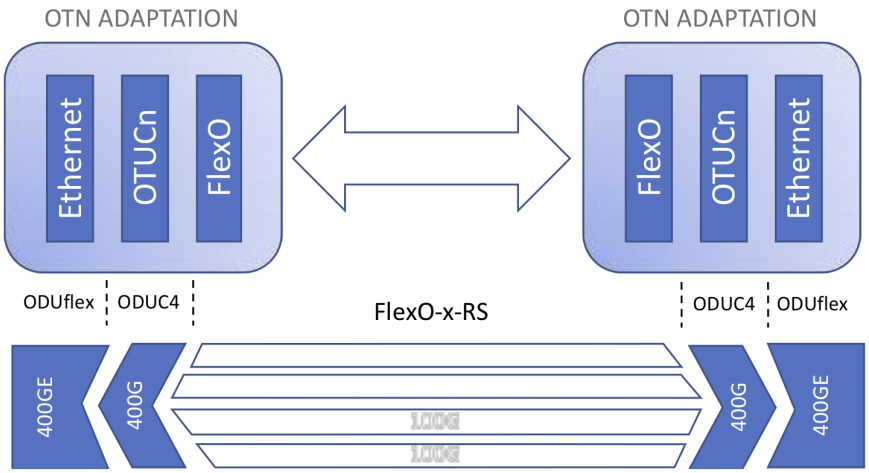

Figure 60: FlexO-x-RS Sub-Rating and Channelization

The FlexO lane architecture is based on 400GE and uses the same FEC used in both 100GE and 400GE (KP4 Reed-Solomon (RS) FEC defined by IEEE 802.3-2018 clause 91 [4]) is used for each 100 Gb/s FlexO stream). With FlexO-x-RS, multiple member interfaces are bonded together that consist of multiple lanes (OTSi). For example, a FlexO-x-RS 100G interface can be used for a 4-lane QSFP28 pluggable optic (refer to Figure 61). Note: FOIC2.4 would be used for a FlexO-x-RS 200G interface (e.g. 4-lane CFP2 pluggable optic) and FOIC4.8 would be used for a FlexO-x-RS 400G interface (e.g. 8-lane CFP2 pluggable optic).

The Open ROADM Forum identified no applications to model an OTSi interface for each lane. There is no user specification required on FlexO-x-RS OTSi interface (e.g. modulation format, rate , FEC, transmit power, frequency). As a result, the Open ROADM MSA model of the FlexO short reach interface abstraction (refer to Figure 48: FlexO-x-RS 100G Client Interface Containment Example) deviates from the way the ITU-T G.709.1 [12] modeled the interface. For simplicity, it was agreed to align the FlexO short reach interface model with the FlexO long reach line interface model, where an OTSi interface is modeled per FlexO interface (FOICx.k).

With the Open ROADM MSA model, the OTSi interface is considered a parent port with power monitoring performed at each lane over a child sub-port (refer to Section 4.2 and Figure).

The FlexO interface type (*foic-type*) attribute is used to identify the type of FOIC interface (e.g. *foic1.4, foci2.4, foic3.6, foic4.8*) and the FlexO Instance Identifier (*iid*) attribute is used to identify each FlexO frame/instance interleaved into the FlexO-x-RS interface of a group with different *iid* values carried for each of the FlexO instances transported over the same media element.

4.17 Flexible Rate ODU (ODUflex) Model

In order to transport certain client formats, such as Fibre Channel, video signals, and variable rate packet flows, the concept of a flexible rate ODU container (ODUflex) was introduced (ITU-T G.709 [4]. ODUflex is a single container that cannot be

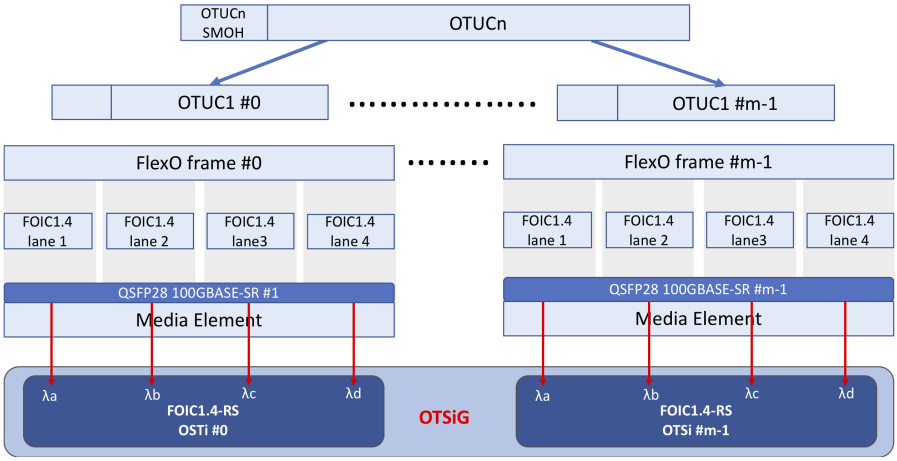

Figure 61: FlexO-x-RS 100G Client Interface Containment Example

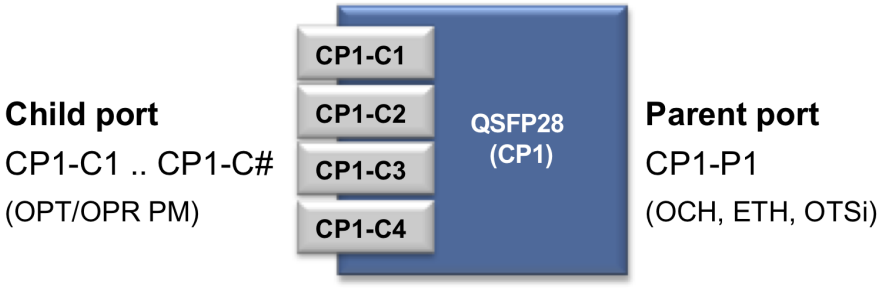

Figure 62: FlexO-x-RS Port Model Abstraction

split across multiple fibre links or wavelengths. ODUflex provides a single manageable entity across the OTN at a permanent fixed-rate (ODUflex (CBR)), or adjusted to changing connectivity demands in the network (ODUflex(GFP), ODUflex(IMP), ODUflex(FlexE)). The ODUflex mapping method is based on the rate and content of the client signal.

4.17.1 ODUflex for 100Gb/s and Below

4.17.1.1 ODUflex(CBR)

ODUflex(CBR) is defined for client transport signals that are Constant Bit Rate (CBR) signals. Only CBR clients greater than 2.488Gb/s are mapped into ODUflex(CBR) using the Bit-synchronous Mapping Procedure (BMP), clients less than 1.22Gb/s are mapped into an ODU0 using the Generic Mapping Procedure (GMP) and clients between 1.244Gb/s and 2.488Gb/s are mapped into an ODU1 using GMP.

The Open ROADM MSA model currently supports 25Gb/s, 200Gb/s, and 400Gb/s CBR signals. For CBR signals, the odu attribute rate=ODUflex-CBR and the *odu* attribute *oduflex-cbr-service* identifies the specific CBR signal rate, where *oduflex-cbr-service* can be *ODUflex-cbr-25G* for 25G CBR signals as defined in ITU-T G.709 17.13.1 [7], *ODUflex-cbr-200G* and *ODUflex-cbr-400G* for 200G/400G CBR signals as defined in ITU-T G.709 17.13.2 [7] and discussed in Section 4.17.1.2).

4.17.1.2 ODUflex(GFP)

ODUflex(GFP) is defined to carry packet data flows (signals that do not have a constant bit rate) encapsulated with the Generic Framing Procedure (GFP). Typically, these signals would be Ethernet and sub-rate Ethernet, but any packet-oriented data can be encapsulated using GFP and mapped into an ODU. Rates of the ODUflex(GFP) container are multiples of 1.25Gb/s frames corresponding to the capacity of an integer number of HO ODUk Tributary Slots (TS). The packet flow is adapted to that rate using GFP. Mapping of GFP frames into and ODUflex is exactly the same as mapping GFP frames into a fixed rate ODUk.

For GFP-F mapped client signals, the Open ROADM MSA model uses the odu attribute *oduflex-gfp-num-ts* to specify the number of TS used. The oduflex-gfp-ts-bandwidth attribute specifies the nominal TS minimum bit-rate (Mbps) per ITU-T G.709 Table 7-8 [4]. Note: the *oduflex-gfp-ts-bandwidth* attribute was incorrectly added to the device model as a read-write attribute, but it should be read-only. A subsequent version of the device model will have the oduflex-gfp-ts-bandwidth attribute correctly defined as read-only.

Note: Sub-rate Ethernet policing is only supported at the receive interface (ingress) Referring to the example in Figure 63,

a sub-rate 5 Gbps is transported over 6.25 Gbps (5×ODU2.ts = 5×1.249 = 6.245 Gbps). The controller pushes down the same provisioning at both Xponders.

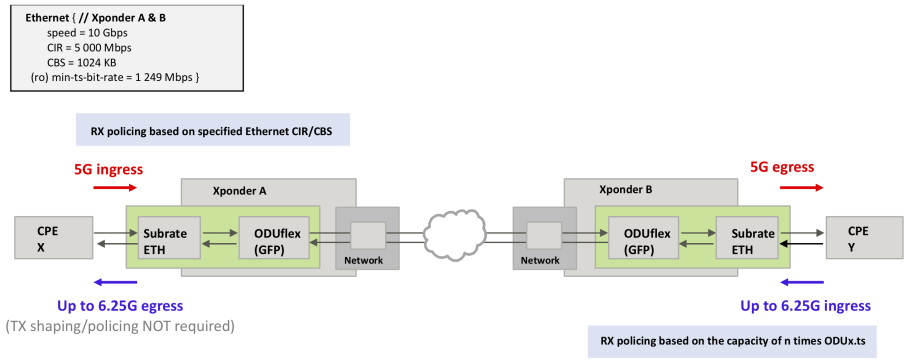

Figure 63: Sub-rate Ethernet Receive Policing/Rate-Limiting Example

4.17.2 ODUflex for B100G

4.17.2.1 ODUflex(CBR) for 25GE, 200GE, 400GE

There is no change to the definition of ODUflex(CBR) for 25GE, 200GE, and 400GE CBR signals (refer to Section 4.17.1.1). Per ITU-T G.709 Amd 2 clause 17.13 [7], ODUflex(CBR) is created by wrapping the client signal with a current OTN frame. A BMP (Bit-synchronous Mapping Procedure) process is used that simply adds OTN overhead to the client signal stream. CBR clients with rates B100G (including 25GE) are mapped into an ODUflex, which in turn is multiplexed into an OPUCn by first asynchronously mapping it into an intermediate structure called an Optical channel Data Tributary Unit (ODTUCn).

For 25GE CBR signals, the ODUflex rate is directly derived from the client signal rate and mapped/multiplexed into a number of OPUCn TS or ODUk TS adequate to carry that signal. ODUk signals can be multiplexed directly into the new ODUCn. In general, clients with rates below the OPU4, including ODU0 or ODU1-mapped clients, can also be carried in the OPUCn by two-stage multiplexing allowing the aggregation of multiple low rate clients into a higher rate current ODUk (k = 2, 3, 4), which can then be multiplexed into the OPUCn.

For 200GE/400GE CBR signals, the ODUflex rate is directly derived from the client signal rate and mapped/multiplexed into a number of OPUCn TS adequate to carry that signal. CBR clients with rates B100G are mapped into an ODUflex, which in turn is multiplexed into an OPUCn by first asynchronously mapping it into and intermediate structure called an ODTUCn. The OPUCn payload area is divided into tributary slots (TS) with each client ODUk occupying an integer number of TS. The ODTUCn is directly byte synchronously mapped (BMP) into a set of OPUCn TS 128-bit sized words. The ODUflex(CBR) rate will have the 5Gbit/s granularity of the OPUCn TS rather than the 1.25Gbit/s granularity associated with the ODUk TS rate.

Clients with rates below the OPU4, including ODU0 or ODU1-mapped clients, can also be carried in the OPUCn by two-stage multiplexing allowing the aggregation of multiple low rate clients into a higher rate current ODUk (k = 2, 3, 4), which can then be multiplexed into the OPUCn.

4.17.2.2 ODUflex(IMP) for up to and B100G

The Idle insertion Mapping Procedure (IMP) is used to map packet clients (including FlexE) with rates 10Gb/s, 25Gb/s, 40Gb/s, 50Gb/s, 75Gb/s, 100Gb/s, and B100G into an OPUCn using a 64B/66B code word stream. Non-Ethernet packet clients are first encapsulated into Ethernet with the resulting 64B/66B stream from the ODUflex(IMP) mapping. ODUflex(IMP) are multiplexed into the OPUCn with a client agnostic generic mapping procedure (GMP) due to its ability to handle any client rate into any server rate. The Open ROADM MSA model uses the odu attribute oduflex-imp-s for IMP mapped client signals per ITU-T G.709 12.2.6 and Table 7-3 [4].

4.17.2.3 ODUflex(FlexE) for up to and B100G

A FlexE client defined by the Optical Interworking Forum (OIF) (OIF-FLEXE-02.0 [11] is a stream of 64B/66B characters associated with an Ethernet MAC packet flow, and occupies one or more repeating calendar slots. FlexE defines a CBR signal that is constructed as a FlexE Group carry one or more FlexE clients. There are three options for carrying the FlexE information over OTN:

1. Terminate the FlexE and carrying the FlexE clients as Ethernet clients over OTN by either using ODUflex (IMP) or the associated specified CBR mapping.
2. Simply treat each FlexE Group PHY as a 100GBASE-R signal and transport it accordingly (FlexE-unaware mapping).

3. Exploit calendar fill information within the FlexE stream to allow carrying a lower rate signal over OTN (FlexE-aware mapping).

For FlexE-unaware clients, each FlexE Group PHY is treated as a 100GBASE-R signal and mapped accordingly, while ODUflex(FlexE) is used for packet client rates B100G that are FlexE-aware. FlexE-aware clients are mapped into ODUflex(FlexE) using a bit-synchronous generic mapping procedure (BGMP). The Open ROADM MSA model uses the odu attribute oduflex-flexe-n for FlexE-aware client signals per ITU-T G.709 17.12 [4]. Note: Although the Open ROADM MSA supports the transport of FlexE clients in OTN, the Open ROADM MSA currently does not support FlexE interfaces.

4.18 Equipment Protection Model

The OpenROADM device model supports the management and control of hardware redundancy and equipment protection.

Supported equipment protection types are:

- 1:1 protection - one hardware unit is protected by another unit (active/standby)
- 1:N protection - one hardware unit protects N other units

The model supports the protection of *circuit-packs*. To describe equipment protection, the device model defines a list of *circuit-pack* protection groups (*circuit-pack-pg*, also called PG in the following).

Such a *circuit-pack-pg* protection group consists of:

- name of the *circuit-pack-pg* (key)
- a single list of descriptors defining the members of the protection group (*cp-pg-descriptor*). This list contains all protected and protecting *circuit-packs*.
- configuration and status attributes

A circuit pack descriptor (*cp-pg-descriptor*) represents a single circuit pack within a circuit pack protection group, and consists of:

- name of the *cp-pg-descriptor* (key)
- leafref to the underlying *circuit-pack*
- configuration and status attributes

The naming scheme of circuit pack descriptors is vendor specific, for example:

- re-use the name of the represented circuit pack
- simple numbers, e. g. ”1”, ”2”, ”3”, ...
- or any other scheme

The Yang model of circuit pack protection group and circuit pack descriptor is as follows:

```
+--rw circuit-pack-pg*       [name]
   +--rw name                string
   +--rw prot-architecture   prot-architecture-type
   +--ro service-state       pg-service-state-type
   +--ro redundancy-state    pg-redundancy-state-type
   +--ro lockout-state?      pg-lockout-state-type
   +--rw cp-pg-descriptor*      [name]
      +--rw name                string
      +--rw pg-circuit-pack     -> /org-openroadm-device/
                                    circuit-packs/circuit-pack-name
      +--ro member-state        pg-member-state-type
      +--ro switch-request      pg-member-switch-request-type
      +--ro member-active       boolean
```

4.18.1 Attributes of the Equipment Protection Group *circuit-pack-pg*

The protection architecture of an equipment protection group is configured with the *prot-architecture* parameter. This is an enumeration with values *one-for-one* (1:1 protection) and *one-for-n* (1:n protection with n $>$ 1). This attribute cannot be modified any more after a protection group has been created.

The service state of an equipment protection group is indicated by the *service-state* parameter. It can take the following values:

- *service-available*: the defined service of the PG is fully available
- *service-partially-available*: the service of the PG is partially available
- *service-temp-unavailable*: (optional) the PG's service is temporarily unavailable because more than one member has not yet been configured, is still initializing, or its state is under evaluation. This state will clear autonomously.
- *service-unavailable*: more than one member is in a failed state and the PG cannot provide its defined service

The redundancy state of an equipment protection group is provided by the *redundancy-state* parameter. It can take the following values:

- *redundancy-available*: additional redundancy is available in the PG. Note that a *locked* member can still provide redundancy, as the equipment protection model allows such a member to become active when needed to keep up the service of the PG.
- *redundancy-not-available*: redundancy is currently not available in the PG
- *configuration-incomplete*: (optional) the configuration of a redundant resource in the PG is not yet complete

The lockout state of an equipment protection group is an optional parameter and indicated by *lockout-state*. *lockout-state* is present for those protection groups where the members support the *lockout* switch request, see 4.18.5. It can take the following values:

- *clear*: none of the members are locked out
- *locked*: one or more of the members are in a *locked* state

4.18.2 Attributes of the Circuit Pack Descriptor *cp-pg-descriptor*

The member state of a circuit pack member of a PG is provided by the *member-state* parameter. It can take the following values:

- *member-ok*: the member is available for service
- *member-temp-failed*: (optional) the member is in a temporary failed state, because it has not yet been configured, is initializing, its database is currently being updated, or its state is under evaluation. The member is not available for service. This state will clear autonomously.
- *member-failed*: the member is in a failed state and out of service. The reason for the failure is indicated by state and alarms on the circuit pack itself.

Switch requests active on a single member of a PG are indicated by the *switch-request* parameter. It can take the following values:

- *no-request*: no switch request exists for the member
- *manual*: (optional) the member is manually switched to inactive and is not available to provide any services
- *locked*: (optional) the member is locked out, i. e. it is only considered for active service when absolutely needed to keep the PG in service
- *auto*: the member automatically switched to inactive because of fault(s) and/or demerits. The *member-state* is either *member-temp-failed* or *member-failed*

Whether a member is currently active or not, within the PG, is shown by the boolean *member-active* parameter.

Note: the combination *locked* and *member-active=true* is possible when the circuit pack is needed to keep up operation of the PG.

4.18.3 Partial Failures of Circuit Packs

In the standard failure scenario of an equipment protection group, the circuit pack descriptors indicate the following:

- an active circuit pack fails,
- its *member-state* changes to *member-failed*,
- its *switch-request* changes to *auto*,
- its *member-active* changes to FALSE,
- the protecting member of the PG takes over and changes its *member-active* to TRUE.

As an extension to this standard failure scenario, the OpenROADM device model also supports partial failures of circuit pack members of a PG. To express such a partial failure of a member, its circuit pack descriptor will show:

- an active circuit pack experiences a partial failure, and for some reason the PG cannot replace it completely by a protecting member,
- its *member-state* changes to *member-failed*,
- its *switch-request* changes to *auto*,
- its *member-active* stays at **TRUE**.

In such a partial failure case, the *service-state* of the whole PG might still be *service-available*, for example when two unrelated functions fail in two circuit packs.

4.18.4 Creation and Deletion of Equipment Protection Group and Circuit Pack Descriptor

Depending on device architecture, equipment protection groups and their member descriptors are auto-provisioned by the device, or must explicitly be created by the controller.

In case of auto-creation/ auto-provisioning by the device:

- the names of *circuit-pack-pg* and *pg-cp-descriptor* objects are fixed and assigned by the device
- the device auto-creates a *circuit-pack-pg* when at least one working and one protecting member exist
- the device auto-deletes a *circuit-pack-pg* when the last working or the last protecting member is deleted

If a circuit pack is member of a *circuit-pack-pg*, its representing *pg-cp-descriptor* is auto-created and auto-deleted when the circuit pack is created or deleted.

4.18.5 Operations on Equipment Protection Groups and Circuit Pack Descriptors

Equipment Protection Groups and Circuit Pack Descriptors can be actively managed by the controller using the *circuit-pack-protection-switch* RPC. It is defined as follows:

```
rpcs:
  +---x circuit-pack-protection-switch
     +---w input
     |  +---w circuit-pack-pg-name     -> /org-openroadm-device/protection-grps/
                                           circuit-pack-pg/name
     |  +---w cp-pg-descriptor-name    -> /org-openroadm-device/protection-grps/
                                           circuit-pack-pg[name=current()/
                                           ../circuit-pack-pg-name]/
                                           cp-pg-descriptor/name
     |  +---w switch-command           enumeration
     +--ro output
        +--ro status            rpc-status
        +--ro status-message?   string
```

The *circuit-pack-pg-name* and *cp-pg-descriptor-name* identify the equipment protection group and contained circuit pack descriptor on which the *switch-command* is executed. The following switch commands are supported by the device model:

- *lockout* - (optional) lock out the addressed PG member from active service and set *switch-request* to *locked*. A *locked* member is normally not used for active service. However, if the service of the PG can only be maintained using locked member(s), the PG can still make it active.
- *manual-switch* - (optional) remove the addressed PG member from active service and set *switch-request* to *manual*. A PG member with a *switch-request* set to *manual* is never used for active service.
- *release* - clear persistent switch request on the addressed PG member. Typically, only *lockout* and *manual-switch* in case of revertive protection are persistent

Note: support for *lockout* and/or *manual-switch* is optional and depends on the implementation of the device and possibly also on the type of member circuit packs in the PG.

4.18.6 Alarms and Notifications of Equipment Protection

Equipment protection currently does not define any specific notifications or alarms. Member circuit packs are covered by their standard equipment alarms, providing full operational support. To prepare for future extensions in the area of alarming or performance management, the device model already defines the *circuit-pack-pg* as a *resource*.

4.18.7 Configuration Examples

The following example shows a 1:1 equipment protection with no active switch request:

```
+--rw circuit-pack-pg*       [name]
   +--rw name                "Example-1"
   +--rw prot-architecture   one-for-one
   +--ro service-state       service-available
   +--ro redundancy-state    redundancy-available
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-1"
      +--rw pg-circuit-pack     "Card-1"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       true
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-2"
      +--rw pg-circuit-pack     "Card-2"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       false
```

The next example shows the same 1:1 equipment protection where one member failed and requests an auto switch:

```
+--rw circuit-pack-pg*       [name]
   +--rw name                "Example-2"
   +--rw prot-architecture   one-for-one
   +--ro service-state       service-available
   +--ro redundancy-state    redundancy-not-available
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-1"
      +--rw pg-circuit-pack     "Card-1"
      +--ro member-state        member-failed
      +--ro switch-request      auto
      +--ro member-active       false
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-2"
      +--rw pg-circuit-pack     "Card-2"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       true
```

This example shows an 1:1 equipment protection where one member has been locked out from service. Note that in this example, the PG still indicates *redundancy-available*, as the *locked* member can still be made active if needed.

```
+--rw circuit-pack-pg*       [name]
   +--rw name                "Example-3"
   +--rw prot-architecture   one-for-one
   +--ro service-state       service-available
   +--ro redundancy-state    redundancy-available
   +--ro lockout-state?      locked
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-1"
      +--rw pg-circuit-pack     "Card-1"
      +--ro member-state        member-ok
      +--ro switch-request      locked
      +--ro member-active       false
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-2"
      +--rw pg-circuit-pack     "Card-2"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       true
```

This example shows an 1:n equipment protection where member ”Card-1” failed and requests an auto switch:

```
+--rw circuit-pack-pg*       [name]
   +--rw name                "Example-4"
   +--rw prot-architecture   one-for-n
   +--ro service-state       service-available
   +--ro redundancy-state    redundancy-not-available
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-1"
      +--rw pg-circuit-pack     "Card-1"
      +--ro member-state        member-failed
      +--ro switch-request      auto
      +--ro member-active       false
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-2"
      +--rw pg-circuit-pack     "Card-2"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       true
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-3"
      +--rw pg-circuit-pack     "Card-3"
      +--ro member-state        member-ok
      +--ro switch-request      no-request
      +--ro member-active       true
```

The final example shows an 1:n equipment protection with two members failed and one member locked out. Whether such a protection group can deliver a partial service or no service any more depends on the concrete implementation.

```
+--rw circuit-pack-pg*       [name]
   +--rw name                "Example-5"
   +--rw prot-architecture   one-for-n
   +--ro service-state       service-partially-available
   +--ro redundancy-state    redundancy-not-available
   +--ro lockout-state?      locked
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-1"
      +--rw pg-circuit-pack     "Card-1"
      +--ro member-state        member-failed
      +--ro switch-request      auto
      +--ro member-active       false
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-2"
      +--rw pg-circuit-pack     "Card-2"
      +--ro member-state        member-failed
      +--ro switch-request      auto
      +--ro member-active       false
   +--rw cp-pg-descriptor*      [name]
      +--rw name                "Card-3"
      +--rw pg-circuit-pack     "Card-3"
      +--ro member-state        member-ok
      +--ro switch-request      locked
      +--ro member-active       true
```

5 DEVICE OPERATION

5.1 SYNCHRONOUS/ASYNCHRONOUS OPERATIONS

The Open ROADM MSA device model includes both synchronous and asynchronous operations; use depends on the completion duration.

5.1.1 Synchronous Operations

Synchronous (sync) operations complete when the RPC response is sent, indicating the success or failure of the operation; no additional notification is required (refer to Figure 64: Synchronous Operation). A synchronous operation is typically used for operations that complete in $<$ 2min

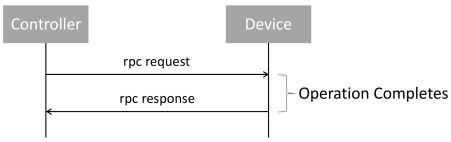

Figure 64: Synchronous Operation

5.1.2 Asynchronous Operations

Asynchronous (async) operations do not complete when the RPC response is sent; an additional notification (transient) is required to indicate the success or failure of the operation (refer to Figure 65: Asynchronous Operation).The RPC response for asynchronous operations usually indicates validation of the command and the command parameters, but not the success/failure of the operation itself. An asynchronous operation is typically used for operations that take $>$ 2min to complete.

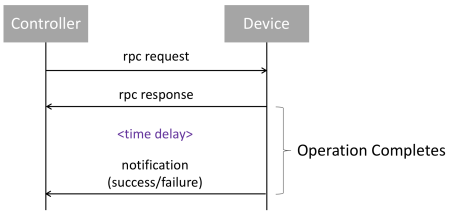

Figure 65: Asynchronous Operation

5.2 FILE OPERATIONS AND STRUCTURE

5.2.1 Local Device File Structure

The file system (associated with an SFTP operation) on the Open ROADM MSA device follows a flat structure with no subdirectories. When an operation generates multiple files, these would be tar'd then zip'd in a single file which is stored locally on the flat structure on the device.

The allocated space on the device shall be large enough to accommodate at least one copy of the files required to support operations requiring files exchange with the Open ROADM controller (ex: debug, syslogs, database for backup and restore, software loads for upgrade, OTDR scan, PM historical file for 15m, 24h and NA granularities, etc.)

5.2.2 Secure Shell File Transfer Protocol (SFTP) Specification

SFTP is a secure file transfer protocol that runs over Secure Shell (SSH) [4], it supports the full security and authentication of SSH and is used to exchange files between the Open ROADM controller and the devices. The Open ROADM device acts as the SFTP client and the Open ROADM controller acts as the SFTP server. In this scenario, the device would initiate the SFTP connection to the server using the IP address (associated with the domain subnet) in the received NETCONF transfer RPC, a file transfer to the device would translate in a "get" operation triggered by the device, and a file transfer from the device would translate in a "put" operation triggered by the device. The SFTP server port can be specified as an optional parameter in the URL of the transfer RPC, if a port is not specified the device shall use default port 22.

5.2.3 File Transfer (upload)

rpc: transfer

input *action*: upload (from device to Open ROADM Controller)

input *local-file-path*: file on the device

input *remote-file-path*: URI of file on the Open ROADM Controller including credentials

It was agreed that the operation be asynchronous with the *rpc-response-status* sent when the request is accepted and a transient notification sent once the operation is complete.

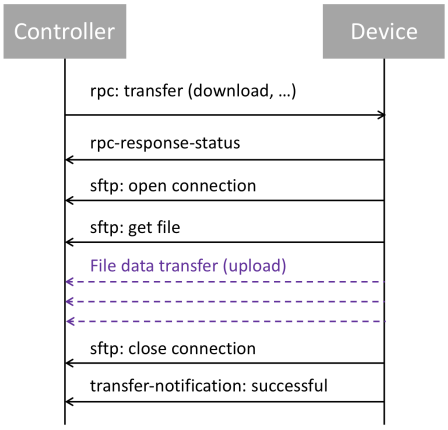

Figure 66: File Transfer (upload)

5.2.4 File Transfer (download)

rpc: transfer

input *action*: download (from Open ROADM Controller to device)

input *local-file-path*: file on the device

input *remote-file-path*: URI of file on the Open ROADM Controller including credentials

It was agreed that the operation be asynchronous with the *rpc-response-status* sent when the request is accepted and a transient notification sent once the operation is complete.

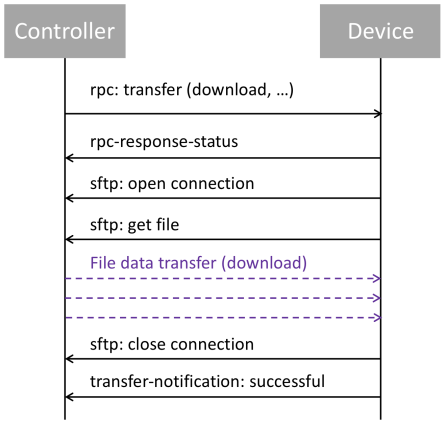

Figure 67: File Transfer (download)

5.2.5 Retrieve File List

rpc: show-file

Input optional *filename*: file(s) to be retrieved

Output *status*: Successful or Failed

Output optional *status-message*

Output list of files *(file)* identified by the filename that includes the size of the file (*file-size*) and a timestamp (*modified-date*) for each file listed

The file list operation is only applicable to the content of the flat Open ROADM MSA directory structure. In terms of operations it was agreed that:

- A “*” or blank (filename) input would list all the files in the Open ROADM MSA flat structure
- Only one filename can be specified as an input
- Partial wildcarding is not supported (ex: abc*)
- An empty list output would be represented by a success status with no file list

It was agreed that the operation be synchronous with the rpc-response-status sent when the request is accepted and the operation is complete.

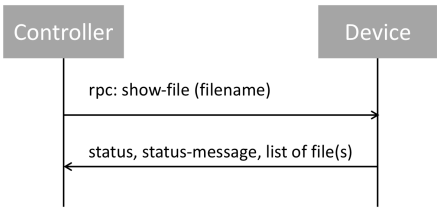

Figure 68: Retrieve File List

5.2.6 Delete File

rpc: delete-file

Input *filename*: name of file to delete

Output *rpc-response-status*

The file delete operation is only applicable to the content of the flat Open ROADM MSA directory structure. In terms of operations it was agreed that:

- Input wildcarding (ex: “*”) is not supported
- Only one filename can be specified as an input

It was agreed that the operation be synchronous with the *rpc-response-status* sent when the request is accepted and the operation is complete.

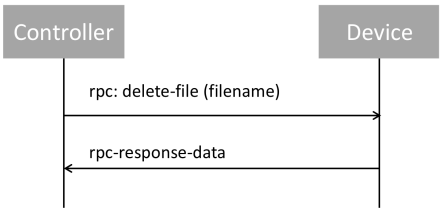

Figure 69: Delete file

5.2.7 File Operation Notifications

Transient notifications were not fully supported in the Open ROADM MSA Version 1.2.1 YANG models. As a short-term workaround, file operation transient notifications were sent as alarms with severity set to ‘indeterminate’. These notifications did not have a corresponding clear notification unlike a normal alarm. Therefore, in Open ROADM MSA Version 1.2.1, traditional alarms were not to be raised with the ‘indeterminate’ status.

This issue has been addressed in Open ROADM MSA Version 2. In this release, specific notifications were added to support transient notifications. For example, notifications were added for create tech info complete, file transfer status, ODU SNCP protection group switch, RSTP topology, and software download/database operations events. In Open ROADM MSA Version 2, the alarm severity ‘indeterminate’ should no longer be used to issue transient notifications.

5.3 DATA COMMUNICATION NETWORK (DCN)

Figure 70 shows the overview of the Open ROADM MSA DCN architecture, where each device acts as a layer 2 bridge and runs Rapid Spanning Tree (RSTP) to provide loop free topology.

For the ROADM Node:

- Each ROADM NE is running in L2 bridging mode

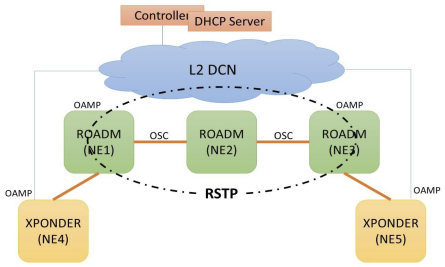

Figure 70: Open ROADM DCN Architecture

- NE1 and NE3 have OAMP connection to DCN while NE2 is only reachable via OSC
- Single bridge with Bridge ports OAMP, OSC1, OSC2, . . .
- RSTP is enabled on OAMP and OSC ports to provide loop free topology
- NE1 and NE3 can be reached via OAMP if RSTP has not blocked that port otherwise reachable via OSC.
- NE2 is reachable via OSC.
- Customer L2 DCN also participate in RSTP or transparently carries RSTP BPDUs if no loops exist in the DCN formed (e.g., appears as a transparent LAN).

For Xponder Node:

- Has single management port OAMP.
- Acts as a host since there is only single management port.
- Does not need to participate in the RSTP exchange as the transponder node is a host.

When a ROADM or xponder device is powered up, it runs IPv4 and IPv6 DHCP clients and gets an IP address from the DHCP server sitting on the same LAN. The IP address may be either IPv4 or IPv6 depending on the operator's DCN configuration. This IP address is called temporary IP address. When DHCP server allocates temporary IP address for a NE, then Controller is notified for this new IP address allocation. The controller can login to this NE using temporary address and can then provision the permanent IP address as per the service provider planning.

Provisioned IP address on the device can be IPV6 or IPV4. A device also allows the provisioning of the default gateway. The IP address and prefix length should be specified in the same edit config operation.

5.4 DATA COMMUNICATION NETWORK (DCN) – GCC Support

To support Remote transponder (and other) use cases (see figure 71) the DCN architecture of section 5.3 is extended by GCC links. GCC links are defined by ITU-T G.7712 [13] and extend the DCN network by inband communication links:

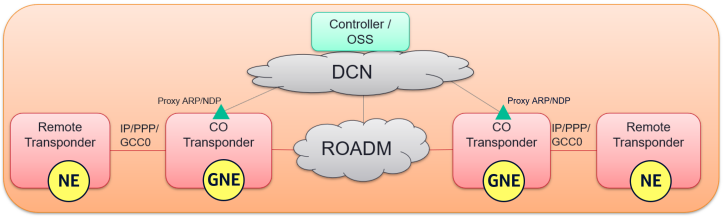

Figure 71: DCN Architecture with Remote Transponder and GCC

In this architecture:

- A single *CO Transponder* serves as Gateway Network Element (GNE) for several *Remote Transponders* (NE)
- The GNE is attached to the global DCN via its OAMP Ethernet interface
- The NEs are connected to their GNE via a single-hop GCC link
- Packet routing between the Ethernet DCN and the GCC-connected NEs uses ProxyARP (for IPv4) and/or ProxyNDP (for IPv6)
- To set up initial connectivity for remote NEs, *Light Touch Provisioning*, see section 6.4, is used

5.4.1 GCC Configuration Profile

OpenROADM uses the following GCC profile:

- GCC is used as defined by ITU-T G.7712 [13] clause 8.2.2
- IPv4 and IPv6 frames are mapped into PPP (RFC 1661 [14], RFC 1332 [15] and RFC 5072 [16]), PPP frames are transmitted bit-synchronously using HDLC framing (RFC 1662 [17]) using the ServerLayer/COMMS_A function of ITU-T G.798 [18] section 13.3.3
- In this release only GCC0 is supported (ServerLayer OTUk or OTUCn). In case of the OTUCn server layer, the standard option to only use the first GCC0 (first OTUCn instance) is used, see ITU-T G.709-2020 [7], NOTE in section 15.7.2.2

In the Yang model the GCC layer is modelled by an interface sub-container. The PPP layer is not currently modelled as no provisionable parameters are defined in this MSA release:

```
+--rw if:interface* [name]
   ...
   +--rw gcc!
   |  +--rw gcc-channel-type    enumeration (gcc0, gcc1, gcc2, gcc1-gcc2)
```

5.4.2 PPP Configuration Profile

The PPP encapsulation and the Link Control Protocol (LCP) are defined by RFC 1661 [14]. The following fixed protocol configuration is used:

- The MRU uses a fixed value of 1500 octets (section 6.1)
- Authentication and Quality Protocol are not required to be supported (sections 6.2 and 6.3)
- The *Magic Number* is chosen by the device “in the most random manner possible” (section 6.4)
- Protocol Field Compression and Address and Control Field Compression are not required to be supported (sections 6.5 and 6.6)
- PPP LCP Extensions of RFC 1570 [19] are not required to be supported

The PPP LCP echo supervision uses hard-coded, vendor-specific values for the lcp-echo-interval and lcp-echo-failure parameters. A later MSA release might make these parameters configurable, if needed by a future feature like support of an IP routing protocol over GCC. The implementation limits for the lcp-echo parameters are:

- Total echo failure detection time: 9 - 60 secs
- Echo interval: 3 - 10 secs
- Failure count: 3 - 20

The HDLC-like framing for PPP encapsulated packets is defined by RFC 1662 [17]. In the GCC protocol stack the bit-synchronous version is used, with the following configuration options:

- Length of the FCS: default value is 16 bit. For OpenROADM, support of the FCS-Alternatives configuration option of RFC 1570 [19] is not required
- Support of an Async Control Character Map (ACCM) is not required. Note: ACCM is only relevant for asynchronous or octet-synchronous encapsulation, see RFC 1662 section 7.1 “other types of links”.

To support unique IPv4 and IPv6 addresses for nodes having multiple interfaces, the OpenROAM Yang model supports the “softwareLoopback” interface type. This loopback interface has no supporting port or other supporting interface and it allows the controller to provision a single IPv4 and/or single IPv6 address to an OpenROADM device. All GCC interfaces of a node are “unnumbered”, which means they don’t have an assigned IP address. Instead, they “borrow” the IP address (IPv4 and/or IPv6) from the loopback interface. (Note: in the IPv6 case, an unnumbered interface still has its own link-local address)

OpenROADM supports the assignment of the same IPv4 and/or IPv6 address to the “softwareLoopback” and “OAMP” interfaces as long as no IP routing protocol is enabled. This is the case for the Remote Transponder use case and it is therefore possible to operate the Gateway Network Elements (GNE / CO transponders) with a single IPv4 and/or IPv6 address.

The IPv4 packet encapsulation over PPP is defined by the IPCP protocol of RFC 1332 [15]. Support of the updates in RFC 3241 [20] (“Robust Header Compression”) is not required.

The configuration options of the IPCP protocol are handled as follows:

- Options 1 and 2 (IP-Addresses and IP-Compression-Protocol) are not supported
- Option 3 (IP-Address) is used to convey the local IPv4 address borrowed from the loopback interface to the remote side. The remote side then has the option to use this IP address to automatically set up routing entries. As this is an optional behavior, the controller will in any case also set up the required host and default routes.
- If a peer does not provide the IP-Address option, it is requested by NAKing the option, see section 3.3 of RFC 1332.
- If a peer (or both) does not provide its IPv4 address to the other side, or if local and remote IPv4 addresses conflict (duplicate addresses), the IPv4 link stays down.
- After the IPv4-over-PPP link has been set up successfully, the (unnumbered) GCC interface is set active using the IPv4 address borrowed from the loopback interface.

The IPv6 packet encapsulation over PPP is defined by the IP6CP protocol of RFC 2472 [21].

Note: RFC 2472 has been obsoleted by RFC 5072 [16] with further updates by RFC 5172 [22] and RFC 8064 [23]. As this release of OpenROADM does not implement auto-configuration of global IPv6 addresses, these updates are not relevant.

The configuration options of the IP6CP protocol are handled as follows:

- Support for *interface-identifier* is mandatory as described in section 4.1. The negotiated interface-identifiers provide the local and remote IPv6 link-local addresses of the PPP link.
- Support for the IPv6-Compression-Protocol is not required
- The interface identifiers needed by the IP6CP protocol are not provisioned but calculated internally by a device. The RFC proposes the use of (possibly randomized) unique identifiers, like a MAC address present in the device inventory.

Note: IP6CP only negotiates the IPv6 link-local addresses of the PPP link and ensures uniqueness on the link. The global IPv6 addresses are provisioned on the loopback interfaces and are then borrowed by the unnumbered GCC interfaces. IPv6 address and routing setup for global addresses is fully done by the controller.

In general, PPP link setup and shutdown is done autonomously by the OpenROADM device. When LCP or IPCP or IP6CP link setup fails, or an existing link gets terminated, the device will restart link setup without controller intervention. After all required IP, PPP and GCC addresses and parameters have been provisioned by the controller, the link will autonomously transition between up and down states.

Specifically this means:

- If the loopback interface does not exist (has not yet been created by the controller) or has no IP address assigned to it, GCC link establishment will stop at the LCP layer, i. e. there will be no network layer setup using IPCP or IP6CP.
- Precondition for IPCP link setup is an assigned IPv4 address on the loopback interface (and completion of the underlying LCP layer)
- Precondition for IP6CP link setup is an assigned IPv6 address on the loopback interface (and completion of the underlying LCP layer)
- A GCC link can have an IPv4 network link, an IPv6 network link, or both
- Note: concurrent IPv4 and IPv6 support over a single GCC link is fully defined and possible, but implementation of this configuration is optional.

State changes of an GCC interface are reported using the *linkDown* alarm. A provisioned GCC interface raises this alarm when one or both of the following conditions are true:

- (ipv4/enabled == TRUE) and (an IPv4 address is provisioned or borrowed) and (IPCP NCP is in Closed state)
- (ipv6/enabled == TRUE) and (an IPv6 address is provisioned or borrowed) and (IP6CP NCP is in Closed state)

5.4.3 IPv4 and IPv6 Modelling

Yang modelling of the IP interfaces follows RFC 7277 [24] and interfaces with IPv4 and/or IPv6 support have additional 'ipv4', 'ipv4-state', 'ipv6' and 'ipv6-state' sub-containers. In MSA 12, the following interfaces have IP support:

- physical Ethernet ports
- software loopback
- GCC interfaces

RFC 7277 is only partially implemented by the MSA 12 OpenROADM Yang model and modelling of the following parts will be covered by a later release. Implementation of these (un-modelled) aspects follows the default requirements of the corresponding IPv4 and IPv6 RFCs:

- neighbor configuration and state
- duplicate address detection
- IPv6 auto-configuration

OpenROADM provides some specific additions to the IP Yang model not covered by RFC 7277:

- DHCP client control (default for the OAMP Ethernet interface is 'enabled', 'disabled' for all others)
- IP address source definition for unnumbered interfaces
- ProxyARP and ProxyNDP control

The IPv4 and IPv6 interface sub-containers are defined as follows:

```
+--rw if:interface* [name]
   ...
   +--rw ipv4!
   |  +--rw enabled?            boolean
   |  +--rw forwarding?         boolean
   |  +--rw mtu?                uint16
   |  +--rw dhcpc-enabled?      boolean
   |  +--rw (address-type)?
   |  |  +--:(numbered-address)
   |  |  |  +--rw address* [ip]
   |  |  |     +--rw ip               inet:ipv4-address-no-zone
   |  |  |     +--rw (subnet)         <details omitted>
   |  |  +--:(unnumbered-address)
   |  |     +--rw address-src?        -> /org-openroadm-device/interface/name
   |  +--rw proxy-arp!
   |     +--rw enabled?         boolean
   |     +--rw proxy-subnet* [ipv4-subnet] {ipv4-proxy-arp-subnet}?
   |        +--rw ipv4-subnet    inet:ipv4-prefix
   +--ro ipv4-state!
   |  +--ro forwarding?         boolean
   |  +--ro mtu?                uint16
   |  +--ro address* [ip]
   |     +--ro ip               inet:ipv4-address-no-zone
   |     +--ro (subnet)?        <details omitted>
   |     +--ro origin?          ip-address-origin (other, static, dhcp,
   |                                               link-layer, random)
   +--rw ipv6!
   |  +--rw enabled?            boolean
   |  +--rw forwarding?         boolean
   |  +--rw mtu?                uint32
   |  +--rw dhcpc-enabled?      boolean
   |  +--rw (address-type)?
   |  |  +--:(numbered-address)
   |  |  |  +--rw address* [ip]
   |  |  |     +--rw ip               inet:ipv6-address-no-zone
   |  |  |     +--rw prefix-length    uint8
   |  |  +--:(unnumbered-address)
   |  |     +--rw address-src?        -> /org-openroadm-device/interface/name
   |  +--rw proxy-ndp!
   |     +--rw enabled?         boolean
   |     +--rw proxy-subnet* [ipv6-subnet] {ipv6-proxy-ndp-subnet}?
   |        +--rw ipv6-subnet    inet:ipv6-prefix
   +--ro ipv6-state!
      +--ro forwarding?         boolean
      +--ro mtu?                uint32
      +--ro address* [ip]
         +--ro ip               inet:ipv6-address-no-zone
         +--ro prefix-length    uint8
         +--ro origin?          ip-address-origin (other, static, dhcp,
                                                   link-layer, random)
         +--ro status?          enum (preferred, deprecated, invalid,
                   inaccessible, unknown, tentative, duplicate, optimistic)
```

Pre-MSA 12 OpenROADM Yang models supported IP address configuration via special leaves in the /device/info sub-tree. Starting with MSA 12.0, these leaves will no longer be supported:

- ipAddress
- prefix-length
- defaultGateway

- source
- current-ipAddress
- current-prefix-length
- current-defaultGateway

When a pre-MSA 12 device is SW-upgraded to MSA 12 (or later), the contents of the /device/info leaves will automatically be migrated into the ipv4 and ipv6 sub-containers of the OAMP Ethernet interface by the device.

