Wire- and AirEquipment

or how to deal with SFPs[1] and ODUs

1 Introduction

The current Equipment Model implementations of the microwave outdoor unit (ODU) are different, which lead to the necessity of a vendor-specific implementation of the Application layer. For automation purposes, an efficient interfacing to microwave devices is required. The objective of this document is to describe how the ONF Core Model - Equipment Model shall be used to expose a harmonized API for microwave ODUs.

Information modeling of Equipment has to cope with a couple of specific aspects.

> Example 1: The Capability information of the individual WireInterface instance depends on the type of SFP (SFP shall represent all types of physical connectors in this text), but not all types and models of SFPs, which might be used with the device, are known during design time of the firmware of the device.
>
> Example 2: Further on, some concrete Transmitter (e.g. SFP or ODU) might be replaced due to failure and the replacing Transmitter[2] might just be similar, but not of identical model or hardware release. Nevertheless, an already instantiated Interface object should continue existing and also operating, in case original and replacing Transmitter have sufficiently identical characteristics.

This document does not describe technology specific attributes, which are augmented to the ONF Core Model, but how to use the existing attributes of the Core Physical Model.

2 Contradicting Approaches

The resulting information modeling has to cope with contradictory processes.

Roll-out

The Plan-Build-Operate approach represents nowadays method of humans using planning tools to define a future should-be-network. The planning is often done before the devices are actually existing in the network and it shall be exactly executed, or not at all. This means that Interfaces[3] and ExpectedEquipment are instantiated and configured on the application layer first and on the device later.

The Plug'n'Play Approach assumes that applications are automatically configuring already existing devices. In divergence to the Planning Driven Approach, ExpectedEquipment is instantiated on the device first (namely identically to the ActualEquipment) and afterwards on the application layer.

Repair

Replacing Transmitters is another area of contradictory requirements. Surely, the majority of users is expecting the configuration of an interface[4] to "survive" a failure of the associated Transmitter. It can also be anticipated that the replacing Transmitter might not be of the same version or even model.

The modeling described in this document leaves it to the operators' responsibility to decide about which hardware related information shall be assessed for assuring sufficient compatibility of an existing Interface Configuration and a replacing Transmitter.

[1] SFP shall represent all types of physical connectors in this text, e.g. a soldered RJ45 connector shall be seen as a permanently plugged SFP.
[2] "Transmitter" could be either soldered port (e.g. RJ45), SFP or ODU.
[3] "Interface" (starting with upper case) relates to a technology specifically amended instance of the LayerProtocol class.
[4] "interface" (starting with lower case) relates the general linguistic usage.

3 Requirements

Indicate physical connector related with Transmitter

User and Application must have a possibility to identify the location of the Transmitter at the outside of the device (e.g. SFP cage or IF connector for the ODU). This is important for example for changing a faulty ODU or identifying which IF ports are still available on physical devices.

Replacement Transmitter without remove logical configuration

It shall be possible to replace a broken Transmitter without changing the related logical termination points.

Possibility to distinguish type of physical equipment support

The Equipment object models many of physical types of components (Board, SFP, Ports, ODU). It shall be possible to clearly identify the type of component and model by a human readable name. So, users can quickly identify and filter component types or models.

4 Successful Configuration

In accordance with Requirement no. 3 of the TR-545 DMIP, a <commit> operations for changing configuration values of an Interface shall be denoted successful, whenever the changes have been successfully validated against the characteristic of an ExpectedEquipment.

This means that any process of instantiating and configuring an Interface must start with defining the ExpectedEquipment.

5 General Understanding

In general, the management interface defined in related information models and also this specification is for making the functions of the hardware available. Some function not available on the hardware cannot be made available by the management interface.

An instance of the Equipment class represents an abstract potential for equipment, which is already bound to a specific position.

An instance of the ActualEquipment class represents a concrete module, which is actually plugged into the Device.

The group of instances of the ExpectedEquipment class, which is associated with the instance of the Equipment class, represents the variety of modules that could be plugged for operating the same interface configuration (according to general linguistic usage).

An instance of the Connector class represents a physical connector, which will be available with the ExpectedEquipment at the outside of the device.

All existing Equipment instances have to be associated (composite) by the _equipment[*] attribute to the ControlConstruct instance. The _occupyingFru[5][0..1] attribute at the Holder instance is just referencing (shared) already existing Equipment instances.

Independently from the mounting form (split-mount or pure-outdoor), there shall be only one top level Equipment associated (_top-level-equipment) with a ControlConstruct of a microwave. In case of split-mount, the IF port is represented by an instance of Holder and the ODU by a subordinate instance of Equipment.

6 Instantiation

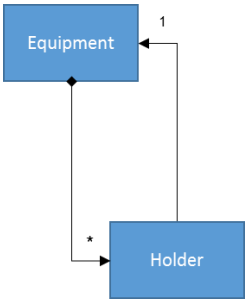
Root Instantiation

(1) ControlConstruct as well as ActualEquipment associated with hardware already present in the device shall be instantiated as the device is booting up.

