TELECOM INFRA PROJECT

MUST

Use Cases and Technical Requirements for Wireless Backhaul SDN Domain Controller & Network Equipment

TIP OOPT PG - Version: 2.0

Date Aug, 2022

Document version: v2.0

Authors:

- **Eduardo Yusta**
  - Telefonica CTIO
  - eduardo.yustapadilla@telefonica.com
- **Dimitrios Siomos**
  - Principal Network Expert - Deutsche Telekom Group
  - dsiomos@cosmote.gr
- **Arturo Mayoral López de Lerma**
  - Connectivity Technologies & Ecosystems Manager - Meta Connectivity
  - adelerma@fb.com
- **Ferrari, Gabriele,**
  - Senior Engineer - Vodafone Italy
  - Gabriele.Ferrari@vodafone.com
- **Solomzi Mnyaka**
  - MTN Group Technology-Transport – Wireless Technology
  - Solomzi.mnyaka@mtn.com
- **Marko Liuhanen**
  - Technology Manager - Telia
  - marko.liuhanen@teliacompany.com

Change Tracking

Date	Revision	Author(s)	Comment
25/03/2021	V1.0	All	Consolidated version 1.0
25/08/2022	V2.0	All	Including new set of use cases and technical requirements for the Northbound interface.

Table of Contents

Table of Figures

1 Introduction

Currently wireless networks are operated through vendor Network Management Systems (NMS) using proprietary interfaces. Operations and configuration/maintenance activities are performed manually and statically. This adds complexity toward the higher-level applications and OSS systems that need to manage many proprietary interfaces. In the case of Microwave (MW) networks, these scale upward to non-reasonable levels due to technical complexity.

Integration and management complexity can often be a showstopper that blocks (barrier to) introduction of a new vendor solutions that could otherwise be technically relevant for the network. Furthermore, integration costs and efforts caused by diversity in both NMSes and the installed base prevents operators from using advanced applications that could provide more sophisticated features such as power management or multi-layer coordination.

The main driver for introducing the MW SDN domain controller is to reduce such complexity and enable more simplified integrations and operations of diverse platforms. The proposed architecture model is aligned with the view of the multi-vendor SDN domain controller, introducing its domain controller as a vendor-agnostic configurator of the MW network. It focuses on simplifying configuration by leaving the service definition to upper layer applications.

While MW deployment can be applied with some regional criteria (linked in many cases to simplified network operation and more efficient management of aspects such as repair time and spare part oversight), wireless transport networks are often deployed on a point-to-point basis. Therefore it's common to have more than one vendor within any region (even small ones), with a vendor map that changes more dynamically than in other network planes.

Beyond the attempt to address these problems, enabling SDN in the MW transport domain, and bring standard and open APIs across the different functional blocks may enable also new applications to be developed on top these standard APIs.

1.1 Goal

The purpose of this document is to define the use cases, standard interfaces and architectural guidelines of the Wireless Backhaul domain in charge of the management and control of MW transport network.

The content developed in this document follows a building block approach, with a prioritised set of Use Cases designed from the South Bound Interface (SBI) perspective of the target architecture. These use cases are pragmatic and support the progressive simplification and acceleration of the technology interoperability maturity and deployment. Further work will continue to expand on other SBI and North Bound Interface (NBI) Use Cases in the near future.

The target audience of this document are network service providers planning and engineering teams, who are in charge of the definition of their RFQs or RFI documents, who can benefit from this common definition of SDN enabled use cases; technology suppliers account managers and Product Line managers who can benefit for an early deep understanding of the shared operator interests and technical implementations proposed in this document; and also Standardization Bodies which can drive requirements included in this document to develop further their standards to fulfil the expectations raised by the network operators signing the

present document.

1.2 Reference architecture

The preferred open transport SDN architecture within a single operator network is based on a hierarchical controller (HCO) along with several technology-specific SDN controllers (COs). The high-level hierarchical SDN architecture and the relationship with higher layers (such OSS) defined in TIP OOPT MUST Sub-group is depicted in Figure 1.

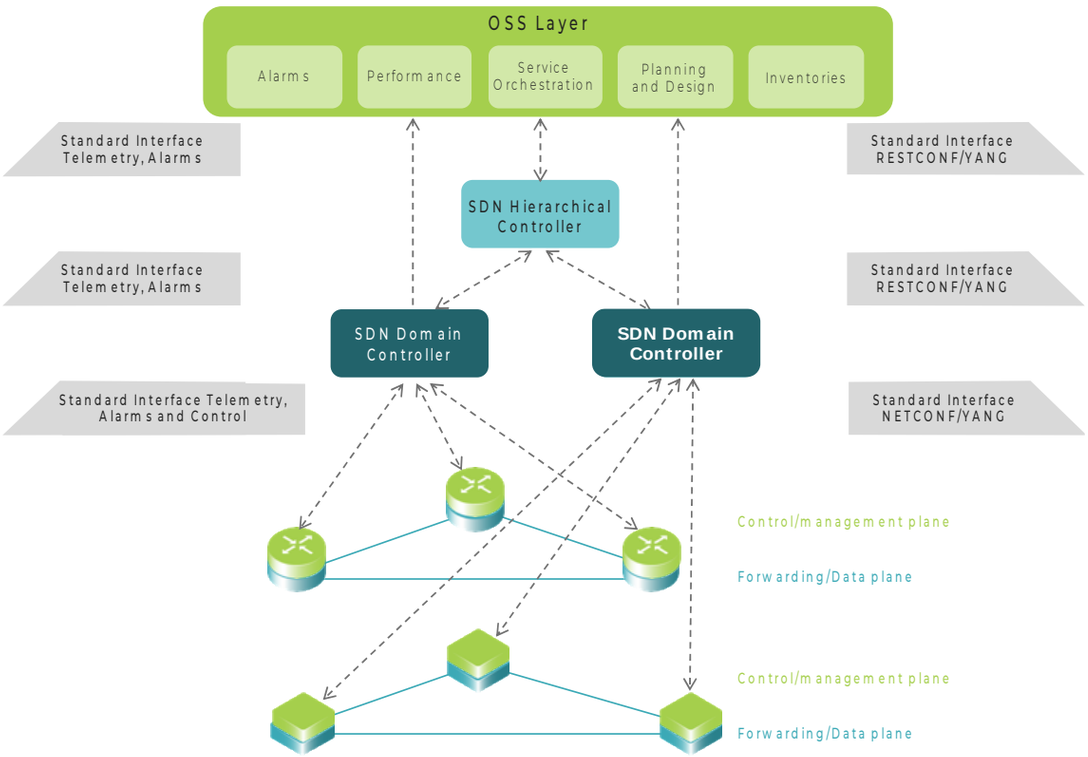


Figure 1: MUST general Hierarchical SDN architecture

If we focus on the MW transport domain, a single multi-vendor SDN Controller is expected to provide vendor agnostic network management functions over a multi-vendor network deployment. The details of the inner controller architecture and its engagement with the rest of MUST SDN reference architecture is illustrated in Figure 2.

The MW transport SDN Domain controller, shall be able to provide, topology discovery functions, service and connectivity provisioning, network activation and configuration, fault management, performance analysis and network inventory. These management areas are the scope to be fulfilled by the technical definition of use cases, which will progressively cover all areas described, by introducing standard and open APIs as the core of their technical requirement guidelines for their fulfilment.

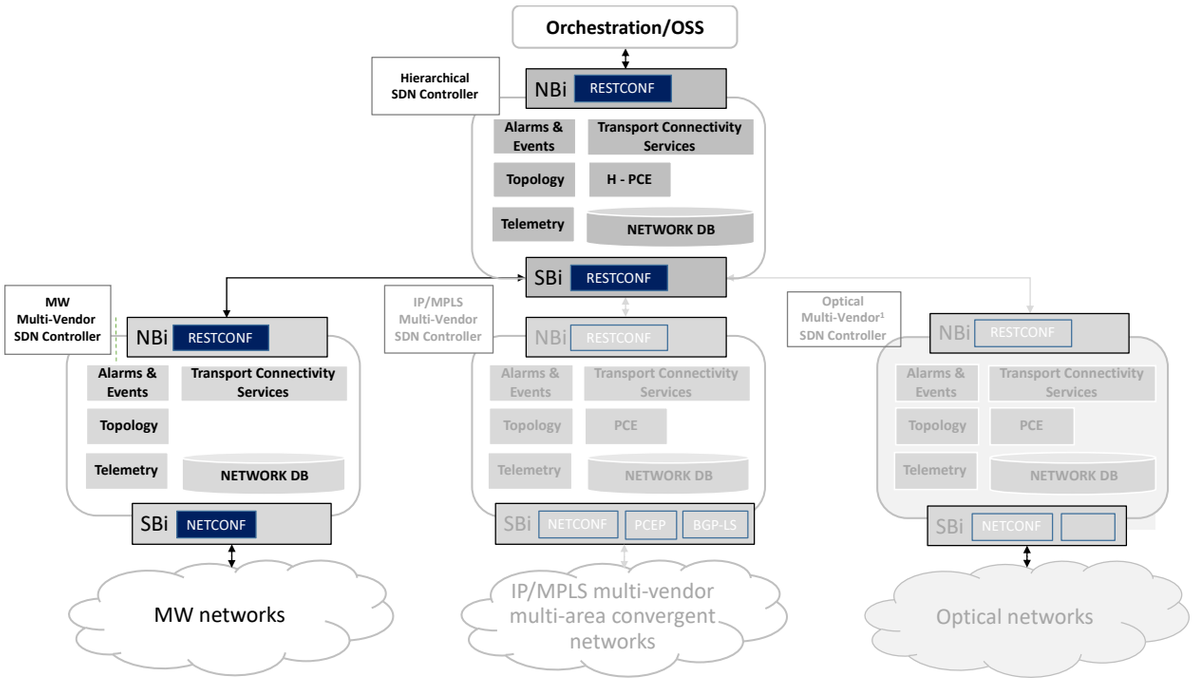


Figure 2: MUST SDN Transport architecture with the focus on the MW domain

2 Use Cases

This section includes the different categories of use cases defined by MUST subgroup. Each document use case includes a common ID, Summary, Benefits and motivation and Detail description sections which describe the goals and common technical guidelines of the target use case.

Moreover, up to two possible implementations have been proposed for most of use cases based on ONF and IETF/IEEE works. Each use case is self-contained, including references to the target data models and standard documents, and also a set of minimum requirements agreed for each proposed implementation.

2.1 Network & Services Auto-Discovery

2.1.1 Use Case 1.1 - Hardware inventory auto-discovery

Use Case	Hardware inventory auto-discovery
Id	Use Case 1.1
Summary	Use case focuses on the retrieval of Network Elements (NE) physical information to build a consolidated hardware inventory at network level. Individual elements composing the NE and their relevant parameters (like serial numbers, hardware model types, etc.), hierarchical relations between elements (slots and hosted cards) and details related to available physical connectors are part of the scope of the use case.
Benefits and Motivation	Main motivation is enabling the creation of a single harmonized hardware inventory across a MW/mmWave multi-vendor network, including the different types of equipment under operation. Keeping an up-to-date hardware inventory is key to support multiple network processes from planning to operation, as for example identifying available resources and optimizing network expansion, network wide efficient planning of hardware upgrades, or quick identification of elements that may suppose risks according to operational information. Abstraction of vendor and technology specific aspects reduces the need of support of specialized teams to the rest of groups managing the information which optimizes tasks in terms of effort and time.

Detailed description	The SDN agnostic controller should be able to retrieve complete and harmonised inventory data from any NE within the network domain. MW networks include radio links and HW of different type (e.g., split systems, full outdoor systems), mixing compact and modular (slotted) equipment models, each including different HW modules and connectors, supporting the available interfaces.
	
	To build a consolidated hardware inventory, the main blocks of information identified as a target are:
	• ID fields: ID / naming fields shall identify uniquely the network element within the full network domain and provide extra information at application level.
	• High-level descriptive information of the NE. Typical parameters describing the NE, relevant for inventory purposes are:
	  ○ Network element equipment type
	  ○ IP address
	• Root HW element A complete list of the HW elements within the NE, with specific ID/naming/state fields to identify them as well as complete descriptive information, depending on the detail required at application level. e.g:

	○ HW element model type ○ Manufacturing vendor ○ Product code, part number ○ Serial number and HW version ○ HW detailed description and other relevant HW properties (swappability) • Listing of connectors available at the different elements of the NE, with ID and descriptive information. It Is typically of interest to identify supported interfaces by each HW element or connector in order to cross reference hardware with supported carriers, physical interface capabilities or performance aspects. • Slots available to host other elements within each HW element, with ID and descriptive information. • Dependency relations between HW elements within the NE, to identify HW element hierarchy and occupied and empty slots in modular hardware. • Administrative and operational states of the NE and the composing elements may also be included to check the status of the HW. Optionally, it may be of interest to have not only the possibility of identifying installed hardware within a NE, but also the view of planned equipment.
Proposed implementation ONF -Data models	Implementation of this use case relies almost exclusively in the ONF CIM classes, especially in the equipment class. General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3. Implementations shall as well be compliant with the related open backhaul transmitter Equipment specification v1.0 which is available here. Parameters for the use case implementation with the ONF MW model will be available at two levels: • First, general ID/naming/state fields as well as general NE descriptive fields shall be directly available at control Construct (root level of the NE in ONF modeling) • Rest of the use case parameters shall be available at the equipment list of the control Construct, in different parts of the equipment tree, which will be detailed below. `+--rw control-construct!` `  +--…` `  +--rw equipment* [uuid]` `	+--[Global naming/ID/state fields]` `	+--ro is-field-replaceable? Boolean` `	+--rw expected-equipment* [local-id]` `		+--[Local naming/ID/state fields]` `		+--swappability` `		+--manufactured-thing` `		+--…` `	+--rw actual-equipment` `		+--[Local naming/ID/state fields]` `		+--swappability` `		+--manufactured-thing` `		+--…` `	+--rw connector* [local-id]` `		+--[Local naming/ID/state fields]` `		+--…` `	+--rw contained-holder* [local-id]` `		+--[Local naming/ID/state fields]` `		+--rw occupying-fru? -> /control-construct/equipment/uuid` `		+--…` At controlConstruct level:

USE CASE INFORMATION BLOCK	MODEL CLASS/PARAMETER	REQUIRED PATH	COMMENTS
	control-construct	**/core-model:control-construct**	
NE ID	uuid	/core-model:control-construct/uuid	*Unique identified of the NE*
NE NAMING FIELDS	name[value-name]	/core-model:control-construct/name	*List of value-name & name pairs. More than one element may be specified with different value-names.* *As example, one pair value-name="externalLabel" & name="XXXX" matching the external label implemented by the operator for the equipment*
NE DESCRIPTIVE INFORMATION	extension[value-name]	*/core-model:control-construct/extension*	*List of value-name & name pairs. More than one element may be specified with different value-names.* *As examples:* *one pair value-name="neIpV4Address" & name="xxx.yyy.zzz.aaa" showing the NE IP* *one pair value-name="neType" & name="aaaaa" showing the model type of the equipment*
	top-level-equipment	/core-model:control-construct/top-level-equipment	*pointer to the root hw element within the NE* *leafref path--> "/core-model:control-construct/core-model:equipment/core-model:uuid"*
NE STATE FIELDS	administrative-state	/core-model:control-construct/administrative-state	
	operational-state	/core-model:control-construct/operational-state	

The equipment list shall include an instance per each specific HW element within the node. Each instance will include specific ID/naming/state fields, as well as an *expected-equipment* list, an *actual-equipment* container (in case the HW element is plugged and activated), Within the expected or actual-equipment, several (same in both) containers will include most of the relevant parameters for the use case implementation, mainly the *manufactured-thing and the swappability* ones. The next table shows the required parameters.

USE CASE INFORMATION BLOCK	MODEL CLASS/PARAMETER	REQUIRED PATH	COMMENTS
	equipment[uuid]	**/core-model:control-construct/equipment**	
HW ELEMENT ID FIELDS	uuid	**PATH_equipment**/uuid	
HW ELEMENT NAMING FIELDS	name[value-name]	**PATH_equipment**/name	*List of value-name & name pairs. Several entries may be specified with different value-names.* *E.g: value-name="equipmentLabel" & name="aaaaa" matching the typical labels on the outside of the NE identifying elements*
HW ELEMENT DETAILS	is-field-replaceable	**PATH_equipment**/is-field-replaceable	
	is-hot-swappable	**PATH_equipment**/actual-equipment/swappability/is-hot-swappable	

	manufacturer-name	**PATH_equipment**/ actual-equipment/manufactured-thing/ manufacturer-properties/manufacturer-name	*Same parameters also available and required each instance of the expected-equipment list* */core-model:control-construct/equipment/expected-equipment/*
	type-name	**PATH_equipment**/actual-equipment/manufactured-thing/ equipment-type/type-name	
	description	**PATH_equipment/**actual-equipment/manufactured-thing/ equipment-type/description	
	model-identifier	**PATH_equipment**/actual-equipment/manufactured-thing/ equipment-type/model-identifier	
	part-type-identifier	**PATH_equipment**/actual-equipment/manufactured-thing/ equipment-type/part-type-identifier	
	version	**PATH_equipment**/actual-equipment/manufactured-thing/ equipment-type/version	
	serial-number	**PATH_equipment/**actual-equipment/manufactured-thing/ equipment-instance/serial-number	
	manufacture-date	**PATH_equipment**/actual-equipment/manufactured-thing/ equipment-instance/ manufacture-date	
	asset-instance-identifier	**PATH_equipment**/actual-equipment/manufactured-thing/equipment-instance/ asset-instance-identifier	
HW ELEMENT STATE FIELDS	administrative-state	**PATH_equipment**/actual-equipment/ administrative-state	
	operational-state	**PATH_equipment**/actual-equipment/ operational-state	

Additional containers including parameters related to other properties of the HW elements like physical dimensions, thermal/power ratings, etc. are available and can be considered as an option for to expand the use case scope.

HW elements that contain slots to host other equipment elements within the NE shall include for each equipment instance the *contained-holder* list. Of special interest, aside the holder ID/naming/state fields will be the *occupying-fru* field, which serves to model relations between HW elements in the node, pointing to uuid of the hosted element.

USE CASE INFORMATION BLOCK	*MODEL CLASS/PARAMETER*	*REQUIRED PATH*	*COMMENTS*
	contained-holder[local-id]	**/core-model:control-construct/ equipment/contained-holder**	
HOLDER ID FIELDS	local-id	**PATH_contained-holder**/local-id	
HOLDER NAMING FIELDS	name[value-name]	**PATH_contained-holder**/name	*List of value-name & name pairs. More than one element may be specified with different value-names. As example: one pair value-name="slotLabel" & name="<string>" showing the slot ID label on the outside of the NE identifying elements*
EQUIPMENT DEPENDENCY RELATIONS	occupying-fru	**PATH_contained-holder**/occupying-fru	*shall point to the uuid of the equipment instance placed in this holder or slot. leafref path--> "/core-model:control-construct/ equipment/uuid"*

HOLDER STATE FIELDS	administrative-state	**PATH_contained-holder**/administrative-state	
	operational-state	**PATH_contained-holder**/operational-state	

Additional containers to extend with further details the holder information are available as part of the ONF CIM contained-holder definition. They may optionally be used to expand the use case further.
In case the HW element corresponding to any equipment instance includes physical connectors the *connector* list defined within equipment class shall be available, including as a minimum:

USE CASE INFORMATION BLOCK	*MODEL CLASS/PARAMETER*	*REQUIRED PATH*	*COMMENTS*
	connector[local-id]	**/core-model:control-construct/equipment/**	
CONNECTOR ID FIELDS	local-id	**PATH_equipment**/local-id	
CONNECTOR NAMING FIELDS	name[value-name]	**PATH_equipment** /name	*List of value-name & name pairs. More than one element may be specified with different value-names.* *As example: one pair value-name="connectorLabel" & name="<string>" showing the connetcor ID label typically on the outside of the device*
CONNECTOR DETAILS	connector-type	**PATH_equipment**/connector-type	
CONNECTOR STATE FIELDS	administrative-state	**PATH_equipment**/administrative-state	
	operational-state	**PATH_equipment**/connector/operational-state	

Additional parameters are available in the connector class, that may be optionally used to expand the use case scope if needed.
In many cases, it becomes relevant to link modelled interfaces with the physical HW supporting them. In this cases, additionally to the equipment class, implementations need to support both the logcalTerminationPoint list at the controlConstruct (mandatory for most of the use cases, and its TR-532 MW model PAC ltpAugment (see section 3 for details). This augment to the logicalTerminationPoint class includes pointers from interface termination points to the physical equipment(s) and connector supporting the interface.

USE CASE INFORMATION BLOCK	*MODEL CLASS/PARAMETER*	*REQUIRED PATH*	*COMMENTS*
	ltp-augment-pac	**/core-model:control-construct/logical-termination-point/ltp-augment:ltp-augment-pac**	
INTERFACE REFERENCES TO PHYSICAL EQUIPMENT	equipment	**PATH_ltp-augment-pac/**ltp-augment-capability/equipment	*list of pointers to the uuids of equipment instances supporting the interface* *paths-> "/core-model:control-construct/equipment/uuid";*
	connector	**PATH_ltp-augment-pac/**ltp-augment-capability/connector	*pointer to the local-id of the interface physical connector* *leafref path-> "/core-model:control-construct/equipment/connector/local-id"*

For the use case implementation, in terms of workflow, the controller may retrieve the

information required for the use case in a single step for each of the NEs under control. The network element will be addressed by the controller using its uuid available at control-construct level.

In case the reference between HW elements and connectors and interfaces (via logical-termination-point list) is required to complement the HW inventory, and it is not available at the controller linked to other use cases that will require the extraction of this list and its fields, it can also be considered in a single step or as part as a two-step approach, extracting separately the equipment and the logical-termination-point list, in the latter case including the pointers within the ltp-augment container.

Proposed implementation IETF/IEEE - Data models

The IETF RFC 8348 “A YANG Data Model for Hardware Management” defines a YANG data model for the management of hardware on a single server. The YANG module "ietf-hardware", defined in the IETF RFC 8348, has the following structure:

```
module: ietf-hardware
    +--rw hardware
         +--ro last-change? yang:date-and-time
         +--rw component* [name]
              +-- <parameters>
    notifications:
              +--- <parameters>
```

It should be noted that the YANG module “ietf-hardware” also includes parameters relevant to software and firmware versions.
The Data Model version is provided by the Revision Statement included in the ietf-hardware (IETF RFC 8348).

Per Hardware Instance		
+--rw hardware **+--ro last-change? yang:date-and-time** **+--rw component* [name]**		
Parameter	**Type / Sub-parameter**	**Description / Application (indicative)**
+--rw name	string	NE name
+--rw class	identityref	
+--ro physical-index?	int32 {entity-mib}?	Model number, Chassis id
+--ro description?	string	NE description
+--rw parent?	-> ../../component/name	
+--rw parent-rel-pos?	int32	
+--ro contains-child* ->	../../component/name	
+--ro hardware-rev?	string	HW version
+--ro firmware-rev? string	string	FW version
+--ro software-rev? string	string	SW version
+--ro serial-num? string	string	Serial number
+--ro mfg-name? string	string	
+--ro model-name? string	string	Model version
+--rw alias? string	string	
+--rw asset-id?	string	
+--ro is-fru?	boolean	
+--ro mfg-date?	yang:date-and-time	
+--rw uri*	inet:uri	
+--ro uuid?	yang:uuid	Hardware id
+--rw state {hardware-state}?	| +--ro state-last-changed? yang:date-and-time | +--rw admin-state? admin-state | +--ro oper-state? oper-state | +--ro usage-state? usage-state | +--ro alarm-state? alarm-state | +--ro standby-state? standby-state	

+--ro sensor-data {hardware-sensor}?	+--ro value? sensor-value +--ro value-type? sensor-value-type +--ro value-scale? sensor-value-scale +--ro value-precision? sensor-value-precision +--ro oper-status? sensor-status +--ro units-display? string +--ro value-timestamp? yang:date-and-time +--ro value-update-rate? uint32	
Notifications:		
Parameter	**Type / Sub-parameter**	**Description / Application (indicative)**
+---n hardware-state-change		
+---n hardware-state-oper-enabled {hardware-state}?	| +--ro name? -> /hardware/component/name | +--ro admin-state? -> /hardware/component/state/admin-state | +--ro alarm-state? -> /hardware/component/state/alarm-state	
+---n hardware-state-oper-disabled {hardware-state}?	+--ro name? -> /hardware/component/name +--ro admin-state? -> /hardware/component/state/admin-state +--ro alarm-state? -> /hardware/component/state/alarm-state	

2.1.2 Use Case 1.2 - Firmware inventory auto-discovery

Use Case	Firmware inventory auto-discovery
Id	Use Case 1.2
Summary	Use case focuses on the retrieval of microwave network element SW/FW information, complementing the hardware inventories and enabling a clear identification of the firmware packages and individual components running and available in the different equipment modules of the NE, as well the firmware module dependencies.
Benefits and Motivation	Main motivation is enabling the creation of a single harmonized SW/FW inventory across a MW/mmWave multi-vendor network, covering the different types of equipment under operation. Updated network element SW/FW inventories are key for example to validate that network elements are running SW/FW which has been certified as part of the typical testing processes run by mobile operator technology departments. It is also essential to identify quickly parts of the plant under operation that may be at risk in case of bugs or issues with specific equipment and its SW/FW that need to be updated as soon as possible. Additionally, in combination with the hardware inventory and the network topology view, maintaining a consolidated SW/FW inventory is also a basic need to plan efficiently network upgrade processes as new SW/FW releases are developed by equipment suppliers and tested and certified by operators. In combination with standard SBI-based firmware management use cases, there is a large potential to get operational benefits linked to network upgrades automation.
Detailed description	MW/ mmWave NEs are composed of different HW elements, subsystems and boards (internal / external) which will be running specific SW/FW modules that will dictate its behaviour and characteristics. Network operation and planning require to identify SW/FW running at the NE equipment modules. As a minimum scope, the use case considers retrieving basic details characterizing the FW/SW modules running on each HW element. For this, typically, the key elements being: • ID/naming field(s) of the NE to identify uniquely within the domain. Additional

	descriptive fields (type of NE, vendor, equipment names, etc.) may be added also to the FW use case, similarly to the HW inventory use case. • ID/naming fields of each HW module that includes SW/FW • SW/FW module name and version on each HW element in the NE However, in MW/mmWave equipment, it is typical that SW/FW bundles or packages are used as a way to download necessary FW to the NE, having then each specific piece of individual FW distributed internally to all the HW elements within the node before their activation. It is also common that NEs have the capability to store more than one package in dedicated benches (having running / standby FW), that also to enable FW upgrade operations. Packages may also be dedicated to specific elements within the node (e.g., ODU / IDU). Therefore, learning the full SW/FW hierarchy (packages and individual images within them) and retrieving the status of activation of the different modules (available/standby/running) becomes also highly desirable. This is also required as the base for FW management use cases. To cover this additional scope, retrieval of additional information is required: • ID/naming fields of each FW/SW element in the NE. • Class of SW/FW modules: to distinguish packages from individual FW modules, also helpful as descriptive information about the type of SW/FW module (FPGA code, BOOT/BIOS, etc.) • Dependency relations between SW/FW: mainly to identify individual modules or images that are part of a package. Typically, available as pointers from parent (package) to children (images). • Potential for activation for the different SW/FW modules: to identify those can be managed individually for activation and de-activation • Current status of activation of each SW/FW module
Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes (equipment) and the support of a specific TR-532 MW technology specific augment to the CIM, firmware_1.0.0, with resources (papyrus UML, overview and documentation) available here and latest yang module firmware-1-0 here . General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3. The firmware PAC attaches to the control-construct, augmenting it with a firmware-collection. SW/FW inventory relies on the parameters available at the firmware-component-list within this object. `module: firmware-1-0` `  augment /core-model:control-construct:` `    +--rw firmware-collection` `       +--ro firmware-component-list* [local-id]` `	+--ro local-id` `	+--ro name* [value-name]` `		+--ro value-name string` `		+--ro value? string` `	+--ro firmware-component-pac` `		+--ro firmware-component-capability` `			+--…` `		+--ro firmware-component-status` `			+--…` `	+--ro subordinate-firmware-component-list* -> /core-model:control-construct/ firmware:firmware-collection/firmware-component-list/local-id` ` +--…` The next tables shows the required parameters for the use case implementation, covering the full

scope as described in the detailed description section. First, details from the NE from which the FW information is retrieved.

USE CASE INFORMATION BLOCK	MODEL CLASS/PARAMETER	REQUIRED PATH	COMMENTS
	control-construct	/core-model:control-construct	
NE ID	uuid	/core-model:control-construct/uuid	Unique identified of the NE
NE NAMING FIELDS (OPTIONAL)	name[value-name]	/core-model:control-construct/name	List of value-name & name pairs. More than one element may be specified with different value-names. As example, one pair value-name="externalLabel" & name="XXXX" matching the external label implemented by the operator for the equipment
NE DESCRIPTIVE INFORMATION (OPTIONAL)	extension[value-name]	/core-model:control-construct/extension	List of value-name & name pairs. More than one element may be specified with different value-names. As examples: one pair value-name="neIpV4Address" & name="xxx.yyy.zzz.aaa" showing the NE IP one pair value-name="neType" & name="aaaaa" showing the model type of the equipment
	top-level-equipment	/core-model:control-construct/top-level-equipment	pointer to the root hw element within the NE leafref path--> "/core-model:control-construct/core-model:equipment/core-model:uuid"
NE STATE FIELDS (OPTIONAL)	administrative-state	/core-model:control-construct/administrative-state	
	operational-state	/core-model:control-construct/operational-state	

FW and HW inventories are typically complementary, so the detailed information may already be available in the controller in the HW inventory. In this case only the NE uuid is mandatory, avoiding duplication.

