

Open ROADM MSA Device White Paper

for release 7



7 v1.2 11/05/2020

Please download the latest version on <http://OpenROADM.org>

1 TABLE OF CONTENTS

1	TABLE OF CONTENTS.....	2
2	HIGH LEVEL DEVICE DESCRIPTION	6
2.1	PHYSICAL DESIGN	6
2.2	FUNCTIONAL DESIGN	6
2.2.1	Open ROADM Controller	6
2.2.2	Reconfigurable Add-Drop Multiplexer (ROADM)	6
2.2.3	In-Line Amplifier (ILA)	8
2.2.4	Xponder	9
2.2.4.1	Transponder.....	10
2.2.4.2	Muxponder/Switchponder.....	11
2.2.4.3	Regenerator	12
2.2.5	Pluggable Optics	13
2.2.5.1	External Pluggable Optics	13
2.3	OPEN INTERFACES	13
2.3.1	W (Single-Wavelength Interface).....	13
2.3.2	Wr (Single-Wavelength Interface for ROADM Add/Drop Ports).....	13
2.3.3	MW (Multi-Wavelength Interface)	13
2.3.4	MWi (Multi-Wavelength Interface for ILAs)	14
2.3.5	OSC (Optical Supervisory Channel).....	14
3	HIGH LEVEL MODEL DESCRIPTION	14
3.1	YANG DATA MODEL.....	14
3.2	DEVICE OBJECT MODEL.....	15
3.2.1	System Model	17
3.2.2	Equipment Model	18
3.2.3	Links Model.....	18
3.2.4	Interfaces Model.....	19
3.2.5	Connectivity Matrices	20
3.2.5.1	Connection Map.....	20
3.2.5.2	Switching Pools	21
3.2.6	ROADM Model.....	24
3.2.6.1	ROADM WDM Abstractions	24
3.2.6.2	Flexible Grid Capabilities.....	25
3.2.6.3	Media Channel (MC) Interfaces (<i>mc-ttp</i>)	26
3.2.6.4	Network Media Channel (NMC) Interfaces (<i>nmc-ctp</i>).....	27
3.2.6.5	ROADM Cross-Connections.....	29
3.2.7	ILA Model.....	30
3.2.7.1	ILA WDM Abstractions	30
3.2.8	Xponder Model	30
3.2.8.1	Xponder Abstractions	30
3.2.8.2	Slot and Port Capabilities Announcement	31
3.2.8.3	MC Capabilities	38
3.2.8.4	ODU Interface	38
3.2.8.5	Transponder Model	41
3.2.8.6	Muxponder/Switchponder Model	42
3.2.8.7	Regenerator Model.....	47
3.2.8.8	Bookended Xponders Model	51
3.2.9	Protection Groups and Protection Switching Model	52
3.2.9.1	Uni-Directional Protection Switching	53
3.2.9.2	Bi-Directional Protection Switching	53
3.2.9.3	Protection Switch Commands.....	53
3.2.9.4	Linear OTN Protection Modes.....	54
3.2.9.5	1+1 Line Protection Switching.....	55

3.2.9.6	1+1 Path Protection Switching	56
3.2.9.7	1+1 Path Protection Group Creation.....	57
3.2.9.8	1+1 Path Protection Group Deletion.....	58
3.2.9.9	1+1 Path Protection Group Changes.....	58
3.2.9.10	SNCP Back-to-Back Ring Topologies.....	60
3.2.10	Protocols Model.....	65
3.2.11	Users Model.....	65
3.2.12	Tandem Connection Monitoring (TCM) Model	66
3.2.13	Trail Trace Identifiers (TTI) Model	67
3.2.14	Beyond 100G (B100G) Model	68
3.2.14.1	Sub-Rate OTUCn (OTUCn-M)	69
3.2.14.2	Optical Tributary Signal (OTSi)	70
3.2.14.3	Logical Port	71
3.2.15	FlexO Model for B100G.....	72
3.2.15.1	FlexO Long-Reach Line Interfaces (FlexO-x-DO).....	73
3.2.15.2	FlexO Short-Reach Client Interfaces (FlexO-x-RS)	75
3.2.16	Flexible Rate ODU (ODUflex) Model	77
3.2.16.1	ODUflex for 100Gb/s and Below	77
3.2.16.2	ODUflex for B100G.....	78
4	DEVICE OPERATION	80
4.1	SYNCHRONOUS/ASYNCHRONOUS OPERATIONS.....	80
4.1.1	Synchronous Operations.....	80
4.1.2	Asynchronous Operations	80
4.2	FILE OPERATIONS AND STRUCTURE	81
4.2.1	Local Device File Structure.....	81
4.2.2	Secure Shell File Transfer Protocol (SFTP) Specification.....	81
4.2.3	File Transfer (upload).....	81
4.2.4	File Transfer (download).....	82
4.2.5	Retrieve File List.....	83
4.2.6	Delete File	83
4.2.7	File Operation Notifications.....	84
4.3	DATA COMMUNICATION NETWORK (DCN).....	84
4.4	NETCONF PROTOCOL.....	85
4.4.1	Secure Shell (SSH) Support	86
4.4.2	Notification Support	86
4.4.2.1	Change Notification Support.....	86
4.4.2.2	Replay Support.....	88
4.4.3	Monitoring Support	89
4.5	SYSLOG PROTOCOL.....	89
4.6	TELEMETRY STREAMING SERVICE.....	90
4.6.1	Dial-Out Mode	91
4.6.2	Dial-In Mode	93
4.7	DEVICE FACEPLATE LABELS	95
4.8	DEVICE STATE DEFINITIONS AND TRANSITIONS	96
5	DEVICE DISCOVERY, COMMISSIONING, AND AUGMENTATION	98
5.1	DEVICE DISCOVERY AND COMMISSIONING	98
5.1.1	Planned Device Template (Step 1).....	99
5.1.2	Physical Device Installation, System Initialization, and Auto-Provisioning (Steps 2 & 3)	99
5.1.3	Device Initialization (Step 4)	100
5.1.4	Controller IP Address Assignment Discovery (Step 5)	100
5.1.5	Correlation of the Planned Template and the Temporary Discovered NE (Step 6).....	100
5.1.6	Device Commissioning (Step 7).....	101
5.1.6.1	Initial Validation	101

5.1.6.2	Node Identifier and IP Address Provisioning.....	102
5.1.6.3	Shelf Provisioning.....	102
5.1.6.4	Circuit Pack Provisioning	102
5.1.6.5	Degree, SRG, ILA, Xponder, Physical Links, Interfaces, and Protocols Provisioning	103
5.1.6.6	User Account Provisioning	103
5.1.6.7	Date and Time Setting.....	103
5.1.6.8	Permanent Node Discovery	103
5.2	DEVICE AUGMENTATION	103
5.3	OMS LINK PLANNING AND DISCOVERY	104
6	DEVICE SOFTWARE, FIRMWARE, AND DATABASE OPERATIONS	105
6.1	SOFTWARE AND FIRMWARE OPERATIONS	105
6.1.1	Software Upgrade.....	105
6.1.2	Software Downgrade.....	107
6.1.3	Firmware Upgrade	107
6.2	DATABASE OPERATIONS	108
6.2.1	Database Backup	108
6.2.2	Database Restore.....	108
6.2.3	Database Restore to Factory Defaults	111
7	PERFORMANCE MONITORING (PM), ALARMS, AND TCAS.....	111
7.1	PERFORMANCE MONITORING.....	111
7.1.1	Current PM List	111
7.1.2	Historical PM List	112
7.1.2.1	File-based Historical PM Retrieval	112
7.1.2.2	Clearing PMs	114
7.1.3	PM Behavior	114
7.1.3.1	Analog and Digital PM.....	114
7.1.3.2	Granularity	114
7.1.3.3	Validity	115
7.1.3.4	Other PM Behavior Notes	116
7.2	ALARMS	116
7.3	THRESHOLD CROSSING ALERTS (TCAs).....	116
8	PROVISIONING ACTION USE CASES	118
8.1	CONNECTION MANAGEMENT	118
8.1.1	Xponder Interfaces Creation (TX)	118
8.1.2	Add-Link Creation	121
8.1.3	Express Link Creation.....	123
8.1.4	Drop-Link Creation.....	126
8.1.5	Xponder Interfaces Creation (RX)	129
8.1.6	Xponder Interfaces Deletion (TX/RX).....	131
8.1.7	Add-Link Deletion	132
8.1.8	Express Link Deletion	133
8.1.9	Drop Link Deletion	135
8.2	INCREMENTAL HARDWARE	135
8.2.1	Degree Growth	135
8.2.1.1	Create Entities.....	136
8.2.1.2	Create Circuit-Pack Entities.....	137
8.2.1.3	Create Degree Entity.....	141
8.2.1.4	Create Physical Link Entities.....	142
8.2.2	SRG Growth	143
8.2.2.1	Create Shelf Entities.....	143
8.2.2.2	Create Circuit-Pack Entities.....	144
8.2.2.3	Create Shared Risk Group (SRG) Entity	146
8.2.2.4	Create Physical Link Entities.....	146

8.2.3	Xponder Growth	147
8.2.3.1	Create Shelf Entities	147
8.2.3.2	Create Circuit-Pack Entities	148
8.2.3.3	Create Xponder Entity	150
8.2.3.4	Create Physical Link Entities	151
9	MAINTENANCE ACTION USE CASES	151
9.1	DEVICE RESET/RESTART OPERATION	151
9.1.1	System Level Restart	152
9.1.2	Circuit Pack Level Restart	153
9.2	OPERATE/RELEASE LOOPBACK ON AN INTERFACE	154
9.2.1	Facility Loopbacks	154
9.2.2	Terminal Loopbacks	155
9.2.3	Loopback for B100G Interfaces	156
9.3	BIT-ERROR RATE (BER) TEST APPROACH	156
9.3.1	WDM Layer BER Test	156
9.3.2	OTN Layer BER Test	157
9.4	DISABLE AUTOMATIC LASER SHUTDOWN	157
9.5	OPTICAL TIME DOMAIN REFLECTOR (OTDR) SCAN	158
9.6	GENERATE/COLLECT TROUBLE SHOOTING FILE(S)	160
9.7	RETRIEVE ROADM CONNECTION PORT TRAIL	161
9.8	MISSING TOPICS (TO BE ADDRESSED IN A FUTURE VERSION OF THE WHITEPAPER)	161
10	REFERENCES	162
APPENDIX A:	MANIFEST FILE FOR SOFTWARE DOWNLOAD AND DATABASE OPERATIONS	163
A.1	SOFTWARE DOWNLOAD OPERATION	163
A.2	DATABASE BACKUP OPERATION	164
A.3	DATABASE RESTORE OPERATION	164
A.4	MANIFEST FILE YANG MODEL	165
A.4.1	Manifest File Tree View	165
A.4.1.1	Software Manifest Tree View	165
A.4.1.2	Database Backup Manifest Tree View:	166
A.4.1.3	Database Restore Manifest Tree View:	166
A.4.2	Manifest File Attributes and Commands	167
A.4.2.1	Common Manifest File and Command Attributes	167
A.4.2.2	Common Manifest Commands	168
A.4.2.3	Software Manifest File (<i>sw-manifest</i>)	169
A.4.2.4	Database Backup Manifest File (<i>db-backup-manifest</i>)	170
A.4.2.5	Database Restore Manifest File (<i>db-restore-manifest</i>)	171
A.5	MANIFEST FILE EXAMPLES	173
A.5.1	Software Manifest File Example	173
A.5.2	Database Backup Manifest File Example	176
A.5.3	Database Restore Manifest File Example	177
APPENDIX B:	PERFORMANCE MONITORING (PM) AND ALARM TABLES	179

2 HIGH LEVEL DEVICE DESCRIPTION

2.1 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.

2.2 FUNCTIONAL DESIGN

2.2.1 Open ROADM Controller

The Open ROADM MSA architecture assumes the existence of an Open ROADM controller¹ 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.

2.2.2 Reconfigurable Add-Drop Multiplexer (ROADM)

The Open ROADM MSA defines a ROADM device capable of providing colorless and directionless (or colorless, directionless and contentionless) 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 device² 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 1: 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

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.

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 (W_r)** interface defines the optical specifications for the add/drop ports of the ROADM devices

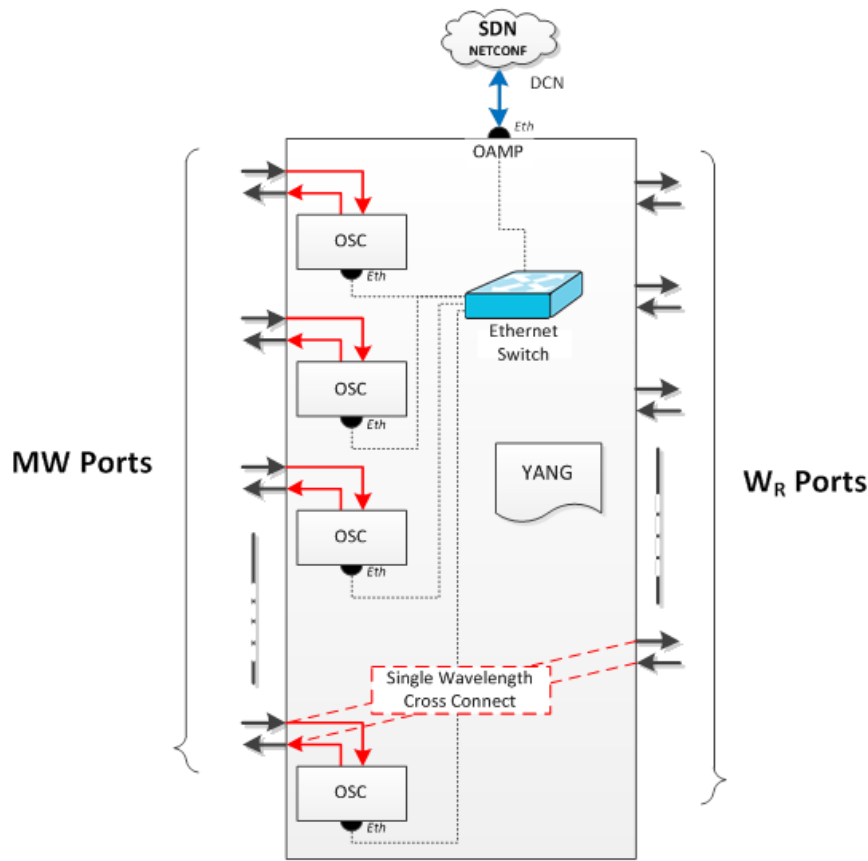


Figure 1: ROADM Interfaces

Refer to the Open ROADM MSA specification found at openroadm.org for the C/D ROADM specifications³:

- 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- W_r " tab
- Local control specs defined in the "Local Control" tab

³ The current version of the MSA specification as of the writing of this document is 20190611-Open-ROADM-MSA-specification-ver-3-01.xlsx

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

2.2.3 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 2xN 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 2: 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.

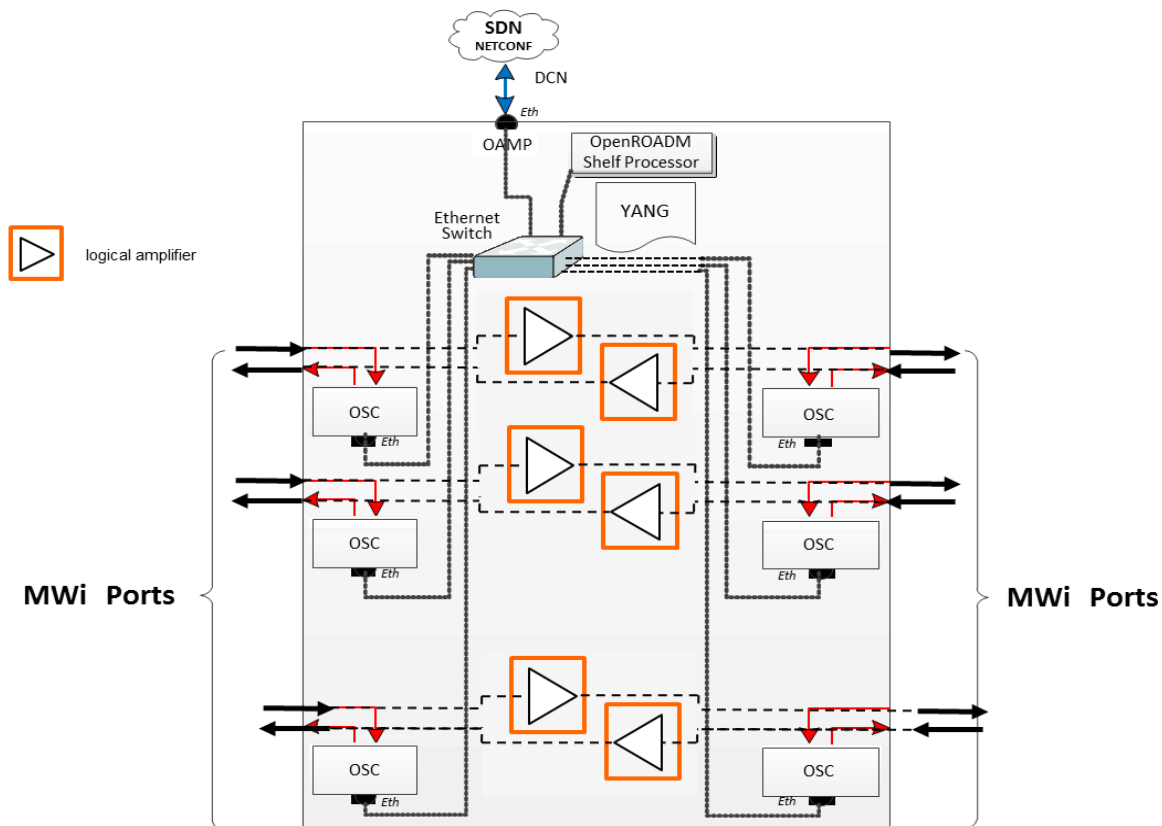


Figure 2: In-line Amplifier Interfaces

Please refer to the Open ROADM MSA specification found at openroadm.org 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
- 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.

2.2.4 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 2.3.1 (refer to *Figure 3: 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 openroadm.org 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 (to be included in the next version of the optical specification)
- 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

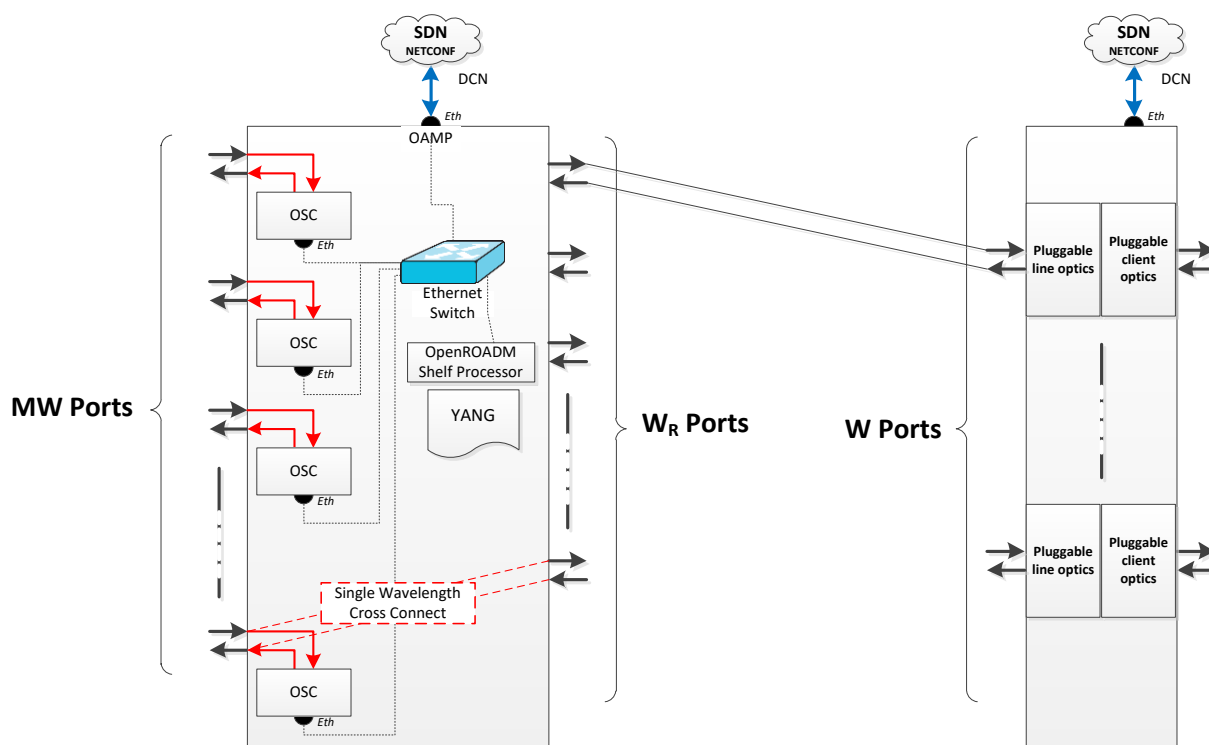


Figure 3: Xponder Interfaces

2.2.4.1 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

- **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
- Line-side
 - 100G Line-side pluggable type must be CFP2-ACO or CFP2-DCO with LC connectors
 - 200G-400G Line-side pluggable types must be CFP2-DCO with LC connectors (to be included in the next version of the optical specification)
- Client-side
 - 100G client interfaces/ports must be pluggable QSFP28 with LC connectors and support:
 - 100GBASE-R mapped into OPU4 using PCS codeword transparent Ethernet mapping, PT=07
 - OTU4
 - 400G client interfaces/ports must be pluggable QSFP56-DD with LC connectors
 - 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 openroadm.org for the Transponder functional specifications ("W TRPN functional" tab).

2.2.4.2 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 openroadm.org) defines:

- API using NETCONF interface with a YANG-based data model abstracts the management, control and provisioning of multi-vendor switch devices
- Line-side interfaces 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 type must be CFP2-ACO or CFP2-DCO with LC connectors
 - 200G-400G Line-side pluggable types must be CFP2-DCO with LC connectors (to be included in the next version of the optical specification)
- Client-side
 - 1GE client interfaces/ports must be a pluggable SFP with LC connectors and support:
 - 1000BASE-X mapped into ODU0 using TTT and GMP, ITU-T G.709 [A] Clause 17.7.1.1 1000BASE-X Transcoding, PT=07
 - 10G client interfaces/ports must be a pluggable SFP+ with LC connectors and support:
 - 10GBASE-R mapped into OPU2 using GFP-F, ITU-T G.7041[Q] Clause 7.1 Ethernet MAC Payload, PT=05
 - 10GBASE-R mapped into OPU2 using GFP-F, ITU-T G.7041[Q] 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 must be a pluggable QSFP28 with LC connectors and 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 openroadm.org 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).

2.2.4.3 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 3.2.8.7.1) 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 3.2.8.7.2). 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 3.2.8.7.3) 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 interfaces 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 type must be CFP-DCO, CFP2-ACO or CFP2-DCO with LC connectors
- 200G-400G Line-side pluggable types must be CFP2-ACO or CFP2-DCO with LC connectors (to be included in the next version of the optical specification)

2.2.5 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.

Client interfaces on the Transponder must use standard QSFP28/QSFP56-DD pluggable optics, while client interfaces on the Muxponder and Switchponder must use standard SFP/SFP+/QSFP28/QSFP56-DD pluggable optics.

Note: the Open ROADM MSA does not dictate nor preclude the use of pluggable optics to perform other functions within the ROADM or Xponder device.

2.2.5.1 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.

The architecture and requirements for external pluggable optics are still under study in the Open ROADM MSA.

2.3 OPEN INTERFACES

2.3.1 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 3: Xponder Interfaces* illustrates a functional representation.

2.3.2 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 1: ROADM Interfaces* illustrates a functional representation.

2.3.3 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 ROADMs (refer Section 2.3.5 for more details on the OSC). *Figure 1: ROADM Interfaces* illustrates a functional representation.

2.3.4 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 2: In-line Amplifier Interfaces* illustrates a functional representation.

2.3.5 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 1: ROADM Interfaces* and *Figure 2: In-line Amplifier Interfaces* illustrate a functional representation.

3 HIGH LEVEL MODEL DESCRIPTION

3.1 YANG DATA MODEL

The Open ROADM MSA YANG data models are based on YANG v1.1 per RFC 7950 [FF] 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.

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 but not fully validated.

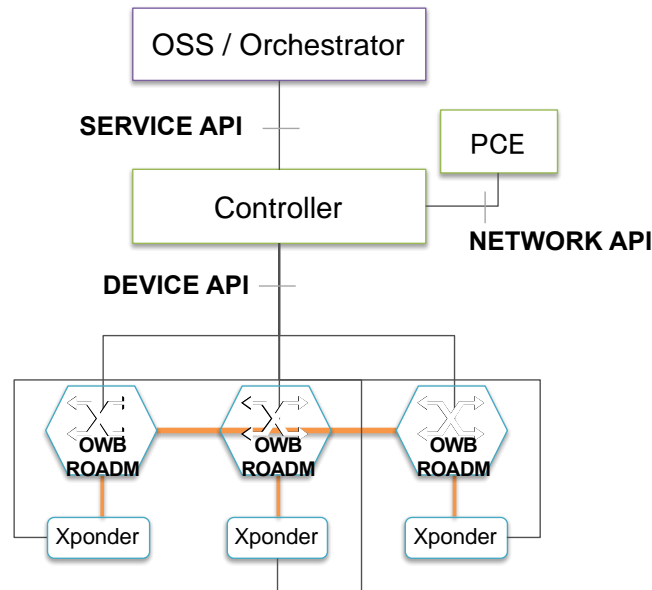


Figure 4: Open ROADM YANG Data Model Organization

3.2 DEVICE OBJECT MODEL

The Open ROADM MSA Device Object Model is shown in Figure 5: Open ROADM Device Object Model, while the specific objects of the Device Model to abstract the implementation of the ROADM, ILA, and Xponder devices are shown in Figure 6: Open ROADM Device Object Model Abstractions.

Note: MC and NMC replace OCH from v1.2.1 on a ROADM device

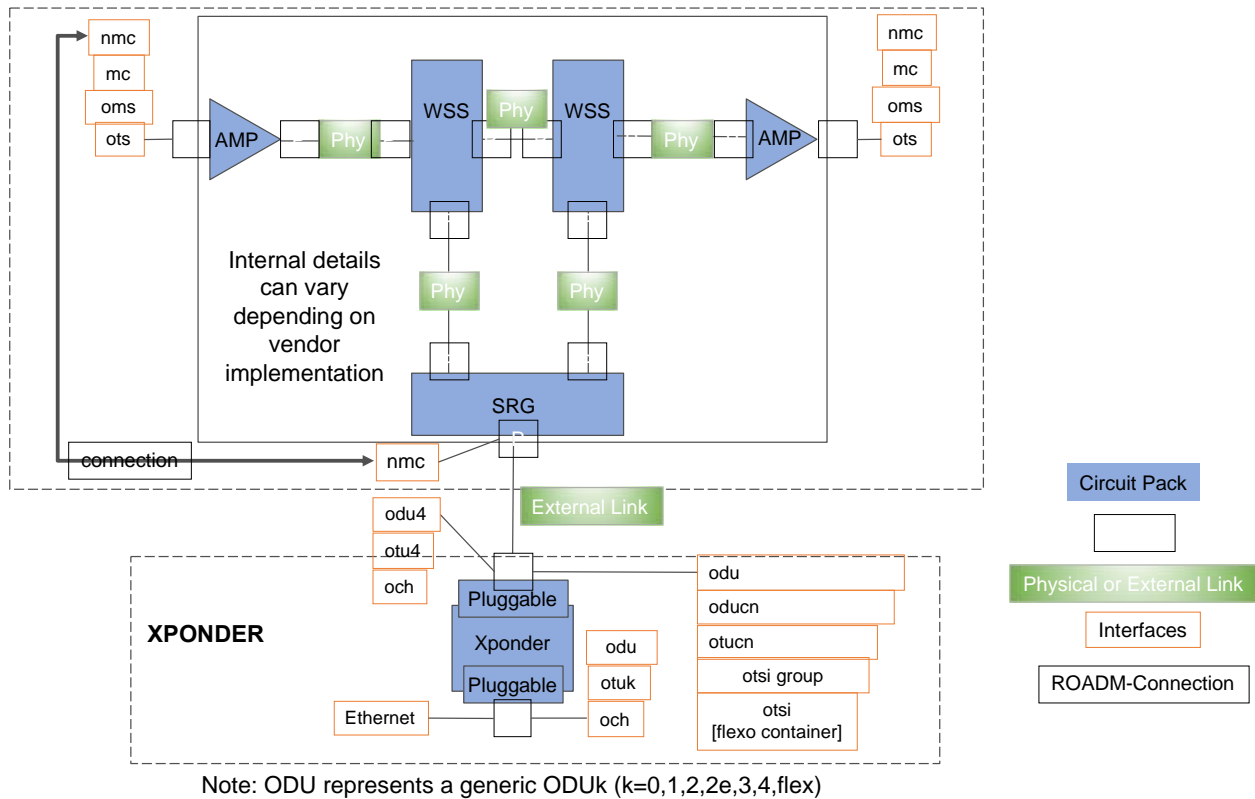


Figure 5: Open ROADM Device Object Model

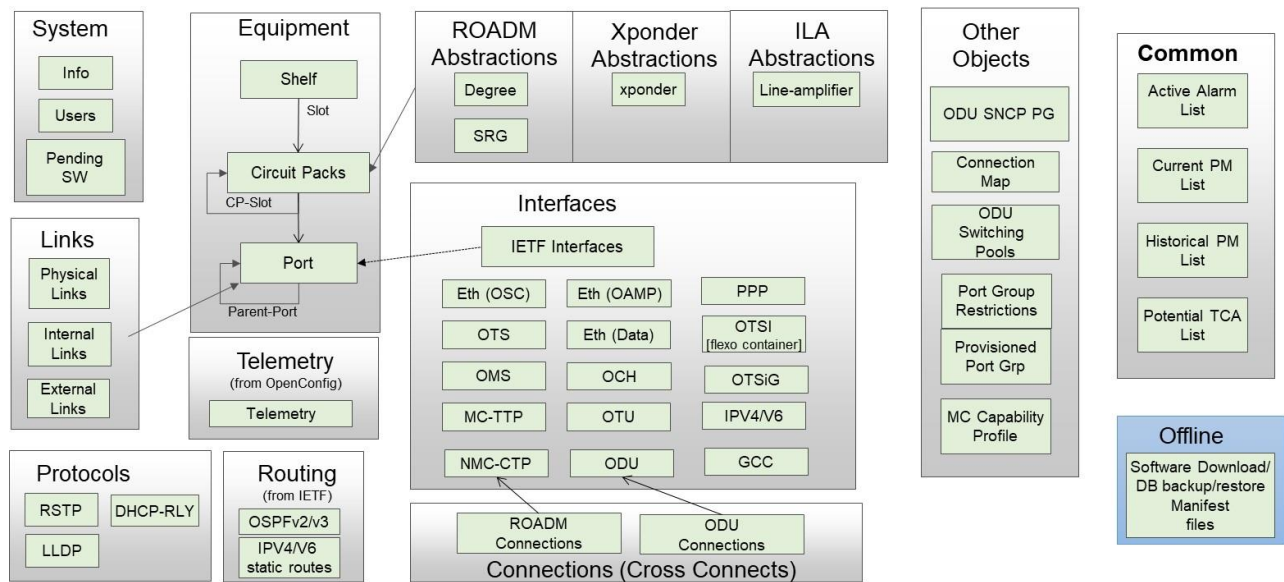


Figure 6: Open ROADM Device Object Model Abstractions

3.2.1 System Model

System level attributes are contained in the *info* container that includes a node identifier, office location (CLLI), vendor/model/serial-id, IP address info (IPv4, IPv6), software version/build, Open ROADMSA version, current time, location information. It also contains some capabilities related to the maximum number of degrees and SRGs supported, and the maximum number of historical PM bins for 15min and 24hr granularity.

```
+--rw info
| +--rw node-id?
| +--rw node-number?
| +--rw node-type
| +--rw cli?
| +--ro vendor
| +--ro model
| +--ro serial-id
| +--rw ipAddress?
| +--rw prefix-length?
| +--rw defaultGateway?
| +--ro source?
| +--ro current-ipAddress?
| +--ro current-prefix-length?
| +--ro current-defaultGateway?
| +--ro macAddress?
| +--ro softwareVersion?
| +--ro software-build?
| +--ro openroadm-version?
| +--rw template?
| +--ro current-datetime?
| +--rw lifecycle-state?
| +--rw geoLocation
| | +--rw latitude?
| | +--rw longitude?
| +--ro max-degrees?
| +--ro max-srgs?
| +--ro max-num-bin-15min-historical-pm?
| +--ro max-num-bin-24hour-historical-pm?
```

3.2.2 Equipment Model

The equipment model includes shelf (including associated slots) and circuit pack (including associated ports) information (refer to Figure 7: Equipment Model).