[5] FRU = Field Replaceable Unit

Equipment and Holder

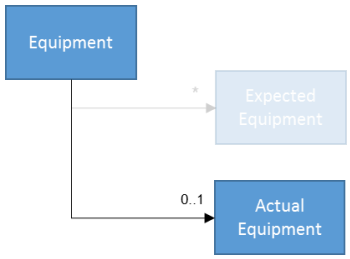
(2) Holder as well as associated Equipment shall be automatically instantiated as soon as the ExpectedEquipment in the overlying level of the hierarchy got instantiated. Every existing Holder instance shall always be referencing an Equipment instance (the _occupyingFru attribute at the Holder object shall always contain the UUID of an Equipment object that waits to be detailed with ExpectedEquipment and ActualEquipment objects).



Example 3: As soon as an Ethernet board accommodating ten SFP cages got instantiated as ExpectedEquipment, the corresponding ten Holder and ten Equipment objects shall be instantiated. Everything shall be prepared for instantiating an SFP as ExpectedEquipment in the next step.

ActualEquipment

The Equipment class of the ONF Core IM comprises a reference to 0 or 1 ActualEquipment instances.



(3) If there is no real hardware present (includes: never before, just pulled, or broken), which is implementing the Equipment, there shall be no instance of ActualEquipment.

Example 4: If there is no ODU connected (includes: never before, IF cable just pulled, or ODU broken), there shall be no instance of ActualEquipment.

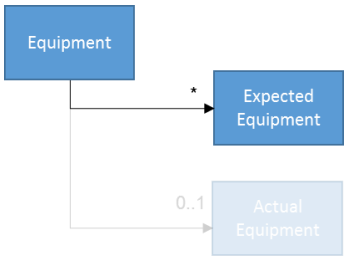
(4) When a new hardware component is put into the device, the device shall automatically instantiate an ActualEquipment object and associate it with the Equipment object.

Example 5: When an SFP is put into the cage, the device shall automatically instantiate an ActualEquipment object and associate it with the Equipment object.

ExpectedEquipment

The Equipment class of the ONF Core IM comprises a reference to a list of an undefined number of ExpectedEquipment instances.

The instantiation of ExpectedEquipment has to support the Plug'n'Play process, which requires an automated creation of the ExpectedEquipment instance, as well as the Plan-Build-Operate process, which includes configuration of the ExpectedEquipment via management interface without the represented hardware component actually being present.



(5) An arbitrary number of ExpectedEquipment objects might be instantiated and associated to the Equipment object via the management interface at any time. Associating more than one ExpectedEquipment object with an Equipment object makes exclusively sense, if all those ExpectedEquipment objects are representing hardware with identical characteristics.

Example 6: Several models of SFPs, potentially even of different vendors, but with the same characteristics might be instantiated as ExpectedEquipment as a routine to facilitate smooth spare part appliance.

Warning: If the operator would configure components of different characteristics into the ExpectedEquipment list, the Device will not check for compatibility, but operate an existing configuration that had been made based on the characteristics of the ExpectedEquipment, which had originally been instantiated first.

(6) A single ExpectedEquipment object shall be automatically instantiated by the device whenever an ActualEquipment object gets instantiated and no ExpectedEquipment object is already existing.

Example 7: Given the situation that a modular device, which does not contain any board except the one that holds the management interface, gets powered and switched on for the first time. Because the chassis of the device is there, a highest level Equipment object and ActualEquipment and ExpectedEquipment objects, which are representing the chassis, must be instantiated. In addition,

Holder and subordinated Equipment objects representing all slots must be instantiated. At the correct subordinated Equipment object, ActualEquipment and ExpectedEquipment objects, which are representing the management module, plus Holder and subordinated Equipment objects, which are representing the management interfaces, must be instantiated. If there would be an already plugged SFP on that board, it would also have to be represented by ActualEquipment and ExpectedEquipment objects (subordinated Holder and Equipment objects would not be required).

Connector

(7) As soon as an ExpectedEquipment object has been instantiated, the device shall automatically instantiate all Connector objects, which are related with this hardware.

Example 8: As soon as an ExpectedEquipment object, which is representing an SFP, has been instantiated, the device shall automatically instantiate a Connector.

Example 9: As soon as an ExpectedEquipment object, which is representing an Ethernet board with four RJ45 ports at its front, has been instantiated, the device shall automatically instantiate four Connector objects.

LogicalTerminationPoint and LayerProtocol

(8) As soon as an ExpectedEquipment object has been instantiated, the device shall automatically instantiate all LogicalTerminationPoint objects (including technology specifically augmented LayerProtocol objects), which are unquestionable related with this hardware. (Some combinations of board, port and SFP might support several kinds of logical layers, e.g. Ethernet and synchronization. In such case, the payload interface shall be instantiated, e.g. Ethernet.)