5.4.4 Duplicate Address Detection

Duplicate address detection by an OpenROADM device is limited:

- IPv4 duplicate address detection is provided implicitly by the PPP IPCP protocol. By definition, an OpenROADM PPP node will reject Configure-Requests if a peer announces a (peer-) local address that is already assigned to the current node. (Note: this behavior can be enabled in standard PPP daemons by using/not using the local/remote-ip and accept-local/accept-remote options)
- No specific IPv6 duplicate address detection is defined for the MSA 12 release. This would require implementation of the NDP protocol across the GCC link. It is up to the upper layers of the OpenROADM system hierarchy (controller, planning tools, etc.) to ensure uniqueness of provisioned IPv6 addresses of devices.

5.4.5 Routing Configuration

No IP routing protocol is being used for the “Remote Transponder” use case. All IP routes are static and configured manually by the controller:

- default routes (IPv4 and IPv6) in GCC-connected NEs are manually created as part of the Light Touch Provisioning (LTP) step
- as an option, depending on network architecture, the default routes (IPv4 and IPv6) in the GNE are initially setup by DHCP (IPv4) and NDP (IPv6) as part of One Touch Provisioning (OTP). The controller may replace them by static default route(s) in a later step.
- host routes to individual NEs in the GNE are set up manually by the controller. Note: PPP implementations often have an option to automatically set up a host route to the IPv4 peer. This option is ignored as the controller will provision the host routes to GCC NEs in any case.

When the controller provisions entries in a device’s routing table, it will use the single routing-instance name “MCN” and the routing-protocol name will be “STATIC”.

5.4.6 ProxyARP and ProxyNDP

ProxyARP (RFC 1027 [25]) and ProxyNDP (RFC 4861 [26]) is used on the GNE to let all NE global addresses (IPv4 and IPv6) appear as being locally connected to the Ethernet on the OAMP port. The OpenROADM model supports two schemes for configuring ProxyARP or ProxyNDP:

- the ProxyARP or ProxyNDP implementation of the device may automatically proxy all hosts or subnets for which a suitable routing table entry exists (implicit configuration). In this case, no further provisioning (other than enabling the proxy function on an interface) is required from the controller.
- the ProxyARP or ProxyNDP implementations require an explicit provisioning of host or subnet addresses to be proxied. The device indicates this case by announcing support for the *ipv4-proxy-arp-subnet* or *ipv6-proxy-ndp-subnet* Yang features. When these Yang features are announced by a device, the controller will provision the global addresses of GCC-reachable nodes to be proxied by the GNE. The OpenROADM model allows the configuration of subnets (prefix length < 32 or < 128) or of hosts (prefix length $= 32$ or 128). However, support of subnet provisioning is optional for a device to support.

By default, ProxyARP and ProxyNDP are disabled on all interfaces. For a GNE, the controller will enable the proxy function(s) on the OAMP Ethernet interface only.

5.5 CRAFT TERMINAL ACCESS

Some operations, for example the *Light Touch Provisioning* procedure (see 6.4), need access to the OpenROADM device via a *Craft Terminal*. For OpenROADM, the only application needed to run on a craft terminal is a standard Web browser.

The craft terminal is connected to the OpenROADM device using a standard Ethernet cable, plugged into the “CIT” port of the device. Connectivity between craft terminal and device is established using the IPv4 protocol. The device has a fixed, vendor-specific IP address, which is known to the service technician (vendor documentation). The IP address of the craft

terminal itself may be variable and is established by one of the following two mechanisms. Both mechanisms are supported without further configuration by standard Windows installations:

- The OpenROADM device implements a DHCP server on the CIT port. This server provides a (possibly dynamic) IP address to the connected craft terminal from the IPv4 private address space of RFC 1918 [27], namely 10.0.0.0/8, 172.16.0.0/12 or 192.168.0.0/16.
- The device takes the CIT port address from the IPv4 link-local address space of RFC 3927 [28], i. e. 169.254.0.0/16. The craft terminal then automatically selects a suitable link-local address for its own use, using the algorithm defined in section 2 of the RFC.

Regardless of which of the two options an OpenROADM device implements, the service technician will just point the web browser on the craft laptop to the fixed, vendor-specific IP address to execute the desired operation. The URL to be used looks like: **https:**//<**vendor-specific-IP**>/<**operation-path**>

The vendor-specific IP address for craft access to an OpenROADM device must be the same for all devices of the same type. It should be the same for all devices of a vendor.

5.6 NETCONF PROTOCOL

The Open ROADM MSA device is managed using the NETCONF protocol (per RFC 6241 [11]) on TCP port 830. The default username and password for accessing a NETCONF network element is openroadm/openroadm. The Open ROADM MSA device advertises its capabilities in the NETCONF Hello message. The Hello message provides an indication of support for standard features defined in NETCONF RFCs as well as support for specific namespaces. Configuration data can be written directly to the running configuration datastore or written to the candidate configuration datastore with a subsequent commit to push the configuration to the running datastore. The specific mechanism supported by a device is advertised in the Hello message. A vendor can support writing to the running configuration only, writing to the candidate configuration only, or support writing to both the running and candidate configurations.

NETCONF requires the support for subtree filtering on NETCONF messages. Devices MUST also support XPATH filtering. An Open ROADM MSA device indicates its support for XPATH in the Hello message. Operations on the device should be idempotent when the NETCONF “merge” operation is used. If the data in the merge operation is the same as what already exists on the device (even if it specifies only a partial set of the total attributes on the device), the device will accept the command with no changes. No database change notification would be sent under this case.

5.6.1 Secure Shell (SSH) Support and Idle Timeouts

The NETCONF protocol runs over SSHv2 for security. It is recommended that implementations follow RFC 6242 [29] which is the latest specification of NETCONF over SSH. RFC 6242 [29] provides the procedure for interoperability with NETCONF implementations that support the older NETCONF over SSH described in RFC 4742 [30].

Note: In RFC 4742 [30], “]]>]]” was used as the NETCONF message delimiter. But an issue was identified where this character string could appear in the NETCONF body causing parsing issues. RFC 6242 [29] modifies the message format to support message chunks with explicit message size identifiers to overcome this issue.

If multiple NETCONF sessions are established to a network element, those sessions should be established over separate SSH tunnels.

The Open ROADM MSA device uses the password authentication method for SSH. Once authenticated, the controller will request to open a channel of type “*session*” and invoke the “*netconf*” subsystem.

Devices should support IDLE timeout to clean up inactive sessions. In such case, keep alive messages are required to maintain connectivity. For example, an Open ROADM controller may send periodic retrieves of the info container using a ping mechanism.

The NETCONF layer IDLE timeout is applied for Netconf sessions *without an active subscription*. Sessions with an active subscription are typically used for notifications only and might therefore not carry any traffic for a very long time. Idle supervision on Netconf layer therefore only makes sense if a session has no active subscription.

The Netconf IDLE supervision disconnects a session if there is no Netconf traffic in any direction for a certain period (and there is no subscription). This time-out period is specified by the *info/netconf/idle-timeout* parameter (see below) with a range of 0 . . . 1440 minutes and a default value of 60 minutes. The controller can set the *idle-timeout* to 0 to disable Netconf IDLE supervision. The device indicates support for Netconf IDLE supervision by announcing the *netconf-idle-supervision* feature at session setup.

An OpenROADM device may also indicate support for the SSH keep-alive mechanism (described in section 4.1 of RFC 8071 [31], item ’S7’, using `SSH_MSG_GLOBAL_REQUEST`) by announcing the *ssh-idle-supervision* feature at Netconf session setup. SSH keep-alive is configured by two parameters:

- *info/ssh/client-alive-interval* sets a timeout interval in seconds after which if no data has been received from the client, the server will send an `SSH_MSG_GLOBAL_REQUEST` message to request a response from the client. Accepted range is 0 . . . 3600 seconds with a default value of 120 secs.
- *info/ssh/client-alive-max-count* specifies the maximum number of missing responses to transmitted `SSH_MSG_GLOBAL_REQUEST` messages. If this count is exceeded, the SSH (NETCONF) session will be closed by the device. Accepted range is 0 . . . 20 with a default value of 3.

Setting one of the two parameters to 0 disables SSH keep-alive and idle supervision.

NETCONF and SSH idle supervision is covered in the Yang model as follows:

```
module: org-openroadm-device
  +--rw org-openroadm-device
     +--rw info
     ....
     |  +--rw netconf
     |  |  +--rw idle-timeout?   uint32 {netconf-idle-supervision}?
     |  +--rw ssh
     |  |  +--rw client-alive-interval?    uint32 {ssh-idle-supervision}?
     |  |  +--rw client-alive-max-count?   uint32 {ssh-idle-supervision}?
     ....
```

It is recommended to support both, NETCONF and SSH idle supervision. In addition, or as a (partial) replacement for SSH supervision, an OpenROADM device may enable TCP keep-alive (per RFC 1122 [32]). TCP keep-alive is not announced to the controller and is not configurable.

5.6.2 Notification Support

Open ROADM MSA devices must support NETCONF event notifications per RFC 5277 [33]. Implementations may choose to output the Open ROADM MSA notifications on the optional OPENROADM stream. If this stream is used, only Open ROADM MSA notifications should be exposed on this stream. If the OPENROADM stream is not supported, then implementations must output the Open ROADM MSA notifications on the default NETCONF stream. Note: there is no guarantee only Open ROADM MSA notifications would appear on the default NETCONF stream. The controller should retrieve the list of streams supported by an Open ROADM MSA device. If the device supports the OPENROADM stream, the controller should subscribe to that stream. Otherwise it should subscribe to the NETCONF (default) stream. Open ROADM MSA devices must support the interleave option that allows both notifications and RPCs in the same session.

5.6.2.1 Change Notification Support

The Open ROADM MSA YANG model supports two types of change notifications (*change-notification*) to identify changes on the device: configuration change notifications and operational change notifications. Change notifications identify the time the change took place, who initiated the change (either the server or the specific user including session information), the datastore that was affected (e.g. running configuration), the operation that was done to the entity, and the target of the change. The target points to the element in the data model that has changed.

The details of the change itself are not provided by the *change-notification*. For instance, if an attribute has changed, the notification does not indicate the new value of the attribute. The controller needs to retrieve the target to see the details of the change.

Change notifications are to be supported for any and all configuration data. Any time a configuration entity is created, modified or deleted, there should be a *change-notification* for that entity or a parent entity higher in the YANG data tree.

Only limited sets of operational data will result in change notifications. For the controller to remain in sync with the device, there are a few critical notifications of operational data required:

- Equipment plug in/out events (to understand when new equipment is added to the system, or removed from the system). This should result in a change-notification indicating that the physical inventory data has been changed (*vendor, model, serial-id, type, product-code, manufacture-date, clei,* and *hardware-version*)
- *operational-state* attribute change (to know when the operational status of an entity has changed)
- Capabilities change (to know when a capability announcement has been updated). . .

  1. As a general rule for capabilities, if the change is attached to another entity and once created with the entity it never changed, then a separate notification is not required. The *change-notification* for the parent (or higher) entity would cover the *change-notification* for the child capability

  - For example: *pluggable-optics-holder-capability* (associated with the parent *cp-slot*) *supported-interface-capability* (associated with the parent port or *circuit-pack*) port power capabilities (associated with the parent port or *circuit-pack*) *port-capabilities* (associated with the parent port or *circuit-pack*)

2. For capabilities, the key ones that require separate notifications would be ones not attached to another entity. This includes:
   - *port-group-restrictions*
   - *connection-map*
   - *odu-switching-pools*

Other critical operational data have specific notifications defined:

- LLDP neighbor *change-notification* (for transport topology) via *lldp-nbr-info-change*
- RSTP state *change-notification* (for dataplane topology) via *rstp-topology-change* and *rstp-new-root*

It is recommended in the event of a large number of changes, the controller implement a hold-off and a consolidation in order to reduce the number of queries made to the device. MSA members may discuss future enhancements to include the changed data in the notification itself to avoid subsequent queries to the device.

An example operational change notification is as follows (actual notification would be in XML form):

5.6.2.2 Replay Support

Open ROADM MSA devices must support the ability to request previously logged notifications via a NETCONF replay operation (per RFC 5277 [33]). The NETCONF replay operation is triggered by the inclusion of the optional *startTime* parameter in an event subscription operation (*create-subscription*). Starting from the specified *startTime* (must be specified in the past), previously logged event notifications are sent/re-sent asynchronously in the same way as normal event notifications.

Inclusion of the optional *stopTime* parameter can be used to specify when to stop sending event notifications (must be later than the specified *startTime*). If the *stopTime* parameter is not present, event notifications will continue until the subscription is terminated. If the subscription includes a *stopTime*, the session becomes a normal NETCONF session again after the stopTime has been reached. A *replayComplete* notification is sent once all replay event notifications have been sent. Note: the *stopTime* parameter can only be included in the event notification subscription if the startTime is also specified.

The actual number of stored notifications available for retrieval is implementation specific. As a result, the ability to query the *replayLogCreationTime* (timestamp of earliest available logged notification) and the *replayLogAgedTime* (timestamp of last notification aged out of the log) should be supported in order to determine the availability of event notifications for replay. Note: the *startTime* and *stopTime* are associated with the time an event was generated by the device.

5.6.3 Monitoring Support

Open ROADM MSA devices must support NETCONF monitoring per RFC 6022 [34], including support for retrieving the NETCONF capabilities, datastores, schema and session information. Implementations may support NETCONF monitoring statistics.

5.7 SYSLOG PROTOCOL

The Syslog protocol (RFC 5424 [35]) is a standard method to log device event notifications. An Open ROADM MSA device supports logging of all device event notifications (i.e. no filtering) to a syslog file on its local filesystem and it may support (optional) remote streaming of event notifications to external systems via YANG model provisioning. Remote streaming of event notifications provides the ability to retain a larger volume of logs on a persistent server located remotely.

The Open ROADM MSA Device model is derived from an IETF Syslog YANG Model draft [36] with the addition of the *local-syslog-filename* attribute to identify the filename of the locally stored syslog file:

The Open ROADM MSA device must provide the syslog filename (*local-syslog-filename*) stored on the device's local filesystem so the controller can upload it to an external system via the transfer RPC (see Section 5.2.3 for SFTP details).

The *log-actions* tree (remote container) provides the provisioning data for syslog streaming. The transport choice identifies the destination of the stream (multiple *destinations* are supported by the model) and whether to stream the event notifications via TCP or UDP. The Open ROADM MSA model provides the ability to filter data (*log-selector*), where, the *selector-facility* attribute describes the option to specify no facilities (*no-log-facilities*) or specific facilities (*log-facility*). The *log-facility* attribute provides a list of syslog facilities and severities to stream (can be set to all for all facilities).

```
notification {
    eventTime 2017-10-30T20:35:35.257576+00:00
    change-notification {
      change-time 2017-10-30T20:35:35.219081+00:00
      changed-by {
      server
      }
      datastore running
edit {
target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0/P
↪  13']/vendor
operation
replace
}
edit {
target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0/P
↪  13']/model
operation
replace
}
edit {
target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0/P
↪  13']/serial-id
operation replace
}
edit {
target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0/P
↪  13']/type
operation replace
}
edit {
target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0/P
↪  13']/product-code
operation replace
}
edit
{ target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0
↪  /P13']/manufacture-date
operation replace
}
edit
{ target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/0
↪  /P13']/clei
 operation replace
 }
 edit
 { target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-name='200/
  ↪  0/P13']/hardware-version operation
  ↪  replace
 }
 }
 }
```

YANG sub-tree 57: Example operational change notification

```
+--ro local-syslog-filename
+--rw log-actions
+--rw remote
+--rw destination* [name]
+--rw name
+--rw (transport)
| +--:(tcp)
| | +--rw tcp
| | +--rw address? inet:host
| | +--rw port? inet:port-number
| +--:(udp)
| +--rw udp
| +--rw address? inet:host
| +--rw port? inet:port-number
+--rw log-selector
+--rw (selector-facility)
+--:(no-log-facility)
| +--rw no-facilities?
+--:(log-facility)
+--rw log-facility* [facility]
+--rw facility union
+--rw severity
+--rw severity-operator?
```

YANG sub-tree 58: Syslog

A provisioning example to stream all logged event notifications to a remote syslog server using UDP is shown in 59 (attribute values <name>, <address>, <port>, <severity> can be changed by the operator):

5.8 TELEMETRY STREAMING SERVICE

Streaming telemetry is a push-based streaming mechanism that removes the inefficiencies associated with polling (e.g. pull-based mechanisms like SNMP). The data is streamed automatically and continuously in near real-time from network devices to management systems without the need for polling. Streaming telemetry provides network state indicators, network statistics, and critical infrastructure information that is useful for traffic optimization and reducing troubleshooting time.

The Open ROADM MSA telemetry data is described as a model-driven using YANG with the data to be streamed defined through subscription. The process for streaming telemetry data uses three components:

- Destination specifies the destination(s) to collect the streamed data
- Sensor path specifies the YANG path from which data has to be streamed
- Subscription binds one or more sensor paths to destinations and specifies the criteria to stream data.
- Transport and encoding represents the delivery mechanism of the telemetry data. For the Open ROADM MSA, the YANG data is encoded in JSON at a minimum and streamed over gRPC

Two modes are used to initiate a telemetry session:

- Dial-out mode refers to sessions that originate from the network element to the destination(s) based on the specified subscription.
- Dial-in mode refers to sessions originated by the destination to the network element and subscribes the data to be streamed

For both dial-out and dial-in, a gNMI notification is used to stream the telemetry data to the destinations. In the case of dial-out mode, an additional header is pre-pended to the gNMI notification message to identify the source.

5.8.1 Dial-Out Mode

By default, dial-out mode uses JSON (per RFC7159 [37]) to encode the YANG data and gNMI notification messages to stream the telemetry data (refer to Figure 72).

```
<edit-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
↪  xmlns:org-openroadm-syslog="http://org/openroadm/syslog">
    <target>
        <candidate/>
    </target>
    <config>
        <syslog xmlns="http://org/openroadm/syslog">
            <log-actions>
                <remote>
                    <destination>
                        <name>syslog-server</name>
                        <udp>
                            <address>167.254.219.85</address>
                            <port>512</port>
                        </udp>
                        <log-selector>
                            <log-facility>
                                <facility>all</facility>
                                <severity>all</severity>
                            </log-facility>
                        </log-selector>
                    </destination>
                </remote>
            </log-actions>
        </syslog>
    </config>
</edit-config>
```

YANG sub-tree 59: Provisioning example to stream all logged event notifications to a remote *syslog* server using UDP

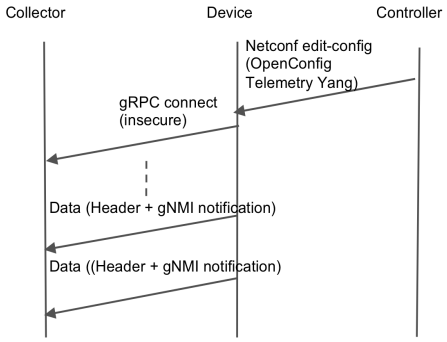

Figure 72: Telemetry Dial-Out Model

```
+--rw telemetry-system
+--rw destination-groups
| +--rw destination-group*
| +--rw group-id
| +--rw config
| | +--rw group-id?
| +--ro state
| | +--ro group-id?
| +--rw destinations
| +--rw destination*
| +--rw destination-address
| +--rw destination-port
| +--rw config
| | +--rw destination-address?
| | +--rw destination-port?
| +--ro state
| +--ro destination-address?
| +--ro destination-port?
```

YANG sub-tree 60: Destination group

```
+--rw telemetry-system
+--rw sensor-groups
| +--rw sensor-group*
| +--rw sensor-group-id
| +--rw config
| | +--rw sensor-group-id?
| +--ro state
| | +--ro sensor-group-id?
| +--rw sensor-paths
| +--rw sensor-path*
| +--rw path
| +--rw config
| | +--rw path?
| | +--rw exclude-filter?
| +--ro state
| +--ro path?
| +--ro exclude-filter?
```

YANG sub-tree 61: Sensor group

As shown in Figure 72, the Open ROADM MSA specifies NETCONF as the management protocol and the OpenConfig Telemetry YANG model.

The process to configure dial-out mode involves the following:

- **Creating a Destination Group**: The destination group specifies the destination address, port, encoding and transport method the network element uses to transmit the telemetry data.

- **Creating a Sensor Group**: The sensor group specifies a list of YANG models to be streamed.

- **Creating a Subscription**: The subscription associates the destination group with a sensor group and sets the streaming method. A source interface is identified that specifies the interface that will be used to establish the session and stream the telemetry data to the identified destination. The streaming method specified by the Open ROADM MSA uses an insecure gRPC connection with the OpenConfig Telemetry protocol field set to `STREAM_GRPC`. Per the gNMI specification [38], the data format must support JSON encoding (per RFC7159 [37]) at a minimum. As a result, the OpenConfig Telemetry sets the encoding field to `ENC_JSON_IETF`. An additional header is prepended to the gNMI notification messages to identify the source (*local-source-address*).

```
+--rw telemetry-system
+--rw subscriptions
+--rw persistent
| +--rw subscription*
| +--rw subscription-name
| +--rw config
| | +--rw subscription-name?
| | +--rw local-source-address?
| | +--rw originated-qos-marking?
| | +--rw protocol?
| | +--rw encoding?
| +--ro state
| | +--ro subscription-name?
| | +--ro subscription-id?
| | +--ro local-source-address?
| | +--ro originated-qos-marking?
| | +--ro protocol?
| | +--ro encoding?
| +--rw sensor-profiles
| | +--rw sensor-profile*
| | +--rw sensor-group -> ..
| | +--rw config
| | | +--rw sensor-group?
| | | +--rw sample-interval?
| | | +--rw heartbeat-interval?
| | | +--rw suppress-redundant?
| | +--ro state
| | +--ro sensor-group?
| | +--ro sample-interval?
| | +--ro heartbeat-interval?
| | +--ro suppress-redundant?
| +--rw destination-groups
| +--rw destination-group*
| +--rw group-id
| +--rw config
| | +--rw group-id?
| +--ro state
| +--ro group-id?
```

YANG sub-tree 62: Subscription

5.8.2 Dial-In Mode

In dial-in mode, the collector requests the subscription to the sensor paths when it establishes a connection with the network element (refer to Figure 73: Telemetry Dial-In Mode)

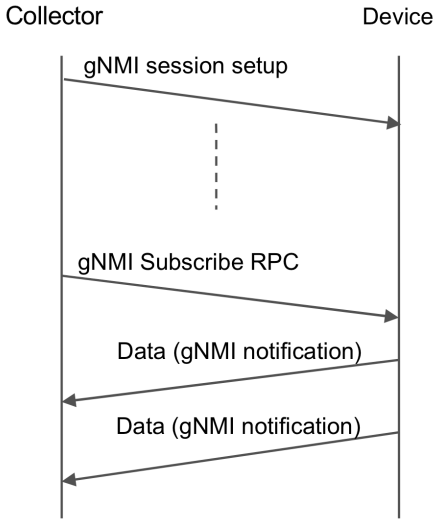

Figure 73: Telemetry Dial-In Mode

Unlike dial-out mode, the Open ROADM MSA dial-in mode specifies gNMI (v0.6.0) as the management protocol with a secure gNMI subscription gRPC using TLS [38] to establish the connection (subscription). As with dial-out mode, the data encoding method is JSON (per RFC7159 [37]).

As required by the gNMI specification, TLS uses v1.2 (per RFC 5246 [39]) configured as follows for Open ROADM:

- Key type: RSA
- Certificate type: x509
- Minimum key size: 1024 bits

Note: the gNMI specification allows for TLS v1.3, providing performance, privacy, and security improvements over v1.2, but it is not backwards compatible with v1.2 implementations.

Once a subscription is established, the dial-in mode uses gNMI notification messages to stream the telemetry data. The following (Example 63) is a sample subscription using gNMI for dial-in mode:

5.9 DEVICE FACEPLATE LABELS

To help the technician visually identify a physical unit on the device, the Open ROADM MSA device model includes labeling attributes intended to represent the most prominent label silkscreen on the physical unit that can be used to identify it. The *faceplate-label* attribute is defined for the *shelf, circuit-pack, and circuit-pack/ports.* A device should also define a label for the *shelf/slots* and *circuit-pack/cp-slots* representing the physical silkscreen labels on the unit that identify the shelf and circuit-pack slots.

The *shelf* and *circuit-pack faceplate-label* is based on what is physically plugged in (retrieved inventory data for *shelf-type, circuit-pack-type, circuit-pack-product-code*). Table 18 defines what the *faceplate-label* should be set to under certain conditions where the *faceplate-label* cannot be determined.

Condition	faceplate-label
not physically	*installed*
not-installed physically installed, but the physical inventory cannot be retrieved (e.g. unit in fault state)	*unknown*
unit doesn't have a faceplate label (e.g. limited real estate, such as fan units)	*none*
pluggable optic	*pluggable*

Table 18: Shelf and Circuit-pack Faceplate Label

```
service gNMI {
rpc Subscribe(stream SubscribeRequest) returns (stream SubscribeResponse);
}
message Notification {
int64 timestamp = 1;
Path prefix = 2;
repeated Update update = 4;
repeated Path delete = 5;
}
message Update {
Path path = 1;
TypedValue val = 3;
}
message TypedValue {
oneof value {
bytes json_val = 10;
}
}
message Path {
repeated PathElem elem = 3;
string target = 4;
}
message PathElem {
string name = 1;
map<string, string> key = 2;
}
enum Encoding {
JSON = 0;
}
message SubscribeRequest {
oneof request {
SubscriptionList subscribe = 1;
}
}
message SubscribeResponse {
oneof response {
Notification update = 1;
bool sync_response = 3;
}
}
message SubscriptionList {
Path prefix = 1;
repeated Subscription subscription = 2;
enum Mode {
STREAM = 0;
}
Mode mode = 5;
Encoding encoding = 8;
}
message Subscription {
Path path = 1;
SubscriptionMode mode = 2;
uint64 sample_interval = 3;
}
enum SubscriptionMode {
SAMPLE = 2;
}
```

YANG sub-tree 63: Sample subscription using gNMI for dial-in mode

5.10 DEVICE STATE DEFINITIONS AND TRANSITIONS

Four types of states are defined for OpenROADM device resources: *administrative-state*, *operational-state*, *equipment-state*, and *lifecycle-state*.

administrative-state is a read-write attribute with three enumerators (*inService*, *outOfService*, and *maintenance*) that is used to support maintenance operations and control alarm notifications against a device resource. More specifically,

1. A resource in "*inService*" administrative-state will discover and forward current alarm notifications on the resource.
2. A resource in "*outOfService*" administrative-state will suppress alarm notifications on the resource. Alarm clearing notifications should be sent for current alarms posted against the resource in the "*outOfService*" administrative-state to avoid staled alarm records in the OSS.
3. A resource is required to be in "*maintenance*" administrative-state for the device to accept maintenance operations (i.e. loopback, test signal) against the resource.
4. A device implementation shall support direct transition from one *administrative-state* to another.
5. A device implementation should not reject any edit or delete operation based on *administrative-state* value of itself or its parent or children. Controller should assume the responsibility following hierarchical rules to maintain system data structure integrity.
6. The administrative state is for alarm suppression only. There should be no impact to the operational behavior of the entity (such as turn on or off lasers) other than alarm suppression due to the setting of the administrative state. Furthermore, operational-state change notification should not be subject to suppression by the administrative state. If there is a need for the device to prevent unintended data plane modifications (i.e. *edit-config* should not disrupt data plane), a future MSA model will have to identify data plane impacting attributes from those having no impact to data plane.

operational-state is a read-only attribute with three enumerators defined (*inService*, *outOfService*, and *degraded*) that reflects the functional status of a resource. A value of "*inService*" indicates the resource is fully functional, while a value of "*outOfService*" implies the resource fails to perform some or all of its functions. In general, it's expected a resource in an *outOfService* operational-state has outstanding alarm/event notification(s) that directly relates to this status. Future MSA release will explore mechanism to make this causal-effect relationship more explicit. The "*degraded*" operational-state value indicates a resource detected some anomaly with some of its functionality, yet it's not severe enough to trigger "*outOfService*" status. Some maintenance or path selection applications may consider the "*degraded*" state value too ambiguous for decision making, so any implementation of this operational state value should have sufficient documentation and justification for downstream users.

equipment-state is a read-write attribute (e.g. *not-reserved-planned*, *reserved-for-facility-planned*, *reserved-for-facility-unvalidated*, *reserved-for-facility-available*, *reserved-for-maintenance-planned*, etc.) that serves as a pass-through field for life-cycle management applications. The semantics of this state and the application of it are outside the scope of the device model itself. The equipment state model is managed by the controller and stored on the device. The actual equipment-state may or may not drive behavior on the Open ROADM MSA device. *lifecycle-state* is a read-write attribute that is used to support the deployment state of devices, shelves, circuit-packs, ports, interfaces, protocols, physical links, degrees, and SRGs (e.g. *deployed, planned, maintenance, deploying, proposed*, etc.).

6 DEVICE DISCOVERY, COMMISSIONING, AND AUGMENTATION

6.1 DEVICE DISCOVERY AND COMMISSIONING

The Open ROADM MSA device is provisioned through a one-touch procedure to simplify the device commissioning step (refer to Figure 74: One-Touch Provisioning Device Commissioning Procedure). The one-touch procedure allows for software and firmware upgrades of the device based on a planned device template and the current software/firmware and Open ROADM MSA version of the device.

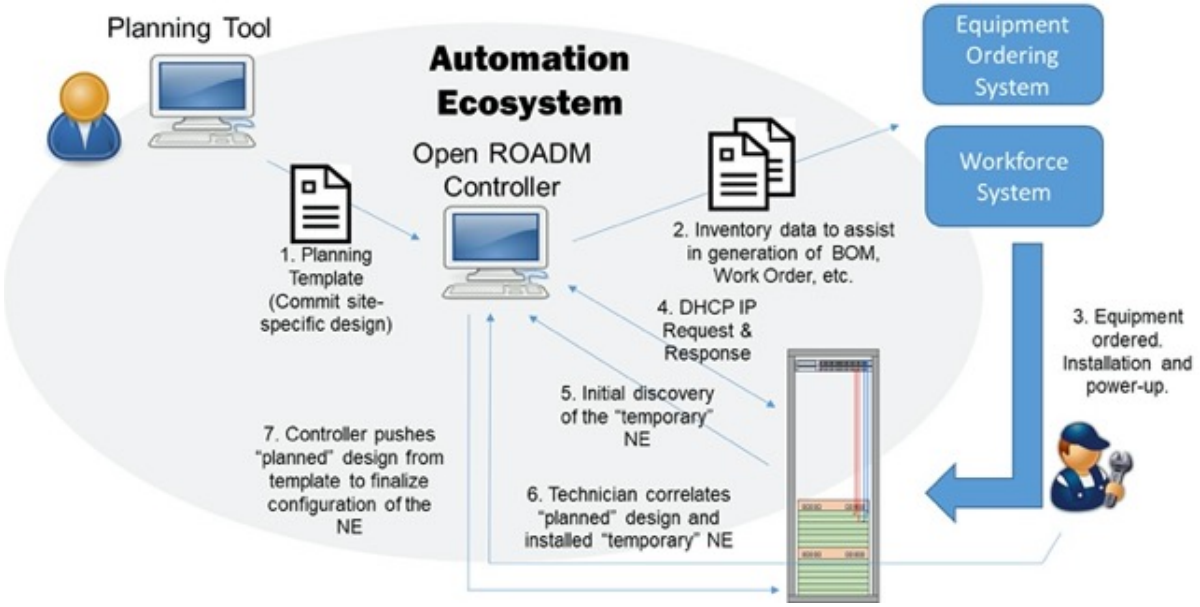

Figure 74: One-Touch Provisioning Device Commissioning Procedure

This section describes an example set of operations by an Open ROADM controller to commission a device. The MSA does not specify the operations required by an Open ROADM controller; it is possible a controller could implement the commissioning steps differently than specified in this section.

The general Open ROADM MSA discovery and commissioning states of the device are shown in Figure 75: Device Discovery and Commissioning States.

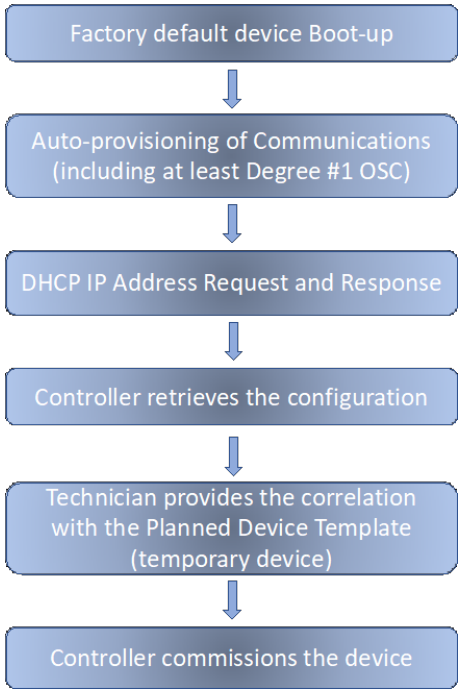

Figure 75: Device Discovery and Commissioning States

6.1.1 Planned Device Template (Step 1)

The planning template for the device to be commissioned is loaded (pushed) into the Open ROADM controller. Note: the specific method for loading the planning template (e.g. planning tool, manually configured, vendor provided, etc.) and the planning template itself are not standardized by the Open ROADM MSA.

However, the planning template must provide enough data (final *node-id*, permanent IP address (*ipAddress*), *shelves/circuit-packs/ports* attributes, etc.) to allow the controller to correctly commission the device (beyond the auto-provisioning behavior) using the Open ROADM MSA device model. One possible implementation of the planning template is a JSON file containing a subset of the Open ROADM MSA device model that would be needed to configure the device.

6.1.2 Physical Device Installation, System Initialization, and Auto-Provisioning (Steps 2 & 3)

The Open ROADM controller may generate inventory information (e.g. BOM, work order, etc.) that allows external ordering and work force systems to automate the ordering of equipment and installation. Once the equipment is ordered, a field technician installs (i.e. physical connectivity) and powers up the equipment.

Upon equipment system initialization, an Open ROADM MSA device will set the node ID to the default value of *'node-id=openroadm'* and set the user account to defaults *'name=openroadm'* and *'password=openroadm'*. Equipment should auto-provision to the point that communications can be established to the node, either through the external LAN port and/or the internal OSC. Both communication paths (external LAN and internal OSC) should be auto-provisioned, since the interface the device will be accessed on is not known by the device.

For equipment (*circuit-packs*) provisioned either automatically or manually, it is expected that the *ports* associated with the *circuit-packs* are auto-created by the device and retrievable via NETCONF.

Note: In general, it is recommended auto-provisioning of the device be disabled, with the exception of enabling the OSC for remote communications (minimum would be the OSC associated with Degree #1). This is to prevent accidental or incorrect auto-provisioning if the installer slots an equipment incorrectly. This would result in the creation of an entity associated with an incorrect slot and difficult to remedy; it is assumed the installer will not have access to the management interface on the Open ROADM MSA device.

6.1.3 Device Initialization (Step 4)

The Open ROADM MSA device initialization, auto-provisioning, and IP address request are triggered via the Dynamic Host Configuration Protocol (DHCP). The operator may choose to have the DHCP server coexist with their Open ROADM controller or have it located separately from the controller.

An Open ROADM MSA device initializes with a DHCP client enabled and requests both an IPv4 and IPv6 address as it is not known which protocol is in use in the service provider's network. Preference is given to IPv6. This mechanism obtains the initial IP address (temporary IP address) and connectivity to the device. The method for querying both IPv4 and IPv6 (parallel/serial, interleaved, timing, etc.) is device dependent.

Note: It is expected the operator will change the DHCP address to a permanent IP address (*ipAddress*) during node commissioning so the IP address (*ipAddress*) will be stable. Please refer to Section 6.1.6.2 for further details.

6.1.4 Controller IP Address Assignment Discovery (Step 5)

The controller discovers the new IP address assigned by the DHCP server. The mechanism for DHCP IP address assignment discovery is outside the scope of the Open ROADM MSA specification.

The controller cannot determine if a given DHCP IP address belongs to an Open ROADM MSA device or some other IP capable node that supports DHCP (e.g. laptop connected to the DCN network). As a result, the controller attempts to connect and login to the NE under the assumption it's an Open ROADM MSA device.

The controller attempts to connect to the IP address (*ipAddress*) with NETCONF over SSH (see Section 5.2.2 for SSH support details) using a default NETCONF port (port 830). The SSH authentication uses the Open ROADM MSA default username (*name*=openroadm) and password (*password*=openroadm). Once the Open ROADM MSA device is discovered (login is successful), the controller attempts to retrieve the *info* container data and obtain the device's *vendor, model*, and serial number (*serial-id*) attributes. The controller adds the discovered Open ROADM MSA device as a 'temporary' NE.

Note: the device commissioning process is terminated if the device is not an Open ROADM MSA device.

6.1.5 Correlation of the Planned Template and the Temporary Discovered NE (Step 6)

At this point in the device commissioning process, the controller does not know the correlation between the planned template data for a node and a newly discovered temporary NE in its inventory. The technician connects to the controller to determine and provide this correlation.

The technician first needs to identify and select the planned template data associated with the installation. The technician enters the CLLI (site location identifier) for the installation into the controller and retrieves all planned template data associated with that site. The technician selects the correct template based on the *vendor, model and node-type* that was installed.

The next step in the correlation process is for the technician to identify the corresponding temporary discovered NE. This is accomplished by the technician entering the device's physical serial number into the controller (Note: the vendor should have a way to easily identify the serial number that represents the node). The controller then provides the technician with the temporary NE that matches the serial number. It is expected that only one temporary NE would be returned. However, if the controller responds with multiple discovered temporary NEs that match the supplied serial number, the technician would need to select one based on the *vendor, model, and node-type* of the installed node.

The technician would then "commit" the selections and the controller would associate the selected planned template data with the temporary discovered NE that matches the installed node [One Touch]. Note: The technician may need to work with the equipment vendor to determine how to identify the physical serial number of the installed device that will be used for this correlation.

6.1.6 Device Commissioning (Step 7)

Once the correlation has been made between the planned template data and a temporary discovered NE, the controller begins to provision the device. The controller pushes the planned template configuration to the device using the NETCONF *edit-config* RPC (with the merge operation) and rediscovers the device as a permanent device with a permanent IP address (*info/ipAddress*). There may be processing involved in the controller, which takes both the planned template data and the current state of the device as input and determines the set of operations that need to be performed on the device.

Note: the use of the merge operation allows the provisioning to succeed even if the entity (e.g., *shelf, circuit-pack, port*, etc.) was auto-provisioned due to the device's idempotent behavior.

The controller commissions the device in the following order based on the Open ROADM MSA device YANG models:

- Initial *info* container validation
- Provision *info* container, including the permanent *node-id* and IP address (*ipAddress*)
- Provision *shelves*
- Provision *circuit-packs* and associated *ports*
- Provision *degree*
- Provision SRG (*shared-risk-group*)
- Provision ILA (*line-amplifier*)
- Provision *xponder*
- Provision Physical Links *(physical-link*)
- Provision interfaces (OAMP and OSC Ethernet, OTS, OMS)
- Provision protocols (LLDP and RSTP)
- Provision user accounts (*name, password, group*)
- Set date and time (*set-current-datetime* RPC)

Special handling is done for some of the device commissioning steps (refer to the following sub-sections for details).

6.1.6.1 Initial Validation

Before configuring the device, the controller validates the device's *vendor, model*, and *node-type* information in the info container against the planned template data values to ensure the correct device will be provisioned. If the *vendor, model*, and *node-type* do not match, then an error is raised and the device commissioning process is terminated.

The controller also validates the planned template *openroadm-version* with the device *openroadm-version* and decides on the device commissioning steps (note: 2.2 and 2.2.1 are considered the same openroadm-version). Software and/or firmware upgrade scripts maybe executed to upgrade the device from its current version to the desired version specified in the planned device template. Software download details can be found in Appendix A and firmware download details can be found in Section 7.1.3. If for any reason a software or firmware upgrade fails, the device commissioning will be terminated.

Note: currently in-service software downgrades are **not** supported by Open ROADM MSA devices.

6.1.6.2 Node Identifier and IP Address Provisioning

Once the device is validated, the controller provisions the *info* container, including the permanent *node-id*, permanent IP address (*ipAddress*), *prefix-length*, and optional gateway (*defaultGateway*). The controller then deletes the temporary discovered device and terminates the session to the temporary IP address and triggers a temporary device discovery process using the permanent ipAddress.

The syntax for the *node-id* is as follows:

- length “7..63”
- pattern “([a-zA-Z][a-zA-Z0-9-]5,61[a-zA-Z0-9])”

When the permanent *ipAddress* is provisioned (*edit-config info/ipAddress*), the device should disable DHCP and release the temporary IP address. The communication session between the controller and device will most likely be terminated as the IP end point is no longer valid on the device. The controller will re-establish the NETCONF session using the new permanent IP address (*ipAddress*) and initiate a warm restart of the device (restart RPC). Refer to Section 9.1.1 for system restart model details.

6.1.6.3 Shelf Provisioning

The controller provisions the shelves container (*edit-config/shelves*) and waits a certain amount of time (implementation specific) before checking for any alarms (e.g. *equipmentMismatch, equipmentFault, powerProblemA, powerProblemB, fanCoolingFail*, etc.). If an alarm exists (controller looks for a subset of alarms) after the timeout, the device commissioning process is terminated and a notification is sent to the user.

Note: The device may auto-provision some equipment and interfaces (e.g., *shelves, circuit-packs*, and *interfaces* to support in-band communication for ZTP). In this case, the controller may send provisioning of the same equipment and/or interfaces. It is expected that this provisioning be accepted by the device in this case. (Note: it is expected that typically the provisioning provided by the vendor template in this case matches the auto-provisioning behavior in which case idempotent behavior is expected by the device without an error, or if not, that the provisioning changes be accepted by the device.)

6.1.6.4 Circuit Pack Provisioning

The controller provisions the *circuit-packs* container and all associated ports (*edit-config/circuit-packs*) and waits a certain amount of time (implementation specific) before checking for any alarms (e.g. *equipmentMismatch, equipmentFault, powerProblemA, powerProblemB, fanCoolingFail*, etc.). If an alarm exists (controller looks for a subset of alarms) after the timeout, the device commissioning process is terminated and a notification is sent to the user.

6.1.6.5 Degree, SRG, ILA, Xponder, Physical Links, Interfaces, and Protocols Provisioning

The controller provisions the degree container (*edit-config/degree*), SRG container (*edit-config/shared-risk-group*), ila container (*edit-config/ila*), xponder container (*edit-config/xponder*), *physical-links* container (*edit-config/ physical-links*), interfaces container (*edit-config/interfaces*) (including management Ethernet (OAMP, OSC) interfaces, OTS interface, OMS interface), and the protocols container (*edit-config/protocols*) (LLDP, RSTP).

6.1.6.6 User Account Provisioning

The controller provisions the user accounts to be established on the device, including the user account’s name, password, and group. Currently only one group is defined called ”sudo” that has full access to the device. The assumption under the Open ROADM MSA is that user security roles are managed at the controller, not on the device itself.

The *username* (name) is a string between 3-32 characters. The first character must be a lowercase letter and the remaining characters can be a lowercase letter or a number. The username (name) may be treated as case insensitive on some devices. The *password* is a string between 8-128 characters. Allowed characters in the password field include lowercase and uppercase letters, numbers and the special characters: ! $ % ^( ) _ + ~{ } [ ]. The following characters are not allowed in the password string: \| ? # @ &. Note: the chg-password RPC is used to change an existing password.

Once the new user account is provisioned, the controller logs out of the device (default ‘*openroadm*’ account) and re-logs in with the newly provisioned user account and deletes the default ‘*openroadm*’ account. This procedure must be followed to delete an account, as you cannot delete the account being used for the currently active session.