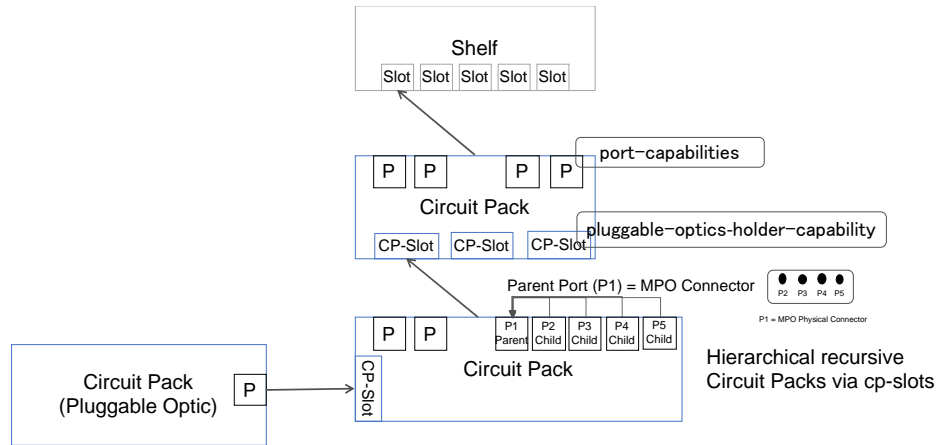


Figure 7: Equipment Model

The circuit pack's *circuit-pack-name* is unique within the node, this includes a circuit-pack contained in another circuit-pack (i.e. both circuit-pack names need to be unique). 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 recommendation is to concatenate the *circuit-pack-name* and *port-name* with the “##” character 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*.

3.2.2.1.1 MPO Connector Model

Referring to Figure 7: 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.

3.2.3 Links Model

Links connect circuit pack ports. Referring to Figure 8: Links Model, three types of links are modeled in a device:

- **Physical Link** is the fiber or electrical cable between two circuit packs
- **External Link** is the fiber between circuit packs/ports on one node to another node circuit pack/port

- **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.

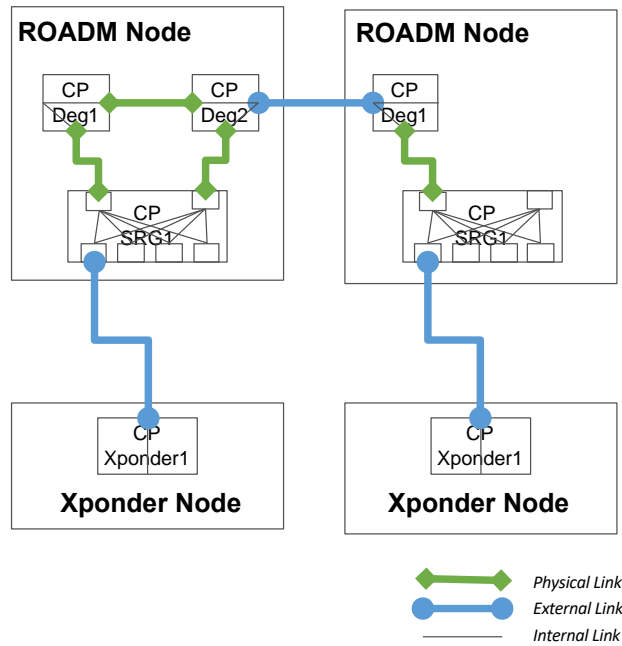


Figure 8: Links Model

3.2.4 Interfaces Model

Interfaces represent the facilities associated with port objects on the device (refer to Figure 9: ROADM Interfaces, Figure 10: ILA Interfaces, and Figure 11: Xponder Interfaces). Other interfaces include IPv4, IPv6, GCC, and PPP.

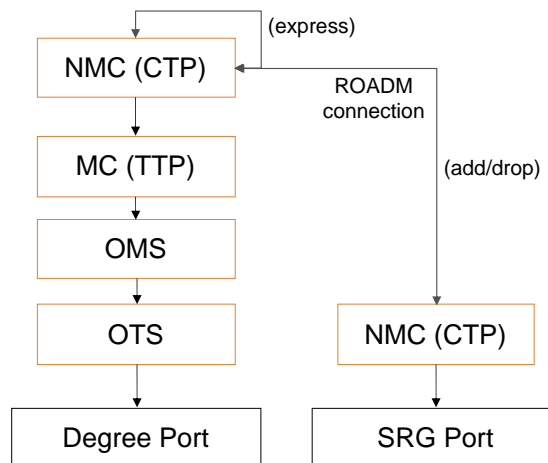


Figure 9: ROADM Interfaces

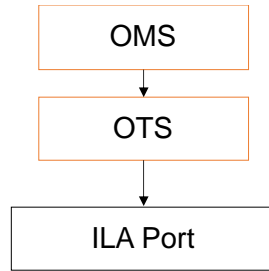


Figure 10: ILA Interfaces

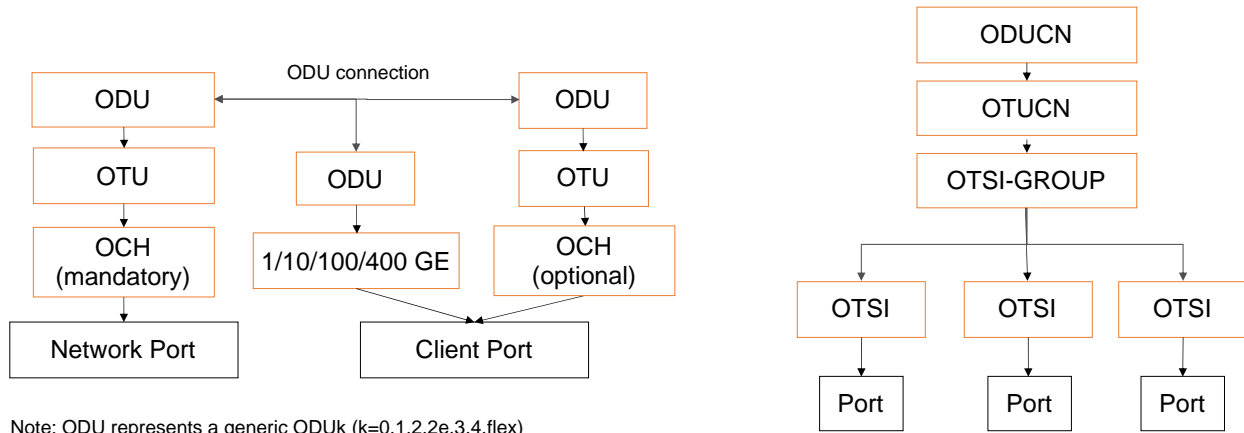


Figure 11: Xponder Interfaces

3.2.5 Connectivity Matrices

3.2.5.1 Connection Map

Connection Maps reflect the connectivity for ROADMs, transponders, and uni-directional regenerators devices. Connection Maps are used to indicate the port-to-port connectivity between external ports of a ROADM or xponder. Connection Maps represent uni-directional connectivity between the ports.

Within a ROADM device, it describes what Degrees and SRGs can be interconnected by a *roadm-connection*. For transponders and uni-directional 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
  
```

3.2.5.2 Switching Pools

Switching Pools describe the connectivity associated within the ODU layer by providing the ODU connectivity between external ports of muxponders, switchponders, and bi-directional regenerators. 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* [circuit-pack-name slot-name]
+--ro circuit-pack-name
+--ro slot-name
```

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 12: Non-Blocking Muxponder. 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 (P1-Pn) in it.

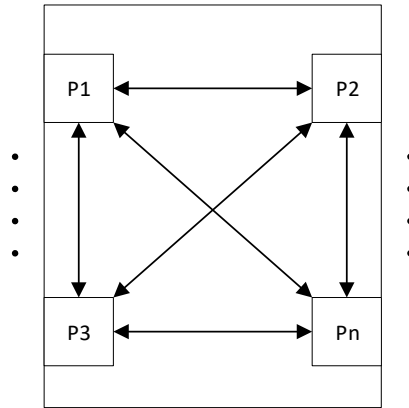


Figure 12: Non-Blocking Muxponder

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 13: Blocking Muxponder*) 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 *switching-pool-type* comprised of 10 *non-blocking-lists* with each such list having one unique client port and a common line port. For example (P1, P11), (P2, P11), (P3, P11), ..., (P10, P11).

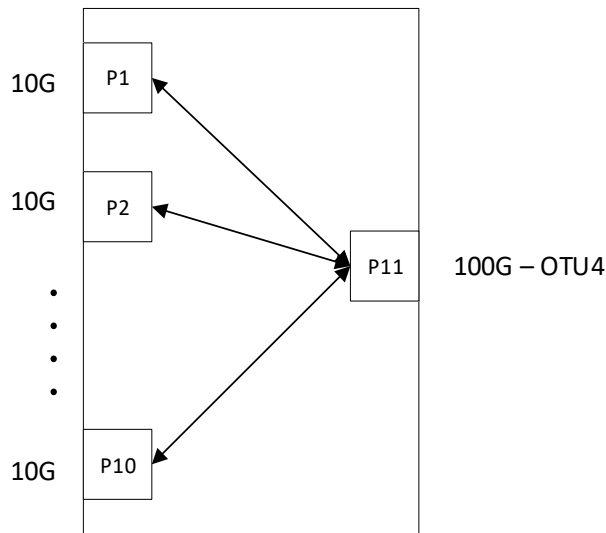


Figure 13: Blocking Muxponder

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 10x10G 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 14: Switching Pool with Interconnect BW*.

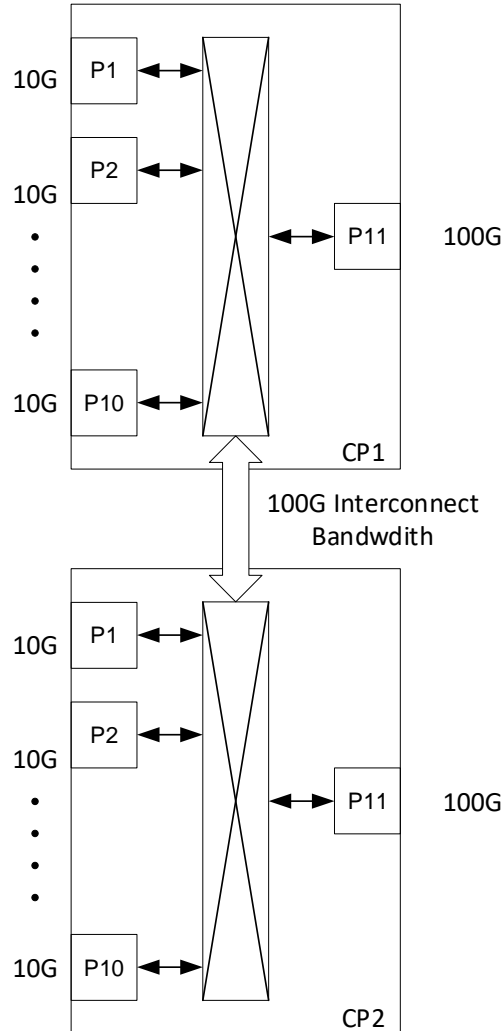


Figure 14: Switching Pool with Interconnect BW

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 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.

3.2.6 ROADM Model

3.2.6.1 ROADM WDM Abstractions

The ROADM model includes abstractions for the Degree and Shared Risk Group (SRG) of the ROADM.

3.2.6.1.1 Degree Abstraction

Degree defines the grouping of circuit packs that form a line degree. The Degree abstraction includes the ROADM degree number, lifecycle state (planned, deployed, maintenance etc.), circuit packs list identifying all circuit packs that make up the degree, line side ports, OTDR circuit pack port, and the flexgrid capabilities.

```
+--rw degree* [degree-number]
| +--rw degree-number
| +--rw lifecycle-state?
| +--ro max-wavelengths
| +--rw circuit-packs* [index]
| | +--rw index
| | +--rw circuit-pack-name
| +--rw connection-ports* [index]
| | +--rw index
| | +--rw circuit-pack-name
| | +--rw port-name
| +--rw otldr-port
| | +--rw circuit-pack-name?
| | +--rw port-name?
| +--ro mc-capabilities*
```

3.2.6.1.2 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 add/drop unit(s) for 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 srg-number
| +--rw lifecycle-state?
| +--ro wavelength-duplication
| +--rw circuit-packs* [index]
| | +--rw index
| | +--rw circuit-pack-name
| +--ro mc-capabilities*

```

3.2.6.2 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 follows:

```

+--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?

```

MC capability profiles can be advertised against any port (*ports**), degree (*degree**), and SRG (*shared-risk-group**). 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*

```

MC capabilities samples for fixed grid and flexible grid capable ROADM nodes are shown in *Table 1: Sample MC Capabilities for ROADM Devices*.

Table 1: 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

The figures in *Table 1: 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.

3.2.6.3 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. 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 follows:

```
| +--rw mc-ttp
| | +--rw min-freq?
| | +--rw max-freq?
| | +--ro center-freq?
| | +--ro slot-width?
```

MC-TTP are provisioned on the OMS interface. An OMS interface can have one or more *MC-TTPs* and the frequency ranges of MC on a degree port cannot overlap.

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).

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

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

where: $\text{min-slots} \leq m \leq \text{max-slots}$ 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 ≥ 191.325 THz and the maximum frequency ≤ 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 15: Media Channel (MC), slot width and center frequency being read only data are calculated according to these formulas:

$$\text{slot-width (GHz)} = \text{slot-width-granularity} * m$$

$$\text{center-freq (THz)} = 193.1 + \text{center-freq-granularity} / 1000 * n$$

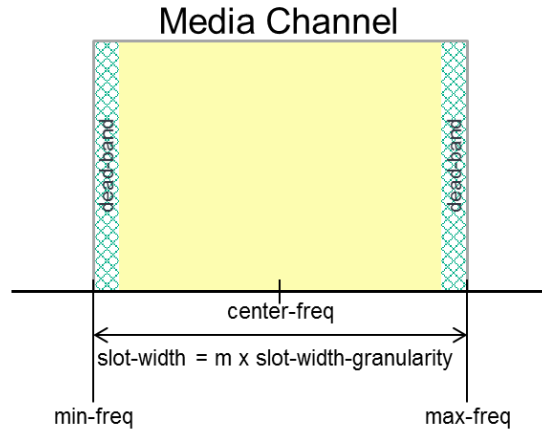


Figure 15: Media Channel (MC)

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:

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

3.2.6.4 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 nmc-ctp
| | +--rw frequency?
| | +--rw width?
```

Referring to *Figure 16: Network Media Channel (NMC)*:

*frequency (THz) = 193.1 + center-freq-granularity / 1000 * n*, where *n* is a positive or negative integer including 0

*width (GHz) = slot-width-granularity * m - 4*2*, where $\text{min-slots} \leq m \leq \text{max-slots}$ and 4 GHz are considered for the dead bands

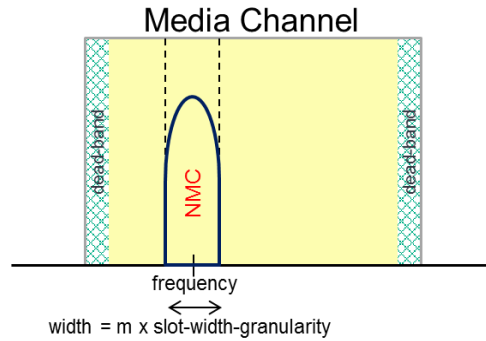


Figure 16: Network Media Channel (NMC)

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 17: MC-TTP and MC-CTP Relationship*).

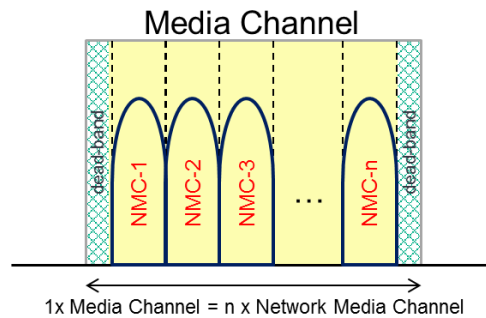


Figure 17: MC-TTP and MC-CTP Relationship

Figure 18: MC-TTP and NMC-CTP Relationship shows the relationship of the *MC-TTP* and *NMC-CTP* to each other and how they relate to the ports of a degree and SRG.

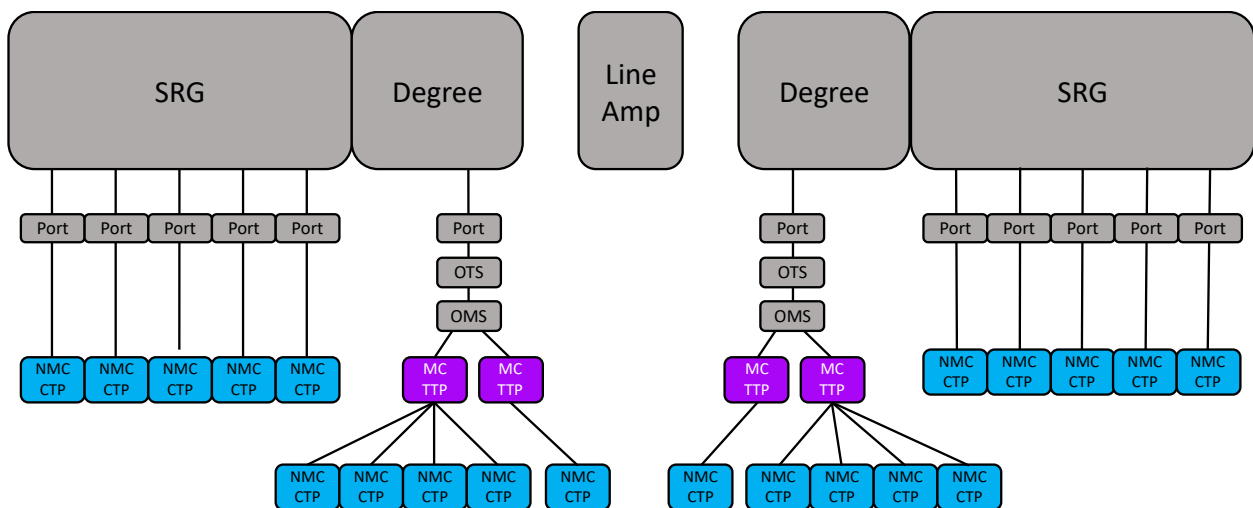


Figure 18: MC-TTP and NMC-CTP Relationship

NMC-CTP are only used by the ROADM device; 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) = $193.1 + 50 / 1000 * n$
- 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).

3.2.6.5 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 19: Flex-Grid *NMC-CTP* to *NMC-CTP* Cross-Connections). ROADM cross-connections connect NMC interfaces on MC/OMS and NMC on SRG add/drop ports. They support Express, Add and Drop traffic, and controls the optical control mode for the channel. When cross connecting *NMC-CTPs*, the frequency and width must match.

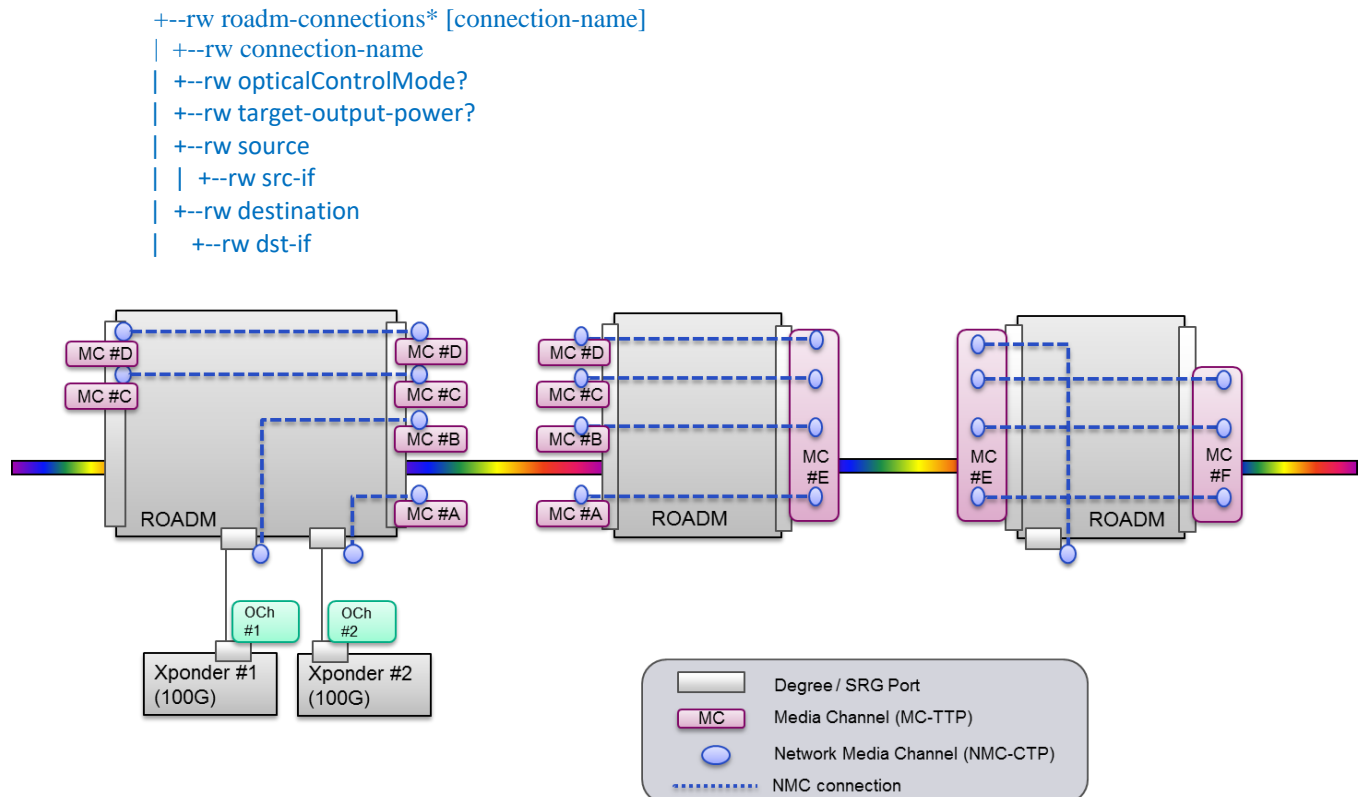


Figure 19: Flex-Grid *NMC-CTP* to *NMC-CTP* Cross-Connections

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.

3.2.7 ILA Model

3.2.7.1 ILA WDM Abstractions

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
| +--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 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
```

3.2.8 Xponder Model

3.2.8.1 Xponder Abstractions

The Xponder model 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.

```

+--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?

```

The Open ROAD 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 2: Xponder Model* for details).

Table 2: Xponder Model

xponder	xpdr-type	port-qualifier	connectivity announcement	explicit cross-connect?	section reference
Transponder	tpdr	xpdr-client/network	connectivity-map	no	3.2.8.5
Muxponder	mpdr	switch-client/network	switching pool	yes	3.2.8.6
OTN Switch/ Switchponder	switch	switch-client/network	switching pool	yes	3.2.8.6
Regenerator	regen	switch-client/network	switching pool	yes	3.2.8.7.1
	regen-uni	xpdr-client/network	connectivity-map	no	3.2.8.7.2

3.2.8.2 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.

```

+--ro cp-slots* [slot-name]
+--....
+--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
  +--ro supported-interface-capability* [if-cap-type]
    +--ro if-cap-type
    +--ro split-lambda-profile-name*
    +--ro otsigroup-capability-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?

+--rw ports* [port-name]
  | ....
  | +-- 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 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?

```

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*.

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 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?
```

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 10x10G muxponder may have the restriction that client port 1 must be mapped into trib port 1, trib slots 1-8,

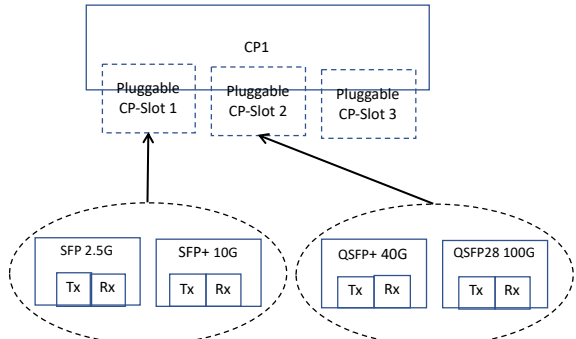
client port 2 must be into trib port 2, trib slots 9-16, etc... The *mpdr-client-restrictions* allow for a common provisioning model for muxponder applications. The muxponder capabilities is profile-based (*muxp-profile*) and identified by the *muxp-profile-name*. The is defined *muxp-profile* as follows:

```

+--ro muxp-profile* [profile-name]
|  +--ro profile-name
|  +--ro odu-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*

```

An example of a circuit pack with three pluggable slots is illustrated in *Figure 20: Port Capabilities*. 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.



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,}

Figure 20: Port Capabilities

A further example is illustrated in *Figure 21: Port Capabilities - Multi Phy*. 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.

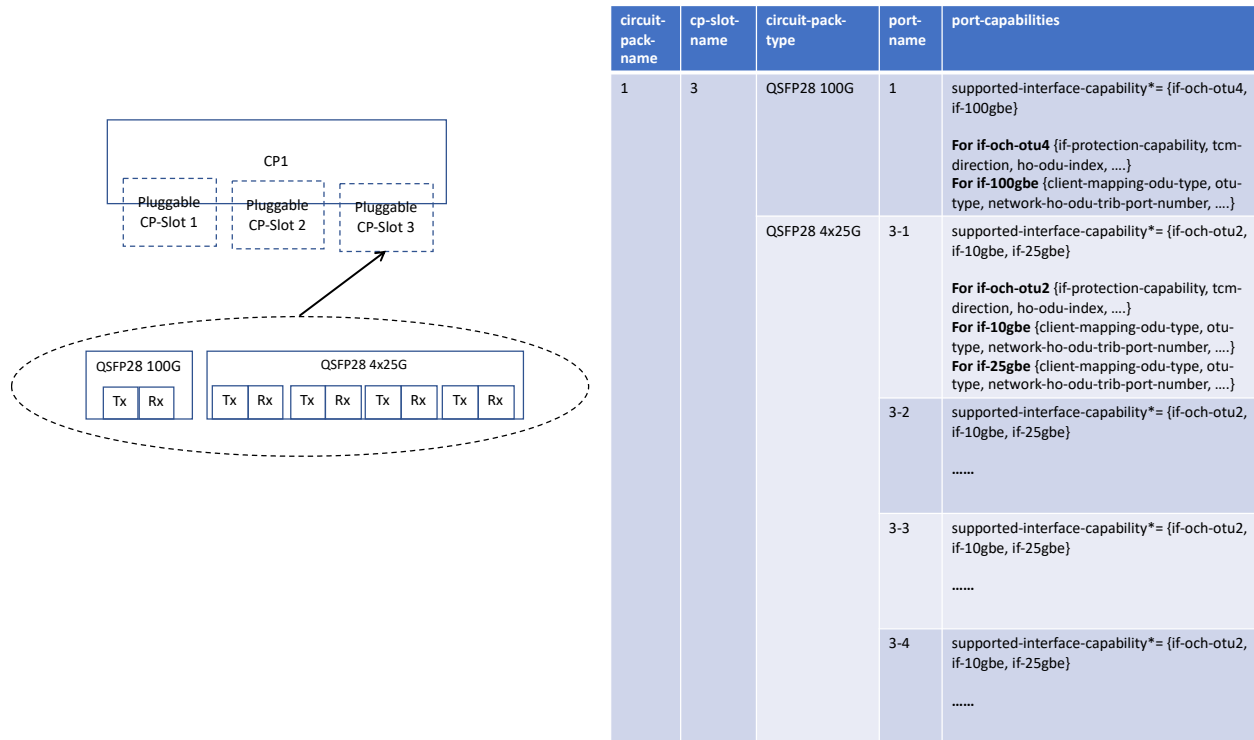


Figure 21: Port Capabilities - Multi Phy

3.2.8.2.1 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 x SFP based ports. It may be that these ports are software configurable to support a range of services. For example, 8 GbE or 4 x 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 22: Port Group Restriction Interfaces*. 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.

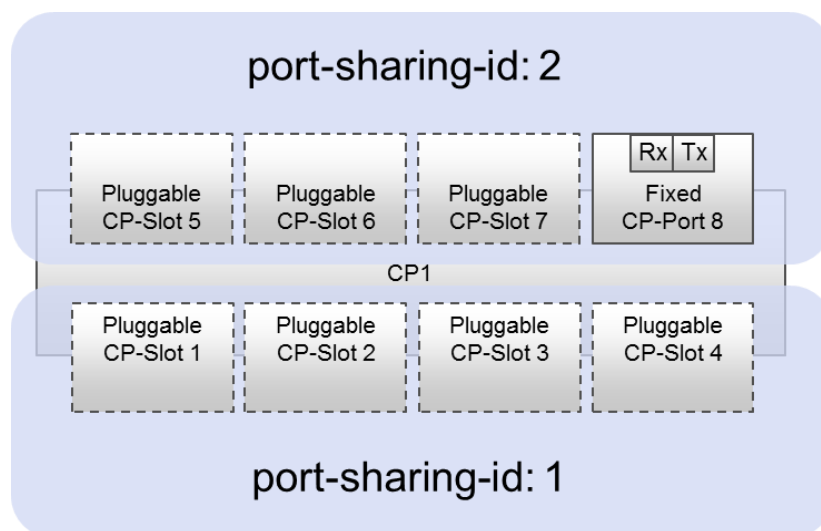


Figure 22: Port Group Restriction Interfaces

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 3: Port Group Restrictions Definition*.

In

*Table 3: Port Group Restrictions Definition***Error! Reference source not found.**, 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..

Table 3: Port Group Restrictions Definition

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						OC48
	CP1		4			CP1	3	3	OTU1
									OC48
					3	CP1	1	1	OTU1
									OC48
						CP1	3	3	GE
									OC3
									OC12
						CP1	4	4	GE

[illegible]

The active or configured port group is advertised via *port-group-restriction/possible-port-config*, referenced by a *port-sharing-id* and *config-id*.

```

+--ro port-group-restriction
|   +--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?

```

3.2.8.2.1.1 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*.

```

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

```

3.2.8.3 MC Capabilities

Like ROADMs (see Section 3.2.6.2), 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)

```

Note: *slot-width-granularity*, *min-slots*, and *max-slots* are not used for xponders. MC capabilities for transponders are only required for the transponder network port. This information is used by the controller to set the transponder frequency on the OCH or OTSI interface.

3.2.8.4 ODU Interface

3.2.8.4.1 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 23: ODU Termination Points.

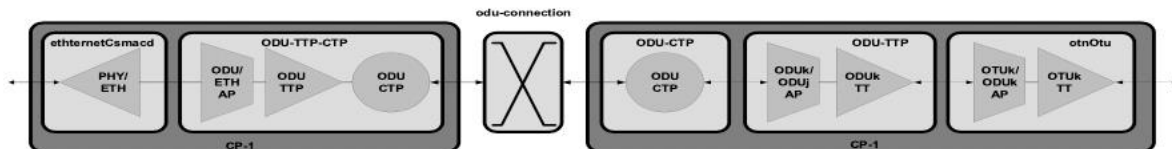


Figure 23: ODU Termination Points

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.

3.2.8.4.2 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 4: ODU Function*).

Table 4: ODU Function

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

			<i>ODU-TTP-CTP</i> 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	<i>ODU-CTP</i>
Regenerator (uni-directional)	network and client	no ODU cross-connects between client and network ODU interfaces	<i>ODU-CTP</i>

Transponder - 100GE/400GE Client:

- 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=*
 - a) *ODU-TTP* for the network-side HO-ODU interface
 - b) *ODU-CTP* for the network-side LO-ODU interface
 - c) *ODU-CTP* for client-side LO-ODU interfaces that are OTU2 interfaces (assumes cross-connect at the ODU2 level)
 - d) *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*

3.2.8.5 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 24: Example of a 400GE transponder. 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.

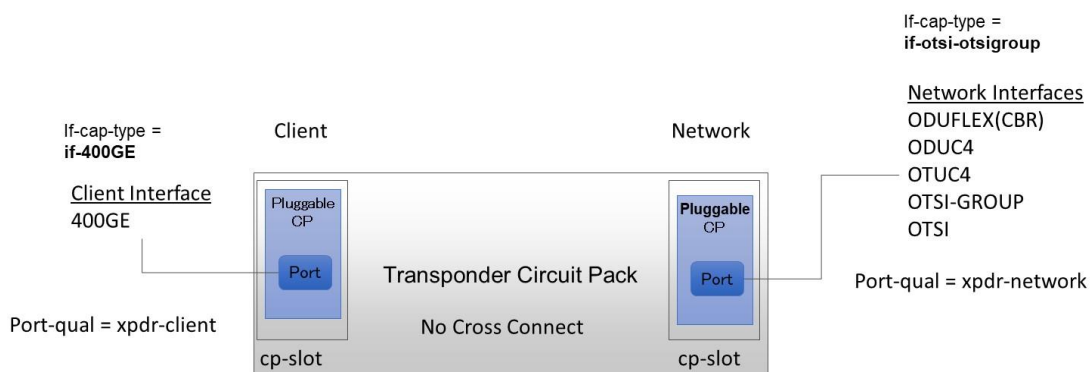


Figure 24: Example of a 400GE transponder

3.2.8.5.1 Connection Map

Like ROADMs and uni-directional regenerators, connection maps (see Section 3.2.5.1) 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.