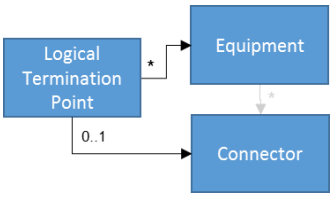
Example 10: As soon as an ExpectedEquipment object, which is representing an SFP, has been instantiated, the device shall automatically instantiate, LTP(WireInterface)[6] and LTP(PureEthernetStructure)[7].

Example 11: As soon as an ExpectedEquipment object, which is representing an Ethernet Switch with four internal interfaces, has been instantiated, the device shall automatically instantiate four LTP(EthernetContainer)*, four LTP(MacInterface)* and four LTP(VlanInterface) (only in case the device supports VLAN bridging).. If there would be fixed RJ45 connectors or already plugged SFPs, which are making these interfaces available at the front of that Ethernet Switch board, corresponding Equipment, Connector, LTP(WireInterface) and LTP(PureEthernetStructure) objects would have to be instantiated, too[8].

Example 12: One or several IpInterfaces shall be automatically created by the device for representing the management interface(s) towards the device[9].

(9) The LogicalTerminationPoint::_equipment : Equipment[1..*] attribute shall be automatically configured with the UUID of the Equipment object, which is determining the characteristics of the LogicalTerminationPoint object.

(10) The LogicalTerminationPoint::_connector : Connector[0..1] attribute of all LogicalTerminationPoint objects, which do not have any UUID in the _serverLtp



[6] LTP(WireInterface) = LogicalTerminationPoint and LayerProtocol object, which are augmented by technology specific information about a WireInterface and the value of its layerProtocolName attribute = "LAYER_PROTOCOL_NAME_TYPE_WIRE_LAYER".

[7] LTP(PureEthernetStructure) = LTP/LP, which are augmented with the technology specific information about a PureEthernetStructure and the value of its layerProtocolName attribute = "LAYER_PROTOCOL_NAME_TYPE_PURE_ETHERNET_STRUCTURE_LAYER".

[8] The examples do not have normative character. Some implementations might instantiate the entire stack of interfaces at the moment the physical connector got defined as an ExpectedEquipment object.

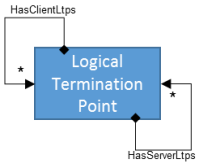
[9] The examples do not have normative character. Some implementations might provide separate connectors for management purposes, others might separate management by VLAN from other traffic on conventional payload connectors.

attribute, shall be automatically configured with the LID[10] of the Connector object, which is representing the physical connector at the outside of the device.

(6) and (8) shall at least apply for all hardware types (e.g. SFP models or ODU versions), which could be purchased from the provider of the device during design time of the firmware or are currently plugged.

(11) If there is a fixed relationship between a server and a client layer, the _clientLtp attribute of the LogicalTerminationPoint, which is representing the server layer, shall automatically be configured to point to the LogicalTerminationPoint, which is representing the client layer.

(12) If there is a fixed relationship between a server and a client layer, the _serverLtp attribute of the LogicalTerminationPoint, which is representing the client layer, shall automatically be configured to point to the LogicalTerminationPoint, which is representing the server layer.



> Example 13: As soon as an LTP(WireInterface) gets instantiated, an LTP(PureEthernetStructure) shall be automatically instantiated, too. The _clientLtp attribute of the LTP(WireInterface) shall automatically be configured to point to the LTP(PureEthernetStructure). The _serverLtp attribute of the LTP(PureEthernetStructure) shall automatically be configured to point to the LTP(WireInterface)[11]. (Because the _serverLtp attribute of the LTP(WireInterface) cannot point to anything, its _connector attribute has to point to a Connector object.)
>
> Example 14: As soon as an ExpectedEquipment object representing a radio has been instantiated, the device shall automatically instantiate a related LTP(AirInterface)[12]. In addition, the LTP(PureEthernetStructure) or the LTP(HybridMwStructure)[13] shall be instantiated including the client/server relationships based on the device capabilities.
>
> Example 15: As soon as an LTP(PureEthernetStructure) gets instantiated and an LTP(EthernetContainer)[14] is already existing (must have automatically been instantiated right after the ExpectedEquipment object, which is representing the Ethernet Switch board, has been instantiated, which is necessarily before the ExpectedEquipment object, which is representing the SFP, could be instantiated), the _clientLtp attribute of the LTP(PureEthernetStructure) shall automatically be configured to point to the LTP(EthernetContainer) and the _serverLtp attribute of the LTP(EthernetContainer) shall automatically be configured to point to the LTP(PureEthernetStructure).

ForwardingDomain and ForwardingConstruct

(13) As soon as an ExpectedEquipment object has been instantiated, the device shall automatically instantiate all ForwardingDomain objects (including technology specific augments), which are unquestionable related with this hardware.

(14) The ForwardingDomain::_ltp : LogicalTerminationPoint [*] attribute shall be automatically configured with the UUIDs of all the LogicalTerminationPoint objects, which are part of the potential forwarding represented by the ForwardingDomain.