The firmware-collection must be available at the control-construct level, including a firmware-component-list. To achieve the use case scope, parameters required for each list instance are:

USE CASE INFORMATION BLOCK	MODEL CLASS/PARAMETER	REQUIRED PATH	COMMENTS
	firmware-component-list*[local-id]	/core-model:control-construct/firmware:firmware-	List including all the firmware modules and related parameters of the NE

		collection/firmware-component-list	
SW/FW ID FIELDS	local-id	PATH_firmware-component-list/local-id	internal ID field, uniquely identifying the NE SW/FW modules within the firmware-collection
DEPENDENCY RELATIONS BETWEEN SW/FW MODULES	subordinate-firmware-component-list	PATH_firmware-component-list/subordinate-firmware-component-list	Pointer to the local-id of the subordinate firmware (e.g packages will include a list of pointers to all the individual child FW images) leafref path--> /core-model:control-construct/firmware:firmware-collection/firmware-component-list/local-id
	firmware-component-capability	/core-model:control-construct/firmware:firmware-collection/firmware-component-list/firmware-component-pac/firmware-component-capability	Includes all the firmware related parameters of each FW component in the NE
SW/FW MODULE NAME	firmware-component-name	PATH_firmware-component-capability/firmware-component-name	Vendor SW/FW name, identifying the module
SW/FW MODULE VERSION	firmware-component-version	PATH_firmware-component-capability/firmware-component-version	Vendor SW/FW version, identifying the module
SW/FW MODULE CLASS/TYPE	firmware-component-class	PATH_firmware-component-capability/firmware-component-class	Identifies the type of SW/FW element (PACKAGE / FIRMWARE / BIOS_CODE / DRIVER ...)
TYPES OF ELEMENTS COMPATIBLE WITH THE SW/FW	related-kinds-of-equipment-list	PATH_firmware-component-capability/related-kinds-of-equipment-list	List of strings to identify the types of equipment compatible with the HW. Shall match the vendor model IDs types in /control-construct/equipment/ actual-equipment/manufactured-thing/ equipment-type/model-identifier, and might be empty for FW packages.
SW/FW POTENTIAL FOR ACTIVATION	individual-activation-is-avail	PATH_firmware-component-capability/individual-activation-is-avail	Represents the possibility of activating individually each SW/FW module, relevant for management purposes
	firmware-component-status	/core-model:control-construct/firmware:firmware-collection/firmware-component-list/firmware-component-pac/firmware-component-status	
ID OF EQUIPMENT INCLUDING THE FW	is-active-on-equipment-list	PATH_firmware-component-status/is-active-on-equipment-list	Points to the uuids of the equipment instances within the NE that include the SW/FW leafref path--> /core-model:control-construct/equipment/uuid
SW/FW STATUS OF ACTIVATION	firmware-component-status	PATH_firmware-component-status/firmware-component-status	Represents the current status of each SW/FW module, relevant for management purposes. (ACTIVE, STAND_BY, LIKE_SUPERIOR_FIRMWARE_COMPONENT,...)

<table>
<tr><td></td><td>With these fields it is possible to retrieve from the NE all the SW/FW modules, identify packages and compose the FW structure of the NE building a detailed inventory, correlate information with the HW inventory and provide information to the SDN domain controller or a specific app running on top for the management and upgrade of the NE FW.

In terms of workflow for the use case implementation, the controller may retrieve the information required for the use case in a single step for each of the NEs under control. The network element will be addressed by the controller using its uuid available at control-construct level.</td></tr>
<tr><td>Proposed implementation IETF/IEEE -Data models</td><td>As discussed in the Hardware inventory auto-discovery use case above, the YANG module “ietf-hardware”, defined in the IETF RFC 8348 “A YANG Data Model for Hardware Management”, includes parameters relevant to software and firmware versions, as well. These parameters are:
<table>
<tr><td colspan="3">Per Hardware Instance</td></tr>
<tr><td>Parameter</td><td>Type / Sub-parameter</td><td>Description / Application (indicative)</td></tr>
<tr><td>+--ro firmware-rev? string</td><td>string</td><td>FW version</td></tr>
<tr><td>+--ro software-rev? string</td><td>string</td><td>SW version</td></tr>
</table>
Please refer to the Hardware inventory auto-discovery use case (IETF/IEEE section) to have the full list of the parameters defined in the IETF RFC 8348 (per Hardware instance).</td></tr>
</table>

2.1.3 Use Case 1.3 - Carrier inventory (radio parameters)

Use Case	Current Alarms
Id	Use Case 1.3
Summary	Use case focuses on the retrieval of NE parameters related to RF carrier layer to build a detailed carrier inventory that complements high-level topology information.
Benefits and Motivation	An updated carrier inventory is a key element for network planning and evolution tasks. Harmonization also enables to simplify the required technology specific expertise from planners, simplifying processes and resources to carry out recurrent tasks. It also becomes the base for many reconciliation tasks, useful for planners in order to check if the field configuration of the radio carriers and complete links corresponds to link designs and Its planning details, and engineering / operation teams to validate that an installation has been done correctly and the radio link follows the planned setup. This brings relevant benefits as well in terms of operational efficiency and network performance. Through integration of the capabilities supported by the carriers and underlying hardware, the carrier inventory becomes a useful tool as well to validate potential upgrades simplifying many usual tasks within the radio link lifecycle. Longer term, being this a key target for the present use case, this harmonized RF carrier inventory, will enable sophisticated automation-oriented applications across a MW/mmWave multi-vendor network, powered by AI/ML technologies and methods. Combined with PM/status/alarm information, applications like automatic prioritization and planning of network expansions, dynamic carrier reconfigurations (for those features or radio aspects that allow so) will become viable.
Detailed description	To build a consolidated carrier inventory, it should be possible to retrieve detailed information in relation to the carrier capabilities (ranges of values and options supported for the air interface

configuration, as supported by the underlying equipment) and the actual carrier configuration of any of the carriers available in any of the NEs of the MW SDN control domain. The main blocks of information identified as a target are:

- **ID fields**
  - Parameters linked to the identification of the carrier, both internal within the model structure and external, according to operator planning rules (to cross reference with external systems)
    - Interface or termination point ID
    - Carrier ID
- **RF activation**
  - Transmitter / receiver activation
- **Carrier RF parameters**
  - Transmitter and receiver supported frequency ranges
  - Supported duplex frequencies
  - Transmitter and receiver configured frequency
  - Configured duplex frequency
  - Configured transmitted power
  - As an option, if supported, polarization selected for the carrier transmission.
- **Parameters describing the transmission modes available and configured for the carrier:**
  - Mainly, those directly related to the carrier capacity:
    - Physical modulation
    - Channel bandwidth
    - Coding rate
    - Symbol rate reduction factors, when applicable
  - Desirable, those others that provide relevant information around the transmission mode
    - Compatibility with XPIC operation
    - Compatibility with ACM
    - RX Threshold
    - Maximum and minimum power
    - ACM Thresholds
  - Direct availability of a harmonized transmission mode capacity parameter (both for the available transmission modes and for the actual ones (min/max) configured for the carrier may help simplify applications using the standard interface.
- **Carrier typical features**
  - ACM including
    - Availability of the feature
    - Configured Minimum and maximum transmission mode (linked to minimum and maximum physical modulation) when available and activated
  - ATPC, including
    - Availability of the feature and supported ATPC range
    - Tx Power (configured and minimum), Lower and Upper thresholds
  - XPIC availability and configuration value (this may be linked to specific transmission modes supporting cross polar operation)
  - BCA (Band & Carrier Aggregation) with specific capabilities and (Layer 1) LAG configuration
- **TDM resources**
  - Identification of total number of E1s / STM1s transported by the carrier (This, being relevant mainly to identify available traffic for Ethernet services in hybrid configurations)
- **XPIC groups**
  - Identification of the XPIC group that the carrier belongs to, in the case it is part

	of one
Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes (logicalTerminationPoint and layerProtocol lists) and the support of a specific TR-532 PAC, AirInterface_2.0.0, with resources (papyrus UML, overview and documentation) available here and latest yang module air-interface-2-0 here . General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3. LTPs corresponding to an air interface (carrier) will include a layer-protocol instance having a layer-protocol-name=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER. The air-interface pac attaches to the layer-protocol CIM providing a conditional augmentation with the MW specific classes and parameters in case this condition applies. In relation to the use case, the Air Interface PAC includes: • Capability information: ranges of supported values for specific parameters or availability of specific features at interface level. • Configuration information: actual configured values for the interface. `module: core-model-1-4` `  +--rw control-construct!` `     +--...` `     +--rw logical-termination-point* [uuid]` `	+--rw uuid` `	+--rw name* [value-name]` `	+--rw server-ltp* -> /control-construct/logical-termination-point/uuid` `	+--rw client-ltp* -> /control-construct/logical-termination-point/uuid` `	+--…` `	+--rw layer-protocol* [local-id]` `		+--rw local-id` `		+--rw layer-protocol-name? (=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER)` `		+--...` `		+--rw air-interface:air-interface-pac` `		+--ro air-interface:air-interface-capability` `		+--rw air-interface:air-interface-configuration` `		+--…` As explained in the annex 3, network element interfaces are represented with a stack of related (client-server) logical Termination Points that correspond to different layer Protocols (transport layers). To include in the inventory the TDM resources transported over the air interface, two additional TR-532 PACs need to be implemented, to augment LTP/LPs on top of the air interface (client-server relations between layers shall be implemented as indicated in the annex): • HybridMicrowaveStructure_2.0.0 PAC, documented here and with latest available yang here. • TdmContainer_2.0.0 PAC, documented here and with latest available yang here. As with the air interface, the augment is conditional, linked to the layer-protocol-name: • Hybrid structure PAC: LAYER_PROTOCOL_NAME_TYPE_HYBRID_MW_STRUCTURE_LAYER • TDM container PAC: LAYER_PROTOCOL_NAME_TYPE_TDM_CONTAINER_LAYER TDM containers are client layers of the hybrid microwave structure layers, which are clients respectively from the lower air interface layers and servers of the former. From the LTP client-

server relations TDM containers on top of a given air interface / carrier may be identified and their related parameters can be then extracted and added to the inventory.

Finally, if desired for inventory purposes, the inclusion of the XPIC groups to which the carrier belongs requires the support of an additional TR-532 technology specific PAC. In this case: CoChannelProfile_1.0.0 PAC, documented here and with latest available yang here

Unlike the previous ones, these PAC augments the profile-collection of the core model available at control construct level, described in the annex 3. Augmentation is conditional for those profile instances having profile-name=PROFILE_NAME_TYPE_CO_CHANNEL_PROFILE.

Parameters within the Cochannel Profile class allow for the identification of all the XPIC groups within the node. Carriers which are part of the different XPIC groups can be obtained from embedded pointers to air interface LTP uuids of those carriers within the group.

The next table shows all the relevant parameters and paths for the use case implementation:

USE CASE INFORMATION BLOCK	MODEL CLASS/PARAMETER	PATH	comments
ID FIELDS	Logical-termination-point[uuid]	/core-model:control-construct/logical-termination-point	
	uuid	PATH_logical-termination-point/uuid	unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC
	name [value-name]	PATH_logical-termination-point/name	additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible
	Layer-protocol[local-id]	/core-model:control-construct/logical-termination-point/layer-protocol	
	local-id	PATH_layer-protocol/local-id	local identifier for layer protocol
	layer-protocol-name	PATH_layer-protocol/layer-protocol-name-type	type of protocol
	air-interface-configuration	/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-configuration	
	air-interface-name	PATH_air-interface-configuration/air-interface-name	
	remote-air-interface-name	PATH_air-interface-configuration /remote-air-interface-name	
RF CARRIER CAPABILITIES	air-interface-capability	/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-capability	
	duplex-distance-is-freely-configurable	PATH_air-interface-capability/duplex-distance-is-freely-configurable	Duplex Spacing

	duplex-distance-list	PATH_air-interface-capability/duplex-distance-list	
	tx-frequency-min	PATH_air-interface-capability/tx-frequency-min	Tx Frequency
	tx-frequency-max	PATH_air-interface-capability/tx-frequency-max	
	rx-frequency-min	PATH_air-interface-capability/rx-frequency-min	Rx Frequency
	rx-frequency-max	PATH_air-interface-capability /rx-frequency-max	
	atpc-is-avail	PATH_air-interface-capability/atpc-is-avail	ATPC available
	atpc-range	PATH_air-interface-capability/atpc-range	ATPC range
	transmission-mode-list	PATH_air-interface-capability /transmission-mode-list	this list characterizes each transmission mode supported by the HW enabling the air interface. Configured min/max for each carrier, or current working mode will point to elements within this list
TRANSMISSION MODE PARAMETERS	transmission-mode-list[transmission-mode-name]	/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-capability/transmission-mode-list	
	transmission-mode-name	PATH_transmission-mode-list/transmission-mode-name	transmission mode / modulation scheme Ids
	modulation-scheme-name	PATH_transmission-mode-list /transmission-mode-name	
	channel-bandwidth	PATH_transmission-mode-list /channel-bandwidth	these three parameters, and the next one (symbol rate reduction) allow to make a calculation of the capacity linked to the transmission mode
	modulation-scheme	PATH_transmission-mode-list /modulation-scheme	
	code-rate	PATH_transmission-mode-list/code-rate	
	symbol-rate-reduction-factor	PATH_transmission-mode-list/symbol-rate-reduction-factor	in case the radio supports reducing the BW for specific transmission modes (e.g robust low modulation order modes)
	xpic-is-avail	PATH_transmission-mode-list/xpic-is-avail	XPIC (i.e. if XPIC is supported and active)
	supported-as-fixed-configuration	PATH_transmission-mode-list/supported-as-fixed-configuration	indicates if the mode is only supported in fixed modulation, with no ACM
	tx-power-min	PATH_transmission-mode-list/tx-power-min	Power levels
	tx-power-max	PATH_transmission-mode-list/tx-power-max	
	am-upshift-level	PATH_transmission-mode-list/am-upshift-level	ACM thresholds
	am-downshift-level	PATH_transmission-mode-list/am-downshift-level	

CARRIER ACTIVATION PARAMETERS	air-interface-configuration	/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-configuration	
	power-is-on	PATH_air-interface-configuration /power-is-on	RF Enabled (i.e. if transmitter/receiver are on)
	transmitter-is-on	PATH_air-interface-configuration /transmitter-is-on	
	receiver-is-on	PATH_air-interface-configuration /receiver-is-on	
CARRIER CONFIGURATION PARAMETERS	tx-frequency	PATH_air-interface-configuration /tx-frequency	
	duplex-distance	PATH_air-interface-configuration duplex-distance	for those carriers that allow for it (See capabilities)
	rx-frequency	PATH_air-interface-configuration/rx-frequency	
	tx-power	PATH_air-interface-configuration/tx-power	
	atpc-is-on	PATH_air-interface-configuration/atpc-is-on	ATPC activation and configuration
	atpc-thresh-upper	PATH_air-interface-configuration/atpc-thresh-upper	
	atpc-thresh-lower	PATH_air-interface-configuration/atpc-thresh-lower	
	atpc-tx-power-min	PATH_air-interface-configuration/atpc-tx-power-min	
	adaptive-modulation-is-on	PATH_air-interface-configuration /adaptive-modulation-is-on	Fixed Modulation / ACM activation
	transmission-mode-max	PATH_air-interface-configuration /transmission-mode-max	Physical modulation, coding, channel size, symbol rate reduction, RX sensitivity and other parameters of the mode are available in the transmission mode list for these modes. When an interface is configured without ACM, so in fixed modulation mode, the reference transmission mode will be represented by the transmission-mode-min parameter.
	transmission-mode-min	PATH_air-interface-configuration /transmission-mode-min	
	xpic-is-on	PATH_air-interface-configuration /xpic-is-on	XPIC activation for those transmission modes that allow for it
XPIC GROUPS	co-channel-profile:co-channel-profile-pac	/core-model:control-construct/profile-collection/profile/co-channel-profile:co-channel-profile-pac	
	co-channel-profile:xpic-is-avail	PATH_co-channel-profile-pac/co-channel-profile-capability/xpic-is-avail	
	co-channel-profile:profile-name	PATH_co-channel-profile-pac/co-channel-profile-configuration/profile-name	
	co-channel-profile:kind-of-co-channel-group	PATH_co-channel-profile-pac/co-channel-profile-configuration/kind-of-co-channel-group	XPIC / MIMO / ALIC

	co-channel-profile:logical-termination-point-list	PATH_co-channel-profile-pac/co-channel-profile-configuration/logical-termination-point-list	list of LTPs in the group
TDM PARAMETERS	tdm-container:tdm-container-configuration	/core-model:control-construct/logical-termination-point/layer-protocol/tdm-container:tdm-container-pac/tdm-container-configuration	
	tdm-container:interface-name	PATH_tdm-container-configuration/interface-name	
	tdm-container:interface-is-on	PATH_tdm-container-configuration /interface-is-on	
	tdm-container:tdm-container-kind	PATH_tdm-container-configuration /tdm-container-kind	type of TDM container (e.g, E1,...)

Additional capability / configuration parameters are also available in ONF model, that may be complementary to enrich the inventory or cover other more specific need from some operators. As an example, those related to air interface encryption, automatic frequency selection capabilities and configuration when HW is deployed that allows for it by the HW, or the setup and configuration of radio signal IDs to lock radio signals.

ONF modelling does not include yet specific parameters for BCA configuration. However, the number of bundled carriers forming a single Ethernet pipe, independently of their frequency or configuration may be indirectly be learned also from the LTP list. LTPs representing air interface layers which correspond to bundled carriers will be servers of structure layers that will have a common Ethernet container client layer. Again, from the client-server relations between LTPs, all the air interfaces that are servers of the same Ethernet Container may be easily identified as part of a bundled link and identified in the inventory.

Polarization is not supported as part of current ONF modelling.

In terms of workflow for the use case implementation, the controller may retrieve the information required for the use case in a single step for each of the NEs under control or breaking it down into individual queries (ltp list – profiles – air interface parameters – tdm interface parameters) to then process and build the inventory at application level. The order of the retrieval does not alter the result of the process.
It is possible that parts of the needed information (like the LTP list) is already retrieved and available at application level as it is part of other use cases.

Proposed implementation IETF/IEEE - Data models

The IETF RFC 8561 “A YANG Data Model for Microwave Radio Link” defines a YANG data model for control and management of radio link interfaces and their connectivity to packet (typically Ethernet) interfaces in a microwave/millimeter wave node. The data nodes for management of the interface protection functionality are broken out into a separate and generic YANG data model in order to make it available for other interface types as well.

The YANG module "ietf-microwave-radio-link", defined in the IETF RFC 8561, has the following structure:

```
module: ietf-microwave-radio-link
        +--rw radio-link-protection-groups
            +--rw protection-group* [name]
         +-- <parameters>
            +--rw xpic-pairs {xpic}?
                +--rw xpic-pair* [name]
        +-- <parameters>
            +--rw mimo-groups {mimo}?
                +--rw mimo-group* [name]
```

```
            +-- <parameters>
    augment /if:interfaces/if:interface:
            +-- <parameters>
```

Per Radio Interface

```
             +--rw radio-link-protection-groups
                 +--rw protection-group* [name]
            +--rw xpic-pairs {xpic}?
                +--rw xpic-pair* [name]
            +--rw mimo-groups {mimo}?
                +--rw mimo-group* [name]
      augment /if:interfaces/if:interface:
```

Parameter	Type / Sub-parameter	Description / Application (indicative)
+--rw carrier-id	string	Carrier ID
+--rw tx-enabled	boolean	RF Enabled
+--rw channel-separation	uint32	Bandwidth (Channels Size)
+--rw (freq-or-distance) +--rw duplex-distance	int32	Duplex Spacing
+--rw (coding-modulation-mode) *+--:(single)* *+--rw single* +--rw selected-cm (Note 1)	identityref	Fixed Modulation
+--rw (coding-modulation-mode) *+--:(adaptive)* +--rw adaptive		Adaptive Modulation
+--ro capabilities +--ro available-max-acm	identityref	Maximum Physical Modulation
+--ro capabilities +--ro available-min-acm	identityref	Minimum Physical Modulation
+--ro actual-tx-cm	identityref	Current Physical Modulation
+--rw tx-frequency	uint32	Tx Frequency
+--rw (freq-or-distance) *+--:(rx-frequency)* +--rw rx-frequency?	uint32	Rx Frequency
+--rw channel-separation	uint32	Channel Bandwidth
+--ro actual-transmitted-level	power	Tx Power
+--ro actual-received-level	power	RX Sensitivity
+--ro actual-snir	decimal64	SNIR
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)* +--rw atpc		Adaptive Tx Power
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)* *+--rw atpc* +--rw atpc-lower-threshold	power	Adaptive Tx Power Minimum
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)*	power	Adaptive Tx Power Maximum

+--rw atpc +--rw atpc-upper-threshold		
+--rw polarization	enumeration	Polarization
+--rw xpic-pairs {xpic} +--rw xpic-pair* [name] +--rw name +--rw enabled? +--rw members	-> /xpic-pairs/xpic-pair/name string boolean if:interface-ref	XPIC
+--rw tdm-connections* [tdm-type] {tdm}? +--rw tdm-type +--rw tdm-connections	 identityref uint16	TDM traffic

Note 1: In case of adaptive mode, this attribute is not defined.
Note 2: The BCA parameter is not currently supported in the YANG module "ietf-microwave-radio-link".

2.1.4 Use Case 1.4 – Wired port inventory

Use Case	Wired port inventory
Id	Use Case 1.4
Summary	Use case focuses on the retrieval of NE parameters related to wired ports to build a detailed port inventory that complements high-level topology information.
Benefits and Motivation	Main motivation is to achieve a harmonized port inventory, which will enable sophisticated automation-oriented applications across a MW/mmWave multi-vendor network, powered by AI/ML technologies and methods.
Detailed description	To build a consolidated port inventory, the SDN agnostic controller shall be able to retrieve harmonized information both covering equipment capabilities and configuration aspects (and additionally status) related to the ethernet ports available in any NE within the control domain. The main blocks of information and parameters identified as a target are: • **NE and interface Identification fields** • **Port level Transmission related information**. Relevant ones being amongst other: o Port Enabled o Port ID o Port Name o Physical Layer Speed o Duplex Mode • **MAC layer information relevant at port level**. Some relevant ones being: o MAC Address o Flow Control (i.e. different types of Ethernet PAUSE frame based flow control that can be enabled) o MTU Size • **Feature-related parameters relevant at port level**. As an example, o Synchronization information o LLDP activation and configuration (TTL, other) o DHCP availability and configuration
Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes (logical Termination Point and layer Protocol lists), with general aspects and requirements applying to the ONF CIM given in section 3, and the support of specific TR-532 PACs: • WireInterface_2.0.0, with resources (papyrus UML, overview and documentation)

available here and latest yang module here.

- MacInterface_1.0.0, with resources (papyrus UML, overview and documentation) available here and latest yang module here.
- LtpAugment_1.0.0, with resources available here and latest yang module here

In addition to these, inclusion in the inventory of port-related synchronization information requires supporting additional specific objects. In the specific case of synchronization, PAC design is ongoing and its final definition (according to ITU-T G.7721-2018) is not completed, so details included here may vary and will be updated accordingly.

- Synchronization PAC with resources available here

LTPs corresponding to a wired interface (corresponding to physical ethernet ports) will include a layer-protocol instance having a layer-protocol-name=LAYER_PROTOCOL_NAME_TYPE_WIRE_LAYER. The wire-interface pac attaches to the layer-protocol CIM providing a conditional augmentation with the MW specific classes and parameters in case this condition applies.

```
module: core-model-1-4
  +--rw control-construct!
     +--...
     +--rw logical-termination-point* [uuid]
     |  +--rw uuid
     |  +--rw name* [value-name]
     |  +--rw server-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--rw client-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--…
     |  +--rw layer-protocol* [local-id]
     |  |  +--rw local-id
     |  |  +--rw layer-protocol-name?
(=LAYER_PROTOCOL_NAME_TYPE_WIRE_LAYER)
     |  |  +--...
     |  |  +--rw wire-interface:wire-interface-pac
     |  |     +--ro wire-interface:wire-interface-capability
     |  |     +--rw wire-interface:wire-interface-configuration
     |  |     +--…
```

Same applies to the MAC Interfaces, having in this case within the LTP an LP instance having a layer-protocol-name= LAYER_PROTOCOL_NAME_TYPE_MAC_LAYER. It shall be possible to relate via LTP client-server relations any MAC layer to the underlying wire interface (see section 3 for LTP/LP layering aspects).

LtpAugment is attached to the LTP ONF CIM class providing a reference to the underlying hardware and physical connector supporting the interface for the lowest layers (wire and air). In this use case serves to relate port inventory to physical equipment for all the wired interfaces.

```
module: ltp-augment-1-0
  augment /core-model:control-construct/logical-termination-
point:
    +--rw ltp-augment-pac
       +--ro ltp-augment-capability
          +--ro equipment*   -> /core-model:control-
construct/equipment/uuid
          +--ro connector?   -> /core-model:control-
construct/equipment/connector/local-id
```

The implementation considers both available information related to interface capabilities and

configuration.

First relevant set for the inventory corresponds to the interface identification fields and the transmission related capabilities and configuration of the wired-interfaces. A summary of the relevant parameters for the use case, needed for the implementation follows:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
INTERFACE ID FIELDS	**Logical-termination-point[uuid]**	***/core-model:control-construct/logical-termination-point***	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point**/name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol LAYER_PROTOCOL_NAME_TYPE _WIRE_LAYER*
	wire-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-configuration***	
	interface-name	**PATH_wire-interface-configuration**/interface-name	
	remote-wire-interface-name	**PATH_wire-interface-configuration** /remote-wire-interface-name	
PORT TRANSMISSION - CAPABILITIES	**wire-interface-capability**	***/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-capability***	
	auto-pmd-negotiation-is-avail	**PATH_wire-interface-capability**/auto-pmd-negotiation-is-avail	
	auto-negotiation-pmd-selection-is-avail	**PATH_wire-interface-capability**/auto-negotiation-pmd-selection-is-avail	*this is relevant to limit the set of PMDs allowed for the auto-negotiation procedure*
	mii-kind	**PATH_wire-interface-capability**/mii-kind	*medium independent interface type e.g. SFP, QSFP, Xenpac, Soldered,…*
	mdi-kind	**PATH_wire-interface-capability**/mdi-kind	*medium dependent interface type e.g. FC / LC / RJ45,…*
	required-medium-kind	**PATH_wire-interface-capability**/required-medium-kind	*Required transmission medium type, e.g. TP_CATx, Single mode, multi mode fiber,*
	wavelength-min-list	**PATH_wire-interface-capability**/wavelength-min-list	
	wavelength-max-list	**PATH_wire-interface-capability**/wavelength-max-list	

	supported-pmd-kind-list[pmd-name]	**PATH_wire-interface-capability**/supported-pmd-kind-list	*list of supported pmds, each described by name / speed / duplex type*
	pmd-name	**PATH_supported-pmd-kind-list**/pmd-name	
	speed	**PATH_supported-pmd-kind-list**/speed	*Line speed, e.g "1000Mbit/s"*
	duplex	**PATH_supported-pmd-kind-list/**duplex	*half / full / undetermined*
PORT TRANSMISSION - CONFIGURATION	**wire-interface-configuration**	***/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-configuration***	
	interface-is-on	**PATH_wire-interface-configuration**/interface-is-on	
	auto-pmd-negotiation-is-on	**PATH_wire-interface-capability**/auto-pmd-negotiation-is-on	*to configure auto negotiation*
	auto-negotiation-pmd-list (*)	**PATH_wire-interface-capability**/auto-negotiation-pmd-list	*If (auto-pmd-negotiation-is-on==1) AND (auto-negotiation-pmd-selection-is-avail==1), this list defines the selection of PMDs the automated negotiation process is allowed to choose from*
	fixed-pmd-kind	**PATH_wire-interface-capability**/fixed-pmd-kind	*when auto negotiation is off, this will set the configured PMD for the interface amongst those available in the capability PMD list.*
	transceiver-configuration-list[transceiver-index] (**)	/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-configuration/transceiver-configuration-list	*list of 1-10 x transceiverConfigurationType* ***Still not reflected in current YANG version (see note)***
	transceiver-index	**PATH_transceiver-configuration-list**/transceiver-index	
	transceiver-is-on	**PATH_transceiver-configuration-list**/transceiver-is-on	*Source: 802.3 45.2.1.8 PMD transmit disable register*
	wavelenght	**PATH_transceiver-configuration-list** /wavelength	*SFF-8690. Wavelength of the signal of laser in pico meter.*

**When a device does not support editing the PMD list for auto negotiation (so, auto-negotiation-pmd-selection-is-avail==0), this should be an empty list. As a default configuration for this field the complete PMD list will be implemented. The operator may then restrict the list of PMDs allow for auto-negotiation deleting some of them.*
***This list replaces two previously existing lists of the configuration class (transceiver-is-on-list and wavelength-list) in order to enable the individual addressing of each element for configuration. This forms part of a series of modifications not yet implemented in new YANGs. Anyway decisions for modelling and implementation are closed, so the table shows the target modelling to avoid displaying information that will be out of date shortly.*

Second set needed for the inventory corresponds to the MAC-level port related parameters, which are available at the LTPs related to each of the Ethernet ports corresponding to their MAC layer interface. A summary of the relevant parameters for the use case, needed for the implementation follows:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
INTERFACE ID FIELDS	**Logical-termination-point[uuid]**	***/core-model:control-construct/logical-termination-point***	

	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point**/name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol LAYER_PROTOCOL_NAME_ TYPE_MAC_LAYER*
	mac-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/mac-interface:mac-interface-pac/mac-interface-configuration***	
	interface-name	**PATH_mac-interface-configuration**/interface-name	
MAC-LEVEL CAPABILITIES	**mac-interface-capability**	***/core-model:control-construct/logical-termination-point/layer-protocol/mac-interface:mac-interface-pac/mac-interface-capability***	
	hardware-mac-address	**PATH_mac-interface-capability**/hardware-mac-address	
	mac-address-configuration-is-avail	**PATH_mac-interface-capability**/mac-address-configuration-is-avail	
	supported-maximum-frame-size-list (*)	**PATH_mac-interface-capability**/supported-maximum-frame-size-list	*-1 in case configuring the maximum frame size is not supported by the HW. Must contain values configurable by the HW.* ***Still not reflected in current YANG version (see note)***
	supported-flow-control-mode-list	**PATH_mac-interface-capability**/supported-flow-control-mode-list	*Potential configurations of flow control (send only, receive only, send & receive, etc..)*
	supported-frame-format-list	**PATH_mac-interface-capability**/supported-frame-format-list	
MAC-LEVEL CONFIGURATION	**mac-interface-configuration**	***/core-model:control-construct/logical-termination-point/layer-protocol/mac-interface:mac-interface-pac/mac-interface-configuration***	
	mac-address-configuration-is-on	**PATH_mac-interface-configuration**/mac-address-configuration-is-on	*Shows if MAC address is enabled to be overwritten by configuration value*
	configured-mac-address	**PATH_mac-interface-configuration**/configured-mac-address	*Configured MAC address, overwriting HW MAC*
	maximum-frame-size	**PATH_mac-interface-configuration**/maximum-frame-size	*configured maximum MTU*
	transmitted-frame-format	**PATH_mac-interface-configuration**/transmitted-frame-format	
	flow-control-mode	**PATH_mac-interface-configuration**/flow-control-mode	*comfigured flow control mode*
	fragmentation-allowed	**PATH_mac-interface-configuration**/fragmentation-allowed	

**This list replaces two previously existing parameters of the configuration class (maximum-frame-size-min*

maximum-frame-size-max). This forms part of a series of modifications not yet implemented in new YANGs. Anyway, decisions for modelling and implementation are closed, so the table shows the target modelling to avoid displaying information that will be out of date shortly.