6.1.6.7 Date and Time Setting

The controller will synchronize the date and time with the device by setting the controller date and time directly on the device via a ‘*set-current-datetime* RPC. Note: the Open ROADM MSA does not support the Network Time Protocol (NTP).

6.1.6.8 Permanent Node Discovery

After all device commissioning steps are completed, the controller will end the device commissioning process and trigger discovery of the device as a permanent device under a new session. At this point, the device is considered commissioned and fully discovered; normal operation begins.

6.2 DEVICE AUGMENTATION

Similar to device discovery and commissioning (see Section 6.1) augmentation or changes (e.g. addition of a new degree, SRG, or xponder) to a previously deployed Open ROADM MSA device are provisioned through a zero-touch procedure to simplify the augmentation step. The zero-touch procedure allows for changes to a device based on a new planning template for the device and the current software/firmware and Open ROADM MSA version of the device. The MSA does not specify the operations required by an Open ROADM controller; it is possible a controller could implement the augmentation steps differently than specified in this section.

The controller augments the device in the same order as device commissioning based on the contents of the new planning template data (see Section 6.1for provisioning details):

- Provision *shelves*
- Provision *circuit-packs* and associated *ports*
- Provision *degree*
- Provision SRG (*shared-risk-group*)
- Provision ILA (*line-amplifier*)
- Provision *xponder*
- Provision Physical Links (*physical-link*)
- Provision *interfaces* (OAMP and OSC Ethernet, OTS, OMS)
- Provision *protocols* (LLDP and RSTP)

Like the planning template in device commissioning, the new planning template for device augmentation is loaded (pushed) into the Open ROADM controller. Note: the specific method for loading the planning template (e.g. planning tool, manually configured, vendor provided, etc.) and the planning template itself are not standardized by the Open ROADM MSA. However, the planning template must provide enough data (*shelves/circuit-packs/ports attributes*, etc.) to allow the controller to correctly augment the device using the Open ROADM MSA device model. One possible implementation of the planning template is a JSON file containing a subset of the Open ROADM MSA device model that would be needed to configure the device.

In the device augmentation procedure, the controller sends the provisioning commands to the device immediately after receiving the new planning template. Provisioning commands are sent regardless of the state of the equipment, installed or not (e.g. pre-provisioning). If the equipment is installed (based on physical inventory information), the equipment and interfaces are placed in an *inService administration-state* prior to checking for any alarms generated against the newly provisioned equipment. If no new alarms are detected, the equipment is moved to the deployed state (*lifecycle-state=deployed*). It is expected any required firmware updates are performed automatically during the device augmentation process.

If the equipment is not installed, the equipment and interfaces are placed in an *outOfService administration-state* in order to suppress alarms. However, a plug-in event (*change-notification*) is expected even if the associated *shelf* or *circuit-pack* is in an *outOfService administration-state* (see *change-notification* example in section 5.6.2.1). Note: the equipment installation detection, alarm checking, and deployment steps are done on an individual basis (one at a time) with consideration for equipment hierarchy.

6.3 OMS LINK PLANNING AND DISCOVERY

Optical Multiplex Section (OMS) link data is pushed to the controller using the Open ROADM MSA network model[6]. The OMS link plan should include the near end and far end degree information on the two sides of the link. The Open ROADM MSA device supports the Link Layer Discovery Protocol (LLDP) over the Ethernet OSC. LLDP is used to support the discovery of the link in the network.

Two key fields are used for the link discovery in the LLDP message:

- *SysName* (TLV Type 5): contains the NE's *node-id*
- *PortID* (TLV Type 2, sub-type 5 interface name or sub-type 7 locally assigned): contains the NE's OSC interface name

Open ROADM MSA implementations:

[6]The Open ROADM network model resides on the Open ROADM controller (or other management system) and not on the device.

- may choose to send either the sub-type 5 or sub-type 7 *PortID* format. Whichever format is selected, the value that is sent should match the Ethernet OSC's interface name.
- should be able to process the receipt of either the sub-type 5 or sub-type 7 *PortID*.
- should support the ability to receive and ignore other LLDP mandatory and optional TLVs that are not directly used by the Open ROADM MSA.

An LLDP change notification (*lldp-nbr-info-change notification*) is issued from the Open ROADM MSA device whenever the neighbor LLDP information is changed. The controller uses this information to correlate the OSC transmit LLDP information with the OSC receive LLDP information to discover the link connectivity.

The discovered link is compared to the planned OMS link. If they match, then the controller will measure the span loss and take a baseline OTDR reading of the link (refer to Section 10.5 for OTDR scan details) to complete the OMS link commissioning.

6.4 LIGHT TOUCH PROVISIONING (LTP)

Light Touch Provisioning (LTP) is an operation to perform initial provisioning of an OpenROADM device to achieve DCN connectivity and access to the OpenROADM controller. The One Touch Provisioning (OTP) procedure described in section 6.1 achieves the same goal under the conditions of already having DCN connectivity and reachability of a DHCP server. When these two conditions are not met, LTP is used instead of OTP. One typical example is the "Remote Transponder" configuration described in section 5.4. For this (and similar) configuration, some initial provisioning is required to set up DCN connectivity, e. g. via a GCC link.

In this example of setting up a GCC link, Light Touch Provisioning would cover:

- selection of the connecting link and port
- equipment provisioning (IO card, optical module)
- optical parameters of the connecting port (frequency, power)
- OTN parameters (OTU or OTSI/FlexO/OTSIG/OTUCn). gcc-channel-type is fixed to GCC0.
- IP parameters (loopback address, default route)

The general Light Touch Provisioning procedure consists of the following steps during device installation:

- the device vendor provides "LTP provisioning templates" (one for IPv4, one for IPv6) for the remote transponder device type. These templates are in JSON format (see org-openroadm-ltp-template.yang)
- the OpenROADM controller generates the "LTP provisioning file" for the device to be installed. This file is in XML format and is provided to the installation technician
- device hardware is installed, and fiber links are connected
- the installer connects a craft terminal to the CIT port of the device (see section 5.5)
- the installer points a web browser to the fixed IP address of the device. This address is taken from vendor documentation and provided together with the LTP provisioning file
- the web browser displays the LTP page of the device
- the installer uses the "provisioning file" upload button on this web page to transfer the LTP provisioning file to the device and start execution
- web browser displays status of the LTP operation
- installer verifies connectivity to the OpenROADM controller
- OpenROADM controller completes device provisioning using the standard ZTP mechanisms and configuration templates

The Light Touch Provisioning concept and mechanism is supported by OpenROADM devices supporting MSA 12.0 or newer. However, LTP does not have a dependency on the version of the Yang model of a device. For special cases or for testing, it is therefore possible to implement LTP already on devices supporting a model earlier than MSA 12.0.

For the "Remote Transponder" use case described in section 5.4, GCC and LTP support is needed. Transponder hardware for "remote" deployment must therefore be upgraded to MSA 12.0 (or newer) before delivery to the installation site.

6.4.1 Format of LTP Templates and Provisioning Files

The “LTP Provisioning Template” is a JSON file provided by the device vendor (similar to the manifest files defined elsewhere in this whitepaper). Its format is defined by the org-openroadm-ltp-template.yang file of the MSA model. It consists of a header section and a sequence of Netconf command to perform the LTP provisioning steps on the device. As the Netconf commands are different, separate templates are needed for IPv4 and IPv6 provisioning.

When a specific (remote transponder) device is to be installed, the controller generates the “LTP Provisioning File” for this device. This is an XML file (to simplify processing in the device) and follows the same org-openroadm-ltp-template.yang format specification as the template. To create the provisioning file, the controller replaces the variables in the template by their concrete values and converts the result into XML. This conversion also provides the correct XML namespaces needed by the Netconf protocol and the OpenROADM MSA model.

In the template, different command sequences can be specified for different device software versions.

The header section includes:

- device vendor, device model and node-type
- some free-form “management information”, which is meta-data added by the controller to identify the device, time stamp the file creation, and any other installation process information needed by the installation technicians
- global timeout value for commands (how long to wait for a command response)

Each Netconf command sequence section includes:

- list of applicable software versions
- sequence of Netconf commands: XML documents of edit-config or rpc Netconf commands, optional response timeout

The syntax of LTP templates is formally defined by a Yang file. The corresponding tree view is as follows:

```
module: org-openroadm-ltp-template
  +--rw ltp-template
     +--rw vendor                    string
     +--rw model*                    string
     +--rw node-type*                or-cmn-node-types:node-types
     +--rw management-information?   string
     +--rw global-sync-timeout?      uint16
     +--rw command-sequence*         [sequence-index]
        +--rw sequence-index           uint8
        +--rw device-sw-version*       string
        +--rw command*                 [command-index]
           +--rw command-index           uint8
           +--rw netconf-operation       <anydata>
           +--rw sync-timeout?           uint16
```

An LTP template contains references to variables, which are replaced by the controller when an LTP template is transformed into an LTP provisioning file. The following table defines the available variable names:

__LTP_MANAGEMENT_INFORMATION	Meta-data inserted by the management system that converts the JSON template into an LTP provisioning file for a specific system. This management information is some text, intended for human consumption and used by the installation technicians to locate and identify the target system. It may contain other operator specific information as needed by an operator's installation process.
__NODEID **__CLLI**	Primary device identification variables. These values are made available by the installation file generator and the vendor-specific template decides whether they are used or not. Example: a template might decide to not use them to avoid additional restarts during LTP. NODEID and CLLI data should also be present in the __LTP_MANAGEMENT_INFORMATION and in the filename of the generated LTP provisioning file.
__IPADDRESS	Mandatory IPv4 or IPv6 address under which the remote transponder will be reachable after the LTP procedure. Note: As the configuration commands are different, separate templates must be provided for IPv4 and IPv6.
__PREFIXLENGTH **__DEFAULTGATEWAY**	The "prefixlength" and "defaultgateway" variables are supported by LTP. For the Remote Transponder use case they are not typically needed as the remote transponder is an IP host connected to its gateway via a point-to-point link.
__GCCLINK_FREQUENCY	DWDM frequency of the line signal (e. g. OTUC4 or OTU4)
__LATITUDE **__LONGITUDE** **__RACK_1** **__SHELFPOSITION** **__DUEDATE**	Secondary device identification

Table 19: Variables in LTP Templates

6.4.2 General Rules for LTP Provisioning

The following rules generally apply to all LTP provisioning templates:

- Provision the /device/info subtree, see also list of variables in previous section, incl. potential vendor-specific restrictions
- Generally, set administrative-state of all objects provisioned below to "inService"
- Provision minimum set of shelves, cards, optical modules and ports to operate the device and the line interface carrying the GCC channel (e. g. shelf, controller, fans, power supplies, optical modules, etc.)
- Provision frequency (use variable) and optical power (fixed) of line interface
- Provision OTN stack of line interface
- Provision softwareLoopback interface (with __IPADDRESS, prefix-length=32 or 128, etc.)
- Disable DHCPv4 and DHCPv6 client on Ethernet DCN interface
- Provision GCC interface (supporting interface, ipv6 subcontainer, gcc subcontainer)
- Provision default route towards GCC interface

6.4.3 Light Touch Provisioning for other Use Cases

In future releases, LTP might also be used for other use cases, for example:

- establish communication between OpenROADM device and controller over OSC without DHCP

- establish communication over standard OAMP Ethernet port but without DHCP server in the network

Common for all LTP use cases is the provisioning of the static IP address of the device and the establishment of the relationship between physical device (identified by location, CLLI) and configuration in the controller (identified by static IP address). For these cases the initial part of the One Touch Procedure to establish the relationship using the device serial number is no longer required.

6.4.4 Access to Light Touch Provisioning

The LTP functionality is accessed by opening the LTP web page on the craft terminal (see section 5.5). The URL for LTP is: **https://<vendor-specific-ip>/ltp**

To access the LTP function, the user must authenticate. If the device is still in factory default state, both username and password are set to “openroadm”.

A typical LTP web interface might look like:

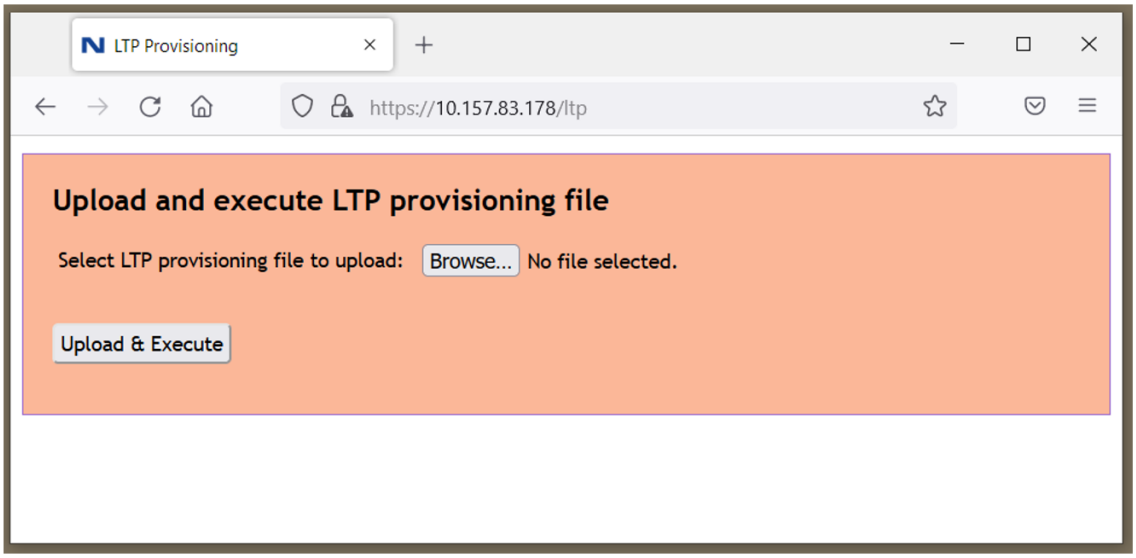

Figure 76: Example Web Interface for LTP

7 DEVICE SOFTWARE, FIRMWARE, AND DATABASE OPERATIONS

7.1 SOFTWARE AND FIRMWARE OPERATIONS

The Open ROADM MSA API defines software and firmware operations, including the ability to upgrade the current software version of the device, as well as, firmware updates to circuit packs.

7.1.1 Software Upgrade

A software upgrade is expected to take place as a series of operations as specified by the vendor in the manifest file for the specific device. The procedure is to transfer the software load from external storage (SFTP server) to the device and then initiate the software upgrade procedure.

The following device RPCs are defined by the Open ROADM MSA model for software upgrade operations:

- *transfer [action=download]* is used by the device to download a software file from the SFTP server to the device prior to a sw-stage operation (see Section 5.2.4 for SFTP details).
- *sw-stage [filename]* is used by the device to stage a software download operation. A sw-stage-notification is generated as a result of a sw-stage operation.
- *delete-file [filename]* is used by the device to delete a software file from the device during or after a sw-stage operation (see Section 5.2.6 for details).
- *sw-activate [version] [validationTimer]* is used by the device to activate a new software load after a sw-stage operation. A sw-activate-notification is generated as a result of a sw-activate operation.
- *cancel-validation-timer [accept]* is used by the device to cancel a validation timer after a sw-activate operation and accept or reject the new software load. For vendor implementations that don't support a validation timer, support for this command is not required

A *pending-sw* attribute is provided as a read-only attribute that is only populated and used during the staging or validation phase of an in-progress software upgrade operation. During the staging phase, the software version (*sw-version*) being staged is provided and during the validation phase, the remaining time of the validation timer (*sw-validation-timer*) and the software activation time (*activation-date-time*) are provided (refer to Table 20

Refer to Appendix A.1 for software download manifest file details.

7.1.2 Software Downgrade

A software downgrade is not supported in-service in Open ROADM MSA devices.

7.1.3 Firmware Upgrade

Firmware upgrades are supported as of Open ROADM MSA v2. Firmware is associated with circuit-packs. A device can provide information about the firmware on the circuit pack with the *software-load-version* attribute and a list of either circuit-pack features or components. Implementations may describe their firmware as features, components, or both.

The circuit pack features list the firmware as a series of features or capabilities that the firmware enables (e.g. OTN delay measurement feature). The feature list provides an indication if the feature is activated (*activated=true*) or not activated (*activated=false*). If the feature is listed as not being activated, then this indicates there is new firmware available for the circuit pack.

The component list provides the capability to describe the firmware as a series of firmware versions associated with different components on the circuit pack. This list provides the details of the current load version (*current-version*) and a version to apply (*version-to-apply*). If the version to apply is different from the current version, then this indicates a new firmware is available for this circuit pack.

If the loading of the new firmware is non-service affecting, the firmware should be applied automatically to the circuit-pack during a software upgrade operation (see Section 7.1.1 for details). For firmware that is service affecting, a cold restart or power cycle of the circuit pack may be required to load the new firmware (see Section 10.1.2 for circuit-pack level restart model details).

The *boot=true* attribute indicates the firmware is a boot-loader or similar that doesn't have redundancy or protection if the firmware upgrade fails. For this reason, bootloader firmware must be updated manually.

The *fw-update* RPC is an asynchronous operation defined to provide an operator the ability to manually load the firmware into a circuit-pack. This command can also be used to force the reload of a firmware load into the circuit-pack. Note: implementations may not support the upgrade of firmware while the new software is in the validation stage of a software upgrade operation. In this case, firmware download would be supported after the software load is committed.

Operation	Async/Sync	Notification	Pending-SW
transfer	async	transfer-notification	<no effect>
delete-file	sync	None	<no effect>
sw-stage	sync	None	After staging completes sw-version: new (to be activated) version sw-validation-timer: not present sw-activation-date-time: not present
	async	sw-stage-notification	
sw-activate (with validation)	async	sw-activate-notification sw-active-notification-type = activate (sent before or after reboot)	sw-version: new version (running) sw-validation-timer: remaining time of validation timer sw-activation-date-time: time of activation of the new load
sw-activate (without validation)	async	sw-activate-notification sw-active-notification-type = activate sw-activate-notification sw-active-notification-type = commit (two notifications sent, activate can be before or after reboot, commit after reboot)	Not present
cancel-validation-timer (accept=true)	sync	None	Not present
	async	sw-activate-notification sw-active-notification-type = commit	Not present
cancel-validation-timer (accept=false)	async	sw-activate-notification sw-active-notification-type = cancel (after reboot)	Not present
Automatic Rollback	automatic	sw-activate-notification sw-active-notification-type = cancel (after reboot)	Not present

Table 20: Software Download Notifications and Pending-SW

7.2 DATABASE OPERATIONS

The Open ROADM MSA API defines operations against the database, including the ability to backup a database to an external storage location (SFTP server) and restore a database from an external storage device (STP server). Change notifications are supported against the database to identify configuration changes to the device (refer to Section 5.6.2.1).

7.2.1 Database Backup

The database backup operation allows the local device configuration data to be backed-up into a file and stored in an external storage location (SFTP server). The procedure is to initiate a backup on the device to store the database as a local file on the device and then transfer the file off the device into an external storage location (SFTP server). The following device RPCs are defined by the Open ROADM MSA model for a database backup operation:

- *db-backup [filename]* is used by the device to perform a database backup operation. A db-backup-notification is generated as a result of a db-backup operation.
- *transfer [action=upload]* is used by the device to upload a database backup file to the SFTP server after a db-backup operation (see Section 5.2.3 for SFTP details).
- *delete-file [filename]* is used by the device to delete a database file from the device. After a transfer operation (see Section 5.2.6 for details)

Operation	Async/Sync	Notification	Reboot	Database-Info
transfer	async	transfer-notification	None	<no effect>
delete-file	sync	None	None	<no effect>
db-backup	sync	None	None	<no effect>
	async	db-backup-notification		

Table 21: Database Backup Operation Notifications, Reboot, and Database-Info

Refer to Appendix A.2 for database backup manifest file details.

7.2.2 Database Restore

The database restore operation allows the service provider to restore a previous database backup. This operation may be necessary if there was a device database corruption or configuration issue, and the service provider wants to revert to a previously known state. The procedure is to transfer the backup database file from external storage (SFTP server) to the device and then initiate the restoration of the database. The device would then load the database into non-persistent memory, reboot and restart using the saved database. Note: some vendor implementations may not require a reboot of the device to restore a database.

The following device RPCs are defined by the Open ROADM MSA model for database restore operations:

- *transfer [action=download]* is used by the device to download a database backup file from the SFTP server to the device prior to a db-restore operation (see Section 5.2.4 for SFTP details).
- *db-restore [filename] [nodeIDCheck]* is used by the device to perform a database restore operation. The *nodeID-Check* is used to specify whether a database for restoration is a backup of the intended device. If required (*nodeI-DCheck=true*), the node-id identified in the database file is compared against the current node-id of the device. A *db-restore-notification* is generated as a result of a *db-restore* operation.
- *delete-file [filename]* is used by the device to delete a database file from the device after a *db-restore* operation (see Section 5.2.6 for details)
- *db-activate [rollBackTimer]* is used by the device to activate a database backup after a *db-restore* operation. A *db-activate-notification* is generated as a result of a *db-activate* operation
- *cancel-rollback-timer [accept]* is used by the device to cancel a rollback timer after a *db-activate* operation and accept or reject the newly restored database. For vendor implementations that don't support a database rollback, support for this command is not required.

A *database-info* attribute is provided as a read-only attribute identifying information about when the current database was created or last restored (*last-restored-time*). During the validation phase of an in-progress database restore operation, the remaining time of the rollback timer (*rollback-timer*) and the database activation time (activation-date-time) are provided (refer to Table 22).

Refer to Appendix A.3 for database restore manifest file details.

Operation	Async/Sync	Notification	Reboot	Database-Info
transfer	async	transfer-notification	None	<no effect>
delete-file	sync	None	None	<no effect>
db-restore	sync	None	None	<no effect>
	async	db-restore-notification		
db-activate (with rollback)	sync	None	None	last-restored-time: time of DB activate rollback-timer: remaining time of the rollback timer activation-date-time: time of activation of the restored database
	async	db-activate-notification db-active-notification-type = activate (if reboot required, sent before or after reboot)	Implementation specific (may or may not reboot)	
db-activate (without rollback)	sync	None	None	last-restored-time: time of DB activate rollback-timer: not shown activation-date-time: not shown
	async	db-activate-notification db-active-notification-type = activate db-activate-notification db-active-notification-type = commit (if reboot required, two notifications sent, before or after activate reboot, commit after reboot)	Implementation specific (may or may not reboot)	
cancel-rollback-timer (accept=false)	sync	None	None	last-restored-time: original time of last-restored-time before the db-activate was issued rollback-timer: not shown activation-date-time: not shown
	async	db-activate-notification db-active-notification-type = cancel (if reboot required, after reboot)	Implementation specific (may or may not reboot)	
Automatic Rollback	automatic	db-activate-notification db-active-notification-type = cancel (if reboot required, after reboot)	Implementation specific (may or may not reboot)	Not present

Table 22: Database Restore Operation Notifications, Reboot, and Database-Info

7.2.3 Database Restore to Factory Defaults

There may be instances when a device needs to be restored to its factory default configuration. A database-init RPC is modeled against the device to perform this operation. Note: this operation is not driven by the manifest file; it is driven by the controller.

Note: DHCP is re-enabled and the username and password are set to “openroadm”.

```
<get>
    <filter>
        <currentPmlist xmlns="http://org/openroadm/pm">
            <currentPm>
                <granularity>15min</granularity>
            </currentPm>
        </currentPmlist>
    </filter>
</get>
```

YANG sub-tree 64: NETCONF to retrieve the current PM registers with a 15-minute granularity

```
<get>
    <filter>
        <currentPmlist xmlns="http://org/openroadm/pm">
            <currentPm>
                <granularity>24Hour</granularity>
            </currentPm>
        </currentPmlist>
    </filter>
</get>
```

YANG sub-tree 65: NETCONF to retrieve the current PM registers with a 24-hour granularity

8 PERFORMANCE MONITORING (PM), ALARMS, AND TCAS

8.1 PERFORMANCE MONITORING

The device model supports the retrieval of performance monitoring (PM) data. There are two primary lists associated with PM – currentPmlist and historicalPmlist. Refer to Appendix B: for details on supported PMs.

8.1.1 Current PM List

The *currentPmlist* provides the PM data that is currently being collected and updated on the device. The PM is collected in 15-minute, 24-hour, and untimed (granularity = NA) bins. An example invocation using NETCONF to retrieve the current PM registers with a 15-minute granularity is as follows (Example 64):

An example invocation using NETCONF to retrieve the current PM registers with a 24-hour granularity is as follows (Example 65):

8.1.2 Historical PM List

The *historicalPmlist* contains all the binned PM values collected by the device during some time window in 15-min intervals and 1-day intervals. An example invocation using NETCONF to retrieve the first bin from the historical PM registers with a 15-minute granularity is as follows (Example 66):

Note: support for retrieving the historical PM list using a NETCONF get is optional for vendors to support.

8.1.2.1 File-based Historical PM Retrieval

Implementations must support the ability to retrieve historical PMs using a file transfer mechanism (default mechanism). An asynchronous RPC mechanism is defined to support the ability to retrieve the historical PM via a file generated by the device.

The *collect-historical-pm-file* RPC is used to initiate the creation of the historical PM file. The input to the RPC indicates the start bin (*from-bin-number*) and end bin (*to-bin-number*) of the request, along with the granularity. Note that granularity is singular so separate RPC calls would be required to obtain the 15-min and 24-hour granularities. Historical data is not applicable to the untimed (*notApplicable*) granularity.

```
<get>
    <filter>
        <historicalPmlist xmlns="http://org/openroadm/pm">
            <historicalPm>
                <id/>
                <resource/>
                <layerRate/>
                <binned-pm>
                    <bin-number>1</bin-number>
                </binned-pm>
                <granularity>15min</granularity>
            </historicalPm>
        </historicalPmlist>
    </filter>
</get>
```

YANG sub-tree 66: NETCONF to retrieve the first bin from the historical PM

The response to the RPC provides the filename used to store the PM data. The NE auto-generates the *pm-filename* and it is recommended that this filename be a unique name for every RPC call. This command should return immediately upon validating the RPC command and initiating the PM retrieval and file storage to avoid a timeout on the controller.

In the background, the device should collect the requested PM data and store it in the file. Upon completion of putting the data in the file, the device should issue a notification indicating that the PM file collection has been successfully completed or failed (*historical-pm-collect-result*). This notification should indicate the success or failure of the operation, and if it failed, it would also provide an error indication in the *status-message*. The notification will also contain the filename to allow for correlation with the RPC command. Although the filename is modeled as optional in the RPC response and notification, it is recommended that implementations provide the filename in both the RPC response and the notification so that the controller can easily correlate the notification with the original RPC request.

The controller will then transfer the file from the NE using the file transfer RPC (see Section 5.1). It is the responsibility of the controller to delete the PM file from the device once the file transfer has completed. This will be necessary to avoid exhausting the file store limits if multiple PM files are kept on the device.

The file content should be written in an xml format based on the Open ROADM MSA *historical-pm-list* YANG definition and the file should be compressed using gzip. An example file format is shown below (Example 67):

8.1.3 Clearing PMs

An asynchronous RPC mechanism (*clear-pm*) is defined to support the ability to clear all PMs or current PMs. PMs can be cleared on a per resource and per pm-type basis. The input to the RPC indicates the *pm-type* (*all* or current) along with the granularity (default is *15min*). Note that granularity is singular so separate RPC calls would be required to obtain the 15-min, 24-hour, and untimed (notApplicable) granularities.

8.1.4 PM Behavior

8.1.4.1 Analog and Digital PM

PM monitor types can be classified as either analog or digital.

Analog PMs are gauge or metered type parameters that can increase or decrease during a collection cycle. These PM monitor types can have an associated raw (instantaneous) reading as well as collect min, max and average statistics from the start of collection within the bin.

An example of an analog PM would be the optical power measurement, e.g. *opticalPowerOutput, opticalPowerOutputMin, opticalPowerOutputMax, and opticalPowerOutputAvg*. For averaging of optical power measurements, the averaging should be done in linear space (milliwatts) and then converted back to the logarithmic space (decibels) for reporting.

Digital PMs are counter type parameters that will be non-decreasing within a collection cycle, e.g. Coding Violations, Errored Seconds Section, In Frame, etc. . .

```
<historical-pm-list>
    <historical-pm-entry>
        <pm-resource-type>port</pm-resource-type>
        <pm-resource-type-extension/>
        <pm-resource-instance>/org-openroadm-device:org-openroadm-device/org-openroadm-dev
↪   ice:circuit-packs[org-openroadm-device:circuit-pack-name='1/0/2/E1']/org-openroadm-dev
↪   ice:ports[org-openroadm-device:port-name='1']</pm-resource-instance>
        <historical-pm>
            <type>vendorExtension</type>
            <extension>chromaticDispersion</extension>
            <location>nearEnd</location>
            <direction>rx</direction>
            <measurement>
                <granularity>15min</granularity>
                <bin-number>1</bin-number>
                <pmParameterValue>0.0</pmParameterValue>
                <validity>complete</validity>
                <completion-time>2018-07-24T10:14:59+00:00</completion-time>
            </measurement>
            <measurement>
                <granularity>15min</granularity>
                <bin-number>2</bin-number>
                <pmParameterValue>0.0</pmParameterValue>
                <validity>complete</validity>
                <completion-time>2018-07-24T09:59:59+00:00</completion-time>
            </measurement>
        </historical-pm>
        <historical-pm>
            ...
        </historical-pm>
        ...
        </historical-pm-entry>
        ...
        <historical-pm-entry>
    </historical-pm-entry>
</historical-pm-list>
```

YANG sub-tree 67: Historical PM-list

8.1.4.2 Granularity

The PM model supports three granularities: 15-minute, 24-hour, and untimed (*notApplicable*). Refer to Table 23 for the requirements that implementations must meet in support of PM granularity.

Granularity	Current/Historical	Analog PM		Digital PM
		Raw	Min/Max/Avg	
Untimed (notApplicable)	**Current**	*Optional*	*Optional*	*Mandatory*
	Historical	*Not Applicable*	*Not Applicable*	*Not Applicable*
15min	**Current**	*Mandatory*	*Mandatory*	*Mandatory*
	Historical	*Optional(snapshot at end of bin)*	*Mandatory*	*Mandatory*
24Hour	**Current**	*Mandatory*	*Mandatory*	*Mandatory*
	Historical	*Optional(snapshot at end of bin)*	*Mandatory*	*Mandatory*

Table 23: PM Granularity Support

Notes:

1. The analog raw PM would have identical values in the untimed, 15m and 24h current bins.
2. Analog raw PM was made mandatory for both 15m and 24h bins even though they would have the same value. This was to allow the full set of raw/min/max/avg to be retrieved together either under the 15m or 24h granularity. Otherwise the PM would have to be retrieved under both the untimed and 15m/24h granularities.
3. Other granularities (such as 1m, 1h) can be optionally supported by implementations. If supported, it is assumed that these granularities would follow the requirements for 15m and 24h granularities.

8.1.4.3 Validity

The validity field can take the values *complete, partial*, and *suspect*.

The behavior of the validity depends on the granularity and current/historical list for analog and digital PM:

1. Analog Raw PM – Untimed current, 15m current, 24h current, 15m historical and 24h historical:
   - *complete*: Indicates that the current (instantaneous) or historical snapshot reading is valid
   - *partial*: Not used
   - *suspect*: Indicates that the current (instantaneous) or historical snapshot reading is not valid
2. Analog Min/Max/Avg PM and Digital PM – untimed current
   - *complete*: Indicates that there has been no invalid reading since the PM monitor type was created or last reset
   - *partial*: Not used
   - *suspect*: Indicates that there has been at least one invalid reading since the PM monitor type was created or last reset
3. Analog Min/Max/Avg PM and Digital PM – 15m and 24h current
   - *complete*: Not used
   - *partial*: Indicates that PM collection is still occurring for this bin, the collection has been continuously been collected since the start of the bin, and there were no invalid readings in the bin
   - *suspect*: Indicates that the PM collection started after the bin start time, the PM was reset in the current bin, or that there was at least one invalid reading in the bin since the start of collection for the bin
4. Analog Min/Max/Avg PM and Digital PM – 15m and 24h historical
   - *complete*: Indicates that PM was collected for the entire duration of the bin and that there were no invalid readings in the bin
   - *partial*: Not used
   - *suspect*: Indicates that the PM collection did not collect data for the entire duration of the bin or that there was at least one invalid reading in the bin since the start of collection for the bin.

When data is not available to be read, whether temporary or permanent, then the validity would be shown as suspect. This could be the case when PM is collected on remote modules and the device is unable to communicate with that module.

8.1.4.4 Other PM Behavior Notes

For analog raw historical PM data, the value represents the instantaneous value at the end of the bin. The bin start and stop times are referenced to UTC time on the device. Note that the Open ROADM MSA only supports UTC time.

- For the 15m bins, the start time would be at xx:00:00, xx:15:00, xx:30:00, and xx:45:00 UTC. The end times would be xx:14:59, xx:29:59, xx:44:59, and xx:59:59 UTC, respectively.
- For the 24h bins, the start time would be at 00:00:00 UTC and the end time would be at 23:59:59 UTC.

When retrieving data from the current bin, the *retrievalTime* represents the time of the retrieval at which the current PM data is retrieved and returned.

For historical bin data, only the end time is reported (completionTime) which represents the end time of the bin. Partial only applies to current binned data to indicate the bin data is valid but hasn't completed the full collection cycle for the bin. When a new bin is started, the validity resets to partial and will either remain partial or transition to suspect once data begins collection. If the bin is still partial at the end of the current bin, then it transitions to complete when propagating into the historical bin.

Once a bin goes suspect, it would typically always be suspect unless the bin is re-initialized (untimed granularity only). Note: implementations that support a concept of "data temporarily not available" due to a temporary access issue (e.g., to a remote circuit-pack) could report the data as suspect and then back to partial/complete once the access issue is resolved.

Changing date/time that could cause a bin to be long or short is reported as suspect. This affects the 15m and 24h current PM data for analog min/max/avg and digital PM. The marking of data as suspect due to long or short may be performed at the end of the bin before rolling into the historical list. This would allow implementations to check the total collection time to see if the data was long or short at the bin end time.

8.2 ALARMS

For details on supported alarms (probable causes), refer to Appendix B:

8.3 Threshold Crossing Alerts (TCAs)

The current OpenROADM release supports ANSI-type thresholding for analogue and digital PM monitor types. Whenever a monitor type crosses a defined threshold within an accumulation period, a transient notification is sent. When the threshold is crossed again in the next period or is already crossed at the beginning of the period, a new notification is sent. The notifications are "transient", i. e. there are no "clears".

For analogue counters, thresholds can be set against the 15-min granularity period, for digital counters, thresholds are supported for both the 15-min and 1-day granularities.

A later OpenROADM release will additionally support ETSI-type thresholding, where threshold crossings are standing conditions with raise and clear events.

For analogue counters, a device may support absolute or relative thresholds:

- in case of absolute thresholds, high and low threshold values define the limits triggering a notification when crossed by the PM monitor type
- in case of relative thresholds, high and low threshold values define the limits relative to a baseline value. This baseline value is established by the device either by an explicit controller-driven operation (RPC) or automatically, e. g. when a port is in operation for a certain time and is error free. The device indicates support for automatic baselining by announcing the Yang feature *tca-auto-baselining*. The relative thresholds are either supported by separate high and low values or by a single "symmetrical" value expressing identical high and low thresholds.

A device indicates whether it supports separate and individual threshold-settings per resource instance (e. g. individual port or interface) or via *TCA profiles*.

Support for TCA profiles is announced by the Yang feature *tca-profiles*. In this case the controller configures threshold values in a preconfigured set of TCA profiles. Such a profile can then be assigned to a resource instance or simultaneously to a group of resource instances. After that, this instance (or group of instances) uses the threshold values from the assigned profile.

8.3.1 Threshold Support Matrix

The following table shows the supported threshold types for each of the available PM monitor types. The representation in the Yang model is explained in the following sub-sections.

For analog PM, TCAs are only supported against the analog raw PM. It is set against the 15-min granularity. The TCA is a transient notification that would be raised when the value was less than the low threshold or greater than the high threshold.

Measurement Type	Threshold Type	Counter Types supported for ANSI Thresholding					
		Analog				Digital	
		Raw			Min / Max / Avg		
		Untimed	15-min	1-day	Untimed / 15-min / 1-day	Untimed	15-min / 1-day
absolute	high	-	raise	-	-	-	raise
	low	-	raise	-	-	-	-
relative	high	-	raise	-	-	-	-
	low	-	raise	-	-	-	-
	symmetrical	-	raise	-	-	-	-

Figure 77: TCA Support Matrix

Only one transient notification for high and low threshold crossing would be issued per 15m time period. The TCA would be issued again for subsequent 15m time periods (when bins roll over to a new binning period) if the value was above/below the high/low threshold, or again if the threshold is crossed.

Analog raw TCAs should be consistent with the min and max readings. If a min reading goes below the low threshold or the max reading goes above the high threshold, then a TCA should have been issued for the analog raw PM.

Thresholds for analogue counters can also be relative to a baseline value. Depending on a device's architecture, relative thresholds may be specified using two individual high and low values or using a single symmetrical value.

For digital PM, thresholds can be independently set against the 15m or 24h granularities.

Note: It may be possible that implementations do not support TCAs for all PM monitor types. A future version of this document would clarify which PM monitor types must support TCAs.

8.3.2 Baselining and Relative Thresholds

Analogue monitors often use baselining where a snapshot of the current PM reading is used as reference for subsequent threshold supervision. Typical applications are optical receive and transmit powers: after a system has been installed, a (new) optical channel has been configured and has achieved a stable condition, a baselining operation determines and stores the current optical power values.

Threshold supervision can then use these stored values as a reference.

Baselining can be an explicit operation, triggered via the *tca-set-baseline* RPC by the controller, or can be executed autonomously by the device when it detects that a newly configured entity (e. g. port and optical channel) has reached a stable state. The device announces support for this latter functionality by indicating the Yang feature *tca-auto-baselining*.

Baselined values and the current state of baselining are stored persistently by the device. Value and state can be retrieved by the controller from the list of baseline-entries:

```
+--rw potential-tca-list
|  +--rw tca-entry* [tca-resource-type
|              tca-resource-type-extension tca-resource-instance]
|     +--rw tca-resource-instance        instance-identifier
|     +--rw tca-resource-type            resource-type-enum
|     +--rw tca-resource-type-extension  string
|        ...
|     +--ro baseline-entry* [type extension location direction]
|     |  +--ro type              pm-names-enum
|     |  +--ro extension         string
|     |  +--ro location          location-type
|     |  +--ro direction         direction-type
|     |  +--ro baseline          pm-data-type
|     |  +--ro baseline-state    baseline-state-enum
|     |  +--ro baseline-time?    yang:date-and-time
```

For a single *baseline-entry*, the *baseline* attribute contains the current baselined value, *baseline-state* the current state with values (*not-set, manually-set, automatically-set*), and *baseline-time* the date and time of the baselining. *baseline-time* is

absent when the state is *not-set*.

When a PM monitor uses baselining, its thresholds are specified relative to its baselined value. Relative thresholds can be specified by two separate high/low values or by a single symmetric value. The device indicates the use of baselining, absolute vs. relative thresholds and high/low vs. symmetric values in the *pm-entry* structure contained in the *potential-tca-list* or *tca-profile-list*:

```
+--rw pm-entry* [...]
      +--ro measurement-type?   measurement-enum
      +--rw measurement* [...]
         +--rw threshold-type       threshold-enum
         +--rw enabled?             boolean
         +--rw threshold-value      pm-data-type
         +--ro threshold-default?   pm-data-type
```

The *measurement-type* can take the values *absolute* or *relative* and with this the device indicates whether the thresholds are specified by the controller as absolute values or relative to a baseline.

The *threshold-type* takes the values *high*, *low*, or *symmetrical* and indicates whether the device implements high/low or symmetrical thresholding for a PM monitor. Note: *threshold-type* is RW in the Yang model as it serves as list key within the measurement list.

8.3.3 Threshold Value Settings and TCA Profile Assignment

A device can support individual threshold values per resource instance, or it can support threshold value set-up via TCA profiles. Profile support is indicated by the Yang feature *tca-profiles*. A mixture of individual threshold settings for some resources and setting via profile for other resources is not supported.

When a device supports individual (per resource instance) threshold values, these values are set directly per *tca-entry* in the *potential-tca-list* tree:

```
+--rw potential-tca-list
|  +--rw tca-entry* [tca-resource-type
              tca-resource-type-extension tca-resource-instance]
|     +--rw tca-resource-instance        instance-identifier
|     +--rw tca-resource-type            resource-type-enum
|     +--rw tca-resource-type-extension  string
|     +--rw enabled                      boolean
           ...
|     +--rw pm-entry* [type extension location direction]
                      {not tca-profiles}?
|        +--rw type                pm-names-enum
|        +--rw extension           string
|        +--rw location            location-type
|        +--rw direction           direction-type
|        +--ro measurement-type?   measurement-enum
|        +--rw measurement* [granularity threshold-type]
|           +--rw granularity          pm-granularity
|           +--rw threshold-type       threshold-enum
|           +--rw enabled?             boolean
|           +--rw threshold-value      pm-data-type
|           +--ro threshold-default?   pm-data-type
```

The device auto-creates and auto-deletes *potential-tca-list*/*tca-entry* instances when the corresponding resource instance(s) are created or deleted (and support thresholding). On creation, all attributes of the *tca-entry* subtree are set to device specific defaults as follows:

- thresholding is disabled, i. e. the *enabled* attribute(s) are set to FALSE
- if TCA profiles are supported, the direct or inherited profile configuration is set to a a default profile and the applicable profile list is set up depending on resource type
- if the resource instance supports one or more PM monitors with baselining functionality, these PM monitors are set up in the *baseline-entry* list, their state is set to *not-set*.

- if the device does NOT support TCA profiles, the pm-entry list is populated with all PM monitors and their associated threshold attributes

In case of profile support, a device provides a set of preconfigured TCA profiles. These profiles can be modified by the controller. Each resource instance (if it supports thresholding) indicates which ones of the profiles it can support.

To enable thresholding for a single resource instance (or a set of resource instances, see next section) the controller assigns one of the supported profiles to the instance. This profile assignment is done via a *tca-entry* in the *potential-tca-list* tree:

```
+--rw potential-tca-list
|  +--rw tca-entry* [tca-resource-type
                      tca-resource-type-extension tca-resource-instance]
|     +--rw tca-resource-instance        instance-identifier
|     +--rw tca-resource-type            resource-type-enum
|     +--rw tca-resource-type-extension  string
|     +--rw enabled                      Boolean
|     +--rw (tca-profile-configuration)? {tca-profiles}?
|     |  +--:(direct-configuration)
|     |  |  +--rw tca-configured-profile      string
|     |  |  +--ro tca-applicable-profiles*    string
|     |  +--:(inherited-configuration)
|     |     +--ro tca-active-profile          string
|     |     +--ro tca-profile-rootresource    instance-identifier
         ...
```

If the resource instance supports a direct profile configuration, the *tca-applicable-profiles* attribute contains the list of profiles supported by this instance. If all available profiles can be supported, the device can leave the applicable profiles list empty.

To assign a profile the controller sets *tca-configured-profile* to one of the supported TCA profiles.

A device architecture might support resource reconfigurations (e. g. rate changes of an existing interface) that also change the value of the *tca-applicable-profiles* list. If in such a case the *tca-configured-profile* is no longer in the *tca-applicable-profiles* list, the device will reset the configured profile to the creation time default.

8.3.4 Shared TCA Profiles

Depending on device architecture, multiple resource instances may share a single TCA profile. Such a related set of resource instances forms a sub-tree in the overall instance tree. The single TCA profile for such a sub-tree is set on the root instance of this sub-tree. Children of such a sub-tree root cannot have their profile set, but they do indicate the identity of the root and the profile set-up on the root.

In the Yang model, the root object of a profile-sharing sub-tree provides the *direct-configuration* set of attributes, the child instances the *inherited-configuration* attributes, see previous section.

The creation time for the individual tca-entry objects for a sub-tree is device-specific. A device may create the corresponding tca-entry immediately after a resource instance is created, or at a later point in time during the creation of the sub-tree.

The following diagram shows a typical scenario of such a configuration:

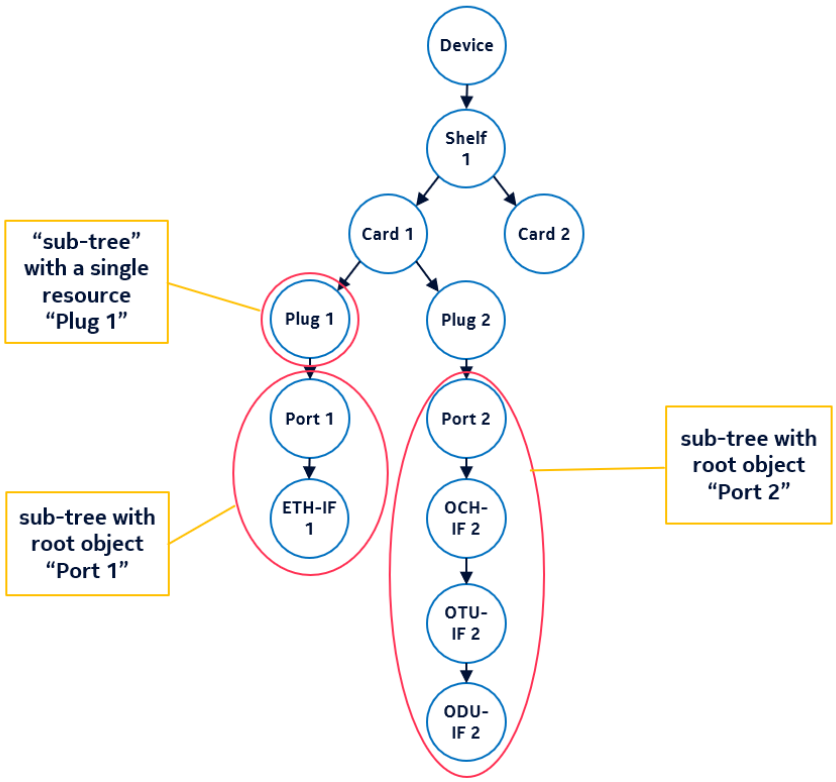

Figure 78: Shared TCA Profiles

8.3.5 TCA Profile Definition

A TCA profile is a list of individual "PM" entries (*pm-entry*). Each such entry defines all the thresholds for a certain combination of resource class and PM monitor.