3.2.8.5.2 Client/Network Mapping

As described in Section 3.2.14, B100G clients are mapped into a OPUCn 5G Tributary Slot (TS) #A.B per ITU-T G.709 clause 20.1 [A], where A=1..n and B=1..20 (5G TS). The mapping table below (refer to *Table 5: Transponder 400GE to ODUflex(400GE) to OTUC4*) shows the corresponding Tributary Port Number (TPN) and Tributary Slot (TS) for a 400GE mapped to a 400G network interface. **Note: 400G network interfaces will be added to the next version of the optical specification.**

Table 5: Transponder 400GE to ODUflex(400GE) to OTUC4 Mapping

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
---	-------------------------------------------------------------------------------------------------------

3.2.8.6 Muxponder/Switchponder Model

3.2.8.6.1 ODU Cross-Connections

ODU cross-connections (*odu-connection*) are applicable to muxponders, switchponders, and bi-directional regenerators (see Section 3.2.8.7.1). 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

```

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 3.2.8.6.2.2) 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.

3.2.8.6.2 Muxponder Specific Model

3.2.8.6.2.1 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 3.2.5.2) 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 x 1GE clients mapping into a 10G network (refer to *Table 6: Muxponder 8 x 1GE to 10G Mapping (OTU2)*)
- 10 x 10GE/OTU2 clients mapping into a 100G network (refer to *Table 7: Muxponder 10 x 10GE/OTU2 to 100G Mapping (OTU4)*). Mapping of the 10GE into an ODU2 and ODU2e are both supported.

Table 6: Muxponder 8 x 1GE to 10G Mapping (OTU2)

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 7: Muxponder 10 x 10GE/OTU2 to 100G Mapping (OTU4)

Client Port	Network ODU mapping
1	TPN 1, TS 1-8
2	TPN 2, 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
----	------------------

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

As described in Section 3.2.14, B100G (200G-400G) are mapped into a OPUCn 5G Tributary Slot (TS) #A.B per ITU-T G.709 clause 20.1 [A], 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). **Note: 200G-400G network interfaces will be added to the next version of the optical specification.**

- 2 x 100GE/OTU4 mapped into a 200G network (refer to *Table 8: Muxponder 2x100GE/OTU4 to 200G Mapping (OTUC2)*)
- 3 x 100GE/OTU4 mapped into a 300G network (refer to *Table 9: Muxponder 3x100GE/OTU4 to 300G Mapping (OTUC3)*)
- 4 x 100GE/OTU4 mapped into a 400G network (refer to *Table 10: Muxponder 4x100GE/OTU4 to 400G Mapping (OTUC4)*)

Table 8: Muxponder 2x100GE/OTU4 to 200G Mapping (OTUC2)

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 9: Muxponder 3x100GE/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

Table 10: Muxponder 4x100GE/OTU4 to 400G Mapping (OTUC4)

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 #3 TPN 4, TS 4.1 - 4.20

3.2.8.6.2.2 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.

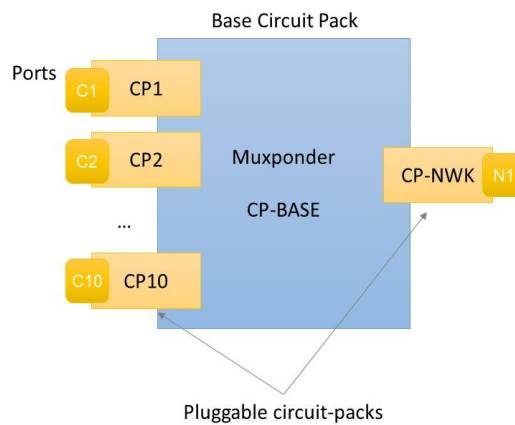


Figure 25: 10x10GE to 100G Muxponder

Figure 25: 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:

```
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}]
    }
}
```

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.

3.2.8.6.2.3 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 26: ODU Connection), and may additionally support the ability to provision this using two uni-directional cross connects.

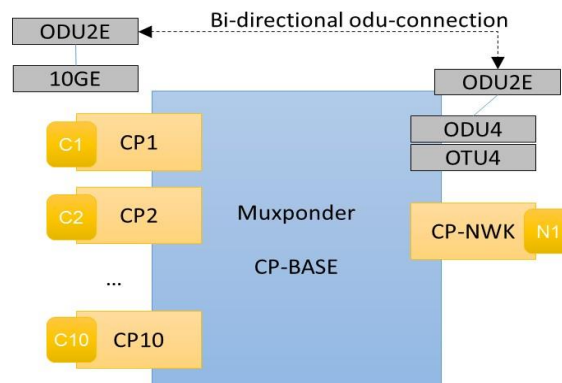


Figure 26: ODU Connection

3.2.8.6.2.4 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.

3.2.8.6.2.5 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.

3.2.8.7 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.

3.2.8.7.1 Bi-Directional Regenerator Model

The baseline bidirectional model example shown in *Figure 27: Bi-directional Regenerator Model Example* 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 3.2.5.2) is used to represent the connectivity between the network ports, as well as, the ODU cross-connect (switch-network) functionality. *Figure 27: 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.

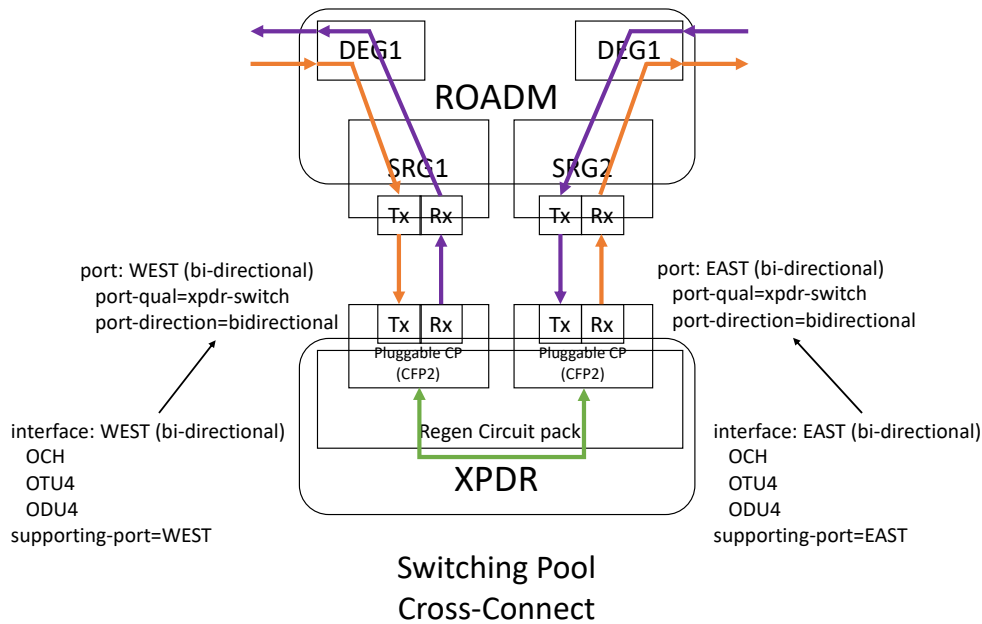


Figure 27: Bi-directional Regenerator Model Example

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 3.2.8.6). The network ports use the *port-qual* of *switch-network* and a switching pool (see Section 3.2.5.2) is used to represent the connectivity between the network ports; explicit ODU cross-connects (see Section 3.2.8.6.2.3) are used to connect the network ports.

An example for a bi-directional regenerator xponder entry would be:

```
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}
```

3.2.8.7.2 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 28: Uni-directional Regenerator Model Example 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.

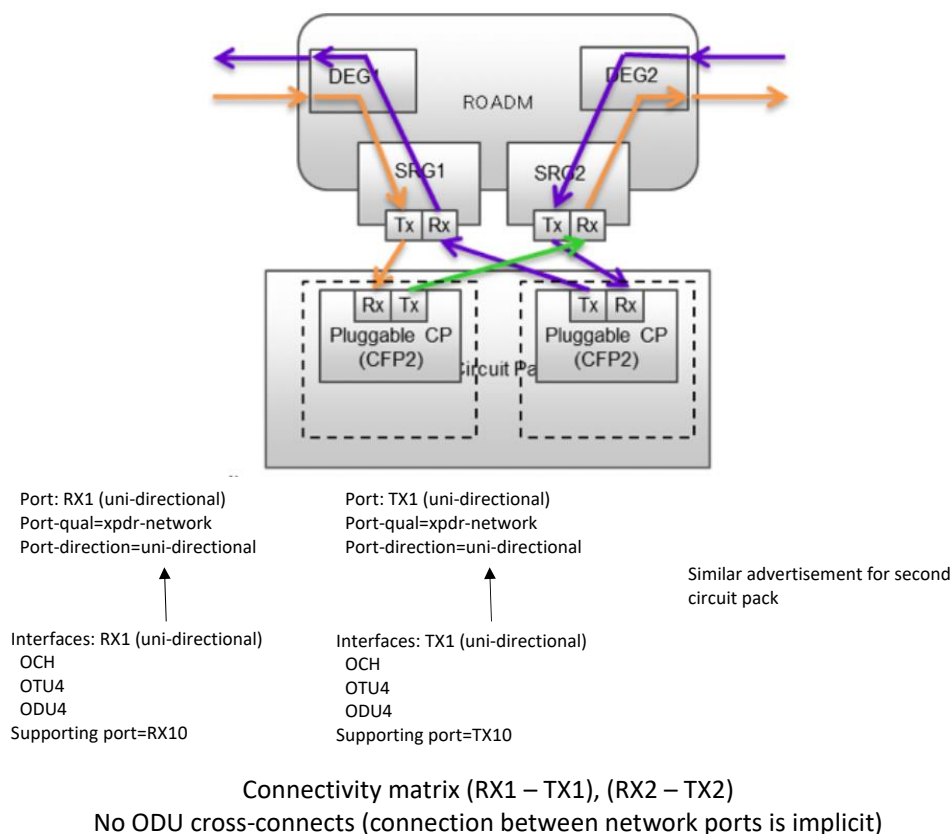


Figure 28: Uni-directional Regenerator Model Example

The Open ROADM MSA models uni-directional regenerators in a similar fashion as a transponder (see Section 3.2.8.5). Like a transponder, 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).

Note: if a bi-directional regenerator is implemented with two uni-directional regenerators, two independent xpnders are identified in the xpnder container for each uni-directional flow.

An example for a uni-directional regenerator xpnder entry would be:

```
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}

```

3.2.8.7.3 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 3.2.8.7.1). 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.

Error! Reference source not found. Figure 29: Bi-directional Regeneration Using Back-to-Back Transponders Example 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.

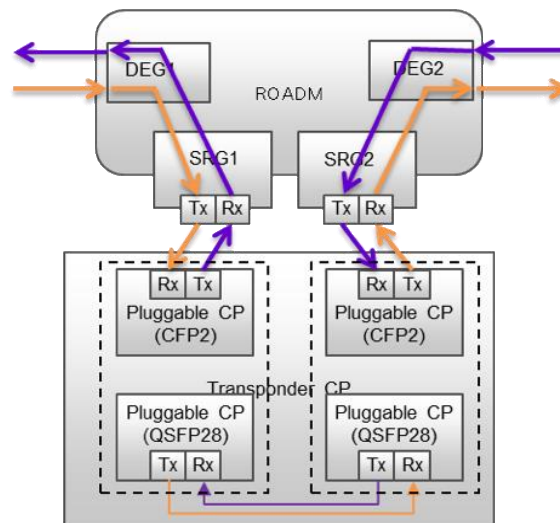


Figure 29: Bi-directional Regeneration Using Back-to-Back Transponders Example

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 xpdr container.

```

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

```

3.2.8.8 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 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 30: Bookended Xponder Configurations.

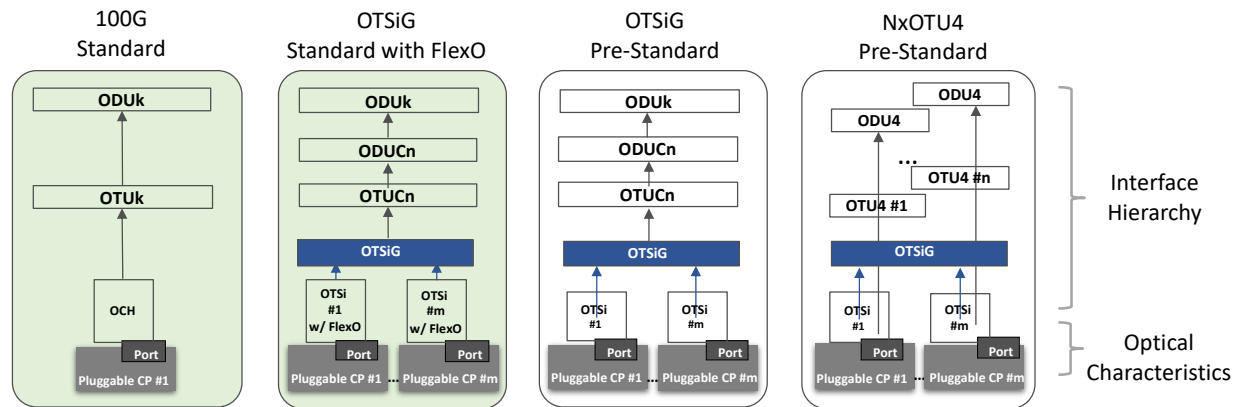


Figure 30: Bookended Xponder Configurations

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 30: Bookended Xponder Configurations, 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 NxOTU4 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 3.2.8.2).

3.2.9 Protection Groups and Protection Switching Model

ITU-T G.873.1 [P] 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 31: Protection Group***Error! Reference source not found.**). 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.

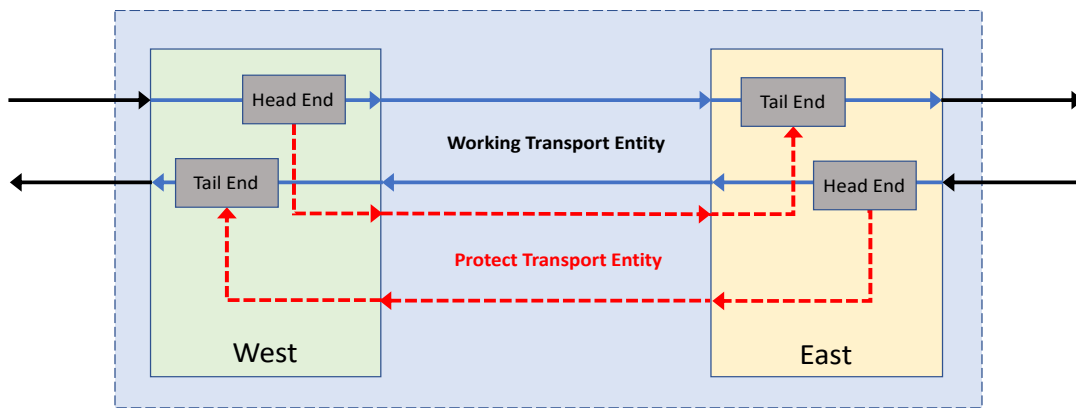


Figure 31: Protection Group

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 [P] 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 [M].

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

The Open ROADM MSA models protection groups and protection switching as follows:

```

+--rw protection-grps
|   +--rw odu-sncp-pg* [name]
|   |   +--rw name
|   |   +--rw level
|   |   +--rw prot-type?
|   |   +--rw switching-direction?

```

```

| +--rw revertive?
| +--rw mode
| +--rw protection-trigger-level?
| +--rw wait-to-restore?
| +--rw holdoff-timer
| | +--rw holdoff?
| | +--rw holdoff-multiplier?
| +--rw working-if
| +--rw pg-interfaces*
| +--ro active-if?

```

3.2.9.1 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*).

3.2.9.2 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.

3.2.9.3 Protection Switch Commands

The end-to-end switch commands (*odu-sncp-protection-switch/switch-command*) supported by the Open ROADMSA are shown in *Table 11: End-to-End Protection Switch Commands*.

Table 11: End-to-End Protection Switch Commands

Command (<i>switch-command</i>)	Description
<i>Lock-Out-Protect</i> (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.
<i>Force-Switch</i> (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.
<i>Manual-Switch</i> (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 <i>Lock-Out-Protect</i> request

	active, or if there is a <i>Force-Switch</i> request active, the <i>Manual-Switch</i> is prevented.
<i>Release</i> (CLEAR)	Releases the active near-end <i>Lock-Out-Protect</i> , <i>Force-Switch</i> , <i>Manual-Switch</i> commands or <i>wait-to-restore</i> (WTR) state

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 [P], 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.

3.2.9.4 Linear OTN Protection Modes

The Open ROADM MSA supports the linear OTN protection modes (*odu-sncp-pg/mode*) listed in *Table 12: Linear OTN Protection Modes* per ITU-T G.873.1 [P].

Table 12: Linear OTN Protection Modes

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

				<i>ODUT3, ODUT5, ODUT6)</i>		
<i>SNC/I</i>	SNC with Inherent monitoring	One HO ODUk or One OTUk	ODUKP or ODUKT	ODU SSF/SSD (<i>OTUk/ODUkA-SSF-SSD</i>)	Y	N
<i>CL-SNCG/I</i>	Compound Link SNC Group protection with Inherent monitoring	One HO ODUk	LO ODU	HO ODUkP SSF/SSD (<i>ODUKP/ODUjA-SSF-SSD</i>)	Y	N

The relationship between Line and Path level protection and the associated linear OTN protection modes and protection switch entities is shown in Figure 32: Linear OTN Protection Modes and Associated Protection Switched Entities

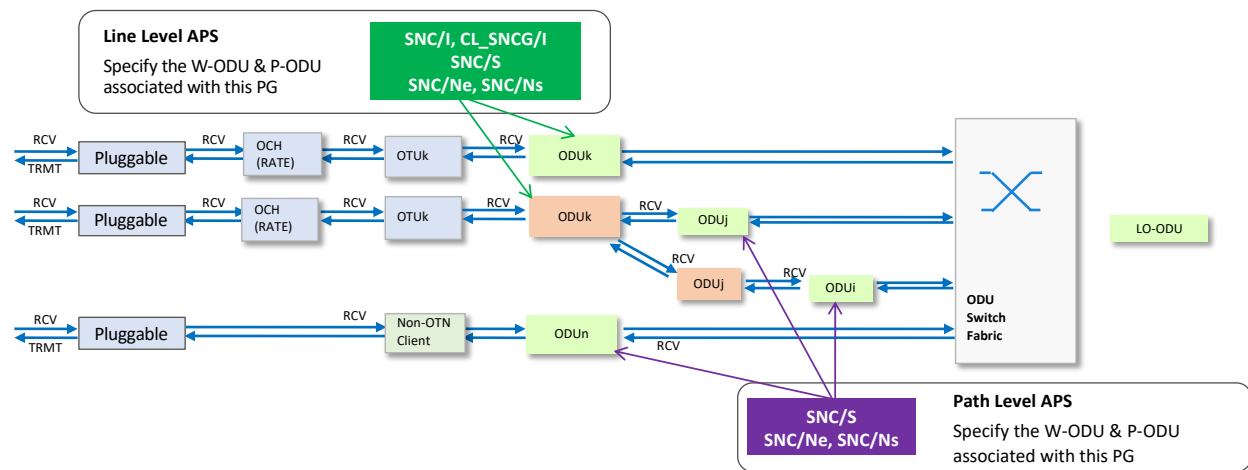


Figure 32: Linear OTN Protection Modes and Associated Protection Switched Entities

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 3.2.12 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.

3.2.9.5 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 33: 1+1 Line Protection).

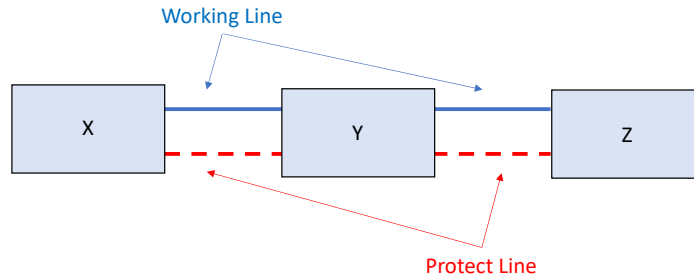


Figure 33: 1+1 Line Protection

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 34: Line Level Automatic Protection Switching (APS) Model*).

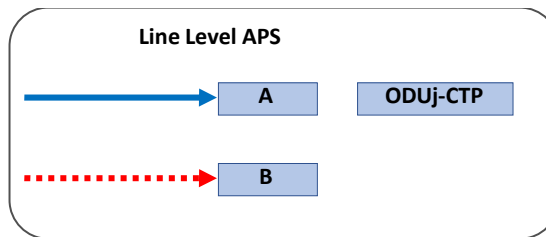


Figure 34: Line Level Automatic Protection Switching (APS) Model

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 the MSA model for the full list of mandatory and optional attributes).

<edit-config>

create Line PG (*odu-sncp-pg*):

level=line

working-if=A

pg-interfaces={A,B}

Commit

3.2.9.6 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 35: 1+1 Path Protection - SNCP Ring Configuration). Note: SNCP does not require that the two paths be completely diverse.

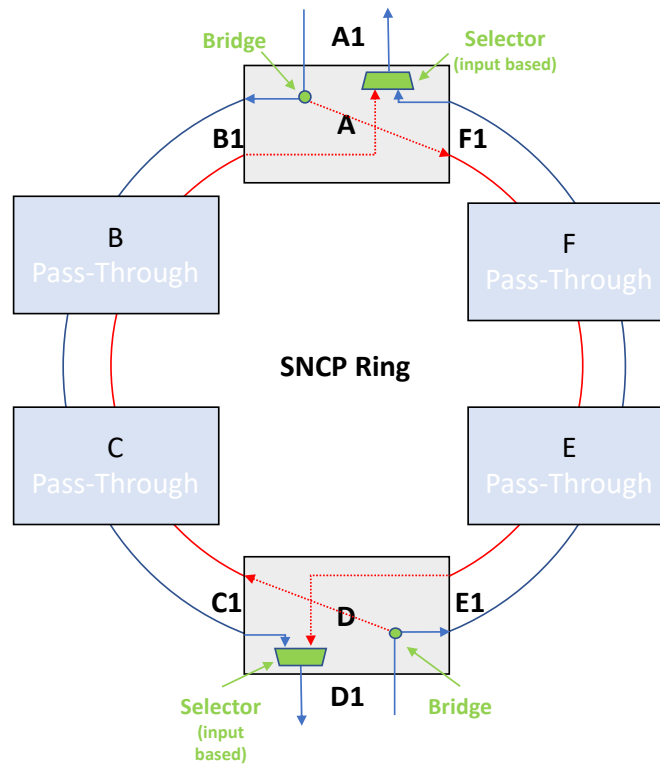


Figure 35: 1+1 Path Protection - SNCP Ring Configuration

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 36: Path Level Automatic Protection Switching (APS) Model). Before the protect path cross-connect 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 **Error! Reference source not found.**3.2.8.6.1).

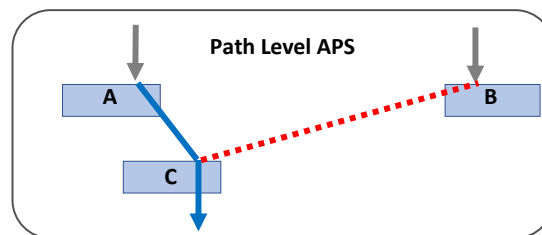


Figure 36: Path Level Automatic Protection Switching (APS) Model

3.2.9.7 1+1 Path Protection Group Creation

The Open ROADM MSA allows various ways to create a 1+1 path level protection group (refer to Figure 36: Path Level Automatic Protection Switching (APS) Model), via various multiple operations or a single

operation. 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>

create drop XCON A-C (*odu-connection*):

direction=bi-directional
src-if=A
dst-if=C

create Path PG (*odu-sncp-pg*):

level=path
working-if=A
pg-interfaces={A}

create drop XCON B-C (*odu-connection*):

direction=bi-directional
src-if=B
dst-if=C

Commit

3.2.9.8 1+1 Path Protection Group Deletion

As with the creation of a 1+1 path level protection group (refer to Section 3.2.9.7), deleting an existing 1+1 path level protection group can be achieved via multiple operations or a single operation. Of course, the most efficient procedure is via a single operation (see example below).

<edit-config>

delete drop XCON A-C (*odu-connection*):

direction=bi-directional
src-if=A
dst-if=C

delete drop XCON B-C (*odu-connection*):

direction=bi-directional
src-if=B
dst-if=C

delete Path PG (*odu-sncp-pg*):

level=path
working-if=A
pg-interfaces={A}

Commit

3.2.9.9 1+1 Path Protection Group Changes

The Open ROADMSA supports changes to an existing path-level APS protection group without the need to delete the protection group and creating a new protection group. An example scenario is depicted in *Figure 37: Existing Path Level APS Protection Group Change Migration*, where the drop cross-connection A-C is deleted, drop cross-connection B-C becomes the new working path, and a new

protection path D-C is attached to the protection group. Note: deletion of a protection group with traffic on the deleted path A-C will result in the traffic automatically switching to the remaining path B-C.

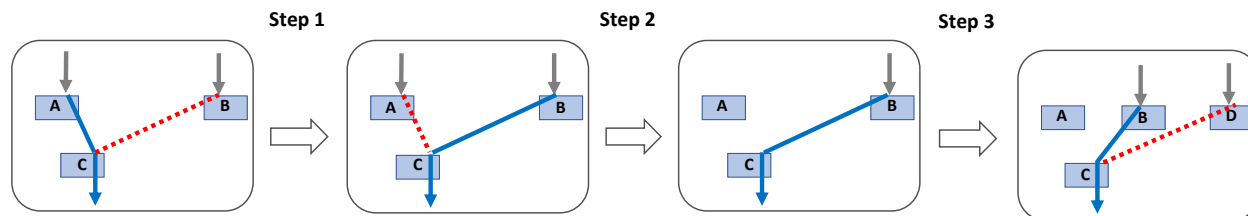


Figure 37: Existing Path Level APS Protection Group Change Migration

In order to minimize the traffic hit before deleting the protection path A-C, the first step should be to change the working path to B-C and then switch the traffic to path B-C. The next step is to delete the protection path A-C and detach it from the protection group in a single operation (remaining path B-C will continue to carry the traffic). The final step is to create a new drop cross-connection D-C and attach it to the protection group in a single operation.

Step 1: Change Path Level APS Path Working Path to Protect Path and Switch Traffic to New Working Path

<edit-config>

edit Path PG (*odu-sncp-pg*):

```
level=path
working-if=B
pg-interfaces={A,B}
```

Commit

<rpc odu-sncp-protection-switch >

switch traffic to path B-C:

```
pg-interface=A
switch-command=Manual-Switch
```

Commit

Step 2: Delete the Old Working Path

<edit-config>

delete drop 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

Step 3: Create a New Protect Path

<edit-config>

create drop XCON D-C (*odu-connection*):

direction=bi-directional

src-if=D

dst-if=C

add to Path PG (*odu-sncp-pg*):

level=path

working-if=B

pg-interfaces={B,D}

Commit

3.2.9.10 SNCP Back-to-Back Ring Topologies

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 38: SNCP Single Node Dual Ring Interconnect, where the traffic signal is broadcasted at the interconnecting node, resulting in duplicate traffic signals transmitted between the rings at the interconnecting node.

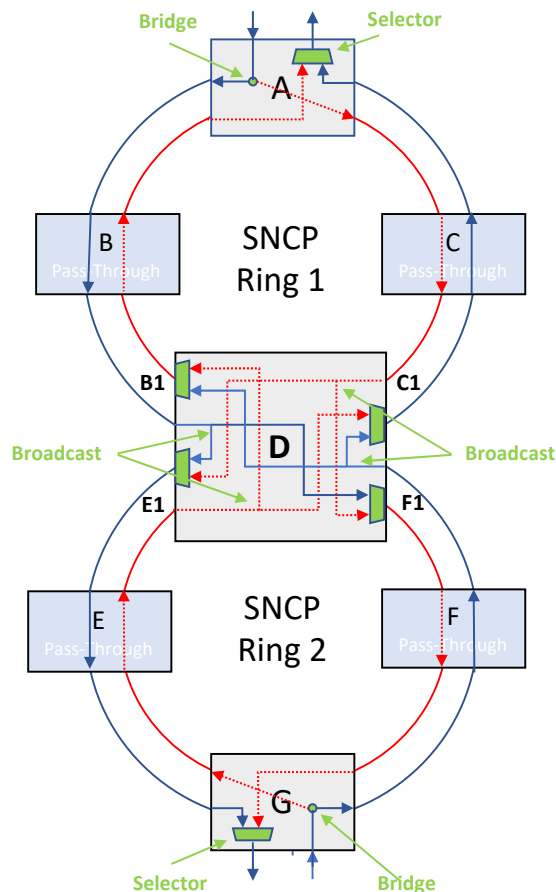


Figure 38: SNCP Single Node Dual Ring Interconnect

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

<edit-config>

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 PG1 XCON B1-F1 (*odu-connection*):

direction=bi-directional
src-if=B1
dst-if=F1

create PG1 XCON C1-F1 (*odu-connection*):

direction=bi-directional
src-if=C1
dst-if=F1

create PG2 XCON B1-E1 (*odu-connection*):

direction=bi-directional
src-if=B1
dst-if=E1

create PG2 XCON E1-C1 (*odu-connection*):

direction=uni-directional
src-if=E1
dst-if=C1

Commit

Using four nodes to interconnect SNCP rings is an example described in ITU-T G.842 [O] (refer to Figure 39: SNCP Four Node Dual Ring Interconnect). 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.

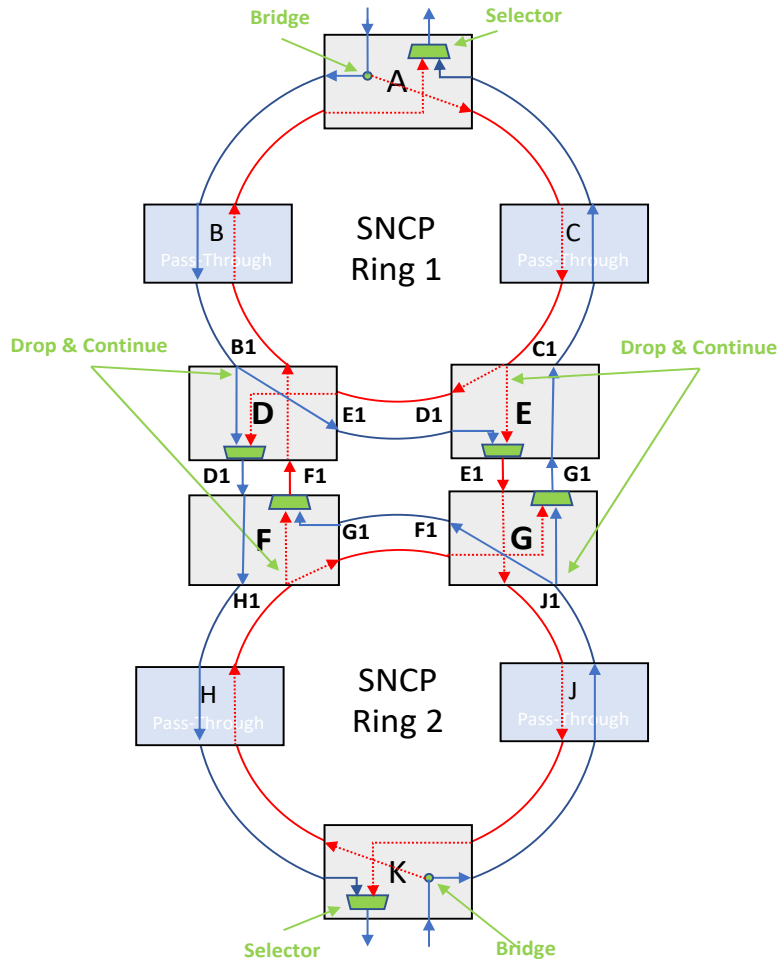


Figure 39: SNCP Four Node Dual Ring Interconnect (ITU-T G.842 [O])

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

Node D: Protection Group w/ Uni-Directional XCON

<edit-config>

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

Node E: Protection Group w/ Uni-Directional XCON

<edit-config>

create Path PG (*odu-sncp-pg*):

```
level=path
working-if=D1
pg-interfaces={D1,C1}
```

create drop XCON D1-E1 (*odu-connection*):

```
direction=uni-directional
src-if=D1
dst-if=E1
```

create drop XCON C1-E1 (*odu-connection*):

```
direction=uni-directional
src-if=C1
dst-if=E1
```

create continue XCON C1-D1 (*odu-connection*):

```
direction=uni-directional
src-if=C1
dst-if=D1
```

create pass-through XCON G1-C1 (*odu-connection*):

```
direction=uni-directional
src-if=G1
dst-if=C1
```

Commit

Node F: Protection Group w/ Uni-Directional XCON

<edit-config>

create Path PG (*odu-sncp-pg*):

```
level=path
working-if=G1
pg-interfaces={G1,H1}
```

create drop XCON G1-F1 (*odu-connection*):

```
direction=uni-directional
src-if=G1
dst-if=F1
```

create drop XCON H1-F1 (*odu-connection*):

```
direction=uni-directional
src-if=H1
dst-if=F1
```

create continue XCON H1-G1 (*odu-connection*):

```
direction=uni-directional
src-if=H1
dst-if=G1
```

create pass-through XCON D1-H1 (*odu-connection*):

```
direction=uni-directional
src-if=D1
dst-if=H1
```

Commit

Node G: Protection Group w/ Uni-Directional XCON

<edit-config>

create Path PG (*odu-sncp-pg*):

```
level=path
working-if=J1
pg-interfaces={F1,J1}
```

create drop XCON J1-G1 (*odu-connection*):

```
direction=uni-directional
src-if=J1
dst-if=G1
```

create drop XCON F1-G1 (*odu-connection*):

```
direction=uni-directional
src-if=F1
dst-if=G1
```

create continue XCON J1-F1 (*odu-connection*):

```
direction=uni-directional
src-if=J1
dst-if=F1
```

create pass-through XCON E1-J1 (*odu-connection*):


```
direction=uni-directional
src-if=E1
dst-if=J1
```

Commit

3.2.10 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 (note: GCC0 implementation requirements have not be discussed in the Open ROADM forum).