ONF model considers separated classes for config and state parameters. In the case status information is required to complement the wired inventory use case or to check current status of the configured interface in parallel to the set configuration, it is available at the status class of the Wire and MAC interfaces. Relevant state fields for the use case being:

USE CASE INFO BLOCK	***MODEL CLASS / PARAMETER***	***PATH***	*comments*
PORT TRANSMISSION - STATUS	**wire-interface-status**	***/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-status***	
	interface-status	**PATH_wire-interface-status**/interface-status	*802.3 30.3.2.1.7, PHY exists, is powered and can be managed*
	link-is-up	**PATH_wire-interface-status**/link-is-up	*comms established with the remote site, together with transceiver status*
	pmd-is-up	**PATH_wire-interface-status**/pmd-is-up	*inverse of 802.3 45.2.1.2.3 fault 1.1.7*
	pmd-kind-cur	**PATH_wire-interface-status**/pmd-kind-cur	*current PMD*
	pmd-negotiation-state	**PATH_wire-interface-status**/pmd-negotiation-state	*status of the auto negotiation procedure (e.g. disabled, enabled, completed, failed,…)*
	receiver-status-list(**)	/core-model:control-construct/logical-termination-point/layer-protocol/wire-interface:wire-interface-pac/wire-interface-status/receiver-status-list	*list [1..10] x receiverStatusType.* ***Still not reflected in current YANG version (see note)***
	receiver-index	**PATH_receiver-status-list**/pmd-negotiation-state	
	receive-signal-is-detected	**PATH_receiver-status-list**/pmd-negotiation-state	
	rx-level-cur	**PATH_receiver-status-list**/pmd-negotiation-state	*current receive level dBm*
MAC-LEVEL PORT STATUS	**mac-interface-status**	***/core-model:control-construct/logical-termination-point/layer-protocol/mac-interface:mac-interface-pac/mac-interface-status***	
	interface-status	**PATH_mac-interface-status**/interface-status	*e.g up/down/dormant/…*
	mac-address-cur	**PATH_mac-interface-status***/mac-address-cur*	
	flow-control-mode-cur	**PATH_mac-interface-status***/flow-control-mode-cur*	
	received-ethernet-frame-format-cur	**PATH_mac-interface-status***/received-ethernet-frame-format-cur*	

***This list replaces two previously existing lists of the configuration class (receive-signal-is-detected-list and rx-level-cur-list) in order to enable the individual addressing of each element for configuration. This forms part of a series of modifications not yet implemented in new YANGs. Anyway, decisions for modelling and implementation are closed, so the table shows the target modelling to avoid displaying information that will be out of date shortly.*

At the time of writing standardization work in ONF in relation to synchronization has not completely finished, so details included here may vary and would need to be updated accordingly if modifications are agreed**.** PTP synchronization parameters at port level are contained within a specific class of the ITU model, the PTP_pac within the SyncLPSpec, which is a specific augments of the layer protocol core model class. This way, specific LTPs within the network element logical-termination-point will be available to model synchronization functions. Sink and source synchronization elements are modelled using the bi-directional property of the LTPs as indicated

in the ITU recommendation and the relation with the interface LTPs (related ultimately with the physical ports) can be done using client server relations available at the LTP elements and the associated port ID field.
Within the PTP_pac, the next parameters are available, and can be attached to the port inventory to show the relevant aspects in relation to the PTP port configuration:

USE CASE INFO BLOCK	MODEL CLASS/PARAMETER	COMMENTS
PORT-LEVEL SYNCH PARAMETERS	ptpPortEnableStatus	Indicate whether to enable this PTP port or not.
	ptpPortState	The current PTP state of the PTP port, such as master, slave, passive, initializing, listening, premaster, uncalibrated, and faulty.
	ptpAsymmetryCorrection	The asymmetry correction value of this PTP port.
	ptpTwoStepFlag	Indicate whether one-step or two-step mechanism is adopted.
	ptpUdpEgressConfiguration	The configuration of PTP UDP encapsulation, including destinationIpAddress (string) and ipProtocolType (IPv4/IPv6).
	ptpMacEgressConfiguration	The configuration of PTP MAC encapsulation, including destinationMacAddress (string) and vlanCconfiguration (string).
	ptpAnnounceInterval	The sending interval of PTP announce message.
	ptpAnnounceReceiptTimeout	It is used for fault detection of PTP announce messages.
	ptpSyncInterval	The sending interval of PTP Sync message.
	ptpMinDelayReqInterval	The sending interval of PTP Delay_req message.
	ptpMasterOnly	The per-port attribute masterOnly
	ptpLocalPriority	The per-port attribute localPriority

DHCP and LLDP are not yet part of the ONF TR-532 model so port related parameters in relation to these features are not available in an ONF implementation.
In relation to use case implementation workflow, retrieval of the LTP list and layer protocol local ids and names may be done in a separate step, being the base for many use cases, as it serves to have the complete overview of physical ports and the interface stack of any NE (LTP client-server relations and layer-protocol-names within the layer protocol class of each LTP serve to build the complete layer stack of a NE).
Addressing then a single or several interfaces (in this case wire and MAC interfaces) corresponding to a port to build the inventory can be done using the control-construct uuid to identify the NE, and then the LTP uuid and the layer protocol local-id for the desired interface within the node, extracting the desired fields, indicated in the implementation.
Filtering makes also possible to retrieve the necessary data from all the interfaces and then post process at application level, so the workflow is flexible.

Proposed implementation IETF/IEEE -Data models

1. The IETF RFC 8343 “A YANG Data Model for Interface Management” defines a YANG data model for the management of network interfaces. It is expected that interface-type-specific data models augment the generic interfaces data model defined in the RFC 8343. The data model includes definitions for configuration and system state (status information and counters for the collection of statistics). The YANG data model conforms to the Network Management Datastore Architecture (NMDA) defined in RFC 8342.
The YANG module "ietf-interfaces", defined in the IETF RFC 8343, has the following structure:

```
module: ietf-interfaces
        +--rw interfaces
             +--rw interface* [name]
                  <parameters>
```

Per Ethernet Port
+--rw interfaces

+--rw interface* [name]		
Parameter	Type / Sub-parameter	Description / Application (indicative)
+--rw enabled	boolean	Port Enabled
+--ro if-index	int32	Port ID
+--rw name	string	Port Name
+--ro speed	yang:gauge64	Physical Layer Speed
+--rw type	identityref	Duplex Mode
+--ro phys-address	yang:phys-address	MAC Address

2. The YANG module "ieee802-ethernet-interface", defined in the IEEE 802.3.2-2019: “IEEE Standard for Ethernet - YANG Data Model Definitions”, is available here:

https://github.com/YangModels/yang/blob/master/standard/ieee/published/802.3/ieee802-ethernet-interface.yang

Per Ethernet Port	
Parameter	Description / Application (indicative)
flow-control	Flow Control

3. The YANG module "ieee802-dot1ab-lldp", defined in the IEEE P802.1ABcu: “IEEE Draft Standard for Local and Metropolitan Area Networks - Station and Media Access Control Connectivity Discovery Amendment: YANG Data Model”, is available here:

https://github.com/YangModels/yang/blob/master/standard/ieee/draft/802.1/ABcu/ieee802-dot1ab-lldp.yang

Per Ethernet Port		
Parameter	Type / Sub-parameter	Description / Application (indicative)
lldp-cfg		LLDP
message-tx-hold-multiplier		LLDP TTL

4. The IETF RFC 8575 “YANG Data Model for the Precision Time Protocol (PTP)” defines a YANG data model for the configuration of devices and clocks using the Precision Time Protocol (PTP) as specified in IEEE Std 1588-2008. It also defines the retrieval of the configuration information, the data sets and the running states of PTP clocks. The YANG module in this document conforms to the Network Management Datastore Architecture (NMDA). It is available here:

https://tools.ietf.org/html/rfc8575

The YANG module "ietf-ptp", defined in the IETF RFC 8575, has the following structure:

```
module: ietf-ptp
        +--rw ptp
       +--rw instance-list* [instance-number]
                  <parameters>
       +--rw transparent-clock-default-ds
                  <parameters>
       +--rw transparent-clock-port-ds-list* [port-number]
                  <parameters>
```

Per Ethernet Port		
+--rw ptp +--rw instance-list* [instance-number] +--rw transparent-clock-default-ds +--rw transparent-clock-port-ds-list* [port-number]		
Parameter	Type / Sub-parameter	Description / Application (indicative)
+--rw instance-number uint32	uint16	

+--rw port-ds-list* [port-number] +--rw port-number		
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw port-state	port-state-enumeration	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw underlying-interface	if:interface-ref	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw log-min-delay-req-interval	int8	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw peer-mean-path-delay	time-interval-type	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw log-announce-interval	int8	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw announce-receipt-timeout	uint8	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw log-sync-interval	int8	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw delay-mechanism	delay-mechanism-enumeration	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw log-min-pdelay-req-interval	int8	
+--rw instance-number uint32 +--rw port-ds-list* [port-number] +--rw version-number	uint8	
+--rw instance-number uint32 +--rw transparent-clock-port-ds-list* [port-number] +--rw port-number	uint16	
+--rw instance-number uint32 +--rw transparent-clock-port-ds-list* [port-number] +--rw log-min-pdelay-req-interval	int8	
+--rw instance-number uint32 +--rw transparent-clock-port-ds-list* [port-number] +--rw faulty-flag	boolean	
+--rw instance-number uint32 +--rw transparent-clock-port-ds-list* [port-number] +--rw peer-mean-path-delay	time-interval-type	

5. The IETF draft “Yang Data Model for DHCP Protocol” defines a YANG data model for DHCP Server, Relay andClient, including configuration and running state. It is available here:

https://datatracker.ietf.org/doc/html/draft-liu-dhc-dhcp-yang-model-07

The YANG module "ietf-dhcp", defined in the IETF draft, has the following structure:

```
module: ietf-dhcp
```

```
          +--rw dhcp
         +--rw server
                    <parameters>
         +--rw relay
                    <parameters>
         +--rw client
                    <parameters>
```

6. The IETF draft "Common Interface Extension YANG Data Models" defines two YANG modules that augment the Interfaces data model defined in the "YANG Data Model for Interface Management" with additional configuration and operational data nodes to support common lower layer interface properties, such as interface MTU. It is available here:

https://datatracker.ietf.org/doc/html/draft-ietf-netmod-intf-ext-yang

The YANG module "ietf-if-extensions", defined in the IETF draft, has the following structure:

```
   module: ietf-if-extensions
          augment /if:interfaces/if:interface:
              +--rw carrier-delay {carrier-delay}?
                    <parameters>
              +--rw dampening! {dampening}?
                    <parameters>
              +--rw encapsulation
                    <parameters>
              +--rw loopback?           identityref {loopback}?
              +--rw max-frame-size?     uint32 {max-frame-size}?
              +--ro forwarding-mode?    identityref
          augment /if:interfaces/if:interface:
              +--rw parent-interface     if:interface-ref {sub-
interfaces}?
          augment /if:interfaces/if:interface/if:statistics:
              +--ro in-discard-unknown-encaps?    yang:counter64
                                {sub-interfaces}?
```

2.1.5 Use Case 1.5 - Services inventory

Use Case	Services inventory
Id	Use Case 1.5
Summary	Use case focuses on the retrieval of NE parameters related to services that are provisioned in the MW and mmWave systems. In general, MW and mmWave technologies can act as switches or router devices. As the vast majority of wireless backhaul implementations today rely on Ethernet services, the present analysis focus on L2 parameters.
Benefits and Motivation	Service inventory information relies on different proprietary EMS tools and platforms in a multi-vendor network environment. By having standardized equipment interfaces and data models, operators will have a harmonized service inventory across a MW/mmWave multi-vendor network.

Detailed description	The agnostic SDN controller shall be able to retrieve from any NE within the network domain service-related parameters to build a consolidated harmonized inventory, which serves also as the base for service management use cases. Specific focus is on the configured VLANs (802.1Q) as well as switch and VLAN interface capabilities and configuration.
	
	• Network element and interface Identification fields
	• VLAN interface and switch configuration: This will consider setting the specific configuration of the VLAN interfaces or those general parameters applicable to the MW ethernet switch. Within scope:
	  ○ Interface activation
	  ○ Component type (e.g. C-VLAN, S-VLAN)
	  ○ Default VLAN ID configuration
	  ○ Default Priority configuration

	○ Ingress VLAN filtering enable / disable ○ Ingress Tag filtering type configuration ○ PCP bits interpretation type ○ P-bit to priority queue mapping list ○ Priority queue to P-bit mapping list
Proposed implementation ONF -Data models	Implementation of this use case relies on several ONF CIM classes and specific TR-532 MW technology specific augments. Within the CIM classes, the interfaces and the different layers of transport are modelled using the logical Termination Point and layer Protocol lists, while the internal switch(es) within the NE rely on the forwarding Domain, forwarding Construct and fcPort classes. General aspects, requirements and class descriptions of these ONF CIM classes are given in section 3. In summary, the additional core model classes and specific TR-532 PACs that need to be supported are: • **LogicalTerminationPoint (LTP) and layerProtocol (LP)**: these serve to model interface termination points linked to physical ports, available at the different transport layers. In this use case they will serve to model VLAN interfaces. ○ LTPs are generally augmented by the TR-532 **ltpAugment_1.0.0** PAC, to link logical interfaces to physical ports. that needs. Ltp Augment PAC resources are available here with latest yang module here. ○ LPs corresponding to a VLAN layer interface are augmented by TR-532 **vlanInterface_1.0.0** PAC, to include MW specific VLAN modelling. Vlan Interface PAC resources are available here with latest yang module here. • **Forwarding-domain (FD)**: This class serves as the base to model the ethernet switching within the MW NE. Each forwarding domain is constrained to a list of the available LTPs of a given transport layer (same layer-protocol) to establish actual forwarding. In this use case will represent the VLAN switch, enabling potential forwarding between available VLAN interfaces. ○ FD is augmented by the TR-532 **vlanFd_1.0.0** PAC, to include MW VLAN switch specific modelling. Resources are available here with latest yang module here. • **Forwarding-construct (FC)**: serves to represent enabled forwarding between two or more FcPorts at a particular specific layer-protocol. In this use case represents an actual VLAN between switch ports. ○ FD is augmented by the TR-532 **vlanFc_1.0.0** PAC, to include MW VLAN switch specific modelling. Resources are available here with latest yang module here. • **FC-port**: serves to associate FCs to the existing LTPs that model the interfaces. This class does not require technology specific augmentation. The next tree summarizes the relevant classes and attachments from the TR-532 augments: `--rw control-construct!` `  +--uuid` `  +--...` `  +--rw equipment*` `	+--...` ` +--rw logical-termination-point*` `	+--uuid` `	+--...` `	+--rw ltp-augment-pac` `	+--rw layer-protocol*` `		+--rw local-id` `		+--rw layer-protocol-name` `(=LAYER_PROTOCOL_NAME_TYPE_VLAN_LAYER for VLAN interfaces)` `		+--...` `		+--rw vlan-interface-pac` ` +--rw forwarding-domain* [uuid]`

```
     |  +--uuid
     |  +--rw layer-protocol-name?
     |  +--rw logical-termination-point*    -> /control-
construct/logical-termination-point/uuid
     |  +--...
     |  +--rw vlan-fd-pac
     |  +--rw fc* [uuid]
     |  |  +--uuid
     |  |  +--rw layer-protocol-name?
     |  |  +--...
     |  |  +--rw vlan-fc-pac
     |  |  +--rw fc-port* [local-id]
     |  |  |  +--rw local-id
     |  |  |  +--rw logical-termination-point*        ->
/control-construct/logical-termination-point/uuid
     +--…
```

For the service inventory cases, three main needs are typically present:

- Retrieval of the forwarding domain list, including:
  - Component type, and key capabilities and configuration / state of the MW VLAN switch
  - forwarding Construct list, to retrieve existing L2 VLANs, relevant VLAN configuration parameters and the ports/interfaces in which they are set
- Retrieval of the key capabilities and configuration / state of all or specific VLAN interfaces

The list of forwarding-domains available at control-construct level will contain all the FDs available within the NE. VLAN switching will be modelled by FDs having a specific protocol type, layer-protocol-name = LAYER_PROTOCOL _NAME_TYPE_VLAN_LAYER. The FD will include the list of LTPs (VLAN interfaces) relevant for the potential forwarding, and the TR-532 vlan-fd augment than includes all the MW specific relevant parameters. The next table shows the key ones considering the use case scope (it also includes general state information, that can be interesting at application level):

USE CASE INFO BLOCK	*MODEL CLASS/ PARAMETER*	*REQUIRED PATH*	*COMMENTS*
NE ID FIELDS	uuid	/core-model:control-construct/uuid	
	name[value-name]	/core-model:control-construct/name	*List of value-name & name pairs. More than one element may be specified with different value-names. As example, one pair value-name="externalLabel" & name="XXXX" matching the external label implemented by the operator for the equipment*
VLAN switch ID and related LTPs	**forwarding-domain[uuid]**	***/core-model:control-construct/forwarding-domain***	
	uuid	**PATH_forwarding-domain**/uuid	
	name[value-name]	**PATH_forwarding-domain**/name	
	layer-protocol-name	**PATH_forwarding-domain**/layer-protocol-name	*protocol layer (LAYER_PROTOCOL_NAME_ TYPE_VLAN_LAYER)*
	logical-termination-point	**PATH_forwarding-domain**/logical-termination-point	*list of LTPS that constitute this potential domain for forwarding at the specific protocol layer*

VLAN SWITCH CAPABILITIES	**vlan-fd-capability**	***/core-model:control-construct/forwarding-domain/vlan-fd:vlan-fd-pac/vlan-fd-capability***	
	component-id	**PATH_vlan-fd-capability**/component-id	
	supported-sub-layer-protocol-name-list	**PATH_vlan-fd-capability**/supported-sub-layer-protocol-name-list	*supported amongst: C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	maximum-number-of-vlan-ids	**PATH_vlan-fd-capability**/maximum-number-of-vlan-ids	
	traffic-classes-is-avail	**PATH_vlan-fd-capability**/traffic-classes-is-avail	*devices that may support mapping priority into different traffic classes*
VLAN SWITCH CONFIGURATION	**vlan-fd-configuration**	***/core-model:control-construct/forwarding-domain/vlan-fd:vlan-fd-pac/vlan-fd-configuration***	
	fd-name	**PATH_vlan-fd-configuration**/fd-name	*operator naming*
	sub-layer-protocol-name	**PATH_vlan-fd-configuration**/sub-layer-protocol-name	*C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	mac-address	**PATH_vlan-fd-configuration**/mac-address	*MAC address configured for bridge component*
	traffic-classes-is-on	**PATH_vlan-fd-configuration**/traffic-classes-is-on	*when disable, single priority to all traffic in the bridge*
VLAN SWITCH STATUS	**vlan-fd-status**	***/core-model:control-construct/forwarding-domain/vlan-fd:vlan-fd-pac/vlan-fd-status***	
	fd-status	**PATH_vlan-fd-status**/fd-status	
	number-of-ports-cur	**PATH_vlan-fd-status**/number-of-ports-cur	
	mac-address-cur	**PATH_vlan-fd-status** /mac-address-cur	

Other capability parameters are available at the capability class of the VLAN-fd PAC, that may be relevant for more complex inventory tasks.

The configured VLANs in a given FD can be retrieved through the embedded forwarding-construct list. Each FC instance, includes also the relevant MW technology specific parameters within the vlan-fc TR-532 augment. Each FC (VLAN) includes a list of fc-ports that will point to the specific VLAN interface (LTP/LP). Linkage between physical ports and LTPs/interfaces can also be learned from the ltp-augment available at each LTP, which will point to the equipment supporting the interface and the relevant physical connector.

The next table shows the relevant parameters that need to be supported and extracted for any given element of the FC list, for inventory purposes:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*REQUIRED PATH*	*COMMENTS*
FC OBJECT IDs	**fc[uuid]**	***/core-model:control-construct/forwarding-domain/fc***	
	uuid	**PATH_fc**/uuid	
	name[value-name]	**PATH_fc**/name	
VLAN CAPABILITIES	**vlan-fc-capability**	***/core-model:control-construct/forwarding-domain/fc/vlan-fc:vlan-fc-pac/vlan-fc-capability***	

	supported-sub-layer-protocol-name-list	**PATH_vlan-fc-capability**/supported-sub-layer-protocol-name-list	***Supported Component types*** *C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
VLAN CONFIGURATION	**vlan-fc-configuration**	***/core-model:control-construct/forwarding-domain/fc/vlan-fc:vlan-fc-pac/vlan-fc-configuration***	
	fc-name	**PATH_vlan-fc-configuration**/fc-name	***operator naming****, optional on top of the VLAN ID*
	sub-layer-protocol-name	**PATH_vlan-fc-configuration**/sub-layer-protocol-name	***VLAN configured Component type:*** *C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	vlan-id	**PATH_vlan-fc-configuration**/vlan-id	***VLAN ID***
VLAN - RELATED INTERFACES	**fc-port[local-id]**	***/core-model:control-construct/forwarding-domain/fc/fc-port***	**List of LTPs of the VLAN**
	local-id	**PATH_fc-port**/local-id	
	logical-termination-point	**PATH_fc-port** /logical-termination-point	*pointer to the specific LTPs in the logical-termination-point-list (uuids)*

Capabilities and configuration information of any specific VLAN interfaces are available via the TR-532 vlan-interface augment of the specific LTP/LP representing the interface. The main parameters that need to be supported to fulfil the use case description are (including the capability and the configuration)

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*REQUIRED PATH*	*COMMENTS*
INTERFACE ID FIELDS	**Logical-termination-point[uuid]**	***/core-model:control-construct/logical-termination-point***	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point/**name	*additional field for labeling the LTP according to operator rules.* *Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol LAYER_PROTOCOL_NAME _TYPE_VLAN_LAYER*
	vlan-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration***	
	interface-name	**PATH_vlan-interface-configuration**/interface-name	

INTERFACE CAPABILITY	**vlan-interface-capability**	***/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-capability***	
Component and interface type	supported-sub-layer-protocol-name-list	**PATH_vlan-interface-capability**/supported-sub-layer-protocol-name-list	*C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	supported-interface-kind-list	**PATH_vlan-interface-capability**/supported-interface-kind-list	*D_BRIDGE_PORT, C_VLAN_BRIDGE_PORT, CUSTOMER_EDGE_PORT, PROVIDER_EDGE_PORT, UPLINK_ACCESS_PORT, …*
Ingress tag filtering	configuring-ingress-tag-filtering-is-avail	**PATH_vlan-interface-capability**/configuring-ingress-tag-filtering-is-avail	
Ingress VLAN filtering	ingress-vlan-id-filtering-is-avail	**PATH_vlan-interface-capability**/ingress-vlan-id-filtering-is-avail	
PCP interpretation	available-pcp-bits-interpretation-kind-list	**PATH_vlan-interface-capability**/available-pcp-bits-interpretation-kind-list	*8P0D, 7P1D, 6P2D, 5P3D, undefined*
P-bit to priority mapping	configuring-pcp-bits-decoding-is-avail	**PATH_vlan-interface-capability**/configuring-pcp-bits-decoding-is-avail	
Priority to p-bit mapping	configuring-pcp-bits-encoding-is-avail	**PATH_vlan-interface-capability**/configuring-pcp-bits-encoding-is-avail	
default VLAN priority	number-of-available-priorities	**PATH_vlan-interface-capability**/number-of-available-priorities	
INTERFACE CONFIG	**vlan-interface-configuration**	***/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration***	
Component and interface type	sub-layer-protocol-name	**PATH_vlan-interface-configuration**/sub-layer-protocol-name	***configured component type:*** *C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	interface-kind	**PATH_vlan-interface-configuration**/interface-kind	***configured interface type:*** *D_BRIDGE_PORT, C_VLAN_BRIDGE_PORT, CUSTOMER_EDGE_PORT, PROVIDER_EDGE_PORT, UPLINK_ACCESS_PORT, …*
default VLAN ID	default-vlan-id	**PATH_vlan-interface-configuration**/default-vlan-id	
default VLAN priority	default-priority	**PATH_vlan-interface-configuration/**default-priority	
Ingress tag filtering	ingress-tag-filtering	**PATH_vlan-interface-configuration**/ingress-tag-filtering	*Untagged and priority frames, only tagged, all frames*
Ingress VLAN filtering	ingress-vlan-id-filtering-is-on	**PATH_vlan-interface-configuration**/ingress-vlan-id-filtering-is-on	
PCP interpretation	pcp-bits-interpretation-kind	**PATH_vlan-interface-configuration**/pcp-bits-interpretation-kind	
P-bit to priority mapping	**pcp-bit-to-priority-mapping-list[to-be-decoded-pcp-bits-value]**	/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration/pcp-bit-to-priority-mapping-list	

	to-be-decoded-pcp-bits-value	**PATH_pcp-bit-to-priority-mapping-list**/to-be-decoded-pcp-bits-value	*0 to 7*
	associated-priority-value	**PATH_pcp-bit-to-priority-mapping-list**/associated-priority-value	*0 to 7*
	associated-drop-eligibility	**PATH_pcp-bit-to-priority-mapping-list**/associated-drop-eligibility	
Priority to p-bit mapping	**pcp-bits-encoding-mapping-list [to-be-encoded-priority-value to-be-encoded-drop-eligibility]**	/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration/pcp-bits-encoding-mapping-list	
	to-be-encoded-priority-value	**PATH_pcp-bits-encoding-mapping-list**/to-be-encoded-priority-value	*0 to 7*
	associated-pcp-bits-value	**PATH_pcp-bits-encoding-mapping-list**/associated-pcp-bits-value	*0 to 7*
	to-be-encoded-drop-eligibility	**PATH_pcp-bits-encoding-mapping-list**/to-be-encoded-drop-eligibility	

As workflow related aspects for the use case implementation, the SDN Controller will need to retrieve both the forwarding Domain list (or a specific forwarding Domain identified by its uuid) to extract the switch capabilities, configuration and the configured L2 VLANs as well as the complete list of VLAN interfaces (selecting the relevant parameters) to build the NE service inventory at application level. The order of the actions will not alter the result, being also possible using filtering to retrieve all the relevant info in a single action

Proposed implementation IETF/IEEE -Data models

The YANG module "ieee802-dot1q-bridge", defined in the IEEE 802.1QcpTM: "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks -- Amendment 30: YANG Data Model", is available here: https://github.com/YangModels/yang/blob/master/standard/ieee/published/802.1/ieee802-dot1q-bridge.yang

Parameter	Description / Application (indicative)
/dot1q:bridges/bridge / component/type	Component type (e.g. C-VLAN, S-VLAN)
/if:interfaces/interface/dot1q:bridgeport/ pvid	Default VLAN ID
/if:interfaces/interface/dot1q:bridgeport/ default-priority	Default Priority
/if:interfaces/interface/dot1q:bridgeport/ enable-ingressfiltering	Enable Ingress VLAN filtering
/dot1q:bridges/bridge / component/ bridgevlan/ vlan/egressports	Ingress Tag filtering
/if:interfaces/interface/dot1q:bridgeport/ pcp-selection	PCP selection
/if:interfaces/interface/dot1q:bridgeport/ pcp-decodingtable	P-bit to priority queue mapping list
/if:interfaces/interface/dot1q:bridgeport/ pcp-encodingtable	Priority queue to P-bit mapping list

2.2 Service Provisioning

2.2.1 Use Case 2.1 - VLAN service provisioning

Use Case	VLAN Service provisioning
Id	Use Case 2.1
Summary	Use case focuses on L2 service provisioning for MW NEs, specifically on the typical VLAN management (creation, configuration, deletion) between the switch ports of the MW network elements, including basic configurability of the relevant parameters of the created VLANs.
Benefits and Motivation	MW links and MW NEs are typically managed as layer 2 devices in the lower part of the mobile backhaul aggregation network. Even if typically, regional considerations are applied to the MW deployments, to simplify operational aspects, in this section of the aggregation it is common to find MW link chains composed by HW from different vendors and equipment types. Access network dynamic evolution requires constant management of transport services in the aggregation section, and in networks of a certain scale, hundreds of VLAN management procedures need to be carried out daily. In many cases, service creation is still managed with a low degree of automation and requires manual operation in each of the elements of the MW chains, requiring also specific management by vendor or equipment type. This then constitutes a time consuming and inefficient process, dependent on multiple specific systems and EMS proprietary tools, with large room for improvement and a clear need for automation, which would provide obvious benefits in terms of simplicity, harmonization and resource efficiency. There is also a clear need of increasing automation coming from the deployment of newer mobile standard access networks like 5G, and a need for multi-domain common orchestration of service creation, which requires as well as harmonized and automated way to manage VLAN creation in the MW network. In this respect, this use case also becomes an enabler of the controller NBI service provisioning use case.

Detailed description	It shall be possible to manage VLANs (802.1Q) on the switch ports (LAN/WAN) of any MW NE under control of the MW SDN agnostic controller, using a harmonized SBI model and procedures for the management. This use case becomes then an enabler of multi-hop and multi-domain service provisioning NBI use cases for a MW agnostic controller. The main relevant needs are:
	• VLAN interface and switch configuration: This will consider setting the specific configuration of the VLAN interfaces or those general parameters applicable to the MW ethernet switch. Within scope:
	  ○ Interface activation
	  ○ Component type (e.g. C-VLAN, S-VLAN)
	  ○ Default VLAN ID configuration
	  ○ Default Priority configuration
	  ○ Ingress VLAN filtering enable / disable
	  ○ Ingress Tag filtering type configuration
	  ○ PCP bits interpretation type
	  ○ P-bit to priority queue mapping list
	  ○ Priority queue to P-bit mapping list
	• VLAN creation: It will be possible to create a new VLAN between two specific ports (and related interfaces) of the switch within the MW NE. The new VLAN will have a specific VLAN ID, which shall be configurable by the operator.
	• VLAN management: Once created, it shall be possible to configure specific parameters that apply to the newly created VLAN, additional to those generally available for the interface. For example:
	  ○ VLAN activation / de-activation
	  ○ VLAN ID modification
	• VLAN deletion. It will be possible to delete an existing VLAN between two specific ports, and all related configuration.