(15) As soon as an ExpectedEquipment object has been instantiated, the device shall automatically instantiate all ForwardingConstruct and FcPort objects (including technology specific augments), which are required for describing a potential default configuration. The FcPort::_ltp : LogicalTerminationPoint [0..2] attribute shall be

[10] Value of localId attribute

[11] The examples do not have normative character. Some implementations might provide separate connectors for management purposes, others might separate management by VLAN from other traffic on conventional payload connectors.

[12] LTP(AirInterface) = LogicalTerminationPoint and LayerProtocol object, which are augmented by technology specific information about a AirInterface and the value of its layerProtocolName attribute = “LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER”.

[13] LTP(HybridMwStructure) = LogicalTerminationPoint and LayerProtocol object, which are augmented by technology specific information about a HybridMwStructure and the value of its layerProtocolName attribute = LAYER_PROTOCOL_NAME_TYPE_HYBRID_MW_STRUCTURE_LAYER”.

[14] LTP(EthernetContainer) = LogicalTerminationPoint instance, which is associated with a LayerProtocol instance, which is technology specifically augmented with the EthernetContainer model (value of the layerProtocolName attribute = “LAYER_PROTOCOL_NAME_TYPE_ETHERNET_CONTAINER_LAYER”).

automatically configured with the UUIDs of the LogicalTerminationPoint objects, which are part of the actual forwarding represented by the respective ForwardingConstruct.

Profile

(16) As soon as an ExpectedEquipment object has been instantiated, the device shall automatically instantiate at least one default Profile object (including technology specific augmentation), for each type of Profile supported by this equipment type (except the CoChannelProfile or any other Profile, which is pointing into direction to the LTPs).

> Example 16: If a change to the WRED configuration would affect all interfaces at the device, a Profile(WredProfile) shall be instantiated, as soon as the ExpectedEquipment object, which is representing the chassis, has been instantiated.

(17) If there would be an attribute, which is referencing a Profile object, with a multiplicity of 1 or higher (composite association) included in the technology specifically amended LogicalTerminationPoint, this attribute shall be configured to associate the default Profile object right after instantiation of the LogicalTerminationPoint and its extension.

Sequence

(18) After Instantiation of the ExpectedEquipment object, the related Connector objects shall be automatically instantiated first, the related Profile objects (if any) second, and the related LogicalTerminationPoint objects third.

(19) After instantiation of the LogicalTerminationPoint, the associations towards Client and Server LTPs (as far as already existing) shall be configured first, the association to the Connector object (if required) shall be configured second, the association to the Equipment object third and potential associations towards Profile objects forth (if any).

(20) After all Connector, Profile and LogicalTerminationPoint objects, which are related to some ExpectedEquipment object, have been instantiated and associated, all Holder, Equipment and ActualEquipment objects of the subordinate layer shall be recursively and automatically created (first) and associated (second).

(21) After that, the first ExpectedEquipment objects of the subordinate layer shall be created (first) and associated with the Equipment object (second).

Externally initiated Instantiations

(22) If instantiation of a LogicalTerminationPoint gets initiated via the management interface, values of the LogicalTerminationPoint::_equipment attribute and the LayerProtocol::layerProtocolName attributes must be provided during instantiation. Both attributes must not be changed during lifetime of the LogicalTerminationPoint object.

(23) If instantiation of a Profile gets initiated via the management interface, the value of the Profile::ProfileName attribute must be provided. Value of this attribute must not be changed during lifetime of the Profile object. (Please be aware, that attributes at Profile objects are in general invariant. This means that all configuration values have to be provided during instantiation.)

Characteristics of Unknown Hardware

(24) In case some hardware (e.g. SFP), which was not known to the vendor during design time of the management interface, gets plugged and operated with the device, the necessary information for restoring the related logical objects (e.g. ExpectedEquipment, Profiles and LTPs incl. capability information) shall be read from the component and permanently stored inside the Device.

> Example 17: A latest generation SFP, which was not known during design of the firmware, is plugged and operated in the Device. It broke, but didn't become replaced before the device got hit by a power failure. During rebooting, the Device has to re-create all logical objects on the management interface. Because the Device can no longer read the characteristics, which are required to re-create the LTP (incl. Capability information) from the broken SFP, it must be possible to retrieve the necessary information from an internal storage.

7 Configuration

Interface Configuration

(25) Any change of the Interface Configuration shall be validated against its Capability information (and naturally the actual hardware characteristics).

(26) The Capability information at an Interface shall perpetually embody the characteristics of that type of ExpectedEquipment, which initiated the instantiation of the LogicalTerminationPoint (ExpectedEquipment instantiated first).

ExpectedEquipment Configuration

(27) An existing ExpectedEquipment shall be invariant.

It is the operators' responsibility to assure that additionally instantiated ExpectedEquipment objects (e.g. representing SFP types, which have been approved for alternative use) are matching the Capability information that has been written into the technology specifically augmented LayerProtocol object during its instantiation.

8 Deletion

ExpectedEquipment Deletion

(28) At the moment the last remaining ExpectedEquipment object gets disassociated from the Equipment object, all associated Connector and LogicalTerminationPoints objects including their associated technology specifically augmented LayerProtocol objects have to be automatically deleted by the Device.

