On defining ONU Service Port Capability attribute

Glen Kramer

From June 1904.4 meeting …

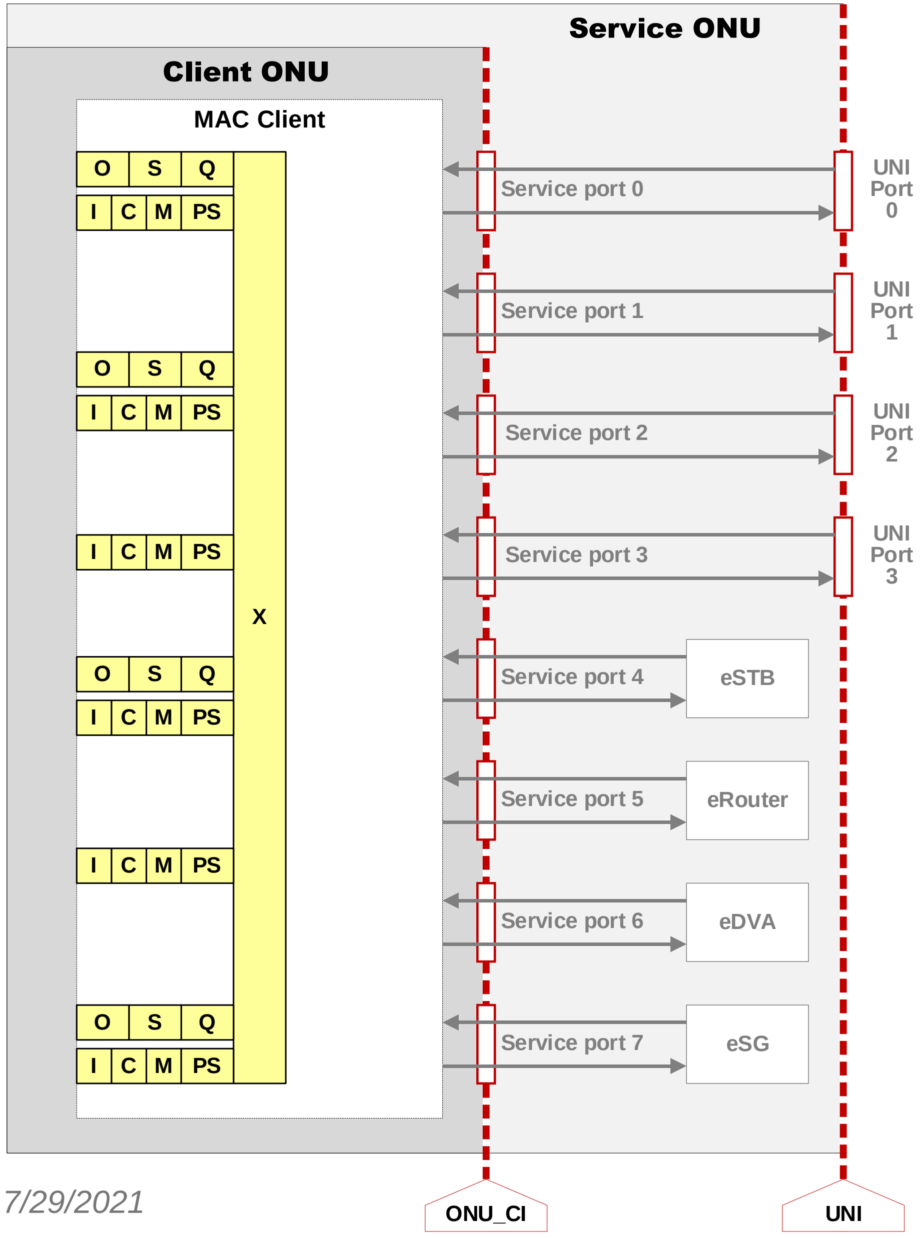
Discussion on the floor

- ❑ Service Port types should identify specific UNI port instance.

  For example,

  1. UNI ports always reported first (lower indexes represent UNIs)
  2. Additional TLV to query UNI index for specific Service Port
  3. Reserve range of code-points for UNIs where code point value = UNI index.

Method #1: Service port index = UNI port index



← This ONU is capable of supporting 8 service ports.