Proposed implementation ONF -Data models	Implementation requires a similar class and TR-532 PAC support than the one already introduced for the extraction of the service inventory. Key difference being that provisioning focuses on configuration classes of the TR-532 PACs, and that the relevant actions for the provisioning as VLAN creation/deletion/etc. are managed through dedicated RPCs specified in the TR-532 PACs, which need to be supported for the correct management of the services at NE level. In summary, the core model classes and specific TR-532 PACs that need to be supported are: • **Logical Termination Point (LTP) and layer Protocol (LP)**: these serve to model interface termination points linked to physical ports, available at the different transport layers. In this use case they will serve to model VLAN interfaces. o LTPs are generally augmented by the TR-532 **ltpAugment_1.0.0** PAC, to link logical interfaces to physical ports. that needs. Ltp Augment PAC resources are available here with latest yang module here. o LPs corresponding to a VLAN layer interface are augmented by TR-532 **vlanInterface_1.0.0** PAC, to include MW specific VLAN modelling. Vlan Interface PAC resources are available here with latest yang module here. • **Forwarding-domain (FD)**: This class serves as the base to model the ethernet switching within the MW NE. Each forwarding domain is constrained to a list of the available LTPs of a given transport layer (same layer-protocol) to establish actual forwarding. In this use case will represent the VLAN switch, enabling potential forwarding between available VLAN interfaces. o FD is augmented by the TR-532 **vlanFd_1.0.0** PAC, to include MW VLAN switch specific modelling. Resources are available here with latest yang module here. • **Forwarding-construct (FC)**: serves to represent enabled forwarding between two or more FcPorts at a particular specific layer-protocol. In this use case represents an actual VLAN between switch ports. o FD is augmented by the TR-532 **vlanFc_1.0.0** PAC, to include MW VLAN switch specific modelling. Resources are available here with latest yang module here. • **FC-port**: serves to associate FCs to the existing LTPs that model the interfaces. This class does not require technology specific augmentation. The next tree summarizes the relevant classes and attachments from the TR-532 augments: `--rw control-construct!` `  +--uuid` `  +--...` `  +--rw equipment*` `	+--...` ` +--rw logical-termination-point*` `	+--uuid` `	+--...` `	+--rw ltp-augment-pac` `	+--rw layer-protocol*` `		+--rw local-id` `		+--rw layer-protocol-name` `(=LAYER_PROTOCOL_NAME_TYPE_VLAN_LAYER for VLAN interfaces)` `		+--...` `		+--rw vlan-interface-pac` ` +--rw forwarding-domain* [uuid]` `	+--uuid` `	+--rw layer-protocol-name?` `	+--rw logical-termination-point* -> /control-construct/logical-termination-point/uuid` `	+--...` `	+--rw vlan-fd-pac` `	+--rw fc* [uuid]` `		+--uuid` `		+--rw layer-protocol-name?` `		+--...`

```
     |   |   +--rw vlan-fc-pac
     |   |   +--rw fc-port* [local-id]
     |   |   |   +--rw local-id
     |   |   |   +--rw logical-termination-point*       ->
/control-construct/logical-termination-point/uuid
     +--…
```

The use case considers within scope the configuration of specific aspects of the VLAN layer of the available interfaces in the NE, as well as the general NE VLAN switching properties. For the interfaces, the relevant classes and configuration parameters being:

USE CASE INFO BLOCK	MODEL CLASS / PARAMETER	REQUIRED PATH	COMMENTS
INTERFACE ID FIELDS	**Logical-termination-point[uuid]**	***/core-model:control-construct/logical-termination-point***	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point**/name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol LAYER_PROTOCOL_NAME_ TYPE_VLAN_LAYER*
	vlan-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration***	
	interface-name	**PATH_vlan-interface-configuration**/interface-name	
INTERFACE CONFIG	**vlan-interface-configuration**	***/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration***	
Component and interface type	sub-layer-protocol-name	**PATH_layer-protocol**/sub-layer-protocol-name	***configured component type:*** *C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	interface-kind	**PATH_layer-protocol**/interface-kind	***configured interface type:*** *D_BRIDGE_PORT, C_VLAN_BRIDGE_PORT, CUSTOMER_EDGE_PORT, PROVIDER_EDGE_PORT, UPLINK_ACCESS_PORT, …*
default VLAN ID	default-vlan-id	**PATH_layer-protocol**/default-vlan-id	
default VLAN priority	default-priority	**PATH_layer-protocol**/default-priority	

Ingress tag filtering	ingress-tag-filtering	**PATH_layer-protocol**/ingress-tag-filtering	*Untagged and priority frames, only tagged, all frames*
Ingress VLAN filtering	ingress-vlan-id-filtering-is-on	**PATH_layer-protocol**/ingress-vlan-id-filtering-is-on	
PCP interpretation	pcp-bits-interpretation-kind	**PATH_layer-protocol/**pcp-bits-interpretation-kind	
P-bit to priority mapping	**pcp-bit-to-priority-mapping-list[to-be-decoded-pcp-bits-value]**	/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration/pcp-bit-to-priority-mapping-list	
	to-be-decoded-pcp-bits-value	**PATH_pcp-bit-to-priority-mapping-list**/to-be-decoded-pcp-bits-value	*0 to 7*
	associated-priority-value	**PATH_pcp-bit-to-priority-mapping-list**/associated-priority-value	*0 to 7*
	associated-drop-eligibility	**PATH_pcp-bit-to-priority-mapping-list/**associated-drop-eligibility	
Priority to p-bit mapping	**pcp-bits-encoding-mapping-list[to-be-encoded-priority-value to-be-encoded-drop-eligibility]**	/core-model:control-construct/logical-termination-point/layer-protocol/vlan-interface:vlan-interface-pac/vlan-interface-configuration/pcp-bits-encoding-mapping-list	
	to-be-encoded-priority-value	**PATH_pcp-bits-encoding-mapping-list**/to-be-encoded-priority-value	*0 to 7*
	associated-pcp-bits-value	**PATH_pcp-bits-encoding-mapping-list**/associated-pcp-bits-value	*0 to 7*
	to-be-encoded-drop-eligibility	**PATH_pcp-bits-encoding-mapping-list**/to-be-encoded-drop-eligibility	

Other configuration parameters are available at the configuration class of the VLAN interface to enable more complex use cases that require additional details in relation to vlan ide internal and egress mappings, service access prioritization, mapping to traffic classes or priority overwriting.

At VLAN-FD level, the relevant configuration parameters are:

USE CASE INFO BLOCK	*MODEL CLASS/ PARAMETER*	*REQUIRED PATH*	*COMMENTS*
NE ID FIELDS	uuid	/core-model:control-construct/uuid	
	name[value-name]	/core-model:control-construct/name	*List of value-name & name pairs. More than one element may be specified with different value-names. As example, one pair value-name="externalLabel" & name="XXXX" matching the external label implemented by the operator for the equipment*
VLAN switch ID and related LTPs	**forwarding-domain[uuid]**	***/core-model:control-construct/forwarding-domain***	
	uuid	**PATH_forwarding-domain**/uuid	
	name[value-name]	**PATH_forwarding-domain**/name	
	layer-protocol-name	**PATH_forwarding-domain**/layer-protocol-name	*protocol layer (LAYER_PROTOCOL_NAME_TYPE_VLAN_LAYER)*

	logical-termination-point	**PATH_forwarding-domain**/logical-termination-point	*list of LTPS that constitute this potential domain for forwarding at the specific protocol layer*
VLAN SWITCH CONFIGURATION	**vlan-fd-configuration**	***/core-model:control-construct/forwarding-domain/vlan-fd:vlan-fd-pac/vlan-fd-configuration***	
	fd-name	**PATH_vlan-fd-configuration**/fd-name	*operator naming*
	sub-layer-protocol-name	**PATH_vlan-fd-configuration**/sub-layer-protocol-name	*C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT , undefined*
	mac-address	**PATH_vlan-fd-configuration**/mac-address	*MAC address configured for bridge component*
	traffic-classes-is-on	**PATH_vlan-fd-configuration**/traffic-classes-is-on	*when disable, single priority to all traffic in the bridge*

To provision a new VLAN between ports of the managed NE, ONF model includes specific RPCs within the VLAN-fd and VLAN-fc PACs, which will shall serve to set up and configuring specific parameters of the service. To create a new L2 VLAN with a specific VLAN ID between two specific ports after having configured VLAN interfaces, two steps (each linked to a specific RPC) are required:

First, a new instance of FC needs to be added to the list within the desired FD representing the VLAN bridge. To do this, the create-vlan-fc RPC within the VLAN-FD PAC is required:

RPC:	Type	Description
+---x create-vlan-fc		Creation of a new VLAN with an specific VLAN ID in the required VLAN-FD. The new FC will be created with the same sub-layer-protocol of the VLAN-FD
| +---w input		
| | +---w affected-vlan-fd?	leafref	-> /core-model:control-construct/forwarding-domain/uuid
| | +---w new-vlan-id?	uint64	
| +--ro output		
| +--ro created-vlan-fc?	leafref	-> /core-model:control-construct/forwarding-domain/fc/uuid

This RPC creates a new VLAN of the same sub layer protocol of the referenced VLAN Bridge. As inputs, a reference to the VLAN Bridge and the VLAN ID shall be provided. As output, the reference to the uuid of the newly created FC within the FD. The resulting VLAN will not yet connect to any interface, this will be subject of the second step and RPC. In case of error, harmonized responses are defined:

- #[onf:VLAN ID not available.]# if the VLAN ID is either out of range of configurable values or occupied by some special service
- #[onf:Referenced object is invalid.]# if the referenced Forwarding Domain does not exist or does not belong to the VLAN layer
- #[onf:Resources not available.]# if e.g. the VLAN table would already be full

Second, the new VLAN within the FD, needs to be connected to the desired interfaces. This will require creating new instances of FC-ports within the desired FC, with references to the target VLAN interfaces that are desired to be connected.

RPC:	Type	Description
+---x create-vlan-fc-port		Adds a VLAN interface of the same sub layer to an already existing VLAN (VlanFc)
| +---w input		

| | +---w affected-vlan-fc?	Leafref	-> /core-model:control-construct/forwarding-domain/fc/uuid
| | +---w associated-vlan-interface?	leafref	-> /core-model:control-construct/logical-termination-point/uuid
| +--ro output		
| +--ro created-vlan-fc-port? string	string	

This RPC (within the VLAN-FC PAC) adds a VLAN interface of the same sub layer to an already existing VLAN (Vlan FC). The references of the VLAN (Vlan FC) and the to be connected VLAN interface are required as input. As an output, the RPC provides a string identifying the newly created fc-port (local-id). In case of error, harmonized messages are defined:

- #[onf:Referenced object is invalid.]# if the referenced ForwardingConstruct or LogicalTerminationPoint does not exist or does not belong to the VLAN layer and same sub layer (e.g. C-VLAN or S-VLAN)
- #[onf:VLAN interface cannot be added.]# if the referenced VLAN interface cannot be added to the referenced VLAN (VlanFc)

So, to complete the VLAN creation between two ports, this step needs to be repeated twice to add two fc-ports linked to the desired VLAN interfaces. Typically, this completes the VLAN creation stage.
Specific configuration parameters of the VLAN-FC are:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*REQUIRED PATH*	*COMMENTS*
VLAN CONFIGURATION	vlan-fc-configuration	*/core-model:control-construct/forwarding-domain/fc/vlan-fc:vlan-fc-pac/vlan-fc-configuration*	
	fc-name	**PATH_vlan-fc-configuration**/fc-name	***operator naming,*** *optional on top of the VLAN ID*
	sub-layer-protocol-name	**PATH_vlan-fc-configuration**/sub-layer-protocol-name	***VLAN configured Component type:*** *C_VLAN_COMPONENT, S_VLAN_COMPONENT, D_BRIDGE_COMPONENT, EDGE_RELAY_COMPONENT, undefined*
	vlan-id	**PATH_vlan-fc-configuration**/vlan-id	***VLAN ID***

VLAN deletion is also a process based on specific RPCs designed for that, within the VLAN FD and VLAN FC ONF PACs. As for VLAN creation, one RPC decouples the interfaces from the FC-ports and deletes the FC-ports of the specific VLAN ID from the FC in which they were created, and another one deletes the VLAN (deleting the FC from the FD in which it was created)

RPC:	Type	Description
+---x delete-vlan-fc-port		
+---w input		
+---w affected-vlan-fc?	leafref	-> /core-model:control-construct/forwarding-domain/fc/uuid
+---w obsolete-vlan-interface?	leafref	-> /core-model:control-construct/logical-termination-point/uuid

This first deletion RPC (within the VLAN-FC PAC) will remove a specific VLAN interface from an existing VLAN (VlanFc). The references of the VLAN (Vlan FC) and the VLAN interface that needs to be disconnected must be provided as input. When the operation fails, harmonized error messages need to be considered:

- #[onf:Referenced object is invalid.]# if the referenced ForwardingConstruct or LogicalTerminationPoint does not exist or does not belong to the VLAN layer and same sub layer (e.g. C-VLAN or S-VLAN)
- #[onf:VLAN interface cannot be disconnected.]#, if the referenced VLAN interface cannot be removed from the referenced VLAN (VlanFc)

The RPC will be repeated for the VLAN interfaces that need to be disconnected.

RPC:	Type	Description
+---x delete-vlan-fc		
+---w input		
+---w affected-vlan-fd?	leafref	-> /core-model:control-construct/forwarding-domain/uuid
+---w obsolete-vlan-fc?	leafref	-> /core-model:control-construct/forwarding-domain/fc/uuid

This second deletion RPC (within the VLAN-FD PAC) removes an existing VLAN (FC) from the desired VLAN bridge (FD). It includes automatically removing all interfaces from that VLAN before doing this. The following error-messages are to be send, if the operation fails:

- #[onf:VLAN cannot be deleted.]#, e.g. if the VLAN is required for some special service.
- #[onf:Referenced object is invalid.]#, if the referenced ForwardingConstruct or does not belong to the VLAN layer

As related workflow aspects for the use case implementation, three general steps/actions may be considered:

1. **Gerenal VLAN switch and VLAN interfaces configuration**
2. **New VLAN Creation**:
    a. VLAN-fd RPC to create VLAN: create-vlan-fc
    b. VLAN-fc RPC to create the VLAN ports in the new fc and attach to specific VLAN interfaces: create-vlan-fc-port
3. **Existing VLAN complete deletion**:
    a. VLAN-fd RPC to delete an existing VLAN: delete-vlan-fc

It is also possible to delete individually fc ports for reconfiguration actions, using the VLAN-fc available RPC: delete-vlan-fc-port.

2.3 Performance Analysis & Prediction

2.3.1 Use Case 3.1 - Air interface (carrier) signal status

Use Case	Air Interface (Carrier) Signal Status
Id	Use Case 3.1
Summary	Extraction of the Instantaneous status parameters of one or more air interfaces of the NE (Typically SNR, RSL, Transmit power, Power status, Transmission mode like XPD or XPI,Link latency, Feature activation state.
Benefits and Motivation	The main motivation is enabling a much more advanced and efficient network analysis and management based on intelligent algorithms, leveraging automation and ensuring an optimum usage of network and spectrum resources, reducing the exposure to risks of network failures and enhancing availability. Homogeneous extraction of the current state of the air interfaces in any node of a multi-vendor network becomes one of the key building blocks of advanced automation and AI powered algorithms. State information of the air interfaces complements other general use cases like inventory or planning reconciliation, where the current values of radio parameters like power, frequency, etc. or state of activation of features can be obtained and displayed together with the planning view of configuration or on top of the desired microwave network clusters. Other applications like the acceptance of new radio link installations can also make use of this information in order to to check or even perform automatic validation test, changing configuration parameters of the radio interface and polling the status of the interface to take measurements and conduct the process. Status retrieval also sets the base of complex applications like automated troubleshooting, complementing PM, configuration and alarm information coming from the network elements in the network. The possibility of developing all these advanced applications on top of a single agnostic SDN domain controller constitutes and extra benefit, avoiding the dependency on multiple parallel systems, simplifying and raising the efficiency of network planning and operational processes.

Detailed description	An SDN MW agnostic controller shall be able to retrieve in real time the current state information of any (or all) the air interfaces within a NE under control. The main parameters identified as targets are:
	• **Interface state:** this group will consists of all the parameters needed to identify if an interface is up and working properly. As a minimum, the interface status (up / down / etc…) needs to be available. Other related parameters that may be available, like those showing if, on top of the interface state, the link between source and destination are working properly, or the powering state depending on the availability of them in the models are desirable and can be considered here.
	• **Feature activation state:** this group will include those that show the current state of activation of the interface typical features.. As an example:
	  ○ ACM
	  ○ ATPC
	  ○ XPIC
	• **Radio parameter state:** In this category, all the state information related to radio parameters, which is typically key for most of the aforementioned applications, both expanding the planned / configured inventory view with actual and providing the real time radio interface information for automation applications. Within scope of the use case:
	  ○ Transmitted frequency
	  ○ Received frequency
	  ○ Current modulation or transmitted mode, for instantaneous radio capacity calculations
	  ○ Transmitted power
	  ○ Receive power
	  ○ Signal to interference ratio
	  ○ Cross polar discrimination factor
	  ○ RF temperature

Proposed implementation ONF -Data models

Implementation of this use case relies on ONF CIM classes (logicalTerminationPoint and layerProtocol lists) and the support of a specific TR-532 PAC, *AirInterface_2.0.0,* with resources (papyrus UML, overview and documentation) available here and latest yang module *air-interface-2-0* here . General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3.

LTPs correspoinding to an air interface (carrier) will include a layer-protocol instance having a layer-protocol-name=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER. The air-interface pac attaches to the layer-protocol CIM providing a conditional augmentation with the MW specific classes and parameters in case this condition applies.

In relation to the use case, the Air Interface PAC includes a status class which integrates all the parameters relevant for the use case, including both feature state information and instantaneous measures corresponding to the radio aspects of the NE carriers. The model considers separation of state and configuration parameters.

```
module: core-model-1-4
  +--rw control-construct!
     +--...
     +--rw logical-termination-point* [uuid]
     |  +--rw uuid
     |  +--rw name* [value-name]
     |  +--rw server-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--rw client-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--…
     |  +--rw layer-protocol* [local-id]
     |  |  +--rw local-id
     |  |  +--rw layer-protocol-name?
(=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER)
     |  |  +--...
     |  |  +--rw air-interface:air-interface-pac
     |  |     +--…
     |  |     +--rw air-interface:air-interface-status
     |  |     +--…
```

The next table summarizes the relevant parameters for the use case, including both general interface identification parameters from the CIM classes and the specific air interface information within the air-inteface PAC.

USE CASE INFO BLOCK	MODEL CLASS / PARAMETER	PATH	comments
ID FIELDS	**Logical-termination-point[uuid]**	*/core-model:control-construct/logical-termination-point*	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point**/name	*additional field for labeling the LTP according to operator rules.* *Each item of the list includes "value" and "value-name" so more than one label is possible*

	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol layer LAYER_PROTOCOL_NAME _TYPE_AIR_LAYER*
	air-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-configuration***	
	air-interface-name	**PATH_air-interface-configuration**/air-interface-name	
	remote-air-interface-name	**PATH_air-interface-configuration**/remote-air-interface-name	
	air-interface:air-interface-status	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-status***	
INTERFACE STATE	air-interface:interface-status	**PATH_air-interface-status**/interface-status	
	air-interface:link-is-up	**PATH_air-interface-status**/link-is-up	
	air-interface:radio-power-is-up	**PATH_air-interface-status**/radio-power-is-up	
FEATURE ACTIVATION STATE	air-interface:atpc-is-up	**PATH_air-interface-status**/atpc-is-up	
	air-interface:xpic-is-up	**PATH_air-interface-status**/xpic-is-up	
RADIO PARAMETER STATE	air-interface:tx-frequency-cur	**PATH_air-interface-status**/tx-frequency-cur	
	air-interface:rx-frequency-cur	**PATH_air-interface-status**/rx-frequency-cur	
	air-interface:transmission-mode-cur	**PATH_air-interface-status**/transmission-mode-cur	*channel bandwidth, physical modulation, coding rate, symbol rate reduction factors, etc. can be retrieved for this transmission mode from the transmission model list (see carrier inventory use case)*
	air-interface:rx-level-cur	**PATH_air-interface-status**/rx-level-cur	
	air-interface:tx-level-cur	**PATH_air-interface-status**/tx-level-cur	
	air-interface:snir-cur	**PATH_air-interface-status**/snir-cur	
	air-interface:xpd-cur	**PATH_air-interface-status**/xpd-cur	
	air-interface:rf-temp-cur	**PATH_air-interface-status**/rf-temp-cur	*RF temperature*

Addressing a single air interface should be possible via filtering using first the control-construct uuid, then the LTP uuid and, finally, the layer protocol local-id.

In addition to the specific interface-status and link-is-up parameters available at the TR-532 air interface PAC status class, the core model objects that serve as the base for the technology specific augment (the logical-termination-point and layer-protocol corresponding to the desired interface also include specific *"operational-state"* parameters which can also be used complementing the former. It is mandatory that consistency between core model status parameters and TR-532 interface specific parameter exists. The same applies to the core model

"operational-state" parameters that exist for any of the equipments that serve as support of . These rules (as well as the fields required for identification of objects like equipment or LTPs) are as specified in the transmitterEquipment specification v1.0 https://github.com/openBackhaul/equipment/tree/tsp with which the ONF implementation needs to be compliant.

Parameters can be accessed in a single step, so need for a quite specific workflow. When data for all the radio interfaces available at the network element are needed, they can be extracted in several subsequent requests, each individual to the specific interface or in a single request from the controller for all interfaces, to be then post processed at application level.

Proposed implementation IETF/IEEE - Data models

Based on the the YANG module "ietf-microwave-radio-link", defined in the IETF RFC 8561.

Per Radio Interface

```
          +--rw radio-link-protection-groups
              +--rw protection-group* [name]
          +--rw xpic-pairs {xpic}?
              +--rw xpic-pair* [name]
          +--rw mimo-groups {mimo}?
              +--rw mimo-group* [name]
    augment /if:interfaces/if:interface:
```

Parameter	Type / Sub-parameter	Description / Application (indicative)
+--rw carrier-id	string	Carrier ID
+--ro uuid / +--ro if-index	yang:uuid / int32	Termination point / interface ID and HW supporting the interface
+--ro capabilities +--ro max-tx-frequency +--ro max-rx-frequency	uint32	Max frequency
+--ro capabilities +--ro min-tx-frequency +--ro min-rx-frequency	uint32	Min Frequency
+--rw (freq-or-distance) +--rw duplex-distance	int32	Duplex seperation
+--ro capabilities +--ro available-max-acm	identityref	Max Modulation
+--ro capabilities +--ro available-min-acm	identityref	Min Modulation
+--rw channel-separation	uint32	Bandwidth (Channel Size)
+--rw (coding-modulation-mode) *+--:(single)*	identityref	Fixed Modulation & Coding

+--rw single +--rw selected-cm (Note 1)		
+--rw (coding-modulation-mode) *+--:(adaptive)* +--rw adaptive		Adaptive Modulation & Coding
+--rw xpic-pairs {xpic} +--rw xpic-pair* [name] +--rw name +--rw enabled? +--rw members	-> /xpic-pairs/xpic-pair/name string boolean if:interface-ref	XPIC Group
+--ro actual-transmitted-level	power	Tx Power Level
+--ro actual-received-level	power	Rx Sensitivity
+--ro actual-snir	decimal64	SNIR
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)* +--rw atpc		ATPC
+--rw radio-link-protection-groups *+--rw protection-group* [name]* +--rw name +--rw protection-architecture-type +--rw members +--rw operation-type +--rw working-entity +--rw revertive-wait-to-restore +--rw hold-off-timer +--ro status	string identityref if:interface-ref enumeration if:interface-ref uint16 uint16 identityref	Space Diversity (SD)

+--rw tx-enabled	boolean	RF Carrier on/off
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)* *+--rw atpc* +--rw maximum-nominal-power	power	Tx power Configuration
+--rw (power-mode) *+--:(rtpc)* *+--:(atpc)* *+--rw atpc* +--rw atpc-lower-threshold +--rw atpc-upper-threshold	power power	ATPC Thresholds
+--ro capabilities +--ro available-min-acm	identityref	Min physical modulation
+--ro capabilities +--ro available-max-acm	identityref	Max physical modulation

Note 1: In case of adaptive mode, this attribute is not defined.

2.3.2 Use Case 3.2 - Air interface (carrier) PM counters

Use Case	Air interface (carrier) PM counters
Id	Use Case 3.2
Summary	Extraction of the history of performance monitoring (PM) counter sets that apply to the air interfaces of any of the network elements under control of an SDN MW agnostic controller. Typical measurements representing the radio behavior of a link like SNR, RSL, transmit power, transmission mode distribution, etc. are the focus of the present use case.
Benefits and Motivation	Together with alarm information, PM counters constitute one of the main sources of information used in operational procedures in any domain of the operator networks. Specific OSS systems store and process vast amounts of performance measurements of different granularity, generating elaborated KPIs, and complex reports which are used to define and trigger specific operations on the network equipment as re-configurations, to identify causes impacting network performance optimizing corrective actions, and quality of experience, as well as to provide input that is used to carry out and develop the recurrent network upgrade and expansion plans. While the PM information typically spans multiple interfaces and parts of the network elements, this use case focuses on air interfaces, which are the more characteristic within the MW transport domain. Similar use cases can and will be developed for other interfaces (like wired ports) and protocol layers (like Ethernet traffic, with RMON counters being key in this respect). A key benefit of a harmonized SBI to retrieve the PM information is the simplifcation the OSS layer,

	that through a single element like the SDN controller will have a single way (leveraging also on common NBIs) to retrieve and integrate in the OSS data lakes the history of network performance of all the network elements under control. Not only simplification, but also evolution towards increased automation becomes a relevant benefit and target of the use case. On top of PM information, OSS systems or specific domain advanced AI applications may be developed to bring an extra level of efficiency and automation to the operational processes. AI based analysis of counters, cross correlation with alarm information, pattern and trend identification for corrective and predictive analysis, to anticipate congestion and carry out automated planning of network expansions and link/cluster reconfiguration are some of the examples of applications that will feed from this information
Detailed description	An SDN MW agnostic controller shall be able to retrieve the available history (or a single set, if desired) of PM sets corresponding to any (or all) of the air interfaces of a network element under control. The same applies to the current monitoring period available at the NE. When sets corresponding to different granularities are available, it shall be possible as well to retrieve all or any of them for the specific period under consideration. Considering the typical monitoring processes as well as future applications, the main air interface related PM counters considered for the use case are: ○ **Air interface errors and availability counters** ▪ Errored seconds (ES / SES / CSES) and blocks ▪ Availability / unavailability (UAS) ○ **Power, interference and other radio related counters** ▪ Transmitted power ▪ Receive power ▪ Signal to interference ratio ▪ Cross polar discrimination factor ▪ RF temperature ○ **Modulation or transmission mode distribution,** for radio capacity calculations over the period Availability of the previous parameters with different aggregation in the period (maximum / minimum / average / seconds distribution) benefits the use case implementation. Other “non-radio” parameters also relevant for the use case: • Identification fields of the NE and interfaces • Considering that there may be different persistence in NEs from different vendors, parameter(s) showing the number of stored PM sets are also relevant for the use case. • To differentiate between sets of different granularity, (a) dedicated parameter(s) indicating the type of set (15 minutes, 24 hours, ...) or period length (typically in seconds)
Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes (logicalTerminationPoint and layerProtocol lists) and the support of a specific TR-532 PAC, *AirInterface_2.0.0,* with resources (papyrus UML, overview and documentation) available here and latest yang module *air-interface-2-0* here . General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3. LTPs correspoinding to an air interface (carrier) will include a layer-protocol instance having a layer-protocol-name=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER. The air-interface pac attaches to the layer-protocol CIM providing a conditional augmentation with the MW specific classes and parameters in case this condition applies. The air Interface PAC includes two classes relevant in relation to the performance counters. Current performance class is related to the measurements for the current 15/24H period (that can be retrieved before the period is finished), while the historical performance class relates to the

history of past 15M / 24H completed periods which are available in the network element. PM counters available at both classes are the same, with the differences being related to the nature of the period (currently ongoing vs. completed historical ones).

```
module: core-model-1-4
  +--rw control-construct!
     +--...
     +--rw logical-termination-point* [uuid]
     |  +--rw uuid
     |  +--rw name* [value-name]
     |  +--rw server-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--rw client-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--…
     |  +--rw layer-protocol* [local-id]
     |  |  +--rw local-id
     |  |  +--rw layer-protocol-name?
(=LAYER_PROTOCOL_NAME_TYPE_AIR_LAYER)
     |  |  +--...
     |  |  +--rw air-interface:air-interface-pac
     |  |     +-- ro air-interface:air-interface-capability
     |  |     +-- rw air-interface:air-interface-configuration
     |  |     +-- ro air-interface:air-interface-status
     |  |     +--ro air-interface:air-interface-current-
performance
     |  |     +--ro air-interface:air-interface-historical-
performances
     |  |     +--…
```

Additionally, capability, configuration and status classes include specific parameters for respectively showing the potential of the interface to support PM measurements, enabling/disabling the PM recording at the interface[1] and show the current status of activation of PM.

USE CASE INFO BLOCK	MODEL CLASS / PARAMETER	PATH	comments
INTERFACE ID	Logical-termination-point[uuid]	*/core-model:control-construct/logical-termination-point*	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is augmented with the TR-532 air-interface PAC*
	name [value-name]	**PATH_logical-termination-point** /name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*

[1] *In the specific case of the configuration, devices in many cases don't allow for the activation at interface level, but on a network element level, so future changes in the model related to this are an open aspect.*

	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol layer LAYER_PROTOCOL_NAME _TYPE_AIR_LAYER*
	air-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-configuration***	
	air-interface-name	**PATH_air-interface-configuration**/air-interface-name	
PM MANAGEMENT	**air-interface-capability**	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-capability***	
	performance-monitoring-is-avail	**PATH_air-interface-capability**/performance-monitoring-is-avail	*availability of PM at the air interface*
	air-interface-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-configuration***	
	performance-monitoring-is-on	**PATH_air-interface-configuration**/performance-monitoring-is-on	*PM activation*
	air-interface:air-interface-status	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-status***	
	performance-monitoring-is-up	**PATH_air-interface-status**/performance-monitoring-is-up	*PM activation state*

The next table shows the summary of the model parameters that are required for the air interface PM use case implementation. The table focuses on historical performance (in terms of supported parameters, current and historical performance classes are the same. Differences are on specific fields corresponding to the nature of the data, as will be detailed later).

```
|   |  +--rw air-interface:air-interface-pac
     |   |     +--…
     |   |     +--ro air-interface:air-interface-historical-
performances
     |   |        +--ro air-interface:number-of-historical-
performance-sets?   int16
     |   |        +--ro air-interface:time-of-latest-change?
yang:date-and-time
     |   |        +--ro air-interface:historical-performance-
data-list* [granularity-period period-end-time]
     |   |        |  +--ro air-interface:suspect-interval-flag?
boolean
     |   |        |  +--ro air-interface:history-data-id?
string
     |   |        |  +--ro air-interface:granularity-period
granularity-period-type
     |   |        |  +--ro air-interface:period-end-time
yang:date-and-time
     |   |        |  +--ro air-interface:performance-data
     |   |        |  |  +--…
```

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
PM SET INFORMATION	**air-interface-historical-performances**	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-historical-performances***	***general container for the PM history of the NE. Includes general fields showing the available sets and latest change date***
	number-of-historical-performance-sets	**PATH_air-interface-historical-performances/**number-of-historical-performance-sets	*number of PM available sets in the NE*
	time-of-latest-change	**PATH_air-interface-historical-performances/**time-of-latest-change	
PM PERIOD INFORMATION	**historical-performance-data-list[granularity-period period-end-time]**	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-historical-performances/historical-performance-data-list***	***List that includes the PM data as well as the paramenters that characterize the period (15M/24H, time, etc.)***
	granularity-period	**PATH_historical-performance-data-list/**granularity-period	*indicates period of the counter 15M /24H*
	period-end-time	**PATH_historical-performance-data-list/**period-end-time	*date-time format that identifies the period (end) to which the PM belongs to. Together with the granularity-period constitutes the key of the historical-performance-data-list*
	history-data-id	**PATH_historical-performance-data-list/**history-data-id	
	suspect-interval-flag	**PATH_historical-performance-data-list/**suspect-interval-flag	*marks periods with potentially inconsistent data*
PM DATA	**performance-data**	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-historical-performances/historical-performance-data-list/performance-data***	***container that includes all the PM counters available in the given period***
AVAILABILITY AND ERRORS	es	**PATH_performance-data/**es	
	ses	**PATH_performance-data/**ses	
	cses	**PATH_performance-data/**cses	
	unavailability	**PATH_performance-data/**unavailability	
	defect-blocks-sum	**PATH_performance-data/**defect-blocks-sum	
TRANSMITTER POWER	tx-level-min	**PATH_performance-data/**tx-level-min	*Minimum in the period*
	tx-level-max	**PATH_performance-data**/tx-level-max	*Maximum in the period*
	tx-level-avg	**PATH_performance-data**/tx-level-avg	*Average value for the period*
RECEIVED POWER	rx-level-min	**PATH_performance-data**/rx-level-min	*Minimum in the period*
	rx-level-max	**PATH_performance-data**/rx-level-max	*Maximum in the period*
	rx-level-avg	**PATH_performance-data**/rx-level-avg	*Average value for the period*
SNR	snir-min	**PATH_performance-data**/snir-min	*Minimum in the period*
	snir-max	**PATH_performance-data**/snir-max	*Maximum in the period*
	snir-avg	**PATH_performance-data**/snir-avg	*Average value for the period*

CROSS POLAR (XPD)	xpd-min	**PATH_performance-data**/xpd-min	*Minimum in the period*
	xpd-max	**PATH_performance-data**/xpd-max	*Maximum in the period*
	xpd-avg	**PATH_performance-data**/xpd-avg	*Average value for the period*
TEMPERATURE (RF)	rf-temp-min	**PATH_performance-data**/rf-temp-min	*Minimum in the period*
	rf-temp-max	**PATH_performance-data**/rf-temp-max	*Maximum in the period*
	rf-temp-avg	**PATH_performance-data**/rf-temp-avg	*Average value for the period*
PERIOD	time-period	**PATH_performance-data** /time-period	number of seconds of the period
TRANSMISSION MODE DISTRIBUTION	***time-xstates-list[time-xstate-sequence-number]***	***/core-model:control-construct/logical-termination-point/layer-protocol/air-interface:air-interface-pac/air-interface-historical-performances/historical-performance-data-list/performance-data/time-xstates-list***	***this list is included within the performance data registering the distribution of transmission modes in the period. Distribution is obtained as seconds of each transmission mode within to overall period. Base for carrier capacity calculations in each period***
	time-xstate-sequence-number	**PATH_time-xstates-list**/time-xstate-sequence-number	*sequence number*
	transmission-mode	**PATH_time-xstates-list**/transmission-mode	*transmission mode (details like channel size, modulation, coding rate, symbol rate reduction, etc.can be extracted from the transmission mode list in the air interface capabilities)*
	time	**PATH_time-xstates-list**/time	*number of seconds of the period in which the carrier operated in the transmission mode*

Addressing a single air interface should be possible via filtering using first the control-construct uuid, then the LTP uuid and, finally, the layer protocol local-id. The complete PM of all the available air interfaces of the network element can also be retrieved with a single request.