```
+--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?
```

3.2.11 Users Model

The default username/password is openroadm/openroadm.

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

rpcs:
+---x chg-password
| +---w input
| | +---w currentPassword
| | +---w newPassword
| | +---w newPasswordConfirm
+---ro output
| +--ro status
| +--ro status-message?
```

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

3.2.12 Tandem Connection Monitoring (TCM) Model

TCM overhead supports monitoring of arbitrary sub-network connections. Per ITU-T G.709 [A], 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 ODU 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 40: Nested and Cascaded ODUk Monitored Connections (ITU-T G.709 [A]), monitored connections A1-A2/B1-B2/C1-C2 and A1-A2/B3-B4 are considered nested, while B1-B2/B3-B4 are considered cascaded.

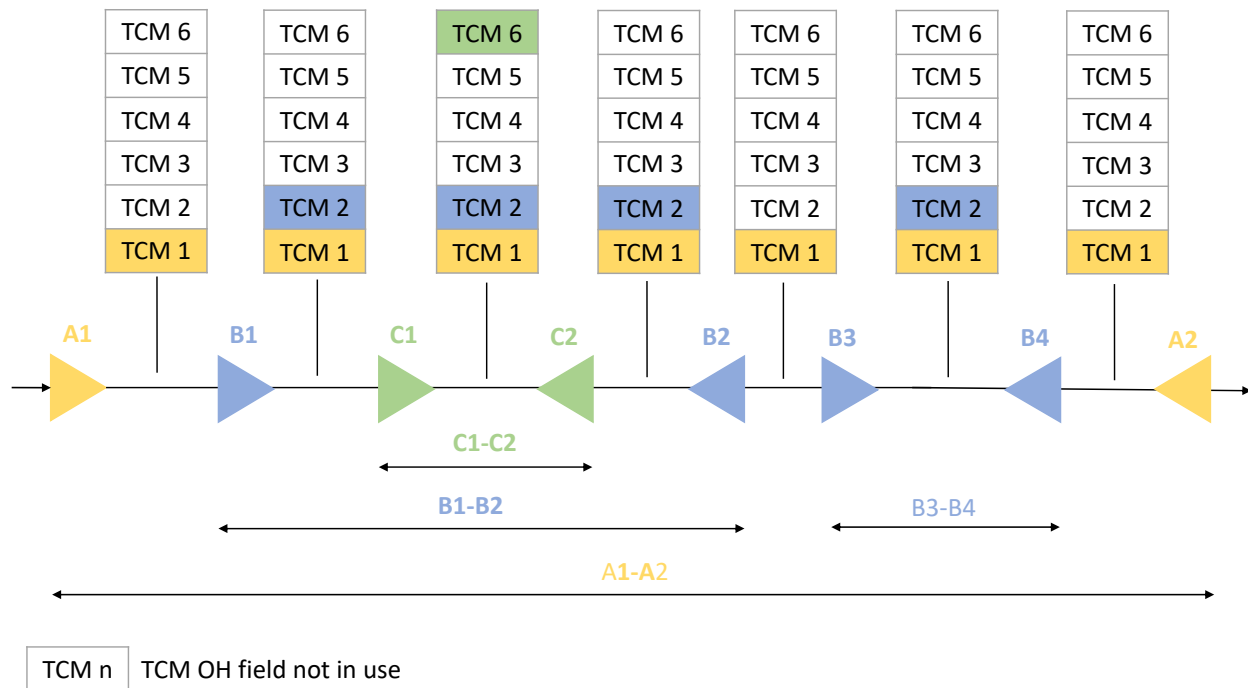


Figure 40: Nested and Cascaded ODUk Monitored Connections (ITU-T G.709 [A])

The Open ROADMSA 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 41: TCM Direction for an example. It should be noted that upgrading from Open ROADMSA 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).

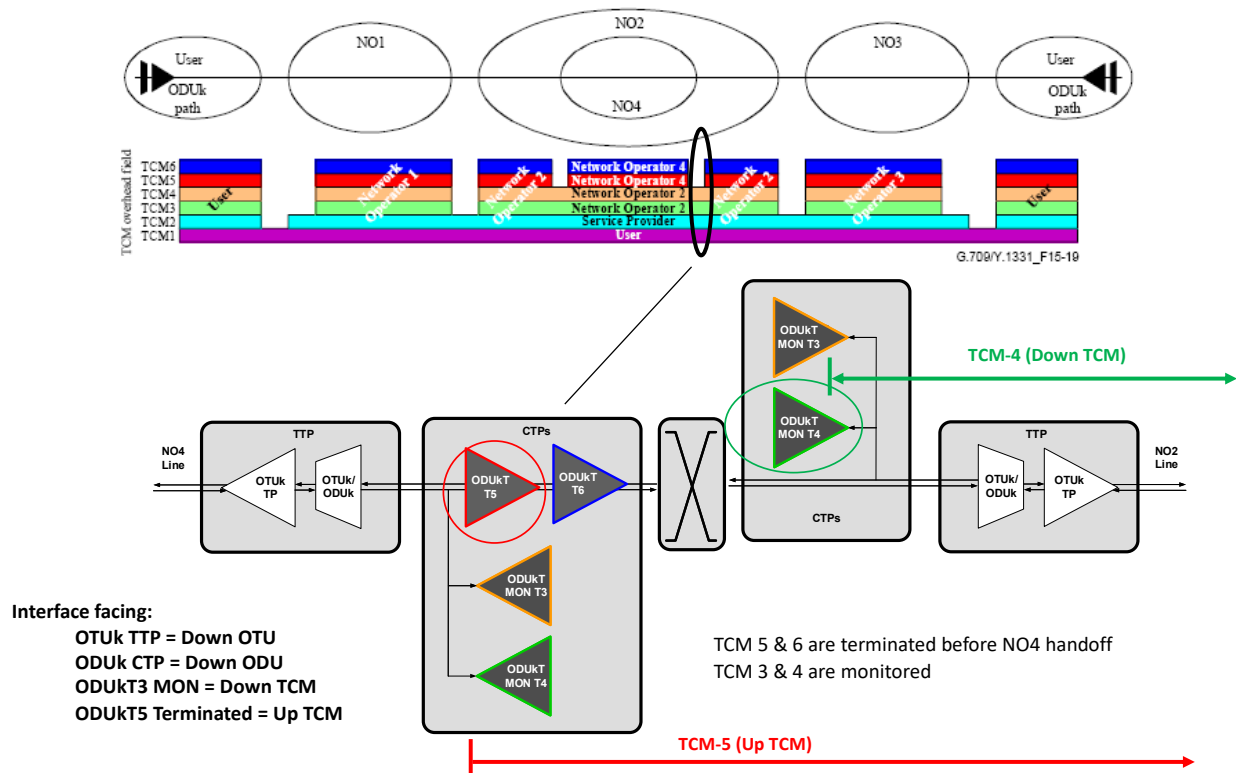


Figure 41: TCM Direction

3.2.13 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 [D].

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-operator*) is generated when any changes to the received TTI attributes (*accepted-sapi*, *accepted-dapi*, *accepted-operator*) occur.

A TTI Mismatch (TIM) alarm (*trailTraceldentifierMismatch*) 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*.

3.2.14 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 [A] (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 3.2.16.2), 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. ODTUCn.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 3.2.12 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.

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

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.

```
+--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*
...

```

3.2.14.1 Sub-Rate OTUCn (OTUCn-M)

Sub-rates are not defined for the OPUCn and ODUCn; they are defined in terms of n x 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*n tributary slots).

OTUCn-M sub-rates can accommodate an OTUCn with an incremental 25Gb/s bit rate, for example:

- 125G OTUC2-25
- 150G OTUC2-30
- 200G OTUC2(-40)
- 250G OTUC3-50
- 300G OTUC3(-60)
- 350G OTUC4-70
- 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 Amd 1 [E] 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*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.

```
+--rw otu
...

```

```

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

```

3.2.14.2 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:

```

+--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?
...

```

FlexO is modeled under OTSi interfaces (refer to Section 3.2.15 for FlexO details):

```

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

```

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 *otsi-group* container is used:

```

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

```

It should be noted that the *otsi-group-id* is mandatory for B100G FlexO (refer to Section 3.2.15 for FlexO details); it is not required for non-FlexO B100G. A series of capabilities are defined against an OTSiG:

```

+--ro otsigroup-capability-profile*
  +--ro profile-name
  +--ro if-cap-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?

```

3.2.14.3 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 42: Logical Port). The *logical-port* can be modeled as a physical hardware port or a true logical port that doesn't exist in the hardware.

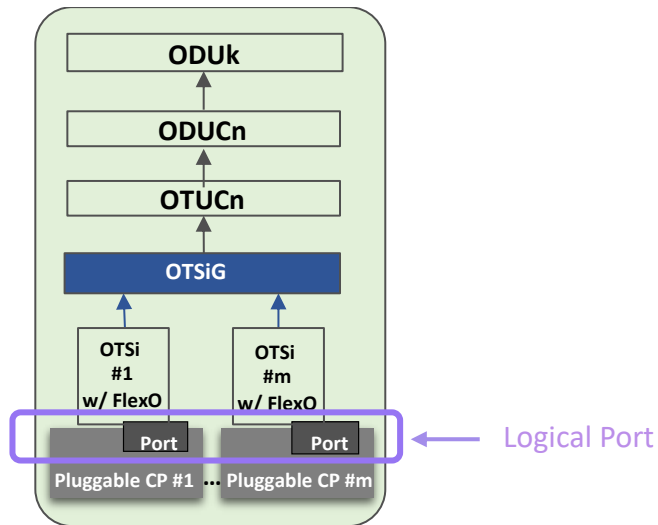


Figure 42: Logical Port

The *logical-port* can be driven by the planned template or automatically created by the device when the associated OTSiG is provisioned (see Section 3.2.14.2 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*.

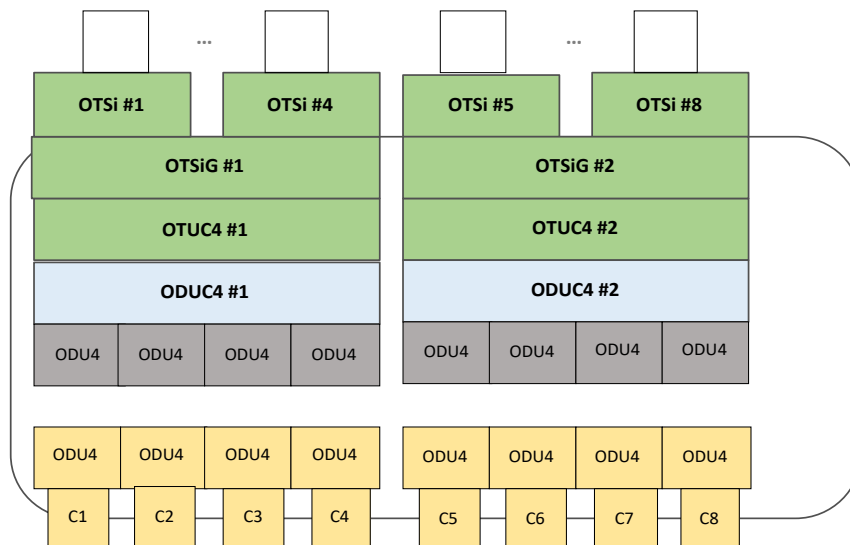


Figure 43: Xponder Example

Figure 43: Xponder Example 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 43: Xponder Example, Table 13: B100G Interface Attributes Example shows the B100G interfaces and how to populate the associated attributes.

Table 13: B100G Interface Attributes Example

interface	supporting-interface-list	supporting-port	supporting-circuit-pack-name
ODUC4 #1	OTUC4 #1		
OTUC4 #1	OTSiG #1	logical-port	circuit-pack
OTSiG #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

specified by Controller

generated by device

3.2.15 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) [DD] for B100G OTN signals called FlexO (ITU-T G.709.1 [H][I][J], ITU-T G.709.3 [K][L][M]). 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:

- a set of 100Gb/s optical PHY streams (n 100Gb/s PHYs)
- a set of 200Gb/s optical PHY streams (m 200Gb/s PHYs, $m = \text{ceiling}(n/2)$)
- a set of 300Gb/s optical PHY streams (m 300Gb/s PHYs, $m = \text{ceiling}(n/3)$) for line-side only
- a set of 400Gb/s optical PHY streams (m 400Gb/s PHYs, $m = \text{ceiling}(n/4)$)

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.

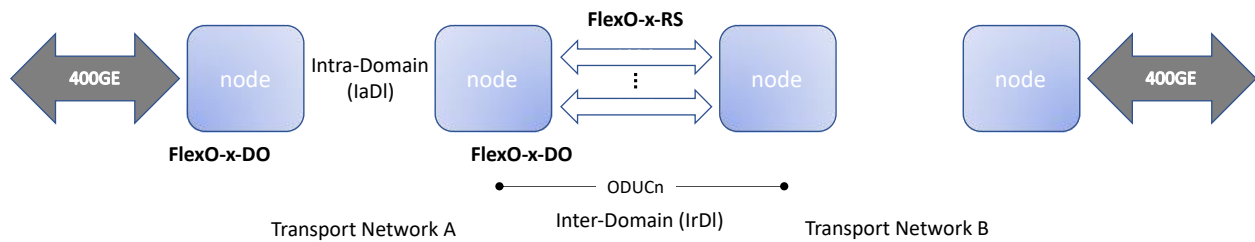


Figure 44: FlexO Line Intra-Domain Interface (IaDI) and FlexO Client Inter-Domain Interface (IrDI)

Referring to Figure 44: FlexO Line Intra-Domain Interface (IaDI) and FlexO Client Inter-Domain Interface (IrDI), FlexO-x-DO (high performance soft-decision FEC implementation) enables the ability to decouple 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 3.2.15.1) and optionally, the FlexO short reach group (see Section 3.2.15.2).

FlexO is modeled under OTSi interfaces:

```

+--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*

```

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.

3.2.15.1 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 45: FlexO-x-DO Bonding). 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 details⁴.

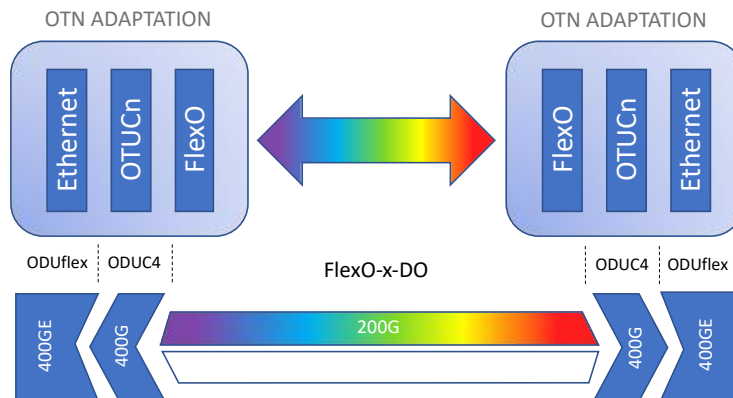


Figure 45: FlexO-x-DO Bonding

In the example shown in Figure 46: FlexO-x-DO 100G Line Interface Containment Example, 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).

⁴ <http://openroadm.org/download.html> OpenROADM MSA3.01 W-Port Digital Specification.docx

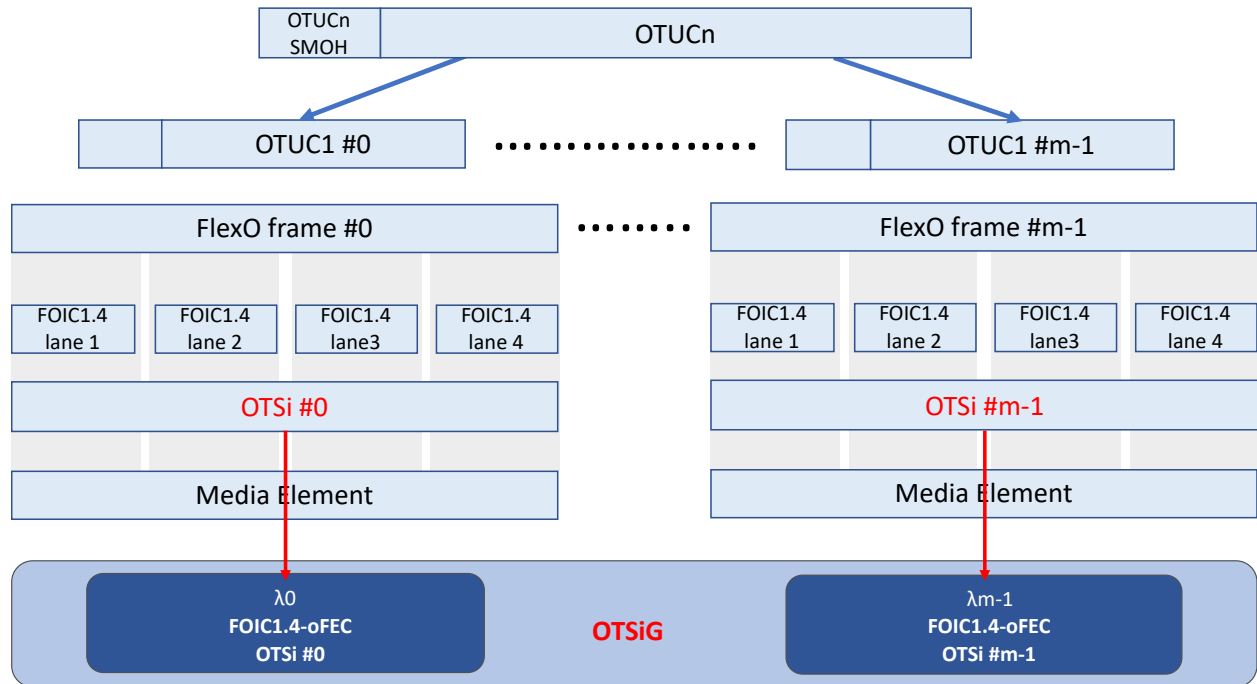


Figure 46: FlexO-x-DO 100G Line Interface Containment Example

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 Cor1 [J]) transported over the same media element.

3.2.15.2 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 47: FlexO-x-RS Sub-Rating and Channelization, FlexO-x-RS is defined for interoperable multi-vendor 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
- 400GBASE-R pluggable optics by interleaving four 100G FlexO “instances” over each 400G PHY

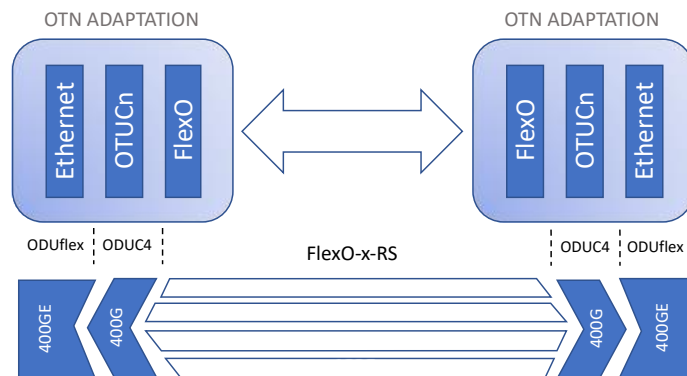


Figure 47: FlexO-x-RS Sub-Rating and Channelization

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 [A]) 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 48: FlexO-x-RS 100G Client Interface Containment Example). 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 [H][I][J] 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).

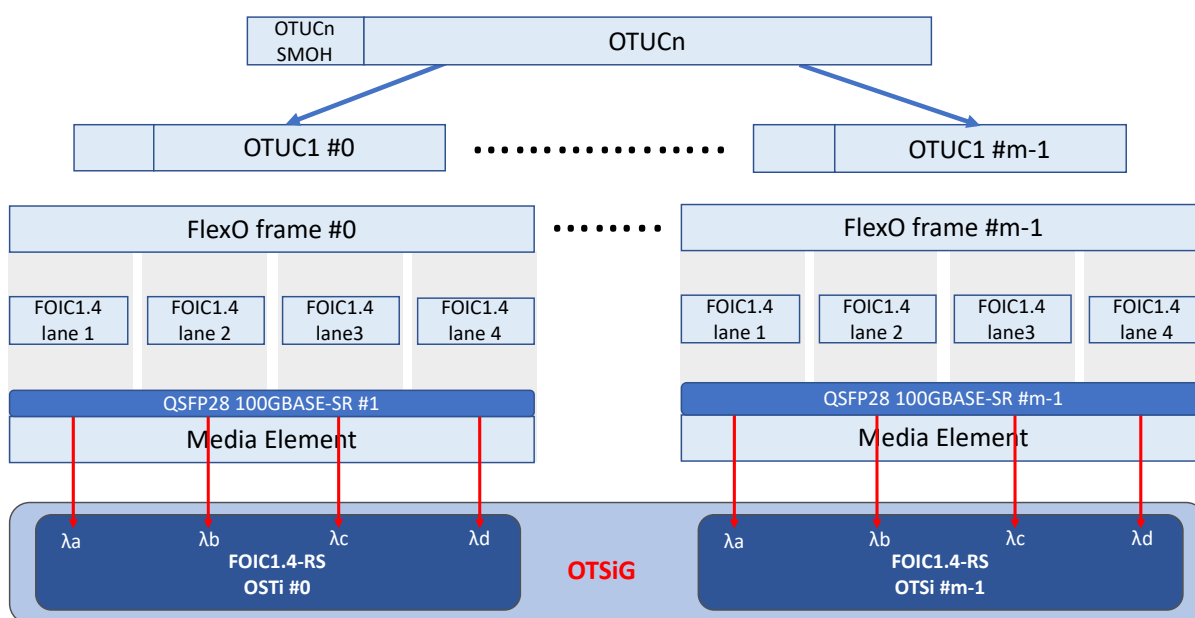


Figure 48: FlexO-x-RS 100G Client Interface Containment Example

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 3.2.2 and Figure 49: FlexO-x-RS Port Model Abstraction).

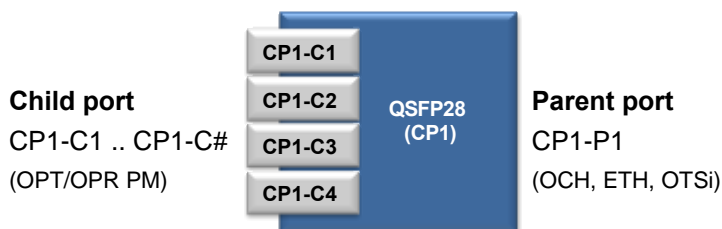


Figure 49: FlexO-x-RS Port Model Abstraction

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.

3.2.16 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 [A]). ODUflex is a single container that cannot be 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.

3.2.16.1 ODUflex for 100Gb/s and Below

3.2.16.1.1 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 [E], *ODUflex-cbr-200G* and *ODUflex-cbr-400G* for 200G/400G CBR signals as defined in ITU-T G.709 17.13.2 [E] and discussed in Section 3.2.16.2.1).

3.2.16.1.2 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 an 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 [A]. **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 50: Sub-rate Ethernet Receive Policing/Rate-Limiting Example, a sub-rate 5 Gbps is transported over 6.25 Gbps ($5 \times \text{ODU2.ts} = 5 \times 1.249 = 6.245$ Gbps). The controller pushes down the same provisioning at both Xponders.

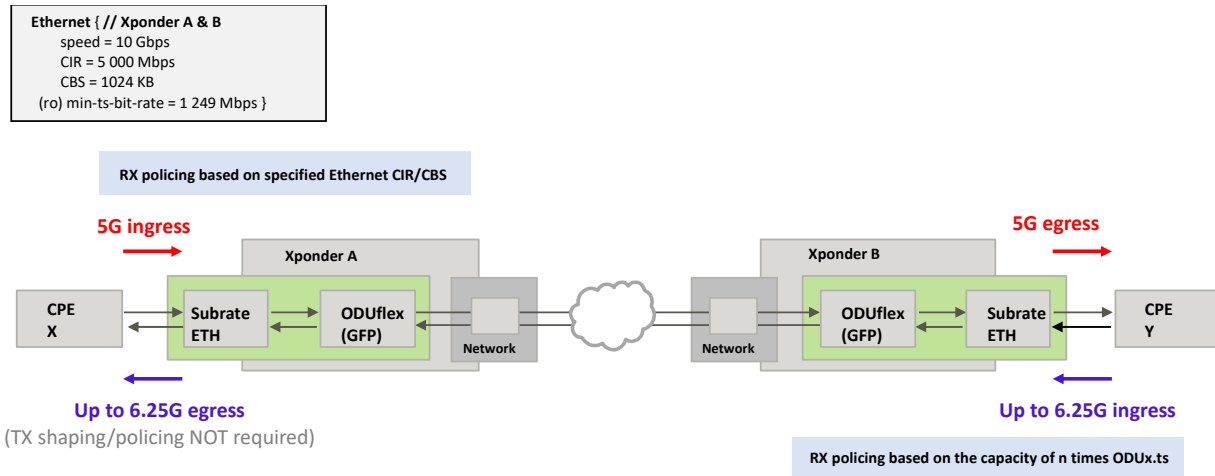


Figure 50: Sub-rate Ethernet Receive Policing/Rate-Limiting Example

3.2.16.2 ODUflex for B100G

3.2.16.2.1 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 3.2.16.1.1). Per ITU-T G.709 Amd 2 clause 17.13 [E], 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 ODTUCn. 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 an 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.

3.2.16.2.2 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 [A].

3.2.16.2.3 ODUflex(FlexE) for up to and B100G

A FlexE client defined by the Optical Interworking Forum (OIF) (OIF-FLEXE-02.0 [DD]) 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-flex-e-n* for FlexE-aware client signals per ITU-T G.709 17.12 [A].

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 DEVICE OPERATION

4.1 SYNCHRONOUS/ASYNCHRONOUS OPERATIONS

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

4.1.1 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 51: Synchronous Operation*). A synchronous operation is typically used for operations that complete in <2min

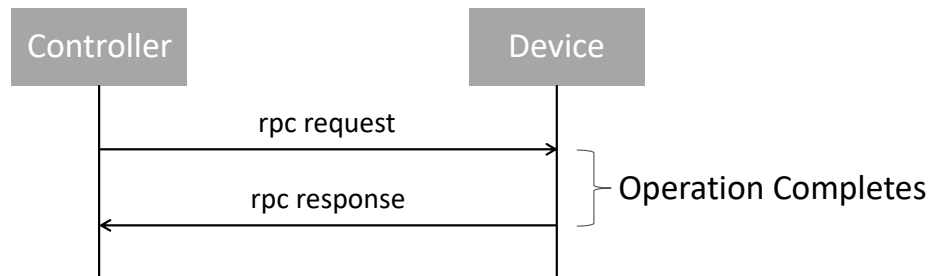


Figure 51: Synchronous Operation

4.1.2 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 52: 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.

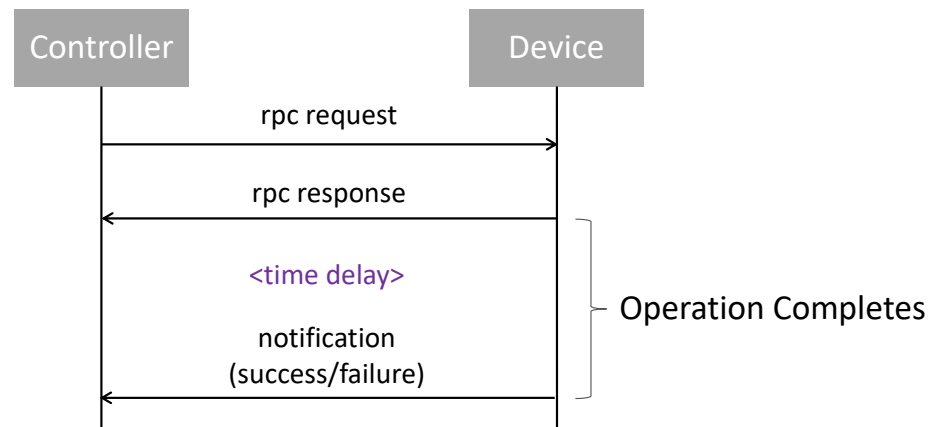


Figure 52: Asynchronous Operation

4.2 FILE OPERATIONS AND STRUCTURE

4.2.1 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.)

4.2.2 Secure Shell File Transfer Protocol (SFTP) Specification

SFTP is a secure file transfer protocol that runs over Secure Shell (SSH) [A], 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.

4.2.3 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.

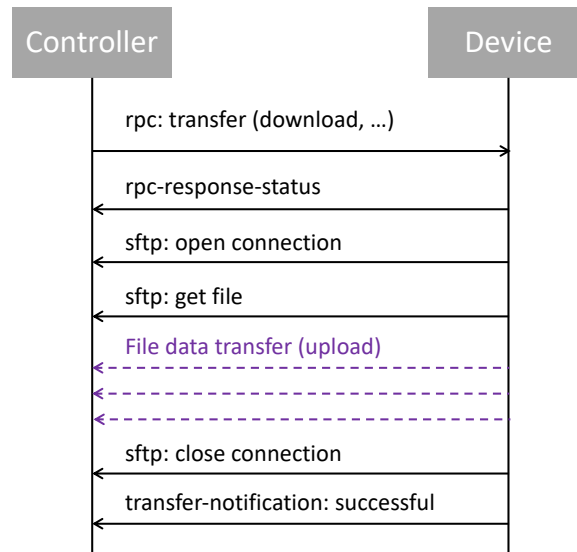


Figure 53: File Transfer (upload)

4.2.4 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.

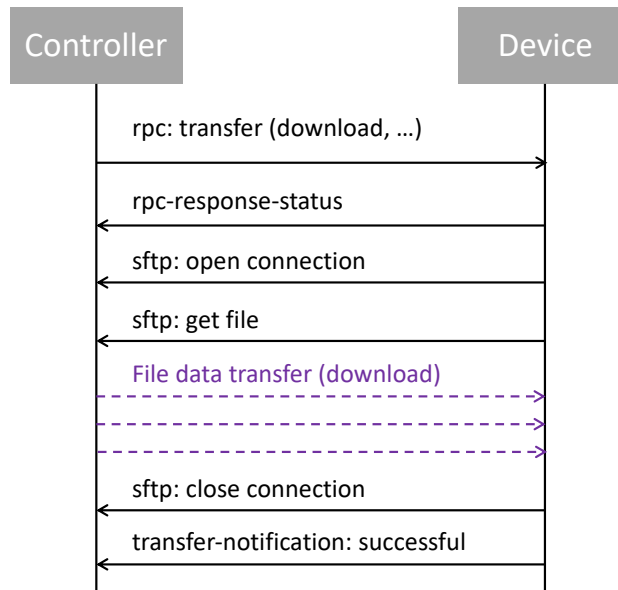


Figure 54: File Transfer (download)

4.2.5 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.