(29) At the moment the last ExpectedEquipment object gets disassociated from the overlaying Equipment object, all subordinated Holder and Equipment objects including their associated ActualEquipment and ExpectedEquipment objects and so on have to be automatically deleted by the Device.

(30) Lowering the number of ExpectedEquipment objects (or even deleting the first one) shall neither impact the LogicalTerminationPoint instance nor the values of the technology specifically augmented LayerProtocol object as long as at least one ExpectedEquipment object remains.

9 Information

ActualEquipment

(31) The ActualEquipment object (incl. associated objects) shall contain as much as possible information about the currently plugged hardware component.

For pluggable SFPs, this shall include at least the following information:

<table>
<tr><th>Attribute Name in Core IM</th><th>To be filled with following data according to SFF-8472 Rev12.3</th></tr>
<tr><td>ManufacturerProperties::manufacturerIdentifier</td><td>Vendor OUI [Address A0h, Bytes 37-39]
The vendor organizationally unique identifier field (vendor OUI) is a 3-byte field that contains the IEEE Company Identifier for the vendor. A value of all zero in the 3-byte field indicates that the Vendor of the SFP did not specify the OUI on the EEPROM.
The 3 bytes expressed in hex should be separated by hyphen.
The most-significant-byte and most-significant-bit comes first.
Example:
<table>
<tr><th colspan="3">OUI</th><th></th></tr>
<tr><td>AC</td><td>DE</td><td>48</td><td>hex</td></tr>
<tr><td>10101100</td><td>11011110</td><td>01001000</td><td>bits</td></tr>
<tr><td colspan="4">| most significant byte</td></tr>
<tr><td colspan="4">| most significant bit</td></tr>
</table>
</td></tr>
<tr><td>ManufacturerProperties::manufacturerName</td><td>Vendor name [Address A0h, Bytes 20-35]
The vendor name is a 16 character field that contains ASCII characters, left-aligned and padded on the right with ASCII spaces (20h). The vendor name shall be the full name of the corporation, a commonly accepted abbreviation of the name of the corporation, the SCSI company code for the corporation, or the stock exchange code for the</td></tr>
</table>

Attribute Name in Core IM	To be filled with following data according to SFF-8472 Rev12.3
	corporation. According to SFF 8472 Rev12.3 the vendor of the SFP has to fill at least one of the vendor name or the vendor OUI fields with valid data.
EquipmentType::partTypeIdentifier	Vendor PN [Address A0h, Bytes 40-55] The vendor part number (vendor PN) is a 16-byte field that contains ASCII characters, left-aligned and padded on the right with ASCII spaces (20h), defining the vendor part number or product name. A value of all zero in the 16-byte field indicates that the Vendor of the SFP did not specify the vendor PN on the EEPROM.
EquipmentType::version	Vendor Rev [Address A0h, Bytes 56-59] The vendor revision number (vendor rev) is a 4-byte field that contains ASCII characters, left-aligned and padded on the right with ASCII spaces (20h), defining the Vendor's product revision number. A value of all zero in the 4-byte field indicates that the Vendor of the SFP did not specify the vendor revision on the EEPROM.
EquipmentInstance::manufactureDate	Date Code [Address A0h, Bytes 84-91] The date code is an 8-byte field that contains the Vendor's date code in ASCII characters. A0h Description 84-85 ASCII code, two low order digits of year. (00 = 2000). 86-87 ASCII code, digits of month (01 = Jan through 12 = Dec) 88-89 ASCII code, day of month (01-31) 90-91 ASCII code, vendor specific lot code, may be blank The date type is a profile of the ISO 8601 standard for representation of dates using the Gregorian calendar. The profile is defined by the full-date production in Section 5.6 of RFC 3339. Values shall follow the pattern: ^\\d{4}-\\d{2}-\\d{2}$. Reference: RFC 3339: Date and Time on the Internet: Timestamps.
EquipmentInstance::serialNumber	Vendor SN [Address A0h, Bytes 68-83] The vendor serial number (vendor SN) is a 16 character field that contains ASCII characters, left-aligned and padded on the right with ASCII spaces (20h), defining the Vendor's serial number for the transceiver. A value of all zero in the 16-byte field indicates that the Vendor of the SFP did not specify the vendor SN on the EEPROM.
PhysicalProperties::temperature	Current temperature (in degree Celsius) inside the transceiver.

For ODUs and Chassis, this shall include at least the following information:

Attribute Name in Core IM	To be filled with following information
ManufacturerProperties::manufacturerIdentifier	Describes the IEEE Company identifier of the vendor of the ODU. Might be left blank, if the vendor has no IEEE Company identifier.
ManufacturerProperties::manufacturerName	Vendor's name. Might be left blank, if manufacturerIdentifier is filled.
EquipmentType::partTypeIdentifier	Uniquely identifies the type of ODU in the vendor's product lists.
EquipmentType::version	Identifies the revision number of the type of hardware of the ODU.
EquipmentInstance::manufactureDate	Vendor's date code for the ODU. The date type is a profile of the ISO 8601 standard for representation of dates using the Gregorian calendar. The profile is defined by the full-date production in Section 5.6 of RFC 3339. Values shall follow the pattern: ^\\d{4}-\\d{2}-\\d{2}$. Reference: RFC 3339: Date and Time on the Internet: Timestamps.
EquipmentInstance::serialNumber	Vendor's serial number of the ODU (0 = not applicable).
PhysicalProperties::temperature	Current temperature (in degree Celsius) inside the ODU.