A resource class is identified by: (brown color in the Yang tree below)

- resource type (e. g. shelf, circuit-pack, port, interface, connection, link, protection group, etc.)
- resource type extension (in case of vendor-specific extension)
- interface type (e. g. Ethernet, OCH, OTU, ODU, OTSI, NMC, etc.)
- interface rate (e. g. Ethernet line rate, OTUn, ODUn, etc.)

A PM monitor is identified by: (red color in the Yang tree below)

- PM counter name (*type*)
- extension (in case of vendor-specific counters)
- location (near-end, far-end, etc.)
- direction (RX, TX, N/A, etc.)

In the Yang model, the TCA profiles are defined by the following set of attributes:

```
+--rw tca-profile-list
   +--rw tca-profile-entry* [tca-profile-name]
      +--rw tca-profile-name           string
      +--rw tca-profile-description    string
      +--ro tca-profile-access         enumeration (enum: read, write)
      +--rw pm-entry* [tca-resource-type tca-resource-type-extension
                       tca-if-type tca-if-rate
                       type extension location direction]
         +--rw tca-resource-type              resource-type-enum
         +--rw tca-resource-type-extension    string
         +--rw tca-if-type                    tca-if-type-union
         +--rw tca-if-rate                    tca-if-rate-union
         +--rw type                           pm-names-enum
         +--rw extension                      string
         +--rw location                       location-type
         +--rw direction                      direction-type
         +--ro measurement-type?              measurement-enum
                                             (enum: absolute, relative)
         +--rw measurement* [granularity threshold-type]
            +--rw granularity          pm-granularity
            +--rw threshold-type       threshold-enum
                                       (enum: high, low, symmetrical)
            +--ro threshold-access     enumeration (enum: read, write)
            +--rw enabled?             boolean
            +--rw threshold-value      pm-data-type
            +--ro threshold-default?   pm-data-type
```

The *measurement* list (in green color) contains all the thresholds for a resource class / PM monitor combination.

It is possible that each and every profile offered by a device contains all possible threshold definitions. In such a case any profile can be assigned to any resource instance providing thresholding functionality.

It is also possible to have profiles with a subset of threshold definitions. In this case, each resource instance (with thresholding functionality) will announce the profiles it can support via the *tca-applicable-profiles* attribute.

The architecture of the TCA profiles allows to offer different sets of thresholds for different resource types, different interface types (e. g. Ethernet vs. OTN) or different rates (e. g. 10GBE, 100GBE, ODU2, ODU4, etc.).

A TCA profile can be read-only (*tca-profile-access=read*). In this case, none of its attributes can be modified by the controller (incl. *tca-profile-description*, *enabled*, and *threshold-value*).

If a TCA profile is writeable (*tca-profile-access=write*), the attributes *tca-profile-description*, *enabled*, and *threshold-value* can be modified by the controller.

Regardless of whether a profile is read-only or writeable:

- the controller cannot create new profiles or delete existing ones
- the controller cannot add or remove *pm-entry* items of a profile
- the controller cannot add or remove *measurement* items (thresholds) for a *pm-entry*

Each threshold item in the measurement list has its own *threshold-access* attribute with values *read* and *write* to indicate whether the threshold value can be modified by the controller.

For read-only profiles, *threshold-access* is always *read*, i. e. read-only.

For writeable profiles, *threshold-access* may indicate read-only if the threshold value depends on settings for another resource class or because of another interdependency. Typical example: threshold values for TCM1-related resource classes are writeable, but values for TCM2 . . . TCM6 related thresholds are read-only because they follow the values set for TCM1.

9 PROVISIONING ACTION USE CASES

This chapter describes provisioning workflows required to route client traffic over the network.

In an Open ROADM network, all configurations are performed through a device's NETCONF interfaces. However, in the interest of simplifying and alleviate the explanation, the XML encoding is not detailed in this description. The operations to provision equipment, interfaces and connections are presented in a descriptive way.

◊ A diamond symbol in the description indicates a point in which the flow of actions will depend on a validation or a decision that needs to take place at that point.

The elements to be configured, as shown in Figure 79 and Figure 80, are described in terms of the functional blocks considered by the Open ROADM model. Configuration of the Xponders will be described first.
Then we will describe the configuration of the add/drop link in the first ROADM, followed by the express link for intermediate ROADM NEs,and finally the turn back link configuration.
The deletion of interfaces and connections is presented next in a dedicated section.
The last section describes the provisioning of the actual equipment.

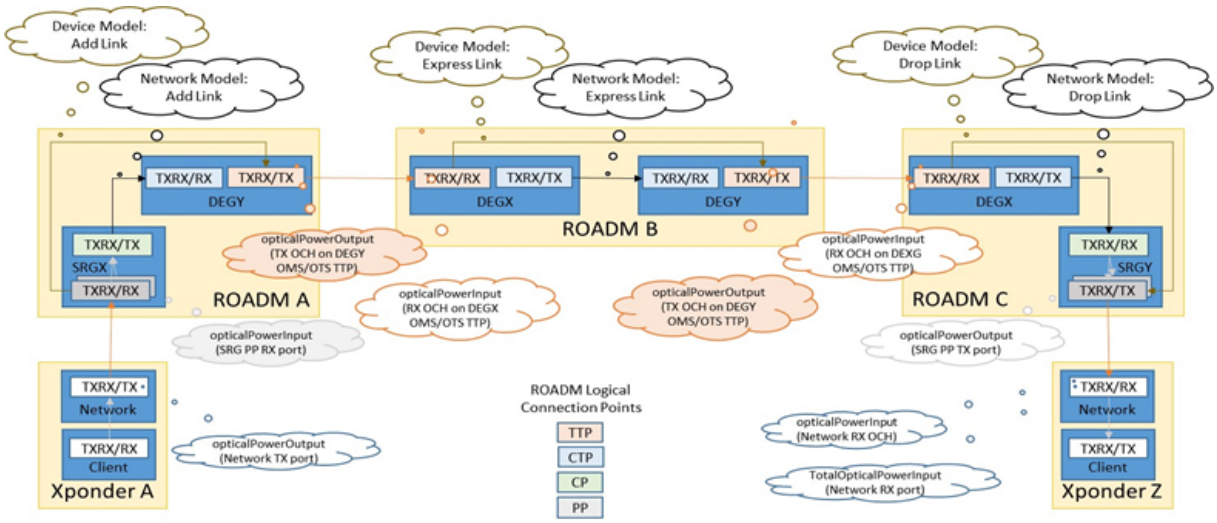

Figure 79: Functional blocks considered by the Open ROADM model

9.1 XPONDER MANAGEMENT – SERVICE CREATION

This chapter collects several use cases for the provisioning of 100G and 400G in Muxponders/Switches and Transponders.

- 9.1.1 describes the use cases:
  - configurations of Transponder
  - configurations of Muxponders/Switchponders
  - connections between Client and Network interfaces
  - configurations of 3R regenerators
- 9.1.3 describes provisioning of client interfaces
  - the actual configuration of client interfaces
- 9.1.4 describes provisioning of network interfaces
  - the actual configuration of network interfaces
- 9.1.5 describes the power configuration of network interfaces

The controller will use the port capabilities exposed by the device to formulate the provisioning commands for the interfaces and the correct sequence of provisioning commands.

9.1.1 Xponder Configuration - Use cases

This section presents several use cases and the sequence of commands needed for their provisioning. This section only describes commands required for the Muxponders, Switches and Transponder.
The actual provisioning of all the interfaces is in the following sections

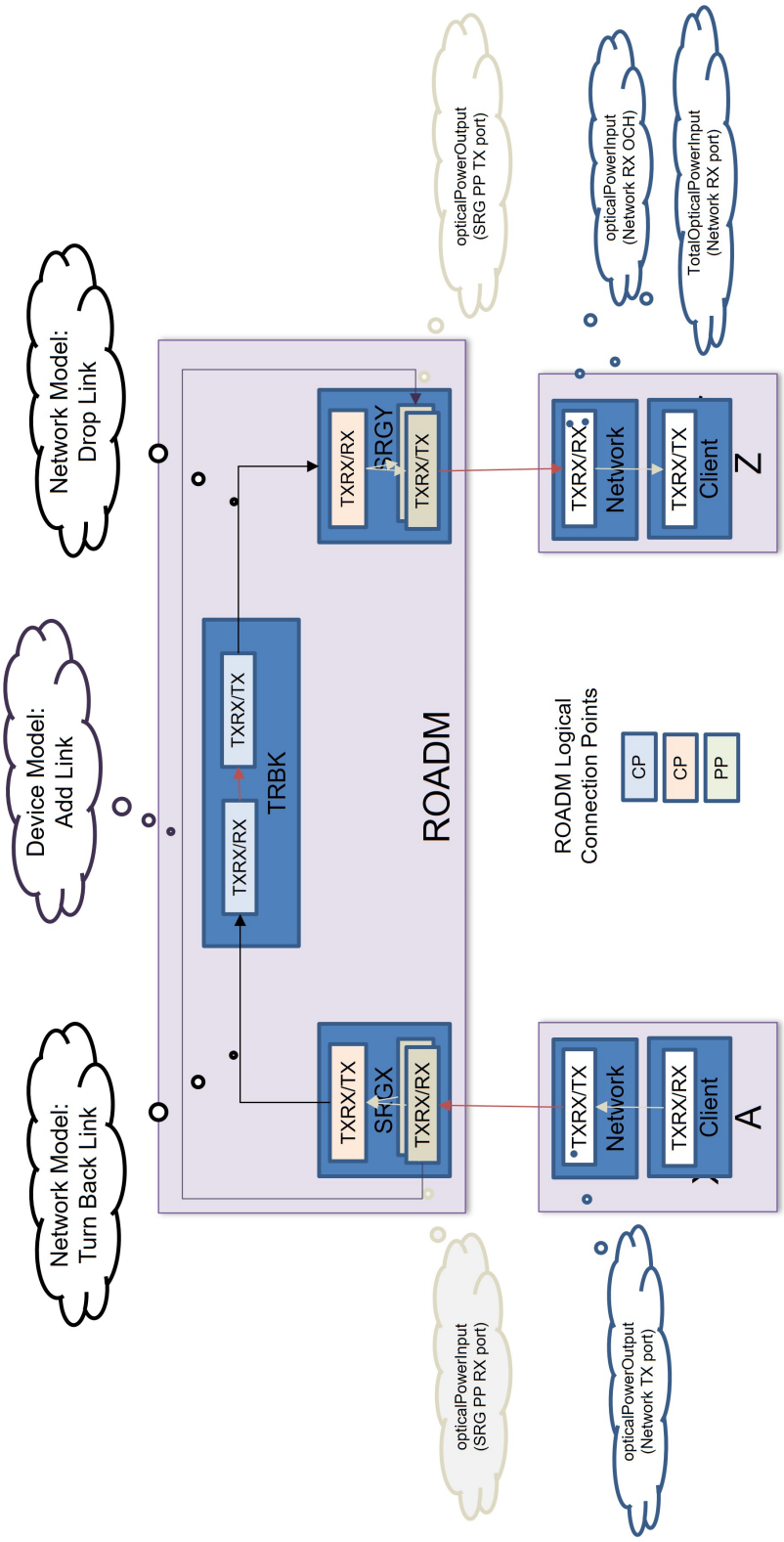

Figure 80: Functional blocks considered by the Open ROADM turn back module

For **Transponders**, the xponder container will be provisioned as “Tpdr”, port qualifiers will be provisioned as **xpdr-network** or **xpdr-client**. The use of Transponders does not need explicit provision of cross connections by the controller: connectivity in a transponder is fixed and it is advertised as **connection map**. Connection maps describes the unidirectional connectivity from client port to network port, and vice versa[7].

For **Muxponders/Switches**, the xponder container will be provisioned as “Mpdr”/“Switch”, port qualifiers will be provisioned as **switch-network** or **switch-client**. The use of Muxponders/Switches requires explicit provision of cross connections between client and network interfaces. Connectivity in a Muxponders/Switches is configurable by the controller and the switch/Muxponder capabilities is advertised as **switching pool**[8]. Connectivity in a Muxponders/Switches may be subject to restrictions exposed by the device in the mpdr-client-restriction.
For **Regenerators**, the xponder container will be provisioned as “Regen” or “Regen-uni”, port qualifiers will be provisioned as **xpdr-network** or **xpdr-client**. The use of Transponders does not need explicit provision of cross connections by the controller: connectivity in a regenerator is fixed and it is advertised as as **connection map**.

9.1.1.1 Use Case 1 - 100GE – Transponder – 100G Wavelength

Transponder do not need explicit provision of cross connections. Once client and network interfaces are correctly provisioned, client traffic will be routed to the network interface.

For Ethernet Traffic provisioned in transponders, OpenROADM MSA models the ODU interface only on the **network** side, not on the **client** side (see also 9.1.3.1 on the provisioning of the 100GE client interfaces). The **Network** side ODU interface will have CTP-TTP function and will be a terminated interface (Monitoring-mode).

This use case requires the following steps[9]:

1. Provision of the network interfaces as described in section 1.1.3.1 (100G Network sides)
2. Provision of the client interface as described in 1.1.2.1 (100GE Client)
3. Optional: The controller can verify that a connection exists between client and network transponder ports:

```
Get-config (org-openroadm-device/connection-map[
      source/circuit-pack-name=xponder_X_network_circuit_pack
  and source/port-name=xponder_X_client_port
  and destination/circuit-pack-name=xponder_X_client_circuit_pack
  and destination/port-name=xponder_X_network_port])
```

9.1.1.2 Use Case 2 - OTU4 – Transponder – 100G Wavelength

Transponder and Regenerators do not need explicit provision of cross connections. Once client and network interfaces are correctly provisioned, client traffic will be routed to the network interface.

For OTU traffic provisioned in transponders, the ODU interface will be modelled on both **network** and **client** side. Both ODU interfaces will have CTP function and will be not-terminated interfaces (Monitoring-mode).

This use case requires the following steps:

1. Provision of the network interfaces as described in section 1.1.3.1 (100G Network sides)
2. Provision of the client interface as described in 1.1.2.2 (OTU4 Client)
3. Optional: The controller can verify that a connection exists between client and network transponder ports:

```
Get-config (org-openroadm-device/connection-map[
      source/circuit-pack-name = xponder_X_network_circuit_pack
  and source/port-name = xponder_X_client_port
  and destination/circuit-pack-name = xponder_X_client_circuit_pack
  and destination/port-name = xponder_X_network_port])
```

[7]The actual trigger for starting to advertise the connection map will depend on the vendor implementation, however, once client and network circuit pack and port are provisioned and the xponder container is created, equipment vendor are expected to advertise transponder’s connection map
[8]Switching Pool will advertise both cp-slot and port, the actual trigger for starting to advertise the Switching Pool will depend on the vendor implementation, however, once client and network circuit pack and port are provisioned, the xponder container is created, and the line side OTSI/opticalChannel is created, equipment vendor are expected to advertise Switching Pool
[9]Depending on Controller implementation, all necessary interfaces can be created in a single or multiple edit-configs

9.1.1.3 Use Case 3 - 100GE or OTU4 – Muxponders/Switches – 100G Wavelength

A single 100GE or OTU4 Interface can be provisioned on a 100G Muxponder/Switch client port to be transported to the Muxponder/Switch network port. Muxponders/Switches need explicit provision of cross connections. Once client and network interfaces are correctly provisioned, an odu-connections between client odu and network odu needs to be created to route client traffic to the network interface.

In this case, independently on the client traffic type, the ODU interface will be modeled on both **network** and **client** side. The ODU interfaces on both sides will have CTP function and will be not-terminated interfaces (Monitoring-mode).

This use case requires the following steps:

1. Provision of the network interfaces as described in section 1.1.3.1 (100G Network sides)
2. Check the Switching Pool
3. Provision of the client interface as described in 1.1.2.1 (100GE Clients) or 1.1.2.2 (OTU4 Clients)
4. Create an odu-connections between client odu and network odu:

```
Edit-config (org-openroadm-device/odu-connection)
  odu-connection
      set connection-name  =  \<Client_odu-Network_odu>"
      set source/src-if to Client_odu
      set destination/dst-if to Network_odu
```

9.1.1.4 Use Case 4 - 400GE – Transponder – 400G Wavelength

Transponder and Regenerators do not need explicit provision of cross connections. Once client and network interfaces are correctly provisioned, client traffic will be routed to the network interface.

For Ethernet Traffic provisioned in transponders, OpenROADM MSA models the ODU interface only on the **network** side, not on the **client** side (see also the section 1.1.2.3 on the provisioning of the 400GE client interfaces). The Network side ODU interface will have CTP-TTP function and will be a terminated interface (Monitoring-mode).

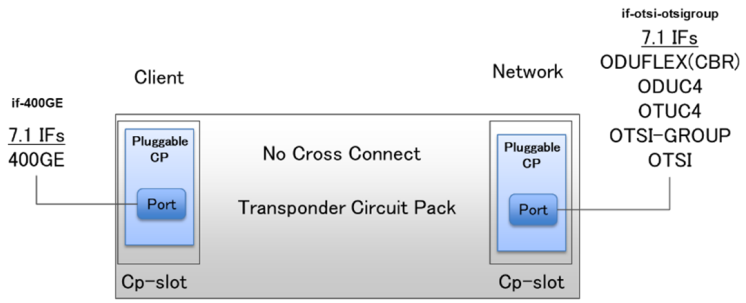

Figure 81: 400GE Transponder Diagram

This use case requires the following steps:

1. Provision of the network interfaces as described in section 1.1.3.2 (400GE Network sides)
2. Check the Switching Pool
3. Provision of the client interface as described in 1.1.2.3 (400GE Clients)
4. Optional: The controller can verify that a connection exists between client and network transponder ports:

```
Get-config (org-openroadm-device/connection-map[
      source/circuit-pack-name = xponder_X_network_circuit_pack
  and source/port-name = xponder_X_client_port
  and destination/circuit-pack-name = xponder_X_client_circuit_pack
  and destination/port-name = xponder_X_network_port])
```

9.1.1.5 Use Case 5 – OTUC4 – Transponder – 400G Wavelength

For future specification.
We plan to document this case in a future release.

9.1.1.6 Use Case 6 – 400GE OR OTUC4 – Muxponders – 400G Wavelength

For future specification[10]
Although this use case is a valid case in the MSA, it is currently not a support priority.

9.1.1.7 Use Case 7 – 4X100GE/OTU4 – Muxponders/Switches - 400G Wavelength

A variable number (1 to 4) of 100GE and/or OTU4 client interfaces can be provisioned on the client ports of a 400G Muxponders/Switches. One client interface for each port.

Independently on the client traffic type, 100GE or OTU4, each of the client tributaries will have an ODU4 interface provisioned (refer to 1.1.2.1 and 1.1.2.2 on the provisioning of the 100GE and OTU4 **client** interfaces).
And an equal number of ODU4 interfaces is provisioned on the network side of the Muxponder/Switch. During the provisioning of the network-side ODU4 interfaces, the Controller also configures 'Parent-odu-allocation' information (i.e., trib-port-number and opucn-trib-slots) to manage the multiplex.

Cross connection between client and network ODU4s will have to be explicitly created..

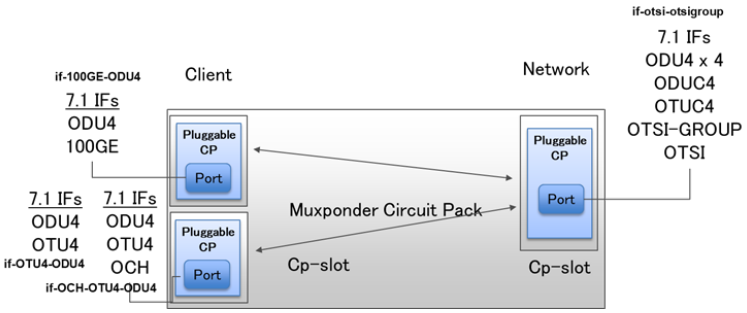

Figure 82: 4x100GE/OTU4 Muxponder Diagram

The **Network** side ODU4 interfaces will have the only CTP function and they will be a not-terminated interfaces (Monitoring-mode). Note that on the **Network** side there will also be the higher order ODU interface (i.e., ODUC4) that will have CTP-TTP function and will be terminated.
The **Client** side ODU interface will have CTP function and Monitoring-mode will depend on the client type refer to 1.1.2.1 and 1.1.2.2.

This use case requires the following steps:

1. Provision of the client interfaces (between 1 and 4) as described in 1.1.2.1 (100GE Clients) / 1.1.2.2 (OTU4 Clients)
2. Check the Switching Pool
3. Provision of the network interfaces as described in section 1.1.3.2 (400G Network Side)
4. For each client ODU that was provisioned (between 1 and 4):
   - create an odu-connection between the client and the network ODUs:

```
Get-config (org-openroadm-device/connection-map[
      source/circuit-pack-name = xponder_X_network_circuit_pack
  and source/port-name = xponder_X_client_port
  and destination/circuit-pack-name = xponder_X_client_circuit_pack
  and destination/port-name = xponder_X_network_port])
  and destination/port-name = xponder_X_network_port])
```

9.1.1.8 Use Case 8 – 4X100GE/OTU4 – Muxponders/Switches – Split Lambda

For future specification
Although this use case is a valid case in the MSA, it is currently not a support priority.

9.1.1.9 Use Case 9 - Configurations of 3R Regenerators

Open ROADM model supports uni-directional and bi-directional regenerators; also, back-to-back transponders are supported, and they can be used as regenerators.

[10] OTUC4 client case is a future development case

Current Priority for this specification is OFEC (OTSI) bidirectional regenerators for 200G and 400G.
200G and 400G regen support will be advertised with two different profiles in the otsigroup-capability-profile.
They will both have the if-OTUCn-ODUCn-regen: if-OTUCn-ODUCn-regen as if-cap-type [11].

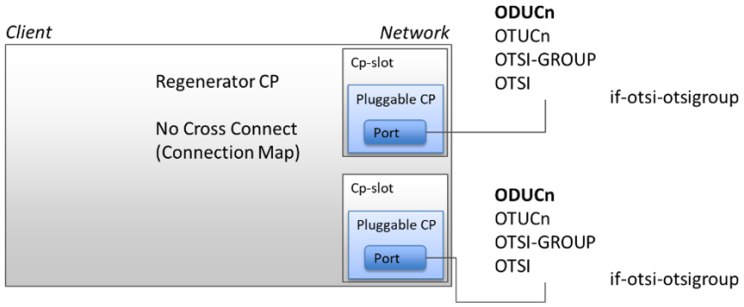

Figure 83: 400G Regen Diagram

Regenerators network ports will use the same optical specification as 100G/200G/300G/400G xponder network ports. The **regenerated layer** is the ODUCN (i.e., ODUC1, ODUC2, ODUC3, ODUC4) and pass thru. OTUCN is terminated and

100G, 300G OFEC regenerators, recoloring (wavelength change), Legacy 100G SCFEC (Och) regenerator and back-to-back transponders used as regenerators are all supported cases in the model but none of is a development priority.

9.1.2 Alien Wavelength

For future specification
We plan to document this case in a future release.

9.1.3 CLIENT-SIDE INTERFACES PROVISIONING

This section describes the sequence of validations and actions required to configure ports and interfaces on the client side of a Xponder.

Each of the client types considered is described in one of the following sub-sections.

9.1.3.1 100GE Clients

For 100GE clients, the 100GE interface is always required, ODU4 interface is required only for client port on Muxponders/Switches.

Input parameters:

- node-id (xponderX)
- xponder client circuit-pack-name[12] (xponderX-client-CP)
- xponder client port-name (xponderX-client-port)

Main actions sequence for the configuration of 100GE client Interfaces:

1. get the status of the port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port])
```

2. get the list of interfaces already present on the client port

```
Get-config (org-openroadm-device/interface [ filtering
      supporting-circuit-pack-name = xponderX-client-CP
  and supporting-port-name = xponderX-client-port])
```

◇ If the required interface(s) (100GE and/or ODU4) are already present on the client port, no creation is required, and the sequence ends successfully.
If the interfaces on the Xponder client port are not present the sequence proceeds to create them - next step below:

[11] MSA 8.1 introduced a new interface capability definition to support OTUCn regenerators: if-OTUCn-ODUCn-regen in the supported-if-capability.
[12] "xponder network/client circuit-pack-name" can indicate a card (integrated optics) or a pluggable, CFP/SFP, (pluggable optics) in OR model both Cards and pluggable are identified as Circuit-pack

3. retrieve the list of interfaces supported by the xponderX-client-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port]
     /port-capability/supported-interface-capability/if-cap-type)
```

◇ To configure a 100GE interface, the port-capabilities/supported-interface-capability/if-cap-type list must contain either "if-100GE" or "if-100GE-ODU4". [13]

4. set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP])
    set equipment-state to not-reserved-inuse
```

5. set administrative-state of the supporting port to inService

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP]
    /ports [ port-name = xponderX-client-port])
    set administrative-state to inService
```

6. create the 100GE Ethernet interface[14] over the supporting client port:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  100GE-supporting-CP-supporting-port
      set type to ethernetCsmacd
      set supporting-circuit-pack-name= xponderX-client-CP
      set supporting-port = xponderX-client-port
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set Ethernet leaf values:
          set speed to 100000
          set fec off
```

◇ During the provisioning of a 100GE in a **Transponder** the sequence completes successfully after the previous step. Only during provisioning of a 100GE in a **Muxponder/Switch**, the sequence proceeds one further step to create an ODU4 interface supported by the 100GE interface[15], this ODU interface is necessary for the ODU-connection that is only required when routing traffic in a Muxponders/Switches[16].

7. If provisioning in a Muxponder or Switch, Create an ODU4 interface on the supporting 100GE interface:

```
Edit-config (org-openroadm-device/interface)
  interface
      set type to otnOdu
      set supporting-circuit-pack-name = xponderX-client-CP
      set supporting-port = xponderX-client-port
      set supporting-interface-list = 100GE-supporting-CP-supporting-port
            (Eth above)
      set administrative-state to inService
      set ODU leaf values
          set rate to ODU4
          set monitor mode to terminated
          set odu-function to ODU-TTP-CTP
          set opu
          payload-type to 0x07
```

[13] Muxponders/Switches require "if-100GE-ODU4", while Transponders require "if-100GE"

[14] Fec in the client 100GE can be either off (for LR4) or rsfec (for SR4)

[15] For MSA after 7, supporting-circuit-pack-name and supporting-port are not required during provisioning. The controller will not be required to provide supporting-circuit-pack-name and port for all interfaces above the first layer (ODU, otsi-group, OTUC4 . . .). The NE will include supporting-circuit-pack-name and port in the logical-port container in the if-otsi-otsigroup port capabilities.

[16] For MSA after 7, the "supporting-interface-list" replaces the "supporting-interface" that was used in the . earlier MSA version

9.1.3.2 OTU4 Clients

Not all vendors support the same model on OTN client interfaces. Some model the Optical channel and some do not.
So, the OCH Interface is provisioned and exposed only if the port support it. The OTU4 interface is always provisioned either over the supporting port or, when present, over a supporting OCH interface. Finally, the ODU4 interface is always provisioned over the supporting OTU interface.

Input parameters:

- node-id (xponderX)
- xponder client circuit-pack-name[17] (xponderX-client-CP)
- xponder client port-name (xponderX-client-port)

Main actions sequence for the configuration of 100GE client Interfaces:

1. get the status of the port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port])
```

2. get the list of interfaces already present on the client port

```
Get-config (org-openroadm-device/interface [ filtering
      supporting-circuit-pack-name = xponderX-client-CP
  and supporting-port-name = xponderX-client-port])
```

◇ If the required interface (OCH, OTU4, and ODU4) are already present on the client port, no creation is required, and the sequence ends successfully.
If the interfaces on the Xponder client port are not present the sequence proceeds to create them - next step below:

3. retrieve the list of interfaces supported by the xponderX-client-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port]
     /port-capability/supported-interface-capability/if-cap-type)
```

◇ To configure a 100GE interface, the port-capabilities/supported-interface-capability/if-cap-type list must contain either “if-OTU4-ODU4” or “if-OCH-OTU4-ODU4”. The controller will provision an och only when the “if-OCH-OTU4-ODU4” capability is exposed. On the other hand, when the “if-OTU4-ODU4” capability is exposed, the OCH interface will not be provisioned. in the latter case, the sequence proceeds to 5 without provisioning an OCh.

4. provision an OCH and OTU4 - only when “if-OCH-OTU4-ODU4” is the IF capability exposed
   (a) set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP])
    set equipment-state to not-reserved-inuse
```

   (b) set administrative-state of the supporting port to inService

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP]
    /ports [ port-name = xponderX-client-port])
    set administrative-state to inService
```

   (c) create the OCH interface over the supporting client port:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name =
```

[17] “xponder network/client circuit-pack-name” can indicate a card (integrated optics) or a pluggable, CFP/SFP, (pluggable optics) in OR model both Cards and pluggable are identified as Circuit-pack

```
        OCH-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
    set type to opticalChannel
    set supporting-circuit-pack-name= xponderX-client-CP
    set supporting-port = xponderX-client-port
    set administrative-state to inService
    set circuit-id (optional planning attribute)
    set OCH leaf values:
        set rate to R100G
```

(d) create an OTU4 interface over the supporting och[18]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name =
          OTU4-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
      set type to otnOtu
      set supporting-circuit-pack-name= xponderX-client-CP
      set supporting-port = xponderX-client-port
      set supporting-interface-list =
          OCH-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
          (och above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set OTU leaf values:
          set rate to OTU4
          set fet to rsfec
```

After creation of the OCH an OTU4, the sequence continues with 6 below

5. provision an OTU4 on the supporting client port - only when “if-OTU4-ODU4” is the IF capability exposed

(a) set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP])
    set equipment-state to not-reserved-inuse
```

(b) set administrative-state of the supporting port to inService

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP]
    /ports [ port-name = xponderX-client-port])
    set administrative-state to inService
```

(c) create an OTU4 interface over the supporting client port

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name =
          OTU4-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
      set type to otnOtu
      set supporting-circuit-pack-name= xponderX-client-CP
      set supporting-port = xponderX-client-port
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set OTU leaf values:
          set rate to OTU4
          set fet to rsfec
```

6. Create an ODU4 interface on the supporting OTU4 interface:

[18] For MSA version after 7, during provisioning of all interfaces supported by other interfaces (in this example, OTU4 and ODU4) the controller can opt to set only the supporting-interface-list leaving supporting-circuit-pack-name and supporting-port not defined. When the controller retrieves the interfaces from the device after provisioning, the device will populate supporting CP and port. For MSA after 7, “supporting-interface-list” replaces the “supporting-interface” that was used in earlier MSA version

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name =
              ODU4-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
      set type to otnOdu
      set supporting-circuit-pack-name = xponderX-client-CP
      set supporting-port = xponderX-client-port
      set supporting-interface-list =
              OTU4-XPDR<n>-CLIENT<m>-supporting-CP-supporting-port
              (OTU4 above)
      set administrative-state to inService
      set ODU leaf values
          set rate to ODU4
          set monitor mode to not-terminated
          set odu-function to ODU-CTP
```

9.1.3.3 400GE Client

Input parameters:

- node-id (xponderX)
- xponder client circuit-pack-name[19] (xponderX-client-CP)
- xponder client port-name (xponderX-client-port)

Main actions sequence for the configuration of 400GE client Interfaces:

1. get the status of the port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port])
```

2. get the list of interfaces already present on the client port

```
Get-config (org-openroadm-device/interface [ filtering
      supporting-circuit-pack-name = xponderX-client-CP
  and supporting-port-name = xponderX-client-port])
```

◇ If the 400GE interface is already present on the client port, no creation is required, and the sequence ends successfully. If the interfaces on the Xponder client port are not present the sequence proceeds to create them - next step below:

3. retrieve the list of interfaces supported by the xponderX-client-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-client-CP]
     /ports [ port-name = xponderX-client-port]
     /port-capability/supported-interface-capability/if-cap-type)
```

◇ To configure a 400GE interface, the port-capabilities/supported-interface-capability/if-cap-type list must contain either “if-400GE” or “if-400GE-oduflexcbr”. [20][21]

4. set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP])
    set equipment-state to not-reserved-inuse
```

5. set administrative-state of the supporting port to inService

[19] “xponder network/client circuit-pack-name” can indicate a card (integrated optics) or a pluggable, CFP/SFP, (pluggable optics) in OR model both Cards and pluggable are identified as Circuit-pack

[20] When the pluggable CP is provisioned as QSFP56 or DD, the 400GE is the only capability advertised

[21] that Muxponders/Switches provisioning requires “if-400GE-oduflexcbr”, while Transponder provisioning requires “if-400GE”

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-client-CP]
    /ports [ port-name = xponderX-client-port])
    set administrative-state to inService
```

6. create the 400GE Ethernet interface over the supporting client port:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  400GE-supporting-CP-supporting-port
      set type to ethernetCsmacd
      set supporting-circuit-pack-name= xponderX-client-CP
      set supporting-port = xponderX-client-port
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set Ethernet leaf values:
          set speed to 400000
          set fec to rsfec
```

♢ During the provisioning of a 100GE in a **Transponder** the sequence completes successfully after the previous step. Provisioning of a 400GE in a Muxponders/Switches is left for future specification.

9.1.3.4 OTUCN Client

For future specification
Although this use case is a valid case in the MSA, it is currently not a support priority.

9.1.4 NETWORK-SIDE INTERFACES PROVISIONING

This section describes the sequence of validations and actions required to configure ports and interfaces on the network side of a Xponder.

Note that Xponder interfaces are supposed to be bi-directional. Each of the modulation format considered is described in one of the following sub-section.

9.1.4.1 100G LEGACY (STAIRCASE) NETWORK SIDE

Input parameters:

- node-id (xponderX)
- xponder network circuit-pack-name[22] (xponderX-network-CP)
- xponder network port-name (xponderX-network-port)
- channel central frequency
- transmit power
- modulation format (100G W in this case)

Main actions sequence for the configuration of 100GE network Interfaces:

1. get the status of the port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port])
```

2. get the list of interfaces already present on the network port

```
Get-config (org-openroadm-device/interface [ filtering
      supporting-circuit-pack-name = xponderX-network-CP
  and supporting-port-name = xponderX-network-port])
```

[22] "xponder network/network circuit-pack-name" can indicate a card (integrated optics) or a pluggable, CFP/SFP, (pluggable optics) in OR model both Cards and pluggable are identified as Circuit-pack

◊ If the OCH is already present on the network port, the sequence will fail. if no Och existis, the sequence proceeds to create the necessary interfaces.

3. retrieve the list of interfaces supported by the xponderX-network-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /port-capability/supported-interface-capability/if-cap-type)
```

4. retrieve the power port capabilities of the xponderX-network-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /transponder-port/port-power-capability-min-tx)
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /transponder-port/port-power-capability-max-tx)
```

◊ To configure a legacy 100G W (staircase), the port-capabilities/supported-interface-capability/if-cap-type list must contain “if-OCH-OTU4-ODU4”.

5. set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-network-CP])
    set equipment-state to not-reserved-inuse
```

6. set administrative-state of the supporting port to inService

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-network-CP]
    /ports [ port-name = xponderX-network-port])
    set administrative-state to inService
```

7. create the OCH interface over the supporting network port[23]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  OCH-XPDR<n>-network<m>-frequency
      set type to opticalChannel
      set supporting-circuit-pack-name= xponderX-network-CP
      set supporting-port = xponderX-network-port
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set OCH leaf values:
          set rate to R100G
          set frequency
          set modulation-format (QPSK )
          set FEC (Staircase)
```

8. create an OTU4 interface over the supporting och interface[24]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  OTU4-XPDR<n>-network<m>
      set type to otnOtu
      set supporting-circuit-pack-name= xponderX-network-CP
      set supporting-port = xponderX-network-port
      set supporting-interface-list =
```

[23] Xponder output power can be set here or later on (section 1.1.4)

[24] supporting-circuit-pack-name and supporting-port are not required during provisioning for MSA ¿ 7
For MSA ¿ 7, we have “supporting-interface-list”. Earlier MSA version have instead a “supporting-interface”

```
                OCH-XPDR<n>-network<m>-frequency (och above)
        set administrative-state to inService
        set circuit-id (optional planning attribute)
```

9. Create an ODU4 interface on the supporting OTU4 interface:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  ODU4-XPDR<n>-network<m>
      set type to otnOdu
      set supporting-circuit-pack-name = xponderX-network-CP
      set supporting-port = xponderX-network-port
      set supporting-interface-list =
            OTU4-XPDR<n>-network<m>-supporting-CP-supporting-port
            (OTU4 above)
      set administrative-state to inService
```

9.1.4.2 100G Enhanced (OFEC) Network Side

For future specification We plan to document this case in a future release.

9.1.4.3 400G Network Side

Input parameters:

- node-id (xponderX)
- xponder network circuit-pack-name (xponderX-network-CP)
- xponder network port-name (xponderX-network-port)
- channel central frequency
- transmit power
- modulation format (400G W in this case)

Service provisioning steps for a 400G carrier wavelength are mostly not affected by the slient (400GE WDM Service vs. 4x100G) The oduflex interface will not be provisioned over a Muxponders/Switches Network port.

400G network will use the otsi-if, not och-if.
Network side Interfaces provisioned for a Muxponder (to transport 100G tributaries) are:

- otsi, otsi-group, otucn, oducn, odu4[25]

Network side Interfaces provisioned for a Transponder (to transport a 400G tributary) are:

- otsi, otsi-group, otucn, oducn, oduflex

Main actions sequence for the configuration of 100GE network Interfaces:

1. get the status of the port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port])
```

2. get the list of interfaces already present on the network port

```
Get-config (org-openroadm-device/interface [ filtering
      supporting-circuit-pack-name = xponderX-network-CP
  and supporting-port-name = xponderX-network-port])
```

◊ If the OTSI is already present on the network port, the sequence will fail. if no OTSI existis, the sequence proceeds to create the necessary interfaces.

[25] one ODU4 for each 100G tributary

3. retrieve the list of interfaces supported by the xponderX-network-port

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /port-capability/supported-interface-capability/if-cap-type)
\item retrieve the power capabilities of  the xponderX-network-port
{ \color{blue}
\begin{verbatim}
  Get-config (org-openroadm-device
       /circuit-packs [ circuit-pack-name = xponderX-network-CP]
       /ports [ port-name = xponderX-network-port]
       /transponder-port/port-power-capability-min-tx)
  Get-config (org-openroadm-device
       /circuit-packs [ circuit-pack-name = xponderX-network-CP]
       /ports [ port-name = xponderX-network-port]
       /transponder-port/port-power-capability-max-tx)
```

4. retrieve the mc-capability-profile-name of the xponderX-network-port and gather the min/max edge frequency of that profile

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /mc-capability-profile-name)
Get-config (org-openroadm-device/mc-capability-profile [profile-name]
     /min-edge-freq
Get-config (org-openroadm-device/mc-capability-profile [profile-name]
     /max-edge-freq
```

5. retrieve the list of otsigroup-capability-profile supported by the xponderX-network-port and gather the information

```
Get-config (org-openroadm-device
     /circuit-packs [ circuit-pack-name = xponderX-network-CP]
     /ports [ port-name = xponderX-network-port]
     /port-capability/supported-interface-capability/otsigroup-capability-profile-name)
Get-config (org-openroadm-device/otsigroup-capability-profile [profile-name])
```

$\diamondsuit$ The controller will confirm that the capabilities reported for the supporting port match the requirement for the channel. To configure a legacy 400G W, the port-capabilities/supported-interface-capability need to have the “if-otsi-otsigroup” element in the if-cap-type list *and* an otsigroup-capability-profile that respect two conditions: the otucn-n-rate equal to 4 and the otn-odu-mux-hierarchy-profile should report in the “supported-lo-odu-type” the “ODUflex-cbr” for 400G client and/or ”ODU4” for 100G clients.
If the xponderX-network-port supports all the capabilities required by the 400G application, the sequence proceeds to next step below.

6. set equipment-state of the supporting circuit-pack to not-reserved-inuse

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-network-CP])
    set equipment-state to not-reserved-inuse
```

7. set administrative-state of the supporting port to inService

```
Edit-config (org-openroadm-device
    /circuit-packs[circuit-pack-name = xponderX-network-CP]
    /ports [ port-name = xponderX-network-port])
    set administrative-state to inService
```

8. create the OTSI interface over the supporting network port[26][27]:

[26] Depending on Controller and NEs implementation, the necessary interfaces can be created in a single edit-config
[27] provision-mode to ‘explicit’ is used to provision open ROADM standard mode; provision-mode to ‘profile’ is used to provision the bookended mode. And, optical-operational-mode is required only when provisioning using the provision-mode = “profile”

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  OTSI-XPDR<n>-network<m>-frequency
      set type to otsi
      set supporting-circuit-pack-name= xponderX-network-CP
      set supporting-port = xponderX-network-port
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set OTSI leaf values:
          set provision-mode (\profile" or \explicit")
          set optical-operational-mode (optional)
          set otsi-rate to R400G-otsi
          set frequency
          set transmit-power
          set modulation-format (dp-qam16)
          set FEC (OFEC)
          set FlexO leaf values
              set foic-type \foic4.8"
              set iid [1..4]
```

9. create an OTSI-group interface over the supporting OTSI interface[28]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  OTSIG-XPDR<n>-network<m>
      set type to otsi-group
      set supporting-interface-list =
              OTSI-XPDR<n>-network<m>-frequency (OTSI above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set otsi-group leaf values
          set group-rate (R400G-otsi)
          set group-id (1)
```

10. Create an OTUCn (n=4) interface on the supporting OTSIG interface:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  OTUC4-XPDR<n>-network<m>
      set type to otnOdu
      set supporting-interface-list =
              OTSIG-XPDR<n>-network<m> (OTSIG above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set OTUCN leaf values
          set rate to OTUCn
          set oducn-n-rate to 4
```

11. Create an ODUCn (n=4) interface on the supporting OTUCn interface:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  ODUC4-XPDR<n>-network<m>
      set type to otnOdu
      set supporting-interface-list =
              OTSIG-XPDR<n>-network<m> (OTSIG above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set ODUCN leaf values
          set rate to ODUCn
          set oducn-n-rate to 4
          set monitor-mode to terminated
```

[28] Controller will define the group-id for the otsi-group interface. In this example we use “1”.

```
        set odu-function to ODU-TTP
        set OPU leaf values
            set opu payload-type to 22
```

◊ Network side ODU layer creation will depend on the application:
To transport a single 400G tributary in a transponder, the controller will create, on Network side, a single ODUFlex interface on top of the ODUCn.
To Transport 100G tributaries in a Muxponder, the controller will create, on Network side, ODU4 on top of the ODUCn. One ODU4 for each tributary.