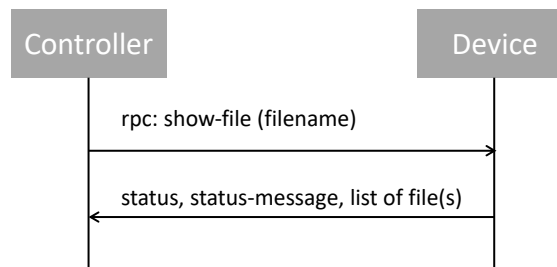


Figure 55: Retrieve File List

4.2.6 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.

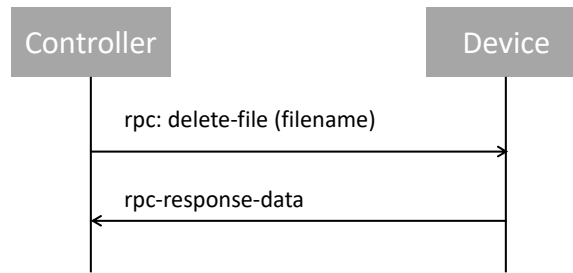


Figure 56: Delete File

4.2.7 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.

4.3 DATA COMMUNICATION NETWORK (DCN)

Figure 57: Open ROADM DCN Architecture 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.

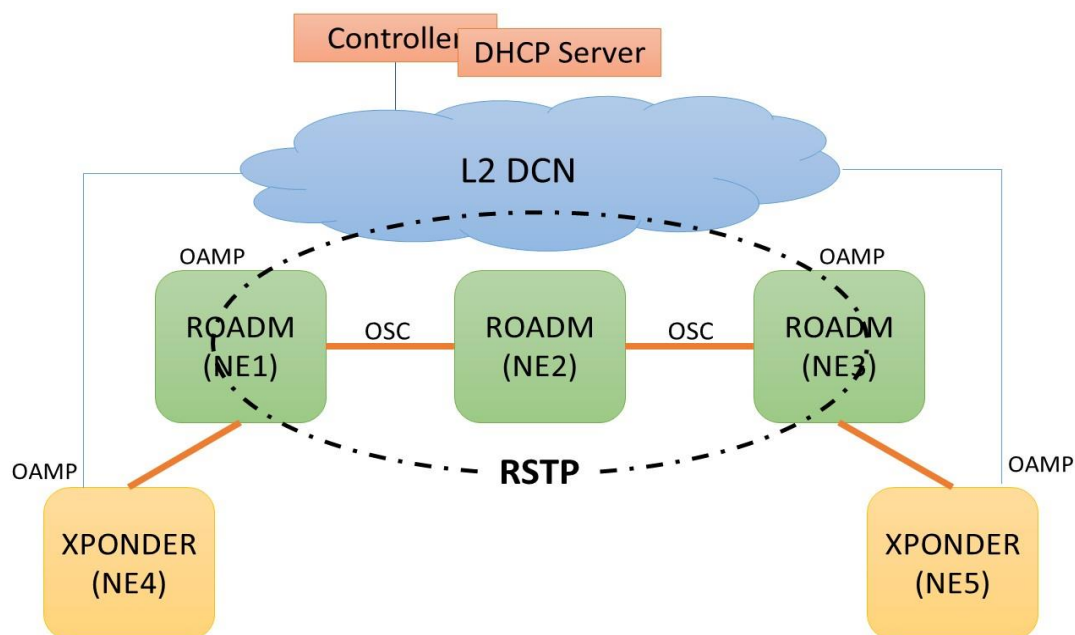


Figure 57: Open ROADM DCN Architecture

For the ROADM Node:

- Each ROADM NE is running in L2 bridging mode
- 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.

4.4 NETCONF PROTOCOL

The Open ROADM MSA device is managed using the NETCONF protocol (per RFC 6241 [DD]) 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.

4.4.1 Secure Shell (SSH) Support

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

Note: In RFC 4742 [X], "[>]" 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 [EE] 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.

4.4.2 Notification Support

Open ROADM MSA devices must support NETCONF event notifications per RFC 5277 [Z]. 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.

4.4.2.1 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 holdoff 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):

```
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/P13']/vendor
        operation replace
    }
    edit {
        target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-
name='200/0/P13']/model
        operation replace
    }
    edit {
        target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-
name='200/0/P13']/serial-id
        operation replace
    }
    edit {
        target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-
name='200/0/P13']/type
        operation replace
    }
    edit {
        target /org-openroadm-device:org-openroadm-device/circuit-packs[circuit-pack-
name='200/0/P13']/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
    }
}
}

```

4.4.2.2 Replay Support

Open ROADM MSA devices must support the ability to request previously logged notifications via a NETCONF replay operation (per RFC 5277 [Z]). 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.

4.4.3 Monitoring Support

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

4.5 SYSLOG PROTOCOL

The Syslog protocol (RFC 5424 [AA]) 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 [C] with the addition of the *local-syslog-filename* attribute to identify the filename of the locally stored syslog file:

```

+--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?](#)

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 4.2.3 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).

A provisioning example to stream all logged event notifications to a remote syslog server using UDP is show below (attribute values shown in *green* can be changed by the operator):

```
<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>
```

4.6 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.

4.6.1 Dial-Out Mode

By default, dial-out mode uses JSON (per RFC7159 [FF]) to encode the YANG data and gNMI *notification* messages to stream the telemetry data (refer to Figure 58: Telemetry Dial-Out Model).

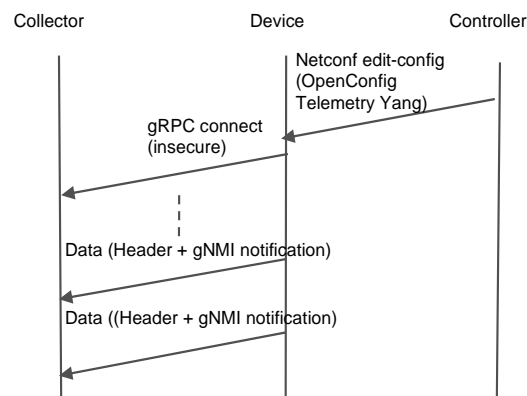


Figure 58: Telemetry Dial-Out Model

As shown in *Figure 58: Telemetry Dial-Out Model*, 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.

```
+--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?
```

- **Creating a Sensor Group:**

The sensor group specifies a list of YANG models to be streamed.

```
+--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?
```

- **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

(<https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md#23-structured-data-types>), the data format must support JSON encoding (per RFC7159 [FF]) 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?

```

4.6.2 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 59: Telemetry Dial-In Mode)

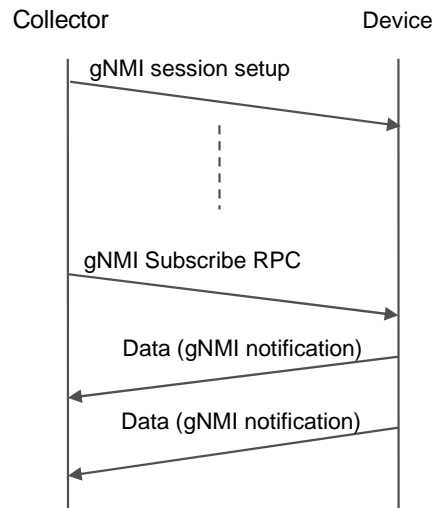


Figure 59: Telemetry Dial-In Mode

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

(<https://github.com/openconfig/reference/blob/master/rpc/gnmi/gnmi-specification.md#23-structured-data-types>) to establish the connection (subscription). As with dial-out mode, the data encoding method is JSON (per RFC7159 [FF]).

As required by the gNMI specification, TLS uses v1.2 (per RFC 5246 [Y]) 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 is a sample subscription using gNMI for dial-in mode:

```

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;
}

```

4.7 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*). **Error! Reference source not found.** defines what the *faceplate-label* should be set to under certain conditions where the *faceplate-label* cannot be determined.

Table 14: Shelf and Circuit-pack Faceplate Label

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

The *circuit-pack/ports/faceplate-label* is intended to represent the physical labeling on the unit associated with a fixed port; it's based on the *circuit-pack-type*, since the Open ROADM MSA device model bases the list of *circuit-pack/ports* on the provisioned type (*circuit-pack-type*).

Pluggable optics have limited faceplate real estate and can be sourced from a number of pluggable vendors. For this reason, a device may not know what the silk-screening on the pluggable module is. In this case, the device should set the *circuit-pack/faceplate-label* and *circuit-pack/ports/faceplate-label* for pluggable optics should both be set to *pluggable*.

For passive equipment, it is typically expected that vendors would be able to display the appropriate faceplate label based on the provisioned *shelf-type* or *circuit-pack-type*. However, if the faceplate label is not known, a vendor may display the faceplate label as *unknown*.

4.8 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,

- i. A resource in “*inService*” administrative-state will discover and forward current alarm notifications on the resource.
- ii. 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.

- iii. 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.
- iv. A device implementation shall support direct transition from one *administrative-state* to another.
- v. 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.
- vi. 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.

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.).

5 DEVICE DISCOVERY, COMMISSIONING, AND AUGMENTATION

5.1 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 60: 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.

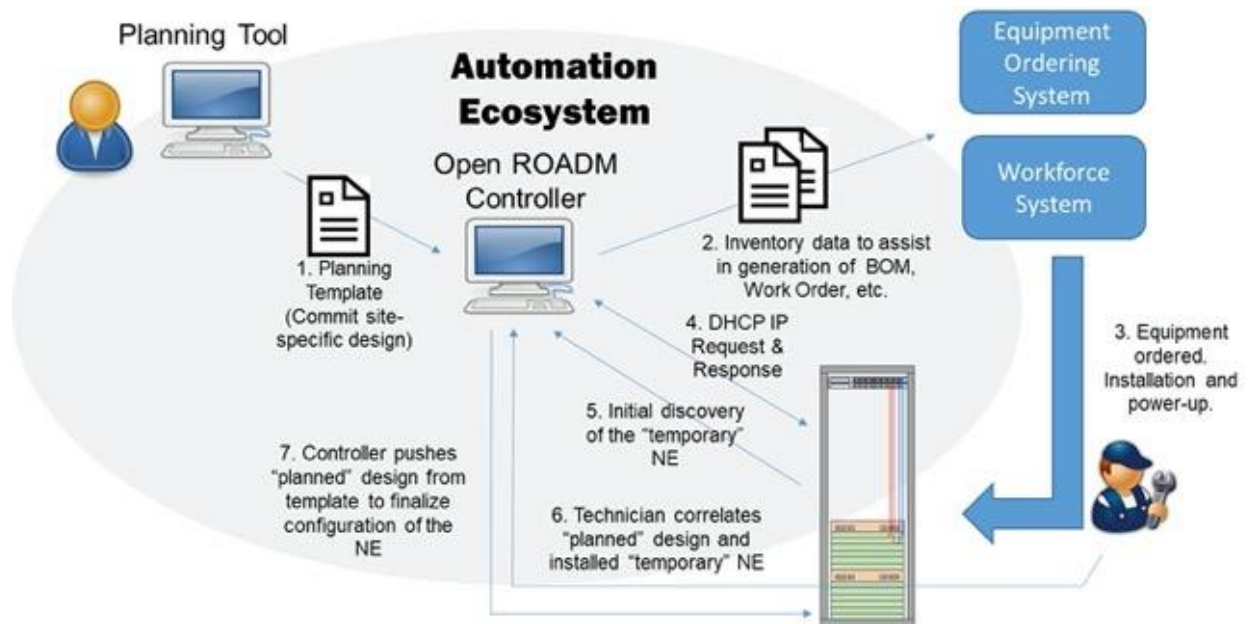


Figure 60: One-Touch Provisioning Device Commissioning Procedure

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 61: Device Discovery and Commissioning States.

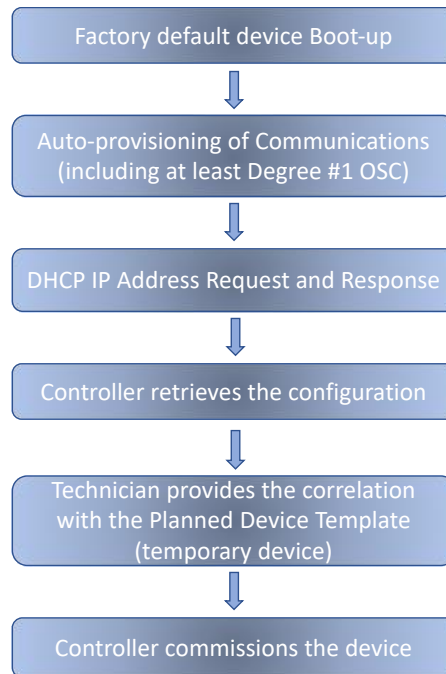


Figure 61: Device Discovery and Commissioning States

5.1.1 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.

5.1.2 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.

5.1.3 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 5.1.6.2 for further details.

5.1.4 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 4.2.2 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.

5.1.5 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.

5.1.6 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).

5.1.6.1 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 Section Appendix A.1 and firmware download details can be found in Section 6.1.3. 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.

5.1.6.2 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.

5.1.6.3 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.)

5.1.6.4 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.

5.1.6.5 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).

5.1.6.6 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.

5.1.6.7 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).

5.1.6.8 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.

5.2 DEVICE AUGMENTATION

Similar to device discovery and commissioning (see Section) 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 5 for 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 below).

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.

5.3 OMS LINK PLANNING AND DISCOVERY

OTN Management System (OMS) link data is pushed to the controller using the Open ROADM MSA network model⁵. The OMS link plan should include the near end and far end degree information on the

⁵ The Open ROADM network model resides on the Open ROADM controller (or other management system) and not on the device.

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:

- 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 9.5 for OTDR scan details) to complete the OMS link commissioning.

6 DEVICE SOFTWARE, FIRMWARE, AND DATABASE OPERATIONS

6.1 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.

6.1.1 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 4.2.4 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 4.2.6 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 15: Software Download Notifications and Pending-SW*)

Table 15: Software Download Notifications and Pending-SW

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
	async	sw-stage-notification	sw-version: new (to be activated) version sw-validation-timer: not present sw-activation-date-time: not present
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	Not present

		after reboot, commit after reboot)	
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

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

6.1.2 Software Downgrade

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

6.1.3 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 6.1.1 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 9.1.2 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.

6.2 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 4.4.2.1).

6.2.1 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 4.2.3 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 4.2.6 for details)

Table 16: Database Backup Operation Notifications, Reboot, and Database-Info

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		

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

6.2.2 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 4.2.4 for SFTP details).
- **db-restore** [*filename*] [*nodeIDCheck*] is used by the device to perform a database restore operation. The *nodeIDCheck* is used to specify whether a database for restoration is a backup of the intended device. If required (*nodeIDCheck=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 4.2.6 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 17: Database Restore Operation Notifications, Reboot, and Database-Info*).

Table 17: Database Restore Operation Notifications, Reboot, and Database-Info

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
	async	db-activate-notification db-active-	Implementation specific	

		notification-type = activate (if reboot required, sent before or after reboot)	(may or may not reboot)	rollback-timer: remaining time of the rollback timer activation-date- time: time of activation of the restored database
db-activate (without rollback)	sync	None	None	last-restored-time: time of DB activate
	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)	rollback-timer: not shown activation-date- time: not shown
cancel-rollback-timer (accept=true)	sync	None	None	last-restored-time: time of DB activate
	async	db-activate- notification db-active- notification-type = commit		rollback-timer: not shown activation-date- time: not shown
cancel-rollback-timer (accept=false)	sync	None	None	last-restored-time: original time of last- restored-time before the db- activate was issued
	async	db-activate- notification db-active- notification-type = cancel (if reboot required, after reboot)	Implementation specific (may or may not reboot)	rollback-timer: not shown activation-date- time: not shown

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
---------------------------	-----------	------------------------------------------------------------------------------------------------------------	----------------------------------------------------	-------------

Refer to Appendix A.3 for database restore manifest file details.

6.2.3 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"

7 PERFORMANCE MONITORING (PM), ALARMS, AND TCAS

7.1 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.

7.1.1 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:

```
<get>
  <filter>
    <currentPmList xmlns="http://org/openroadm/pm">
      <currentPm>
        <granularity>15min</granularity>
      </currentPm>
    </currentPmList>
  </filter>
</get>
```

An example invocation using NETCONF to retrieve the current PM registers with a 24-hour granularity is as follows:

```
<get>
  <filter>
    <currentPmList xmlns="http://org/openroadm/pm">
```

```

        <currentPm>
          <granularity>24Hour</granularity>
        </currentPm>
      </currentPmlist>
    </filter>
  </get>

```

7.1.2 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:

```

<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>

```

Note: support for retrieving the historical PM list using a NETCONF get is optional for vendors to support.

7.1.2.1 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.

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 4.1). 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:

```
<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-device:circuit-
packs[org-openroadm-device:circuit-pack-name='1/0/2/E1']/org-openroadm-device: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-entry>
<historical-pm-entry>
...
```

</historical-pm-entry>
</historical-pm-list>

7.1.2.2 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.

7.1.3 PM Behavior

7.1.3.1 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..

7.1.3.2 Granularity

The PM model supports three granularities: 15-minute, 24-hour, and untimed (*notApplicable*). Refer to Table 18: PM Granularity Support for the requirements that implementations must meet in support of PM granularity.

Table 18: PM Granularity Support

Granularity	Current/ Historical	Analog PM		Digital PM
		Raw	Min/Max/Avg	
Untimed (notApplicable)	Current	<i>Optional</i>	<i>Optional</i>	<i>Mandatory</i>
	Historical	<i>Not Applicable</i>	<i>Not Applicable</i>	<i>Not Applicable</i>
15min	Current	<i>Mandatory</i>	<i>Mandatory</i>	<i>Mandatory</i>
	Historical	<i>Optional</i> <i>(snapshot at end of bin)</i>	<i>Mandatory</i>	<i>Mandatory</i>
24Hour	Current	<i>Mandatory</i>	<i>Mandatory</i>	<i>Mandatory</i>
	Historical	<i>Optional</i> <i>(snapshot at end of bin)</i>	<i>Mandatory</i>	<i>Mandatory</i>

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.

7.1.3.3 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.

7.1.3.4 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.

7.2 ALARMS

For details on supported alarms (probable causes), refer to Appendix B:.

7.3 THRESHOLD CROSSING ALERTS (TCAs)

TCAs can be set against the PM monitor types. TCAs are supported against both analog and digital PM.

- Potential TCA List provides the list of TCAs supported per resource (should be created automatically by the device)
 - Thresholds are set using the potential TCA list
 - Threshold values are absolute threshold values. Relative thresholds are not supported on the device in this version of the MSA
 - Low and High thresholds are supported. Note that not all PM types will support both a high and low threshold. For example, counter type PMs (digital PMs) will only support a high threshold.
- TCA notifications provide the notification when a high or low threshold is crossed.

- TCA notifications are transient in nature (no persistent state on threshold crossing and no clear notifications are sent).

Table 19: TCA Support

Granularity	Current/ Historical	Analog PM		Digital PM
		Raw	Min/Max/Ave	
Untimed (notApplicable)	Current	<i>Not Applicable</i>	<i>Not Applicable</i>	<i>Not Applicable</i>
15min	Current	<i>TCA – Mandatory</i> <i>(high and low threshold</i> <i>where it makes sense)</i>		<i>Mandatory</i>
24Hour	Current			<i>Mandatory</i>

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. 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.

For digital PM, the threshold can be independently set against the 15m or 24h granularity.

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 PROVISIONING ACTION USE CASES

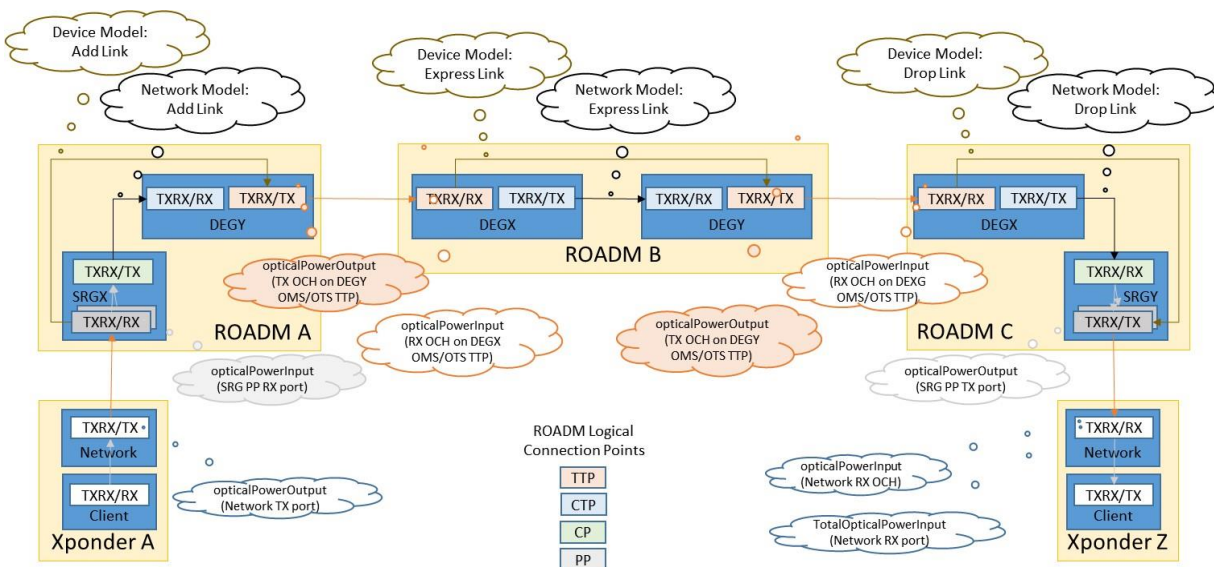


Figure 62: Connection and Growth Management

Figure 62: Connection and Growth Management is used to provide context for this section.

8.1 CONNECTION MANAGEMENT

This section will discuss the different basic operations needed to configure interfaces and connections on the ROADM and Xponder nodes. Interface and connection deletion is presented in a second step. These actions are performed through a device's NETCONF interfaces. XML encoding is not detailed and only basic NETCONF operations are presented in order to simplify and alleviate the description.

In the same way, because some ports/interfaces can be either uni (TX/RX) or bi-directional (TXRX), we try to be as generic as possible in the action sequence description. Consequently, such ports will be specified as [TXRX/TX] or [TXRX/RX].

8.1.1 Xponder Interfaces Creation (TX)

Configures the ports and interfaces on both line and client sides of a xponder in the TX direction. Xponder interfaces are supposed to be bi-directional. However, we describe the sequence of actions following the provisioning methodology; where the connections are configured sequentially across the pre-calculated path in one direction first, and then in the reverse direction. Thus, we only give precision on the elements and configuration actions that will be engaged at a time for a specific purpose. Real implementation may differ from the proposed sequence of action. As another example, we consider applying power settings just after the connection is created, but they could be performed in a second step.

This example considers the case of transponder equipment mapping a single 100GE client signal into a 100G OTU4 DWDM signal with a modulation format of QPSK.

Input parameters:

- *node-id* (xponder X)
- xponder network *circuit-pack-name* (xponder X network circuit pack)
- xponder network *port-name* (xponder X network port)
- xponder client *circuit-pack-name* (xponder X client circuit pack)
- xponder client *port-name* (xponder X client port)
- *frequency*
- *rate*
- *modulation-format*
- *transmit-power*

Main actions sequence:

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]/ports[port-name= "<xponder X network port>"]/)

retrieve xponder X network port *transponder-port*/port-power-capability (min/max tx)

retrieve xponder X network port supported-interface-capability

Get-config (org-openroadm-device/interface[supporting-circuit-pack-name="< xponder X network circuit pack >" and supporting-port="< xponder X network port >"])

Check if OCH interface present on network port and the status of the port

If NO OCH corresponding interface created on the Xponder network port

(either bi-directional port or uni-directional TX port)

equipment-state of circuit-pack and admin-state of network-port shall be set appropriately:

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]) nc:operation=replace

set equipment-state of OCH interface supporting circuit-pack to not-reserved-inuse

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]/ports[port-name= "<xponder X network port>"]) nc:operation=replace

set admin state of OCH interface supporting port to inService

OCH interface shall be created on network port:

edit-config (org-openroadm-device/interface>") nc:operation=create

create an OCH interface XPDR<n>-NETWORK<m>-[TXRX/TX]-[frequency] on supporting port

set name = "<XPDR<n>-NETWORK<m>-[TXRX/TX]-[frequency] >"

set type to opticalChannel

set supporting-circuit-pack-name="< xponder X network circuit pack >"

set supporting-port ="< xponder X network port >"

set rate, frequency, modulation-format (QPSK)

set admin state of OCH interface to inService

OTU4 interface shall be created on supporting OCH interface:

edit-config (org-openroadm-device/interface) nc:operation=create

create an otnOtu interface *OTU4* on supporting port/OCH interface

set name = "<OTU4-XPDR<n>-NETWORK<m>-TXRX/TX>"

set type to otnOtu

```

set supporting-circuit-pack-name="< xponder X network circuit pack >"
set supporting-port ="< xponder X network port >"
set supporting-interfaces ="< OCH interface>"
set admin state of OTU4 interface to inService

```

ODU4 interface shall be created on supporting OTU4 interface:

```

edit-config (org-openroadm-device/interface) nc:operation=create
create an otnOdu interface ODU4 on supporting port/OTU4 interface
set name = "<ODU4-XPDR<n>-NETWORK<m>-TXRX/TX>"
set type to otnOdu
set supporting-circuit-pack-name="< xponder X network circuit pack >"
set supporting-port ="< xponder X network port >"
set supporting-interfaces "<OTU4-XPDR<n>-NETWORK<m>-TXRX/TX>"
set admin state of ODU4-1 interface to inService

```

Output power shall be configured on the OCH interface:

```

edit-config (org-openroadm-device/interface[name= "<XPDR<n>-NETWORK<m>-[TXRX/TX]-
[frequency]>"]/och/transmit-power>) nc:operation=replace
set transmit-power to output-power

```

The controller verifies that the output power reaches the set value (verify pmParameterName/type = opticalPowerOutput):

```

Get-config (/currentPmList/currentPm/currentPmList/currentPm[resource/resource/port-name"<
xponder X network port >"and granularity="15min"])

```

The controller verifies that connection between client and network xponder circuit packs and ports exists:

```

Optional: 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>"])

```

Check if interface present on client port and the status of the port

```

Get-config (org-openroadm-device/interface[supporting-circuit-pack-name="< xponder X client circuit
pack >" and supporting-port-name="< xponder X client port >"])

```

If NO corresponding interface created on the Xponder client port

(either bi-directional port or uni-directional RX port)

equipment-state of circuit-pack and admin-state of network-port shall be set appropriately :

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit
pack>"]) nc:operation=replace

```

set equipment-state of 100GE interface supporting circuit-pack to not-reserved-inuse

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit
pack>"]/ports[port-name= "<xponder X client port>"]) nc:operation=replace

```

set admin state of 100GE interface supporting port to inService

100GE Ethernet interface shall be created on client port:

```

edit-config (org-openroadm-device/interface) nc:operation=create

```

create a 100GE interface 100GE on supporting port

```

set name = "<100GE-XPDR<n>-CLIENT<m>-TXRX/TX>"
set type to ethernetCsmacd

```



```

set supporting-circuit-pack-name="< xponder X client circuit pack >"
set supporting-port ="< xponder X client port >"
set speed, fec, duplex, mtu, auto-negotiation, curr-speed
set admin state of 100GE interface to inService

```

End of sequence

8.1.2 Add-Link Creation

Configures the ports, interfaces and connections for an add link from a SRG-PP to Degree TTP logical point. One shall note that the power settings could be applied in a second step, after the interfaces and the connection have been created. Some actions are optional and may differ from one implementation to another.

This example considers the case of a ROADM connection between two nmc-ctp where on the degree side the nmc-ctp interface is the only child of a mc-ttp interface. Furthermore nmc-ctp and mc-ttp interfaces have the same center frequency and NMC width is included inside the MC width.

Input parameters:

- *node-id* (SRGX, DEGY)
- *circuit-pack* (SRGX CircuitPack, DEGY CircuitPack)
- *supporting-port* (SRGX-PPN, DEGY-TTP)
- *min-freq* and *max-freq* for the mc-ttp interface
- *frequency* and *width* for the nmc-ctp interface
- calculated output power

Main actions sequence:

When optional is mentioned, it usually refers to a step where the controller performs some verification steps that are not mandatory.

```

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<SRGX CircuitPack >"]/ports [port-
name= "<SRGX-PPN >"]/interfaces)

```

If NO nmc-ctp (bi-directional) corresponding interface created on the SRGX-PPN

```

edit-config(org-openroadm-device/circuit-packs[circuit-pack-name= "<SRGX CircuitPack >"]/ports
[port-name= "<SRGX-PPN >"]/) nc:operation=replace

```

```

set admin state of nmc-ctp interface supporting PP port to inService

```

```

edit-config (org-openroadm-device/interface) nc:operation=create

```

```

create nmc-ctp interface 1 (SRGX-PPN-[TXRX/RX]-[frequency]) on supporting PP port

```

```

set name = "<SRGX-PPN-[TXRX/RX]-[ frequency] >"

```

```

set type to networkMediaChannelConnectionTerminationPoint

```

```

set supporting-circuit-pack-name= "< SRGX CircuitPack >"

```

```

set supporting-port = "< SRGX-PPN >"

```

```

set frequency

```

```

set width

```

```

set admin state of nmc-ctp interface inService

```

```

Get-config (org-openroadm-device/ circuit-packs[circuit-pack-name= "<DEGY-CircuitPack>"]/ports[port-
name= "<DEGY-TTP>/interfaces")

```

If NO OTS (bi-directional) corresponding interface created on the DEGY-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create OTS-interface (OTS-DEGY-TTP-[TXRX/TX]) on supporting TTP port
  set name = "<OTS-DEGY-TTP-[TXRX/TX]>"
  set type to opticalTransport
  set supporting-circuit-pack-name= "< DEGY CircuitPack >"
  set supporting-port = "< DEGY-TTP >")
  set admin state of OTS-interface to inService
```

If NO OMS (bi-directional) corresponding interface created on the DEGY-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create OMS-interface (OMS-DEGY-TTP-[TXRX/TX]) on supporting TTP port
  set name = "<OMS-DEGY-TTP-[TXRX/TX]>"
  set type to openROADMOpticalMuiltiplex
  set supporting-circuit-pack-name= "< DEGY CircuitPack >"
  set supporting-port = "< DEGY-TTP >")
  set supporting-interfaces = "<OTS-DEGY-TTP-[TXRX/TX]>"
  set admin state of OMS-interface to inService
```

If NO mc-ttp (bi-directional) corresponding interface created on the DEGY-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create mc-ttp interface (mmc-ttp-DEGY-TTP-[TXRX/TX]-[frequency]) on supporting TTP port
  set name = "<mmc-ttp-DEGY-TTP-[TXRX/TX]-[frequency]>"
  set type to mediaChannelTrailTerminationPoint
  set supporting-circuit-pack-name= "< DEGY CircuitPack >"
  set supporting-port = "< DEGY-TTP >")
  set supporting-interfaces = "<OMS-DEGY-TTP-[TXRX/TX]>"
  set min-freq
  set max-freq
  set admin state of mc-ttp-interface to inService
```

If NO nmc-ctp (bi-directional) corresponding interface created on the DEGY-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create nmc-ctp-interface2 (nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]) on supporting TTP port
  set name = "<nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"
  set type to networkMediaChannelConnectionTerminationPoint
  set supporting-circuit-pack-name= "< DEGY CircuitPack >"
  set supporting-port = "< DEGY-TTP >")
  set supporting-interfaces = "<mc-ttp-DEGY-TTP-[TXRX/TX]-[frequency]>"
  set frequency
  set width
  set admin state of nmc-ctp-interface2 to inService
```

`edit-config (org-openroadm-device/roadm-connections) nc:operation=create`

Set connection from nmc-ctp-Interface1 to nmc-ctp-Interface 2

Set connection-name = "<SRGX-PPN-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"

```
set source/src-if = "<SRGX-PPN-[TXRX/RX]-[ frequency] >"
set destination/dst-if = "<nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"
```

Power setting and control-mode change could be made in a second step after all connections are created

```
edit-config (org-openroadm-device/roadm-connections[connection-name= "<SRGX-PPN-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=replace
```

```
set optical control-mode = power
```

```
set target-output-power = calculated output power
```

In a final step, the controller verifies that the output power reaches the set value (verify pmParameterName/type = opticalPowerOutput):

```
Get-config (/currentPmList/currentPm[resource/resource/interface-name="< nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>" and granularity="15min"])
```

And once current-output-power has converged towards expected output power, control-mode is moved to gainLoss

```
edit-config (org-openroadm-device/roadm-connections[connection-name= "<SRGX-PPN-[TXRX/RX]-[frequency]-to- nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=replace
```

```
Connection from nmc-ctp-Interface 1(PP) to OCH-Interface2 (TTP)
```

```
set optical control-mode = gainLoss
```

End of sequence

8.1.3 Express Link Creation

Configures the ports, interfaces and connections for an express link between 2 Degree TTP logical points in one direction. One shall note that the power settings could be applied in a second step, after the interfaces and the connections have been created. Some actions are optional and may differ from one implementation to another.