ExpectedEquipment

(32) If the ExpectedEquipment object has been automatically instantiated by the device, exactly the following information shall be copied from the ActualEquipment object:

Attribute Name in Core IM
ManufacturerProperties::manufacturerIdentifier
ManufacturerProperties::manufacturerName

Attribute Name in Core IM
EquipmentType::partTypeIdentifier
EquipmentType::version

If an ExpectedEquipment object is instantiated via the management interface, it is up to the operator to decide about which fields to be filled.

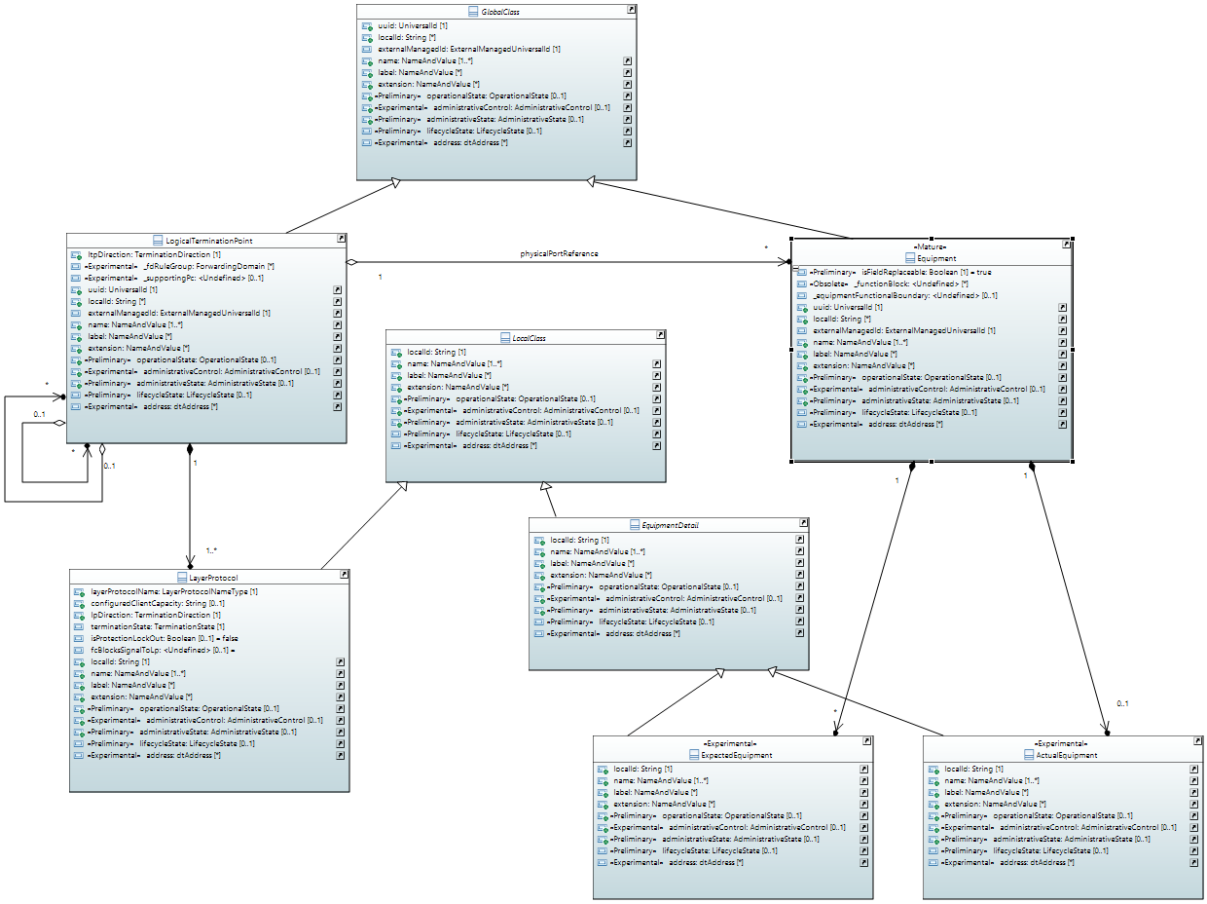
It is strongly recommended to define at least the following minimum set of information:

Attribute Name in Core IM
ManufacturerProperties::manufacturerIdentifier
ManufacturerProperties::manufacturerName
EquipmentType::partTypeIdentifier

Capability

(33) For being able to automatically instantiate LogicalTerminationPoints including their technology specifically augmented LayerProtocol objects, Capability information according to the respective technology specific interface definitions have to be stored on the device for those components, which were provided by the vendor during release of the management interface.