12. Create a lo-ODU interface on the supporting ODUCn interface:

    (a) Create an ODUFlex interface on the supporting ODUCn interface (Transponder use case)[29]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  oduFlex-XPDR<n>-network<m>
      set type to otnOdu
      set supporting-interface-list =
              ODUC4-XPDR<n>-network<m> (ODUC4 above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set ODU leaf values
          set rate to ODUflex-cbr
          set oduflex-cbr-service to ODUflex-cbr-400G
          set monitor-mode to terminated
          set odu-function to ODU-TTP-CTP
          set OPU leaf values
              set opu payload-type to 32
          set parent-odu-allocation leaf values
              set trib-port-number to 1
              set opucn-trib-slots  [1.1 .. 1.20], [2.1 .. 2.20],
                                    [3.1 .. 3.20], [4.1 .. 4.20]
```

    (b) Create ODU4 interfaces on the supporting ODUCn interface (Muxponder)[30][31]:

```
Edit-config (org-openroadm-device/interface)
  interface
      set interface-name  =  odu4-XPDR<n>-network<m>.x (x = 1..4)
      set type to otnOdu
      set supporting-interface-list =
              ODUC4-XPDR<n>-network<m> (ODUC4 above)
      set administrative-state to inService
      set circuit-id (optional planning attribute)
      set ODU leaf values
          set rate to ODU4
          set monitor-mode to not-terminated
          set odu-function to ODU-CTP
          set parent-odu-allocation leaf values
              set trib-port-number to y (y = 1..4)
              set opucn-trib-slots  [1.1 .. 1.20]
```

9.1.4.4 Bookended Mode

Improved performance can be achieved if vendors at both end points of a service deploy optical module that support the same operational mode with better optical specification than the default OpenROADM W interfaces.

[29] The total count of opucn-trib-slots for the ODUflex-cbr interface will be 80 TS, 4 channels, 20 TS each

[30] You need as many ODU4 as the number of tributary you are ready to provision (max 4).
Each of these lower order odu interfaces on the supporting oduc will have a unique names; the naming convention in this example will use unique extension for each lower order odu, e.g., ".1".

[31] The controller identifies the parent-odu-allocation information (both trib-port-number and opucn-trib-slots) for the network ODU4 interface from the mpdr-client-restriction (which also includes muxp-profile) defined for the if-cap-type if-100GE-ODU4 on the client pluggable-optics-holder-capability.
The total count of opucn-trib-slots will be 20 for each of the ODU4interface (each slot is 5GHz)

At the ends of the service you can have equipment from the same vendor or form different vendors; however, the bookend mode implies that both ends of a service will use the same Modulation format and compatible Optical module. Both ends of the service will support the same bookend operational mode. The bookend mode would be available for all xponder types: transponder, muxponders, and bi-directional regenerators.
The same module could be able to support the Open ROADM standard mode, and/or non-standard operational modes (bookended).
It is expected that the hardware provisioning will be independent on the operational mode: circuit-pack-type, circuit-pack-product-code, circuit-pack-mode and, port-type will not have to change to deploy one of the available operational mode: only the OTSI provisioning will be changed.

Supported operating modes are advertised under pluggable optic holders and port capabilities. Vendors should advertise the optical modes that are supported by the device and are provisionable via the profile provision-mode. Bookend mode will be advertised using their defined profile-name in the optical-operational-mode-profile list in the port-capabilities. Standard 400G W interface is currently not expected to be advertised in the optical-operational-mode-profile list; we continue to provision standard 400G as explicit provision-mode.

If a module is provisioned with an unsupported operational mode, the NE should alarm the module.

A Bookend mode is provisioned by setting the provision-mode of the OTSI to profile (Profile based provisioning) and then provisioing a specific optical-operational-mode. Provisioning of the other interfaces (OTSIG and above) is not affected by bookended mode.

9.1.4.5 400G Regen Interface

For future specification
We plan to document this case in a future release.

9.1.4.6 200G 16QAM

For future specification.
We plan to document this case in a future release.

9.1.4.7 300G 8QAM

For future specification.
We plan to document this case in a future release.

9.1.5 Xponder Power Setting

Input parameters:

- node-id (xponderX)
- xponder network circuit-pack-name (xponderX-network-CP)
- xponder network port-name (xponderX-network-port)
- target-transmit-power

Main actions sequence for the configuration of 100GE network Interfaces:

1. The controller configure the target output power of the OTSI interface
   (a) for OFEC channel (OTSI)

```
Edit-config (org-openroadm-device
    /interface[name= OTSI-XPDR<n>-network<m>-frequency]
    /otsi)
    set transmit-power to the target-transmit-power
```

   (b) for staircase channel (Och)

```
Edit-config (org-openroadm-device
    /interface[name= OCH-XPDR<n>-network<m>-frequency]
    /och)
    set transmit-power to the target-transmit-power
```

2. The controller verifies that the output power reaches the set value

```
Get-config (currentPmlist/currentPm
      [resource/port-name = xponderX-network-port
  and granularity="15min"]
      /pmParameterName/type = opticalPowerOutput)
```

9.1.6 BER Test

For future specification.
We plan to document this case in a future release.

9.2 ROADM MANAGEMENT – SERVICE CREATION

Once the Xponder interfaces and connections are configured, routing traffic to its destination requires creation of cross-connections on the ROADM across the routing path. We use ROADM-connections for this.

A ROADM-connection between the tributary port in the SRG and the Network port in the direction of the source ROADM will provide the add link in the first ROADM.
ROADM-connection in the intermediate ROADM will be used to provide the express link from one degree to another.

Finally, the drop link in the destination ROADM will be provided by ROADM-connection in the last ROADM. Refer to Figure 79 in the beginning of this chapter. A ROADM-connection between two tributary ports on SRGs from the same device will enable the turn back function in a ROADM. Refer to Figure 80.

A ROADM-connection is always created only between two nmc-ctp.
The nmc-ctp interface on the SRG side is the child of a supporting Wr port.
The nmc-ctp interface on the degree side is the only child of a mc-ttp interface[32].

MC interfaces are provisioned in terms of their min-freq and max-freq, NMC interfaces are provisioned in terms of their central frequency and bandwidth. The controller will calculate and configure all necessary frequency values[33].

As MSA agreed to currently support only the case of one nmc per mc, nmc-ctp and mc-ttp interfaces have the same center frequency and the NMC width is always fully contained inside the MC width:

- For 100G channels, MC width will be set as of 50 GHz and the NMC width is 40 GHz
- For 400G channels, MC width will be set as of 87.5 GHz and the NMC width is 75 GHz

9.2.1 Add and Drop Link Creation

This section presents the sequence of validations and commands needed for the provisioning of ports, interfaces (i.e., MC and NMC) and ROADM-connection of the add link from an SRG-PP to Degree TTP logical points and of the drop link in the opposite direction.
We consider in the examples applying power settings just after the creation of interfaces/connections, but, depending on the controller implementation, the power configuration could also be performed in a second step.

Input parameters:

- node-id (xponderX)
- SRG/Deg numbers (SRGX, DEGY)
- circuit pack (SRGX-Circuit-Pack, DEGY-Circuit-Pack)
- supporting port (SRGX-PPN, DEGY-TTP)
- min-freq and max-freq for the mc-ttp interface
- central-frequency and width for the nmc-ctp interface
- calculated output power

9.2.1.1 Provision of NMC on SRG Side

Main actions sequence:

1. get the interfaces on the SRGX-PPN supporting port

[32] An mc interface can generally support several nmc interfaces, however we agreed to currently limit MSA operations to the case of one nmc per mc
[33] The controller will verify required power and frequency versus the transmission power and transmission frequency range as exposed by the mc-capabilities published by the device for both the degree and the SRG

```
Get-config (org-openroadm-device
    /Circuit-pack [ circuit-pack-name = SRGX_Circuit_Pack]
    / ports [port-name = SRGX-PPN ]
    /interfaces)
```

◇ Check if nmc-ctp interface is present on SRGX-PPN supported port
If an nmc-ttp (bi-directional) interface is already present on the SRGX-PPN supporting-port this sequence will end successfully.
Otherwise, the sequence will create the required nmc-ttp

2. Set admin state of nmc-ctp interface supporting PP port to inService

```
Edit-config (org-openroadm-device
    /circuit-pack [ circuit-pack-name = SRGX_Circuit_Pack]
    /ports [port-name = SRGX-PPN ]
    set the administrative-state to 'inService'
```

3. create a nmc-ctp interface on the supporting PP port of the SRGX-PPN

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = nmc-SRGX-supporting_CP/supporting_port
      set type to networkMediaChannelConnectionTerminationPoint
      set supporting-circuit-pack-name= SRGX_Circuit_Pack
      set supporting-port = SRGX-PPN
      set admin state of nmc-ctp interface inService
      set circuit-id (optional planning attribute)
      set nmc-ctp leaf values:
          set frequency
          set width
```

4. The following step is optional. It aims at verifying that input power is compliant with MSA specifications:

```
Get-config (currentPmlist/currentPm
      [resource/interface-name = nmc-ctp-DEGX-TTP-frequency
      and granularity="15min"]
      /pmParameterName/type = opticalPowerInput)
```

◇ Check if power reading from the PM (pmParameterName/type = opticalPowerInput) is withing reasonable distance from the the expected input power derived from MSA specifications on line port channel output power reduced of the exppected patch cord loss.

9.2.1.2 Provision of the NMC on DEG Side

Main actions sequence:

1. get the interfaces on the DEGY-TTP supporting port

```
Get-config (org-openroadm-device
    /Circuit-pack [ circuit-pack-name = DEGY_Circuit_Pack]
    /ports [port-name = DEGY-TTP ]
    /interfaces)
```

◇ Check if OTS interface is present on DEGY-TTP supported port.
If there is no OTS (bi-directional) interface on the DEGY-TTP port, the sequence will create it.

2. create the OTS interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = OTS-DEGY-TTP
      set type to opticalTransport
      set supporting-circuit-pack-name= DEGY_Circuit_Pack
      set supporting-port = DEGY-TTP
      set admin state of nmc-ctp interface inService
```

◇ Check if OMS interface is present on DEGY-TTP supported port.
If there is no OMS (bi-directional) interface on the DEGY-TTP port, the sequence will create it on top of the supporting the OTS interface.

3. create the OMS interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = OMS-DEGY-TTP
      set type to openROADMOpticalMulitplex
      set supporting-circuit-pack-name= DEGY_Circuit_Pack
      set supporting-port = DEGY-TTP
      set supporting-interfaces-list = OTS-DEGY-TTP
      set admin state of nmc-ctp interface inService
```

◇ Check if mc-ctp interface is present on DEGY-TTP supported port.
If there is no mc-ttp (bi-directional) interface on the DEGY-TTP port, the sequence will create it on top of the supporting the OMS interface

4. create the MC-TTP interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = mc-ttp-DEGY-TTP-frequency
      set type to mediaChannelTrailTerminationPoint
      set supporting-circuit-pack-name= DEGY_Circuit_Pack
      set supporting-port = DEGY-TTP
      set supporting-interfaces-list = OTS-DEGY-TTP
      set admin state of mc-ctp interface inService
      set min-freq
      set max-freq
```

◇ Check if nmc-ctp interface is present on DEGY-TTP supported port.
If an nmc-ttp (bi-directional) interface is already present on the DEGY-TTP supporting-port this sequence will end successfully.
Otherwise, the sequence will create the required nmc-ttp.

5. create the NMC-TTP interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = NMC-DEGY-TTP-frequency
      set type to networkMediaChannelConnectionTerminationPoint
      set supporting-circuit-pack-name= DEGY_Circuit_Pack
      set supporting-port = DEGY-TTP
      set supporting-interfaces-list = mc-ttp-DEGY-TTP-frequency
      set admin state of nmc-ctp interface inService
      set frequency
      set width
```

6. The following step is optional and aims at verifying that input power is compliant with MSA specifications (verify pmParameterName/type = opticalPowerInput

```
Get-config (currentPmlist/currentPm
      [resource/interface-name = nmc-ctp-DEGX-TTP-frequency
      and granularity="15min"]
      /pmParameterName/type = opticalPowerInput)
```

◇ Check if power reading from the PM (pmParameterName/type = opticalPowerInput) vs. the expected input power derived from MSA specifications : per channel output power of preceding span (per channel output power offset diagram)- span loss

9.2.1.3 Provision of the Add ROADM Connection

Main actions sequence:

1. create a roadm-connection between the nmc-ctp interfaces

```
Edit-config (org-openroadm-device/roadm-connections)
  roadm-connections
      set name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set source/src-if = SRGX-PPN-frequency
      set destination/dst-if = nmc-ctp-DEGY-TTP-frequency
```

2. Power setting and control-mode change[34]

```
Edit-config (org-openroadm-device/roadm-connections
  [ connections-name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set optical control-mode = power
      set target-power = calcualted-target-power
```

3. in a final step, the controller verifies that the output power reaches the set value

```
  Get-config (currentPmlist/currentPm
        [resource/interface-name = nmc-ctp-DEGY-TTP-frequency
        and granularity="15min"]
        /pmParameterName/type = opticalPowerOutput)
```

◇ Check if power reading from the PM (pmParameterName/type = opticalPowerOutput) vs. the calculated output power provisioned in the previous step.
Once output power reading from the PM has converged towards the expected output power, control-mode is changed to gainLoss.

4. control-mode changes to gainLoss

```
Edit-config (org-openroadm-device/roadm-connections
  [ connections-name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set optical control-mode = gainLoss
```

9.2.1.4 Provision of Drop ROADM Connection

Main actions sequence:

1. create a roadm-connection between the nmc-ctp interfaces

```
Edit-config (org-openroadm-device/roadm-connections)
  roadm-connections
      set name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set source/src-if = nmc-ctp-DEGY-TTP-frequency
      set destination/dst-if = SRGX-PPN-frequency
```

2. Power setting and control-mode change

```
Edit-config (org-openroadm-device/roadm-connections
  [ connections-name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set name = SRGX-PPN-[frequency]-to-nmc-ctp-DEGY-TTP
      set optical control-mode = power
```

9.2.2 Express Link Creation

This section presents the sequence of validations and commands needed for the provisioning of ports, interfaces, and ROADM-connection of the express link from a Degree-TTP to a Degree TTP logical points. rodam-connections are unidirectional, the provisioning of the express link requires two of them in opposite direction.

[34] The output power is calculated from MSA specifications: per channel output power offset diagram

We consider in the examples applying power settings just after the creation of interfaces/connections, but, depending on the controller implementation, the power configuration could also be performed in a second step.

Input parameters:

- node-id (xponderX)
- DEG/Deg numbers (DEGX, DEGY)
- circuit pack (DEGX-Circuit-Pack, DEGY-Circuit-Pack)
- supporting port (DEGX-PPN, DEGY-TTP)
- min-freq and max-freq for the mc-ttp interface
- central-frequency and width for the nmc-ctp interface
- expected input power (on both ends)
- calculated target power (on both ends)

9.2.2.1 Provision of the NMC on Both DEG Sides

Main actions sequence[35]:

1. get the interfaces on the DEGX-PPN (DEGX-PPN) supporting port

```
Get-config (org-openroadm-device
    /circuit-pack [ circuit-pack-name = DEGX_Circuit_Pack]
    /ports [port-name = DEGX-PPN ]
    /interfaces)
```

◇ Check if OTS interface is present on DEGX-TTP supported port.
If there is no OTS (bi-directional) interface on the DEGX-TTP port, the sequence will create it.

2. create the OTS interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = OTS-DEGX-TTP
      set type to opticalTransport
      set supporting-circuit-pack-name= DEGX_Circuit_Pack
      set supporting-port = DEGX-TTP
      set admin state to inService
```

◇ Check if OMS interface is present on DEGX-TTP supported port.
If there is no OMS (bi-directional) interface on the DEGX-TTP port, the sequence will create it on top of the supporting the OTS interface.

3. create the OMS interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = OMS-DEGX-TTP
      set type to openROADMOpticalMulitplex
      set supporting-circuit-pack-name= DEGX_Circuit_Pack
      set supporting-port = DEGX-TTP
      set supporting-interfaces-list = OTS-DEGX-TTP
      set admin state to inService
```

◇ Check if mc-ctp interface is present on DEGX-TTP supported port.
If there is no mc-ttp (bi-directional) interface on the DEGX-TTP port, the sequence will create it on top of the supporting the OMS interface

4. create the MC-TTP interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = mc-ttp-DEGX-TTP-frequency
```

[35] This sequence has to be run twice, once for each of the two Deg

```
set type to mediaChannelTrailTerminationPoint
set supporting-circuit-pack-name= DEGX_Circuit_Pack
set supporting-port = DEGX-TTP
set supporting-interfaces-list = OMS-DEGX-TTP
set admin state to inService
set min-freq
set max-freq
```

◇ Check if nmc-ctp interface is present on DEGX-TTP supported port.
If an nmc-ttp (bi-directional) interface is already present on the DEGX-TTP supporting-port this sequence will end successfully.
Otherwise, the sequence will create the required nmc-ttp.

5. create the NMC-TTP interface

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = NMC-DEGX-TTP-frequency
      set type to networkMediaChannelConnectionTerminationPoint
      set supporting-circuit-pack-name= DEGX_Circuit_Pack
      set supporting-port = DEGX-TTP
      set supporting-interfaces-list = mc-ttp-DEGX-TTP-frequency
      set admin state of nmc-ctp interface inService
      set frequency
      set width
```

6. The following step is optional and aims at verifying that input power is compliant with MSA specifications (verify pmParameterName/type = opticalPowerInput

```
Get-config (currentPmlist/currentPm
      [resource/interface-name = nmc-ctp-DEGX-TTP-frequency
      and granularity="15min"]
      /pmParameterName/type = opticalPowerInput)
```

◇ Check if power reading from the PM (pmParameterName/type = opticalPowerInput) vs. the expected input power derived from MSA specifications : per channel output power of preceding span (per channel output power offset diagram)- span loss

9.2.3 Turn Back Link Creation

This section presents the sequence of validations and commands needed for the provisioning of ports, interfaces (i.e., NMC) and ROADM-connection of the turn back link from an SRG-PP to SRG-PP logical points. Turn back links are provisioned in both directions.

Input parameters:

- node-id (xponderX)
- SRG numbers (SRGX, SRGY)
- circuit pack (SRGX-Circuit-Pack, SRGY-Circuit-Pack)
- supporting port (SRGX-PPN, SRGY-PPN)
- central-frequency and width for the nmc-ctp interface

9.2.3.1 Provision of NMC on SRG X Side

Main actions sequence:

1. get the interfaces on the SRGX-PPN supporting port

```
Get-config (org-openroadm-device
    /Circuit-pack [ circuit-pack-name = SRGX_Circuit_Pack]
    / ports [port-name = SRGX-PPN ]
    /interfaces)
```

◇ Check if nmc-ctp interface is present on SRGX-PPN supported port
If an nmc-ttp (bi-directional) interface is already present on the SRGX-PPN supporting-port this sequence will end successfully.
Otherwise, the sequence will create the required nmc-ttp

2. Set admin state of nmc-ctp interface supporting PP port to inService

```
Edit-config (org-openroadm-device
    /circuit-pack [ circuit-pack-name = SRGX_Circuit_Pack]
    /ports [port-name = SRGX-PPN ]
    set the administrative-state to 'inService'
```

3. create a nmc-ctp interface on the supporting PP port of the SRGX-PPN

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = nmc-SRGX-PPN-frequency
      set type to networkMediaChannelConnectionTerminationPoint
      set supporting-circuit-pack-name= SRGX_Circuit_Pack
      set supporting-port = SRGX-PPN
      set admin state of nmc-ctp interface inService
      set circuit-id (optional planning attribute)
      set nmc-ctp leaf values:
          set frequency
          set width
```

4. The following step is optional. It aims at verifying that input power is compliant with MSA specifications:

```
Get-config (currentPmlist/currentPm
      [resource/interface-name = nmc-SRGX-PPN-frequency
      and granularity="15min"]
      /pmParameterName/type = opticalPowerInput)
```

◇ Check if power reading from the PM (pmParameterName/type = opticalPowerInput) is withing reasonable distance from the the expected input power derived from MSA specifications on line port channel output power reduced of the expected patch cord loss.

9.2.3.2 Provision of NMC on SRG Y Side

Main actions sequence:

1. get the interfaces on the SRGY-PPN supporting port

```
Get-config (org-openroadm-device
    /Circuit-pack [ circuit-pack-name = SRGY_Circuit_Pack]
    / ports [port-name = SRGY-PPN ]
    /interfaces)
```

◇ Check if nmc-ctp interface is present on SRGY-PPN supported port
If an nmc-ttp (bi-directional) interface is already present on the SRGY-PPN supporting-port this sequence will end successfully.
Otherwise, the sequence will create the required nmc-ttp

2. Set admin state of nmc-ctp interface supporting PP port to inService

```
Edit-config (org-openroadm-device
    /circuit-pack [ circuit-pack-name = SRGY_Circuit_Pack]
    /ports [port-name = SRGY-PPN ]
    set the administrative-state to 'inService'
```

3. create a nmc-ctp interface on the supporting PP port of the SRGY-PPN

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = nmc-SRGY-PPN-frequency
```

```
        set type to networkMediaChannelConnectionTerminationPoint
        set supporting-circuit-pack-name= SRGY_Circuit_Pack
        set supporting-port = SRGY-PPN
        set admin state of nmc-ctp interface inService
        set circuit-id (optional planning attribute)
        set nmc-ctp leaf values:
            set frequency
            set width
```

4. The following step is optional. It aims at verifying that input power is compliant with MSA specifications:

```
    Get-config (currentPmlist/currentPm
          [resource/interface-name = nmc-SRGY-PPN-frequency
          and granularity="15min"]
          /pmParameterName/type = opticalPowerInput)
```

♢ Check if power reading from the PM (pmParameterName/type = opticalPowerInput) is withing reasonable distance from the the expected input power derived from MSA specifications on line port channel output power reduced of the expected patch cord loss.

9.2.3.3 Provision of the Turn Back Connections

Main actions sequence:

1. create a roadm-connection between the nmc-ctp interfaces

```
Edit-config (org-openroadm-device/roadm-connections)
  roadm-connections
      set name = SRGX-PPN-[frequency]-to-SRGY-PPN
      set source/src-if = nmc-SRGX-PPN-frequency
      set destination/dst-if = nmc-SRGY-PPN-frequency
```

2. Power setting and control-mode change (target-power is not set for turn back links)

```
Edit-config (org-openroadm-device/roadm-connections
  [ connections-name = SRGX-PPN-[frequency]-to-SRGY-PPN
      set name = SRGX-PPN-[frequency]-to-SRGY-PPN
      set optical control-mode = power
```

3. create a roadm-connection between the nmc-ctp interfaces in the opposite direction

```
Edit-config (org-openroadm-device/roadm-connections)
  roadm-connections
      set name = SRGY-PPN-[frequency]-to-SRGX-PPN
      set source/src-if = nmc-SRGY-PPN-frequency
      set destination/dst-if = nmc-SRGX-PPN-frequency
```

4. Power setting and control-mode change (target-power is not set for turn back links)

```
Edit-config (org-openroadm-device/roadm-connections
  [ connections-name = SRGY-PPN-[frequency]-to-SRGX-PPN
      set name = SRGY-PPN-[frequency]-to-SRGX-PPN
      set optical control-mode = power
```

9.3 ROADM MANAGEMENT – SERVICE DELETION

9.3.1 Add and Drop Link Deletion

This section presents the sequence of validations and commands needed for the deleting ROADM-connections, and interfaces (i.e., MC and NMC) of the add link from an SRG-PP to Degree TTP logical points and the drop link in the opposite direction.

We currently limit MSA operations to the case of the degree nmc-ctp interface being the only child of a mc-ttp interface, thus, nmc-ctp and mc-ttp are deleted both.

Input parameters:

- node-id (xponderX)
- SRG/Deg numbers (SRGX, DEGY)
- circuit pack (SRGX-Circuit-Pack, DEGY-Circuit-Pack)
- supporting port (SRGX-PPN, DEGY-TTP)
- Interfaces names (SRGX-PPN-frequency, nmc-ctp-DEGY-TTP-frequency)
- connections names (SRGX-PPN-frequency-to-nmc-ctp-DEGY-TTP-frequency)
- minimum output power (-60dBm)

1. set the ADD roadm-connection target output power to -60dBm

```
Edit-config (org-openroadm-device/roadm-connection
      [connection-name = SRGX-PPN-frequency-to-nmc-ctpDEGY-TTP-frequency])
      set target-output-power = -60dBm
```

2. Set admin state SRG PP port supporting the nmc-ctp interface to outOfService:

```
Edit-config (org-openroadm-device/circuit-pack
      [circuit-pack-name = SRGX_Circuit_Pack
      /ports [port-name= SRGX-PPN)
      set admin state to outOfService
```

3. Delete the ADD roadm-connection:

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = SRGX-PPN-frequency-to-nmc-ctpDEGY-TTP-frequency])
```

4. confirm the presence of the Drop roadm-connection:

```
Get-config (org-openroadm-device/roadm-connection)
```

◇ check if the corresponding Drop roadm connection (nmc-ctp-DEGY-TTP-frequency-to-SRGX-PPN-frequency) is present. If it is present, delete it.

5. delete the Drop roadm-connection if it is present

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = nmc-ctp-DEGY-TTP-frequency-to-SRGX-PPN-frequency])
```

6. delete the nmc-ctp interface on the supporting SRGX PP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-ctp-DEGY-TTP-frequency])
```

7. delete the nmc-ctp interface on the supporting DEGY TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-ttp-DEGY-TTP-frequency])
```

8. delete the mc-ctp interface on the supporting DEGY TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = mc-ttp-DEGY-TTP-frequency])
```

9.3.2 Express Link Deletion

This section presents the sequence of validations and commands needed for the deleting ROADM-connections, and interfaces (i.e., MC and NMC) of the express link between 2 Degrees TTP logical points. Connections in both direction will be deleted before the deletion of the nmc and mc interfaces on both supporting ports.

We currently limit MNSA operations to the case of the degree nmc-ctp interface being the only child of a mc-ttp interface, thus, nmc-ctp and mc-ttp are deleted both.

Input parameters:

- node-id (xponderX)
- DEG/Deg numbers (DEGX, DEGY)
- circuit pack (DEGX-Circuit-Pack, DEGY-Circuit-Pack)
- supporting port (DEGX-PPN, DEGY-TTP)
- Interfaces names (DEGX-PPN-frequency, nmc-ctp-DEGY-TTP-frequency)
- connections names (nmc-ctpDEGX-TTP-frequency-to-nmc-ctpDEGY-TTP-frequency, nmc-ctpDEGY-TTP-frequency-to-nmc-ctpDEGX-TTP-frequency)
- minimum output power (-60dBm)

1. set the first express roadm-connection target output power to -60dBm

```
Edit-config (org-openroadm-device/roadm-connection
      [connection-name = nmc-ctpDEGX-TTP-frequency-to-nmc-ctpDEGY-TTP-frequency])
      set target-output-power = -60dBm
```

2. Set admin state DEG PP port supporting the nmc-ctp interface to outOfService:

```
Edit-config (org-openroadm-device/circuit-pack
      [circuit-pack-name = DEGY_Circuit_Pack
      /ports [port-name= DEGY-PPN)
      set admin state to outOfService
```

3. Delete the first express roadm-connection:

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = nmc-ctpDEGX-TTP-frequency-to-nmc-ctpDEGY-TTP-frequency])
```

4. set the first express roadm-connection target output power to -60dBm
5. confirm the presence of the other roadm-connection:

```
Get-config (org-openroadm-device/roadm-connection)
```

$\diamondsuit$ check if the corresponding other roadm connection (nmc-ctpDEGY-TTP-frequency-to-nmc-ctpDEGX-TTP-frequency) is present. If it is present, delete it.

6. set the second express roadm-connection target output power to -60dBm

```
Edit-config (org-openroadm-device/roadm-connection
      [connection-name = nmc-ctpDEGY-TTP-frequency-to-nmc-ctpDEGX-TTP-frequency])
      set target-output-power = -60dBm
```

7. Set admin state DEG PP port supporting the nmc-ctp interface to outOfService:

```
Edit-config (org-openroadm-device/circuit-pack
      [circuit-pack-name = DEGX_Circuit_Pack
      /ports [port-name= DEGX-PPN)
      set admin state to outOfService
```

8. Delete the second express roadm-connection:

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = nmc-ctpDEGY-TTP-frequency-to-nmc-ctpDEGX-TTP-frequency])
```

9. delete the nmc-ctp interface on the supporting DEGX TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-ttp-DEGX-TTP-frequency])
```

10. delete the mc-ctp interface on the supporting DEGX TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = mc-ttp-DEGX-TTP-frequency])
```

11. delete the nmc-ctp interface on the supporting DEGY TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-ttp-DEGY-TTP-frequency])
```

12. delete the mc-ctp interface on the supporting DEGY TTP port

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = mc-ttp-DEGY-TTP-frequency])
```

9.3.3 Turn Back Link Deletion

This section presents the sequence of validations and commands needed for the deleting ROADM-connections and interfaces (i.e., NMC) of the turn back from an SRG-PP to SRG-PP logical points in both directions.

Input parameters:

- node-id (xponderX)
- SRG numbers (SRGX, DEGY)
- circuit pack (SRGX-Circuit-Pack, SRGY-Circuit-Pack)
- supporting port (SRGX-PPN, SRGY-PPN)
- Interfaces names (nmc-SRGX-PPN-frequency, nmc-SRGY-PPN-frequency)
- connections names (SRGX-PPN-[frequency]-to-SRGY-PPN, SRGY-PPN-[frequency]-to-SRGX-PPN)

1. set the nmc-ctp interfaces on the supporting SRG PP port to outOfService

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = nmc-SRGX-PPN-frequency
      set admin state of nmc-ctp interface outOfService
```

```
Edit-config (org-openroadm-device/interfaces)
  interface
      set name = nmc-SRGY-PPN-frequency
      set admin state of nmc-ctp interface outOfService
```

2. set the admin states SRG PP port supporting the nmc-ctp interfaces to outOfService:

```
Edit-config (org-openroadm-device/circuit-pack
      [circuit-pack-name = SRGX_Circuit_Pack
      /ports [port-name= SRGX-PPN)
      set admin state to outOfService
```

```
Edit-config (org-openroadm-device/circuit-pack
      [circuit-pack-name = SRGY_Circuit_Pack
      /ports [port-name= SRGY-PPN)
      set admin state to outOfService
```

3. delete the roadm-connections:

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = SRGX-PPN-[frequency]-to-SRGY-PPN])
```

```
Edit-config (org-openroadm-device/roadm-connection
  roadm-connection operation = delete
      [connection-name = SRGY-PPN-[frequency]-to-SRGX-PPN])
```

4. confirm the presence of the turn back roadm-connection:

```
Get-config (org-openroadm-device/roadm-connection)
```

5. delete the nmc-ctp interfaces on the supporting SRG PP ports

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-SRGX-PPN-frequency])
```

```
Edit-config (org-openroadm-device/interface
  interface operation = delete
      [name = nmc-SRGY-PPN-frequency])
```

9.4 INCREMENTAL DEGREE, SRG, TURN BACK AND XPONDERS

This chapter describes the provisioning of new degrees, SRG, turn back modules/functions and Xponders.

To provision these entities, it is necessary to first provision the incremental hardware that is required, i.e., Shelf, Circuit-pack and Physical Links, and then provision the management entities for Degree, SRG and Xponders. There is no management entity for the turn back module or function.

These two steps are described in the following two sections.

9.4.1 Incremental Hardware

9.4.1.1 Incremental Shelf

Hardware augmentation may require provisioning new shelf entities associated with the degree, SRG, or Xponder that we want to create.

Provision a shelf and set the shelf attributes:

Input parameters:

- node-id*
- shelf-name*
- shel-type*
- rack location (rack location within the office)
- shelf position (within the rack)
- administrative-state
- equipment-state
- due-date

* Node-id, shelf-name and shelf-type are mandatory. All other parameters are optional.

```
Edit-config (org-openroadm-device/shelf
  shelf
      set the shelf-name
      set the shelf-type (to the vendor-specific shelf type)
      set the administrative-state to inService
      set the location
      set the shelf-position
      set the equipment-state
      set the due-date (if desired)
```

- Vendors define the type of shelf based on the shelf-type.
- The rack and shelf-position identifies the location of the shelf within an office. The rack is the rack identifier, and the shelf-position identifies the location of the shelf within the rack. The naming scheme for these identifiers are up to the operator.
- Equipment-state tracks the lifecycle states of the equipment. It takes on the following values:
  1. reserved-for-facility-planned: equipment is planned for use by a service
  2. not-reserved-planned: equipment is planned by not reserved for any purpose
  3. reserved-for-maintenance-planned: equipment is planned for use as a maintenance spare
  4. reserved-for-facility-unvalidated: equipment is reserved for use by a service but not validated against planned equipment
  5. not-reserved-unvalidated: equipment is not reserved for any purpose and not validated against planned equipment
  6. unknown-unvalidated: unknown equipment not validated against planned equipment
  7. reserved-for-maintenance-unvalidated: equipment is to be used for use as a maintenance spare but not validated against planned equipment
  8. reserved-for-facility-available: reserved for use by a service and available
  9. not-reserved-available: not reserved for use by a service and available
  10. reserved-for-maintenance-available: reserved as a maintenance spare and available
  11. reserved-for-reversion-inuse: equipment that is reserved as part of a home path for a service that has been temporarily re-routed
  12. not-reserved-inuse: equipment in use for a service
  13. reserved-for-maintenance-inuse: maintenance spare equipment that is in use as a maintenance spare

Read-only data

```
Get-config (org-openroadm-device/shelf
  [shelf-name = shelf-id])
```

- Physical inventory data is provided from the shelf.
  - It includes: vendor, model, serial-id, type (typically should match the shelf-type), product-code, manufacture-date, CLEI and hardware version.
- The slots container identifies the slots on a shelf that can host circuit-packs.
  - The label field would be optionally present to represent the silkscreen of the slot if the slot-name does not match the silkscreen.
  - The provisioned-circuit-pack would be present if there is a circuit pack that is both provisioned and plugged into the slot.
  - the slot-status provides information about the operational status of the slot including if there is a circuit-pack plugged into the slot and if there is a mismatch.

9.4.1.2 Incremental Circuit-pack

Hardware augmentation may require provisioning new circuit-pack entities associated with a degree, SRG, or Xponder that we want to create.

Provision a circuit-pack and set its attributes

Input parameters:

- node-id *
- circuit-pack-name *
- circuit-pack-type *
- circuit-pack-product-code
- circuit-pack-mode
- shelf *
- slot *
- subSlot
- parent circuit-pack-name
- parent cp-slot-name
- administrative-state
- equipment-state
- due-date

* Node-id, circuit-pack-name, circuit-pack-type, shelf, and slot are mandatory. All other parameters are optional.

```
Edit-config (org-openroadm-device/circuit-pack)
  circuit-pack
      set the circuit-pack-name
      set the circuit-pack-type
            (following vendor-specific defined circuit-pack-type)
      Set the circuit-pack-product-code (if required)
      Set the circuit-pack-mode (if required)
      Identify the location of the circuit-pack within Node
          Define the Shelf
          Define the Slot
          Define the subSlot (if required)
          Define the parent circuit-pack-name (if required)
          Define the parent cp-slot-name
      Set the equipment-state
      Set the due-date (if desired)
```

- Vendors define the type of circuit pack based on the circuit-pack-type (mandatory) and circuit-pack-product-code (optional). The combination of these two attributes should uniquely identify a circuit pack hardware.
  - If a circuit pack hardware have different operational modes, this is provisioned using the circuit-pack-mode.
- Identify the location of the circuit-pack within Node via the shelf, slot, and, optionally, subSlot attribute
  - If present, the subSlot name should match to cp-slot-name as defined in the parent-circuit-pack container
- The hierarchical location of circuit packs should use
  - the shelf and the slot fields for circuit packs that plug directly into a shelf (in this case, there will not be a parent-circuit-pack container).
  - add the parent-circuit-pack and cp-slot fields for circuit packs that plug into other circuit packs (e.g. pluggable optics that plug into a circuit pack).
- If circuit packs are plugged into a parent circuit pack, then the parent-circuit-pack container is mandatory. In this case, the parent is specified by providing both the parent's circuit-pack-name and cp-slot-name for the cp-slot that the circuit pack is plugging into.

```
<circuit-packs>
  <circuit-pack-name>1/1</circuit-pack-name>
  <circuit-pack-type>CPTYPE1</circuit-pack-type>
  <shelf>1</shelf>
  <slot>1</slot>
</circuit-packs>
<circuit-packs>
  <circuit-pack-name>1/1/1</circuit-pack-name>
```

```
  <circuit-pack-type>CPTYPE2</circuit-pack-type>
  <shelf>1</shelf>
  <slot>1</slot>
  <subSlot>1</subSlot>
  <parent-circuit-pack>1/1</parent-circuit-pack>
     <circuit-pack-name>1/1</circuit-pack-name>
     <cp-slot-name>1</cp-slot-name>
</circuit-packs>
```

- If a circuit pack occupies multiple slots, the provisioned-circuit-pack and the slot-status information should match for all slots involved. For example (circuit-pack occupying slots 5 and 6):

```
<slots>
  <slot-name>5</slot-name>
  <label>5</label>
  <provisioned-circuit-pack>1-5</ provisioned-circuit-pack >
  <slot-status>installed-prov-match</slot-status>
</slots>
<slots>
  <slot-name>6</slot-name>
  <label>6</label>
  <provisioned-circuit-pack>1-5</ provisioned-circuit-pack >
  <slot-status>installed-prov-match</slot-status>
</slots>
```

- equipment-state tracks the lifecycle states of the equipment. The values are the same as indicated for the shelf

Read-only data:

```
get-config (/org-openroadm-device/shelves [ Circuit-pack-name = circuit-pack-name ])
```

- Physical inventory data is provided from the circuit-pack
  - It includes vendor, model, serial-id, type (typically should match the circuit-pack-type), product-code (typically should match the circuit-pack-product-code), manufacture-date, CLEI and hardware version.
- Circuit-pack-category provides additional classification of the circuit-pack by the vendor.
- The cp-slots container identifies the “slots” on a circuit-pack that can host other circuit-packs.
  - The label field would be optionally present to represent the silkscreen of the cp-slot if the slot-name does not match the silkscreen.
  - The slot-type indicates if the slot is used to hold pluggable optics (pluggable-optics-holder) or other (non-optical-pluggable circuit packs).
  - The provisioned-circuit-pack would be present if there is a child circuit pack provisioned that is plugged into the cp-slot.
  - In addition, the slot-status provides information about the operational status of the slot including if there is a circuit-pack plugged into the slot and if there is a mismatch.
- The cp-slots container also contains capability information representing information about what can be used and created from the cp-slot.
- The is-pluggable-optics indicates if this circuit-pack represents an optical pluggable module.
- Firmware information including the software-load-version, firmware features list and firmware component lists.

9.4.1.3 Incremental Circuit-pack Ports

The port entities associated with the circuit pack are configured together with the circuit pack. Ports are expected to be auto-created upon creation of the circuit-pack. in the meantime, it may be nececessary to edit the attributes of any number of the circuit-pack’s ports.

Note: Recommendation to use the “merge” operation on editing port attributes.

Input parameters:

- node-id *

- circuit-pack-name *
- port-name *
- port-qual
- circuit-id
- administrative-state
- logical-connection-point
- Launch-cable-length and port-direction for OTDR ports [36]

* Node-id, circuit-pack-name and port-name are mandatory. All other parameters are optional.

```
Edit-config (org-openroadm-device/circuit-pack
  [circuit-pack-name]/ ports )
      Set the port-name
      Set the port-type to the vendor-specific port type
      Set the port-qualifier and logical-connection-point as below.
          (Note that some ports may not have these attributes set)
      Set the administrative-state of the port to inService
      For OTDR ports only:
          Set the launch-cable-length
          Set the port-direction.
```

- Vendors define the port-type for the port. This value is not standardized in the Open ROADM MSA specifications. This field is optional.
- The port-qual can take one the following values or not be present.
  - roadm-external for ROADM degree network port or for ROADM SRG add/drop port
  - otdr for ports on an OTDR
  - roadm-internal for any internal ports on the data path (in degree, SRG, or turn back module).
  - xpdr-network for xponder network ports
  - xpdr-client for xponder client port
  - switch network
  - switch-client
  - For all other ports, the port-qual is not present
- circuit-id is a user's defined field to identify the circuit that may be associated with this port. Its use is optional.
- The logical connection point is used to support the mapping at the controller level between the device model and network model.
  - For the ROADM degree network port, the logical connection point should be set to the format "DEG<n>-TTP-RX", "DEG<n>-TTP-TX" or "DEG<n>-TTP-TXRX" depending on the port-direction
    - E.g., for degree 1 with uni-directional ports, there should be two ports – one with logical connection point = "DEG1-TTP-RX" and the other with logical connection point = "DEG1-TTP-TX"
  - For the ROADM degree internal ports that connect to another degree or SRG, the logical connection point should be set to the format "DEG<n>-CTP-RX", "DEG<n>-CTP-TX" or "DEG<n>-CTP-TXRX" depending on the port-direction
    - E.g., "DEG1-CTP-RX" and "DEG1-CTP-TX"
  - For the ROADM SRG add/drop port, the logical connection point should be set to the format "SRG<n>-PP<m>"
    - E.g., port number 7 on SRG1 would have the logical connection = "SRG1-PP7"
  - For the ROADM SRG internal ports that connect to another degree, SRG or turn back module, the logical connection point should be set to the format "SRG<n>-CP-RX", "SRG<n>-CP-TX" or "SRG<n>-CP-TXRX" depending on the port-direction
    - E.g., uni-directional ports connecting SRG 1 to degree 1 will be two internal ports with logical connection points "SRG1-CP-RX" and "SRG1-CP-TX"

[36] OTDR ports are present only in degrees

  - For the turn back module internal ports that connect to an SRG, the logical connection point should be set to the format “TRBK<n>-CTP-RX”, “TRBK<n>-CTP-TX” or “TRBK<n>-CTP-TXRX” depending on the port-direction
    - E.g., uni-directional ports connecting Turn Back 1 to SRG 1 will be two internal ports with logical connection points “TRBK1-CTP-RX” and “TRBK1-CTP-TX”
  - For the xponder network port, the logical connection point should be set to the format XPDR<n>-NETWORK<m>
  - For the xponder client ports, the logical connection point should be set to the format XPDR<n>-CLIENT<m>
- For OTDR ports, the OTDR launch cable length sets the length of the launch cable and the port-direction indicates if port is associated with the degree receive or transmit port. Currently, the Open ROADM MSA runs the OTDR over the degree receive port.

Read-only data:

- port-wavelength-type
  - Set to multi-wavelength for the ROADM network degree ports.
  - Set to wavelength for the ROADM SRG add/drop port and for the xponder network and client ports
  - This attribute may not be present for other ports.
- port-direction indicates if the port is modelled as a uni-directional or bi-directional port, and if uni-directional, whether the port is transmit or receive. The values for port-direction are “tx”, “rx” or “bi-directional”
- facelabel is meant to convey the silkscreen on the device to aid in physically locating the port on the device (e.g., provided to the local technician for troubleshooting). The faceplate-label is a mandatory RO field. Vendors should provide the faceplate-label even if the silkscreen matches the port-name.
- supported-interface-capability is used to indicate which interface(s) can be built on the port. The ROADM network port list of interface capabilities (if-cap-type) is: if-OTS-OMS for ROADM degree.
- The partner-port identifies the associated port when the ports are uni-directional.
- The interface container contains a list of provisioned interfaces associated with the port.
- port-capabilities is used to indicate the capabilities of a port:
  - for ROADM SRG add/drop port, port-capabilities includes which interface(s) can be built on the port. The list of interface capabilities include: if-NMC.
  - for the roadm-port, port-capabilities includes the minimum and maximum aggregate powers supported by the port.
  - for the xponder network port, the list of interface capabilities include: if-OCH-OTU4-ODU4 or if-otsi-otsigroup
  - for the xponder client port, the list of interface capabilities may include (but not limited to): if-100GE, if-100GE-ODU4, if-10GE, if-10GE-ODU2, if-10GE-ODU2e, if-1GE, if-1GE-ODU0, if-OTU4-ODU4, if-OCH-OTU4-ODU4, if-OCH-OTU2E-ODU2E, if-OTU4-ODU4, if-OTU2-ODU2, etc.
    - The client supports 1GE, 10GE, 100GE, OTU2 and OTU4.
    - The OCH interface on client side OTU interfaces is optional. Vendor should use the appropriate interface capability type to indicate if OCH is required or not
  - The transponder-port capabilities details the minimum and maximum aggregate powers supported by the port. This field applies to the xponder network and client ports

9.4.1.4 Incremental Physical Link

Physical link entities describe the fibering and/or cabling supporting the ROADM degree and/or SRG functionality. Fibering and cabling will connect different CP port within the degree or SRG, between the degree or SRG and common units, between the degree or SRG and other pre-existing degrees and SRGs.

Input parameters:

- node-id *
- phisical-link-name *
- source and destination port identifiers *

* All parameters are mandatory.

```
Edit-config (org-openroadm-device/physical-link)
  physical-link
      Set the physical-link-name
      Identify the source circuit-pack-name and port-name (TX)
      Identify the destination circuit-pack-name and port-name (RX)
```

- Physical links are always uni-directional
  - Physical links Source should indicate the TX port
  - Physical links Destination should indicate the RX port
- Physical links can represent a single bi-directional cable (e.g., Ethernet cable).
  - Bidirectional cable are modeled as two uni-directional physical link
- Physical links can also represent MPO connections.
  - Both the paretn physical connector and the child connectors (child ports within the MPO) should be modeled

9.4.2 Degree Growth

Degrees are to be deployed from degree number 1 to degree number N sequentially, up to the maximum number of degrees supported by the NE. This means that degree m would not be deployed before degree (m-1).

The growth of degrees is independent of the number of SRGs currently deployed.

To deploy a new degree, the following provision would take place:

- Provision new shelves as necessary (refer to section 1.4.1.1)
- Provision new circuit packs that will support the new degrees , (refer to section 1.4.1.2).
  This includes:
  - setting the TTP and CTP logical connection point attributes
    (against the port entities)
  - provision interfaces supported by the port entities (e.g., OTS, OMS, OSC)
- Provision the degree container for the new degree which include:
  - The list of circuit packs associated with the new degree.
  - The list of external connection ports associated with the degree
- Provision the protocols associated with the new direction (e.g., LLDP, RSTP)
- Provision the new physical links associated with the degree. This may include:
  - Physical links internal to the degree, including the physical link for the OSC jumper that connects to the OSC pluggable circuit pack.
  - Physical links connecting the new degree to the existing degrees
  - Physical links connecting the new degree to the existing SRGs
  - Physical links to the OTDR unit

9.4.2.1 Create Degree Entity

Creates new degree entity that defines the circuit-packs associated with the degree, and the connection ports for the external connections. Input parameters:

- node-id *
- degree-number *
- list of circuit packs associated with the degree*
- list of connection-ports (external ports) associated with the degree *
- For the OTDR application, the OTDR's circuit-pack-name and port-name that is used to monitor this degree.