This example considers the case of a ROADM connection between two nmc-ctp where on both degree sides the nmc-ctp interfaces are the only children of a mc-ttp interface. Furthermore nmc-ctp and mc-ttp interfaces have the same center frequency and NMC width is included in MC width.

A bi-directional express link will be created in 2 steps (connections are uni-directional) applying this sequence in both directions.

Input parameters:

- *node-id* (DEGX, DEGY)
- *circuit-pack* (DEGX CircuitPack, DEGY CircuitPack)
- *supporting-port* (DEGX-TTP, DEGY-TTP)
- *min-freq* and *max-freq* for mc-ttp interfaces
- *frequency* and *width* for nmc-ctp interfaces
- expected input power
- calculated output power

Main actions sequence:

```
Get-config (org-openroadm-device/ circuit-packs[circuit-pack-name= "<DEGX-CircuitPack>"]/ports[port-name= "<DEGX-TTP>"]/interfaces)
```

If NO OTS (bi-directional) corresponding interface created on the DEGX-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create OTS-interface (OTS-DEGX-TTP-[TXRX/RX]) on supporting TTP port
set name = "<OTS-DEGX-TTP-[TXRX/RX]>"
set type to opticalTransport
set supporting-circuit-pack-name= "< DEGX CircuitPack >"
set supporting-port = "< DEGX-TTP >")
set admin state of OTS-interface to inService
```

If NO OMS (bi-directional) corresponding interface created on the DEGX-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create OMS-interface (OMS-DEGX-TTP-[TXRX/RX]) on supporting TTP port
set name = "<OMS-DEGX-TTP-[TXRX/RX]>"
set type to openROADMOpticalMuiltiplex
set supporting-circuit-pack-name= "< DEGX CircuitPack >"
set supporting-port = "< DEGX-TTP >")
set supporting-interfaces ="<OTS-DEGX-TTP-[TXRX/RX]>"
set admin state of OMS-interface to inService
```

If NO mc-ttp (bi-directional) corresponding interface created on the DEGX-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create mc-ttp-interface1 (mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]) on supporting TTP port
set name = "<mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]>"
set type to mediaChannelTrailTerminationPoint
set supporting-circuit-pack-name= "< DEGX CircuitPack >"
set supporting-port = "< DEGX-TTP >")
set supporting-interfaces ="<OMS-DEGX-TTP-[TXRX/RX]>"
set min-freq
set max-freq
set admin state of mmc-ttp-interface1 to inService
```

If NO nmc-ctp (bi-directional) corresponding interface created on the DEGX-TTP

`edit-config (org-openroadm-device/interface) nc:operation=create`

```
create nmc-ctp-interface1 (DEGX-TTP-[TXRX/RX]-[frequency]) on supporting TTP port
set name = "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]>"
set type to networkMediaChannelConnectionTerminationPoint
set supporting-circuit-pack-name= "< DEGX CircuitPack >"
set supporting-port = "< DEGX-TTP >")
set supporting-interfaces ="<mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]>"
set frequency
set width
set admin state of nmc-ctp-interface1 to inService
```

The following step may be optional and aims at verifying that input power is compliant with MSA specifications (verify pmParameterName/type = opticalPowerInput):

Optional : Get-config (/currentPmlist/currentPm[resource/resource/interface-name="<nmc-ctpDEGX-TTP-[TXRX/TX]-[frequency]>" and granularity="15min"])

retrieve opticalPowerInput

compare to expected input power derived from MSA specifications : per channel output power of preceding span (per channel output power offset diagram)- span loss

Get-config (org-openroadm-device/ circuit-packs[circuit-pack-name= "<DEGY-CircuitPack>"]/ports [port-name= "<DEGY-TTP>"]/interfaces")

If NO OTS (bi-directional) corresponding interface created on the DEGY-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

create OTS-interface (OTS-DEGY-TTP-[TXRX/TX]) on supporting TTP port

set name = "<OTS-DEGY-TTP-[TXRX/TX]>"

set type to opticalTransport

set supporting-circuit-pack-name= "< DEGY CircuitPack >"

set supporting-port = "< DEGY-TTP >")

set admin state of OTS-interface to inService

If NO OMS (bi-directional) corresponding interface created on the DEGY-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

create OMS-interface (OMS-DEGY-TTP-[TXRX/TX]) on supporting TTP port

set name = "<OMS-DEGY-TTP-[TXRX/TX]>"

set type to openROADMOpticalMux

set supporting-circuit-pack-name= "< DEGY CircuitPack >"

set supporting-port = "< DEGY-TTP >")

set supporting-interfaces ="<OTS-DEGY-TTP-[TXRX/TX]>"

set admin state of OMS-interface to inService

If NO mc-ttp (bi-directional) corresponding interface created on the DEGY-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

create mc-ttp-interface2 (DEGY-TTP-[TXRX/TX]-[frequency]) on supporting TTP port

set name = "<mc-ttp-DEGY-TTP-[TXRX/TX]-[frequency]>"

set type to mediaChannelTrailTerminationPoint

set supporting-circuit-pack-name= "< DEGY CircuitPack >"

set supporting-port = "< DEGY-TTP >")

set supporting-interfaces ="<OMS-DEGY-TTP-[TXRX/TX]>"

set min-freq

set max-freq

set admin state of mmc-ttp-interface2 to inService

If NO nmc-ctp (bi-directional) corresponding interface created on the DEGY-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

create nmc-ctp-interface2 (DEGY-TTP-[TXRX/TX]-[frequency]) on supporting TTP port

set name = "<nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"

set type to networkMediaChannelConnectionTerminationPoint

set supporting-circuit-pack-name= "< DEGY CircuitPack >"

set supporting-port = "< DEGY-TTP >")

```

set supporting-interfaces = "<mc-ttp-DEGY-TTP-[TXRX/TX]-[frequency]>"
set frequency
set width
set admin state of nmc-ctp-interface2 to inService

```

edit-config (org-openroadm-device/roadm-connections) nc:operation=create

```

set connection from nmc-ctp-Interface1 to nmc-ctp-Interface 2
set connection-name = "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-
TTP-[TXRX/TX]-[frequency]>"
set source/src-if = "<nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency] >"
set destination/dst-if = "<nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"

```

Power setting and control-mode change could be made in a second step after all connections are created

edit-config (org-openroadm-device/roadm-connections[connection-name= "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=replace

```

set optical control-mode = power
set target-output-power = calculated output power

```

The output power is calculated from MSA specifications: per channel output power offset diagram)

In a final step, the controller verifies that the output power reaches the set value (verify pmParameterName/type = opticalPowerOutput):

Get-config (/currentPmList/currentPm[resource/resource/interface-name="< nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>" and granularity="15min"])

And once current-output-power has converged towards expected output power, control-mode is moved to gainLoss

edit-config (org-openroadm-device/roadm-connections[connection-name= "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=replace

```

Change optical-control-mode of connection from nmc-ctp-Interface 1(TTP) to nmc-ctp-Interface2 (TTP)
set optical control-mode = gainLoss

```

End of Sequence

8.1.4 Drop-Link Creation

Configures the ports, interfaces and connections for drop link between a TTP and a PP logical points. One shall note that the power settings could be applied in a second step, after the interfaces and the connections have been created. Some actions are optional and may differ from one implementation to another.

This example considers the case of a ROADM connection between two nmc-ctp where on the degree side the nmc-ctp interface is the only child of a mc-ttp interface. Furthermore nmc-ctp and mc-ttp interfaces have the same center frequency and NMC width is included in MC width.

Input parameters:

- *node-id* (DEGX, SRGY)
- *supporting-circuit-pack* (DEGX CircuitPack, SRGY CircuitPack)
- *supporting-port* (DEGX-TTP, SRGY-PPM)
- *min-freq* and *max-freq* for mc-ttp interface
- *frequency* and *width* for nmc-ctp interfaces

- calculated output power
- expected input power

Main actions sequence:

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<DEGX-CircuitPack>"]/ports[port-name= "<DEGX-TTP>"]/interfaces)

If NO OTS (bi-directional) corresponding interface created on the DEGX-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

```
create OTS-interface (OTS-DEGX-TTP-[TXRX/RX]) on supporting TTP port
  set name = "<OTS-DEGX-TTP-[TXRX/RX]>"
  set type to opticalTransport
  set supporting-circuit-pack-name= "< DEGX CircuitPack >"
  set supporting-port = "< DEGX-TTP >")
  set admin state of OTS-interface to inService
```

If NO OMS (bi-directional) corresponding interface created on the DEGX-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

```
create OMS-interface (OMS-DEGX-TTP-[TXRX/RX]) on supporting TTP port
  set name = "<OMS-DEGX-TTP-[TXRX/RX]>"
  set type to openROADMOpticalMuiltiplex
  set supporting-circuit-pack-name= "< DEGX CircuitPack >"
  set supporting-port = "< DEGX-TTP >")
  set supporting-interfaces ="<OTS-DEGX-TTP-[TXRX/RX]>"
  set admin state of OMS-interface to inService
```

If NO mc-ttp (bi-directional) corresponding interface created on the DEGX-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

```
create mc-ttp-interface (mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]) on supporting TTP port
  set name = "<mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]>"
  set type to mediaChannelTrailTerminationPoint
  set supporting-circuit-pack-name= "< DEGX CircuitPack >"
  set supporting-port = "< DEGX-TTP >")
  set supporting-interfaces ="<OMS-DEGX-TTP-[TXRX/RX]>"
  set min-freq
  set max-freq
  set admin state of mc-ttp-interface to inService
```

If NO nmc-ctp (bi-directional) corresponding interface created on the DEGX-TTP

edit-config (org-openroadm-device/interface) nc:operation=create

```
create nmc-ctp-interface1 (DEGX-TTP-[TXRX/RX]-[frequency]) on supporting TTP port
  set name = "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]>"
  set type to networkMediaChannelConnectionTerminationPoint
  set supporting-circuit-pack-name= "< DEGX CircuitPack >"
  set supporting-port = "< DEGX-TTP >")
  set supporting-interfaces ="<mc-ttp-DEGX-TTP-[TXRX/RX]-[frequency]>"
```

```

set frequency
set width
set admin state of nmc-ctp-interface1 to inService

```

The following step may be optional and aims at verifying that input power is compliant with MSA specifications (verify pmParameterName/type = opticalPowerInput):

Optional: Get-config (/currentPmList/currentPm[resource/resource/interface-name="<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]>" and granularity="15min"])

```

retrieve opticalPowerInput

```

```

compare to expected input power derived from MSA specifications : per channel output power
of preceding span (per channel output power offset diagram)- span loss

```

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<SRGY CircuitPack >"]/ports[port-name= "<SRGY-PPN >"]/interfaces)

If NO nmc-ctp (bi-directional) corresponding interface created on the SRGY-PPN

```

edit-config(org-openroadm-device/circuit-packs/[circuit-pack-name= "<SRGY CircuitPack >"]/ports
/[port-name= "<SRGY-PPN >"]/) nc:operation=replace

```

```

set admin state of nmc-ctp interface supporting PP port to inService

```

```

edit-config (org-openroadm-device/interface) nc:operation=create

```

```

create nmc-ctp-interface 2 (SRGY-PPN-[TXRX/TX]-[frequency]) on supporting PP port

```

```

set name = "<SRGY-PPN-[TXRX/TX]-[ frequency] >"

```

```

set type to networkMediaChannelConnectionTerminationPoint

```

```

set supporting-circuit-pack-name= "< SRGY CircuitPack >"

```

```

set supporting-port = "< SRGY-PPN >"

```

```

set frequency

```

```

set width

```

```

set admin state of nmc-ctp interface 2 to inService

```

```

edit-config (org-openroadm-device/roadm-connections) nc:operation=create

```

```

Set connection from nmc-ctp-interface1 to nmc-ctp-interface2 nmc-ctp-DEGX-TTP-[TXRX/RX]-
[frequency]-to- SRGY-PPM-[TXRX/TX]-[frequency]

```

```

Set connection-name = "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to- SRGY-PPM-[TXRX/TX]-
[frequency]>"

```

```

set source/src-if = "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]>"

```

```

set destination/dst-if = "<SRGY-PPN-[TXRX/TX]-[ frequency] >"

```

```

set optical control-mode = power

```

In a final step, the controller verifies that the output power reaches the set value (verify pmParameterName/type = opticalPowerOutput):

Get-config (/currentPmList/currentPm[resource/resource/port-name="<SRGY-PPN>" and granularity="15min"])

```

retrieve SRG opticalPowerOutput

```

End of sequence

8.1.5 Xponder Interfaces Creation (RX)

Configures the ports and interfaces on both line and client sides of xponder equipment in the RX direction.

This example considers the case of transponder equipment mapping a single 100GE client signal into a 100G OTU4 DWDM signal with a modulation format of QPSK.

Input parameters:

- *node-id* (xponder X)
- xponder network *circuit-pack-name* (xponder X network circuit pack)
- xponder network *port-name* (xponder X network port)
- xponder client *circuit-pack-name* (xponder X client circuit pack)
- xponder client *port-name* (xponder X client port)
- SRGY supporting port (SRGY-PPN)
- *frequency*
- *rate*
- modulation format

Main actions sequence:

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]/ports[port-name= "<xponder X network port>"]/transponder-port)

retrieve xponder X network port **port-power-capability** (min/max rx)

retrieve xponder X network port **supported-interface-capability**

Optional: Check SRGY-PPN opticalPowerOutput retrieved in drop-link creation is compatible with min-max rx-port-power-capability

Get-config (org-openroadm-device/interface[supporting-circuit-pack-name="< xponder X network circuit pack >" and supporting-port-name="< xponder X network port >"])

Check if OCH interface present on network port and the status of the port

If NO OCH corresponding interface created on the Xponder network port

(either bi-directional port or uni-directional RX port)

equipment-state of circuit-pack and admin-state of network-port shall be set appropriately :

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]) nc:operation=replace

set equipment-state of OCH interface supporting circuit-pack to not-reserved-inuse

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit pack>"]/ports[port-name= "<xponder X network port>"]) nc:operation=replace

set admin state of OCH interface supporting port to inService

OCH interface shall be created on network port:

edit-config (org-openroadm-device/interface) nc:operation=create

create an OCH interface XPDR<n>-NETWORK<m>-[TXRX/RX]-[frequency] on supporting port

set name = "<XPDR<n>-NETWORK<m>-[TXRX/RX]-[frequency] >"

set type to opticalChannel

set supporting-circuit-pack-name="< xponder X network circuit pack >"

```

set supporting-port = "< xponder X network port >"
set rate, frequency, modulation-format (QPSK)
set admin state of OCH interface to inService"
OTU4 interface shall be created on supporting OCH interface:
edit-config (org-openroadm-device/interface) nc:operation=create
create an otnOtu interface OTU4 on supporting port/OCH interface
set name = "<OTU4-XPDR<n>-NETWORK<m>-TXRX/RX>"
set type to otnOtu
set supporting-circuit-pack-name="< xponder X network circuit pack >"
set supporting-port = "< xponder X network port >"
set supporting-interfaces = "< OCH interface>"
set admin state of OTU4 interface to inService

```

```

ODU4 interface shall be created on supporting OTU4 interface:
edit-config (org-openroadm-device/interface) nc:operation=create
create an otnOdu interface ODU4 on supporting port/OTU4 interface
set name = "<ODU4-XPDR<n>-NETWORK<m>-TXRX/RX>"
set type to otnOdu
set supporting-circuit-pack-name="< xponder X network circuit pack >"
set supporting-port = "< xponder X network port >"
set supporting-interfaces = "<OTU4-XPDR<n>-NETWORK<m>-TXRX/RX>"
set admin state of ODU4 interface to inService

```

The controller verifies that connection between client and network xponder circuit packs exists:

Optional: Get-config (org-openroadm-device/connection-map[source/circuit-pack-name="<xponder X network circuit pack >" and source/port-name="<xponder X network port>" and destination/circuit-pack-name="<xponder X client circuit pack > and destination/port-name="<xponder X client port>"])

Check if interface present on client port and the status of the port

Get-config (org-openroadm-device/interface[supporting-circuit-pack-name="< xponder X client circuit pack >" and supporting-port-name="< xponder X client port >"])

If NO corresponding interface created on the Xponder client port

(either bi-directional port or uni-directional TX port)

equipment-state of circuit-pack and admin-state of network-port shall be set appropriately :

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit pack>"]) nc:operation=replace

```

set equipment-state of 100GE interface supporting circuit-pack to not-reserved-inuse

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit pack>"]/ports[port-name= "<xponder X client port>"]) nc:operation=replace

```

set admin state of 100GE interface supporting port to inService

100GE Ethernet interface shall be created on client port:

```

edit-config (org-openroadm-device/interface) nc:operation=create

```

create a 100GE interface 100GE on supporting port

```

set name = "<100GE-XPDR<n>-CLIENT<m>-TXRX/TX>"

```

```

set type to ethernetCsmacd

```

```

set supporting-circuit-pack-name="< xponder X client circuit pack >"
set supporting-port ="< xponder X client port >"
set speed, fec, duplex, mtu, auto-negotiation, curr-speed
set admin state of 100GE interface to inService

```

End of sequence

8.1.6 Xponder Interfaces Deletion (TX/RX)

Deletes the interfaces and un-configures the ports on both line and client sides of xponder equipment in both RX and TX directions.

Input parameters:

- *node-id* (xponder X)
- xponder network *circuit-pack-name* (xponder X network circuit pack)
- xponder network *port-name* (xponder X network port)
- xponder client *circuit-pack-name* (xponder X client circuit pack)
- xponder client *port-name* (xponder X client port)

Main actions sequence:

```

edit-config (org-openroadm-device/interface[name= "<XPDR<n>-NETWORK<m>-[TXRX/TX]-[frequency]
>"]/och/transmit-power) nc:operation=replace

```

```

    set transmit power to minimum level

```

```

    set transmit-power to -5 dBm

```

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit
pack>"]) nc:operation=replace

```

```

    set equipment-state of OCH interface supporting circuit-pack to not-reserved-available

```

```

edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit
pack>"]/ports[port-name= "<xponder X network port>"]) nc:operation=replace

```

```

    set admin state of network port to outOfService

```

```

Get-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X network circuit
pack>"]/ports[port-name= "<xponder X network port>"]/interfaces)

```

Check interfaces configured on the line port(s) (either 1 bi-directional or 2 uni-directional ports)

If OCH interface is unique and bi-directional

```

    edit-config (org-openroadm-device/interface[name= "<ODU4-XPDR<n>-NETWORK<m>-TXRX
>"]) nc:operation=delete

```

```

        delete an ODU4 interface

```

```

    edit-config (org-openroadm-device/interface[name= "<OTU4-XPDR<n>-NETWORK<m>-TXRX"])
nc:operation=delete

```

```

        delete an OTU4 interface

```

```

    edit-config (org-openroadm-device/interface[name= "<XPDR<n>-NETWORK<m>-[TXRX]-
[frequency] >"]) nc:operation=delete

```

```

        delete an OCH interface XPDX-LINEN-TXRX-[frequency]

```

If 2 OCH uni-directional interfaces are present

```

edit-config (org-openroadm-device/interface[name= "<ODU4-XPDR<n>-NETWORK<m>-TX>"])
nc:operation=delete
    delete first ODU4 interface
edit-config (org-openroadm-device/interface[name= "<ODU4-XPDR<n>-NETWORK<m>-RX>"])
nc:operation=delete
    delete second ODU4 interface
edit-config (org-openroadm-device/interface[name= "<OTU4-XPDR<n>-NETWORK<m>-TX>"])
nc:operation=delete
    delete first OTU4 interface
edit-config (org-openroadm-device/interface[name= "<OTU4-XPDR<n>-NETWORK<m>-RX>"])
nc:operation=delete
    delete second OTU4 interface
edit-config (org-openroadm-device/interface[name= "<XPDR<n>-NETWORK<m>-[TX]-
[frequency] >"]) nc:operation=delete
    delete first OCH interface
edit-config (org-openroadm-device/interface[name= "<XPDR<n>-NETWORK<m>-[RX]-
[frequency] >"]) nc:operation=delete
    delete second OCH interface
edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit pack>"])
nc:operation=replace
    set equipment-state of 100GE interface supporting circuit-pack to not-reserved-available
Get-config (org-openroadm-device/interface[supporting-circuit-pack-name="<xponder X client circuit
pack >" and supporting-port-name="<xponder X client port >"])
    Check interfaces configured on the client port(s) (either 1 bi-directional or 2 uni-directional ports)
edit-config (org-openroadm-device/circuit-packs[circuit-pack-name= "<xponder X client circuit
pack>"]/ports[port-name= "<xponder X client port>"]) nc:operation=replace
    set admin state of 100GE interface supporting port to outOfService
If client interface is unique and bi-directional
edit-config (org-openroadm-device/interface[name= "<100GE-XPDR<n>-NETWORK<m>-
TXRX>"]) nc:operation=delete
If 2 uni-directional client interfaces are present
edit-config (org-openroadm-device/interface[name= "<100GE-XPDR<n>-NETWORK<m>-TX >"])
nc:operation=delete
edit-config (org-openroadm-device/interface[name= "<100GE-XPDR<n>-NETWORK<m>-RX >"])
nc:operation=delete
End of sequence

```

8.1.7 Add-Link Deletion

Configures ports, deletes connections and interfaces for an add link from a SRG-PP to a Degree TTP logical point.

This example considers the case of a ROADM connection between two nmc-ctp where on the degree side the nmc-ctp interface is the only child of a mc-ttp interface, thus, both nmc-ctp and mc-ttp are deleted.

Input parameters:

- *node-id* (SRGX, DEGY)
- *circuit-pack* (SRGX-CircuitPack, DEGY-CircuitPack)
- *supporting-port* (SRGX-PPN, DEGY-TTP)
- *minimum-output-power* (-60 dBm)
- *interface* (SRGX-PPN-[TXRX/RX]-[frequency])
- *interface* (nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency])
- *connection* SRGX-PPN-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]

Main actions sequence:

`edit-config (org-openroadm-device/roadm-connections/[connection-name= "<SRGX-PPN-[TXRX/RX]-[frequency]-to- nmc-ctpDEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=replace`

Set connection power from SRGX-PPN-[TXRX/RX] -[frequency] to nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency] to the minimum acceptable value:

set target-output-power = -60dBm

`edit-config(org-openroadm-device/circuit-packs/[circuit-pack-name= "<SRGX CircuitPack >"]/ports/[port-name= "<SRGX-PPN>"]) nc:operation=replace`

set admin state of nmc-ctp interface supporting PP port to outOfService

`edit-config (org-openroadm-device/roadm-connections/[connection-name= "<SRGX-PPN-[TXRX/RX]-[frequency]-to- nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"])nc:operation=delete`

delete connection SRGX-PPN-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]

`Get-config (org-openroadm-device/roadm-connections/)`

check if an nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to-SRGX-PPN-[TXRX/TX]-[frequency] connection is present

If (SRGX-PPN & nmc-ctp-DEGY-TTP are uni-directional OR

NO nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to-SRGX-PPN-[TXRX/TX]-[frequency] connection present)

`edit-config (org-openroadm-device/interface/[name= "<SRGX-PPN-[TXRX/RX]-[frequency] >"]) nc:operation=delete`

delete an nmc-ctp- interface on supporting SRGX PP port

`edit-config (org-openroadm-device/interface/[name= "<nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency] >"]) nc:operation=delete`

delete an nmc-ctp interface on supporting DEGY TTP port

`edit-config (org-openroadm-device/interface/[name= "<mc-ttp-DEGY-TTP-[TXRX/TX]-[frequency] >"]) nc:operation=delete`

delete an mc-ttp interface on supporting DEGY TTP port

End of sequence

8.1.8 Express Link Deletion

Configures the ports and deletes the interfaces and the connections for an express link between 2 Degree TTP logical points. Express link in this example is deleted in 1 step. Interfaces can be either bi (TXRX) or uni-directional (RX or TX) so that when we state [TXRX/TX], [TXRX/RX], or [TXRX/RX/TX], only one (TXRX) or the other option (TX or RX) is valid depending on the type of interface . If the express link

connections were deleted in 2 steps, then in the case of bi-directional nmc-ctp interfaces, we would delete these last-only in the second step.

Input parameters:

- *node-id* (DEGX, DEGY)
- *circuit-pack* (DEGX CircuitPack, DEGY CircuitPack)
- *supporting-port* (DEGX-TTP, DEGY-TTP)
- connection *nmc-ctp-DEGX-TTP-[TXRX/ RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/ TX]-[frequency]*

Main actions sequence:

```
edit-config (org-openroadm-device/roadm-connections/[connection-name= "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=merge
```

Set Connection nmc-ctp-DEGX-TTP-[TXRX/ RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/ TX]-[frequency] to low power

set target-output-power = -60dBm

```
edit-config (org-openroadm-device/roadm-connections/[connection-name= "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=delete
```

Delete connection nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGY-TTP-[TXRX/TX]-[frequency]

```
edit-config (org-openroadm-device/roadm-connections/[connection-name= "<nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to-nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=merge
```

Set Connection nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to- nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency] to low power

set target-output-power = -60dBm

```
edit-config (org-openroadm-device/roadm-connections/[connection-name= "<nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to- nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency]>"]) nc:operation=delete
```

Delete connection nmc-ctp-DEGY-TTP-[TXRX/RX]-[frequency]-to- nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency]

```
edit-config (org-openroadm-device/interface/[name= "<nmc-ctp-DEGX-TTP-[TXRX/TX/RX]-[frequency]>"]) nc:operation=delete
```

delete nmc-ctp interface(s) on supporting DEGX TTP port

```
edit-config (org-openroadm-device/interface/[name= "<nmc-ctp-DEGY-TTP-[TXRX/TX/RX]-[frequency]>"]) nc:operation=delete
```

delete nmc-ctp interface(s) on supporting DEGY TTP port

```
edit-config (org-openroadm-device/interface/[name= "<mc-ttp-DEGX-TTP-[TXRX/TX/RX]-[frequency]>"]) nc:operation=delete
```

delete mc-ttp interface(s) on supporting DEGX TTP port

```
edit-config (org-openroadm-device/interface/[name= "mc-ttp-DEGY-TTP-[TXRX/TX/RX]-[ frequency] >"]) nc:operation=delete
```

delete mc-ttp interface(s) on supporting DEGY TTP port

End of sequence

8.1.9 Drop Link Deletion

Configures ports, deletes connections and interfaces for drop link between a TTP and a PP logical points.

This example considers the case of a ROADM connection between two nmc-ctp where on the degree side the nmc-ctp interface is the only child of a mc-ttp interface, thus, both nmc-ctp and mc-ttp are deleted.

Input parameters:

- *node-id* (DEGX, SRGY)
- *circuit-pack* (DEGX CircuitPack, SRGYCircuitPack)
- *supporting-port* (DEGX-TTP, SRGY-PPM)
- connection *nmc-ctp-DEGX-TTP-[TXRX/ RX]-[frequency]-to-SRGY-PPN-[TXRX/ TX]-[frequency]*

Main actions sequence:

```
edit-config(org-openroadm-device/circuit-packs/[circuit-pack-name= "<SRGY CircuitPack >"]/ports  
/[port-name= "<SRGY-PPM>"]/) nc:operation=replace
```

set admin state of nmc-ctp interface supporting PP port to outOfService

```
edit-config (org-openroadm-device/roadm-connections/[connection-name= "< nmc-ctp-DEGX-TTP-  
[TXRX/RX]-[frequency]-to- SRGY-PPM-[TXRX/TX]-[frequency]>"]) nc:operation=delete
```

delete connection nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]-to-SRGY-PPN-[TXRX/TX]-[frequency]

```
Get-config (org-openroadm-device/roadm-connections/)
```

checks if an SRGY-PPN-[TXRX/RX]-[frequency]- to- nmc-ctp-DEGX-TTP-[TXRX/TX]-[frequency]
connection is present

**If (SRGY-PPN & DEGX-TTP are uni-directional OR NO SRGY-PPN-[TXRX/ RX]-[frequency]- to-DEGX-
TTP-[TXRX/ TX]-[frequency] connection present)**

```
edit-config (org-openroadm-device/interface/[name= "<nmc-ctp-DEGX-TTP-[TXRX/RX]-[frequency]  
>"]) nc:operation=delete
```

delete an nmc-ctp interface on supporting DEGX TTP port

```
edit-config (org-openroadm-device/interface/[name= "<mc-ttp-DEGX-TTP-[TXRX/RX]-[ frequency]  
>"]) nc:operation=delete
```

delete an mc-ttp interface on supporting DEGX TTP port

```
edit-config (org-openroadm-device/interface/[name= "<SRGY-PPN-[TXRX/TX]-[frequency] >"])  
nc:operation=delete
```

delete an nmc-ctp interface on supporting SRGY PP port

End of sequence

8.2 INCREMENTAL HARDWARE

8.2.1 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.

In order to deploy a new degree, the following provision would take place:

- Provision new shelves as necessary
- Provision new circuit packs that will support the new degrees, and configure the ports associated with the circuit packs
 - Including setting the TTP and CTP logical connection point attributes (against the port entities)
- 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 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

8.2.1.1 Create Entities

Creates new shelf entities associated with a degree augmentation, if required.

Input parameters:

- *node-id*
- *shelf-name*
- *shelf-type*
- *rack* location
- 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.

- 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:
 - *reserved-for-facility-planned*: equipment is planned for use by a service
 - *not-reserved-planned*: equipment is planned by not reserved for any purpose
 - *reserved-for-maintenance-planned*: equipment is planned for use as a maintenance spare
 - *reserved-for-facility-unvalidated*: equipment is reserved for use by a service but not validated against planned equipment
 - *not-reserved-unvalidated*: equipment is not reserved for any purpose and not validated against planned equipment
 - *unknown-unvalidated*: unknown equipment not validated against planned equipment
 - *reserved-for-maintenance-unvalidated*: equipment is to be used for use as a maintenance spare but not validated against planned equipment

- *reserved-for-facility-available*: reserved for use by a service and available
- *not-reserved-available*: not reserved for use by a service and available
- *reserved-for-maintenance-available*: reserved as a maintenance spare and available
- *reserved-for-reversion-inuse*: equipment that is reserved as part of a home path for a service that has been temporarily re-routed
- *not-reserved-inuse*: equipment in use for a service
- *reserved-for-maintenance-inuse*: maintenance spare equipment that is in use as a maintenance spare

Read-only data:

- Physical inventory data is provided for the shelf including the 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 provisioned that is plugged into the 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.

[edit-config \(/org-openroadm-device/shelves\) nc:operation=create](#)

Set the *shelf-type* and other shelf attributes.

Set the *shelf-name*

Set the *shelf-type* to the *vendor-specific* shelf type

Set the *administrative-state* of the shelf to *inService*

Set the rack, *shelf-position*, *equipment-state* and *due-date*, if desired.

These attributes are for tracking purposes only.

End of sequence

8.2.1.2 Create Circuit-Pack Entities

Creates new circuit-pack entities associated with a degree augmentation.

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.