To extract the current PM information, the air-interface-current-performance class is needed. In terms of performance data, the class is the same than the historical, and most of the commented aspects for implementation hold. The key differences relevant for the use case description being:

- The number of current sets will be 0-2 (assuming typical 15 minutes / 24 hour sets). Only one set per granularity as maximum should be available in the NE.
- A timestamp and an elapsed time (seconds within the period) replace the period-end-time of the historical class, to be able to support requests at any time within the period and indicate the total time for KPI calculations.

```
|  |  +--rw air-interface:air-interface-pac
     |  |     +--…
     |  |     +--ro air-interface:air-interface-current-
performance
     |  |     |  +--ro air-interface:number-of-current-
performance-sets?   int8
     |  |     |  +--ro air-interface:current-performance-data-
list* [granularity-period]
     |  |     |  |  +--ro air-interface:timestamp?
yang:date-and-time
     |  |     |  |  +--ro air-interface:suspect-interval-flag?
boolean
```

```
     |  |     |  |  +--ro air-interface:elapsed-time?
int64
     |  |     |  |  +--ro air-interface:scanner-id?
string
     |  |     |  |  +--ro air-interface:granularity-period
granularity-period-type
     |  |     |  |  +--ro air-interface:performance-data
  || |||+--...
```

Parameters can be retrieved in a single step by the SDN controller, so need for a specific workflow. When data for all the radio interfaces available at the network element are needed, they can be extracted in N subsequent requests, each individual to the specific interface or in a single request from the controller. It is also possible to retrieve an specific available set filtering using the historical performance data keys.

Proposed implementation IETF/IEEE -Data models

Based on the the YANG module "ietf-microwave-radio-link", defined in the IETF RFC 8561.

Per Radio Interface

```
            +--rw radio-link-protection-groups
                +--rw protection-group* [name]
            +--rw xpic-pairs {xpic}?
                +--rw xpic-pair* [name]
            +--rw mimo-groups {mimo}?
                +--rw mimo-group* [name]
        augment /if:interfaces/if:interface:
```

Parameter	Type / Sub-parameter	Description / Application (indicative)
+--rw carrier-id	string	Carrier ID
+--rw (coding-modulation-mode) *+--:(single)* *+--rw single* +--rw selected-cm (Note 1)	identityref	Fixed Modulation
+--rw (coding-modulation-mode) *+--:(adaptive)* +--rw adaptive		Adaptive Modulation
+--ro capabilities +--ro available-max-acm	identityref	Maximum Physical Modulation
+--ro capabilities +--ro available-min-acm	identityref	Minimum Physical Modulation
+--ro actual-tx-cm	identityref	Current Physical Modulation
+--ro actual-transmitted-level	power	Tx Power
+--ro actual-received-level	power	RX Sensitivity

+--ro actual-snir	decimal64	SNIR
+--ro error-performance-statistics +--ro es	yang:counter32	ES thresholds
+--ro error-performance-statistics +--ro ses	yang:counter32	SES thresholds

Note 1: In case of adaptive mode, this attribute is not defined.

2.3.3 Use Case 3.3 – Ethernet statistics/PM

Use Case	Ethernet statistics/PM
Id	Use Case 3.3
Summary	Extraction of Ethernet (RMON) state and PM information for ethernet interfaces implemented in a network element under the control of an SDN MW agnostic controller and obtention of the carrier instantaneous and historical utilization.
Benefits and Motivation	As it was already presented in use cases 2.3.1 and 2.3.2, state information and PM counters constitute one of the main sources of information used in operational procedures in any domain of the operator networks. Complementing the specific parameters that characterize the radio features and behavior of the air interfaces (power, interference, etc.), ethernet statistics and counters are another key source of information typically retrieved and stored in operator OSS systems, where it is processed together with radio information for planning and operational tasks. Equipment re-configurations and network evolution processes rely heavily on this type of information in order to be carried out properly. Ethernet traffic statistics and PM also enable, in parallel to carrier capacity instantaneous state and historical information to analyze carrier utilization, which is a key parameter to trigger link upgrades and network expansion. With a harmonized SBI to retrieve Ethernet statistics and PM, impact in relation to simplifcation the OSS systems can be achieved and, more importantly, an additional degree of automation can be achieved and applied to relevant processes like network expansion. Predictive analysis of the evolution of ethernet traffic and carrier / link utilization is an enabler of multiple applications and favors the shift towards preventive identification of bottlenecks in the network and pro-active expansion, benefitting largely network quality and bringing efficiency to the network investments, that can be prioritized automatically towards the areas and clusters which may be more critical. Cross correlation of traffic with radio statistics and PM and alarm information becomes also relevant in root cause analysis identification and dynamic management of network resources ethernet traffic pattern and trend identification also becomes the base for many other relevant applications. As some examples, those focused on bringing energy efficiency to the microwave transport networks, where the identification of periods with low utilization enable the dynamic implementation of energy saving procedures like activation of eco modes or switching of carriers in multi-carrier links, or dynamic management of quality of service in transport clusters depending on cluster-wide traffic analysis ang high/low link utilization periods.
Detailed description	An SDN MW agnostic controller must be able to retrieve in real time the ethernet related state and PM counter information for any (or all) ehernet interface within a NE under control.The main targets of the use case are related to the ethernet traffic on the interface and the utilization of the available radio capacity, which will receive the main focus. However, complementary ethernet traffic related state information (drop ratios, etc.) are usually available in the NEs and become highly desirable to provide a richer set of information to OSS systems and domain advanced applications on top of the SDN controller.

	• **NE and interface identification fields**
	• **Interface state:** this group will consists of all the parameters needed to identify if the ethernet interface is up and working properly. As a minimum, the interface status (up / down / etc…) needs to be available.
	• **Traffic and rate statistics:** this group include as key parameters those that show the transferred bytes by the interface and the current instantaneous transfer rate (in most of the cases this considers an specific period for averaging). Total transferred bytes serve to make calculations of total traffic and rates for specific periods non matching the PM typical ones, if needed.
	  o Total traffic volume (bytes in / out)
	  o Current ethernet traffic rate (bps in / out)
	• **Frame related statistics:** this group includes frame level statistics complementing byte level traffic volumes, including drop ratios, for those applications requiring a more precise traffic behavior analysis. As an option, having the visibility of the frame size distribution of the transferred traffic, which is typically available as statistic information, may serve as a complement for those advanced applications.
	  o Total frames (frames in/out)
	  o Forwarded, errored and dropped frames (frames in/out)
	  o Number of transferred frames of different typical sizes (64/128…1518)
	
	In parallel to the current statistics, the use case also requires PM related information, in the same way already described for the air interface radio PM counter use case 2.3.2. Same considerations applying also in this case. Linked to the statistics within scope, the main ethernet traffic / RMON PM counters considered for the use case are:
	• **Traffic and rate PM:**
	  o Traffic volume (bytes in/out)
	  o Minimum – Maximum ethernet traffic rate (bps in/out). Average can be calculated directly from traffic volume and counter period if not directly available.
	• **Frame related PM**
	  o Total frames (frames in/out)
	  o Forwarded, errored and dropped frames (frames in/out)
	
	It must be noted that the traffic volume (stats / PM) extracted in this use case can be combined with the air interface capacity information that can be retrieved as described in the previous use cases in order to calculate air interface utilization.

Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes (logicalTerminationPoint and layerProtocol lists), with general aspects and requirements applying to the ONF CIM given in section 3, and the support of specific TR-532 PACs: • EthernetContainer_2.0.0, with resources (papyrus UML, overview and documentation) available here and latest yang module here. • MacInterface_1.0.0, with resources (papyrus UML, overview and documentation) available here and latest yang module here. **At the time of writing, some changes have already been agreed for the TR-532 model related to RMON statistics and counters that affect these specific PACs. The main change with relation to the present use case being the re-organization of parameters between PACS, with most of the statistics and counters previously available in the MAC interface PAC (frame related parameters) now being moved to the Ethernet container PAC, for simplicity**. Parameter types and descriptions are to be kept unchanged. Implementation details given in this section **refer to the defined target (still not reflected in the repository YANGs)**, to avoid providing here information that would otherwise be obsolete in short. LTPs corresponding to the ethernet layer of an available interface (linked to ethernet ports and

their related lower level air or wire interfaces) will include a layer-protocol instance having a layer-protocol-name=LAYER_PROTOCOL_NAME_TYPE_ETHERNET_CONTAINER_LAYER. The ethernet-container pac attaches to the layer-protocol CIM class providing a conditional augmentation with the MW specific ethernet model, in case this condition applies.

```
module: core-model-1-4
  +--rw control-construct!
     +--...
     +--rw logical-termination-point* [uuid]
     |  +--rw uuid
     |  +--rw name* [value-name]
     |  +--rw server-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--rw client-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--…
     |  +--rw layer-protocol* [local-id]
     |  |  +--rw local-id
     |  |  +--rw layer-protocol-name?
     |  |  +--...
     |  |  +--rw ethernet-container:ethernet-container-pac
     |  |     +--ro ethernet-container:ethernet-container-
capability
     |  |     +--rw ethernet-container:ethernet-container-
configuration
     |  |     +--ro ethernet-container:ethernet-container-status
     |  |     +--ro ethernet-container:ethernet-container-
current-performance
     |  |     +--ro ethernet-container:ethernet-container-
historical-performances
     |  |     +--…
```

Same applies to the MAC Interfaces, having in this case within the LTP an LP instance having a layer-protocol-name= LAYER_PROTOCOL_NAME_TYPE_MAC_LAYER. It shall be possible to relate via LTP client-server relations any MAC layer to its underlying ethernet container interface (see section 3 for LTP/LP layering aspects).

Both the Ethernet Container and the MAC Interface PACs include two classes relevant in relation to the performance counters. Current performance class is related to the measurements for the current 15/24H period (that can be retrieved before the period is finished), while the historical performance class relates to the history of past 15M / 24H completed periods which are available in the network element. Instantaneous statistics information is available within the status class.

Base classes of the core model provide identification fields for the NE, the LTPs and the layer protocols, and the PAC configuration including amongst others, additional ID information for the ethernet layer. Capability, configuration and status class also include parameters related to the activation of the PM measurements and statistics for the interface. Counter sets available at both performance classes are the same, with the differences being related to the nature of the period (currently ongoing vs. completed historical ones).

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
INTERFACE ID	**Logical-termination-point[uuid]**	***/core-model:control-construct/logical-termination-point***	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is*

			augmented with the TR-532 air-interface PAC
	name [value-name]	**PATH_logical-termination-point** /name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol**/local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol layer LAYER_PROTOCOL_NAME_TYPE_ETHERNET_CONTAINER_LAYER*
	ethernet-container-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-configuration***	
	ethernet-container-name	**PATH_ethernet-container-configuration**/ethernet-container-name	
PM MANAGEMENT	**ethernet-container-capability**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-capability***	
	performance-monitoring-is-avail	**PATH_ethernet-container-capability**/performance-monitoring-is-avail	*availability of PM at the ethernet container*
	ethernet-container-configuration	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-configuration***	
	performance-monitoring-is-on	**PATH_ethernet-container-configuration**/performance-monitoring-is-on	*PM activation*
	ethernet-container:ethernet-container-status	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-status***	
	performance-monitoring-is-up	**PATH_ethernet-container-status**/performance-monitoring-is-up	*PM activation state*

Addressing a single ethernet interface should be possible via filtering using first the control-construct uuid, then the LTP uuid and, finally, the layer protocol local-id.

The statistics available in the status class, required for the implementation for this use case are:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
ETH TRAFFIC AND FRAME RELATED STATISTICS	**ethernet-container-status**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-status***	
INTERFACE STATUS	interface-status	**PATH_ethernet-container-status**/interface-status	*up/down/unknown/ lower_layer_down/…*

TRANSFERRED BYTES	total-bytes-input	**PATH_ethernet-container-status**/total-bytes-input	*Bytes, avg. throughput can be calculated polling this parameter subsequent times*
	total-bytes-output	**PATH_ethernet-container-status**/total-bytes-output	
TOTAL FRAMES	total-frames-input	**PATH_ethernet-container-status**/total-frames-input	***currently at MAC interface PAC, will be moved to Ethernet container (current YANG drafts still not reflecting the decision)***
	total-frames-output	**PATH_ethernet-container-status**/total-frames-output	
FORWARDED FRAMES	forwarded-frames-input	**PATH_ethernet-container-status**/forwarded-frames-input	
	forwarded-frames-output	**PATH_ethernet-container-status**/forwarded-frames-output	
ERRORED FRAMES	errored-frames-input	**PATH_ethernet-container-status**/errored-frames-input	
	errored-frames-output	**PATH_ethernet-container-status**/errored-frames-output	
DROPPED FRAMES	dropped-frames-input	**PATH_ethernet-container-status**/dropped-frames-input	
	dropped-frames-output	**PATH_ethernet-container-status**/dropped-frames-output	
FRAME SIZE DISTRIBUTION	frames-of-64-byte	**PATH_ethernet-container-status**/frames-of-64-byte	
	frames-of-65-to-127-byte	**PATH_ethernet-container-status**/frames-of-65-to-127-byte	
	frames-of-128-to-255-byte	**PATH_ethernet-container-status**/frames-of-128-to-255-byte	
	frames-of-256-to-511-byte	**PATH_ethernet-container-status**/frames-of-256-to-511-byte	
	frames-of-512-to-1023-byte	**PATH_ethernet-container-status**/frames-of-512-to-1023-byte	
	frames-of-1024-to-1518-byte	**PATH_ethernet-container-status**/frames-of-1024-to-1518-byte	
TIMESTAMP	timestamp	**PATH_ethernet-container-status**/timestamp	*for averaging and other calculations*

In relation to historical performance, the next table summarizes the relevant parameters:

```
module: ethernet-container-2-0
    +--rw ethernet-container-pac
       +--...
       +--ro ethernet-container-historical-performance
          +--ro number-of-historical-performance-sets?
          +--ro time-of-latest-change?
          +--ro historical-performance-data-list* [granularity-
period period-end-time]
          |  +--ro history-data-id?
          |  +--ro granularity-period
          |  +--ro period-end-time
          |  +--ro suspect-interval-flag?
          |  +--ro performance-data
          |  |  +--...
```

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
PM SET INFORMATION	**ethernet-container-historical-performances**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-historical-performances***	***general container for the PM history of the NE. Includes general fields showing the available sets and latest change date***

	number-of-historical-performance-sets	**PATH_ethernet-container-historical-performances**/number-of-historical-performance-sets	*number of PM available sets in the NE*
	time-of-latest-change	**PATH_ethernet-container-historical-performances**/time-of-latest-change	
PM PERIOD INFORMATION	**historical-performance-data-list[granularity-period period-end-time]**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-historical-performances/historical-performance-data-list***	***List that includes the PM data as well as the paramenters that characterize the period (15M/24H, time, etc.)***
	granularity-period	**PATH_historical-performance-data-list**/granularity-period	*indicates period of the counter 15M /24H*
	period-end-time	**PATH_historical-performance-data-list**/period-end-time	*date-time format that identifies the period (end) to which the PM belongs to. Together with the granularity-period constitutes the key of the historical-performance-data-list*
	history-data-id	**PATH_historical-performance-data-list**/history-data-id	
	suspect-interval-flag	**PATH_historical-performance-data-list**/suspect-interval-flag	*marks periods with potentially inconsistent data*
PM COUNTER DATA	**performance-data**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-historical-performances/historical-performance-data-list/performance-data***	***container that includes all the PM counters available in the given period***
THROUGHPUT (MAX)	maxBytesPerSecondOutput	**PATH_performance-data**/tx-ethernet-bytes-max-m	*bytes/second, previously named "tx-ethernet-bytes-max-m",* ***YANG still does not reflect this change***
TRAFFIC VOLUME (BYTES)	totalBytesOutput	**PATH_performance-data**/tx-ethernet-bytes-sum	*bytes, previously named "tx-ethernet-bytes-sum",* ***YANG still does not reflect this change***
TRANSMITTED FRAMES	total-frames-input	**PATH_performance-data**/total-frames-input	*frames, previously at MAC interface PAC, will be moved to Ethernet container.* ***Current yang drafts still don't reflect this decision****.*
	total-frames-output	**PATH_performance-data**/total-frames-output	
FORWARDED FRAMES	forwarded-frames-input	**PATH_performance-data**/forwarded-frames-input	
	forwarded-frames-output	**PATH_performance-data**/forwarded-frames-output	
ERRORED FRAMES	errored-frames-input	**PATH_performance-data**/errored-frames-input	
	errored-frames-output	**PATH_performance-data**/errored-frames-output	
DROPPED FRAMES	dropped-frames-input	**PATH_performance-data**/dropped-frames-input	
	dropped-frames-output	**PATH_performance-data**/dropped-frames-output	
PERIOD SECONDS	time-period	**PATH_performance-data**/time-period	

Current period PM information is available at the ethernet-container-current-performance class. In terms of performance data parameters, the class is the same than the historical, so most of the commented aspects for implementation hold, addressing the specific object. The key differences relevant for the use case description being:

- The number of current sets will be 0-2 (assuming typical 15 minutes / 24 hour sets). Only one set per granularity as maximum should be available in the NE.
- A timestamp and an elapsed time (seconds within the period) replace the period-end-time of the historical class, to be able to support requests at any time within the period and indicate the total time for KPI calculations.

```
module: ethernet-container-2-0
    +--rw ethernet-container-pac
       +--...
       +--ro ethernet-container-current-performance
       |  +--ro number-of-current-performance-sets?   int8
       |  +--ro current-performance-data-list* [granularity-
period]
       |  |  +--ro timestamp?                yang:date-and-time
       |  |  +--ro suspect-interval-flag?    boolean
       |  |  +--ro elapsed-time?             int64
       |  |  +--ro scanner-id?               string
       |  |  +--ro granularity-period        granularity-period-
type
       |  |  +--ro performance-data
       |  |  |  +--...
```

As optional information, queue utilization PM is also available at the performance data list, which can complement the present use case scope for traffic analysis applications, to identify periods with larger buffering.

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
PM HISTORY DATA	**performance-data**	***/core-model:control-construct/logical-termination-point/layer-protocol/ethernet-container:ethernet-container-pac/ethernet-container-historical-performances/historical-performance-data-list/performance-data***	
QUEUE UTILIZATION	**queue-utilization-list[queue-name]**	**PATH_performance-data/**queue-utilization-list	
	queue-name	**PATH_performance-data/**queue-utilization-list/queue-name	
	max-queue-length	**PATH_performance-data/**queue-utilization-list/max-queue-length	
	avg-queue-length	**PATH_performance-data/**queue-utilization-list/avg-queue-length	

Inclusion of newer parameters, as supported by NE vendors is evaluated periodically. Latency, packets or bytes per queue and others are under discussion and may be supported as part of the natural evolution of the model..

Parameters can be accessed in a single step, so need for a specific workflow. When data for all the ethernet interfaces available at the network element are needed, they can be extracted in N subsequent requests, each individual to the specific interface or in a single request from the controller.

Proposed implementation

IETF/IEEE -Data models	

2.4 Smart Alarm Analysis & Fault Prediction

2.4.1 Use Case 4.1 – Current Alarms

Use Case	Current Alarms
Id	Use Case 4.1
Summary	Use case focuses on the retrieval of NE current alarms, including equipment, radio layer and service layers.
Benefits and Motivation	Operators deploy and operate specific systems to manage alarms in all the network domains. While alarm definition follow in many cases global standards, many parallel specific interfaces need to be considered to the vendor specific management platforms which generates in many cases inefficiencies, complexity to evolve systems and difficulties to build enhanced alarm applications, especially those involving different technologies within a network domain and E2E, across domains. In many cases, particularization or deep specific knowledge around the managed technologies and their differences is required, slowing down troubleshooting or new application development. Main motivation is to achieve a harmonized smart fault management application across a MW/mmWave multi-vendor network, which will lead progressively towards a fully automated and less time-consuming troubleshooting process. AI-based alarm correlation applications or root cause automatic analysis, combining alarms with other sources of information in the operators data lake are relevant examples of the future targets.
Detailed description	The SDN agnostic controller shall be able to retrieve the list of active alarms at any specific interface of an NE within the control domain, as well as receiving asynchronously the alarms triggered by any specific failure or related event. The network element shall inform about the supported alarms for their specific interfaces as well as their related severities, that shall be configurable. The alarms must include a common set of parameters to characterize the event. Aside network element / interface ID fields, key parameters for the use case: • **Event ID**: the alarm must have a sequence number or id that serves to identify the event. • **Alarm id:** ID fields must include a descriptive name for the alarm. Other like numeric IDs, etc can be added additionally. • **Alarm time**: each alarm will include a timestamp showing the time in which the event occurred. • **Alarm severity:** as described in X.733 (warning/minor/major/critical/clear/undefined). Severity of the supported alarms must be configurable • **Event type**: extending the previous fields, showing the category of the alarm, as described in X.733. • **Probable cause**: further extending the previous fields, defining a probable cause for the alarm, as tas described in X.733. These fields are relevant not only for the retrieval of active alarms, but also as parameters that should be part of the asynchronous notifications to the controller. In the case of the notifications, the parameters of the raises and clears shall include all the fields necessary to identify uniquely the object/interface generating it as well as to correlate correctly raises with clears. Configurable TCAs (threshold crossing alarms) for air interfaces also become a highly desirable element, especially for G.826 error performance events and radio power (transmit/receive) events.
Proposed implementation ONF -Data models	Implementation requires support of ONF TR-512 v1.4 core model (version pruned and refactored for MW network elements) as documented in: https://github.com/openBackhaul/core/tree/tsp. The core model serves as the base for the MW technology specific augments (PACs) that form the TR-532 ONF microwave model.

Implementation of this use case relies on ONF CIM classes, TR-532 PACs and the support of NETCONF notifications as defined in RFC5277. General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3, as well as a reference of TR-532 PACs and paths to available resources in the development Github.

Alarms in the ONF MW model are related to the available interfaces in the NE. TR-532 PACs model the MW technology specific aspects of these interfaces concentrating most of the functionality related to alarms in the current-problem class. Additionally, capability class shall include the list of supported alarms (supported-alarm-list) and the configuration class the severity associated to each (problem-kind-severity-list).

The next tree shows an example of the relevant objects, focused on an air interface:

```
module: core-model-1-4
  +--rw control-construct!
     +--...
     +--rw logical-termination-point* [uuid]
     |  +--rw uuid
     |  +--rw name* [value-name]
     |  +--rw server-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--rw client-ltp*                -> /control-
construct/logical-termination-point/uuid
     |  +--…
     |  +--rw layer-protocol* [local-id]
     |  |  +--rw local-id
     |  |  +--rw layer-protocol-name?
     |  |  +--...
     |  |  +--rw air-interface:air-interface-pac
     |  |     +--ro air-interface:air-interface-capability
     |  |     +--rw air-interface:air-interface-configuration
     |  |     +--ro air-interface:air-interface-current-problems
     |  |     |  +--ro air-interface:number-of-current-problems?
     |  |     |  +--ro air-interface:time-of-latest-change?
     |  |     |  +--ro air-interface:current-problem-list*
[sequence-number]
     |  |     |  |  +--ro air-interface:problem-name?
     |  |     |  |  +--ro air-interface:sequence-number
     |  |     |  |  +--ro air-interface:timestamp?
     |  |     |  |  +--ro air-interface:problem-severity?
```

To retrieve the active alarms in any or each available interface of the NE, the key ID fields are:

USE CASE INFORMATION BLOCK	*MODEL CLASS/PARAMETER*	*PATH*	*comments*
ID FIELDS	**control-construct**	**/core-model:control-construct**	
	uuid	/core-model:control-construct/uuid	*unique identifier for the NE*
	name [value-name]	/core-model:control-construct/name	*additional field for labeling the NE according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Logical-termination-point[uuid]	***/core-model:control-construct/logical-termination-point***	
	uuid	**PATH_logical-termination-point**/uuid	*unique identifier for termination points. The LTP layerprotocol (for air interface layers) is*

			augmented with the TR-532 air-interface PAC
	name [value-name]	**PATH_logical-termination-point**/name	*additional field for labeling the LTP according to operator rules. Each item of the list includes "value" and "value-name" so more than one label is possible*
	Layer-protocol[local-id]	***/core-model:control-construct/logical-termination-point/layer-protocol***	
	local-id	**PATH_layer-protocol/**local-id	*local identifier for layer protocol*
	layer-protocol-name	**PATH_layer-protocol**/layer-protocol-name-type	*type of protocol*

Within each TR-532 interface PAC, the relevant parameters related to the alarms are:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
AVAILABLE ALARMS	**{PAC_capability_class}**	***/core-model:control-construct/logical-termination-point/layer-protocol/ {PAC_prefix}:{PAC_grouping}/{PAC_capability_class}***	
	supported-alarm-list	**PATH_{PAC_capability_class}**/supported-alarm-list	*must include the complete list of supported alarms for a given interface*
CONFIGURED SEVERITIES	**{PAC_configuration_class}**	***/core-model:control-construct/logical-termination-point/layer-protocol/ {PAC_prefix}:{PAC_grouping}/{PAC_configuration_class}***	
	problem-kind-severity-list[problem-kind-name]	**PATH_{PAC_configuration_class}**/problem-kind-severity-list	*Configured severity for supported alarms (list in the capabilities)*
	problem-kind-name	**PATH_problem-kind-severity-list**/problem-kind-name	*Alarm name, matching the name in the capabilities list*
	problem-kind-severity	**PATH_problem-kind-severity-list**/problem-kind-severity	*severity type covers WARNING, MINOR, MAJOR, CRITICAL, NON_ALARMED (-clear-), and NOT_YET_DEFINED (-indeterminate-)*
INTERFACE ALARMS	**[PAC_current-problems_class}**	***/core-model:control-construct/logical-termination-point/layer-protocol/ {PAC_prefix}:{PAC_grouping}/{PAC_current-problems_class}***	*These are the parameters that characterize each active alarm*
	number-of-current-problems	**PATH_{PAC_current_problems_class}**/number-of-current-problems	Total number of currently active alarms
	time-of-latest-change	**PATH_{PAC_current_problems_class}**/time-of-latest-change	
ALARM PARAMETERS	**current-problem-list[sequence-number]**	**PATH_{PAC_current-problems_class}**/current-problem-list	
	problem-name	**PATH_current-problem-list**/problem-name	*Alarm name, matching the one included in the capability list*

	sequence-number	**PATH_current-problem-list**/sequence-number	*Unique sequence number*
	timestamp	**PATH_current-problem-list**/current-problem-list/timestamp	*Timestamp corresponding to each alarm*
	problem-severity	**PATH_current-problem-list**/current-problem-list/problem-severity	*severity type covers WARNING, MINOR, MAJOR, CRITICAL, NON_ALARMED (-clear-), and NOT_YET_DEFINED (-indeterminate-)*

PATHs in the previous table have been defined openly to reflex that parameters and objects are common to the different available interfaces. The next table summarizes the corresponding {PAC_prefix} and {PAC_grouping} for the relevant TR-532 interface related PACS, which are the ones relevant for the use case:

{PAC_prefix}	*{PAC_grouping}*	*{PAC_classes}*
air-interface	air-interface-pac	air-interface-capability air-interface configuration air-interface-current-problems
wire-interface	wire-interface-pac	wire-interface-capability wire-interface configuration wire-interface-current-problems
pure-ethernet-structure	pure-ethernet-structure-pac	pure-ethernet-structure-capability pure-ethernet-structure-configuration pure-ethernet-structure-current-problems
hybrid-mw-structure	hybrid-mw-structure-pac	hybrid-mw-structure-capability hybrid-mw-structure-configuration hybrid-mw-structure-current-problems
ethernet-container	ethernet-container-pac	ethernet-container-capability ethernet-container configuration ethernet-container-current-problems
tdm-container	tdm-container-pac	tdm-container-capability tdm-container configuration tdm-container-current-problems
mac-interface	mac-interface-pac	mac-interface-capability mac-interface configuration mac-interface-current-problems
vlan-interface	vlan-interface-pac	vlan-interface-capability vlan-interface configuration vlan-interface-current-problems
ip-interface	ip-interface-pac	ip-interface-capability ip-interface configuration ip-interface-current-problems

In the specific case of the air interfaces PAC, ONF TR_532 model also includes some additional parameters that may be relevant for alarm use cases, depending on the specific application. These would be TCAs (Threshold Crossing Alarms), which are available for G.826 error performance events, transmission power level events and ACM events. Three specific lists are **available at the configuration class** of the air interfaces to configure the type of event, which

need to be supported for those applications requiring air interface TCA:

- G.826 (ES/SES/CSES/UAS) events: **g-826-threshold-cross-alarm-list**
- Transmit/received power events: **xlts-threshold-cross-alarm-list**
- ACM events: **acm-threshold-cross-alarm-list**

Asynchronous notification of the alarms requires implementation of NETCONF notifications, which need to be supported both at the controller and network elements, as per RFC5277 allowing the subscription from the crontroller to receive the desired events. Within TR-532 interface PACs, a similar problem-notification object is used.

```
  NETCONF notifications:
    +---n problem-notification
       +--ro counter?
       +--ro timestamp?
       +--ro object-id-ref?   -> /core-model:control-
construct/logical-termination-point/uuid
       +--ro problem?
       +--ro severity?
```

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
ALARM NOTIFICATION	**problem-notification**	*/{PAC_prefix}:problem-notification*	
	counter	**PATH_problem-notification**/counter	*counts problem notifications*
	timestamp	**PATH_problem-notification**/timestamp	
	object-id-ref	**PATH_problem-notification** /object-id-ref	*pointer to LTP with the alarm /core-model:control-construct/logical-termination-point/uuid*
	problem	**PATH_problem-notification**/problem	*according to capability, supported-alarm-list*
	severity	**PATH_problem-notification**/severity	*according to configuration, problem-severity-list*

Other X.733 parameters like alarm-type or probable-cause are not yet available in the current-problem class, but are already considered for addition as part of future development of the model. In the specific case of these two parameters, consideration is to expand available parameters in the current-problem-class and the problem-notification object.