* Node-id, degree-number, circuit-pack-list, and connection-port-list are mandatory. OTDR application data is optional.

```
Edit-config (org-openroadm-device/degree)
  degree
      Set the degree-number
      provision the list of all circuit-packs associated to the degree
      provision the list of the connection ports associated to the degree
      configure the otdr-port that would monitor the degree
```

- The circuit-packs container lists all circuit-packs associated with the degree.
  - circuit-packs shared with other SRG (or degree) are listed under all the degree (and/or SRG) using them.
- The connection-ports container lists the ports that connect externally to the line transmission fibers.
  - The connection ports refer to the physical connection to the external line fibers.
  - This may also be the logical connections where the OTS interfaces are constructed.
- The otdr-port container provides the OTDR port that will monitor the degree.
  - The OTDR should monitor the degree receiver port.
  - The circuit-pack-name and port-name refer to CP and the port on an OTDR module.

Read-only data:

- The max-wavelength identifies the maximum number of wavelengths supported by the degree.

9.4.3 SRG Growth

SRGs are the add/drop units on the ROADM device. SRGs are to be deployed from SRG number 1 to SRG number N sequentially, up to the maximum number of SRGs supported by the NE. This means that SRG m would not be deployed before SRG (m-1).

The growth of SRGs is independent of the number of degrees currently deployed.
like in the case of the the deployment of degrees, to deploy a new SRG, the following provision would take place:

- Provision new shelves as necessary (refer to section 1.4.1.1)
- Provision new circuit packs that will support the new SRG , (refer to section 1.4.1.2).
  This includes:
  - setting the TTP and CTP logical connection point attributes
    (against the port entities)
- Provision the shared-risk-group container for the new SRG which include:
  - The list of circuit packs associated with the new SRG.
  - The list of external connection ports associated with the SRG
- Provision the new physical links associated with the degree. This may include:
  - Physical links internal to the SRG.
  - Physical links connecting the new SRG to the existing degrees

9.4.3.1 Create Shared Risk Group (SRG) Entity

Creates new shared-risk-group entity that defines the circuit-packs associated with the SRG. Input parameters:

- node-id *
- srg-number *
- list of circuit packs associated with the SRG *

* All parameters are mandatory.

```
Edit-config (org-openroadm-device/degree)
  degree
      Set the srg-number
      provision the list of all circuit-packs associated to the srg
```

- The circuit-packs container lists all circuit-packs associated with the SRG.
  - circuit-packs shared with other SRG (or degree) are listed under all the degree (and/or SRG) using them.

Read-only data:

- The max-add-drop-ports identifies the maximum number of SRG add/drop ports supported by the SRG, regardless of the circuit-packs associated with the maximum SRG configuration are provisioned.
- The current-provisioned-add-drop-ports identify the maximum number of SRG add/drop ports supported by the SRG based on the current provisioning of the SRG.

9.4.4 Turn Back Growth

Turn back modules allow the traffic signal to be looped back from one SRG port to another SRG port. Turn back modules have internal ports that connects to the SRG network-side (internal) ports. To deploy new turn back module equipment, the following provision would take place:

- Provision new shelves as necessary (refer to section 1.4.1.1)
- Provision new circuit packs that will support the new turn back module, (refer to section 1.4.1.2).
  This includes:
  - configuring the ports associated with the circuit packs
- If the turn back module has multiple circuit packs with physical cabling between the circuit-packs, the new physical links associated with the turn back module will have to be provisioned.

The turn back function can also be implemented by a direct optical cabling between two SRG network-side (internal) ports. To deploy the turn back functionality in this manner, the following provisioning would take place (assumption is that the SRG network-side port has already been configured when the SRG was deployed):

- The phyiscal links connecting the SRG network-side ports are provisioned

Note: there is no management entity to represent a turn back module or function.

9.4.5 Xponder Growth

Xponders have network port(s) that connects to the ROADM's SRG add/drop ports and client port(s) that connect to Routers or other client equipment. To deploy new xponder equipment, the following provision would take place:

- Provision new shelves as necessary (refer to section 1.4.1.1)
- Provision new circuit packs that will support the new xponder, (refer to section 1.4.1.2).
  This includes:
  - configuring the ports associated with the circuit packs
- Provision the xponder container for the new transponder, Muxponders/Switches , etc. which include:
  - The xponder type (xpdr-type)
  - For regenerators, indicates if the regenerator supports wavelength change using the recolor attribute
  - The list of network and client ports associated with the xponder.
  - The equipment SRG shared risk (eqpt-srg-id) associated with the ports on the xponder. This identifier is a locally significant identifier (node-wide unique, not globally unique) that identifies the ports that share a common equipment SRG when the ports have the same eqpt-srg-id.
- If the xponder had multiple circuit packs with physical cabling between the circuit-packs, the new physical links associated with the xponder will have to be provisioned.

9.4.5.1 Xponder Type

The MSA supports different types of Xponders differently. The table below summarizes some relevant differences.

xponder	Xpdr-type	Port-Qualifier	Connectivity Advertisement	Explicit Cross Connect?
Transponder	Tpdr	Xpdr-client/network	Connection-map	No
Muxponder	Mpdr	Switch-client/network	Switching Pool	Yes
Regenerator	Regen	Xpdr-network	Connection-map	No
	Regen			
OTN Switch / **Switchponder**	Switch	Switch-client/network	Switching Pool	Yes

Table 24: provisioning Xponders parameter

9.4.5.2 Provisioning a Regenerator

Regenerators will be supported in NE nodes together with transponders, muxponders and Switchponders.

Provisioning is driven by the vendor template, Implementations may require setting the equipment to a special mode to support regenerators.

A possible implementation option is to use the circuit-pack-mode to configure the equipment into a regenerator mode.

Provisioning is driven by the vendor template.
Set to a vendor specific value when the circuit-pack will be configured as a regenerator.

Regenerators are similar to transponders: Port-qual is provisioned as xpdr-network, Port-direction is bidirectional, and connectivity is established by default and exposed via connection-map, explicit provision of ODU cross connects is not required.

Different SRGs will be used for the East and West direction of a regenerator

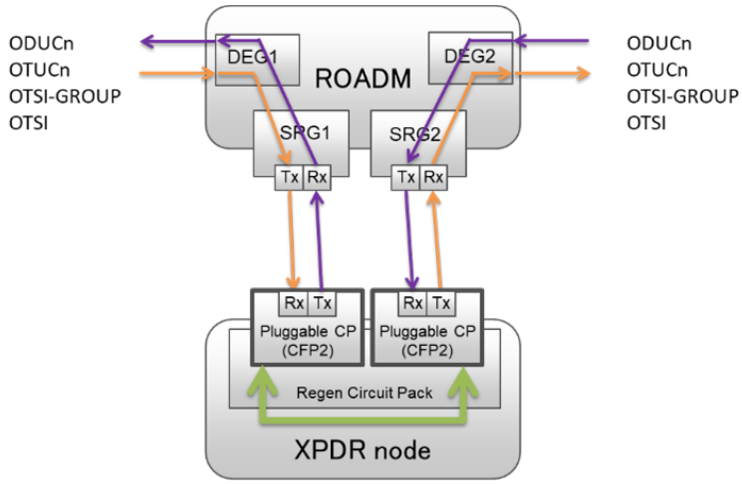

Figure 84: Regnerator - Functional Diagram

Deployment of a regenerator may require setting the equipment to a special mode.

In the Open ROADM model, a regenerator will be described by one XPONDER entity.

9.4.5.3 Create Xponder Entity

Creates new xponder entity that defines the xponder type and the ports associated with the xponder. Input parameters:

- node-id *
- xpdr-number *
- xpdr-type *
- list of ports associated with the xponder *
- eqpt-srg-id for the ports

* node-id, xpdr-number, xpdr-type and xpdr-port list are mandatory, eqpt-srg-id are optional.

```
Edit-config (org-openroadm-device/xponder )
  xponder
      Set the xpdr-number
      Set the xpdr-type
      Set the list of xpdr-ports associated to the xponder including
      the eqpt-srg-id associated with the port
```

- The xpdr-type can be tpdr (transponder), mpdr (Muxponders/Switches), switch (OTN switch or switchponder), regen (regenerator) or regen-uni (uni-directional regenerator)
- The xpdr-port container lists ports associated with the xponder
  - The list of port associated to a xponder group can change over time
  - On creation the list may be partial and new port may be added when needed

- eqpt-srg-id identifies the shared risk of the ports based on common equipment utilization. This value is locally significant to the node (node-wide unique)

Read-only data:

- recolor indicates if a regenerator supports wavelength recolorin
  This only applies if xpdr-type is regen or regen-uni.

10 MAINTENANCE ACTION USE CASES

Changes since last release:

- Updated section on Loopback regarding single versus multiple steps activation
- Updated section on OTDR scan
- Updated section on ‘Retrieve ROADM Connection Port Trail’
- Added new section on ‘Recovery via ZTP and dabase restoration’
- Added new section on ‘In-Service Database Rebuild’
- Added new section on ‘Manual laser enable/disable’
- Added new section on ‘Optical Control loop with span loss update’

10.1 DEVICE RESET/RESTART OPERATION

The Open ROADM MSA supports various types of reset/restarts depending on the intended result. Two types of restarts are defined (*option/warm* or *option/cold*) that can be initiated at a system or circuit-pack level. A notification is sent after the restart completes.

Restarts are initiated via the *restart* RPC and modeled as follows (see YANG sub-tree 68):

10.1.1 System Level Restart

System level restarts are initiated against the device. Typically, a warm restart results in a reset of the main CPU of the device and a cold restart results in a reset of all device CPUs. Implementations may vary. An example system level warm restart is shown in YANG sub-tree 69):

For backward compatibility, the device should also support the format as in YANG sub-tree 70):

10.1.2 Circuit Pack Level Restart

Circuit-pack restarts are typically initiated as a cold restart after a firmware update (firmware download operation) of a circuit pack.

An example cold restart against a circuit-pack is shown in YANG sub-tree 71):

For backward compatibility, the device should also support the format as in YANG sub-tree 72):

```
+---x restart
+---w input
| +---w device
| | +---w node-id?
| +---w resource
| | +---w (resource)?
| | +--:(circuit-pack)
| | | +---w circuit-pack-name
| | +--:(port)
| | | +---w port
| | | +---w circuit-pack-name
| | | +---w port-name?
| | +--:(connection)
| | | +---w connection-name
| | +--:(physical-link)
| | | +---w physical-link-name
| | +--:(internal-link)
| | | +---w internal-link-name
| | +--:(shelf)
| | | +---w shelf-name
| | +--:(srg)
| | | +---w srg-number
| | +--:(degree)
| | | +---w degree-number
| | +--:(service)
| | | +---w service-name
| | +--:(interface)
| | | +---w interface-name
| | +--:(odu-sncp-pg)
| | | +---w odu-sncp-pg-name
| | +--:(other)
| | | +---w other-resource-id
| | +--:(device)
| | | +---w node-id
| | +--:(line-amplifier)
| | | +---w amp-number
| | +--:(xponder)
| | | +---w xpdr-number
| | +--:(versioned-service)
| | | +---w versioned-service-name
| | | +---w version-number
| | +--:(temp-service)
| | +---w common-id
| +---w resourceType
| | +---w type
| | +---w extension?
| +---w option?
```

YANG sub-tree 68: Restart RPC

```
<nc:prc nc:message-id="167.254.172.235-1409"
↪  xmlns:nc="urn:ietf:parms:xml:ns:netconf:base:1.0">
    <restart xmlns="http://org/openroadm/de/operations">
        <device>
            <node-id>owb-xpdr-20</node-id>
        </device>
        <resource>
            <node-id>owb-xpdr-20</node-id>
        </resource>
        <resourceType>
            <type>device</type>
        </resourceType>
        <option>warm</option>
    </restart>
</nc:rpc>
```

YANG sub-tree 69: System level warm restart

```
<nc:prc nc:message-id="167.254.172.235-1409"
↪  xmlns:nc="urn:ietf:parms:xml:ns:netconf:base:1.0">
    <restart xmlns="http://org/openroadm/de/operations">
        <device>
            <node-id>owb-xpdr-20</node-id>
        </device>
        <resource/>
        <resourceType>
            <type>device</type>
        </resourceType>
        <option>warm</option>
    </restart>
</nc:rpc>
```

YANG sub-tree 70: System level warm restart - Backward Compatible

```
<nc:prc nc:message-id="167.254.172.235-1409"
↪  xmlns:nc="urn:ietf:parms:xml:ns:Netconf:base:1.0">
    <restart xmlns="http://org/openroadm/de/operations">
        <device>
            <node-id>owb-xpdr-20</node-id>
        </device>
        <resource>
            <circuit-pack-name>10/01/1</circuit-pack-name>
        </resource>
        <resourceType>
            <type>circuit-pack</type>
        </resourceType>
        <option>cold</option>
    </restart>
</nc:rpc>
```

YANG sub-tree 71: Cold restart against a circuit-pack

```
<nc:prc nc:message-id="167.254.172.235-1409"
↪  xmlns:nc="urn:ietf:parms:xml:ns:Netconf:base:1.0">
    <restart xmlns="http://org/openroadm/de/operations">
        <resource>
            <circuit-pack-name>10/01/1</circuit-pack-name>
        </resource>
        <resourceType>
            <type>circuit-pack</type>
        </resourceType>
        <option>cold</option>
    </restart>
</nc:rpc>
```

YANG sub-tree 72: Cold restart against a circuit-pack - Backward Compatible

10.2 OPERATE/RELEASE LOOPBACK ON AN INTERFACE

Loopbacks are used in conjunction with a test set (generate test signal) to test new interfaces before running live traffic or to logically locate the source of a network failure (i.e. test validity of signal being looped back). The Open ROADM MSA device model supports facility and terminal loopbacks on both Ethernet and OTU client-side and network-side interfaces.

A loopback is initiated on an interface by setting the *maint-loopback/enabled=true* and released by setting the *maint-loopback/enabled=false*. Note: the facility may need to be placed into a maintenance state (*administration-state=maintenance*) prior to enabling the loopback.

NOTE: The Open ROADM MSA model does not specify if the device must support putting the facility into maintenance state AND enable the loopback in a single or multiple edit-config operation, but single edit-config is desirable.

When a loopback is in effect, the device should raise a notification based on the type of loopback (*facilityLoopbackActive, facilityLoopback2Active, or terminalLoopbackActive*).

10.2.1 Facility Loopbacks

A facility loopback (local loopback) is used to test external connections (fiber) to the interface (refer to Figure 85). The type attribute setting for facility loopbacks is based on whether FEC is supported on the interface:

- *type=fac*: FEC is not supported or disabled on the interface, or pre-FEC loopback if FEC is supported.
- *type=fac2*: FEC is supported and enabled on the interface. This is a post-FEC loopback.

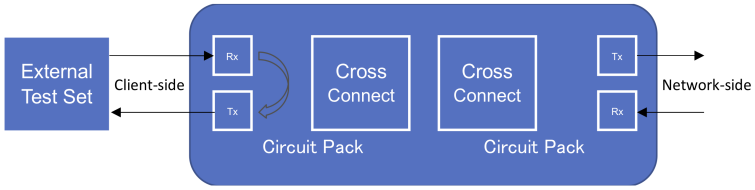

Figure 85: Facility Loopback Operations - Client-side Interface

Note: enabling/disabling facility loopbacks on network-side interfaces follow the same approach as client-side facility loopbacks (typically *type=fac2*, as FEC is likely supported and enabled on network-side interfaces).

Facility loopbacks are also supported on the network side interface as shown in Figure 86.

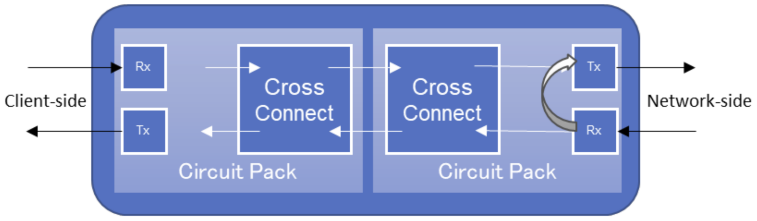

Figure 86: Facility Loopback Operations - Network side interface

10.2.2 Terminal Loopbacks

A terminal loopback (remote loopback) is used to test end-to-end circuits or test internal connections (e.g. incoming signal towards the backplane) to the device (refer to Figure 87). Setting *type=term* is used to establish terminal loopbacks. The test set in a client-side interface terminal loopback would be placed on the client-side interface on the other side of the network.

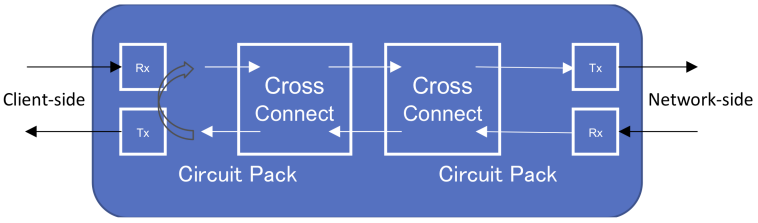

Figure 87: Terminal Loopback Operation - Client-side Interface

Note: enabling/disabling terminal loopbacks on network-side interfaces follow the same approach as client-side terminal loopbacks.

Terminal loopbacks are also supported on the network side interface as shown in Figure 88.

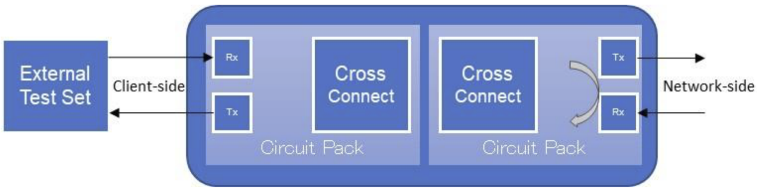

Figure 88: Terminal Loopback Operation - Network-side Interface

10.2.3 Loopback for B100G Interfaces

For B100G interfaces, facility and terminal loopbacks are supported at the OTUCn level.

Figure 89 shows facility and terminal loopbacks at the OTUCn. Since the OTUCn layer does not support FEC, the facility loopback type is set to “fac”.

Loopbacks are not supported at the OTSI or OTSIG level.

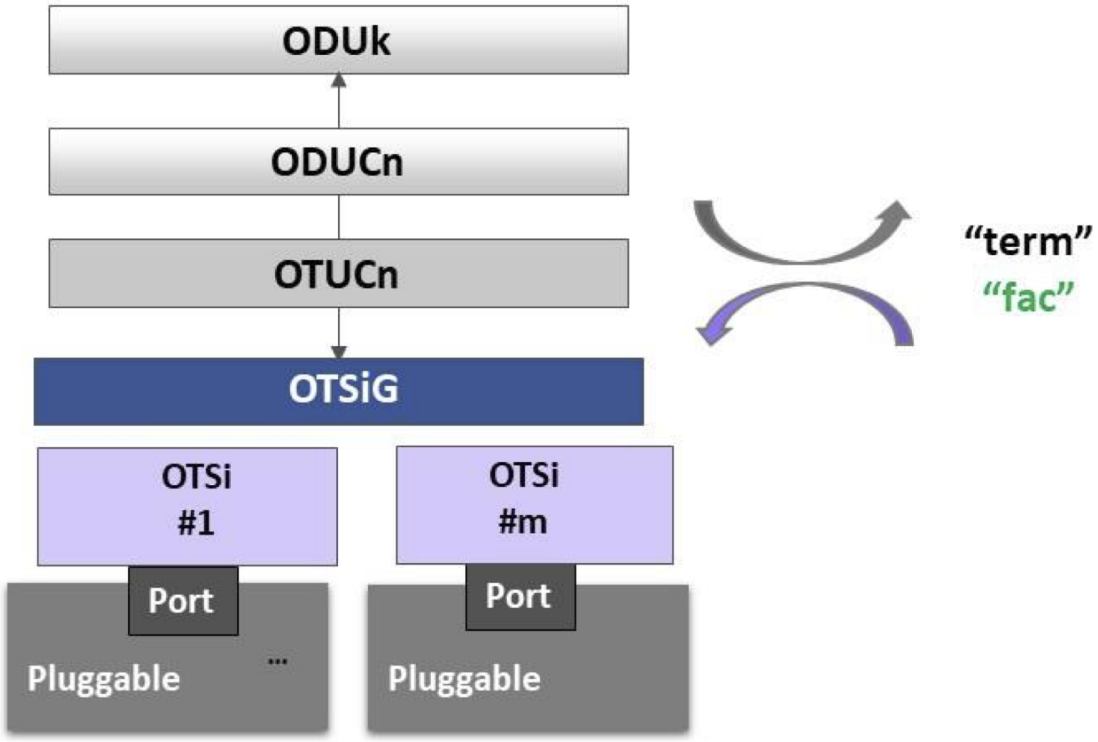

Figure 89: B100G Loopbacks

10.3 BIT-ERROR RATE (BER) TEST APPROACH

10.3.1 WDM Layer BER Test

For the WDM layer, the BER test based on FEC counters used for WDM services. The Open ROADM controller will retrieve the following current PM types for the BER computation:

- 100G
  - OTUk Interface: *preFECCorrectedErrors* (mandatory)
  - OTUk Interface: *FECUncorrectableBlocks* (mandatory)
  - OTUk Interface: *severelyErroredSeconds* (optional, but recommended)
  - OTUk Interface: *unavailableSeconds* (optional)
- B100G

- OTSi Interface: *preFECCorrectedErrors* (mandatory)
- OTSi Interface: *preFECUncorrectable Blocks* (mandatory)
- OTUCn Interface: *severelyErroredSeconds* (optional, but recommended)
- OTUCn Interface: *unavailableSeconds* (optional)

A WDM layer BER test is uni-directional with FEC PMs retrieved from both ends (no loopback). BER computations are attempted during a single 15min bin period.

10.3.2 OTN Layer BER Test

For the OTN layer, the BER test (*maint-testsignal*) is based on an ODU test signal used for OTN services (Ethernet and OTU clients). The controller will initiate the BER test on the client-side ODU unless the ODU indicates *no-oam-function* or *no-maint-testsignal-function*. If one of the two "no function" flags is set, the ODU will not support test signal generation.

If the client-side ODU does not support test signal generation, then the controller will initiate the test on the network side ODU. Note: implementations should not indicate *no-oam-function* or *no-maint-testsignal-function* on both the client-side and network-side ODU.

Prior to enabling an OTN layer BER test, the controller may clear all maintenance and diagnostic counters using the *clear-diagnostic* action or by toggling the enabled attribute. The interface will need to be put into a maintenance state (*administration-state=maintenance*) prior to enabling the test signal functionality. Note: The Open ROADM MSA model does not specify if the device must support putting the facility into maintenance state AND enable the BER test in a single or multiple edit-config operation, but single edit-config is desirable.

An OTN layer BER test is uni-directional with the test signal initiated at both ends (no loopback). Setting *enabled=true* on the desired ODU interface will enable the test signal. The *testPattern* attribute should be set to PRBS31 for OTN services (per ITU-T G.709 [7]). *PRBS31* is used in the "inverted" format per IEEE 802.3-2018 [4] clause 49.2.8. If the test signal is initiated on the client-side ODU, the test signal will be set to the terminal direction (*type=term*). If the test signal is initiated on the network-side ODU, it will be set to the facility direction (*type=fac*).

The following table details the choices for test signal direction depending on test signal type:

Table 25: Test Signal Direction

Rule	Interface Type	ODU-FUNCTION	Test Signal Type Test Signal Alarm	Test Signal Direction
1	ETH	n/a	FAC facilityTestsignalActive	Toward PMA layer
2 [37]	ODU	ODU-TTP	FAC facilityTestsignalActive	Toward parent interface
3a	ODU	ODU-TTP-CTP (with OTU or ODU parent)	FAC facilityTestsignalActive	Toward parent interface
3b	ODU	ODU-TTP-CTP (with ETH parent)	TERM terminalTestsignalActive	Toward switch/connection/ network port
4a [38]	ODU	ODU-CTP	TERM terminalTestsignalActive	Toward switch/connection
4b [39]	ODU	ODU-CTP	FAC facilityTestsignalActive	Toward parent interface

The key statistics are *inSync* for the sync status of the received test signal, *seconds* for the number of seconds the received test signal is in sync, *bitErrors* for a bit error count of the received test signal, and bitErrorRate (optional) for the bit error rate of the received test signal. BER is computed from the *seconds* and *bitErrors* statistics or by the *bitErrorRate*. Implementations must support the *inSync*, *seconds* and *bitErrors* statistics. Support for the *bitErrorRate* is optional.

When a test signal is in effect, the device raises an alarm notification based on the type of test signal (*facilityTestsignalActive* or *terminalTestsignalActive*).

[37] Currently this case is not required to be supported
[38] Current use case restricts test signal toward the network port
[39] Current use case restricts test signal toward the network port

10.4 DISABLE AUTOMATIC LASER SHUTDOWN

In certain scenarios, it may be desirable to disable a laser's automatic shutdown mechanism (e.g. operate the laser safety override control on a given high-power optical transmitter/amplifier unit to allow optical power measurements by an external device). Disabling the automatic laser shutdown of an identified entity is initiated by the Open ROADM MSA device via the *disable-automatic-shutoff* RPC. The identity of the interface can be either the *degree-number* of a ROADM (*rdm*) or the *amp-number* of an ILA (*ila*). The automatic laser shutdown mechanism is re-enabled after a specified support-timer duration (1...600 in seconds, 20 seconds is the default value).

```
+---x disable-automatic-shutoff
| +---w input
| | +---w (degree-or-amp)
| | | +--:(degree)
| | | | +---w degree-number
| | | +--:(amp)
| | | +---w amp-number
| | +---w support-timer?
| +--ro output
| +--ro status
| +--ro status-message?
```

When the automatic laser shutdown is disabled, an alarm (automaticShutoffDisabled) should be raised by the device.

10.5 OPTICAL TIME DOMAIN REFLECTOR (OTDR) SCAN

The OTDR is intended to monitor and characterize the integrity (e.g. measure the length of a span, verify splice loss, locate faults on the fiber link) of the MW input ports of a ROADM (refer to Figure 90) and the MWi input ports of an ILA (refer to Figure 91). The Open ROADM MSA specifies that implementations would use the OTDR on the Rx fiber (OTDR scans the fiber in the opposite direction of the traffic flow).

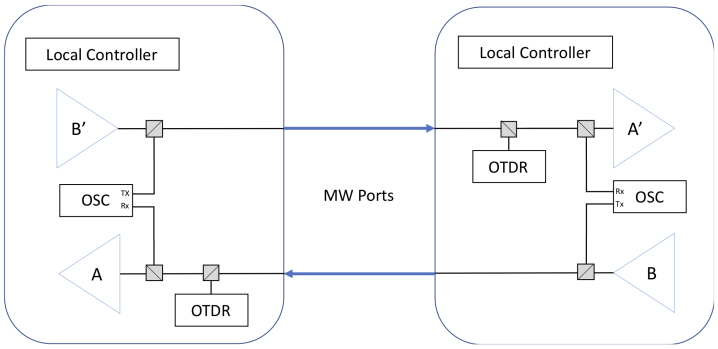

Figure 90: ROADM OTDR Reference Model

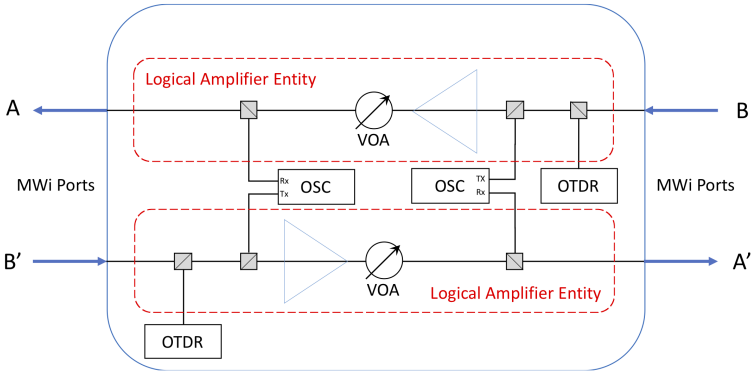

Figure 91: ILA OTDR Reference Model

In these configurations, the OTDR pulses propagate along the fiber. In order to not interfere with other functions of the Open ROADM MSA device, the wavelength of the OTDR must be chosen to be outside the Channel Plan Frequency Range

```
+---x start-scan
| +---w input
| | +---w (degree-or-amp)
| | | +--:(degree)
| | | | +---w degree-number
| | | +--:(amp)
| | | +---w amp-number
| | +---w port-direction?
| | +---w distance?
| | +---w resolution?
| +--ro output
| +--ro status
| +--ro status-message?
```

YANG sub-tree 73: OTDR start scan RPC

as defined in the Common tab and the OSC CWDM Channel Frequency Range as defined in the OSC-Optical Line Port tab of the Open ROADM MSA Specification.

The Open ROADM MSA does not restrict an OTDR implementation (e.g. OTDR integrated in the degree, a separate OTDR circuit-pack, an OTDR pluggable optic, etc.).

An OTDR scan of a fiber link is initiated by the Open ROADM MSA device via the *start-scan* RPC (see YANG sub-tree 73). The identity of the interface for the scan can be either the *degree-number* of a ROADM (*rdm*) or the *amp-number* of an ILA (*ila*). The technician has the option to specify the maximum distance of the scan in meters and the resolution also in meters

Prior to MSA10.1.0, the unit was not explicitly clarified. Starting with MSA10.1.0, the units for the parameters *distance* and *resolution* in the rpc were clarified in the model to be in meters.

Prior to initiating an OTDR scan (*start-scan* RPC), the desired OTDR port (*otdr-port*) must be provisioned by setting the *circuit-pack/ports/port-qual=otdr* and specifying the *launch-cable-length* in meters (default=30m) and the *port-direction* based on whether the OTDR is associated with the receive or transmit port (typically the receive direction).

```
+--rw circuit-packs* [circuit-pack-name]
| +---rw ports* [port-name]
| +--rw port-name
| +--rw port-type?
| +--rw port-qual?
| +--rw port-wavelength-type?
| +--rw otdr-port
| | +--rw launch-cable-length?
| | +--rw port-direction?
```

For a ROADM, the *otdr-port* needs to be correlated with a degree in order to execute the OTDR scan (*start-scan* RPC) on the correct interface:

```
+--rw degree* [degree-number]
| +--rw degree-number
| +--rw otdr-port
| | +--rw circuit-pack-name?
| | +--rw port-name?
```

For an ILA, the otdr-port needs to be correlated with a *line-amplifier* in order to execute the OTDR scan (*start-scan* RPC) on the correct interface. Note: since the same port of an ILA can support transmit and receive, the *otdr-direction* needs to be specified as well:

```
+--rw line-amplifier* [amp-number]
| +--rw amp-number
| +--rw otdr-port* [otdr-direction]
| +--rw otdr-direction
| +--rw circuit-pack-name
| +--rw port-name
```

Note: depending on the device implementation, it may be required to run only one scan at a time as the device may only support one OTDR scan file (i.e. the data of the current scan in-progress may be overwritten with the new scan data).

After an OTDR scan is completed, an *otdr-scan-result* notification is generated by the device identifying the OTDR scan data filename (*result-file*). The Open ROADM MSA does not define the filename format of the OTDR scan data (vendor specific). However, the scan data should follow a standard data format as defined by Telcordia SR-4731 [40] so the data can be stored and analyzed by manufacturers of standard OTDR equipment designed to save trace data and analysis information. The OTDR scan data file identified in the *otdr-scan-result* notification (*result-file*) is retrieved using the transfer RPC (see Section 5.1 for details on file transfers). OTDR specification details can be found in the OTDR tab of the Open ROADM MSA Specification [3].

10.6 GENERATE/COLLECT TROUBLE SHOOTING FILE(S)

The *create-tech-info* RPC is used to generate a device trouble shooting file(s) for in-depth failure analysis by the vendor. The format and data included in the log file is vendor specific and not defined by the Open ROADM MSA device model. Vendors need to provide the operator instructions on how to execute the *create-tech-info* RPC for their device (e.g. provide the specific details on command attributes such as *shelf-id*, *log-option*). The output of the *create-tech-info* RPC includes the optional *shelf-id* and filename of the trouble shooting data (*log-file-name*).

```
+---x create-tech-info
| +---w input
| | +---w shelf-id?
| | +---w log-option?
| +--ro output
| +--ro shelf-id?
| +--ro log-file-name?
| +--ro status
| +--ro status-message?
```

YANG sub-tree 74: Create Tech-info RPC

A create-tech-info-notification should be generated when the file is ready for retrieval (refer to Section 5.1 for file transfer details).

```
+---create-tech-info-notification
| +--ro shelf-id?
| +--ro log-file-name?
| +--ro status
| +--ro status-message?
+---n otdr-scan-result
| +--ro status
| +--ro status-message?
| +--ro result-file?
```

YANG sub-tree 75: Create tech info notification

Although the *log-file-name* is optional in the model, vendors should consider this field to be mandatory in the response and *create-tech-info-notification*. This will allow the controller to understand what file will be created and correlate the notification to the RPC request.

10.7 RETRIEVE ROADM CONNECTION PORT TRAIL

The retrieve ROADM connection port trial RPC is used to identify the external and internal ports that the roadm-connection traverses. This is used to establish the internal topology of the service as defined by the roadm-connection. A port identified in the ports list includes the *circuit-pack-name* and *associated port-name*; it may also include the *rack, shelf, slot, and subSlot*.

Starting with MSA12.1.0, the response to the retrieval will also include additional information *is-physical*, *pm-capable* and *alarm-capable*. These information provide the user more information about the ports and their capability.

The *get-connection-port-trail* rpc is shown in YANG sub-tree 76:

```
+---x get-connection-port-trail
| +---w input
| |+---w connection-name string
| +--ro output
|  +--ro status rpc-status
|  +--ro status-message? string
|  +--ro ports* []
|  +--ro rack? string
|  +--ro shelf? string
|  +--ro slot? string
|  +--ro subSlot? string
|  +--ro circuit-pack-name
|  +--ro port-name
|  +--ro is-physical boolean
|  +--ro pm-capable boolean
|  +--ro alarm-capable boolean
```

YANG sub-tree 76: Get connection port trail RPC

In troubleshooting, a lot of times, it is necessary to identify the optical path traverse by a service and also information about the health of the signal. The original *get-connection-port-trail* provides the first part but user has to issue separate commands to retrieve information (e.g. optical power PM) of the signal at each and every ports of the optical path. It is very desirable to be able to capture al the information in one command.

To address such needs, starting with MSA12.1, a new rpc *get-connection-port-trail-optical-pm* is defined to supplement the original rpc. This new rpc:

- maintains the order of the reporting ports as the *get-connection-port-trail*
- requests the device to also include the optical power PM for the interfaces in the same response if the device supports PM for those ports

The new rpc is shown in YANG sub-tree 77

```
+---x get-connection-port-trail-optical-pm
|  +---w input
|  |  +---w connection-name    string
|  +--ro output
|     +--ro status            rpc-status
|     +--ro status-message?   string
|     +--ro port* [circuit-pack-name port-name]
|     |  +--ro circuit-pack-name          ->
↪  /org-openroadm-device/circuit-packs/circuit-pack-name
|     |  +--ro port-name                  -> /org-openroadm-device/circuit-packs[circuit-p
↪  ack-name=current()/../circuit-pack-name]/ports/port-name
|     |  +--ro is-physical                boolean
|     |  +--ro is-aggregate-or-channel    enumeration
|     |  +--ro pm-retrieval-time          ietf-yang-types:date-and-time
|     |  +--ro pm-data* []
|     |     +--ro type                 pm-names-enum
|     |     +--ro extension?           string
|     |     +--ro location             org-openroadm-common-alarm-pm-types:location
|     |     +--ro direction            org-openroadm-common-alarm-pm-types:direction
|     |     +--ro pmParameterValue     pm-data-type
|     |     +--ro pmParameterUnit?     string
|     |     +--ro validity             validity
|     |     +--ro start-time           ietf-yang-types:date-and-time
|     +--ro interface* [interface-name]
|        +--ro interface-name       -> /org-openroadm-device/interface/name
|        +--ro type                 identityref
|        +--ro pm-retrieval-time    ietf-yang-types:date-and-time
|        +--ro pm-data* []
|           +--ro type                 pm-names-enum
|           +--ro extension?           string
|           +--ro location             org-openroadm-common-alarm-pm-types:location
|           +--ro direction            org-openroadm-common-alarm-pm-types:direction
|           +--ro pmParameterValue     pm-data-type
|           +--ro pmParameterUnit?     string
|           +--ro validity             validity
|           +--ro start-time           ietf-yang-types:date-and-time
```

YANG sub-tree 77: Get connection port trail PM RPC

For the response, the device shall return the information in the order as described in Table 26 which summarizes the list of interfaces supporting optical PM and the reporting order in response to the rpc.

EXPRESS	Add	Drop
Port (1) Pre-Amp Channel TX Pre-Amp Aggr TX Post-Amp Aggr RX Post-Amp channel RX	Port (1) SRG Aggr RX (2) SRG Aggr TX (3) Post-Amp Aggr RX Post-Amp channel RX Post-Amp Aggr TX (if applicable)	Port (1) Pre-Amp Channel TX Pre-Amp Aggr TX SRG Aggr RX (3) SRG Aggr TX (2)
Interface (1) OTS RX OMS RX NMC RX NMC TX OMS TX OTS TX	Interface (1) NMC RX (2) NMC TX OMS TX OTS TX	Interface (1) OTS RX OMS RX NMC RX NMC TX (2)
Notes: (1) Assumes the aggregate powers at external ports are reported against OTS/OMS and channel powers against NMC (2) Assumes SRG aggregate power can be reported against the port or NMC. Only one should be reported. (3) Implementations may support one or more internal monitor points on the SRG. If so, these should be reported.		

Table 26: PM Reporting Order

Note that the PM reported includes both aggregate ports and channel ports.

10.8 Recovery from corrupted or inconsistent configuration

There are times where a device could run into situation where its configuration database is corrupted or has inconsistent entries that could lead to operational anamolies. The anamoly might cause the device to lose communication with the ROADM controller (e.g. corruption in its IP address).

To recover from the corrupted/inconsistent configuration for the device, one way is to re-commission the device via the ZTP and restore the last saved, valid database.

As the goal is to recover communication with the device, this recovery process may or may not affect the service currently running on the device It also relies on having a valid database that can be used to restore the running database in the device.

To facilitate more robust recovery, an optical feature “In-Service Database Rebuild” is introduced in MSA12.1 where the device can enter a recover-mode to protect the transport plane while user can restore the running database either via a database restoration or via a re-run of configuration commands by the ROADM controller. This optional feature is described in the section In-Service Database Rebuild”. See Section 10.9.

10.8.1 Recovery via ZTP and Database Restore

This use case describes the procedure for recovering a node when the CPU complex fails and a local secondary database is not available for recovery. In this section, the node is recovered from a previous backup of the configuration.

The procedure to recover the node from a backup database is shown in Figure 92.

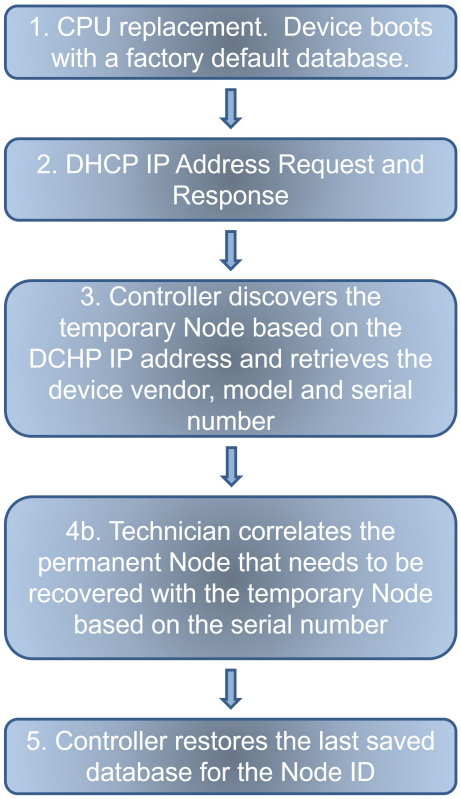

Figure 92: Node Recovery from ZtP and Database Restore of a Previous Backup

Step 1: Main CPU Replacement, System Initialization, and Auto-Provisioning

The field technician will install the CPU replacement and power the equipment. The CPU will initialize with a default database with default values of ‘node-id=openroadm’ and the user account ‘name=openroadm’ and ‘password=openroadm’.

Equipment should be configured by default to the point that communications can be established to the node, either through the external LAN (OAMP) port and/or the internal OSC. Both communication paths (external LAN and internal OSC) should be auto-provisioned, since the interface the device will be accessed on is not known by the device.

Step 2: Temporary IP Address Assignment

The device will request an IP address using DHCP. The device initializes with the DHCP client enabled and requests both an IPv4 and IPv6 address. An IP address (temporary IP address) is assigned from the DHCP server, either an IPv4 or IPv6 address depending on the configuration of the DHCP server.

Note that this temporary IP address will be replaced with a permanent IP address from the database restore operation.

Step 3: Controller Discovery of the Temporary Node

The controller discovers the new IP address assigned by the DHCP server. The controller will login to the device using NETCONF port 830. The SSH authentication uses the Open ROADM MSA default username (name=openroadm) and password (password=openroadm). Once the Open ROADM MSA device is discovered (login is successful), the controller attempts to retrieve the info container data and obtain the device's vendor, model, and serial number (serial-id) attributes. The controller adds the discovered Open ROADM MSA device as a 'temporary' NE.

Step 4: Corrleation of the Permanent Node with the Temporary Node

At this point in the device commissioning process, the controller does not know if this is a new device for ZTP or if this is a device that requires database recovery. The technician connects to the controller to provide this information.

The technician identifies the permanent node that is to be recovered. This permanent node most likely is in a communication loss state. The technician would indicate that this node is to be recovered using the database restore procedure.

The technician will then identify the corresponding temporary discovered NE. This is accomplished by the technician entering the device's physical serial number into the controller. The controller then provides the technician with the temporary NE that matches the serial number. It is expected that only one temporary NE would be returned. However, if the controller responds with multiple discovered temporary NEs that match the supplied serial number, the technician would need to select one based on the vendor, model, and node-type of the node.

The technician would then "commit" the database recovery request. This locks in the association of the permanent Node ID (and the associated backup databases) and the temporary NE discovered by the controller.

Step 5: Database Restore to the Temporary Node

Once the correlation has been made between the permanent Node ID and a temporary discovered NE, the controller will begin the process of recovering the database from the latest saved backup database.

The controller will follow the steps provided in Section 7.2.2 to restore the database.

Following the completion of the database restore operation, the operator may need to manually recover any lost provisioning that was not captured in the database backup. The lost provisioning would be any provisioning changes that took place from the point of the last database backup and the failure of the CPU complex.

10.9 In-Service Database Rebuild (IS-DBR)

10.9.1 Overview

The In-service Database Rebuild (IS-DBR) function is an optional feature offered by the Open ROADM starting in MSA 12.1. This optional feature addresses the rebuilding or restoration of the device configuration from provisioning records when no backup database is available for the restoration.

10.9.2 Use Case

Typically, database backup is performed after provisioning changes, or periodic backup is available to ensure a working database is available. This backup database can be used in case the device needs to be restored to the last configuration. However, it is possible that the backup database might not be available or not up-to-date.

While the chance of the device running into an invalid database might be small, it is highly desirable to have a mechanism to recover the NE's database without interrupting services currently on the NE. This is achieved by re-building the database via replaying the provisioning records that are kept by the Open ROADM controller or higher-level Operation Systems.

Note that for NEs that have a built-in mechanisms that never allows an invalid database condition to occur, this feature might not be needed.

10.9.3 Mechanism

There are several main components for this feature:

- Detection of invalid database and Traffic Preservation
- Notification and Recovery

10.9.3.1 Detection of invalid/missing database and Traffic Preservation

An **invalid database** or **missing database** is defined as a condition where database does not reflect the actual configuration of the NE. For example, when the NE determines the database database is corrupted and the NE could not repair by itself, it is considered having an invalid database. As another example, a NE with service running on it but for whatever reasons, the NE lost completely its running databse or falls back to a "default database" (as when the system first started up) is also considered having an invalid database because the database does not reflect the configuration in the transport plane.

When an **invalid database** condition occurs, the first step is for the NE to detect such condition. The actual mechanism to detect an **invalid database** condition is vendor specific as implementations vary. When an **invalid database** condition is detected, the NE shall reports such condition as alarm to alert the service provider. **The actual mechanism to detect an invalid database is out of scope from this feature description and is vendor implementation specific**.

Note also that when the **invalid database** condition occurs, It is also possible that the situation causes the NE to lose communication with the ROADM controller. For example, a reset of the database or events cause the NE controller to switch to a blank database, which means a lost of the permenant IP address and other communicated related configurations. When this happens, the ROADM controller will detect communication failure to the NE and inform higher level OSS, and eventually leads to a troubleshooting/recovery operation. This falls into the scenario as described in Section 10.8.1. After communication is restored, user can pursue the database restore path. For NE that is capable of the IS-DBR mechanism, it provides an alternative paths to restore the database without interrupting the transport plane.