- 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 can take on different “personalities”, then this is provisioned using the *circuit-pack-mode* attribute (e.g. REGEN mode vs TRANSPONDER mode).
- Circuit packs refer to their location in the shelf via the *shelf*, *slot* and optional *subSlot* attribute. The *subSlot* should be set to the same value as the *cp-slot-name* in the *parent-circuit-pack* container. In general, the hierarchical location of circuit packs should use the *shelf* and *slot* fields for circuit packs that plug directly into a shelf (in this case, there will not be a *parent-circuit-pack* container), or use the *cp-slot* and *parent-circuit-pack* 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. For example:

```
<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&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:

- Physical inventory data is provided for the circuit-pack including the 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.

[edit-config \(/org-openroadm-device/circuit-packs\) nc:operation=create](#)

Set the circuit-pack-type and other *circuit-pack attributes*.

Set the *circuit-pack-name*

Set the *circuit-pack-type* (mandatory) and *circuit-pack-product-code* (optional) to the vendor-specific values

Set the *circuit-pack-mode* as necessary based on vendor-specific values

Set the shelf, slot and subSlot to identify the location of the circuit-pack.

Also set the *parent-circuit-pack circuit-pack-name* and *cp-slot-name* if this circuit-pack plugs into another circuit pack.

Set the *administrative-state* of the shelf to *inService*

Set the *equipment-state* and *due-date* if desired.

These attributes are for tracking purposes only.

End of sequence

8.2.1.2.1 Configure Circuit-Pack Port Entities

This sub-step configures the port entities associated with the circuit pack. Ports are auto-created upon creation of the circuit-pack.

Note: Recommendation to use the “merge” operation on editing port attributes.

Input parameters:

- *node-id*
- *circuit-pack-name*
- *port-name*
- *port-type*

- *port-qual*
- *circuit-id*
- *administrative-state*
- *logical-connection-point*
- For OTDR ports, *launch-cable-length* and *port-direction*

Node-id, *circuit-pack-name* and *port-name* are mandatory. All other parameters are optional.

- Vendors define the *port-type* for the port. This value is not standardized in the Open ROADM MSA specifications. This field is optional.
- For ROADM degree, the *port-qual* can take the value of *roadm-external*, *roadm-internal*, *otdr* or the attribute is not present. The general guideline is that the ROADM network port would be set to *roadm-external* and any internal ports on the data path would be set to *roadm-internal*. The ports on an OTDR are set to *otdr*. For all other ports, the *port-qual* is not present or optionally set to *roadm-internal*.
- *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 if the port-direction is uni-directional receive, uni-directional transmit, or bi-directional, respectively, where <n> is the degree number. For example, 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 if the port-direction is uni-directional receive, uni-directional transmit, or bi-directional, respectively, where <n> is the degree number. For example for degree 1 with uni-directional ports that connect to degree 2, there should be two internal ports – one with logical connection point = "DEG1-CTP-RX" and the other with logical connection point = "DEG1-CTP-TX"
- 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* is set to multi-wavelength for the ROADM network degree 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.
- The *roadm-port capabilities* details the minimum and maximum aggregate powers supported by the port. This field applies to the ROADM network port for the degree.

Main actions sequence:

[edit-config \(/org-openroadm-device/circuit-packs\[circuit-pack-name\]/ports\) nc:operation=merge](#)

Edit the port configuration as necessary.

Set the *port-name*

Set the *port-type* to the *vendor-specific* port type

Set the *port-qualifier* and *logical-connection-point* as above. Note that some ports may not have these attributes set.

Set the *administrative-state* of the port to *inService*

For OTDR ports, set the *launch-cable-length* and *port-direction*.

End of sequence

8.2.1.3 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* associated with the degree (external ports)
- For 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. All other parameters are optional.

- The *circuit-packs* container lists all circuit-packs associated with the degree.
 - Note that if there are shared circuit-packs that is used for more than one degree (or SRG), then that circuit-pack should also be listed under the new degree as well as the old degree (or SRG).
- The *connection-ports* container lists the ports that connect externally to the line transmission fibers.
 - The connection ports should 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 transmit port. The *circuit-pack-name* and *port-name* refer to the port on an OTDR.

Read-only data:

- The *max-wavelength* identifies the maximum number of wavelengths supported by the degree.

Main actions sequence:

`edit-config (/org-openroadm-device/degree) nc:operation=create`

Set the degree attributes.

Set the *degree-number*

Set the list of circuit-packs associated to the degree

Set the list of connection ports associated to the degree

Configure the *otdr-port* that would monitor the degree

End of sequence

8.2.1.4 Create Physical Link Entities

Creates the physical link entities that describe the internal ROADM fiber and/or cabling within the degree, between the degree and common units, between the degree and other existing degrees, and between the degree and SRGs.

Input parameters:

- *node-id*
- *physical-link-name*
- source and destination port identifiers

node-id, *physical-link-name*, source *circuit-pack-name*, source *port-name*, destination *circuit-pack-name* and destination *port-name* are mandatory.

- Physical links are always uni-directional
 - Source should indicate the TX port
 - Destination should indicate the RX port
- Physical links can represent a single bi-directional cable (e.g., Ethernet cable). In this case, the physical link is still modeled as two uni-directional entities.
- Physical links can also represent MPO connections. Both the physical connector and the logical connectors (child ports within the MPO) can be modeled

Main actions sequence:

`edit-config (/org-openroadm-device/ physical-link) nc:operation=create`

Set the physical link end points.

Set the *physical-link-name*

Set the physical link source *circuit-pack-name* and *port-name* (TX)

Set the physical link destination *circuit-pack-name* and *port-name* (RX)

End of sequence

This completes the commissioning of a new degree.

8.2.2 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 are independent of the number of degrees currently deployed.

The deployment of SRGs is similar to the deployment of degrees. In order to deploy a new SRG, the following provision would take place:

- Provision new shelves as necessary
- Provision new circuit packs that will support the new SRGs, and configure the ports associated with the circuit packs
 - Including 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.
- Provision the new physical links associated with the SRG. This may include:
 - Physical links internal to the SRG
 - Physical links connecting the new SRG to the existing degrees
 - Physical links connecting the new SRG to the existing SRGs

8.2.2.1 Create Shelf Entities

Creates new shelf entities associated with a SRG augmentation, if required.

Input parameters:

- *node-id*
- *shelf-name*
- *shelf-type*
- *rack* location
- *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.

Main actions sequence:

`edit-config (/org-openroadm-device/shelves) nc:operation=create`

Set the *shelf-type* and other shelf attributes.

Set the *shelf-name*

Set the *shelf-type* to the *vendor-specific* shelf type

Set the *administrative-state* of the shelf to *inService*

Set the *rack*, *shelf-position*, *equipment-state* and *due-date*, if desired.

These attributes are for tracking purposes only.

End of sequence

8.2.2.2 Create Circuit-Pack Entities

Creates new circuit-pack entities associated with a SRG augmentation.

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*, *slot* are mandatory. All other parameters are optional.

Main actions sequence:

[edit-config \(/org-openroadm-device/circuit-packs\) nc:operation=create](#)

Set the *circuit-pack-type* and other circuit-pack attributes.

Set the *circuit-pack-name*

Set the *circuit-pack-type* (mandatory) and *circuit-pack-product-code* (optional) to the vendor-specific values

Set the *circuit-pack-mode* as necessary based on vendor-specific values

Set the *shelf*, *slot* and *subSlot* to identify the location of the circuit-pack.

Also set the *parent-circuit-pack circuit-pack-name* and *cp-slot-name* if this circuit-pack plugs into another circuit pack.

Set the *administrative-state* of the shelf to *inService*

Set the *equipment-state* and *due-date* if desired.

These attributes are for tracking purposes only.

End of sequence

8.2.2.2.1 Configure Circuit-Pack Port Entities

This sub-step configures the port entities associated with the circuit pack. Ports are auto-created upon creation of the circuit-pack.

Note: Recommendation to use the “merge” operation on editing port attributes.

Input parameters:

- *node-id*
- *circuit-pack-name*
- *port-name*

- *port-type*
- *port-qual*
- *circuit-id*
- *administrative-state*
- *logical-connection-point*

node-id, *circuit-pack-name* and *port-name* are mandatory. All other parameters are optional.

Differences specific to SRGs are described below.

- For ROADM SRGs, the *port-qual* can take the value of *roadm-external*, *roadm-internal*, or the attribute is not present. The general guideline is that the ROADM SRG add/drop port would be set to *roadm-external* and any internal SRG ports on the data path would be set to *roadm-internal*. For all other ports, the *port-qual* is not present or optionally set to *roadm-internal*.
- The logical connection point is used to support the mapping at the controller level between the device model and network model.
 - For the ROADM SRG add/drop port, the logical connection point should be set to the format “SRG<n>-PP<m>”, where <n> is the SRG number and <m> is the add/drop port pair identifier. For example for SRG 1 add/drop port #7 would have the logical connection point set to SRG1-PP7.
 - For the ROADM SRG internal ports that connect to another degree or SRG, 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 if the *port-direction* is uni-directional receive, uni-directional transmit, or bi-directional, respectively, where <n> is the SRG number. For example for SRG 1 with uni-directional ports that connect to degree 1, there should be two internal ports – one with logical connection point = “SRG1-CP-RX” and the other with logical connection point = “SRG1-CP-TX”

Read-only data:

- *port-wavelength-type* is set to *wavelength* for the ROADM SRG add/drop port. This attribute may not be present for other ports.
- *port-capabilities* is used to indicate the capabilities of a *port*, including which interface(s) can be built on the port. The ROADM SRG add/drop port, the list of interface capabilities include: *if-NMC*.
- The *roadm-port* capabilities details the minimum and maximum aggregate powers supported by the port. This field applies to the ROADM SRG add/drop ports.

Main actions sequence:

[edit-config \(/org-openroadm-device/circuit-packs\[circuit-pack-name\]/ports\) nc:operation=merge](#)

Edit the port configuration as necessary.

Set the *port-name*

Set the *port-type* to the *vendor-specific* port type

Set the *port-qualifier* and *logical-connection-point* as above. Note that some ports may not have these attributes set.

Set the *administrative-state* of the port to *inService*

End of sequence

8.2.2.3 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

node-id, *srg-number*, and *circuit-pack-list* are mandatory.

- The *circuit-packs* container lists all circuit-packs associated with the SRG.
 - Note that if there are shared circuit-packs that are used for more than one SRG (or degree), then that circuit-pack should also be listed under the new SRG as well as the old SRG (or degree).

Read-only data:

- The *max-add-drop-ports* identifies the maximum number of SRG add/drop ports supported by the SRG, regardless if all 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.

Main actions sequence:

`edit-config (/org-openroadm-device/shared-risk-group) nc:operation=create`

Set the SRG attributes.

Set the *srg-number*

Set the list of *circuit-packs* associated to the SRG

End of sequence

8.2.2.4 Create Physical Link Entities

Creates the physical link entities that describe the internal ROADM fiber and/or cabling within the SRG, between the SRG and common units, between the SRG and other degrees, and between the SRG and other existing SRGs.

Input parameters:

- *node-id*
- *physical-link-name*
- source and destination port identifiers

node-id, *physical-link-name*, source *circuit-pack-name*, source *port-name*, destination *circuit-pack-name* and destination *port-name* are mandatory.

- Physical links are always uni-directional
 - Source should indicate the TX port
 - Destination should indicate the RX port
- Physical links can represent a single bi-directional cable (e.g., Ethernet cable). In this case, the physical link is still modeled as two uni-directional entities.

- Physical links can also represent MPO connections. Both the physical connector and the logical connectors (child ports within the MPO) can be modeled

Main actions sequence:

`edit-config (/org-openroadm-device/ physical-link) nc:operation=create`

Set the physical link end points.

Set the *physical-link-name*

Set the physical link source *circuit-pack-name* and *port-name* (TX)

Set the physical link destination *circuit-pack-name* and *port-name* (RX)

End of sequence

This completes the commissioning of a new SRG.

8.2.3 Xponder Growth

Xponders have network port(s) that connects to the ROADMs's SRG add/drop ports and client port(s) that connect to Routers or other client equipment. The deployment of xponders is similar to the deployment of degrees and SRGs. In order to deploy new xponder equipment, the following provision would take place:

- Provision new shelves as necessary
- Provision new circuit packs that will support the new xponder, and configure the ports associated with the circuit packs
 - Including setting the logical connection point attributes (against the port entities)
- Provision the xponder container for the new transponder, muxponder, 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*.
- Provision the new physical links associated with the xponder as applicable. The physical links would be needed if the xponder had multiple circuit packs with physical cabling between the circuit-packs.

8.2.3.1 Create Shelf Entities

Creates new shelf entities associated with a xponder augmentation, if required.

Input parameters:

- node-id*
- shelf-name*
- shelf-type*
- rack-location*
- 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 \(/org-openroadm-device/shelves\) nc:operation=create](#)

Set the *shelf-type* and other shelf attributes.

Set the *shelf-name*

Set the *shelf-type* to the *vendor-specific* shelf type

Set the *administrative-state* of the shelf to *inService*

Set the *rack*, *shelf-position*, *equipment-state* and *due-date*, if desired.

These attributes are for tracking purposes only.

End of sequence

8.2.3.2 Create Circuit-Pack Entities

Creates new circuit-pack entities associated with a xponder augmentation.

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*, *slot* are mandatory. All other parameters are optional.

Main actions sequence:

[edit-config \(/org-openroadm-device/circuit-packs\) nc:operation=create](#)

Set the *circuit-pack-type* and other circuit-pack attributes.

Set the *circuit-pack-name*

Set the *circuit-pack-type* (mandatory) and *circuit-pack-product-code* (optional) to the *vendor-specific* values

Set the *circuit-pack-mode* as necessary based on *vendor-specific* values

Set the *shelf*, *slot* and *subSlot* to identify the location of the circuit-pack.

Also set the *parent-circuit-pack circuit-pack-name* and *cp-slot-name* if this circuit-pack plugs into another circuit pack.

Set the *administrative-state* of the shelf to *inService*

Set the *equipment-state* and *due-date* if desired. These attributes are for tracking purposes only.

End of sequence

8.2.3.2.1 Configure Circuit-Pack Port Entities

This sub-step configures the port entities associated with the circuit pack. Ports are auto-created upon creation of the circuit-pack.

Note: Recommendation to use the “merge” operation on editing port attributes.

Input parameters:

- *node-id*
- *circuit-pack-name*
- *port-name*
- *port-type*
- *port-qual*
- *circuit-id*
- *administrative-state*
- *logical-connection-point*

node-id, *circuit-pack-name* and *port-name* are mandatory. All other parameters are optional.

- For xponders, the *port-qual* can take the value of *xpdr-network*, *xpdr-client*, or the attribute is not present. The general guideline is that the *xpdr network* ports would be set to *xpdr-network* and *xpdr client* ports would be set to *xpdr-client*. For all other ports, the *port-qual* is not present.
- The *logical-connection-point* is used to support the mapping at the controller level between the device model and network model.
 - For the *xpdr network* port, the logical connection point should be set to the format *XPDR<n>-NETWORK<m>*.
 - For the *xpdr client* ports, the logical connection point should be set to the format *XPDR<n>-CLIENT<m>*.
 - Where *<n>* is the logical transponder/muxponder/xponder number, and *<m>* are the network and client ports that make up the transponder/muxponder/xponder. Typically, the network *<m>* will be 1 for transponders and muxponders. For transponders, the client *<m>* will be 1 and for muxponders the client *<m>* will be 1 .. m where m = max number of client muxponder ports.

Read-only data:

- *port-wavelength-type* is set to wavelength for the *xpdr network* and client ports. This attribute may not be present for other ports.
- *port-capabilities* is used to indicate the capabilities of a port, including which interface(s) can be built on the port.
 - The *xpdr network* port, the list of interface capabilities include: *if-OCH-OTU4-ODU4* or *if-otsi-otsigroup*
 - The *xpdr 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.

Main actions sequence:

[edit-config \(/org-openroadm-device/circuit-packs\[circuit-pack-name\]/ports\) nc:operation=merge](#)

Edit the port configuration as necessary.

Set the *port-name*

Set the *port-type* to the *vendor-specific* port type

Set the *port-qualifier* and *logical-connection-point* as above. Note that some ports may not have these attributes set.

Set the *administrative-state* of the port to *inService*

End of sequence

8.2.3.3 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, and xpdr-port list are mandatory.

- The *xpdr-type* can be *tpdr* (transponder), *mpdr* (muxponder), *switch* (OTN switch or switchponder), *regen* (regenerator) or *regen-uni* (uni-directional regenerator)
- The *xpdr-port* container lists all ports associated with the xponder.
- *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 recoloring. This only applies if *xpdr-type* is *regen* or *regen-uni*.

Main actions sequence:

[edit-config \(/org-openroadm-device/xponder\) nc:operation=create](#)

Set the xponder attributes.

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

End of sequence

8.2.3.4 Create Physical Link Entities

Creates the physical link entities that describe the internal xponder fibering and/or cabling, if applicable.

Input parameters:

- *node-id*
- *physical-link-name*
- source and destination *port* identifiers

node-id, *physical-link-name*, source *circuit-pack-name*, source *port-name*, destination *circuit-pack-name* and destination *port-name* are mandatory.

- Physical links are always uni-directional
 - Source should indicate the TX port
 - Destination should indicate the RX port
- Physical links can represent a single bi-directional cable (e.g., Ethernet cable). In this case, the physical link is still modeled as two uni-directional entities.
- Physical links can also represent MPO connections. Both the physical connector and the logical connectors (child ports within the MPO) can be modeled

Main actions sequence:

```
edit-config (/org-openroadm-device/ physical-link) nc:operation=create
```

Set the physical link end points.

Set the *physical-link-name*

Set the physical link source *circuit-pack-name* and *port-name* (TX)

Set the physical link destination *circuit-pack-name* and *port-name* (RX)

End of sequence

This completes the commissioning of a new xponder unit(s).

9 MAINTENANCE ACTION USE CASES

9.1 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:

```
+---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?

```

9.1.1 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 would be as follows:

```

<nc:prc nc:message-id="167.254.172.235-1409" xmlns:nc="urn:ietf:params: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>

```

For backward compatibility, the device should also support the following example format:

```

<nc:prc nc:message-id="167.254.172.235-1409" xmlns:nc="urn:ietf:params: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>

```

9.1.2 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* would be as follows:

```

<nc:prc nc:message-id="167.254.172.235-1409" xmlns:nc="urn:ietf:params: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>

```

For backward compatibility, the device should also support the following example format:

```

<nc:prc nc:message-id="167.254.172.235-1409" xmlns:nc="urn:ietf:params: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>

```

9.2 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 ROADMSA 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.

When a loopback is in effect, the device should raise a notification based on the type of loopback (*facilityLoopbackActive*, *facilityLoopback2Active*, or *terminalLoopbackActive*).

9.2.1 Facility Loopbacks

A facility loopback (local loopback) is used to test external connections (fiber) to the interface (refer to Figure 63: Facility Loopback Operations - Client-side Interface). 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.

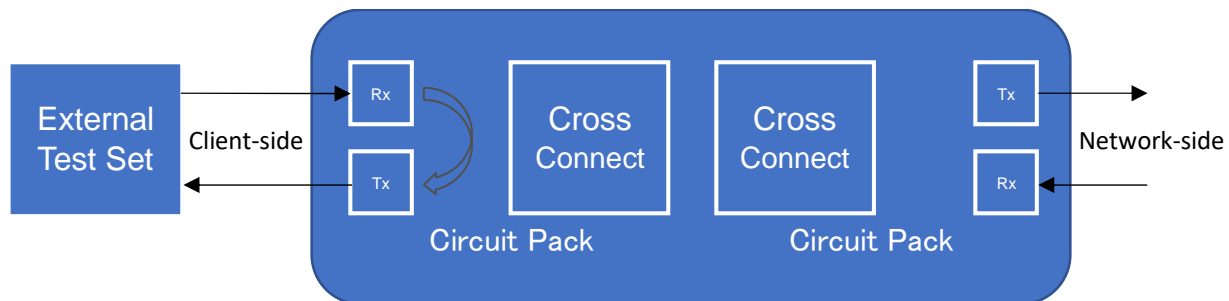


Figure 63: Facility Loopback Operations - Client-side Interface

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 64: Facility Loopback Operations - Network side itnerface*.

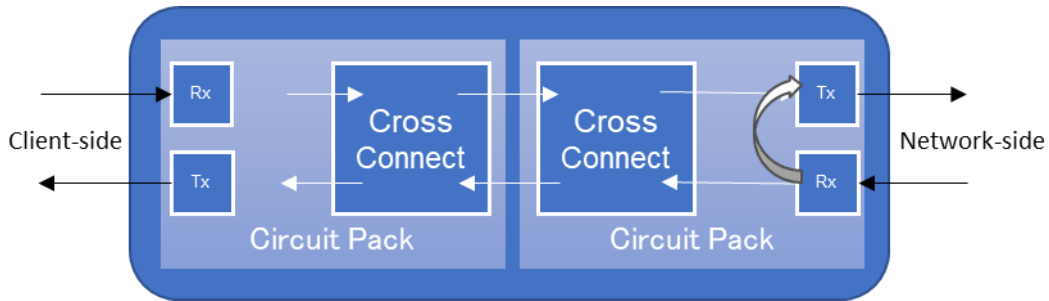


Figure 64: Facility Loopback Operations - Network side itnerface

9.2.2 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 65: Terminal Loopback Operation - Client-side Interface). 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.

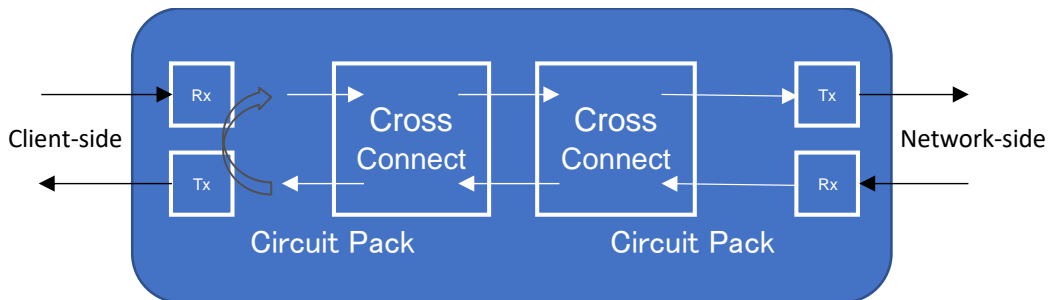


Figure 65: Terminal Loopback Operation - Client-side Interface

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 66: Terminal Loopback Operations - Network side Interface*.

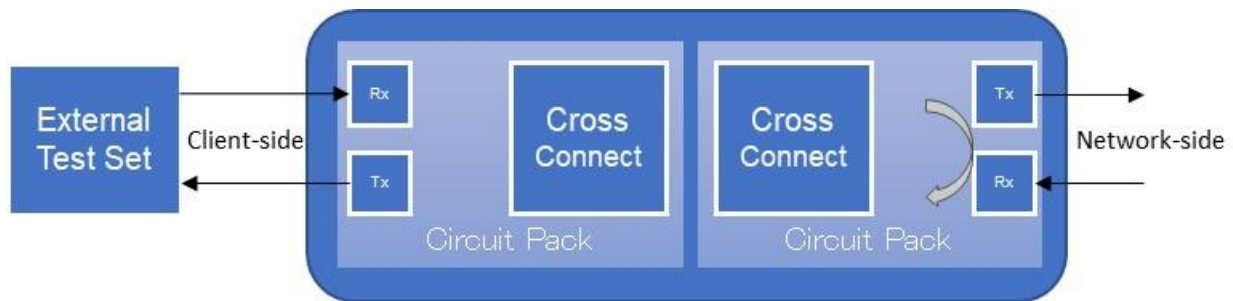


Figure 66: Terminal Loopback Operations - Network side Interface

9.2.3 Loopback for B100G Interfaces

For B100G interfaces, facility and terminal loopbacks are supported at the OTUCn level.

Figure 67: B100G Loopbacks 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.

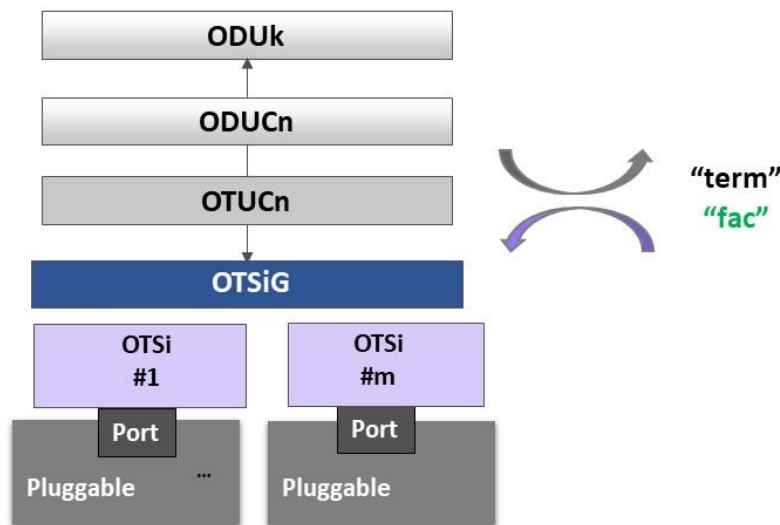


Figure 67: B100G Loopbacks

9.3 BIT-ERROR RATE (BER) TEST APPROACH

9.3.1 WDM Layer BER Test

For the WDM layer, the BER test is 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.

9.3.2 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*. If the client-side ODU does indicate *no-oam-function*, then the controller will initiate the test on the network-side ODU. Note: implementations should not indicate *no-oam-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.

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 [D]). For client OTU interfaces, the test signal will be set to the terminal direction (*type=term*) if the test signal is initiated on the client-side ODU, or the test signal will be set to the facility direction (*type=fac*) if the test signal is initiated on the network-side ODU. The controller will not send the *type* attribute for 100GE clients, regardless if the test signal is initiated on the client-side or network-side ODU interface.

The key statistics are *isSync* 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 *isSync*, *seconds* and *bitErrors* statistics. Support for the *bitErrorRate* is optional.

When a test signal is in effect, the device should raise a notification based on the type of test signal (*facilityTestsignalActive* or *terminalTestsignalActive*).

9.4 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.

9.5 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 68: ROADM OTDR Reference Model) and the MWi input ports of an ILA (refer to Figure 69: ILA OTDR Reference Model). 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).

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 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.

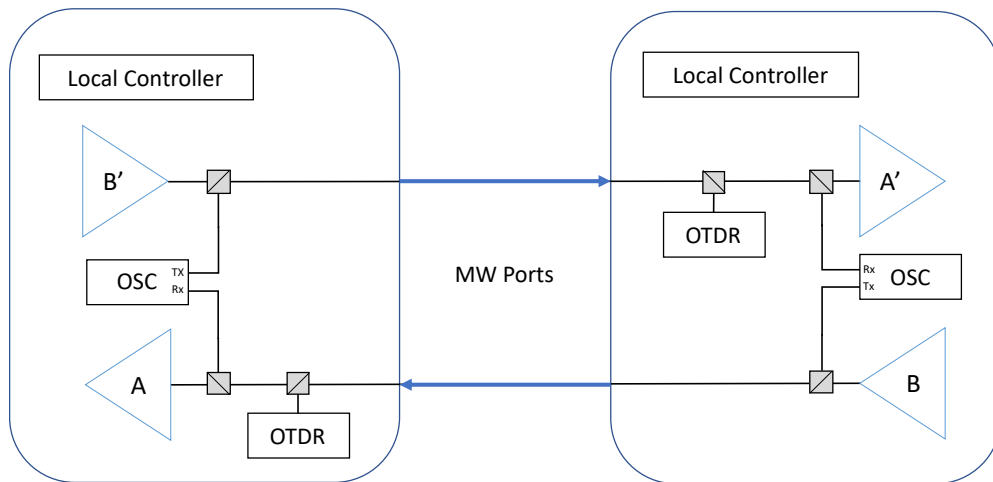


Figure 68: ROADM OTDR Reference Model

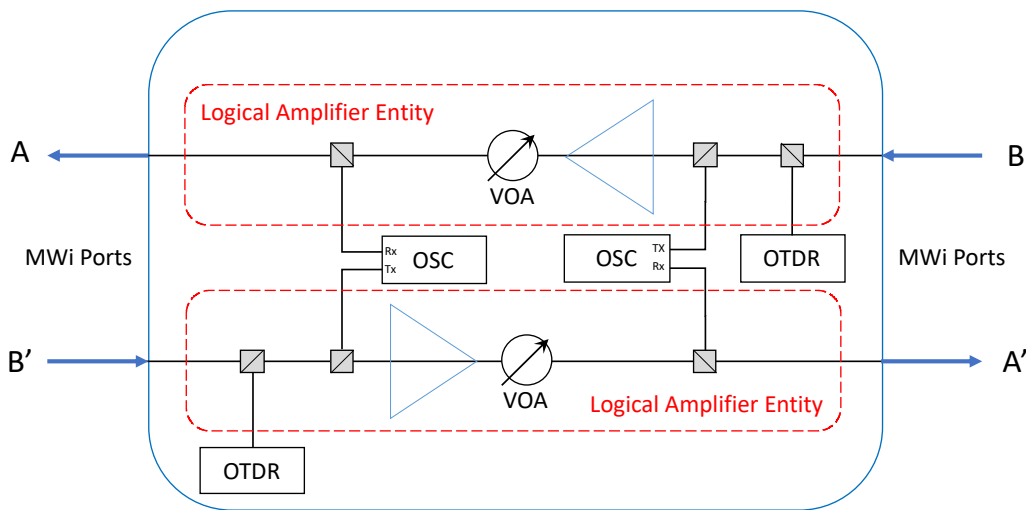


Figure 69: ILA OTDR Reference Model

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. 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 kilometers and the *resolution* in meters (how close events can be spaced and still be recognized as different events). **Note: The Open ROADM MSA Device model will be updated to reflect the unit type for the *distance* (kilometers) and *resolution* (meters) attributes.**

```
+---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?
```

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 [HH] 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 4.1 for details on file transfers).

OTDR specification details can be found in the OTDR tab of the Open ROADM MSA Specification.

9.6 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?

```

A *create-tech-info-notification* should be generated when the file is ready for retrieval (refer to Section 4.1 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?

```


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.

9.7 RETRIEVE ROADM CONNECTION PORT TRAIL

The retrieve ROADM connection port trail 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*.