Parameters can be accessed in a single step, so there is no need for an specific workflow. When data for all the interfaces available at the network element are needed, they can be extracted in N subsequent requests, each individual to the specific interface or in a single request from the controller

Proposed implementation IETF/IEEE - Data models

The YANG module "ietf-alarms", defined in the IETF RFC 8632 “A YANG Data Model for Alarm Management”, has the following structure:

```
     +--rw control
     |  +--rw max-alarm-status-changes?   union
     |  +--rw notify-status-changes?      enumeration
     |  +--rw notify-severity-level?      severity
     |  +--rw alarm-shelving {alarm-shelving}?
     |        ...
     +--ro alarm-inventory
```

```
|      |  +--ro alarm-type* [alarm-type-id alarm-type-qualifier]
|      |        ...
|      +--ro summary {alarm-summary}?
|      |  +--ro alarm-summary* [severity]
|      |  |     ...
|      |  +--ro shelves-active?   empty {alarm-shelving}?
|      +--ro alarm-list
|      |  +--ro number-of-alarms?   yang:gauge32
|      |  +--ro last-changed?       yang:date-and-time
|      |  +--ro alarm* [resource alarm-type-id alarm-type-
qualifier]
|      |  |     ...
|      |  +---x purge-alarms
|      |  |     ...
|      |  +---x compress-alarms {alarm-history}?
|      |        ...
|      +--ro shelved-alarms {alarm-shelving}?
|      |  +--ro number-of-shelved-alarms?      yang:gauge32
|      |  +--ro shelved-alarms-last-changed?   yang:date-and-
time
|      |  +--ro shelved-alarm*
|      |  |       [resource alarm-type-id alarm-type-qualifier]
|      |  |     ...
|      |  +---x purge-shelved-alarms
|      |  |     ...
|      |  +---x compress-shelved-alarms {alarm-history}?
|      |        ...
|      +--rw alarm-profile*
|               [alarm-type-id alarm-type-qualifier-match
resource]
|               {alarm-profile}?
|         +--rw alarm-type-id                          alarm-type-
id
|         +--rw alarm-type-qualifier-match             string
|         +--rw resource                               resource-
match
|         +--rw description                            string
|         +--rw alarm-severity-assignment-profile
|                 {severity-assignment}?
|                 ...
```

Moreover, the YANG module "ietf-alarms-x733" augments the alarm inventory, the alarm lists and the alarm notification with X.733 and X.736 parameters.

The Data Model version is provided by the *Revision Statement, ietf-alarms (IETF RFC 8632)*.

Parameter	Type / Sub-parameter	Description / Application (indicative)
resource alarm-type-id alarm-type-qualifier	union identityref string	Current Alarm Id
/alarm-notification/resource /alarm-notification/alarm-type-id /alarm-notification/alarm-type-qualifier /alarm-notification/time		Current Alarm Notification Id
last-raised	yang:date-and-time	Current Alarm Creation Time Stamp
event-type	enumeration *ietf-alarms-x733 (RFC8362)*	Current Alarm Event Type
probable-cause probable-cause-string	uint32 string *ietf-alarms-x733 (RFC8362)*	Current Alarm Probable Cause
alarm-type-qualifier (alarm-type-id)	string (identityref)	Current Alarm Specific Problem
/alarms/alarm-list/alarm /alarms/shelved-alarms/shelved-alarm		Current Alarm Reporting Mechanism
perceived-severity	enumeration	Current Alarm Severity

2.5 Configuration and Activation

2.5.1 Use Case 5.1 – Firmware download and activation

Use Case	Firmware download and activation
Id	Use Case 5.1
Summary	This use case complements the firmware inventory, targeting the management of the firware available on the network elements, with a key focus on the download of new firmware to the network element, monitoring of the downloading process and the activation of the firmware within the network element.
Benefits and Motivation	An standardized way to download and activate new firmware in a multi-vendor network provides significant benefits in terms of operational simplicity, with a high potential to reduce significantly the time and effort to upgrade the network. This also will also enhance network quality and performance, as time to upgrade to new FW releases that correct bugs or bring newer features are shortened. Cluster and network-wide upgrades across all installed vendors and products may be automated and optimized taking advantage of all the information visible for a SDN MW domain agnostic controller, that includes topology, configuration, radio model types and hardware, etc., bringing a highly positive operational impact.
Detailed description	A basic need of the use case will be to identify clearly FW available for activation. This will be one of the relevant operations considered within scope. Details around FW inventory have already been covered in use case 2.1.2. • **Firwmare inventory check:** with this operation, FW available and suitable for activation (typically packages) can be checked, confirmed and selected for activation. On top of this, the FW management use case considers three additional operations, which are required to enable the main management identified needs: • **Firmware download:** new firmware needs to be downloaded to the NEs. With this operation a FW bundle will be downloaded to the appropriate benches or modules within the network element to be hosted as standby firmware, suitable to be activated afterwards. • **Download status check:** it shall be possible to check the status of an active download in order to take decisions around next steps (aborting / activation). • **Download abort:** A method is required to cancel an active download (e.g just in case of a problem or a download time exceeding a maximum timer, etc.). With this operation, an active download can be cancelled with no changes in the running FW of the network element. To carry out the decision of aborting, specific parameters within the model that can be accessed by an agnostic controller will serve to detail the download status. • **Firmware activation:** Firwmare available in the NE needs to be activated. With this operation, once in the network element, the FW package or FW module(s) that are intended for activation can be activated to become running FW in the network element. This operation may require a reboot depending on the FW and vendor implementation so must be planned accordingly. Typically, activating firmware in an standby bench or module means the automatic de-activation of the previously running FW, which will become the standby.

Proposed implementation ONF -Data models	Implementation of this use case relies on ONF CIM classes and the support of a specific TR-532 MW technology specific augment to the CIM, *firmware_1.0.0,* with resources (papyrus UML, overview and documentation) available here and latest yang module *firmware-1-0* here . General aspects and requirements applying to the ONF CIM support and details around its classes are given in section 3. The firmware PAC attaches to the control-construct, inventory relies on the parameters available at the *firmware-component-list* within this object. Augmenting it with a firmware collection that includes both a firmware component list (already covered in detail in the FW inventory case, 2.1.2) and a download container. Within the firmware component list, apart from the ID and status fields, the firmware component class and the individual activation capability will be relevant to identify the FW that is available and suitable for activation, in the inventory check operation. The download container includes specific parameters to retrieve the status of an existing download. `module: firmware-1-0` `  augment /core-model:control-construct:` `    +--rw firmware-collection` `       +--ro firmware-component-list* [local-id]` `	+--ro firmware-component-pac` `		+--ro firmware-component-capability` `			+--...` `			+--ro firmware-component-class?` `firmware-component-class-type` `			+--ro individual-activation-is-avail?` `boolean` `		+--ro firmware-component-status` `		+--...` `	+--ro subordinate-firmware-component-list* ->` `/core-model:control-construct/firmware:firmware-` `collection/firmware-component-list/local-id` `	+--ro local-id string` `	+--...` `	+--ro operational-state?` `operational-state` `	+--ro lifecycle-state?` `lifecycle-state` ` +--ro download` ` +--ro filename? string` ` +--ro download-status? download-` `status-type` ` +--ro download-status-description? String` The complete use case spans several differentiated operations or actions, with some of the operations requiring input parameters in order to be fulfilled. • **Firmware Download**: download path, file name, authentication (user / pass), SSH key to access the server, additional options for the download • **Download status check:** identification of the current status of the download • **Abort download:** no inputs required • **Firmware inventory check:** identification of the node to retrieve the information • **Firmware activation**: ID of the FW component within the NE to be activated, options for the activation. Apart from the presented model structure, to support all the aforementioned actions, the firmware PAC includes three specific RPCs to handle directly some of them. For all the operations described in the use case description, an implementation using ONF

modelling follows:

First operation would be the **Firmware download,** which has a dedicated RPC in the model:

RPC:	Type	Description
+---x download-firmware-component	rpc	Operation for downloading some firmware component (e.g. package) from a server onto the device.
| +---w input		
| +---w source-uri	string	URI specifying the remote-file-path for downloading the firmware component. Format:<protocol>://<host>[:port]/path'
| +---w filename	string	Name of the file to be downloaded
| +---w username-at-file-server?	string	Username to access the file server
| +---w password-at-file-server?	string	Password to access the file server
| +---w ssh-key?	string	Client key for accessing the file server
| +---w force-download	boolean	Parameter to force download, even if file is already present on the device

During the download, the **download status check** may be performed. Status can be retrieved and checked by an agnostic MW domain controller with the next parameters:

USE CASE INFO BLOCK	*MODEL CLASS / PARAMETER*	*PATH*	*comments*
FW DOWNLOAD INFO	**download**	*/core-model:control-construct/firmware:firmware-collection/download*	
	filename	**PATH_download**/filename	*name of the file being downloaded*
	download-status	**PATH_download**/download-status	*enum that includes possible status (successful, unable to connect, file not found, aborted, failed, idle,…)*
	download-status-description	**PATH_download**/download-status-description	*additional details about the download status (free text)*

If needed, the **download can be aborted** with an specific RPC call (no input parameters):

RPC:	Type	Description
+---x abort-firmware-download	rpc	Currently on-going download of a firmware component shall immediately be terminated

If the FW is already in the NE (new download or already present in an standby status), an **inventory check** may be done to check available files and class, status of activation, type and double check individual activation capability, prior to activation. Relevant fields are detailed in the FW inventory use case 2.1.2:

Finally, **FW activation** of the specific firmware components may be completed with a dedicated RPC

RPC:	Type	Description
+---x activate-firmware-component	rpc	Operation for activating some firmware component on the device
+---w input		
+---w firmware-component -> /core-model:control-construct/firmware:firmware-collection/firmware-component-list/local-id	leafref	Reference of the firmware component to be activated

	+---w activation-delay-period uint64 uint64 Number of seconds the device shall delay the execution after receiving the invokation of the activation operation In terms of use case workflow, it depends mainly on two typical situations: • New FW that is downloaded to the NE to be activated: 1. **Firmware download** (RPC) 2. **Downoload check** 3. **(optional) FW download abort (RPC)** **4. Inventory check** 5. **Firmware activation** (RPC) • Activation of FW which is already present in the NE and does not require a new download: **1. Inventory check** 2. **Firmware activation** (RPC)
Proposed implementation IETF/IEEE -Data models	

3 NBI Use Cases

3.1 Direct device management interface

This section includes a first set of relevant use cases defined by MUST subgroup related to the NBI of wireless backhaul SDN domain controllers. Each use case includes a common ID, Summary, Benefits and motivation and Detailed description sections, which describe the goals and common technical guidelines of the target use case.

- Taking as a reference the general architecture described in section 1.2, this chapter considers two types of use case categories:
    - First, a base generic use case will be introduced, which targets providing architectural flexibility to enable external (e.g outside the controller itself) implementation of multiple use cases consuming a single non-abstract harmonized RESTONF/YANG interface. This case applies mainly to agnostic controllers, where network elements corresponding to a similar domain (e.g microwave/mmWave devices) are required to support (natively or through SW mediators) a common NETCONF/YANG SBI.
    - Second, a larger set of specific use cases linked to particular functions, especially those of the highest relevance at the hierarchical control level (multi-domain). For these, implementation has been proposed based on IETF models as specified in relevant drafts and RFCs. This does not limit applicability of the specified NBI interfaces with different underlying SBI models, and both IETF or ONF models as introduced in section 2 can be supported as long as there is a mapping between required parameters (similar cases occur in other domains as IP with OpenConfig SBI and IETF NBIs). In addition to implementation into agnostic controllers, adoption of these NBI use cases might be also relevant in vendor-specific NMS to enable common management of specific use cases over legacy equipment.

Both categories of use cases are non-exclusive, and can be implemented in the same domain controller to achieve a larger flexibility in implementation or to adapt to the specific MNO strategy and network constraints and system map particularities.

Each use case is self-contained, including references to the target data models and standard documents, and also a set of minimum requirements agreed for each proposed implementation.

For the set of specific use cases, this document takes as an additional reference the ETSI GS mWT 024 V1.1.1 (2022-03).

3.1.1 Use case 6.1 – Non-abstract RESTCONF/YANG interface exposing SBI model

Use Case	non-abstract RESTCONF/YANG interface exposing SBI model
Id	Use Case 6.1

Summary	Use case considers the domain controller implementing a RESTCONF NBI where the harmonized data models of the SBI become available (passed through) at the northbound interface.
Benefits and Motivation	Exposing at a common RESTCONF NBI the SBI models selected for the domain allows systems and applications on top of the domain controller to access and manage the complete parameter set of any network element being managed by it. This provides flexibility to implement domain specific applications and use cases reducing complexity of the controller if desired. A micro-service layer on top of the controller might then be deployed to implement flexibly multiple domain specific MW applications or even intermediate functionality to implement abstracted interfaces to higher layer systems having access to the full granularity of available parameters at the SBI model. This way, the option to implement use cases externally via dedicated applications without adding extra complexity to the controller is enabled in parallel to implement functionality on the controller itself, using dedicated standard-based NBIs as those also included in this document. Decisions on how to manage each use case can be then taken depending on use case specifics or availability of relevant standard models applicable to the NBI (e.g common models already adopted as NBI in other network domains). Definition and specification of common application patterns also facilitates application development by multiple parties (SW vendors, MW vendors, in-house, etc.) and the portability of applications between different regional SDN controlled networks.
Detailed description / Proposed implementation	To implement the use case, as with the rest of the use cases presented in this document, the controller needs to provide a RESTCONF interface according to IETF RFC8040, IETF RFC8527 and further relevant extensions at its northbound, and support data modeling according to YANG 1.1 (IETF RFC7950 and relevant extensions). The controller will also, according to the general SDN MW architecture, provide a NETCONF over SSH interface at is SBI according to IETF RFC6241, IETF RFC8526 (and further relevant extensions). The use case is not limited to any specific YANG model at the controller SBI, although in many cases typically the target domain architecture (fully agnostic multi-vendor SDN support) considers that domain devices implement common YANG models at its harmonized management interface (controller SBI). Mechanism and data structure for mounting and representing mounted YANG schema according to IETF RFC8528 and representing the mapping of the mounted YANG schema on the supported data stores according to IETF RFC8525 also needs to be considered. NETCONF event notification handling (IETF RFC5277) is required also support, to enable interface related information to be updated in the network representation based on notifications of the devices, and the controller also is required to enable mechanisms to facilitate applications to register subscribe for receiving notifications (e.g according to IETF RFC8639 and IETF RFC8641). Aside these general aspects, the key element for the implementation of this use case is that the controller shall support an agnostic management protocol translation between its southbound and northbound interfaces that passes through any valid YANG model (MD-SAL2). The controller will comprise a data base that represents the underlying network. NETCONF devices will then be seen as a mount point on it based on this service model-agnostic abstracted architecture, exposing the full device data (configuration and operational data stores) and its capabilities at the controller RESTCONF NBI. Data will be modeled and represented as data tree according to the YANG model defined for the domain devices, with

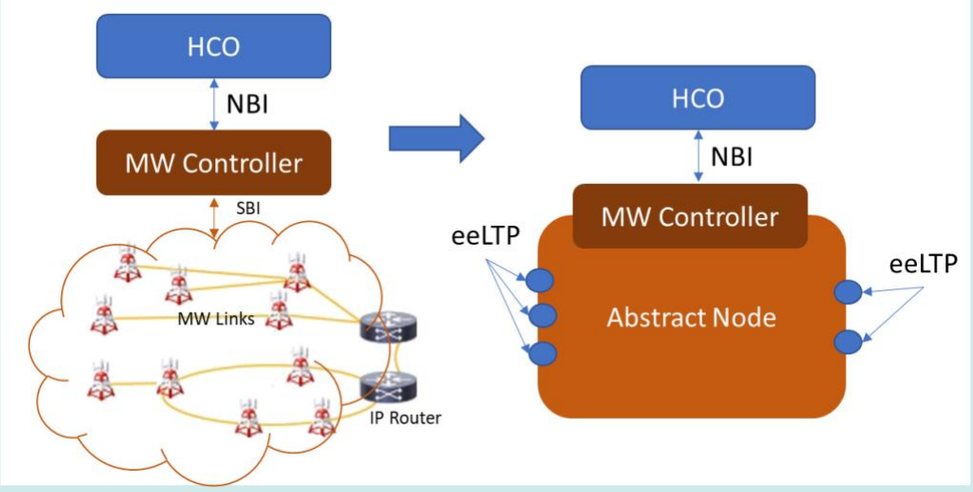
2 MD-SAL (Model Driven Service Abstraction Layer) is OpenDaylight terminology. Key elements in this srchitecture being the independence of device/service models (model agnostic) and a common REST API (in ODL, **RESTCONF connector** built on top of MD-SAL exposes YANG-modeled APIs transparently via HTTP with support for XML/JSON payload).

	possibility to address any element / subtree. And each node will be able to be addressed using its unique identifier to provide unambiguous information.
	These mount points allow applications and users (over RESTCONF) to interact with the mounted devices through operations (GET, PUT, etc..) to learn about device capabilities, configuration, status or carry out configuration actions on the devices.
	As an example (use case is not restricted to any specific model), considering devices at the SBI of the controller are implementing ONF TR512/532 microwave model, with the next high-level tree:
	`module: core-model-1-4`
	`  +--rw control-construct!`
	`     +--rw top-level-equipment*          -> /control-construct/equipment/uuid`
	`     +--rw equipment* [uuid]`
	`          +...`
	`     +--rw logical-termination-point* [uuid]`
	`          +...`
	`     +--rw forwarding-domain* [uuid]`
	`          +...`
	`     +--rw profile-collection`
	`          +...`
	`     +--rw external-managed-id`
	`     +--rw local-id?                     string`
	`     +--rw uuid?                         universal-id`
	`     +--rw name* [value-name]`
	`     +--rw label* [value-name]`
	`     +--rw extension* [value-name]`
	`     +--ro operational-state?            operational-state`
	`     +--rw administrative-control?       administrative-control`
	`     +--ro administrative-state?         administrative-state`
	`     +--rw lifecycle-state?              lifecycle-state`
	`     +--rw address*                      dt-address`
	and an ODL controller (not mandatory, as long as the controller enables functionality as requested, similar to the MD-SAL implemented in ODL architecture) managing these devices, a RESTCONF URL to get the full list of LTPs in a node identified with its node uuid would be:
	https://<controller_IP:port>/rests/data/network-topology:network-topology/topology=topology-netconf/node=<node_uuid>/yang-ext:mount/core-model-1-4:control-construct?fields=logical-termination-point
	First section of the URL identifies the node within the controller topology, with the device data, corresponding to the implemented YANG model under consideration in the device being accessible following RESTCONF syntax, according to the YANG model tree.

3.2 Network & Services Auto-Discovery

3.2.1 Use case 7.1 – Network topology Auto-Discovery Black Box Approach

Use Case	Network topology Auto-Discovery Black Box Approach
Id	Use Case 7.1
Summary	Use case focuses on the retrieval of Ethernet topology information of a MW network by a HCO from MW CO. In black Box approach the HCO sees the MW network as a generic abstract node showing only the external ethernet links termination points.
Benefits and	The HCO is the system responsible for end-to-end service instantiation among different domains, MW-MW and MW-IP, while each domain controller within the hierarchical architecture is in charge to

Motivation	manage connection inside its domain. The topology information inside a domain is essential to enable multiple domain-specific and end to end applications, being one example the multi-domain service provisioning, typically managed by the hierarchical controller.. To simplify overall architecture and limit the volume of information to be shared, the “black box” topology use case does not consider exposing the full detailed topology to the HCO, but a reduced version, being the main focus identification of the border termination points of each domain.
Detailed description / Proposed implementation	The network discovery Use Case is the basic functionality needed for network and service management automation, in that it shall provide the inventory of the Ethernet topology and the Ethernet service information The topology Auto-Discovery for Black Box Approach means: 1. The detailed Ethernet topology information shall not be available from the MW CO nor requested over the NBI by the HCO. 2. The whole domain shall be modelled as a single Ethernet abstract node, showing only the external ethernet links termination points (eeLTPs).  Figure 3: Use case 7.1 – Network topology Auto-Discovery Black Box Approach This UC is applying to a “island” of MW links where all the NEs are fully managed from a single Domain controller. Ideally, having all MW devices supporting a common SBI SDN interface with no elements from other transport domains (as IP devices), a fully complete single "black box” can be retrieved. Considering the case where MW might be from different Vendors and a single agnostic controller cannot manage all NEs (so, in cases where the SBI CO-NE is not fully supported by all devices), this UC can be seen as an aggregation of “black boxes”, as long as vendor-specific controller supports via a common NBI for this use case. The HCO will then see more abstract nodes, each managed by different vendor-specific controller and will be able to compose the topology. For simplicity the picture below shows this case simplified with a chain of 2 links of different Vendors

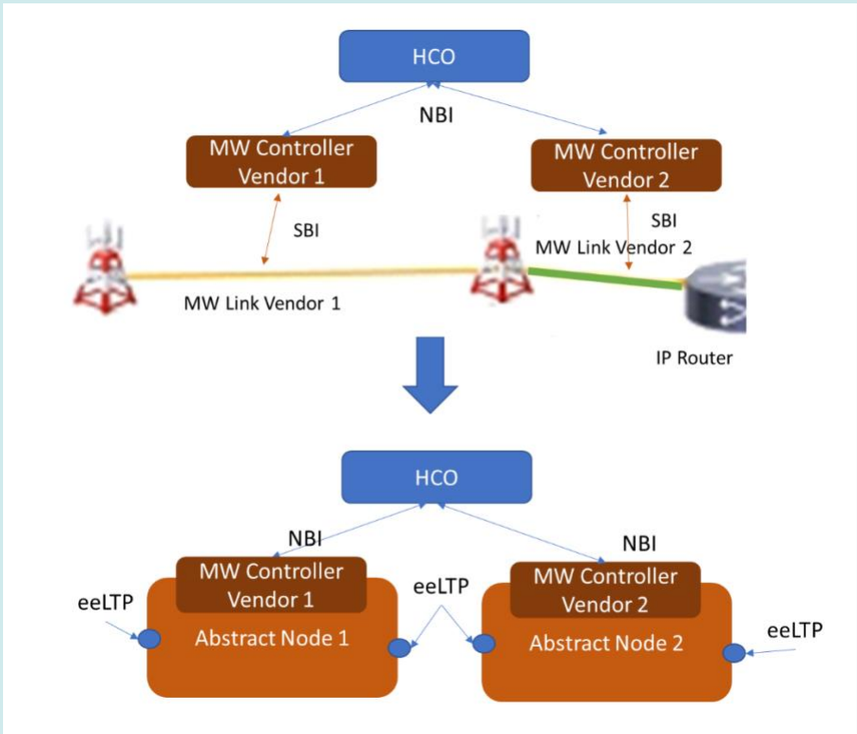


Figure 4: Multi-domain network abstraction black box approach.

The same happens when an IP router is present in a chain of MW links: the router is seen by HCO as a abstract node managed by IP Controller in the middle of 2 abstract node managed by MW Controller.

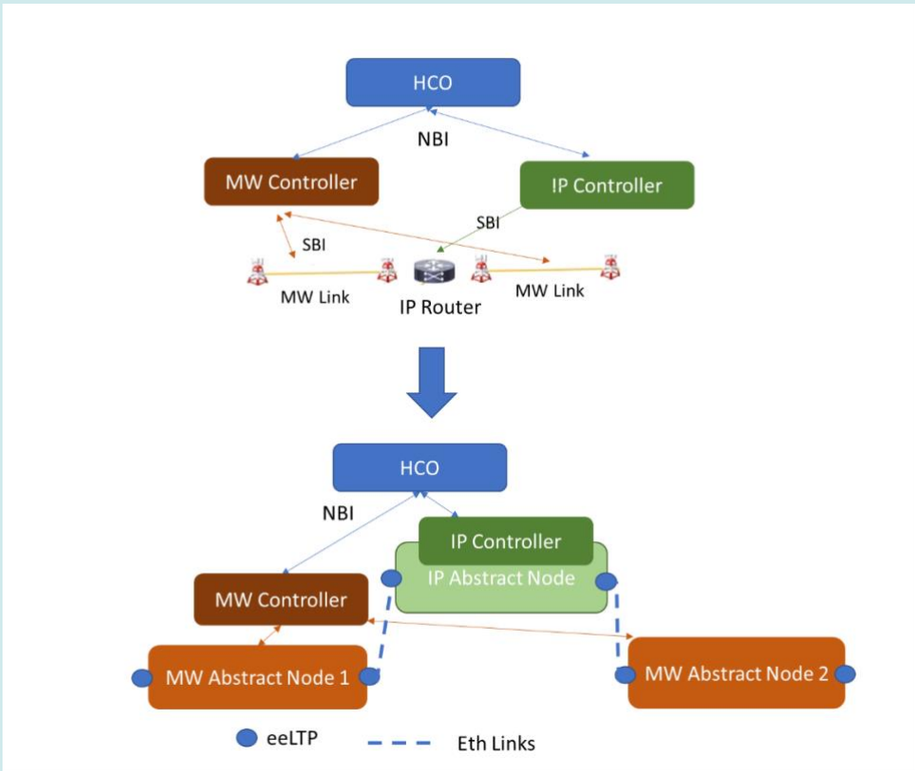


Figure 5: Multi-layer network abstraction black box approach.

One of the main purposes of the HCO is to coordinate and provision services across multiple domains. The HCO to complete the network discovery needs to connect eeLTP of different domain (abstract

node) creating Inter domain link. This could be done by HCO using LLDP as explained in in UC 3.2.3.

Proposed implementation (IETF Data models)

The HCO/CO at NBI shall use YANG models RFC 8345 and RFC 8795 and the Data Model for Ethernet TE Topology, defined in the "ietf-eth-te-topology" YANG module of “draft-ietf-ccamp-eth-client-te-topo-yang to implement the UC and shall use the following mandatory attributes

Ethernet TE topology	
Model	Attributes explanation
module: ietf-network	
+--rw networks	Serves as a top-level container for a list of networks.
+--rw network* [network-id]	Describes a network. A network typically contains an inventory of nodes, topological information (augmented through the network-topology data model), and layering information
| +--rw network-id network-id	Identifies a network
| +--rw network-types	Serves as an augmentation target. The network type is indicated through corresponding presence containers augmented into this container
| | +--rw tet:te-topology!	Its presence identifies the TE topology type
| | +--rw ethtetopo:eth-tran-topology!	Eth transport topology type
| +--rw node* [node-id]	The inventory of nodes of this network
| | +--rw node-id node-id	Uniquely identifies a node within the containing network.
| | +--rw nt:termination-point* [tp-id]	A termination point can terminate a link. Depending on the type of topology, a termination point could, for example, refer to a port or an interface
| | | +--rw nt:tp-id tp-id	Termination point identifier
| | | +--rw tet:te-tp-id? te-types:te-tp-id	An identifier that uniquely identifies a TE termination point
| | | +--rw tet:te!	Indicates TE support.
...	
| | | | +--rw tet:name? string	A descriptive name for the LTP.
...	
| | +--rw tet:te-node-id? te-types:te-node-id	The identifier of a node in the TE topology. A node is specific to a topology to which it belongs
...	
| +--rw tet:te!	Indicates TE support
| +--rw tet:name? string	Name of the TE topology. This attribute is optional and can be specified by the operator to describe the TE topology, which can be useful when 'network-id' (RFC 8345) is not descriptive and not modifiable because of being generated by the system

The “client-facing” attribute of an eLTP define if it is an eeLTP. It is then a mandatory parameter that

shall be present, and shall have the Boolean value “true” for an eeLPT.
This attribute may be configured either manually or automatically.

Ethernet TE topology	
Model	**Attributes explanation**
| | | +--rw ethtetopo:eth-svc!	
| | | +--rw ethtetopo:client-facing? boolean	If the eLTP is an eeLTP the attribute shall be present, and shall have the Boolean value “true”. If the eLTP is not an eeLTP, the attribute shall be missing or, if present, it shall assume the Boolean value “false"

3.2.2 Use case 7.2 - Network topology Auto-Discovery Grey Box Approach

Use Case	Network topology Auto-Discovery Grey Box Approach
Id	Use Case 7.2
Summary	Use case focuses on the retrieval of Ethernet topology information by HCO from MW CO as for black Box approach but including two levels of further information: • NEs and detailed ethernet links topology inside the MW domain. • Geo-localization of NEs
Benefits and Motivation	The added detailed Ethernet connectivity inside MW domain is mainly for troubleshooting activities (e.g display and correlation of alarm or performance details in a multi-domain E2E transport topology), limiting the volume of information shared across NBI, but giving the possibility to identify portions or specific links of the network where a problem occurs. The added detailed of geo-localization information of NEs inside MW domain serves to identify geographically portion(s) of the network that needs increase of services for specific reasons (e.g. touristic area for limited period). This UC is applying to the “islands” of MW links where all the NEs are fully managed from a single Domain controller as already explained in UC 3.2.1
Detailed description	In the Network topology Auto-Discovery Grey Box use case,not only external ethernet links termination points (eeLTPs) are requested by HCO, but also: 1. NEs identifier 2. Internal End Points (ieLTPs) of Ethernet links connecting NEs 3. Ethernet links among eeLTPs and ieLTPs

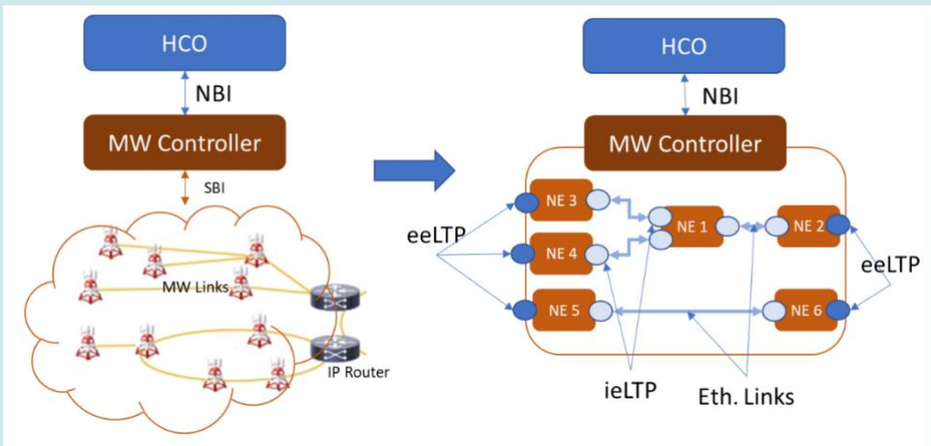


Figure 6: Use case 7.2 – Network topology Auto-Discovery Grey Box Approach.