10 Operational States



ActualEquipment

(34) The value of the ActualEquipment::operationalState attribute shall be disabled, except the currently plugged unit is fully operational.

ExpectedEquipment

(35) The value of the ExpectedEquipment::operationalState attribute shall be disabled, except there is an ActualEquipment object instantiated (regardless of its operational state), which has the exact same values in all those fields, which are not empty in the ExpectedEquipment object. (Empty fields of the ExpectedEquipment object shall be interpreted as wildcards.)

Equipment

(36) The value of the Equipment::operationalState attribute can only be enabled, if the ActualEquipment::operationalState attribute of the associated ActualEquipment object and the ExpectedEquipment::operationalState attribute of at least one of the associated ExpectedEquipment objects is enabled.

LogicalTerminationPoint

(37) An interface must not operate and the value of its LogicalTerminationPoint::operationalState attribute must be disabled, if the value of the Equipment::operationalState attribute of the associated Equipment is disabled.

Technology specific operationalState attributes

(38) The values of all attributes, which are describing some kind of operational status inside some technology specific augmentation of the LayerProtocol class (e.g. transceiverIsOnList, autoPmdNegotiationIsOn, performanceMonitoringIsOn …), should be ignored, if the value of the LogicalTerminationPoint::operationalState attribute is disabled.

Remark

Because the operational state of the LogicalTerminationPoint depends on the operational state of the ExpectedEquipment…

- … extensive use of wildcards allows plugging a wider range of SFPs without updating the list of ExpectedEquipment objects, but increases the risk of mismatching LayerProtocol instances (Capability or Configuration values are mismatching the actual hardware characteristics).
- … reduced use of wildcards, leads to constant need for updating the list of ExpectedEquipment objects on the individual Device, because otherwise Interfaces will not get operational again after replacing broken SFPs e.g. with more recent SFP versions.

So the extend of leaving fields of the ExpectedEquipment objects blank, is a trade-off between maintenance effort and operational stability. Because the operationalState of just one of the listed ExpectedEquipment objects needs to be enabled for activating the interface, the poorest defined ExpectedEquipment object is most relevant.

Annex

Non-normative Annex showing some examples for expressing sequence of Equipment, Holder or Connector.

Sequencing connectors and holders

All objects in the *connector* and *contained-holder* collections of every object in the *equipment* collection shall be identifiable via a sequenceId.

All instances of Holder in the Equipment::containedHolder attribute shall be identifiable by a unique 'sequenceId'. The 'sequence'Id' should be unique for the list of Holder objects of the Equipment object. Gaps in the sequence would be legal.

All instances of Connector in the Equipment::connector attribute shall be identifiable by a locally unique 'sequenceId'. The 'sequence'Id' should be unique for the list of Connector objects of the Equipment object. Gaps in the sequence would be legal.

The sequenceId values of Holders and Connectors can overlap even of the same Equipment object.

Connectors

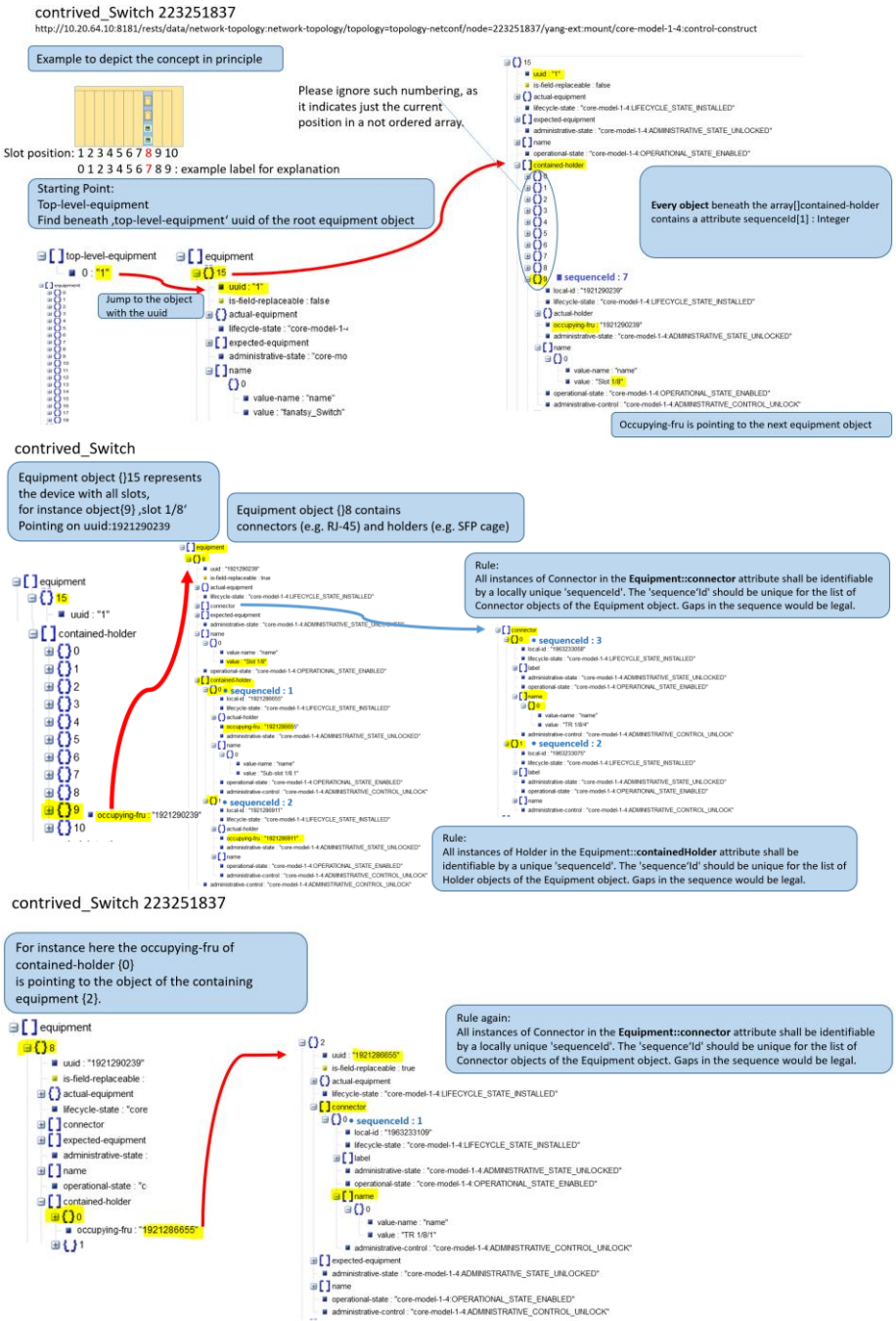
The sequence-id could be listed under “equipment[]/<equipment-object>/connector[]/<connector-object>/as:

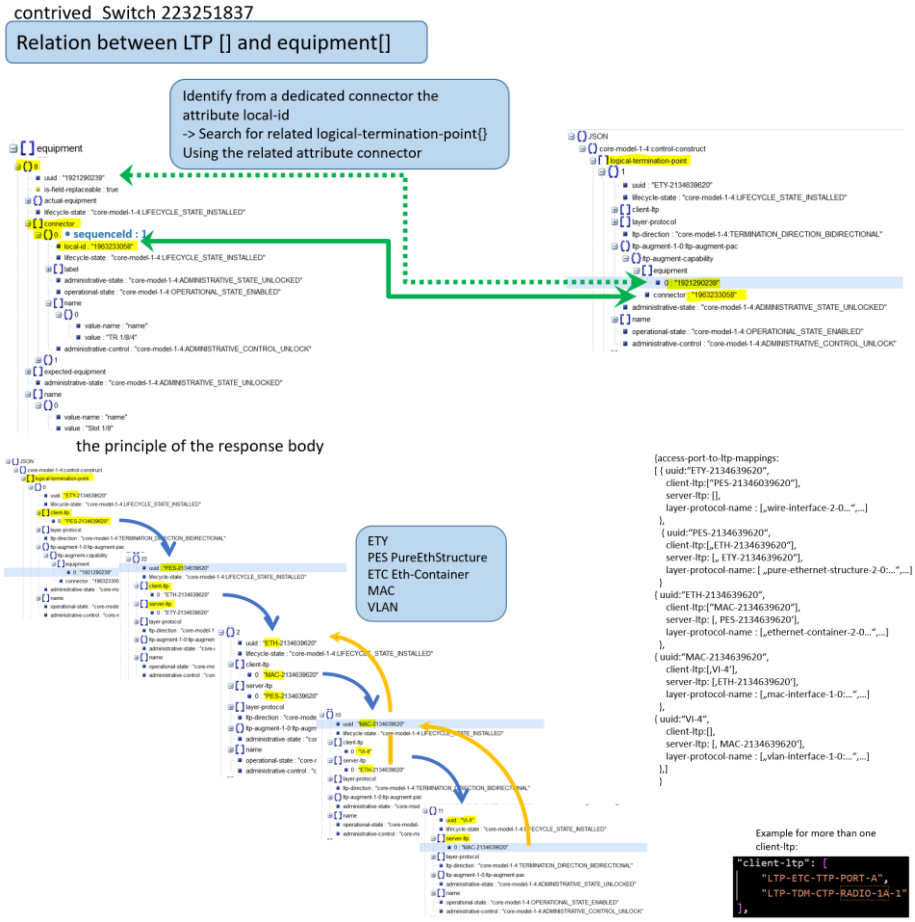
- sequenceId[1] : Integer = -1

Holders

The sequence-id could be listed under “equipment[]/<equipment-object>/contained-holder[]/<holder-object>/” as:

- sequenceId[1] : Integer = -1





Example for more than one client-ltp:

```
"client-ltp": [
    "LTP-ETC-TTP-PORT-A",
    "LTP-TDM-CTP-RADIO-1A-1"
],
```

- end of document -