Upon detection of an invalid/missing database, the NE must preserve traffic currently carried by the transport plane. The device shall enters a *Recover mode*. In this *Recover mode*, the NE ensures changes in configurations (e.g. new edit-config) are not applied to the transport plane. New configurations are applied to the transport plane only when the NE exits the *Recover mode*. The mechanism to clear the *Recover mode* is described in the next sections.

Note: **The actual mechanism to preserve existing traffic is out of scope from this feature description and is vendor implementation specific**.

10.9.3.2 Notification and Recovery

When entering the "Recovery-Mode", the device shall set a new Boolean attribute *recovery-mode* to *true* and shall raise the new alarm *recoveryModeActive* to alert the user to take recovery actions. As mentioned earlier, if communication to the controller is lost, the new alarm will not reach the controller, but it shall be visible after communication is restored.

To clears the NE from the *Recover mode*, the recovery actions can be:

- Case-1: User invokes Database Restore operation – if a backup database is available.

  - After database restore is completed and successful, the device will not apply the restored database until the user explicitly clears the '*Recover mode* via a new RPC to instruct the device to apply the database to the transport plane.

- Case-2: User invokes the configuration RPC reply
  - The database rebuild is accomplished using configuration RPC replay from the ROADM controller.
  - The ROADM controller sends the configuration commands as if the NE was first configured for all the services
  - Then NE accepts the configured data but does not apply them to the transport plane
    - This is similar to the case where the NE supports a running and a candidate database. The IS-DBR mechanism allows NE which does not implement candidate database to recover without interrupting the transport plane.
  - After the RPC replay, the user explicitly clears the *Recover mode* via a new RPC *clear-recovery-mode* to instruct the device to apply the database to the transport plane.

10.9.3.3 Data Model Changes

To support the IS-DBR feature, the following device data model changes are introduced:

- New Attribute in the "Info" container (`org-openroadm-device.yang`):
  - `recovery-mode`: Boolean
  - Read-only, set by the device upon detecting an invalid database/missing database
  - Default value = False
  - Value True = device in recovery mode, waiting for configuration rebuild, traffic is preserved but no operational database
  - Value False = device in normal mode
  - When `recovery-mode=True`:
    - Indicates device is in the traffic preservation mode without a valid database
    - Used as an indication/status for operator to restore the system either via:
      - Database restore (previously backed-up, or offline rebuild)
      - Run the configuration replay
    - When DB-restore or configuration replay completes, user to invoke RPC *clear-recovery-mode* to inform device the recovery is over and to commit the configuration and resume normal operation.

The updated (partial) tree with the new *recovery-mode* leaf in the *info* container is shown in YANG sub-tree 78:

```
+--rw org-openroadm-device
   +--rw info
   |  +--rw node-id?
   ↪   org-openroadm-common-node-types:node-id-type
   |  +--rw node-number?                          uint32
   ....
   |  +--ro current-datetime?                     ietf-yang-types:date-and-time
   |  +--rw lifecycle-state?
   ↪   org-openroadm-common-state-types:lifecycle-state
   |  +--ro recovery-mode?                        boolean {recovery-mode-support}?
```

YANG sub-tree 78: Tree view for recovery-mode

Since the feature is optional, the attribute is defined with: `if-feature recovery-mode-support` declaration.

For this feature, a new probable cause and a new rpc are introduced:

- New Probable Cause:
  - A new probable cause "`recoveryModeActive`" is introduced. It has an enum value of 299.

  - The declaration fragment in org-openroadm-probable-cause.yang is shown in YANG sub-tree 79:

- New RPC:
  - The clearing of the "`recovery-mode`" deserves close user attention to ensure the RPC replay or the database restoration is successful.
  - Upon successful completion of the database rebuild, the rebuilt database shall have the same configuration as the transport plane traffic from the previous database.
  - The user issues the new RPC "`clear-recovery-mode`" to instruct the device to commit the database and apply the configurations to the transport plane. At this point, the database and the transport plane shall be synchronized again.
  - The RPC "`clear-recovery-mode`" does not have any parameters.
    * Device action: The device shall clear "`recoveryModeActive`" alarm and reset the "`recovery-mode`" attribute to "`False`".

```
enum recoveryModeActive {
      value 299;
      description
       "Raised by the device when it enters the recovery-mode due to the detection of an
        ↪  invalid configuration database. During recovery-mode, device preserves and
        ↪  freezes traffic configuration until the user explicitly clears the mode via the
        ↪  clear-recovery-mode RPC.";
    }
```

YANG sub-tree 79: YANG description for recoveryModeActive

Since the feature is optional, the attribute *Recover mode* is defined with: `if-feature recovery-mode-support` declaration.

10.9.4 Example workflow

This section provides the two examples to illustrate the workflow with recovery method using a) database restoration Sec 10.9.4.1; b) using RPC replay Sec 10.9.4.2

10.9.4.1 Clearing Recovery-Mode with DB-Restore

Flow diagram is indicated in Fig 93.

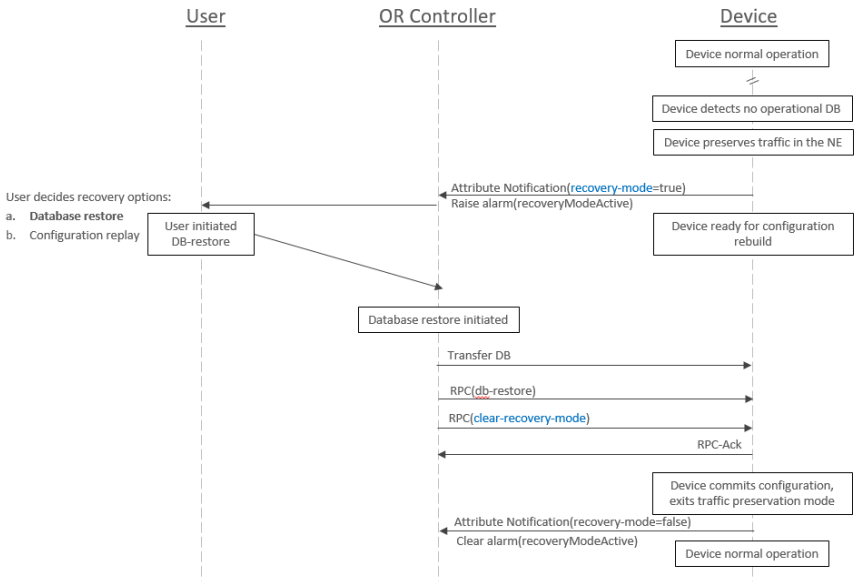

Figure 93: Clearing Recovery-Mode with DB-Restore

10.9.4.2 Clearing Recovery-Mode with RPC replay

Flow diagram is indicated in Fig 94.

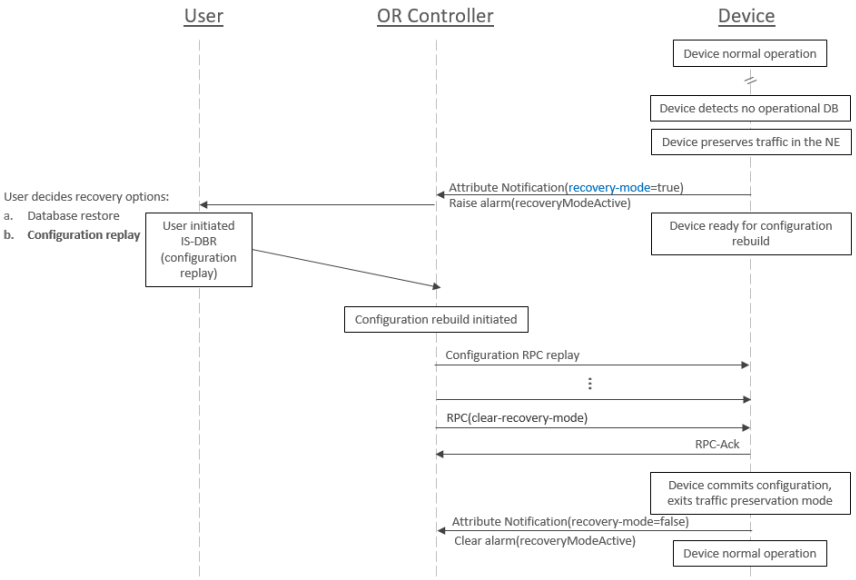

Figure 94: Clearing Recovery-Mode with RPC replay

10.10 Manual Control for Transponder Laser

For maintenance, it is sometimes neccesary for the user to manually turning off the laser of the intended equipment. The model provides the capability to configure if the laser can turn on automatically or stay off for the following optical interfaces:

- OCH
- OTSI
- OTU
- Ethernet

Two new attributes are added to the above mentioned optical interfaces:

- *enable-laser*
- *laser-status*

The *enable-laser* parameter accepts the value *auto* or *off*. When configured to *auto*, the laser can be enabled by the equipement automatically if sufficient provisioning has been completed. When configured to *off*, the laser should be shut off and should not be turned on until the user has configured the value to *auto*.

For backward compatibility, system that is upgraded from MSA version prior to MSA12.1 shall set the value of this new parameter to *auto*, such that the laser is not affected.

The parameter *enable-status* is an operational data where the device returns the current state of the laser. It can have value *unknown*, *on*, or *off*:

- *unknown*: Laser status is unknown. E.g., the CPU cannot determine the status of the laser due to internal communication issues
- *on*: Laser is on
- *off*: Laser is off

When the *enable-laser* is configured to have the value *off*, device shall raise the alarm *enableDisabled* against the concerned interface. The alarm shall be cleared when the *enable-laser* is configured to have the value *auto*. The alarm is intended to remind the user that the laser has been manually disabled. This alarm is not raised if the laser is turned off due to other reasons (e.g. configuration of the interface is incomplete).

The following illustrates the new parameters in the Ethernet interface, but they are also present in the model for OCH, OTSI and OTU interfaces:

```
|  +--rw org-openroadm-eth-interfaces:ethernet
|  |  +--rw org-openroadm-eth-interfaces:speed?                          uint32
|  |  +--rw org-openroadm-eth-interfaces:eth-function?                   eth-function-type
|  |  +--rw org-openroadm-eth-interfaces:fec?                            identityref
|  |  +--rw org-openroadm-eth-interfaces:egress-consequent-action?       enumeration
|  |  +--rw org-openroadm-eth-interfaces:duplex?                         enumeration
|  |  +--rw org-openroadm-eth-interfaces:auto-negotiation?               enumeration
|  |  +--rw org-openroadm-eth-interfaces:enable-laser?                   enumeration
|  |  +--ro org-openroadm-eth-interfaces:laser-status?                   enumeration
|  |  +--ro org-openroadm-eth-interfaces:curr-speed?                     string
|  |  +--ro org-openroadm-eth-interfaces:curr-duplex?                    string
|  |  +--ro org-openroadm-eth-interfaces:max-frame-size?                 uint32
```

YANG sub-tree 80: Ethernet Interface

10.11 Optical control loops with Span Loss changes

To be added to the device white paper

- Span loss changes on existing systems

A Manifest File for Software Download, Database Operations and Timeouts

This section describes the manifest file for software download, database backup, and database restore and timeouts. It includes details of how an Open ROADM device implementation would specify the file transfer, staging, backup, restore, and activation of software loads and database archives. It also provided information about vendor timeout values for retrieve and RPC operations.

The manifest file allows vendors to specify vendor-specific behavior in a common format for software download, database operations and timeouts. This allows the controller to automatically adjust procedures to the vendor specific requirements.

The manifest file is expected to be provided out-of-band (i.e. the file is not provided directly with the device but provided offline by the vendor to the controller).

There may be multiple versions of the manifest file distinguished by the *vendor*, *model*, and *sw-version*. It is expected that the database operations would apply to the current software version running on the Open ROADM device. For software download operations, the *sw-version* would be the software load that the operation would upgrade to (e.g. sw-version = "2.0" and the current software version is "1.2.1"). Note that the *sw-version* is the vendor software version and not the Open ROADM MSA version.

A.1 Software Download Operation

Software download is expected to take place as a series of operations as specified in the manifest file. There should be a manifest file created for the new software load. The manifest file specifies one or more *(*instruction-set). Each *(*instruction-set) would apply to one or more *(*from-sw-version) that would represent the current *(*sw-version) running on the device. This allows different instructions based on the upgrade from different current software versions.

The typical order of operations for software download is as follows:

1. Transfer one or more software files from an SFTP server to the device
2. Stage the software files
   (a) As part of staging, additional software files may be transferred from the SFTP server to the device. The details of what software files to transfer would be vendor-dependent and the file information should be in the software file that was staged. These software files can be in a flat or hierarchical sub-directory structure on the SFTP server.
3. If needed, delete the software file
4. Steps 1-3 may repeat if the device storage does not support transferring all software files at once
5. Activate the software with an optional validation timer. If a validation timer is not provided then the software will auto-commit after the activation completes.
6. Typically, it is expected that the device would reboot at this time (reboot should be automatic)
7. When the device comes up, the operator would evaluate the software load in the validation time (if supported and specified in the manifest file)
   (a) During this time, access to the database may be restricted to allow for rollback. Thus, some or all provisioning activities (e.g., new service activation) may be restricted.
8. If the software load is acceptable, the operator would cancel the validation timer and accept the software load
9. The manifest file can also indicate to automatically accept the software by including the cancel-validation-timer details.
10. The device is now fully functional with the new software release

The operator may also cancel the validation timer and not accept the new software load. Under this case, the device will revert back to the original software load (typically this would result in an additional reboot).

Note: if the device does not support a validation period, then reversion to the original software load may not be supported by the device.

A.2 Database Backup Operation

The database backup operation will create a backup of the current database from the device and transfer it to an SFTP server for offline storage.

The typical order of operations for database backup is as follows:

1. Initiate the database backup operation and specify the filename. The device will create the database backup file on the device for file transfer.
2. When the database file is completed (either as a synchronous command or an asynchronous command with an event notification), the controller would transfer the file from the device to the SFTP server.
3. The controller would then delete the database file from the device.

The database backup manifest file defines two variables: __LOCAL-FILE-PATH and __REMOTE-FILENAME, where:

- __LOCAL-FILE-PATH is the local filename (and path) for the creation of the database backup file
- __REMOTE-FILENAME is the remote filename for storage on the SFTP server to be used in the transfer RPC

It is expected the controller will provide the value of these attributes.

A.3 Database Restore Operation

The database restore operation will restore a previously saved (backup) database file from an SFTP server.

The typical order of operations for database restore is as follows:

1. Transfer the previously backed up database file from the SFTP server to the device
2. Initiate a database restore operation on the device
3. Once the database restore operation completes, the database file can be deleted from the device
4. Activate the database restored file with an optional rollback timer (similar to the software download validation timer). If a rollback timer is not provided then the database will auto-commit after the database activation completes.
5. The device may or may not reboot due to the database restore operation. If a reboot is required, then the reboot should be automatic.
6. When the device comes up, the operator would evaluate the device with the restored database in the rollback time (if supported by the device and specified in the manifest file)
   (a) During this time, access to the database may be restricted to allow for rollback. Thus, some or all provisioning activities (e.g., new service activation) may be restricted.
7. If the database restore is acceptable, the operator would cancel the rollback timer and accept the database
8. The manifest file can also indicate to automatically accept the database by including the cancel-rollback-timer details.
9. The device is now fully functional at the point of the restored database file

The operator may also cancel the rollback timer and not accept the database restore. Under this case, the device will revert back to the database in the state before the database restore (the device may go through another reboot).

Note that if the device does not support a rollback period, then reversion to the previous database may not be supported by the device.

The database restore manifest file defines three variables: __LOCAL-FILE-PATH, __REMOTE-FILENAME, and __NODE-ID-CHECK, where:

- __LOCAL-FILE-PATH is the local filename (and path) where the database file will be stored on the device for restoration
- __REMOTE-FILENAME is the remote database filename on the SFTP server to be used in the transfer-file RPC to the device
- __NODE-ID-CHECK is to have the device validate that the current node-id and the node-id in the database file match to ensure the correct database file is being restored

It is expected the controller will provide the value of these attributes.

A.4 Timeout Information

The controller incorporates timeouts for various operations. The controller will use a fixed timeout value for all vendors. However, this may lead to worst-case timeouts reflecting the lowest common denominator among vendor implementations.

To address this issue, the timeout manifest files allows vendors to publish their specific timeout values corresponding to distinct operations. This file will allow the controller to adjust its timeout settings specifically for that vendor's implementation.

Currently the timeout manifest supports the specification for data retrieval commands and RPC commands.

The controller should document the operations that it supports in the manifest file. The controller should also document the default and maximum timeout values supported for each operation.

Vendors can then provide their timeout values to allow for improved performance (typically earlier detection of failure cases) or if they require additional time above the controller's default value for that operation.

The timeout data in the manifest file is indexed by the controller-vendor and controller-model. The controller-vendor and controller-model would be provided by the controller vendor(s). This allows a device vendor to publish timeout values that work for multiple multiple controller vendors in the same timeout manifest file. This is particularly useful if the supported commands or timeout values may differ based on the controller vendor.

A.5 Manifest File YANG Model

The Manifest file YANG model is included as part of the Open ROADM MSA v11.0 YANG repository (under the common directory).

A.5.1 Software Manifest Tree View

The *sw-manifest* model consists of the following:

```
+--rw sw-manifest!
|  +--rw vendor                  string
|  +--rw model                   string
|  +--rw additional-models*      string
|  +--rw node-type*              org-openroadm-common-node-types:node-types
|  +--rw sw-version              string
|  +--rw openroadm-version?      org-openroadm-common-types:openroadm-version-type
|  +--rw global-async-timeout?   uint16
|  +--rw global-sync-timeout?    uint16
|  +--rw instruction-set* [index]
|     +--rw index                                    uint8
|     +--rw from-sw-version*                         string
|     +--rw is-commit-sw-activate-async?             boolean
|     +--rw cancel-validation-timer-async-timeout?   uint16
|     +--rw cancel-validation-timer-sync-timeout?    uint16
|     +--rw nsa-firmware-delay?                      uint16
|     +--rw resync-after-commit?                     boolean
|     +--rw sw-manifest-commands
|        +--rw sw-manifest-command* [command-order]
|           +--rw command-order              uint8
|           +--rw command                    union
|           +--rw download-file
|           |  +--rw remote-filename    string
|           |  +--rw local-file-path    string
|           |  +--rw timeout?           uint16
|           |  +--rw is-async?          boolean
|           +--rw delete-file
|           |  +--rw filename    string
|           |  +--rw timeout?    uint16
|           |  +--rw is-async?   boolean
|           +--rw sw-stage
|           |  +--rw filename?   string
|           |  +--rw timeout?    uint16
|           |  +--rw is-async?   boolean
|           +--rw sw-activate
|           |  +--rw version                        string
|           |  +--rw validation-timer?              string
|           |  +--rw validation-timer-start-ref?    timer-ref-enum
|           |  +--rw timeout?                       uint16
|           |  +--rw timeout-start-ref?             timer-ref-enum
|           |  +--rw auto-reboot                    uint16
|           |  +--rw login-delay?                   uint16
|           |  +--rw is-async?                      boolean
|           +--rw cancel-validation-timer
|              +--rw wait-time              uint16
|              +--rw timeout?               uint16
|              +--rw is-async?              boolean
|              +--rw resync-after-commit?   boolean
```

A.5.2 Database Backup Manifest Tree View

The *db-backup-manifest* model consists of the following:

```
+--rw db-backup-manifest!
|  +--rw vendor                          string
|  +--rw model                           string
|  +--rw additional-models*              string
|  +--rw node-type*                      org-openroadm-common-node-types:node-types
|  +--rw sw-version?                     string
|  +--rw openroadm-version?              org-openroadm-common-types:openroadm-version-type
|  +--rw global-async-timeout?           uint16
|  +--rw global-sync-timeout?            uint16
|  +--rw db-backup-manifest-commands
|     +--rw db-backup-manifest-command* [command-order]
|        +--rw command-order    uint8
|        +--rw command          union
|        +--rw upload-file
|        |  +--rw remote-filename    string
|        |  +--rw local-file-path    string
|        |  +--rw timeout?           uint16
|        |  +--rw is-async?          boolean
|        +--rw delete-file
|        |  +--rw filename    string
|        |  +--rw timeout?    uint16
|        |  +--rw is-async?   boolean
|        +--rw db-backup
|           +--rw filename?   string
|           +--rw timeout?    uint16
|           +--rw is-async?   boolean
```

A.5.3 Database Restore Manifest Tree View

The *db-restore-manifest* model consists of the following:

```
+--rw db-restore-manifest!
|  +--rw vendor                                  string
|  +--rw model                                   string
|  +--rw additional-models*                      string
|  +--rw node-type*                              org-openroadm-common-node-types:node-types
|  +--rw sw-version?                             string
|  +--rw openroadm-version?
↪  org-openroadm-common-types:openroadm-version-type
|  +--rw global-async-timeout?                   uint16
|  +--rw global-sync-timeout?                    uint16
|  +--rw is-commit-db-activate-async?            boolean
|  +--rw cancel-rollback-timer-async-timeout?    uint16
|  +--rw cancel-rollback-timer-sync-timeout?     uint16
|  +--rw database-init-sync-timeout?             uint16
|  +--rw db-restore-manifest-commands
|     +--rw db-restore-manifest-command* [command-order]
|        +--rw command-order            uint8
|        +--rw command                  union
|        +--rw download-file
|        |  +--rw remote-filename    string
|        |  +--rw local-file-path    string
|        |  +--rw timeout?           uint16
|        |  +--rw is-async?          boolean
|        +--rw delete-file
|        |  +--rw filename    string
|        |  +--rw timeout?    uint16
|        |  +--rw is-async?   boolean
|        +--rw db-restore
|        |  +--rw filename?        string
|        |  +--rw node-id-check?   string
|        |  +--rw timeout?         uint16
|        |  +--rw is-async?        boolean
|        +--rw db-activate
|        |  +--rw rollback-timer?             string
|        |  +--rw rollback-timer-start-ref?   timer-ref-enum
|        |  +--rw timeout?                    uint16
|        |  +--rw timeout-start-ref?          timer-ref-enum
|        |  +--rw auto-reboot                 uint16
|        |  +--rw is-async?                   boolean
|        +--rw cancel-rollback-timer
|           +--rw wait-time    uint16
|           +--rw timeout?     uint16
|           +--rw is-async?    boolean
```

A.5.4 Timeout Manifest Tree View

The *timeout-manifest* model consists of the following:

```
+--rw timeout-manifest!
   +--rw vendor                  string
   +--rw model                   string
   +--rw additional-models*      string
   +--rw node-type*              org-openroadm-common-node-types:node-types
   +--rw sw-version              string
   +--rw openroadm-version?      org-openroadm-common-types:openroadm-version-type
   +--rw global-async-timeout?   uint16
   +--rw global-sync-timeout?    uint16
   +--rw timeout-values* [controller-vendor controller-model]
      +--rw controller-vendor    string
      +--rw controller-model     string
      +--rw retrieval-command* [command-name]
      |  +--rw command-name      string
      |  +--rw timeout?          uint16
      +--rw rpc-command* [command-name]
         +--rw command-name      string
         +--rw timeout?          uint16
         +--rw is-async?         boolean
```

A.6 Manifest File Attributes and Commands

The Manifest commands listed in the Manifest file translate directly to Open ROADM MSA Device RPCs used by the controller to perform the identified software and database operations on the device.

A.6.1 Common Manifest File and Command Attributes

A base set of global manifest file attributes includes:

- **vendor** attribute should match the **device/info/vendor**. It is assumed the **vendor** does not change during a software download or database restore operation.
- **model** attribute should match the **device/info/model**. It is assumed the **model** does not change during a software download or database restore operation.
- **additional-models** attribute lists additional device models to which this manifest file applies. This attribute allows the re-use of a single software repository and software manifest for multiple device models of the same vendor. For details see description of the **model** attribute.
- **node-type** attribute lists the node types to which this manifest applies. A manifest is valid for a device if **device/info/vendor** matches the **vendor** attribute, **device/info/model** matches **model** or **additional-models**, and **device/info/node-type** matches **node-type**. If the **node-type** list attribute is empty, all node types match.
- **sw-version** attribute should match the **device/info/softwareVersion**. For the software download manifest file, the **sw-version** should be set to the value of the **softwareVersion** after the software download completes. For database backup and restore manifest files, the **sw-version** should be set to the currently running **softwareVersion**.
- **openroadm-version** attribute should match the **device/info/openroadm-version**. For the software download manifest file, the **openroadm-version** should be set to the value of the **openroadm-version** after the software download completes. For database backup and restore manifest files, the **openroadm-version** should be set to the currently running **openroadm-version**.
- **global-async-timeout** attribute specifies the default time to wait in seconds for asynchronous commands to complete.
- **global-sync-timeout** attribute specifies the default time to wait in seconds for the RPC response of synchronous commands.

Attributes common to manifest commands include:

- **is-async** attribute indicates whether the operation is asynchronous or synchronous. The controller should determine the success/failure of asynchronous commands based on transient notifications from the device, while the success/-failure of synchronous commands should be based on the RPC response (notification not required).
- **timeout** attribute specifies the time to wait (in seconds) for the RPC response, overriding the **global-async-timeout** (defaults to the **global-async-timeout**) of asynchronous commands and the **global-sync-timeout** (defaults to the **global-sync-timeout**) of synchronous commands.
- **auto-reboot** attribute specifies the time (in seconds) to wait to for the device to reboot. This is the device restart time (e.g. the length of time from device comm loss until the device is ready for login). This timer begins when the controller detects the comm-loss from the device. If the login is not successful when this timer expires, the sw-activate is failed.

  Note: the **auto-reboot** time is an open discussion item for the Open ROADM MSA. The original intent of this attribute was to be the minimum time the controller should wait until logging into the device. However, the attribute was described in such a way that this was the maximum time to complete the reboot.
- **timeout-start-ref** attribute is a qualifier for timeout start applicability. Optional indication if the timeout values will start before or after reboot. Only applicable if **auto-reboot** > 0. Possible values are *before-reboot* or *after-reboot*.
- **wait-time** attribute is used to specify a wait time period after the completion of a software or database activation and prior to canceling an associated **validation-timer** or **rollback-timer**.

A.6.2 Common Manifest Command: Download File

The **download-file** command initiates the device's **transfer RPC** used to transfer software and database file(s) from the SFTP server to the device for activation operations (see section 5.2.4 for details). The controller sends the transfer RPC with *action=download*, *local-file-path=local-file-path*, and uses the *remote-filename* attribute of the manifest command to construct the *remote-file-path* attribute. The *remote-file-path* can include a path that represents the SFTP server's file system structure (e.g. *remote-file-path=sftp://user:password@host[:port]/path/remote-filename*). The file must exist in the specified directory on the SFTP server. Note: the *download-file* command is always asynchronous (*is-async=true*).

A.6.3 Common Manifest Command: Delete File

The **delete-file** command initiates the device's **delete-file RPC** used to delete a software or database file from the device's file system (see section 5.2.6 for details). The *filename* must be specified to ensure the correct file is deleted and may include a path that represents the vendor's device file system structure. Note: the *delete-file* command must be a synchronous command (*is-async=false*).

A.7 Software Manifest File (*sw-manifest*)

The following sub-sections identify the manifest commands used to initiate and activate a software load on the device. The *sw-manifest* file includes an indexed instruction set that allows different upgrade scenarios to have different commands and their order of execution (*command-order*), as well as different attribute values (e.g. timeout, etc.) for each upgrade scenario supported by the device. If specified, the optional *from-sw-version* attribute should be used by the controller to identify the correct instruction set for the desired upgrade scenario (must be provided if multiple instruction sets are specified in the software manifest file).

Note: if the *from-sw-version* is not provided, it is assumed the specified instruction set can be used to upgrade from any software version to the *sw-version* specified in the software manifest file.

The optional *resync-after-commit* attribute indicates whether the controller should resync with the NE following a commit operation. This applies to the case where the commit is automatically executed by the NE, or if the commit is via the **cancel-validation-timer** operation. When using the **cancel-validation-timer** operation, this applies only when *accept=true*.

The optional *is-commit-sw-activate-async* attribute indicates whether the commit operation is asynchronous when it is initiated manually by the **cancel-validation-timer** operation. In case it is asynchronous, the *cancel-validation-timer-async-timeout* specifies the timeout for the **cancel-validation-timer** operation. If not specified, the *global-async-timeout* is used. In case the commit operation is synchronous, the timeout is specified by the *cancel-validation-timer-sync-timeout* attribute, if not specified, the *global-sync-timeout* is used.

The *nsa-firmware-delay* attribute specifies the amount of time after the completion of an auto or manual commit operation for the device to finalize any non-service affecting firmware upgrade. The controller will delay evaluating the circuit-pack-features/components list until after this timer has expired.

Controller processing of the *sw-manifest* file initiates the commands based on the *command-order* and moves to the next command in the following scenarios, otherwise the manifest file is aborted.

Note: the behavior for timeouts (synchronous or asynchronous commands) may depend on the controller implementation per command; it may be considered a successful result, a failed result, or a success/failure based on polling of the device.

A.7.1 Software Staging Command

The **sw-stage** command initiates the device's **sw-stage RPC** used to stage a software download operation after the file(s) has been downloaded to the device (see section A.6.2 for **download-file** command details). The details of what a device does during a staging operation are vendor specific. For example, a device's software load may include a single file or multiple files that may need to be downloaded and subsequently deleted (see section A.6.3 for **delete-file** command details) during the staging operation if the device does not have enough memory for all the required software files that make up the device's software load.

Note: a *filename* for the file to be staged is optional and may or may not be provided in the software manifest file. If a *filename* is not specified, the *sw-stage* RPC would be initiated on the device without a *filename*.

A.7.2 Software Activation Command

The **sw-activate** command initiates the device's **sw-activate RPC** used to activate a device's software load identified by the software version (*version*) in the manifest file command (note: *version* is optional in the *sw-activate* RPC). The details of what a device does during an activation operation is vendor specific. However, a *sw-activate* command must be supported as an asynchronous command, as it is expected the device will initiate an automatic reboot as part of the activation process.

An *auto-reboot* attribute is provided to specify the amount of time (in seconds) to wait for the device to reboot. The *auto-reboot* timer begins when either the response to the **sw-activate RPC** command is received or when comm loss is detected. If the controller fails to login by the time the timer expires, the **sw-activate** operation is considered failed.

The *timeout* is the timeout value for the software activation operation. The value of *timeout* and must be greater than the *auto-reboot* time. If specified, it overrides the *global-async-timeout*. The asynchronous *sw-activate-notification* should be received before this timer expires, otherwise the **sw-activate** operation is considered failed.

The *timeout-start-ref* attribute is a qualifier for indicating when the *timeout* timer will start. Possible values are *before-reboot* or *after-reboot*.

A *validation-timer* attribute (with a value > “00-00-00”) may be specified to allow the operator time to validate the newly activated software load. During the validation time period, the device may lockdown the database and disallow provisioning. This avoids any loss of data or services if the device reverts back to the previous software load (*validation-timer* expires).

The *validation-timer-start-ref* attribute is a qualifier for indicating when the validation timer will start. This attribute should be provided when the *validation-timer* is specified with a value > ”00-00-00”. Possible values are *before-reboot* or *after-reboot*.

The *login-delay* attribute provides the amount of time that the controller will delay before attempting to relogin into the device following the reboot after the **sw-activate** RPC. The timer begins upon receipt of the successful response to the **sw-activate** sw-activate RPC. If not specified, there will be no additional delay and the controller will execute its normal connection retry mechanism.

A *sw-activate-notification* (*sw-active-notification-type=activate*) is expected immediately after the new software load has been activated, but it may be received prior to the reboot (vendor implementation dependent).

Some device implementations may not support a validation time period; in this case, the *validation-timer* attribute should not be specified in the manifest file or specified as *validation-timer = “00-00-00”*. It is expected the device will automatically commit the load with no validation time period. It is also expected, any implementation that does not support a validation time period should immediately generate two notifications (*sw-activate-notification*); one for the activation (*sw-active-notification-type = activate*) and one for the automatic commit (*sw-active-notification-type = commit*).

A.7.3 Cancel Validation Timer Command

The **cancel-validation-timer** command allows the controller to automatically cancel the validation timer and accept the new software load. This command would only be specified in the manifest file if the software activation command indicated a validation period (*validation-timer > “00-00-00”*) and if the operator wants the controller to automatically cancel the validation timer.

In the case of no validation timer, or if the operator wants to manually control the cancelation of the validation timer then this command should not be specified in the manifest file.

The **cancel-validation-timer** command initiates the device’s **cancel-validation-timer RPC** after a specified *wait-time*. The *wait-time* starts from the completion of the software activation. When the *wait-time* timer expires, the controller will issue the *cancel-validation-timer* with *accept=true*. A *sw-activate-notification* with *sw-active-notification-type=commit* is expected after the controller has issued the *cancel-validation-timer (accept=true)* command.

When configuring the manifest file attributes, the device vendor should take care to ensure the *wait-time* is less than the *validation-timer* to avoid an accidental rollback of the software load before the wait-time has expired.

A *sw-activate-notification* with *sw-active-notification-type=cancel* is expected if the validation timer expires or the operator issues a *cancel-validation-timer (accept=false)* command prior to the expiration of the validation timer. In this scenario, the *sw-activate* operation is considered canceled and the device should revert back to the software version prior to the activation.

A.8 Database Backup Manifest File (*db-backup-manifest*)

The following sub-sections identify the manifest commands used to initiate a database backup on the device and the uploading of the generated database file to an external storage location (SFTP server). The *db-backup-manifest* file includes the supported commands and the order the commands are to be executed (command-order).

Controller processing of the *db-backup-manifest* file initiates the commands based on the command-order and moves to the next command in the following scenarios, otherwise the manifest file is aborted.

Note: the behavior for timeouts (synchronous or asynchronous commands) may depend on the controller implementation per command; it may be considered a successful result, a failed result, or a success/failure based on polling of the device.

A.8.1 Database Backup Command

The **db-backup** command initiates the device’s **db-backup [filename] RPC** used to perform a database backup on the device. Even though the Open ROADM MSA device model defines the *filename* as optional, specifying a *filename* should be mandatory for the backup file generated by the device so the file can be deleted from the device after the database backup is completed, if desired (see section A.6.3 for **delete-file** command details).

Depending on the vendor implementation, it is possible a *filename* will not be statically provided in the manifest file, but provided automatically by the controller in the *db-backup* RPC to the device. A *db-backup-notification* is expected after completion of the database backup file generation if the *db-backup* is an async operation.

A.8.2 Upload File Command

The **upload-file** command initiates the device's **transfer RPC** used to transfer a database file from the device to the SFTP server (see section A.7 for details). The controller sends the *transfer* RPC with *action=upload*, *local-file-path=local-file-path*, and uses the *remote-filename* attribute of the manifest command to construct the *remote-file-path* attribute. The *remote-file-path* can include a path that represents the SFTP server's file system structure (e.g. *remote-file-path = sftp://user:password@host[:port]/path/remote-filename*).

A.9 Database Restore Manifest (*db-restore-manifest*)

The following manifest commands are used to initiate and activate a device database restore operation of a database file downloaded from an external storage location (SFTP server). The *db-restore-manifest* file includes the supported commands and the order the commands are to be executed (*command-order*).

Controller processing of the *db-restore-manifest* file initiates the commands based on the *command-order* and moves to the next command in the following scenarios, otherwise the manifest file is aborted.

Note: the behavior for timeouts (synchronous or asynchronous commands) may depend on the controller implementation per command; it may be considered a successful result, a failed result, or a success/failure based on polling of the device.

The optional *is-commit-db-activate-async* attribute indicates whether the database commit operation is asynchronous when it is initiated manually by the **cancel-rollback-timer** operation. In case it is asynchronous, the *cancel-rollback-timer-async-timeout* specifies the timeout for the **cancel-rollback-timer** operation. If not specified, the *global-async-timeout* is used. In case the commit operation is synchronous, the timeout is specified by the *cancel-rollback-timer-sync-timeout* attribute, if not specified, the *global-sync-timeout* is used.

The optional *database-init-sync-timeout* specifies the timeout for the (synchronous) **database-init RPC**. If not specified, the *global-sync-timeout* is used.

A.9.1 Database Restore Command

The **db-restore** command initiates the device's **db-restore RPC** used to restore a downloaded database backup file of the device (see section A.6.2 for download-file command details). Depending on the vendor implementation, it is possible a *filename* will not be statically provided in the manifest file, but provided automatically by the controller in the *db-restore* RPC to the device. The optional *node-id-check* attribute of the manifest command (*nodeIDCheck* attribute in device RPC) specifies whether a *node-id* check is required to validate the database file against the target device. If *node-id-check=true*, the device compares the current *device/node-id* against the *node-id* specified in the database backup file. If the *node-id*'s don't match, the device will fail the *db-restore* operation. A *db-restore-notification* is expected after completion of the database restore operation if the *db-restore* is an async operation.

A.9.2 Database Activation Command

The **db-activate** command initiates the device's **db-activate RPC** used to activate a restored database backup of the device. The details of what a device does during an activation operation is vendor specific.

An *auto-reboot* attribute is provided to specify the amount of time (in seconds) to wait for the device to reboot. A value of "0" should be specified if the device implementation does not require a reboot in order to activate the newly restored database. The *auto-reboot* timer begins when either the response to the **db-activate RPC** command is received or when comm loss is detected. If the controller fails to login by the time the timer expires, the **db-activate** operation is considered failed.

The *timeout* is the timeout value for the database activation operation. The value of *timeout* and must be greater than the *auto-reboot* time. If specified, it overrides the *global-async-timeout*. The asynchronous *db-activate-notification* should be received before this timer expires, otherwise the **db-activate** operation is considered failed.

The *timeout-start-ref* attribute is a qualifier for indicating when the *timeout* timer will start. Possible values are *before-reboot* or *after-reboot*.

A *rollback-timer* attribute (with a value $>$ "00-00-00") may be specified to allow the operator time to validate the newly activated database. During the rollback time period, the device may lockdown the database and disallow provisioning. This avoids any loss of data or services if the device reverts back to the previous database (*rollback-timer* expires). A *db-activate-notification* (*db-active-notification-type = activate*) is expected immediately after the new database has been activated, but it may be received prior to the reboot (vendor implementation dependent).

The *rollback-timer-start-ref* attribute is a qualifier for indicating when the rollback timer will start. This attribute should be provided when the *rollback-timer* is specified with a value $>$ "00-00-00". Possible values are *before-reboot* or *after-reboot*.

Any polling due to a missed *db-activate-notification* (*activate* and/or *commit*) should not take place until after the reboot and re-login.

Some device implementations may not support a rollback time period; in this case, the *rollback-timer* attribute should not be specified in the manifest file or specified as *rollback-timer = "00-00-00"*. It is expected the device will automatically commit the load with no rollback time period. It is also expected, any implementation that does not support a rollback time period should immediately generate two notifications (*db-activate-notification*): one for the activation (*db-active-notification-type = activate*) and one for the automatic commit (*db-active-notification-type = commit*).

Support of the *cancel-rollback-timer* command allows the controller to automatically cancel the validation timer prior to expiring and accept the new database backup (see section A.9.3 for *cancel-rollback-timer* command details).

A.9.3 Cancel Rollback Timer Command

The **cancel-rollback-timer** command allows the controller to automatically cancel the rollback timer and accept the restored database. This command would only be specified in the manifest file if the database activation command indicated a rollback period (*rollback-timer* $>$ *"00-00-00"*) and if the operator wants the controller to automatically cancel the rollback timer.

In the case of no rollback timer, or if the operator wants to manually control the cancelation of the rollback timer, then this command should not be specified in the manifest file.

The **cancel-rollback-timer** command initiates the device's **cancel-rollback-timer RPC** after a specified wait-time. The *wait-time* starts from the completion of the database activation. When the *wait-time* expires, the controller will issue the *cancel-rollback-timer* command with *accept=true*. A *db-activate-notification* with *db-active-notification-type=commit* is expected after the controller has issued the *cancel-rollback-timer (accept=true)* command.

When configuring the manifest file attributes, the device vendor should take care to ensure the *wait-time* is less than the *rollback-timer* to avoid an accidental rollback of the database before the *wait-time* has expired.

A *db-activate-notification* with *db-active-notification-type=cancel* is expected if the rollback timer expires or the operator issues a *cancel-rollback-timer (accept=false)* command prior to the expiration of the rollback timer. In this scenario, the *db-activate* operation is considered canceled and the device should revert back to the database version prior to the activation.

A.10 Timeout Manifest File (*timeout-manifest*)

The following sub-sections identify the manifest commands used to indicate timeout values for retrieval and RPC operations. The information in this file is indexed by *(*controller-vendor) and *(*controller-model). This allows a device vendor to provide timeout values for different controllers in the same file. The controller vendor would provide the values for the *(*controller-vendor) and *(*controller-model).

A.10.1 Timeout for Retrieval Commands

The *(*retrieval-command) list provides a list of retrieval commands and their associated timeouts.

The *(*command-name) identifies the retrieval command. The controller vendors will provide the retrieval commands that the controller supports via the timeout manifest file. The value of this attribute is also determined by the controller vendor.

How the controller vendor publishes this information to the vendor is out of scope of this white paper.

One possible format for the *(*command-name) is the xpath without namespace. E.g., "/org-openroadm-device/interface".

The *(*timeout) would indicate the vendor's timeout for the retrieval of that entity in seconds.

A.10.2 Timeout for RPC Commands

The *(*rpc-command) list provides a list of RPC commands and their associated timeouts.

The *(*command-name) identifies the RPC command. The controller vendors will provide the RPC commands that the controller supports via the timeout manifest file. The value of this attribute is also determined by the controller vendor.

How the controller vendor publishes this information to the vendor is out of scope of this white paper.

One possible format for the *(*command-name) is the RPC name without namespace. E.g., "create-tech-info".

The *(*is-async) attribute indicates if this is an asynchronous operation.

The *(*timeout) would indicate the vendor's timeout for the RPC command in seconds. If the RPC was an asynchronous operation, then the timeout would indicate the total timeout for the operation up to the issuing of the event notification indicating the success or failure of the asynchronous operation.