The second level of information are the geographical coordinates of each NE inside MW domain, so the information shared through NBI are latitude and longitude of NEs.

Proposed implementation (IETF - Data models)

The HCO/CO at NBI shall use YANG models RFC 8345 and RFC 8795 and the Data Model for Ethernet TE Topology, defined in the "ietf-eth-te-topology" YANG module of “draft-ietf-ccamp-eth-client-te-topo-yang to implement the UC and shall use the following mandatory attributes:

Ethernet TE topology	
Model	**Attributes explanation**
module: ietf-network	
+--rw networks	Serves as a top-level container for a list of networks.
+--rw network* [network-id]	Describes a network. A network typically contains an inventory of nodes, topological information (augmented through the network-topology data model), and layering information
| +--rw network-id network-id	Identifies a network
| +--rw network-types	Serves as an augmentation target. The network type is indicated through corresponding presence containers augmented into this container
| | +--rw tet:te-topology!	Its presence identifies the TE topology type
| | +--rw ethtetopo:eth-tran-topology!	Eth transport topology type
| +--rw node* [node-id]	The inventory of nodes of this network
| | +--rw node-id node-id	Uniquely identifies a node within the containing network.
| | +--rw nt:termination-point* [tp-id]	A termination point can terminate a link. Depending on the type of topology, a termination point could, for example, refer to a port or an interface
| | | +--rw nt:tp-id tp-id	Termination point identifier
| | | +--rw tet:te-tp-id? te-types:te-tp-id	An identifier that uniquely identifies a TE termination point
| | | +--rw tet:te!	Indicates TE support.

...	
| | | +--rw tet:name? string	A descriptive name for the LTP.
...	
| | +--rw ethtetopo:eth-svc!	
| | +--rw ethtetopo:client-facing? boolean	If the eLTP is an eeLTP the attribute shall be present, and shall have the Boolean value “true”. If the eLTP is not an eeLTP, the attribute shall be missing or, if present, it shall assume the Boolean value “false"
...	
| | +--rw tet:te-node-id? te-types:te-node-id	The identifier of a node in the TE topology. A node is specific to a topology to which it belongs
...	
| +--rw nt:link* [link-id]	A network link connects a local (source) node and a remote (destination) node via a set of the respective node's termination points. It is possible to have several links between the same source and destination nodes. Likewise, a link could potentially be re-homed between termination points. Therefore, in order to ensure that we would always know to distinguish between links, every link is identified by a dedicated link identifier. Note that a link models a point-to-point link, not a multipoint link
| | +--rw nt:link-id link-id	The identifier of a link in the topology. A link is specific to a topology to which it belongs
| | +--rw nt:source	This container holds the logical source of a particularlink
| | | +--rw nt:source-node? -> ../../../nw:node/node-id	Source node identifier. Must be in the same topology.
| | | +--rw nt:source-tp? leafref	This termination point is located within the source node and terminates the link
| | +--rw nt:destination	This container holds the logical destination of a particular link
| | | +--rw nt:dest-node? -> ../../../nw:node/node-id	Destination node identifier. Must be in the same network
| | | +--rw nt:dest-tp? leafref	This termination point is located within the destination node and terminates the link.
| | +--rw tet:te!	Indicates TE support
| | +--rw tet:te-link-attributes	Link attributes in a TE topology
| | | +--rw tet:name? string	Link name
...	
| | | +--rw tet:max-link-bandwidth	Maximum bandwidth that can be seen on this link in this direction. Units are in bytes per second

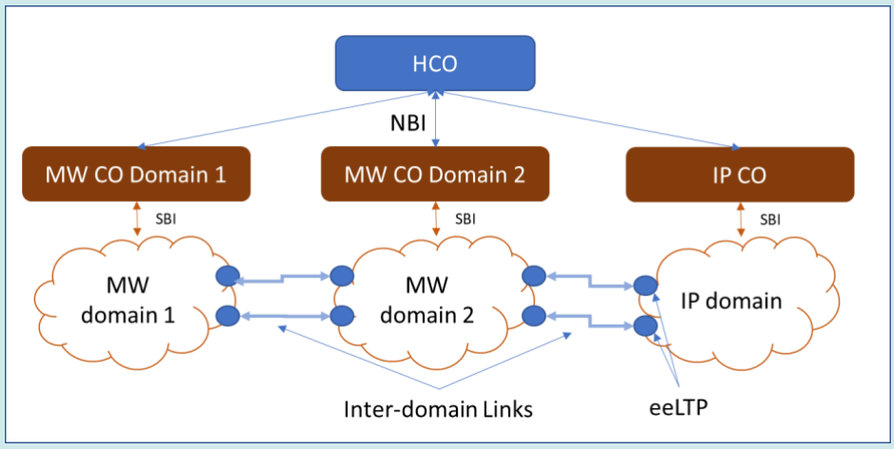
| | | +--rw tet:te-bandwidth	Container that specifies TE bandwidth. The choices can be augmented for specific dataplane technologies.
| | | +--rw (tet:technology)?	Data plane technology type
| | | +--:(ethtetopo:eth)	Eth
| | | +--rw ethtetopo:eth-bandwidth? uint64	Available bandwith value expressed in kilobits per second
...	
| +--rw tet:te!	Indicates TE support
| +--rw tet:name? string	Name of the TE topology. This attribute is optional and can be specified by the operator to describe the TE topology, which can be useful when 'network-id' (RFC 8345) is not descriptive and not modifiable because of being generated by the system

For secondo level of information of latitude and longitude of NEs the following mandatory attributes shall be used:

Ethernet TE topology	
Model	**Attributes explanation**
| | +--ro tet:geolocation	Contains a GPS location.
| | | +--ro tet:altitude? int64	Distance above sea level.
| | | +--ro tet:latitude? geographic-coordinate-degree	Relative position north or south on the Earth's surface.
| | | +--ro tet:longitude? geographic-coordinate-degree	Angular distance east or west on the Earth's surface

3.2.3 Use case 7.3 - Inter-Domain auto-discovery

Use Case	Inter-Domain auto-discovery
Id	Use Case 7.3
Summary	The HCO shall be able to identify eeLTP connectivity across different domains and establish Inter-domain links
Benefits and Motivation	The main scope of the HCO is to manage, coordinate and instantiate services across multiple domains. The CO cannot know or instantiate inter-domain links, so these have to be managed by HCO based on information about the eeLTPs from the corresponding domains.
Detailed description	The HCO shall discover inter domain MW switching points (between two different MW domains in case they are not supporting a common CO SBI) and inter domain MW-IP stitching points (between MW domain and IP domain). Inter-domain links must be modelled by the HCO based on information about the eeLTPs from the corresponding domains. This is performed by assigning a conventional network-wide-unique value to the attribute /nw:networks/nw:network/nw:node/nt:termination-point/tet:te/tet:inter-domain-plug-id (as defined in the "ietf-eth-te-topology" YANG module of “draft-ietf-ccamp-eth-client-te-topo-yang) for the external access links of the MW domain.



That attribute can be configured manually or automatically. Automatic way could use an algorithm based on the LLDP protocol as described in [Annex B] – Inter-domain Auto Discovery – LLDP algorithm.

.

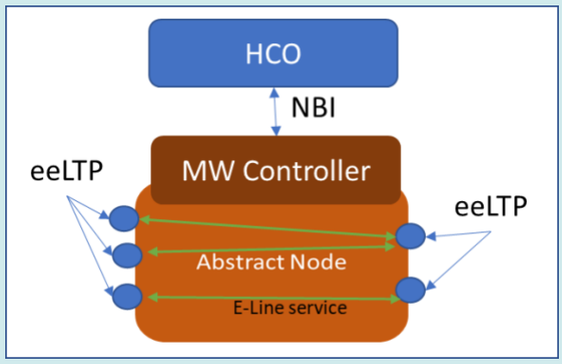
Proposed implementation (IETF Data models)

The HCO/CO at NBI shall use YANG models RFC 8345 and RFC 8795 and the Data Model for Ethernet TE Topology, defined in the "ietf-eth-te-topology" YANG module of "draft-ietf-ccamp-eth-client-te-topo-yang to implement the UC and shall use the following mandatory attributes:

Ethernet TE topology	
Model	**Attributes explanation**
module: ietf-network	
+--rw networks	Serves as a top-level container for a list of networks.
+--rw network* [network-id]	Describes a network. A network typically contains an inventory of nodes, topological information (augmented through the network-topology data model), and layering information
…	
| +--rw node* [node-id]	The inventory of nodes of this network
| | +--rw node-id node-id	Uniquely identifies a node within the containing network.
| | +--rw nt:termination-point* [tp-id]	A termination point can terminate a link. Depending on the type of topology, a termination point could, for example, refer to a port or an interface
| | | +--rw nt:tp-id tp-id	Termination point identifier
| | | +--rw tet:te-tp-id? Te-types:te-tp-id	An identifier that uniquely identifies a TE termination point
…	

	| | | | +--rw tet:inter-domain-plug-id?	Binary	A network-wide unique number that identifies on the network a connection that supports a given inter-domain TE link. This is a more flexible alternative to specifying 'remote-te-node-id' and 'remote-te-link-tp-id' on a TE link when the provider either does not know 'remote-te-node-id' and 'remote-te-link-tp-id' or needs to give the client the flexibility to mix and match multiple topologies.

3.2.4 Use case 7.4 - Service's Auto-discovery

Use Case	Service's Auto-discovery
Id	Use Case 7.4
Summary	The HCO shall receive information of E-Line services in a MW domain. The MW CO shall provide this information through the NBI interface. This UC is applying to the "islands" of MW links where all the NEs are fully managed from a single Domain controller as already explained in UC3.2.1.
Benefits and Motivation	The HCO is responsible for end-to-end service instantiation among different domains, MW-MW and MW-IP, so the collection of already configured services and their modification or deletion is fundamental for its role.
Detailed description	The MW CO shall share information of E-Line services present in its domain to the HCO, that are VLAN-based Ethernet point-to-point. Details about the domain-internal transport mechanism or where which VLAN operations are performed within the MW domain may not be exposed over the NBI to the HCO HCO is to coordinate E2E services across multiple domains, i.e. between eeLTPs The key Information send to HCO by MW CO is: 1. External End Points (eeLTPs) 2. VLAN ID: C-Vlan, S-Vlan, untagged 3. SLA: CIR, PIR etc… 4. Ingress, egress Bandwidth The HCO retrieves information on all e-line services for the entire MW domain. 
Proposed implementation (IETF Data models)	The HCO/CO at NBI shall use YANG models RFC 8345 and RFC 8795 and the Data Model for Ethernet TE Topology, defined in the "ietf-eth-te-topology" YANG module of "draft-ietf-ccamp-eth-client-te-topo-yang to implement the UC and shall use the following mandatory attributes: ***Ethernet TE topology***

Model	Attributes explanation
module: ietf-network	
+--rw networks	Serves as a top-level container for a list of networks.
+--rw network* [network-id]	Describes a network. A network typically contains an inventory of nodes, topological information (augmented through the network-topology data model), and layering information
| +--rw network-id network-id	Identifies a network
| +--rw node* [node-id]	The inventory of nodes of this network
| | +--rw node-id node-id	Uniquely identifies a node within the containing network.
| | +--rw nt:termination-point* [tp-id]	A termination point can terminate a link. Depending on the type of topology, a termination point could, for example, refer to a port or an interface
| | | +--rw nt:tp-id tp-id	Termination point identifier
| | | +--rw tet:te-tp-id? te-types:te-tp-id	An identifier that uniquely identifies a TE termination point
...	
| | | +--rw ethtetopo:port-vlan-id? etht-types:vlanid	Port VLAN ID of the LTP

The HCO/CO at NBI shall use the Transport Network Client Signals Model, defined in the "etht-svc" YANG module of “draft-ietf-ccamp-client-signal-yang” to implement the UC for E-Line information and shall use the following mandatory attributes:

Ethernet Service	
Model	**Attributes explanation**
module: ietf-eth-tran-service	
+--rw etht-svc	ETH services.
+--rw globals	Globals Ethernet configuration data container
| +--rw named-bandwidth-profiles* [bandwidth-profile-name]	List of named bandwidth profiles used by Ethernet services.
| +--rw bandwidth-profile-name string	Name of the bandwidth profile.
| +--rw bandwidth-profile-type? etht-types:bandwidth-profile-type	The type of bandwidth profile.
| +--rw CIR? uint64	Committed Information Rate in Kbps
| +--rw CBS? uint64	Committed Burst Size in in Kbytes
| +--rw EIR? uint64	Excess Information Rate in Kbps. In case of RFC 2698, PIR = CIR + EIR
| +--rw EBS? uint64	Excess Burst Size in KBytes. In case of RFC 2698, PBS = CBS + EBS
| +--rw color-aware? boolean	Indicates weather the color-mode is color-aware or color-blind.
| +--rw coupling-flag? boolean	Coupling Flag
+--rw etht-svc-instances* [etht-svc-name]	The list of p2p ETH service instances

+--rw etht-svc-name string	Name of the ETH service.
...	
+--rw etht-svc-type? etht-types:service-type	Type of ETH service (p2p, mp2mp or rmp).
...	
+--rw etht-svc-end-points* [etht-svc-end-point-name]	The logical end point for the ETH service
| +--rw etht-svc-end-point-name string	The name of the logical end point of ETH service.
...	
| +--rw etht-svc-access-points* [access-point-id]	List of the ETH trasport services access point instances.
| | +--rw access-point-id string	ID of the service access point instance
| | +--rw access-node-id? te-types:te-node-id	The identifier of the access node in the ETH topology.
| | +--rw access-ltp-id? te-types:te-tp-id	The TE link termination point identifier, used together with access-node-id to identify the access LTP.
...	
| +--rw service-classification-type? identityref	Service classification type
| +--rw (service-classification)?	Access classification can be port-based or VLAN based
| | +--:(vlan-classification)	
| | +--rw outer-tag!	Classifies traffic using the outermost VLAN tag.
| | | +--rw tag-type? etht-types:eth-tag-classify	The tag type used for VLAN classification.
| | | +--rw (individual-bundling-vlan)?	VLAN based classification can be individual or bundling.
| | | +--:(individual-vlan)	
| | | | +--rw vlan-value? etht-types:vlanid	VLAN ID value.
...	
| +--rw (direction)?	Whether the bandwidth profiles are symmetrical or asymmetrical
| | +--:(symmetrical)	The same bandwidth profile is used to describe both the ingress and the egress bandwidth profile.
| | | +--rw ingress-egress-bandwidth-profile	The bandwdith profile used in both directions.
| | | +--rw (style)?	Whether the bandwidth profile is named or defined by value
| | | +--:(named)	Named bandwidth profile.
| | | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile.
| | +--:(asymmetrical)	Ingress and egress bandwidth profiles can be specified.
| | | +--rw ingress-bandwidth-profile	The bandwdith profile used in the ingress direction.
| | | | +--rw (style)?	Whether the bandwidth profile is named or defined by value

| | | +--:(named)	Named bandwidth profile.
| | | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile
| | +--rw egress-bandwidth-profile	The bandwdith profile used in the egress direction.
| | +--rw (style)?	Whether the bandwidth profile is named or defined by value
| | +--:(named)	Named bandwidth profile.
| | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile.
| +--rw vlan-operations	Configuration of VLAN operations.
| +--rw (direction)?	Whether the VLAN operations are symmetrical or asymmetrical
| +--:(symmetrical)	
| +--rw symmetrical-operation	Symmetrical operations. Expressed in the ingress direction, but the reverse operation is applied to egress traffic
| +--rw pop-tags? uint8	The number of VLAN tags to pop (or swap if used in conjunction with push-tags)
| +--rw push-tags	The VLAN tags to push (or swap if used in conjunction with pop-tags)
| +--rw outer-tag!	The outermost VLAN tag to push/swap.
| | +--rw tag-type? etht-types:eth-tag-type	The VLAN tag type to push/swap.
| | +--rw vlan-value? etht-types:vlanid	The VLAN ID value to push/swap.
| | +--rw default-pcp? uint8	The default Priority Code Point (PCP) value to push/swap
| +--rw second-tag!	The second outermost VLAN tag to push/swap.
| +--rw tag-type? etht-types:eth-tag-type	The VLAN tag type to push/swap.
| +--rw vlan-value? etht-types:vlanid	The VLAN ID value to push/swap.
| +--rw default-pcp? uint8	The default Priority Code Point (PCP) value to push/swap
+--rw admin-status? identityref	ETH service administrative state.
+--ro state	Ethernet Service states.
+--ro operational-state? identityref	ETH service operational state.
+--ro provisioning-state? identityref	ETH service provisioning state.

3.3 Service provisioning

3.3.1 Use case 8.1 - E-Line services provisioning

Use Case	E-Line services provisioning

Id	Use Case y.y2
Summary	The HCO shall instantiate end-to-end services among different domains, while the MW CO, based on the HCO request, shall instantiate E-line service inside its domain. This UC is applying to a “island” of MW links where all the NEs are fully managed from a single Domain controller as already explained in UC3.2.1.
Benefits and Motivation	The HCO shall coordinate Services among domains without knowing details on how the services are managed inside any specific domain (IP or MW). This simplifies architecture and the volume of information to be shared through the NBI interface.
Detailed description	Based on the HCO request, the MW CO shall instantiate E-line service between eeLTPs The HCO can instantiate a single service, providing all attributes described below to the CO, or create multiple services only replicating the attributes for each service and send it to the CO in “one shot”. Details about the domain-internal transport mechanism or where which VLAN operations are performed within the MW domain. The HCO is to coordinate E2E services across multiple domains, i.e. between eeLTPs Information send to HCO by MW CO. The Information send to MW CO by HCO is: 1. External End Points (eeLTPs) 2. VLAN ID: C-Vlan, S-Vlan, untagged 3. SLA: CIR, PIR etc… 4. Ingress, egress Bandwidth
Proposed implementation IETFData models	The HCO/CO at NBI shall use YANG models RFC 8345 and RFC 8795 and the Data Model for Ethernet TE Topology, defined in the "ietf-eth-te-topology" YANG module of “draft-ietf-ccamp-eth-client-te-topo-yang to implement the UC and shall use the following mandatory attributes:

Ethernet TE topology	
Model	**Attributes explanation**
module: ietf-network	
+--rw networks	Serves as a top-level container for a list of networks.
+--rw network* [network-id]	Describes a network. A network typically contains an inventory of nodes, topological information (augmented through the network-topology data model), and layering information
| +--rw network-id network-id	Identifies a network
| +--rw node* [node-id]	The inventory of nodes of this network
| | +--rw node-id node-id	Uniquely identifies a node within the containing network.
| | +--rw nt:termination-point* [tp-id]	A termination point can terminate a link. Depending on the type of topology, a termination point could, for example, refer to a port or an interface
| | | +--rw nt:tp-id tp-id	Termination point identifier
| | | +--rw tet:te-tp-id? te-types:te-tp-id	An identifier that uniquely identifies a TE termination point
…	
| | | +--rw ethtetopo:port-vlan-id? etht-types:vlanid	Port VLAN ID of the LTP

The HCO/CO at NBI shall use the Transport Network Client Signals Model, defined in the "etht-svc" YANG module of “draft-ietf-ccamp-client-signal-yang” to implement the UC for E-Line information and shall use the following mandatory attributes:

Ethernet Service	
Model	**Attributes explanation**
module: ietf-eth-tran-service	
+--rw etht-svc	ETH services.
+--rw globals	Globals Ethernet configuration data container
| +--rw named-bandwidth-profiles* [bandwidth-profile-name]	List of named bandwidth profiles used by Ethernet services.
| +--rw bandwidth-profile-name string	Name of the bandwidth profile.
| +--rw bandwidth-profile-type? etht-types:bandwidth-profile-type	The type of bandwidth profile.
| +--rw CIR? uint64	Committed Information Rate in Kbps
| +--rw CBS? uint64	Committed Burst Size in in Kbytes
| +--rw EIR? uint64	Excess Information Rate in Kbps. In case of RFC 2698, PIR = CIR + EIR
| +--rw EBS? uint64	Excess Burst Size in KBytes. In case of RFC 2698, PBS = CBS + EBS
| +--rw color-aware? boolean	Indicates weather the color-mode is color-aware or color-blind.
| +--rw coupling-flag? boolean	Coupling Flag
+--rw etht-svc-instances* [etht-svc-name]	The list of p2p ETH service instances
+--rw etht-svc-name string	Name of the ETH service.
...	
+--rw etht-svc-type? etht-types:service-type	Type of ETH service (p2p, mp2mp or rmp).
...	
+--rw etht-svc-end-points* [etht-svc-end-point-name]	The logical end point for the ETH service
| +--rw etht-svc-end-point-name string	The name of the logical end point of ETH service.
...	
| +--rw etht-svc-access-points* [access-point-id]	List of the ETH trasport services access point instances.
| | +--rw access-point-id string	ID of the service access point instance
| | +--rw access-node-id? te-types:te-node-id	The identifier of the access node in the ETH topology.
| | +--rw access-ltp-id? te-types:te-tp-id	The TE link termination point identifier, used together with access-node-id to identify the access LTP.
...	
| +--rw service-classification-type? identityref	Service classification type
| +--rw (service-classification)?	Access classification can be port-based or VLAN based

| | +--:(vlan-classification)	
| | +--rw outer-tag!	Classifies traffic using the outermost VLAN tag.
| | | +--rw tag-type? etht-types:eth-tag-classify	The tag type used for VLAN classification.
| | | +--rw (individual-bundling-vlan)?	VLAN based classification can be individual or bundling.
| | | +--:(individual-vlan)	
| | | | +--rw vlan-value? etht-types:vlanid	VLAN ID value.
...	
| +--rw (direction)?	Whether the bandwidth profiles are symmetrical or asymmetrical
| | +--:(symmetrical)	The same bandwidth profile is used to describe both the ingress and the egress bandwidth profile.
| | | +--rw ingress-egress-bandwidth-profile	The bandwdith profile used in both directions.
| | | +--rw (style)?	Whether the bandwidth profile is named or defined by value
| | | +--:(named)	Named bandwidth profile.
| | | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile.
| | +--:(asymmetrical)	Ingress and egress bandwidth profiles can be specified.
| | | +--rw ingress-bandwidth-profile	The bandwdith profile used in the ingress direction.
| | | | +--rw (style)?	Whether the bandwidth profile is named or defined by value
| | | | +--:(named)	Named bandwidth profile.
| | | | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile
| | | +--rw egress-bandwidth-profile	The bandwdith profile used in the egress direction.
| | +--rw (style)?	Whether the bandwidth profile is named or defined by value
| | +--:(named)	Named bandwidth profile.
| | | +--rw bandwidth-profile-name? leafref	Name of the bandwidth profile.
| +--rw vlan-operations	Configuration of VLAN operations.
| +--rw (direction)?	Whether the VLAN operations are symmetrical or asymmetrical
| +--:(symmetrical)	
| +--rw symmetrical-operation	Symmetrical operations. Expressed in the ingress direction, but the reverse operation is applied to egress traffic
| +--rw pop-tags? uint8	The number of VLAN tags to pop (or swap if used in conjunction with push-tags)

| +--rw push-tags	The VLAN tags to push (or swap if used in conjunction with pop-tags)
| +--rw outer-tag!	The outermost VLAN tag to push/swap.
| | +--rw tag-type? etht-types:eth-tag-type	The VLAN tag type to push/swap.
| | +--rw vlan-value? etht-types:vlanid	The VLAN ID value to push/swap.
| | +--rw default-pcp? uint8	The default Priority Code Point (PCP) value to push/swap
| +--rw second-tag!	The second outermost VLAN tag to push/swap.
| +--rw tag-type? etht-types:eth-tag-type	The VLAN tag type to push/swap.
| +--rw vlan-value? etht-types:vlanid	The VLAN ID value to push/swap.
| +--rw default-pcp? uint8	The default Priority Code Point (PCP) value to push/swap
+--rw admin-status? identityref	ETH service administrative state.

4 [Annex A] - ONF implementation. General aspects

This section aims at providing a general reference for ONF implementations, which will be generally applicable to any of the use cases described in this document. The specifics applicable to each use case are included within each use case section.

4.1 General ONF MW model: key requirements and object summary

The ONF TR-532 MW model is composed of multiple MW technology specific augments (PACs) that are implemented augmenting the TR-512 ONF Core Information Model as base underlying model. Latest Core Information Model corresponds to v1.4, is available at ONF site here, including the complete documentation (it is recommended to start with *TR-512.1_OnfCoreIm-Overview.pdf* as it provides an introduction and guide to the structure of the documents included).

TR-532 MW model PACs are available, toghether with additional resources in this github repository. Base classes of the core model are then augmented with these MW specific models with the target of providing a comprehensive coverage of the key features and configuration aspects of current MW devices, harmonizing their implementation aspects.

4.1.1 ONF Core model TR-512

Implementations need to consider the simplified (pruned and refactored) Core Model v1.4 version available in https://github.com/openBackhaul/core/tree/tsp (latest YANG is available here). Support is mandatory for the MW use case implementation.

The core model documentation provides a detailed description of all the objects and classes. In any case, considering the use cases included in this document, **a high-level general reference of the key classes that need to be implemented to achieve a correct use case implementation follows**.

Because of their abstraction, ONF CIM classes can be used to model different elements or domains. Here, they will be briefly introduced linked to the way in which they have been defined to model MW devices according to the v1.4 pruned and refactored model indicated before:

GlobalClass, LocalClass and State

ONF CIM includes two abstract classes, globalClass and localClass that provide names and identifiers to other CIM classes, that inherit from them depending on their nature. A globalClass represents elements which can exist in their own independently of any others. A LocalClass represents elements inseparable from a GlobalClass, but that may have associations with other instances.

The main global and local class naming and identifying parameters that need to be supported for the MW use case implementations described in this document are:

- UUID: universal unique identifier (global classes). Will serve to identify uniquely global class instances.
- LocalId: identifier unique in a local scope. For example, a localId of a localClass instance is unique in the context of the GlobalClass instance from which it is inseparable.

- Name: this object is a list of *valueName* and *value* pairs, that provide flexibility to implement several naming fields descriptive of the object in which they are included. E.g:
  - Name[1].*valueName* = “Element Name”
  - Name[1].*value* = “MWxxx12345”
- Extension: this object is also a list of *valueName* and *value* pairs, in this case to extend the class with additional properties not directly declared in it. E.g:
  - Extension[1].*valueName* = “IPAddress”
  - Extension[1].*value* = “vvv.xxx.yyy.zzz”

Apart from naming and identification, global and localClass inherit from an additional state object *State_Pac*, which serves to include also some relevant parameters related to the operability, usability and current state of the elements. As key parameters relevant for theMW use cases:

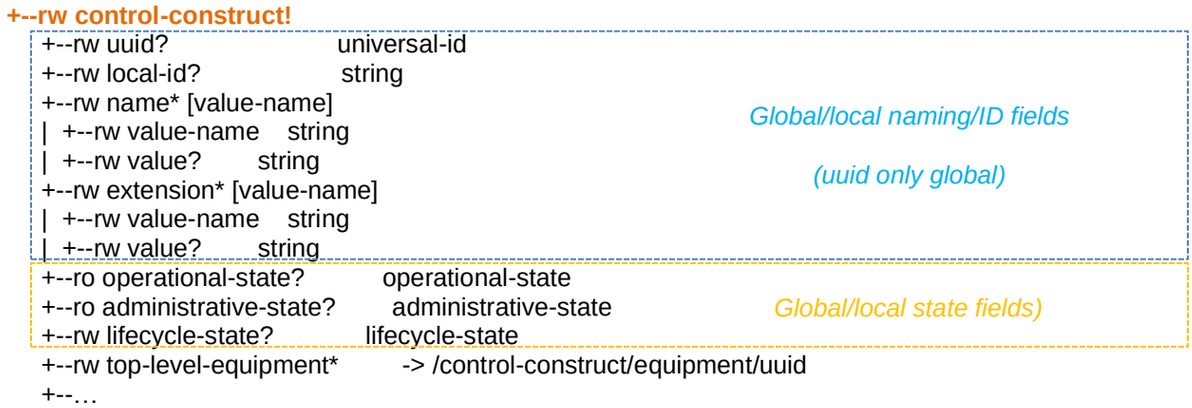
- LifecycleState: to track the planned deployment of the resource
- OperationalState: used to indicate if the resource is installed and working
- AdministrativeState: to represent usage permissions for the resource

All instances of the each of the classes that will be introduced next shall include these (naming and state), depending on their global or local type. A complete reference is available within ONF documentation, section *TR-512.3_OnfCoreIm-Foundation*

ControlConstruct

Amongst other uses within ONF CIM, this global class can be used to model the NE control function. MW network elements will be represented then by a root controlConstruct container, which will include all the objects needed for the complete NE modelling (all the ones that will be described next).

Apart from the aforementioned naming, identification and state parameters related to global classes another relevant parameter for the MW use cases is the *top-level-equipment* reference. This being a pointer to the root equipment object (to be described next), as for example typically the chassis in an split MW NE.



LogicalTerminationPoint and layerProtocol

The controlConstruct shall include a list of logicalTerminationPoints (LTPs), which represent

termination and adaptation functions of one or more transport layers, which are represented by embedded instances of LayerProtocol (LP).

LP instances represent one of the key points for the MW specific technology augmentation of the TR-532 MW model. Here, the layerProtocolName parameter will serve to define the type of transport layer corresponding to the object and will serve as the condition to augment the LP with the specific TR-532 PAC (for example, LTPs corresponding to an air interface will have a LP with an specific layer protocol name that will serve to augment conditionally the LP with the specific TR-532 air interface model).

The encapsulated transport layers have a client-server relationship, which are supported by the LTP parameters *client-ltp* and *server-ltp*, both pointers to another LTP object. Client-server relations are required for the correct implementation of the MW use cases. Both 1:1 and n:1 relationships between client and server need to be modeled for MW model implementation, so the transport layers need to be split over separate instances of LTP (each LTP instance will include a single element layerProtocol list, of a type that corresponds to the modeled transport layer).