```
+---x get-connection-port-trail
| +---w input
| | +---w connection-name
| +--ro output
|   +--ro status
|   +--ro status-message?
|   +--ro ports* []
|     +--ro rack?
|     +--ro shelf?
|     +--ro slot?
|     +--ro subSlot?
|     +--ro circuit-pack-name
|     +--ro port-name
```

9.8 MISSING TOPICS (TO BE ADDRESSED IN A FUTURE VERSION OF THE WHITEPAPER)

- Restore remote communications to the device from a corrupted or inconsistent configuration
 - describe the mechanism to restore communications to the device from corrupted or inconsistent configuration, assuming power cycle of the processor or management blade is ineffective in this case.
- Remote Transponders
- Optical control loops
 - Span loss changes on existing systems

10 REFERENCES

- A. IEEE 802.3-2018 *IEEE Standard for Ethernet*
- B. IETF draft-ietf-secsh-filexfer-13 *SSH File Transfer Protocol*
- C. IETF draft-ietf-netmod-syslog-model-05 *Syslog YANG Model*
- D. ITU-T G.709/Y.1331 (06/2016) *Interfaces for the optical transport network*
- E. ITU-T G.709/Y.1331 Amd 1 (06/2016) *Interfaces for the optical transport network Amendment 1*
- F. ITU-T G.709/Y.1331 Amd 2 (06/2018) *Interfaces for the optical transport network Amendment 2*
- G. ITU-T G.709/Y.1331 Amd 3 (04/2009) *Interfaces for the optical transport network Amendment 3*
- H. ITU-T G.709.1/Y.1331.1 (06/2018) *Flexible OTN short-reach interfaces*
- I. ITU-T G.709.1/Y.1331.1 Amd 1 (04/2019) *Flexible OTN short-reach interfaces Amendment 1*
- J. ITU-T G.709.1/Y.1331.1 Cor 1 (05/2020) *Flexible OTN short-reach interfaces Corrigendum 1*
- K. ITU-T G.709.3/Y.1331.3 (09/2020) *Flexible OTN long-reach interfaces*
- L. ITU-T G.709.3/Y.1331.3 Amd 1 (11/2018) *Flexible OTN long-reach interfaces Amendment 1*
- M. ITU-T G.709.3/Y.1331.3 Err 1 (02/2020) *Flexible OTN long-reach interfaces Erratum 1*
- N. ITU-T 808.1 (05/2014) *Generic protection switching – Linear trail and subnetwork protection*
- O. ITU-T G.842 (04/2007) *Interworking of SDH network protection architectures*
- P. ITU-T G.873.1 (10/2017) *Optical transport network: Linear protection*
- Q. ITU-T G.7041/Y.1303 (08/2016) *Generic framing procedure*
- R. OIF OIF-FLEXE-02.0 (06/2018) *Flex Ethernet 2.0 Implementation Agreement*
- S. RFC 4251 *The Secure Shell (SSH) Protocol Architecture*
- T. RFC 4252 *The Secure Shell (SSH) Authentication Protocol*
- U. RFC 4253 *The Secure Shell (SSH) Transport Layer Protocol*
- V. RFC 4254 *The Secure Shell (SSH) Connection Protocol*
- W. RFC 4716 *The Secure Shell (SSH) Public Key File Format*
- X. RFC 4742 *Using NETCONF Configuration Protocol over Secure Shell (SSH)*
- Y. RFC 5246 *The Transport Layer Security (TLS) Protocol Version 1.2*
- Z. RFC 5277 *NETCONF Event Notifications*
- AA. RFC 5424 *The Syslog Protocol*
- BB. RFC 6020 *YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)*
- CC. RFC 6022 *YANG Module for NETCONF Monitoring*
- DD. RFC 6241 *Network Configuration Protocol (NETCONF)*
- EE. RFC 6242 *Using the NETCONF Protocol over Secure Shell (SSH)*
- FF. RFC 7159 *The JavaScript Object Notation (JSON) Data Interchange Format*
- GG. RFC 7950 *The YANG 1.1 Data Modeling Language*
- HH. Telcordia SR-4731 Issue 2 *Optical Time Domain Reflector (OTDR) Data Format*

Appendix A: MANIFEST FILE FOR SOFTWARE DOWNLOAD AND DATABASE OPERATIONS

This section describes the manifest file for software download, database backup, and database restore operations. 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.

The manifest file allows vendors to specify vendor-specific behavior in a common format for software download and database operations. 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 necessarily 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-sets. Each instruction-set would apply to one or more from-sw-versions 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
- 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 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 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 reboot and revert back to the original software load.

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)
- 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 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 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 reboot and revert back to the database in the state before the database restore.

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 MANIFEST FILE YANG MODEL

The Manifest file YANG model is included as part of the Open ROADM MSA v7 YANG repository (under the *common* directory).

A.4.1 Manifest File Tree View

A.4.1.1 Software Manifest Tree View

The *sw-manifest* model consists of the following:

```
+--rw sw-manifest!  
| +--rw vendor  
| +--rw model  
| +--rw sw-version  
| +--rw global-async-timeout?  
| +--rw global-sync-timeout?  
| +--rw instruction-set* [index]  
|   +--rw index  
|   +--rw from-sw-version*  
|   +--rw is-commit-sw-activate-async?  
|   +--rw cancel-validation-timer-async-timeout?  
|   +--rw cancel-validation-timer-sync-timeout?  
|   +--rw sw-manifest-commands  
|     +--rw sw-manifest-command* [command-order]  
|       +--rw command-order  
|       +--rw command  
|       +--rw download-file  
|         +--rw remote-filename  
|         +--rw local-file-path  
|         +--rw timeout?  
|         +--rw is-async?  
|       +--rw delete-file  
|       +--rw filename
```

```

|      | +--rw timeout? 6
|      | +--rw is-async?
|      +--rw sw-stage
|      | +--rw filename?
|      | +--rw timeout?
|      | +--rw is-async?
|      +--rw sw-activate
|      | +--rw version
|      | +--rw validation-timer?
|      | +--rw timeout?
|      | +--rw auto-reboot
|      | +--rw is-async?
|      +--rw cancel-validation-timer
|      +--rw wait-time
|      +--rw timeout?
|      +--rw is-async?

```

A.4.1.2 Database Backup Manifest Tree View:

The *db-backup-manifest* model consists of the following:

```

+--rw db-backup-manifest!
| +--rw vendor
| +--rw model
| +--rw sw-version?
| +--rw global-async-timeout?
| +--rw global-sync-timeout?
| +--rw db-backup-manifest-commands
|   +--rw db-backup-manifest-command* [command-order]
|     +--rw command-order
|     +--rw command
|     +--rw upload-file
|       +--rw remote-filename
|       +--rw local-file-path
|       +--rw timeout?
|       +--rw is-async?
|     +--rw delete-file
|       +--rw filename
|       +--rw timeout?
|       +--rw is-async?
|   +--rw db-backup
|     +--rw filename?
|     +--rw timeout?
|     +--rw is-async?

```

A.4.1.3 Database Restore Manifest Tree View:

The *db-restore-manifest* model consists of the following:

```

+--rw db-restore-manifest!
  +--rw vendor

```

```

+--rw model
+--rw sw-version?
+--rw global-async-timeout?
+--rw global-sync-timeout?
+--rw is-commit-db-activate-async?
+--rw cancel-rollback-timer-async-timeout?
+--rw cancel-rollback-timer-sync-timeout?
+--rw database-init-sync-timeout?
+--rw db-restore-manifest-commands
  +--rw db-restore-manifest-command* [command-order]
    +--rw command-order
    +--rw command
    +--rw download-file
      | +--rw remote-filename
      | +--rw local-file-path
      | +--rw timeout?
      | +--rw is-async?
    +--rw delete-file
      | +--rw filename
      | +--rw timeout?
      | +--rw is-async?
    +--rw db-restore
      | +--rw filename?
      | +--rw node-id-check?
      | +--rw timeout?
      | +--rw is-async?
    +--rw db-activate
      | +--rw rollback-timer?
      | +--rw timeout?
      | +--rw auto-reboot
      | +--rw is-async?
    +--rw cancel-rollback-timer
      +--rw wait-time
      +--rw timeout?
      +--rw is-async?

```

A.4.2 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.4.2.1 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.

- *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*.
- *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.

- *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.4.2.2 Common Manifest Commands

The following sub-sections identify manifest commands common to software and database operations.

A.4.2.2.1 Download File Command

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 4.2.44.2.4 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 construction 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.4.2.2.2 Delete File Command

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 4.2.6 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.4.2.3 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.

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.4.2.3.1 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.4.2.2.1 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.4.2.2.2 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.4.2.3.2 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 only 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 (length of time from device comm loss until the device is ready to accept a login). The timer specified by the *timeout* duration begins as soon as the *sw-activate* processing begins (comm loss detected) and must be greater than the *auto-reboot* time. If the controller cannot login after the timer expires, the *sw-activate* operation is considered failed.

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). 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).

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.4.2.3.3 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.4.2.4 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.4.2.4.1 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.4.2.2.2 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.4.2.4.2 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 4.2.3 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 construction 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.4.2.5 Database Restore Manifest File (**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.

A.4.2.5.1 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.4.2.2.1 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.4.2.5.2 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 (length of time from device comm loss until the device is ready to accept a login). A value of “0” should be specified if the device implementation does not require a reboot of the device to activate the newly restored database. The timer specified by the *timeout* duration begins as soon as the *db-activate* processing begins (e.g. comm loss detected) and must be greater than the *auto-reboot* time. If the controller fails to login after the timer expires, the *db-activate* operation is considered failed.

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).

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 software load (see Section A.4.2.5.3 for *cancel-rollback-timer* command details).

A.4.2.5.3 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.5 MANIFEST FILE EXAMPLES

A.5.1 Software Manifest File Example

Example software manifest file name, [sw-manifest.json](#), and content:

```
{
  "vendor": "VENDOR",
  "model": "OWB-ROADM",
  "sw-version": "3.0",
  "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": {
                "validation-timer": "02-00-00",
                "auto-reboot": 600,
                "version": "3.0",
                "timeout": 2400,
                "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": {
                "validation-timer": "02-00-00",
                "auto-reboot": 600,
                "version": "3.0",
                "timeout": 2400,
                "is-async": "true"
            }
        }
    ]
}

```

A.5.2 Database Backup Manifest File Example

Example database backup manifest file name, [db-backup-manifest.json](#), and content:

```

{
    "vendor": "VENDOR",
    "model": "OWB-ROADM",
    "sw-version": "3.0",
    "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",
                "timeout": 3000,
                "local-file-path": "__LOCAL-FILE-PATH",
                "is-async": "true"
            }
        },
        {
            "command-order": 3,
            "command": "org-openroadm-manifest-file:delete-file",

```



```

        "delete-file": {
            "filename": "__LOCAL-FILE-PATH",
            "is-async": "false"
        }
    }
}

```

A.5.3 Database Restore Manifest File Example

Example database restore manifest file name, `db-restore-manifest.json`, and content:

```

{
    "vendor": "VENDOR",
    "model": "OWB-ROADM",
    "sw-version": "3.0",
    "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": {
                    "node-id-check": "__NODE-ID-CHECK",
                    "filename": "__LOCAL-FILE-PATH",

```

```
    "timeout": 240,  
    "is-async": "false"  
  },  
  {  
    "command-order": 4,  
    "command": "org-openroadm-manifest-file:db-activate",  
    "db-activate": {  
      "rollback-timer": "01-00-00",  
      "auto-reboot": 600,  
      "timeout": 650  
    }  
  }  
]  
}  
}
```

Appendix B: PERFORMANCE MONITORING (PM) AND ALARM TABLES

Table 20: Performance Monitoring

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
MW Ports							
Optical Power Output (OPOUT- OTS)	MW	OTS-IF	x		opticalPowerOutput		total optical signal power with OSC, includes VOA attenuation
				min	opticalPowerOutputMin		
				max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		
Optical Power Input (OPIN-OTS)	MW	OTS-IF	x		opticalPowerInput		total optical signal power with OSC
				min	opticalPowerInputMin		
				max	opticalPowerInputMax		
				avg	opticalPowerInputAvg		
Optical Power Output (OPOUT- OMS)	MW	OMS-IF			opticalPowerOutput		total optical signal power without OSC, includes VOA attenuation
				min	opticalPowerOutputMin		
				max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		
	MW	OMS-IF			opticalPowerInput		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
Optical Power Input (OPIN- OMS)				min	opticalPowerInputMin		total optical signal power without OSC
				max	opticalPowerInputMax		
				avg	opticalPowerInputAvg		
Optical Return Loss (ORL-OTS)	MW	OTS-IF on tx port	x		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	x		opticalReturnLoss		at MW port(s) B (not including OSC reflection) No tide markings required (e.g. min, max, avg) Either ORL-OTS or

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
							ORL-OMS is supported depending upon vendor implementation. direction, if included is rx (not tx)
OSC Optical Power Transmit (OPT-OSC)	MW	OTS-IF			opticalPowerOutputOSC		OSC Transmit power on MW port
				min	opticalPowerOutputOSCMin		
				max	opticalPowerOutputOSCMax		
				avg	opticalPowerOutputOSCAvg		
OSC Optical Power Receive (OPR-OSC)	MW	OTS-IF			opticalPowerInputOSC		OSC Receive power on MW port
				min	opticalPowerInputOSCMin		
				max	opticalPowerInputOSCMax		
				avg	opticalPowerInputOSCAvg		
Optical Channel Power Transmit (OPT-OCH)	MW	OCH-IF	x		opticalPowerOutput		individual optical signal power on MW port
				min	opticalPowerOutputMin		
				max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
Optical Channel Power Receive (OPR-OCH)	MW	OCH-IF			opticalPowerInput		individual optical signal power on MW port, includes VOA attenuation
				min	opticalPowerInputMin		
				max	opticalPowerInputMax		
				avg	opticalPowerInputAvg		
Wr Ports							
Optical Power Receive (OPR)	Wr	SRG client port	x		opticalPowerInput		optical total signal power on Wr port (from transponder) – single wavelength
				min	opticalPowerInputMin		
				max	opticalPowerInputMax		
				avg	opticalPowerInputAvg		
Optical Power Transmit (OPT)	Wr	SRG client port			opticalPowerOutput		optical total signal power on Wr port (to transponder) – multiple wavelengths
				min	opticalPowerOutputMin		
				max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		
OSC Ports							
Code Violations (CV-PCS)	OSC	Eth-IF			codeViolations		IEEE 802.3ah, Section 45.2.1.44; 8B/10B errors

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
Errored Seconds (ES-PCS)	OSC	Eth-IF			erroredSeconds		IEEE 802.3ah, Section 45.2.1.46
Severely Errored Seconds (SES-PCS)	OSC	Eth-IF			severlyErroredSeconds		IEEE 802.3ah, Section 45.2.1.48
Unavailable Seconds PCS (UAS-PCS)	OSC	Eth-IF			unavailableSeconds		IEEE 802.3ah, Section 45.2.1.52
In frames (INFRAMES-E)	OSC	Eth-IF			inFrames		
In frames errored (INFRAMESERR-E)	OSC	Eth-IF			inFramesErrored		
Out frames (OUTFRAMES-E)	OSC	Eth-IF			outFrames		
Errored Seconds Ethernet (ES-E)	OSC	Eth-IF			erroredSecondsEthernet		IEEE 802.3ah, Section 45.2.1.46
Severely Errored Seconds Ethernet (SES-E)	OSC	Eth-IF			severlyErroredSecondsEthernet		IEEE 802.3ah, Section 45.2.1.48

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
Unavailable Seconds Ethernet (UAS-E)	OSC	Eth-IF			unavailableSecondsEthernet		IEEE 802.3ah, Section 45.2.1.52
OSC Optical Power Receive (OPR-OSC)	OSC	port			opticalPowerInput		OSC Receive power on OSC port
				min	opticalPowerInputMin		
				max	opticalPowerInputMax		
				avg	opticalPowerInputAvg		
OSC Optical Power Transmit (OPT-OSC)	OSC	port			opticalPowerOutput		OSC Transmit power on OSC port
				min	opticalPowerOutputMin		
				max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		
W Ports							
Current Output Power	W	port			opticalPowerOutput		Single wavelength to Wr
				min	opticalPowerOutputMin		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
Current Input Power	W	OCH-IF OTSI-IF		max	opticalPowerOutputMax		
				avg	opticalPowerOutputAvg		
					opticalPowerInput		Single tuned wavelength
				min	opticalPowerInputMin		
				max	opticalPowerInputMax		
Total Input Power	W	port		avg	opticalPowerInputAvg		
					totalOpticalPowerInput		Multiple wavelengths from Wr
				min	totalOpticalPowerInputMin		
				max	totalOpticalPowerInputMax		
pFECcorrErr	W	OTU-IF (on transponder network RX port) OTSI-IF	x		totalOpticalPowerInputAvg		
					preFECCorrectedErrors		Ref) G798 : 6.5.1.3

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
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
	W	ODU-IF			erroredBlockCountTCM1-up		
	W	ODU-IF			erroredBlockCountTCM2-up		
	W	ODU-IF			erroredBlockCountTCM3-up		
	W	ODU-IF			erroredBlockCountTCM4-up		
	W	ODU-IF			erroredBlockCountTCM5-up		
	W	ODU-IF			erroredBlockCountTCM6-up		
	W	ODU-IF			erroredBlockCountTCM1-down		
	W	ODU-IF			erroredBlockCountTCM2-down		
	W	ODU-IF			erroredBlockCountTCM3-down		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
	W	ODU-IF			erroredBlockCountTCM4-down		
	W	ODU-IF			erroredBlockCountTCM5-down		
	W	ODU-IF			erroredBlockCountTCM6-down		
pF_EBC (BEI)	W	OTU-IF ODU-IF			erroredBlockCount		Ref) G.798 : 6.5.1.1 location = farEnd
	W	ODU-IF			erroredBlockCountTCM1-up		
	W	ODU-IF			erroredBlockCountTCM2-up		
	W	ODU-IF			erroredBlockCountTCM3-up		
	W	ODU-IF			erroredBlockCountTCM4-up		
	W	ODU-IF			erroredBlockCountTCM5-up		
	W	ODU-IF			erroredBlockCountTCM6-up		
	W	ODU-IF			erroredBlockCountTCM1-down		
	W	ODU-IF			erroredBlockCountTCM2-down		
	W	ODU-IF			erroredBlockCountTCM3-down		
	W	ODU-IF			erroredBlockCountTCM4-down		
	W	ODU-IF			erroredBlockCountTCM5-down		
	W	ODU-IF			erroredBlockCountTCM6-down		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
pN_delay	W	ODU-IF			delay		Ref) G798 : number of frames between a DMValue toggle event and the received DMp signal value toggle event location = nearEnd
	W	ODU-IF			delayTCM1-up		
	W	ODU-IF			delayTCM2-up		
	W	ODU-IF			delayTCM3-up		
	W	ODU-IF			delayTCM4-up		
	W	ODU-IF			delayTCM5-up		
	W	ODU-IF			delayTCM6-up		
	W	ODU-IF			delayTCM1-down		
	W	ODU-IF			delayTCM2- down		
	W	ODU-IF			delayTCM3- down		
	W	ODU-IF			delayTCM4- down		
	W	ODU-IF			delayTCM5- down		
	W	ODU-IF			delayTCM6- down		
pIAE	W	OTU-IF			IncomingAlignmentError		Ref) G.798 Section 6.2.6.10
pBIAE	W	OTU-IF			backwardIncomingAlignmentError		Ref) G.798 Section 6.2.6.11

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
RX RS-FEC corrected codewords counter	W	ETH-IF			fecCorrectedCodewords		(100GE) only BjFEC=ON Ref) IEEE 802.3- 2018 Sections 45.2.1.112,91 (400GE) Ref) IEEE 802.3- 2018 Sections 45.2.3.61,119
RX RS-FEC uncorrected codewords counter	W	ETH-IF			fecUncorrectedCodewords		(100GE) only BjFEC=ON Ref) IEEE 802.3- 2018 Sections 45.2.1.113,91 (400GE) Ref) IEEE 802.3- 2018 Sections 45.2.3.62,119
RX RS-FEC symbol error	W	ETH-IF			fecSymbolErrors		(100GE) only BjFEC=ON Sum of each lane

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
counter detected in FEC							notified Ref) IEEE 802.3- 2018 Sections 45.2.1.115/116,8 3 (400GE) Ref) IEEE 802.3- 2018 Sections 45.2.3.57/59, 119
RX_ Errored blocks	W	ETH-IF			erroredBlockCount		Ref) IEEE 802.3ba-2018 Sections 45.2.3,62.4,83 direction = rx
TX_ Errored blocks	W	ETH-IF			erroredBlockCount		Ref) IEEE 802.3ba-2018 Section 81.3.a direction = tx
RX_ BIP error counter	W	ETH-IF			codeViolations		Sum of each lane is notified

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
							Ref) IEEE 802.3ba-2018 Sections 45.2.3.46/47, 83 direction = rx
TX_ BIP error counter	W	ETH-IF			codeViolations		Ref) IEEE 802.3ba-2018 Section 81.3.4 direction = tx
PMs currently not mapped to any PMs listed in the Optical Specification							
					bitErrorRate		
					defectSeconds		
					localFaultSeconds		
					remoteFaultSeconds		
					partialRateDiscard		
					protectionSwitchingCount		
					protectionSwitchingDuration		
Errored Seconds (ES)	W	OTU-IF ODU-IF			erroredSeconds		
		ODU-IF			erroredSecondsTCM1-up		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
		ODU-IF			erroredSecondsTCM2-up		
		ODU-IF			erroredSecondsTCM3-up		
		ODU-IF			erroredSecondsTCM4-up		
		ODU-IF			erroredSecondsTCM5-up		
		ODU-IF			erroredSecondsTCM6-up		
		ODU-IF			erroredSecondsTCM1-down		
		ODU-IF			erroredSecondsTCM2-down		
		ODU-IF			erroredSecondsTCM3-down		
		ODU-IF			erroredSecondsTCM4-down		
		ODU-IF			erroredSecondsTCM5-down		
		ODU-IF			erroredSecondsTCM6-down		
Severely Errored Seconds (SES)	W	OTU-IF ODU-IF			severelyErroredSeconds		
		ODU-IF			severelyErroredSecondsTCM1-up		
		ODU-IF			severelyErroredSecondsTCM2-up		
		ODU-IF			severelyErroredSecondsTCM3-up		
		ODU-IF			severelyErroredSecondsTCM4-up		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
		ODU-IF			severelyErroredSecondsTCM5-up		
		ODU-IF			severelyErroredSecondsTCM6-up		
		ODU-IF			severelyErroredSecondsTCM1-down		
		ODU-IF			severelyErroredSecondsTCM2-down		
		ODU-IF			severelyErroredSecondsTCM3-down		
		ODU-IF			severelyErroredSecondsTCM4-down		
		ODU-IF			severelyErroredSecondsTCM5-down		
		ODU-IF			severelyErroredSecondsTCM6-down		
Unavailable Seconds (UAS)	W	OTU-IF ODU-IF			unavailableSeconds		
		ODU-IF			unavailableSecondsTCM1-up		
		ODU-IF			unavailableSecondsTCM2-up		
		ODU-IF			unavailableSecondsTCM3-up		
		ODU-IF			unavailableSecondsTCM4-up		
		ODU-IF			unavailableSecondsTCM5-up		
		ODU-IF			unavailableSecondsTCM6-up		
		ODU-IF			unavailableSecondsTCM1-down		

MSA PM	MW Wr OSC W	Reported Against...	Used by Controller	Tide Marks	Open ROADM MSA YANG version 7.1		Comment
					Type	extension	
		ODU-IF			unavailableSecondsTCM2-down		
		ODU-IF			unavailableSecondsTCM3-down		
		ODU-IF			unavailableSecondsTCM4-down		
		ODU-IF			unavailableSecondsTCM5-down		
		ODU-IF			unavailableSecondsTCM6-down		

Table 21: Alarms

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
W Ports							
OTU - LOS 100GE Signal Det	<p>OTN : It is shown that the input power drop. Ethernet : Ref) IEEE 802.3ba Section 86.2</p> <p>Physical maps to port level LOS</p> <p>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.)</p>	portLossOfLight		port			
		lossOfSignal		otsi-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
	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...	lossOfSignal		och-if			
		linkDown		eth-if			
LOF	Ref) G798 : 6.2.5.1	lossOfFrame		otu-if			
LOM	Ref) G798 : 6.2.5.2	lossOfMultiframe		otu-if otsi-if			
BDI	Ref) G798:6.2.6.6	backwardsDefectIndication		otu-if odu-if			
		backwardsDefectIndicationTCM1-up		odu-if			
		backwardsDefectIndicationTCM2-up		odu-if			
		backwardsDefectIndicationTCM3-up		odu-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		backwardsDefectIndicationTCM4-up		odu-if			
		backwardsDefectIndicationTCM5-up		odu-if			
		backwardsDefectIndicationTCM6-up		odu-if			
		backwardsDefectIndicationTCM1-down		odu-if			
		backwardsDefectIndicationTCM2-down		odu-if			
		backwardsDefectIndicationTCM3-down		odu-if			
		backwardsDefectIndicationTCM4-down		odu-if			
		backwardsDefectIndicationTCM5-down		odu-if			
		backwardsDefectIndicationTCM6-down		odu-if			
Group ID Mismatch	Ref) G.798 (2019-12) 15.3.1.2	groupIdMismatch		otsig-if			
FlexO Map Mismatch	Ref) G.798 (2019-12) 15.3.1.2	flexoMapMismatch		otsig-if			
Loss of Frame and Loss of Multiframe	Ref) G.798 (2019-12) : 15.3.1.2	lossOfFrameAndLossOfMultiframe		otsig-if			
Loss of Lane Alignment	Ref) G.798 (2019-12) 15.3.1.2, 16.7.2	lossOfLaneAlignment		otsi-if otsig-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
Loss of Signal Payload	Ref) G.798 (2019-12) 16.7.2	lossOfSignalPayload		otsi-if			
DEG	Ref) G798:6.2.3	degradedDefect		otu-if odu-if			
		degradedDefectTCM1-up		odu-if			
		degradedDefectTCM2-up		odu-if			
		degradedDefectTCM3-up		odu-if			
		degradedDefectTCM4-up		odu-if			
		degradedDefectTCM5-up		odu-if			
		degradedDefectTCM6-up		odu-if			
		degradedDefectTCM1-down					
		degradedDefectTCM2-down					
		degradedDefectTCM3-down					
		degradedDefectTCM4-down					
		degradedDefectTCM5-down					
		degradedDefectTCM6-down					
TTI	Ref) G798:6.2.2.1	trailTraceIdentifierMismatch		otu-if odu-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		trailTraceIdentifierMismatchTCM1-up		odu-if			
		trailTraceIdentifierMismatchTCM2-up		odu-if			
		trailTraceIdentifierMismatchTCM3-up		odu-if			
		trailTraceIdentifierMismatchTCM4-up		odu-if			
		trailTraceIdentifierMismatchTCM5-up		odu-if			
		trailTraceIdentifierMismatchTCM6-up		odu-if			
		trailTraceIdentifierMismatchTCM1-down		odu-if			
		trailTraceIdentifierMismatchTCM2-down		odu-if			
		trailTraceIdentifierMismatchTCM3-down		odu-if			
		trailTraceIdentifierMismatchTCM4-down		odu-if			
		trailTraceIdentifierMismatchTCM5-down		odu-if			
		trailTraceIdentifierMismatchTCM6-down		odu-if			
AIS	Ref) G798:6.2.6.3.2	alarmIndicationSignal		otu-if odu-if			
		alarmIndicationSignalTCM1-up		odu-if			
		alarmIndicationSignalTCM2-up		odu-if			
		alarmIndicationSignalTCM3-up		odu-if			
		alarmIndicationSignalTCM4-up		odu-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		alarmIndicationSignalTCM5-up		odu-if			
		alarmIndicationSignalTCM6-up		odu-if			
		alarmIndicationSignalTCM1-down		odu-if			
		alarmIndicationSignalTCM2-down		odu-if			
		alarmIndicationSignalTCM3-down		odu-if			
		alarmIndicationSignalTCM4-down		odu-if			
		alarmIndicationSignalTCM5-down		odu-if			
		alarmIndicationSignalTCM6-down		odu-if			
OCI	Ref) G798:6.2.6.8	openConnectionIndication		odu-if			
		openConnectionIndicationTCM1-up		odu-if			
		openConnectionIndicationTCM2-up		odu-if			
		openConnectionIndicationTCM3-up		odu-if			
		openConnectionIndicationTCM4-up		odu-if			
		openConnectionIndicationTCM5-up		odu-if			
		openConnectionIndicationTCM6-up		odu-if			
		openConnectionIndicationTCM1-down		odu-if			
		openConnectionIndicationTCM2-down		odu-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		openConnectionIndicationTCM3-down		odu-if			
		openConnectionIndicationTCM4-down		odu-if			
		openConnectionIndicationTCM5-down		odu-if			
		openConnectionIndicationTCM6-down		odu-if			
LCK	Ref) G798:6.2.6.9	lockedDefect		odu-if			
		lockedDefectTCM1-up		odu-if			
		lockedDefectTCM2-up		odu-if			
		lockedDefectTCM3-up		odu-if			
		lockedDefectTCM4-up		odu-if			
		lockedDefectTCM5-up		odu-if			
		lockedDefectTCM6-up		odu-if			
		lockedDefectTCM1-down		odu-if			
		lockedDefectTCM2-down		odu-if			
		lockedDefectTCM3-down		odu-if			
		lockedDefectTCM4-down		odu-if			
		lockedDefectTCM5-down		odu-if			
		lockedDefectTCM6-down		odu-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
LTC	Ref) G798:6.2.1.4	lossofTandemConnectionTCM1-up		odu-if			
		lossofTandemConnectionTCM2-up		odu-if			
		lossofTandemConnectionTCM3-up		odu-if			
		lossofTandemConnectionTCM4-up		odu-if			
		lossofTandemConnectionTCM5-up		odu-if			
		lossofTandemConnectionTCM6-up		odu-if			
		lossofTandemConnectionTCM1-down		odu-if			
		lossofTandemConnectionTCM2-down		odu-if			
		lossofTandemConnectionTCM3-down		odu-if			
		lossofTandemConnectionTCM4-down		odu-if			
		lossofTandemConnectionTCM5-down		odu-if			
		lossofTandemConnectionTCM6-down		odu-if			
PLM	Ref) G798 : 6.2.4.1	payloadMismatch		opu-if			
CSF	Ref) G798:6.2.10	clientSignalFailDefect		opu-if			
Loss of FEC Alignment	Only BjFEC=ON (i.e 100GBASE-SR4) Ref) IEEE 802.3-2018 Section 45.2.1.111.2, 91	lossOfFECAlignment		eth-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
PCS Loss of Sync (Rcv)	Ref) IEEE 802.3-2018 Section 45.2.3.15.1	lossOfSynchronization		eth-if		rx	
Rx_BASE-R PCS high BER	Ref) IEEE 802.3-2018 Section 45.2.3.15.4, 82	highBER		eth-if		rx	
Loss of Alignment (Rcv)	OR of each lane is notified. Ref) IEEE 802.3-2018 Section 45.2.3.23.1, 83, 119	lossOfAlignment		eth-if		rx	
Rx_Local Fault	Ref) IEEE 802.3-2018 Section 81.3.4, 117.3	localFault		eth-if		rx	
Rx_Remote Fault	Ref) IEEE 802.3-2018 Section 81.3.4, 117.3	remoteFault		eth-if		rx	
PCS Loss of Sync (Tx)	Ref) IEEE 802.3-2018 Section 45.2.3.15.1	lossOfSynchronization		eth-if		tx	
Tx_BASE-R PCS high BER	Ref) IEEE 802.3-2018 Section 45.2.3.16.2	highBER		eth-if		tx	
Loss of Alignment (Tx)	OR of each lane is notified. Ref) IEEE 802.3-2018	lossOfAlignment		eth-if		tx	

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
	Section 45.2.3.23.1, 83, 119						
Tx_Local Fault	Ref) IEEE 802.3-2018 Section 81.3.4	localFault		eth-if		tx	
Tx_Remote Fault	Ref) IEEE 802.3-2018 Section 81.3.4	remoteFault		eth-if		tx	
Local Degraded SER	Ref) IEEE 802.3-2018 Section 45.2.3.60.1, 119"	localDegradedSER		eth-if			
Remote Degraded SER	Ref) IEEE 802.3-2018 Section 45.2.3.60.2, 119"	remoteDegradedSER		eth-if			
PCS FEC Degrade SER	Ref) IEEE 802.3-2018 Section 119 Only default threshold used	fecDegradedSER		eth-if			
MW Mux (Out) Ports							
Automatic Shutoff/ Automatic Laser Shutdown	shut off amplifier laser due to safety, OSC will never be shut off	automaticLaserShutdown		oms-if			

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
Automatic Shutoff Disabled	RPC was triggered to shut off laser safety, standing alarm	automaticShutoffDisabled		oms-if			
Automatic Power Reduction Active (optional)	during high reflection event, power is lowered, but not shut off	automaticPowerReduction		port			
High Reflection/ Low Optical Return Loss	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.	reflectionTooHigh		ots-if or oms-if			
Loss of Signal	May be used by the controller;	lossOfSignal		och-if		tx	

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
	Not masked since there is no LOS on OTS (tx); Optional for device to report						
MW Demux (In) Ports							
Loss of Signal	LOS on OTS includes OSC and OMS; LOS on OTS masks LOS on OMS but does not mask LOS on OSC and OMS LOS on OTS is used by the controller.	lossOfSignal		ots-if		rx	
	LOS on OMS	lossOfSignal		oms-if		rx	
	LOS on OSC	lossOfSignalOSC		ots-if		rx	
	LOS on OCH Masked by LOS on OTS (rx); Optional for device to report	lossOfSignal		och-if		rx	
Link Down	Link Down on OSC	linkDown		eth-if		rx	

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
Far End Fault Indication	Far End Fault Indication on 100M OSC align with MSA Laser Safety "EthFEFI" alarm	farEndFaultIndication		eth-if		rx	
Low Optical Power Warning (Optical Power Degraded)		opticalPowerDegraded		ots-if		rx	
Optical Line Fail	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	opticalLineFail		ots-if		rx	NEND
Wr Mux (In) Ports							
Loss of Signal		portLossOfLight		port		rx	
Optical Power Degraded		opticalPowerDegraded		port		rx	

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
Probable causes currently not mapped to any Alarms listed in Optical Specification							
		administrativeDown					
		automaticSwitchDueToWTR					
		automaticSwitchAwayFromProtectDueToSD					
		automaticSwitchAwayFromProtectDueToSD					
		automaticSwitchAwayFromProtectDueToSF					
		automaticSwitchAwayFromWorkingDueToSF					
		certificateNotInstalled					
		circuitPackActivateFailed					
	(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	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	cmfLossOfSignal					

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
	0x01(G.7041: Table 6-4) is received						
	(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					
		createTechInfoInProgress					
		databaseCorruption					
		databaseLocked					
		databaseRollbackTimerInProgress					
		databaseVersionMismatch					
		diskFull					
		equipmentFault					
		equipmentInterConnectFailure					
		equipmentLedOn					
		equipmentMiscabledConnection					
		equipmentMismatch					

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		equipmentRemoved					
		equipmentWarmup					
		facilityLoopbackActive					
		facilityLoopback2Active					
		facilityTestsignalActive					
		fanCoolingFail					
	Ref) IEEE 802.3-2018 Section 24.3.2.1	farEndFaultIndication					
		firmwareDownloadOrActivationFailure					
		firmwareInitInProgress					
		firmwareVersionMismatch					
		forcedSwitchAwayFromProtect					
		forcedSwitchAwayFromWorking					
		forwardDefectIndication					
	(dLFD G.806:6.2.5.2) is raised when the frame delineation process (clause 6.3.1 of [ITU-T	gfpLossOfFrameDelineation					

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
	G.7041]) is not in the 'SYNC' state						
		lampTest					
		lldpFail					
		lockoutOfProtection					
		lossOfDatabaseRedundancy					
	"G798 : 14.3.10.2 Figure 14-73 (dLOOMFI)"	lossOfOmflIndication					
		manualSwitchAwayFromProtect					
		manualSwitchAwayFromWork					
	"G798 : 6.2.9 (dMSIM)"	multiplexStructureIdentifierMismatch					
		omsPowerOutOfSpecificationHigh					
		omsPowerOutOfSpecificationLow					
		oscPowerOutOfSpecificationHigh					
		oscPowerOutOfSpecificationLow					
		otdrScanInProgress					
		otsSpanlossPowerOutOfSpecificationHigh					

MSA Alarm	comment	cause	extension	resource resourceType	severity	direction *	location *
		payloadMissingIndication					
		powerOutOfSpecificationHigh					
		powerProblemA					
		powerProblemB					
		serverSignalFail					
		shelfProvisioningMode					
		softwareReset					
		softwareStageInProgress					
		softwareSubsystemFailed					
		softwareValidateInProgress					
		softwareVersionMismatch					
		sysNameChanged					
		sysNtpNotSynchronized					
		terminalLoopbackActive					
		terminalTestsignalActive					