A.11 Manifest File Examples

A.11.1 Software Manifest File Example

Example software manifest file name, sw-manifest.json, and content:

```
{
    "vendor": "VENDOR",
    "model": "OWB-ROADM",
    "node-type": "rdm",
    "sw-version": "3.0",
    "openroadm-version": "13.1",
    "global-async-timeout": 601,
    "global-sync-timeout": 401,
    "instruction-set": [{
        "index": 1,
        "from-sw-version": ["2.1", "2.2", "2.3"],
        "is-commit-sw-activate-async": "false",
        "cancel-validation-timer-async-timeout": 1000,
        "cancel-validation-timer-sync-timeout": 400,
        "sw-manifest-commands": {
            "sw-manifest-command": [
               {
                    "command-order": 1,
                    "command": "org-openroadm-manifest-file:download-file",
                    "download-file": {
                        "remote-filename": "OWB-UNIT1.PGM",
                        "local-file-path": "/var/ftp/OWB-UNIT1.PGM",
                        "timeout": 600,
                        "is-async": "false"
                    }
                },
                {
                    "command-order": 2,
                    "command": "org-openroadm-manifest-file:sw-stage",
                    "sw-stage": {
                        "filename": "/var/ftp/OWB-UNIT1.PGM",
                        "timeout": 402,
                        "is-async": "true"
                    }
                },
                {
                    "command-order": 3,
                    "command": "org-openroadm-manifest-file:delete-file",
                    "delete-file": {
                        "filename": "/var/ftp/OWB-UNIT1.PGM",
                        "is-async": "false"
                    }
                },
```

```
            {
                "command-order": 4,
                "command": "org-openroadm-manifest-file:download-file",
                "download-file": {
                    "remote-filename": "OWB-UNIT2.PGM",
                    "local-file-path": "/var/ftp/OWB-UNIT2.PGM",
                    "timeout": 600,
                    "is-async": "false"
                }
            },
            {
                "command-order": 5,
                "command": "org-openroadm-manifest-file:sw-stage",
                "sw-stage": {
                    "filename": "/var/ftp/OWB-UNIT2.PGM",
                    "timeout": 402,
                    "is-async": "true"
                }
            },
            {
                "command-order": 6,
                "command": "org-openroadm-manifest-file:delete-file",
                "delete-file": {
                    "filename": "/var/ftp/OWB-UNIT2.PGM",
                    "is-async": "false"
                }
            },
            {
                "command-order": 7,
                "command": "org-openroadm-manifest-file:sw-activate",
                "sw-activate": {
                    "version": "3.0",
                    "validation-timer": "02-00-00",
                    "validation-timer-start-ref": "before-reboot",
                    "timeout": 2400,
                    "timeout-start-ref": "before-reboot",
                    "auto-reboot": 600,
                    "is-async": "true"
                }
            }
        ]
    }
},
```

```
{
"index": 2,
"from-sw-version": ["1.1", "1.2", "1.3"],
"is-commit-sw-activate-async": "false",
"cancel-validation-timer-async-timeout": 1000,
"cancel-validation-timer-sync-timeout": 400,
"sw-manifest-commands": {
    "sw-manifest-command": [
       {
            "command-order": 1,
            "command": "org-openroadm-manifest-file:download-file",
            "download-file": {
                "remote-filename": "OWB-UNIT1.PGM",
                "local-file-path": "/var/ftp/OWB-UNIT1.PGM",
                "timeout": 600,
                "is-async": "true"
            }
        },
        {
            "command-order": 2,
            "command": "org-openroadm-manifest-file:sw-stage",
            "sw-stage": {
                "filename": "/var/ftp/OWB-UNIT1.PGM",
                "timeout": 402,
                "is-async": "true"
            }
        },
        {
            "command-order": 3,
            "command": "org-openroadm-manifest-file:delete-file",
            "delete-file": {
                "filename": "/var/ftp/OWB-UNIT1.PGM",
                "is-async": "false"
            }
        },
        {
            "command-order": 4,
            "command": "org-openroadm-manifest-file:download-file",
            "download-file": {
                "remote-filename": "OWB-UNIT2.PGM",
                "local-file-path": "/var/ftp/OWB-UNIT2.PGM",
                "timeout": 600,
                "is-async": "true"
            }
        },
        {
            "command-order": 5,
            "command": "org-openroadm-manifest-file:sw-stage",
            "sw-stage": {
                "filename": "/var/ftp/OWB-UNIT2.PGM",
                "timeout": 402,
                "is-async": "true"
            }
        },
```

```
                {
                    "command-order": 6,
                    "command": "org-openroadm-manifest-file:delete-file",
                    "delete-file": {
                        "filename": "/var/ftp/OWB-UNIT2.PGM",
                        "is-async": "false"
                    }
                },
                {
                    "command-order": 7,
                    "command": "org-openroadm-manifest-file:sw-activate",
                    "sw-activate": {
                        "version": "3.0",
                        "validation-timer": "02-00-00",
                        "validation-timer-start-ref": "before-reboot",
                        "timeout": 2400,
                        "timeout-start-ref": "before-reboot",
                        "auto-reboot": 600,
                        "is-async": "true"
                    }
                }
            ]
        }
    }]
}
```

A.11.2 Database Backup Manifest File Example

Example database backup manifest file name, db-backup-manifest.json, and content:

```
{
    "vendor": "VENDOR",
    "model": "OWB-ROADM",
    "node-type": "rdm",
    "sw-version": "3.0",
    "openroadm-version": "13.1",
    "global-async-timeout": 900,
    "db-backup-manifest-commands": {
        "db-backup-manifest-command": [{
                "command-order": 1,
                "command": "org-openroadm-manifest-file:db-backup",
                "db-backup": {
                    "filename": "__LOCAL-FILE-PATH",
                    "timeout": 1000,
                    "is-async": "false"
                }
            },
            {
                "command-order": 2,
                "command": "org-openroadm-manifest-file:upload-file",
                "upload-file": {
                    "remote-filename": "__REMOTE-FILE-PATH",
                    "local-file-path": "__LOCAL-FILE-PATH",
                    "timeout": 3000,
                    "is-async": "true"
                }
            },
            {
                "command-order": 3,
                "command": "org-openroadm-manifest-file:delete-file",
                "delete-file": {
                    "filename": "__LOCAL-FILE-PATH",
                    "is-async": "false"
                }
            }
        ]
    }
}
```

A.11.3 Database Restore Manifest File Example

Example database restore manifest file name, db-restore-manifest.json, and content:

```
{
  "vendor": "VENDOR",
  "model": "OWB-ROADM",
  "node-type": "rdm",
  "sw-version": "3.0",
  "openroadm-version": "13.1",
  "global-async-timeout": 900,
  "global-sync-timeout": 135,
  "is-commit-db-activate-async": "false",
  "cancel-rollback-timer-async-timeout": 1000,
  "cancel-rollback-timer-sync-timeout": 140,
  "database-init-sync-timeout": 140,
  "db-restore-manifest-commands": {
    "db-restore-manifest-command": [
          {
        "command-order": 1,
        "command": "org-openroadm-manifest-file:delete-file",
        "delete-file": {
          "timeout": 240,
          "filename": "__LOCAL-FILE-PATH",
          "is-async": "false"
        }
      },
      {
        "command-order": 2,
        "command": "org-openroadm-manifest-file:download-file",
        "download-file": {
          "remote-filename": "__REMOTE-FILENAME",
          "local-file-path": "__LOCAL-FILE-PATH",
          "timeout": 240,
          "is-async": "true"
        }
      },
      {
        "command-order": 3,
        "command": "org-openroadm-manifest-file:db-restore",
        "db-restore": {
          "filename": "__LOCAL-FILE-PATH",
          "node-id-check": "__NODE-ID-CHECK",
          "timeout": 240,
          "is-async": "false"
        }
      },
      {
        "command-order": 4,
        "command": "org-openroadm-manifest-file:db-activate",
        "db-activate": {
          "rollback-timer": "01-00-00",
          "rollback-timer-start-ref": "before-reboot",
          "timeout": 650,
          "timeout-start-ref": "before-reboot",
          "auto-reboot": 600
        }
      }
    ]
  }
}
```

A.11.4 Timeout Manifest File Example

Example timeout manifest file name, timeout-manifest.json, and content:

```
{
  "vendor": "VENDOR",
  "model": "OWB-ROADM",
  "node-type": "rdm",
  "sw-version": "3.0",
  "openroadm-version": "13.1",
  "global-async-timeout": 600,
  "global-sync-timeout": 120,
  "timeout-values": [
    {
      "controller-vendor": "vendor-A",
      "controller-modeld": "model-A",
      "retrieval-command": [
        {
          "command-name": "/org-openroadm-device",
          "timeout": 900
        },
        {
          "command-name": "/org-openroadm-device/info",
          "timeout": 30
        },
      ],
      "rpc-command": [
        {
          "command-name": "create-tech-info",
          "timeout": 600,
          "is-async": "true"
        },
        {
          "command-name": "clear-recovery-mode",
          "timeout": 120,
          "is-async": "false"
        }
      ]
    }
  ]
}
```

B Performance Monitoring (PM) Tables

B.1 Performance Monitoring on MW Ports

Table 27: MW Port Counters

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Optical Power Output (OPOUT-OTS)	MW	OTS-IF	- min max avg	opticalPowerOutput (*) opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	total optical signal power with OSC, includes VOA attenuation
Optical Power Input (OPIN-OTS)	MW	OTS-IF	- min max avg	opticalPowerInput (*) opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	total optical signal power with OSC
Optical Power Output (OPOUT-OMS)	MW	OMS-IF	- min max avg	opticalPowerOutput opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	total optical signal power without OSC, includes VOA attenuation
Optical Power Input (OPIN-OMS)	MW	OMS-IF	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	total optical signal power without OSC
Optical Return Loss (ORL-OTS)	MW	OTS-IF on tx port	-	opticalReturnLoss (*)	At MW port(s) B (including OSC reflection) No tide markings required (e.g. min, max, avg) Either ORL-OTS or ORL-OMS is supported depending upon vendor implementation. Direction, if included, is rx (not tx)
Optical Return Loss (ORL-OMS)	MW	OMS-IF on tx port	-	opticalReturnLoss (*)	At MW port(s) B (not including OSC reflection) No tide markings required (e.g. min, max, avg) Either ORL-OTS or ORL-OMS is supported depending upon vendor implementation. Direction, if included, is rx (not tx)
OSC Optical Power Transmit (OPT-OSC)	MW	OTS-IF	- min max avg	opticalPowerOutputOSC opticalPowerOutputOSCMin opticalPowerOutputOSCMax opticalPowerOutputOSCAvg	OSC Transmit power on MW port Reference:ITU-T G.872-201701 clause 8.4.2

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
OSC Optical Power Receive (OPR-OSC)	MW	OTS-IF	- min max avg	opticalPowerInputOSC opticalPowerInputOSCMin opticalPowerInputOSCMax opticalPowerInputOSCAvg	OSC Receive power on MW port Reference:ITU-T G.872-201701 clause 8.4.2
Optical Channel Power Transmit (OPT-OCH)	MW	NMC-IF (OCH-IF)	- min max avg	opticalPowerOutput (*) opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	Individual optical signal power on MW port Reported on OCH-IF for MSA v1 and on NMC-IF for MSA v2 and later
Optical Channel Power Receive (OPR-OCH)	MW	NMC-IF (OCH-IF)	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	Individual optical signal power on MW port, includes VOA attenuation Reported on OCH-IF for MSA v1 and on NMC-IF for MSA v2 and later
Drift of Optical Channel Power Transmit	MW	NMC-IF	-	opticalPowerOutputDrift	The delta (dB) between the measured optical power output and expected or targeted optical power output. One use case would be the difference between the measured NMC opticalPowerOutput and the target-output-power set by controller.

B.2 Performance Monitoring on Wr Ports

Table 28: Wr Port Counters

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Optical Power Receive (OPR)	Wr	SRG client port	- min max avg	opticalPowerInput (*) opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	total optical signal power on Wr port (from transponder) – single wavelength
Optical Power Transmit (OPT)	Wr	SRG client port	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	total optical signal power on Wr port (to transponder) – multiple wavelengths

B.3 Performance Monitoring on OSC Ports

Table 29: OSC Port Counters

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Errored Block Count (EBC-PCS)	OSC	ETH-IF	-	erroredBlockCount	IEEE 802.3-2018 clauses: • 1GE: 36.2.4.6 • 10GE: 49.2.4.6, 45.2.3.16.4, 45.2.3.22, 49.2.14.2 • 100GE: 82.2.3.5, 45.2.3.16.4, 45.2.3.22, 82.3.1 • 400GE: 119.2.3.4, 45.2.3.16.4, 45.2.3.22, 119.3.1
BIP Error Counter	OSC	ETH-IF	-	codeViolations	Sum of BIP-8 errors on all lanes • RX: IEEE 802.3-2018 clauses 82.2.8, 82.2.15, 45.2.3.46, 45.2.3.47 • TX: IEEE 802.3-2018 clauses 91.5.2.4, 91.6.14, 45.2.1.117, 45.2.1.118
FEC Corrected and Uncorrected Codewords	OSC	ETH-IF	-	fecCorrectedCodewords fecUncorrectedCodewords	IEEE 802.3-2018 clauses: • 100GE: 91.6.9, 91.6.10, 45.2.1.112, 45.2.1.113 • 400GE: 119.3.1, 45.2.3.61, 45.2.3.62
FEC Symbol Errors	OSC	ETH-IF	-	fecSymbolErrors	IEEE 802.3-2018 clauses: • 100GE: 91.6.12, 45.2.1.115, 45.2.1.116 • 400GE: 119.3.1, 45.2.3.57, 45.2.3.58
Errored Seconds (ES-PCS)	OSC	ETH-IF	-	erroredSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1
Severely Errored Seconds (SES-PCS)	OSC	ETH-IF	-	severelyErroredSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1
Unavailable Seconds (UAS-PCS)	OSC	ETH-IF	-	unavailableSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1
In frames (INFRAMES-E)	OSC	ETH-IF	-	inFrames	RMON MIB RFC 2819: etherStatsPkts (RX)

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
In frames errored (INFRAMESERR-E)	OSC	ETH-IF	-	inFramesErrored	RMON MIB RFC 2819: etherStatsCRCAlignErrors + etherStatsFragments + etherStatJabbers
Out frames (OUTFRAMES-E)	OSC	ETH-IF	-	outFrames	RMON MIB RFC 2819: etherStatsPkts (TX)
Errored Seconds Ethernet (ES-E)	OSC	ETH-IF	-	erroredSecondsEthernet	deprecated, should no longer be used
Severely Errored Seconds Ethernet (SES-E)	OSC	ETH-IF	-	severelyErroredSecondsEthernet	deprecated, should no longer be used
Unavailable Seconds Ethernet (UAS-E)	OSC	ETH-IF	-	unavailableSecondsEthernet	deprecated, should no longer be used
OSC Optical Power Receive (OPR-OSC)	OSC	port	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	OSC Receive power on OSC port Reference:ITU-T G.872-201701 clause 8.4.2
OSC Optical Power Transmit (OPT-OSC)	OSC	port	- min max avg	opticalPowerOutput opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	OSC Transmit power on OSC port Reference:ITU-T G.872-201701 clause 8.4.2

B.4 Performance Monitoring on W/OTN Ports

Table 30: W/OTN Port Counters

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Current Output Power	W	port	- min max avg	opticalPowerOutput opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	Single wavelength to Wr
Current Input Power	W	port	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	Single tuned wavelength (reporting against OCH-IF or OTSI-IF is deprecated)
Total Input Power	W	port	- min max avg	totalOpticalPowerInput totalOpticalPowerInputMin totalOpticalPowerInputMax totalOpticalPowerInputAvg	Multiple wavelengths from Wr
pFECcorrErr	W	OTU-IF OTSI-IF	-	preFECCorrectedErrors (*)	Ref) G.798 6.5.1.3
RX Forward Error Correction Correctable Blocks	W	OTU-IF OTSI-IF	-	FECCorrectableBlocks	Count of corrected FEC Blocks, direction=RX
RX Forward Error Correction Uncorrectable Blocks	W	OTU-IF OTSI-IF	-	FECUncorrectableBlocks	Count of uncorrected FEC Blocks, direction=RX
pN_EBC (BIP-8)	W	OTU-IF ODU-IF	-	erroredBlockCount	Ref) G.798 6.5.1.1 location=nearEnd
pN_EBC (BIP-8)	W	ODU-IF	-	erroredBlockCountTCM1-up erroredBlockCountTCM2-up erroredBlockCountTCM3-up erroredBlockCountTCM4-up erroredBlockCountTCM5-up erroredBlockCountTCM6-up erroredBlockCountTCM1-down erroredBlockCountTCM2-down erroredBlockCountTCM3-down erroredBlockCountTCM4-down erroredBlockCountTCM5-down erroredBlockCountTCM6-down	Ref) G.798 6.5.1.1 location=nearEnd

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
pF_EBC (BEI)	W	OTU-IF ODU-IF	-	erroredBlockCount	Ref) G.798 6.5.1.1 location=farEnd
pF_EBC (BEI)	W	ODU-IF	-	erroredBlockCountTCM1-up erroredBlockCountTCM2-up erroredBlockCountTCM3-up erroredBlockCountTCM4-up erroredBlockCountTCM5-up erroredBlockCountTCM6-up erroredBlockCountTCM1-down erroredBlockCountTCM2-down erroredBlockCountTCM3-down erroredBlockCountTCM4-down erroredBlockCountTCM5-down erroredBlockCountTCM6-down	Ref) G.798 6.5.1.1 location=farEnd
pN_delay	W	ODU-IF	-	delay delayTCM1-up delayTCM2-up delayTCM3-up delayTCM4-up delayTCM5-up delayTCM6-up delayTCM1-down delayTCM2-down delayTCM3-down delayTCM4-down delayTCM5-down delayTCM6-down	Ref) G.798: Number of frames between a DMValue toggle event and the received DMp signal value toggle event location=nearEnd
pIAE	W	OTU-IF	-	incomingAlignmentError	Ref) G.798 6.2.6.10
pBIAE	W	OTU-IF	-	backwardIncomingAlignmentError	Ref) G.798 6.2.6.11
Errored Seconds (ES)	W	OTU-IF ODU-IF	-	erroredSeconds	

MSA PM	Port	Reported Against	Tide Marks	YANG Type (*) = used by controller	Comment
Errored Seconds (ES)	W	ODU-IF	-	erroredSecondsTCM1-up erroredSecondsTCM2-up erroredSecondsTCM3-up erroredSecondsTCM4-up erroredSecondsTCM5-up erroredSecondsTCM6-up erroredSecondsTCM1-down erroredSecondsTCM2-down erroredSecondsTCM3-down erroredSecondsTCM4-down erroredSecondsTCM5-down erroredSecondsTCM6-down	
Severely Errored Seconds (SES)	W	OTU-IF ODU-IF	-	severelyErroredSeconds	
Severely Errored Seconds (SES)	W	ODU-IF	-	severelyErroredSecondsTCM1-up severelyErroredSecondsTCM2-up severelyErroredSecondsTCM3-up severelyErroredSecondsTCM4-up severelyErroredSecondsTCM5-up severelyErroredSecondsTCM6-up severelyErroredSecondsTCM1-down severelyErroredSecondsTCM2-down severelyErroredSecondsTCM3-down severelyErroredSecondsTCM4-down severelyErroredSecondsTCM5-down severelyErroredSecondsTCM6-down	
Unavailable Seconds (UAS)	W	OTU-IF ODU-IF	-	unavailableSeconds	

MSA PM	Port	Reported Against	Tide Marks	YANG Type (*) = used by controller	Comment
Unavailable Seconds (UAS)	W	ODU-IF	-	unavailableSecondsTCM1-up unavailableSecondsTCM2-up unavailableSecondsTCM3-up unavailableSecondsTCM4-up unavailableSecondsTCM5-up unavailableSecondsTCM6-up unavailableSecondsTCM1-down unavailableSecondsTCM2-down unavailableSecondsTCM3-down unavailableSecondsTCM4-down unavailableSecondsTCM5-down unavailableSecondsTCM6-down	

B.5 Performance Monitoring on W/ETH Ports

Table 31: W/ETH Port Counters

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Current Output Power	W	port	- min max avg	opticalPowerOutput opticalPowerOutputMin opticalPowerOutputMax opticalPowerOutputAvg	
Current Input Power	W	port	- min max avg	opticalPowerInput opticalPowerInputMin opticalPowerInputMax opticalPowerInputAvg	(reporting against OCH-IF is deprecated)
Errored Block Count (EBC-PCS)	W	ETH-IF	-	erroredBlockCount	IEEE 802.3-2018 clauses: • 1GE: 36.2.4.6 • 10GE: 49.2.4.6, 45.2.3.16.4, 45.2.3.22, 49.2.14.2 • 100GE: 82.2.3.5, 45.2.3.16.4, 45.2.3.22, 82.3.1 • 400GE: 119.2.3.4, 45.2.3.16.4, 45.2.3.22, 119.3.1
BIP Error Counter	W	ETH-IF	-	codeViolations	Sum of BIP-8 errors on all lanes • RX: IEEE 802.3-2018 clauses 82.2.8, 82.2.15, 45.2.3.46, 45.2.3.47 • TX: IEEE 802.3-2018 clauses 91.5.2.4, 91.6.14, 45.2.1.117, 45.2.1.118
FEC Corrected and Uncorrected Codewords	W	ETH-IF	-	fecCorrectedCodewords fecUncorrectedCodewords	IEEE 802.3-2018 clauses: • 100GE: 91.6.9, 91.6.10, 45.2.1.112, 45.2.1.113 • 400GE: 119.3.1, 45.2.3.61, 45.2.3.62
FEC Symbol Errors	W	ETH-IF	-	fecSymbolErrors	IEEE 802.3-2018 clauses: • 100GE: 91.6.12, 45.2.1.115, 45.2.1.116 • 400GE: 119.3.1, 45.2.3.57, 45.2.3.58
Errored Seconds (ES-PCS)	W	ETH-IF	-	erroredSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1

MSA PM	**Port**	**Reported Against**	**Tide Marks**	**YANG Type** (*) = used by controller	**Comment**
Severely Errored Seconds (SES-PCS)	W	ETH-IF	-	severelyErroredSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1
Unavailable Seconds PCS (UAS-PCS)	W	ETH-IF	-	unavailableSeconds	ITU-T G.7710-2007 clauses 10.1.5, 10.2.1
In frames (INFRAMES-E)	W	ETH-IF	-	inFrames	RMON MIB RFC 2819: etherStatsPkts (RX)
In frames errored (INFRAMESERR-E)	W	ETH-IF	-	inFramesErrored	RMON MIB RFC 2819: etherStatsCRCAlignErrors + etherStatsFragments + etherStatJabbers
Out frames (OUTFRAMES-E)	W	ETH-IF	-	outFrames	RMON MIB RFC 2819: etherStatsPkts (TX)

B.6 Unmapped PM Counters

Table 32: Unmapped PM Counters

MSA PM	Port	Reported Against	Tide Marks	YANG Type (*) = used by controller	Comment
				bitErrorRate BIPErrorCounter preFECbitErrorRate defectSeconds localFaultSeconds remoteFaultSeconds partialRateDiscard protectionSwitchingCount protectionSwitchingDuration	
				delayTime delayTimeTCMn-up delayTimeTCMn-down erroredBlockCountRatio erroredBlockCountRatioTCMn-up erroredBlockCountRatioTCMn-down severelyErroredSecondRatio severelyErroredSecondRatioTCMn-up severelyErroredSecondRatioTCMn-down backwardIncomingAlignmentErrorTCMn-up backwardIncomingAlignmentErrorTCMn-down incomingAlignmentErrorTCMn-up incomingAlignmentErrorTCMn-down	
				gfpCmfFrames gfpErroredFrames gfpErroredSecond gfpSeverelyErroredSecond gfpUnavailableSecond	
				laserBiasCurrent opticalInterconnectLoss opticalSignalNoiseRatio polarizationDependentLoss polarizationModeDispersion chromaticDispersion temperature	

C Alarm Tables

C.1 Alarms on W Ports

Table 33: W Port Alarms

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
OTU - LOS 100GE Signal Det	port OTSI-IF OCH-IF ETH-IF	portLossOfLight lossOfSignal lossOfSignal linkDown		OTN: input power drop ETH: Ref) IEEE 802.3 86.2 maps to physical port level LOS Ethernet - linkDown at ETH-IF, LOS at sub-ports; LOS at parent port if all sub-ports LOS (sub-port LOS is suppressed when port level LOS is raised. Sub-port LOS is cleared and not retrievable when port level LOS is raised.) OCH LOS at transponder only (specific to tuned wavelength.) Reported on both the network and client OCH; the network OCH alarm is used by the controller.
LOF	OTU-IF	lossOfFrame		Ref) G.798 6.2.5.1
LOM	OTU-IF OTSI-IF	lossOfMultiframe		Ref) G.798 6.2.5.2
RDI	OTSI-IF	remoteDefectIndication		Ref) G.798 15.2.1.2 dRDI if the extracted Remote PHY Fault is 1 (FlexO-x TT Sk)
BDI	OTU-IF ODU-IF	backwardsDefectIndication		Ref) G.798 6.2.6.6

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
BDI	ODU-IF	backwardsDefectIndicationTCM1-up backwardsDefectIndicationTCM2-up backwardsDefectIndicationTCM3-up backwardsDefectIndicationTCM4-up backwardsDefectIndicationTCM5-up backwardsDefectIndicationTCM6-up backwardsDefectIndicationTCM1-down backwardsDefectIndicationTCM2-down backwardsDefectIndicationTCM3-down backwardsDefectIndicationTCM4-down backwardsDefectIndicationTCM5-down backwardsDefectIndicationTCM6-down		Ref) G.798 6.2.6.6
Group ID Mismatch	OTSIG-IF	groupIdMismatch		Ref) G.798-201912 15.3.1.2
FlexO Map Mismatch	OTSIG-IF	flexoMapMismatch		Ref) G.798-201912 15.3.1.2
Loss of Frame and Loss of Multiframe	OTSIG-IF	lossOfFrameAndLossOfMultiframe		Ref) G.798-201912 15.3.1.2
Loss of Lane Alignment	OTSI-IF OTSIG-IF	lossOfLaneAlignment		Ref) G.798-201912 15.3.1.2, 16.7.2
Loss of Signal Payload	OTSI-IF	lossOfSignalPayload		Ref) G.798-201912 16.7.2
DEG	OTU-IF ODU-IF	degradedDefect		Ref) G.798 6.2.3
DEG	ODU-IF	degradedDefectTCM1-up degradedDefectTCM2-up degradedDefectTCM3-up degradedDefectTCM4-up degradedDefectTCM5-up degradedDefectTCM6-up degradedDefectTCM1-down degradedDefectTCM2-down degradedDefectTCM3-down degradedDefectTCM4-down degradedDefectTCM5-down degradedDefectTCM6-down		Ref) G.798 6.2.3
TTI	OTU-IF ODU-IF	trailTraceIdentifierMismatch		Ref) G.798 6.2.2.1

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
TTI	ODU-IF	trailTraceIdentifierMismatchTCM1-up trailTraceIdentifierMismatchTCM2-up trailTraceIdentifierMismatchTCM3-up trailTraceIdentifierMismatchTCM4-up trailTraceIdentifierMismatchTCM5-up trailTraceIdentifierMismatchTCM6-up trailTraceIdentifierMismatchTCM1-down trailTraceIdentifierMismatchTCM2-down trailTraceIdentifierMismatchTCM3-down trailTraceIdentifierMismatchTCM4-down trailTraceIdentifierMismatchTCM5-down trailTraceIdentifierMismatchTCM6-down		Ref) G.798 6.2.2.1
AIS	OTU-IF ODU-IF	alarmIndicationSignal		Ref) G.798 6.2.6.3.2
AIS	ODU-IF	alarmIndicationSignalTCM1-up alarmIndicationSignalTCM2-up alarmIndicationSignalTCM3-up alarmIndicationSignalTCM4-up alarmIndicationSignalTCM5-up alarmIndicationSignalTCM6-up alarmIndicationSignalTCM1-down alarmIndicationSignalTCM2-down alarmIndicationSignalTCM3-down alarmIndicationSignalTCM4-down alarmIndicationSignalTCM5-down alarmIndicationSignalTCM6-down		Ref) G.798 6.2.6.3.2
OCI	ODU-IF	openConnectionIndication		Ref) G.798 6.2.6.8

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
OCI	ODU-IF	openConnectionIndicationTCM1-up openConnectionIndicationTCM2-up openConnectionIndicationTCM3-up openConnectionIndicationTCM4-up openConnectionIndicationTCM5-up openConnectionIndicationTCM6-up openConnectionIndicationTCM1-down openConnectionIndicationTCM2-down openConnectionIndicationTCM3-down openConnectionIndicationTCM4-down openConnectionIndicationTCM5-down openConnectionIndicationTCM6-down		Ref) G.798 6.2.6.8
LCK	ODU-IF	lockedDefect		Ref) G.798 6.2.6.9
LCK	ODU-IF	lockedDefectTCM1-up lockedDefectTCM2-up lockedDefectTCM3-up lockedDefectTCM4-up lockedDefectTCM5-up lockedDefectTCM6-up lockedDefectTCM1-down lockedDefectTCM2-down lockedDefectTCM3-down lockedDefectTCM4-down lockedDefectTCM5-down lockedDefectTCM6-down		Ref) G.798 6.2.6.9
LTC	ODU-IF	lossOfTandemConnectionTCM1-up lossOfTandemConnectionTCM2-up lossOfTandemConnectionTCM3-up lossOfTandemConnectionTCM4-up lossOfTandemConnectionTCM5-up lossOfTandemConnectionTCM6-up lossOfTandemConnectionTCM1-down lossOfTandemConnectionTCM2-down lossOfTandemConnectionTCM3-down lossOfTandemConnectionTCM4-down lossOfTandemConnectionTCM5-down lossOfTandemConnectionTCM6-down		Ref) G.798 6.2.1.4

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
PLM	ODU-IF	payloadMismatch		Ref) G.798 6.2.4.1
MSIM	ODU-IF	multiplexStructureIdentifierMismatch		G.798 6.2.9 (dMSIM)
LOOMFI	ODU-IF	lossOfOmfIndication		G.798 14.3.10.2 figure 14-73 (dLOOMFI)
CSF	ODU-IF	clientSignalFailDefect		Ref) G.798 6.2.10
Loss of FEC Alignment	ETH-IF	lossOfFECAlignment		Ref) IEEE 802.3-2018 45.2.1.111.2, 91.
PCS Loss of Sync (Rcv)	ETH-IF	lossOfSynchronization	rx	Ref) IEEE 802.3-2018 45.2.3.15.1
RX PCS High BER	ETH-IF	highBER	rx	Ref) IEEE 802.3-2018 45.2.3.15.4, 82.
RX PCS High SER	ETH-IF	highSER	rx	Ref) IEEE 802.3-2018 91.6.5, 119.2.5.3
Loss of Alignment (Rcv)	ETH-IF	lossOfAlignment	rx	Ref) IEEE 802.3-2018 45.2.3.23.1, 83., 119.
RX Local Fault	ETH-IF	localFault	rx	Ref) IEEE 802.3-2018 81.3.4, 117.3
RX Remote Fault	ETH-IF	remoteFault	rx	Ref) IEEE 802.3-2018 81.3.4, 117.3
PCS Loss of Sync (Tx)	ETH-IF	lossOfSynchronization	tx	Ref) IEEE 802.3-2018 45.2.3.15.1
TX PCS High BER	ETH-IF	highBER	tx	Ref) IEEE 802.3-2018 45.2.3.16.2
Loss of Alignment (Tx)	ETH-IF	lossOfAlignment	tx	Logical OR of lanes is notified Ref) IEEE 802.3-2018 45.2.3.23.1, 83., 119.
TX Local Fault	ETH-IF	localFault	tx	Ref) IEEE 802.3-2018 81.3.4
TX Remote Fault	ETH-IF	remoteFault	tx	Ref) IEEE 802.3-2018 81.3.4
Local Degraded SER	ETH-IF	localDegradedSER		Ref) IEEE 802.3-2018 45.2.3.60.1, 119.
Remote Degraded SER	ETH-IF	remoteDegradedSER		Ref) IEEE 802.3-2018 45.2.3.60.2, 119.
PCS FEC Degraded SER	ETH-IF	fecDegradedSER		Ref) IEEE 802.3-2018 119. Only default threshold used.
Loss Of RC Blocks	ETH-IF	lossOfRcBlocks		Ref) G.798 14.3.20.2 dLRC (loss of Rc blocks) for 200G/400G ETH based on ODUflexP to ETCy adaptation sink function using BMP (ODUflexP/ETCy A Sk)

C.2 Alarms on MW Mux (Out) Ports

Table 34: MW Port (Out) Alarms

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
Automatic Shutoff/ Automatic Laser Shutdown	OMS-IF	automaticLaserShutdown		Shutdown of amplifier laser due to safety. OSC will never be shut off.
Automatic Shutoff Disabled	OMS-IF	automaticShutoffDisabled		RPC was triggered to shut off laser safety, standing alarm.
Automatic Power Reduction Active (optional)	port	automaticPowerReduction		During high reflection event, power is reduced, but not shut off.
High Reflection/ Low Optical Return Loss	OTS-IF OMS-IF	reflectionTooHigh		reflectionTooHigh on OTS-IF is used by the controller. Reported on OTS-IF or OMS-IF, based on vendor implementation (OTS-IF if OSC is included in the measurement; OMS-IF otherwise.)
Loss of Signal	OCH-IF	lossOfSignal	tx	May be used by the controller. Not masked since there is no LOS on OTS(tx). Optional for device to report.

C.3 Alarms on MW Demux (In) Ports

Table 35: MW Port (In) Alarms

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
Loss of Signal	OTS-IF	lossOfSignal	rx	LOS on OTS includes OSC and OMS. LOS on OTS masks LOS on OMS but does not mask LOS on OSC. LOS on OTS is used by the controller.
Loss of Signal	OMS-IF	lossOfSignal	rx	LOS on OMS
Loss of Signal	OTS-IF	lossOfSignalOSC	rx	LOS on OSC
Loss of Signal	OCH-IF	lossOfSignal	rx	LOS on OCH Masked by LOS on OTS(rx) Optional for device to report
Link Down	ETH-IF	linkDown	rx	Link Down on OSC
Far End Fault Indication	ETH-IF	farEndFaultIndication	rx	Far End Fault Indication on 100M OSC Aligns with MSA Laser Safety ”EthFEFI” alarm
Low Optical Power Warning (Optical Power Degraded)	OTS-IF	opticalPowerDegraded	rx	
Optical Line Fail	OTS-IF	opticalLineFail	rx NEND	Combines OMS LOS and linkDown on OSC into one alarm. Optical Line Fail masks OMS LOS but does not mask linkDown on OSC and Ethernet linkdown.
Optical Interconnect Loss Too High	OTS-IF	opticalInterconnectLossHigh	rx NEND	Interconnection loss high alarm for fiber

C.4 Alarms on Wr Mux (In) Ports

Table 36: Wr Port (In) Alarms

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
Loss of Signal	port	portLossOfLight	rx	
Optical Power Degraded	port	opticalPowerDegraded	rx	

C.5 Alarms not currently mapped in Optical Specification

Table 37: Unmapped Alarms

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
		administrativeDown administrativeTesting		
		automaticSwitchDueToWTR automaticSwitchAwayFromProtectDueToSD automaticSwitchAwayFromProtectDueToSF automaticSwitchAwayFromWorkingDueToSD automaticSwitchAwayFromWorkingDueToSF		
		certificateNotInstalled circuitPackActivateFailed		
		cmfForwardDefectIndication		(dCSF G.806:6.2.6.4.1) is raised when a GFP frame with correct tHEC, with aPTI='100' and a valid UPI code 0x04 (G.7041: Table 6-4) is received.
		cmfLossOfSignal		(dCSF G.806:6.2.6.4.1) is raised when a GFP frame with correct tHEC, with aPTI='100' and a valid UPI code 0x01 (G.7041: Table 6-4) is received.
		cmfLossOfSync		(dCSF G.806:6.2.6.4.1) is raised when a GFP frame with correct tHEC, with aPTI='100' and a valid UPI code 0x02 (G.7041: Table 6-4) is received.
		cmfReverseDefectIndication		(dCSF G.806:6.2.6.4.1) is raised when a GFP frame with correct tHEC, with aPTI='100' and a valid UPI code 0x02 (G.7041: Table 6-4) is received.

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
		createTechInfoInProgress databaseCorruption databaseLocked databaseRollbackTimerInProgress databaseVersionMismatch diskFull equipmentFault equipmentInterconnectFailure equipmentLedOn equipmentMiscabledConnection equipmentMismatch equipmentRemoved equipmentWarmup facilityLoopbackActive facilityLoopback2Active facilityTestsignalActive fanCoolingFail	
		farEndFaultIndication		Ref) IEEE 802.3-2018 24.3.2.1
		firmwareDownloadOrActivationFailure firmwareInitInProgress		
		firmwareVersionMismatch		This alarm is raised when the new version of software is no longer compatible with the version of firmware running on the circuit pack.
		forcedSwitchAwayFromProtect forcedSwitchAwayFromWorking forwardDefectIndication		
		gfpLossOfFrameDelineation		(dLFD G.806:6.2.5.2) is raised when the frame delineation process is not in the 'SYNC' state. (ITU-T G.7041 6.3.1)
		incompatibleFirmware		This alarm is raised when the device is provisioned with some feature which requires the firmware on the circuit pack to be updated.
		incompatibleHardware		This alarm is raised when hardware does not match with the application (e.g. FEC type)

MSA Alarm	Reported Against	YANG Probable Cause	Severity DIRN LOCN	Comment
		lampTest lldpFail lockoutOfProtection lossOfDatabaseRedundancy		
		apsProtocolError apsConfigurationMismatch manualSwitchAwayFromProtect manualSwitchAwayFromWorking		
		omsPowerOutOfSpecificationHigh omsPowerOutOfSpecificationLow oscPowerOutOfSpecificationHigh oscPowerOutOfSpecificationLow otdrScanInProgress otsSpanlossPowerOutOfSpecificationHigh payloadMissingIndication powerOutOfRangeHigh powerOutOfRangeLow powerOutOfSpecificationHigh powerOutOfSpecificationLow powerProblemA powerProblemB secondarySoftwareCorruption serverSignalFail shelfProvisioningMode softwareReset softwareStageInProgress softwareSubsystemFailed softwareSyncInProgress softwareValidateInProgress softwareVersionMismatch sysNameChanged sysNtpNotSynchronized terminalLoopbackActive terminalTestsignalActive totalpowerOutOfSpecificationHigh totalpowerOutOfSpecificationLow		

References

[1] "RFC 7950: The YANG 1.1 Data Modeling Language." https://datatracker.ietf.org/doc/html/rfc7950. Accessed: 2022-03-11.

[2] "OpenROADM." http://openroadm.org. Accessed: 2022-03-11.

[3] "OpenROADM MSA W-Port Digital Specification v6.0." https://github.com/OpenROADM/OpenROADM_MSA_Public/wiki/files/20231207_open-roadm_msa_specification_ver6.0.xlsx. Accessed: 2024-01-30.

[4] "IEEE 802.3-2018 IEEE Standard for Ethernet." https://standards.ieee.org/ieee/802.3/7071/. Accessed: 2022-03-31.

[5] "ITU-T G.7041/Y.1303 (08/2016) Generic framing procedure." https://www.itu.int/rec/T-REC-G.7041/. Accessed: 2022-03-11.

[6] "Open All-Photonic Network Functional Architecture." https://iowngf.org/wp-content/uploads/formidable/21/IOWN-GF-RD-Open_APN_Functional_Architecture-2.0.pdf. Accessed: 2024-03-15.

[7] "ITU-T G.709/Y.1331 Cor. 2 (11/2022) Interfaces for the optical transport network." https://www.itu.int/rec/T-REC-G.709-202211-I!Cor2. Accessed: 2024-01-30.

[8] "ITU-T G.873.1 Amd. 1 (02/2022) Optical transport network: Linear protection." https://www.itu.int/rec/T-REC-G.873.1-202202-I!Amd1. Accessed: 2024-01-29.

[9] "ITU-T G.709.3/Y.1331.3 Amd. 1 (11/2022) Flexible OTN long reach interfaces." https://www.itu.int/rec/T-REC-G.709.3-202211-I!Amd1. Accessed: 2024-01-30.

[10] "ITU-T G.842 (04/1997) Interworking of SDH network protection architectures." https://www.itu.int/rec/T-REC-G.842-199704-I. Accessed: 2024-01-30.

[11] "RFC 6241: Network Configuration Protocol (NETCONF)." https://datatracker.ietf.org/doc/html/rfc6241. Accessed: 2022-03-11.

[12] "ITU-T G.709.1/Y.1331.1 Amd. 4 (08/2023) Flexible OTN short reach interfaces." https://www.itu.int/rec/T-REC-G.709.1-202308-P!Amd4. Accessed: 2024-01-30.

[13] "ITU-T G.7712 (08/2019) Architecture and specification of data communication network." https://www.itu.int/rec/T-REC-G.7712-201908-I. Accessed: 2022-06-08.

[14] "RFC 1661: The Point-to-Point Protocol (PPP)." https://datatracker.ietf.org/doc/html/rfc1661. Accessed: 2022-06-08.

[15] "RFC 1332: The PPP Internet Protocol Control Protocol (IPCP)." https://datatracker.ietf.org/doc/html/rfc1332. Accessed: 2022-06-08.

[16] "RFC 5072: IP Version 6 over PPP." https://datatracker.ietf.org/doc/html/rfc5072. Accessed: 2022-11-23.

[17] "RFC 1662: PPP in HDLC-like Framing." https://datatracker.ietf.org/doc/html/rfc1662. Accessed: 2022-06-08.

[18] "ITU-T G.798 (09/2023) Characteristics of optical transport network hierarchy equipment functional blocks." https://www.itu.int/rec/T-REC-G.798-202309-P. Accessed: 2024-01-29.

[19] "RFC 1570: PPP LCP Extensions." https://datatracker.ietf.org/doc/html/rfc1570. Accessed: 2022-06-08.

[20] "RFC 3241: Robust Header Compression (ROHC) over PPP." https://datatracker.ietf.org/doc/html/rfc3241. Accessed: 2022-11-23.

[21] "RFC 2472: IP Version 6 over PPP." https://datatracker.ietf.org/doc/html/rfc2472. Accessed: 2022-06-08.

[22] "RFC 5172: Negotiation for IPv6 Datagram Compression Using IPv6 Control Protocol." https://datatracker.ietf.org/doc/html/rfc5172. Accessed: 2022-11-23.

[23] "RFC 8064: Recommendation on Stable IPv6 Interface Identifiers." https://datatracker.ietf.org/doc/html/rfc8064. Accessed: 2022-11-23.

[24] "RFC 7277: A YANG Data Model for IP Management." https://datatracker.ietf.org/doc/html/rfc7277. Accessed: 2022-06-08.

[25] "RFC 1027: Using ARP to Implement Transparent Subnet Gateways." https://datatracker.ietf.org/doc/html/rfc1027. Accessed: 2022-11-23.

[26] "RFC 4861: Neighbor Discovery for IP version 6 (IPv6)." https://datatracker.ietf.org/doc/html/rfc4861. Accessed: 2022-11-23.

[27] “RFC 1918: Address Allocation for Private Internets.” https://datatracker.ietf.org/doc/html/rfc1918. Accessed: 2022-06-08.

[28] “RFC 3927: Dynamic Configuration of IPv4 Link-Local Addresses.” https://datatracker.ietf.org/doc/html/rfc3927. Accessed: 2022-06-08.

[29] “RFC 6242: Using the NETCONF Protocol over Secure Shell (SSH).” https://datatracker.ietf.org/doc/html/rfc6242. Accessed: 2022-03-11.

[30] “RFC 4742: Using the NETCONF Configuration Protocol over Secure SHell (SSH).” https://datatracker.ietf.org/doc/html/rfc4742. Accessed: 2022-03-11.

[31] “RFC 8071: NETCONF Call Home and RESTCONF Call Home.” https://datatracker.ietf.org/doc/html/rfc8071. Accessed: 2022-06-08.

[32] “RFC 1122: Requirements for Internet Hosts – Communication Layers.” https://datatracker.ietf.org/doc/html/rfc1122. Accessed: 2022-06-08.

[33] “RFC 5277: NETCONF Event Notifications.” https://datatracker.ietf.org/doc/html/rfc5277. Accessed: 2022-03-11.

[34] “RFC 6022: YANG Module for NETCONF Monitoring.” https://datatracker.ietf.org/doc/html/rfc6022. Accessed: 2022-03-11.

[35] “RFC 5424: The Syslog Protocol.” https://datatracker.ietf.org/doc/html/rfc5424. Accessed: 2022-03-11.

[36] “SYSLOG YANG model draft-ietf-netmod-syslog-model-05.” https://datatracker.ietf.org/doc/html/draft-ietf-netmod-syslog-model-05. Accessed: 2022-03-11.

[37] “RFC 7159: The JavaScript Object Notation (JSON) Data Interchange Format.” https://datatracker.ietf.org/doc/html/rfc7159. Accessed: 2022-03-11.

[38] “gNMI:Structured data types.” https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md#23-structured-data-types. Accessed: 2022-03-31.

[39] “RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2.” https://datatracker.ietf.org/doc/html/rfc5246. Accessed: 2022-03-11.

[40] “Telcordia SR-4731 Issue 2 Optical Time Domain Reflector (OTDR) Data Format.”

[41] “OpenROADM MSA Device White Paper.” https://github.com/OpenROADM/OpenROADM_MSA_Public/wiki. Accessed: 2024-01-30.

[42] “ITU-T G.709.2/Y.1331.2 Cor. 1 (09/2020) OTU4 long-reach interface.” https://www.itu.int/rec/T-REC-G.709.2-202009-I!Cor1. Accessed: 2024-01-30.

[43] “ITU-T G.709.4/Y.1331.4 Cor. 2 (02/2022) OTU25 and OTU50 short-reach interfaces.” https://www.itu.int/rec/T-REC-G.709.4-202202-I!Cor2. Accessed: 2024-01-30.

[44] “ITU-T G.798.1 (01/2013) Types and characteristics of optical transport network equipment.” https://www.itu.int/rec/T-REC-G.798.1-201301-I. Accessed: 2024-01-29.

[45] “ITU-T G.805 (03/2000) Generic functional architecture of transport networks.” https://www.itu.int/rec/T-REC-G.805-200003-I. Accessed: 2024-01-29.

[46] “ITU-T G.806 Amd. 1 (11/2022) Characteristics of transport equipment - Description methodology and generic functionality.” https://www.itu.int/rec/T-REC-G.806-202211-I!Amd1. Accessed: 2024-01-29.

[47] “ITU-T G.808 Amd. 1 (03/2018) Terms and definitions for network protection and restoration.” https://www.itu.int/rec/T-REC-G.808-201803-I!Amd1. Accessed: 2024-01-29.

[48] “ITU-T G.808.1 (05/2014) Generic protection switching - Linear trail and subnetwork protection.” https://www.itu.int/rec/T-REC-G.808.1-201405-I. Accessed: 2024-01-29.

[49] “ITU-T G.870/Y.1352 (11/2016) Terms and definitions for optical transport networks.” https://www.itu.int/rec/T-REC-G.870-201611-I. Accessed: 2024-01-29.

[50] “ITU-T G.7703 Amd. 1 (11/2022) Architecture for the automatically switched optical network.” https://www.itu.int/rec/T-REC-G.7703-202211-I!Amd1. Accessed: 2024-01-29.

[51] “RFC 4389: Neighbor Discovery Proxies (ND Proxy).” https://datatracker.ietf.org/doc/html/rfc4389. Accessed: 2022-06-08.

[52] “RFC 6724: Default Address Selection for Internet Protocol Version 6 (IPv6).” https://datatracker.ietf.org/doc/html/rfc6724. Accessed: 2022-06-08.

[53] “RFC 8345: A YANG Data Model for Network Topologies.” https://datatracker.ietf.org/doc/html/rfc8345. Accessed: 2022-03-11.

[54] “RFC 8363: GMPLS OSPF-TE Extensions in Support of Flexi-Grid Dense Wavelength Division Multiplexing (DWDM) Networks.” https://datatracker.ietf.org/doc/html/rfc8363. Accessed: 2022-03-06.

[55] “RFC 8415: Dynamic Host Configuration Protocol for IPv6 (DHCPv6).” https://datatracker.ietf.org/doc/html/rfc8415. Accessed: 2022-06-08.