MW specific parameters corresponding to the different transport layers for the different interfaces of the MW equipment are then modeled as stacks of related (client/server) LTPs with embedded LPs of the specific type, augmented by an specific TR-532 PAC (depending on the layer type). Lowest LTP/LPs will correspond to the MW element air and wire interfaces. TR-532 PACs, and the layered structure of the interfaces will be introduced shortly in the next section.

```
+--rw control-construct!
    +--...
    +--rw logical-termination-point* [uuid]
    |  +--[Global naming/ID/state fields]
    |  +--rw server-ltp*              -> /control-construct/logical-termination-point/uuid
    |  +--rw client-ltp*              -> /control-construct/logical-termination-point/uuid
    |  +--rw layer-protocol* [local-id]
    |  |  +--[Local naming/ID/state fields]
    |  |  +--rw layer-protocol-name?
    |  |  +--…
    |  |  +--TR-532 augment - LP PACs, conditional on  layer-protocol-name
```

Several use cases require a link between the LTPs (interfaces) and the physical HW elements supporting them. The ONF LTP class includes a pointer to a physical equipment *physical-port-reference** that serves to link an specific LTP/LP with the actual HW supporting it. However, for the MW use case implementations described in this document, an specific TR-532 augment needs to be considered. The LTP augment was developed extending the previous concept, to serve as pointer no only to a single physical element but multiple ones, and also allowing for the identification (pointer as well) to the specific connector within the physical equipment linked to the interface.

A complete reference of the LTP and LP classes is available within ONF documentation, section *TR-512.2_OnfCoreIm- ForwardingAndTermination*

Equipment

The ControlConstruct shall include a list of equipment instances, that serve to represent the physical elements within the NE, being then essential for inventory use cases amongst others.

Each equipment instance of the list represents the abstract potential for implementing an specific HW element within the NE.

Equipment corresponds to a global class, so the previously introduced naming, identification and state parameters are included and need to be supported for use case implementation. Aside them, support of several key objects and fields within the equipment class are required for the use cases included in this document:

- Expected-equipment (local class): this represents the list of HW elements that may be potentially plugged as this equipment instance (planned equipment still not deployed/plugged, HW compatible with the equipment configuration and definition).
- Actual-equipment (local class): this single element represents the actual hardware element which is installed and active as equipment instance. Both expected and actual equipment include several (same in both) containers with multiple parameters related to the physical aspects of the specific element, from dimensions or thermal rating to element details like serial number, etc. Relevant containers and parameters are described as part of the specific use case implementation sections.
- Connector (local class): this represents a list of physical connectors available at the equipment instance. Apart from the naming and state parameters, each instance includes some specific parameters descriptive of the connector (type, etc.). Specific fields will be detailed in each use case implementation section
- Contained-Holder (local class): this represents a list of physical locations or slots available within the equipment instance that may serve to install other equipment elements of the NE (other equipment instances from the equipment list). Apart from the naming, identification and state fields, a key parameter mandatory for implementation is the *occupying-fru* field. This is a pointer to an uuid of the equipment installed in the holder, and enables the identification of parent-child relations between equipment instances in the NE. Other relevant parameters will be introduced in the specific use case implementatino sections.

Modem boards (either separated modular cards or embedded in a compact equipment) are required to be modelled as elements that host (even being pysically separated elements in many cases) the ODU/RF functionality. As examples:

- An equipment instance that represents a modem board with an IF port, will include a contained-holder list with (at least, depending on other HW like SFPs) a single element holder that has an *occupying-fru* field that points to the UUID of the connected ODU (which shall also be an element within the equipment list)
- An equipment instance that represents a compact chassis with 2 IF ports, will include a contained-holder list with (at least, depending on other HW like SFPs) two holders with *occupying-fru* fields pointing each to the UUID of the corresponding ODU (which shall also be elements within the equipment list)

In addition to this general reference, **detailed aspects and requirements around specific MW equipment implementation are specified in the Openbackhaul transmitterEquipment specification v1.0, which is available here. ONF MW implementations need to be compliant with it** in order to support the use cases introduced in this document.

```
+--rw control-construct!
  +--…
  +--rw equipment* [uuid]
  |  +--[Global naming/ID/state fields]
  |  +--ro is-field-replaceable?          Boolean
```

```
| +--rw expected-equipment* [local-id]
| | +--[Local naming/ID/state fields]
| | +--…
| +--rw actual-equipment
| | +--[Local naming/ID/state fields]
| | +--…
| +--rw connector* [local-id]
| | +--[Local naming/ID/state fields]
| | +--…
| +--rw contained-holder* [local-id]
| | +--[Local naming/ID/state fields]
| | +--rw occupying-fru?          -> /control-construct/equipment/uuid
| | +--…
```

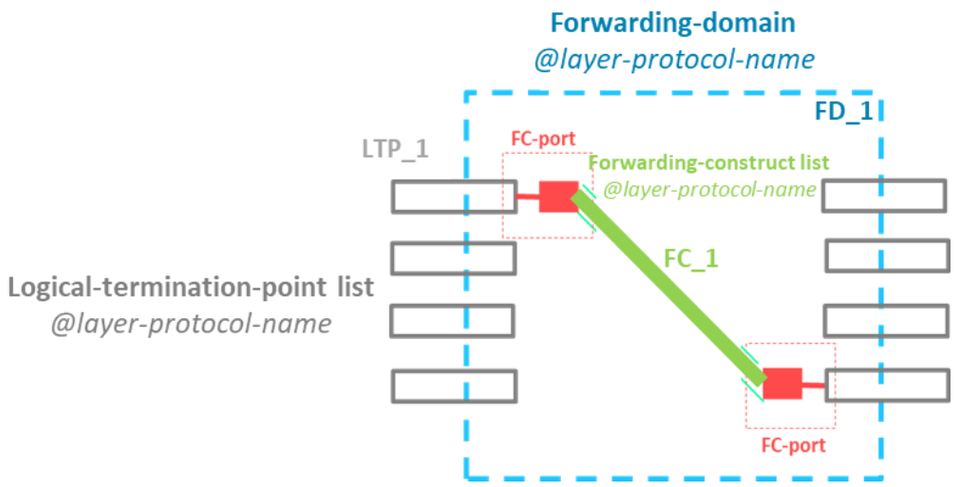
ForwardingDomain and ForwardingConstruct

These classes are needed especially for the implementation of the service provisioning use cases. Within the ONF CIM, the forwardingDomain class (global class) serves to model topological components that have a forwarding capability that provide the opportunity to enable it at one or several protocol layers in their domain (between its ports). The level of abstraction makes it possible to use it to model network-level topologies, but also for other purposes at lower lever, like NE or component level.

In the scope of this document, the forwardingDomain class will be considered to model the internal bridging and switching in the MW NEs, at different levels (MAC / VLAN).

Within the controlConstruct, a list of available forwardingDomains needs to be included, representing the forwarding at the different protocol layers where it is available. ForwardingDomain instances will be considered symmetrical, so they can be directly attached to LTPs, with no need of an specific FD port class.

The ForwardingConstruct class (global class) serves within the ONF CIM to model enabled forwarding between the ports (LTPs within this scope) delimiting the forwardingDomain. A list of the configured forwardingConstructs needs to be available within each forwardingDomain. Each forwardingConstruct instance must include a list of FC-ports (local class) for the association with the LTPs within the forwardingDomain, representing the termination porints between which forwarding is actually enabled. Within this scope, FDs, FCs and FC-ports will be considered symmetric and bidirectional.



```
+--rw control-construct!
  +--...
  +--rw forwarding-domain* [uuid]
  |  +--[Global naming/ID/state fields]
  |  +--rw layer-protocol-name*        layer-protocol-name-type
  |  +--...
  |  +--TR-532 augment - FD PACs, conditional on layer-protocol-name
  |  +--rw fc* [uuid]
  |  |  +--[Global naming/ID/state fields]
  |  |  +--rw layer-protocol-name?              layer-protocol-name-type
  |  |  +--...
  |  |  +--TR-532 augment - FC PACs, conditional on transport layer-protocol-name
  |  |  +--rw fc-port* [local-id]
  |  |  |  +--[Local naming/ID/state fields]
  |  |  |  +--rw logical-termination-point*      -> /control-construct/logical-termination-point/uuid
  |  |  |  +--...
```

Like the LTP/LPs, forwardingDomain and forwardingConstruct are base classes of the ONF CIM that are to be then augmented with technology specific PACs (mainly MAC and VLAN) to model bridging and switching aspects of the MW devices. Both augments to the forwardingDomain and forwardingConstruct classes are part of the TR-532. A brief description of the specific PACs augmenting these classes will be given in the TR-532 section, and specific parameters and requirements for implementation will be given in each use case implementation section.

ProfileCollection

Profiles can be considered as templates, containing specific sets of related attributes which may be applicable simultaneously to different interfaces within the network element. The controlConstruct will include a profileCollection list including all the profile (global class) objects supported and configured in the NE. Apart from the naming, identification and state fields, each profile will include a *profile-name* which will serve to identify its type. Specific rules in relation to profile instanciation can be found in the transmitterEquipment specification.

```
+--rw control-construct!
  +--...
  +--rw profile-collection
  |  +--rw profile* [uuid]
  |  |  +--[Global naming/ID/state fields]
  |  |  +--rw profile-name?          profile-name-type
  |  |  +--TR-532 augment - profile PAC, conditional on profile-name
```

TR-532 includes several MW technology augments for the profile class, to model specific aspects of the MW devices (WRED, Co-channel profiles, etc.). A brief description of those required for the use cases in this document will be given in the next section.

4.1.2 ONF MW model TR-532

As it was introduced before, the ONF Microwave model is composed by multiple technology specific extensions of the ONF CIM, which are implemented by associating specific ONF CIM classes (as layerProtocol, etc.) with conditional packages. These conditional packages are called PACs. Most common attachment point is the LayerProtocol class of the ONF Core IM, for the management of the different network layers at the interfaces of devices.

Model Evolution and current status

Latest official TR-532 release corresponds to version v1.1, published on 2019. This MW model was generated as a result of several years of standardization work, that included a prior v1.0 version (2016) and up to 5 different PoCs between 2015-2019, with participation of more than 10 microwave vendors, several operators and multiple software (application, OSS, etc.) development companies. White papers and reports are available here.

After that, standardization work has continued and modifications to existing TR-532 v1.1 PACs as well as new PACs have been progressively developed to enlarge the use case coverage and provide extended parameter sets to model additional aspects of the microwave devices.

In parallel, specific operator-driven (e.g. Telefonica) projects to test, pilot and introduce MW equipment supporting the ONF MW TR.532 have also contributed and continue contributing to further develop and increase the maturity the models and RESCONF/NETCONF SDN standard interfaces. This maturity, favoured by the continuous support and interoperability tests of these new drafts with major microwave vendors, created a progressive shift from lab testing towards the current real network implementation pilots and introduction projects, already reaching several tens of thousands of ONF interface-enabled devices (mainly with mediation SW) under the management of an SDN controller in some existing networks.

Linked to this evolution, work to consolidate an official v2.0 integrating the latest drafts of the TR-532 PACs is already undergoing. ONF detailed implementations in this document take into consideration the latest available drafts[3] of the TR-532 PACs.

Model structure and PAC overview

All the PACs follow a similar structure, including different classes (presence of all the classes ultimately depends on the type of PAC and specifics of the area of the device or interface / transport layer being modeled):

- Capabilities – parameters linked to the device feature availability, characteristics and parameter value ranges. These are invariant, and configuration of values outside capability ranges should result in error.
- Configuration – parameters corresponding to the actual configuration of the device
- Status – parameters corresponding to the current operational state of the device as well as instantaneous measurement values
- Problems – information related to the problems currently active at the device
- Current Performance – The device informs about the current values of its performance counters
- Historical Performance – The device informs about the values of its performance counters at the end of a well-defined time period, with different granularities

ONF MW model considers separation of config and state parameters, as it derives from the previous PAC class structure. This has been taken into account in the use case reference

[3] As part of the development lifecycle, there might be cases in which modifications to the latest published PACs have already been agreed and approved, but still not translated to a new version of the draft. These cases will be explicitly indicated in the detailed implementation of the use cases.

implementations.

Specific requirements related to the MW model PACs (as required paths / parameters, etc.) will be given in the detailed ONF implementaton section of each use case. However, as a general summary, considering the use case scope of this document, the main PACs that need to be considered for implementation are[4]:

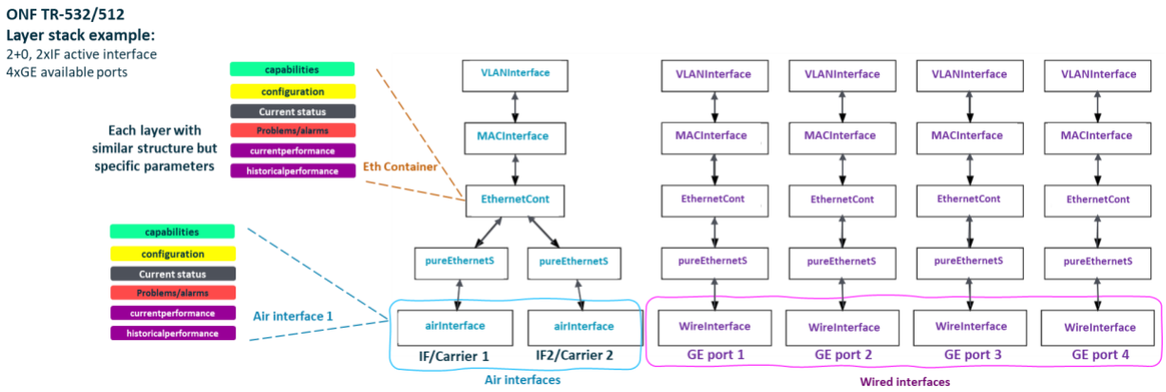
1. Conditional packages associated to the ONF CIM logicalTerminationPoint class:
    - ltpAugment_1.0.0. Serves to link LTPs with physical equipment and their connectors. Relevant for inventory use cases, as an example. Latest yang available here.
2. Conditional packages associated to the ONF CIM layerProtocol class
    - airInterface_2.0.0: models the physical layer of the microwave radio interfaces. Latest available YANG here.
    - wireInterface_2.0.0: models the physical layer[5] of an Ethernet PHY interface (IEEE 802.3). Latest available YANG here.
    - pureEthernetStructure_2.0.0: serves for the structuring of a microwave radio interface into a single Ethernet segment. Latest available YANG here.
    - hybridMWStructure_2.0.0: serves for the structuring of a microwave radio interface into multiple TDM and a single Ethernet segment. Latest available YANG here.
    - ethernetContainer_2.0.0: models interface transport resources for Ethernet. Includes aspects like QoS, queues, traffic, etc. Latest available YANG here.
    - tdmContainer_2.0.0: models interface transport resources for TDM. Latest available YANG here.
    - macInterface_1.0.0: models an Ethernet MAC interface according to IEEE 802.1. Latest available YANG here.
    - vlanInterface_1.0.0: models VLAN interfaces (Port) according to IEEE 802.1Q. Latest available YANG here.
3. Conditional packages associated to the ONF CIM controlConstruct class
    - Firmware_1.0.0: information model and RPCs for managing firmware on the MW devices. Latest available YANG here.
4. Conditional packages associated to the ONF CIM profile class
    - coChannelProfile_1.0.0: Profiles to support modeling of groups of microwave radio interfaces, Latest available YANG here.
5. Conditional packages associated to the ONF CIM forwardingDomain class
    - macFd_1.0.0 : models the device potential forwarding according to IEEE 802.1. Latest available YANG here.
    - vlanFd_1.0.0 : models the device potential forwarding according to IEEE 802.1Q. Latest available YANG here.
6. Conditional packages associated to the ONF CIM forwardingConstruct class
    - vlanFc_1.0.0: models the actual forwarding (VLANs) within the device according to IEEE 802.1Q. Latest available YANG here.
7. Syncronization PAC, with resources available here. This is a particular case, its definition

[4] Github site provides a broader general overview of PACs and resources, with each PAC link including UML, documentation, YANG and additional resources for all available versions

[5] For simplicity, we will consider it within the complete TR-532 model, though the wire interface definition was defined separately in the TR-541 specification.

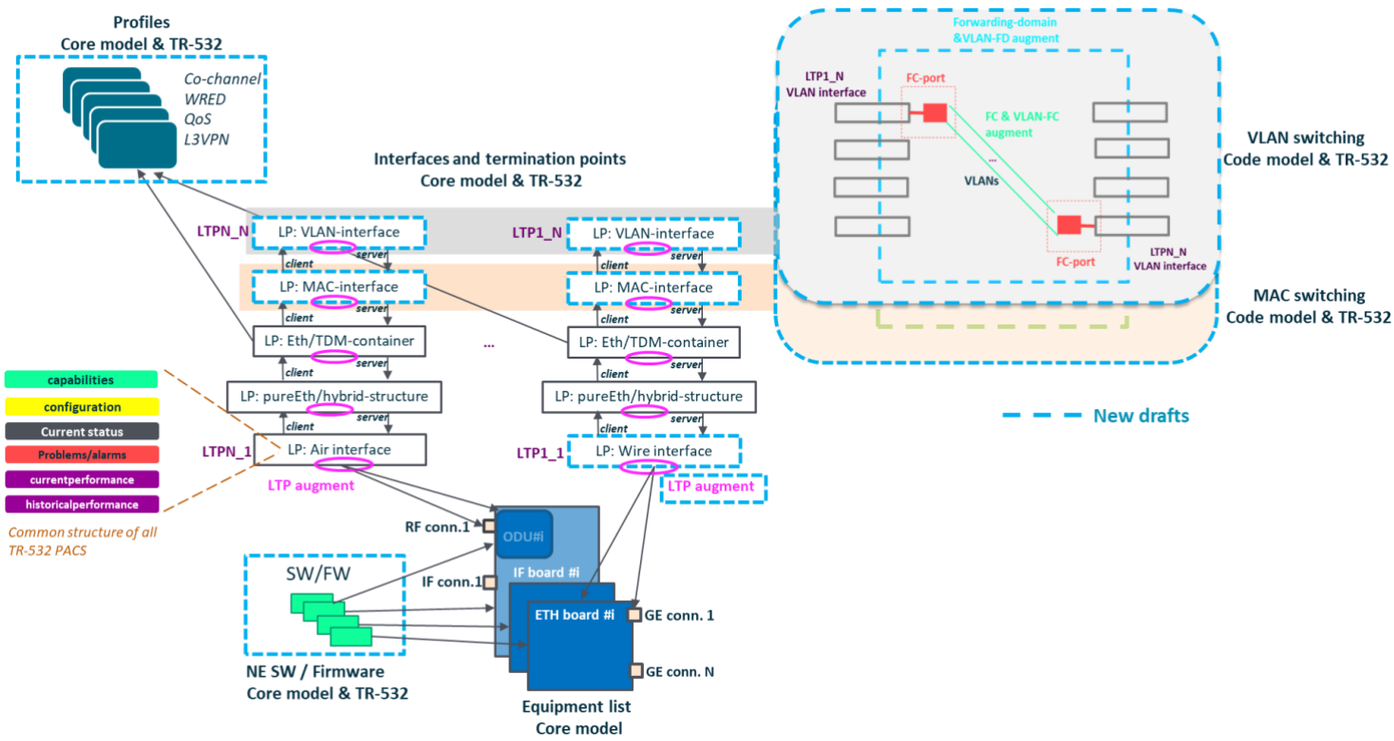
is work ongoing. Design considers ITU-T G.7721-2018 which is already based on ONF core model classes. It includes several specific classes that will be associated to different core model objects (LTPs, ControlConstruct,…)

In terms of transport layering, air and wireInterfaces are the lowest level ones, and will be linked through the ltpAugment to the specific equipment element and connectors corresponding to the interface. On top, subsequent LTPs (with their corresponding LP and TR-532 conditional augment) will be stacked using the client – server parameters available at the LTP class. Air and wire interfaces will not have a server layer. Next figure show ans example of layering for a NE implementing a 2+0 link:



Above phisical layers, the microwave model offers structure PACs (supporting PTP with FDD devices) distinguishing between pure ethernet (which offers a single transport segment of variable size to an ethernet container) and hybrid microwave (which offers multiple fixed size TDM segments, each to be mapped with a tdm container, and a single segment for an ethernet container, of variable size). On top of the ethernet container, LTPs corresponding to MAC, VLAN and IP layers will complete the stack linked to each interface. It must be noted that 1:N relations are possible, typical in the case of bundled N+0 links (multiple air interfaces and structures and single ethernet container) or hybrid links (multiple tdm containers on top of a single air interface and hybrid structure).

The next figure summarizes all the PACs characterizing a microwave NE. Highlighted in the figure, those newer PACs not available in v1.1.



*note: even IF(modem) and RF(ODU) are separated elements, the previous picture shows them together as typically modems (or boards including modem funcionality) are modeled as "holders" of the ODU, capturing the relationship between them in terms of radio transmission.

Default Values

In case of hardware not supporting all functionalities covered by the modeling, the device standard interface shall answer with default values, which have been defined in the model.

Default values in the Microwave Information Model are available and were defined with accordance to the following principles:

- Every attribute except keys shall have a default value.
- The default value shall be inside the value range of the data type of the attribute.
- In case of capability attributes, the default value shall either indicate unavailability of the functionality (if applicable) or be outside the range of reasonable values of the attribute.
- In case of configuration, status and performance attributes, the default value shall either represent the configuration, status or performance measurement value right after starting the device (in case such a "neutral" value is applicable to the attribute) or be outside the range of reasonable values of the attribute.

Notifications & RPCs

Apart from the aforementioned classes (capabilities, configuration, etc.), each TR-532 PAC includes notifications and RPCs (for some specific functions) that need to be supported according to the model definition. Specific aspects will be detailed within each use case detailed implementation section.

In relation to notifications, implementations must be compliant with RFC5277. Controllers

must be able to subscribe to notifications from the NE for the correct implementaion of the use cases.

Generally, for all the PACs, the MW information model considers:

- attributeValueChangeNotification, to be raised when an attribute changes its value. In the Microwave model, it is set on “true” for all attributes within configuration classes or in data types used by their attributes. It has also been set on “true” for status attributes which might be subject to automated changes, but do not represent gradually changing measurement values. It has been set “false” for status attributes which are exclusively following configuration activities (avoiding double messaging) or measuement values.
- objectCreationNotification/objectDeletionNotification, to be raised when an instance of a class has been created or deleted. It has been set on “true” for all “*_Pac” classes.

In addition to these, the problemNotification that has to be raised linked to alarm events. Most of the PACs include a currentProblem class which serves to model raised alarms applicable to the interface and transport layer being modeled, as well as a list of available alarms in the Capabilities class and configured severities (for each available alarm) in the Configuration class. Notifications according to the problemNotification definition, including parameters like the alarm name, timestamp, severity and objectID need to be raised linked to each alarm event.

4.2 TR-545 and additional aspects for implementation

Well-defined models are not sufficient for proper management of a multi-vendor network. The Device Management Interface Profile (DMIP, TR-545) specification was developed (also part of the ONF MW PoCs) to unify the behavior on the southbound interface of the network domain controller to ensure unambiguous states and responses of different vendors’ devices. Implementations need to be compliant with the requirements specified within TR-545, available here.

Devices (and controller in its SBI) are generally required to implement a NETCONF interface according to RFC6241, using SSH (4742) and XML encoding, based on YANG modeling (RFC6020 / RFC7950). Devices must advertise their NETCONF capabilities in an initial exchange according to RFC6241 and support start-up, candidate and running datastores. Subtree filtering is required as part of NETCONF support.

Mediation SW

In case devices do not support natively the standard NETCONF interface requested for implementation, 1:1 mediation SW can be an option for implementation of the interface. This is also relevant for SDN management of legacy devices, which may be an important aspect depending on the specific network, to bring already deployed equipment where a FW upgrade to suppport natively the interface is not possible under the SDN controller domain. This maximizes from the beginning the universe of network elements within the SDN control domain and avoids accelerated swap-outs to deploy network-wide applications that benefits from the SDN approach.

Mediators can also provide some benefits in phases of development of the standard interfaces, as typically accelerated development periods and continuous upgrade are possible. However,

they are an additional element to manage and operate so, it is key that mediators are implemented in a resource efficient way (e.g several thousands within a single VM) and hardened to support typical network operation environments. Although TR-545 already gave a reference, current live implementation experiences provide a much more up-to-date guideline. Current mediator VMs can support support more than 6.000 instances or connected devices on a single VM having 64vCPU / 600GB RAM, with 50% resource utilization when requesting 10 attributes from all the devices every 5 min in parallel.

In this case, additionally, mediation management systems are also needed as part of the solution, specially for large scale implementations. Standardized mediation management system specifications, according to standardized REST API definitions are also under development.

Available resources

Many of the PACs include RESTCONF based interface validators (available in each PAC github.com site) to allow for the automatic validation of the equipment (or mediator) interface implementation against the reference model specified in the respective TR document. This eases and accelerates the development of the interface support and their usage has been key to gain maturity in the Microwave model and the ONF standard interface development. ONF 2.0 drafts have already been tested in several models of major MW vendors using these types of resources amongst others.

Microwave vendors or network operators can make use of these resources to validate equipment support and ease their integration process in multi-vendor network(s).

Other resources like open-source ODL controller instances used within specific projects like ONAP or new specifications for the controller interface towards MW application layer, to support the creation of a flexible harmonized layer of applications on top of SDN controllers are also available or under development in the openbackhaul site.

5 [Annex B] – Inter-domain Auto Discovery – LLDP algorithm.

This section aims at providing a general reference for implementation of an algorithm based on LLDP protocol to calculate automatically the “inter-domain-plug-id” value; that allows the HCO to identify inter-domain links between domains and manage end to end services.

5.1 LLDP protocol and algorithm description

The Link Layer Discovery Protocol (LLDP) is a Vendor agnostic link layer protocol used by equipment to share information of its identity and capabilities in a network.

LLDP uses Ethernet frame information exchanged by equipment. The frame contains LLDP Data Unit that are sequences of type_length_value (TLV).

LLDP Ethernet frame structure									
Preamble	Destination MAC	Source MAC	Ethertype	Chassis ID TLV	Port ID TLV	Time to live TLV	Optional TLVs	Optional End of LLDPDU TLV	Frame check sequence

Some are mandatory TLVs: Chassis ID, Port ID, and Time-to-Live. The mandatory TLVs are followed by optional TLVs.

For security related aspects, it is typically mandatory to ensure that network elements and LLDP related YANG models offer the possibility of switching on/off the LLDP feature at element level to not have it running continuously (protocol exposes sensitive info from the network, as topology can be derived from it) and also to allow for switch off at individual interface level, to disable for example in interfaces facing customer ports (UNI), which are typical requirements set by MNO areas managing security.

The algorithm (as for Figure 7) allows Domain Controllers to calculate the same unique value, without any information exchange between other Controllers and using only mandatory LLDP TVLs.

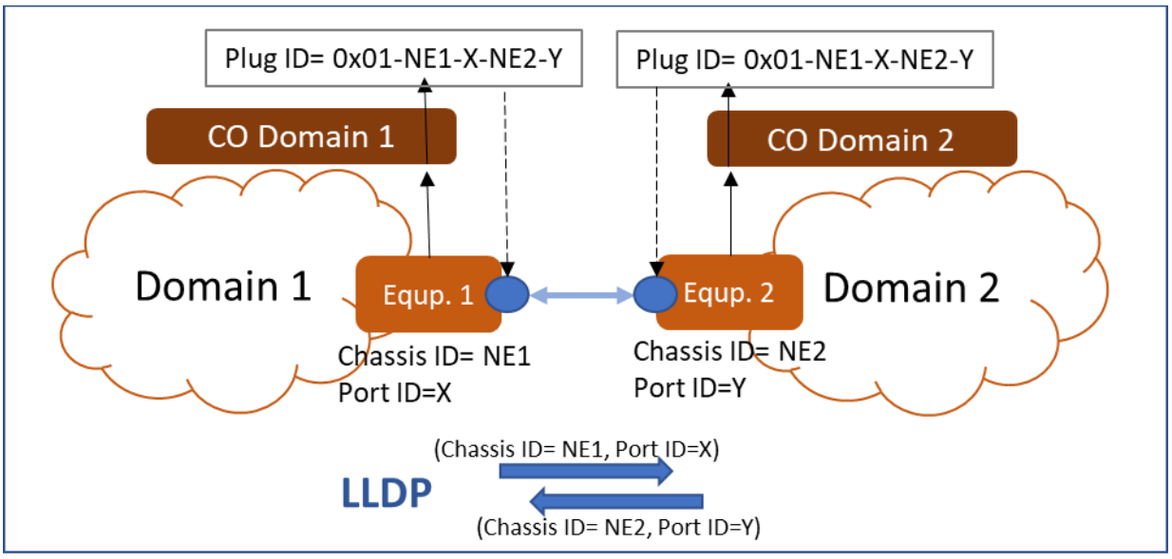


Figure 7: inter-domain auto discovery algorithm

The detailed procedure corresponding to the previous figure, for domain controller 1 and 2 would be:

1. Creating the sequence of values Chassis ID, prefixed by Chassis ID subtype code, and Port ID, prefixed by Port ID subtype code
2. LLDP protocol sends sequence to the other domain, (Token T1)
3. Creating the sequence of values of Chassis ID, prefixed by Chassis ID subtype code, and Port ID, prefixed by Port ID subtype code
4. LLDP protocol receives sequence from the other domain (token T2)
5. Comparing T1 and T2, and concatenates them in the order from bigger to smaller
6. Prefixing a byte of value 0x01 and obtaining [0x01,T1,T2] or [0x01,T2,T1]
7. Domain controllers 1 and 2 calculate the same Plug ID values that the HCO use for establishing stitching point and inter-domain links.

This algorithm also serves to provide automatic discovery capabilities (so, no manual link/parameter configuration) in the topology related use cases, both in black-box or grey box approaches.

Glossary

API	Application Programming Interface
DCI	Data Center Interconnection
CAPEX	Capital Expenditure
DCN	Data Communication Network
DWDM	Dense Wavelength Division Multiplexing
EOE	Electrical-Optical-Electrical
FEC	Forward Error Correction
GE	Gigabit Ethernet
HAL	Hardware Abstraction Layer
HW	Hardware
L0/L1	Layer 0 and Layer 1
LAN	Local Area Network
MAN	Metropolitan Area Networks
NMS	Network Management System
MNO	Mobile Network Operator
MUST	Mandatory Use Case Requirements for SDN Transport
NMDA	Network Management Datastore Architecture
NOS	Network Operating System
OCP	Open Compute Project
OLS	Open Line System
ONIE	Open Network Install Environment
OTN	Optical Transport Network
ROADM	Reconfigurable Optical Add-Drop Multiplexer
SAN	Storage Area Network

SDH Synchronous Digital Hierarchy

SDN Software Defined Network

SW Software

TAI Transponder Abstraction Interface

TIP Telecom Infra Project

TRS Technical Requirement Specification

WDM Wavelength Division Multiplexing

ZTP Zero Touch provisioning