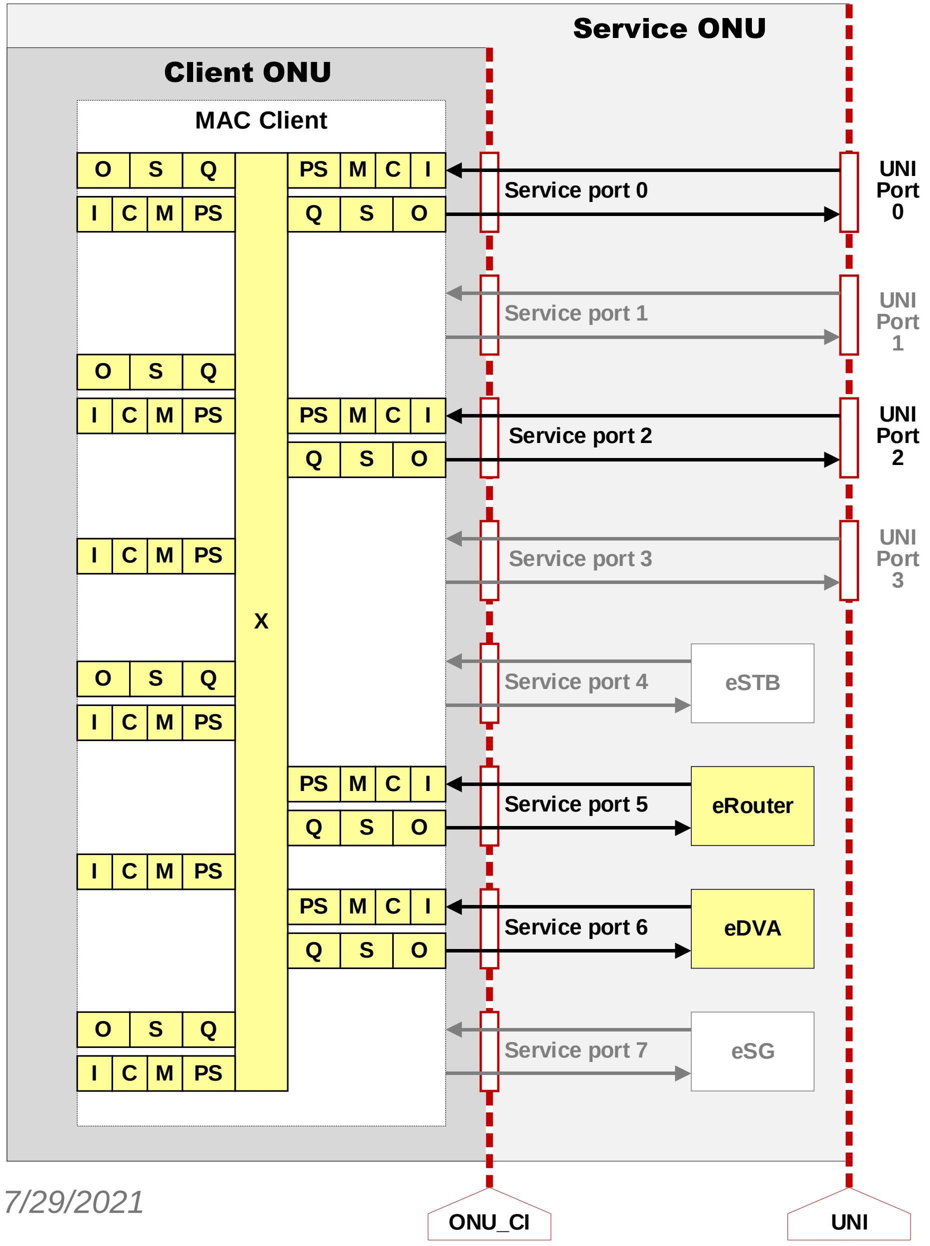
None of the ports have been provisioned yet.

For the given ONU design, the *aOnuSrvPortCapability* attribute always provides the same response →

***aOnuSrvPortCapability* attribute response**

TLV field Value	TLV field Description
0xDB	Branch
0x00-10	Leaf
8	Length
uni_port	Type of service port [0] is UNI #0
uni_port	Type of service port [1] is UNI #1
uni_port	Type of service port [2] is UNI #2
uni_port	Type of service port [3] is UNI #3
estb	Type of service port [4] is sSTB
erouter	Type of service port [5] is eRouter
edva	Type of service port [6] is eDVA
esg	Type of service port [7] is eSG

Method #1 (continued)



← Service ports 0, 2, 5, and 6 were provisioned.

aSrvPortType attribute response

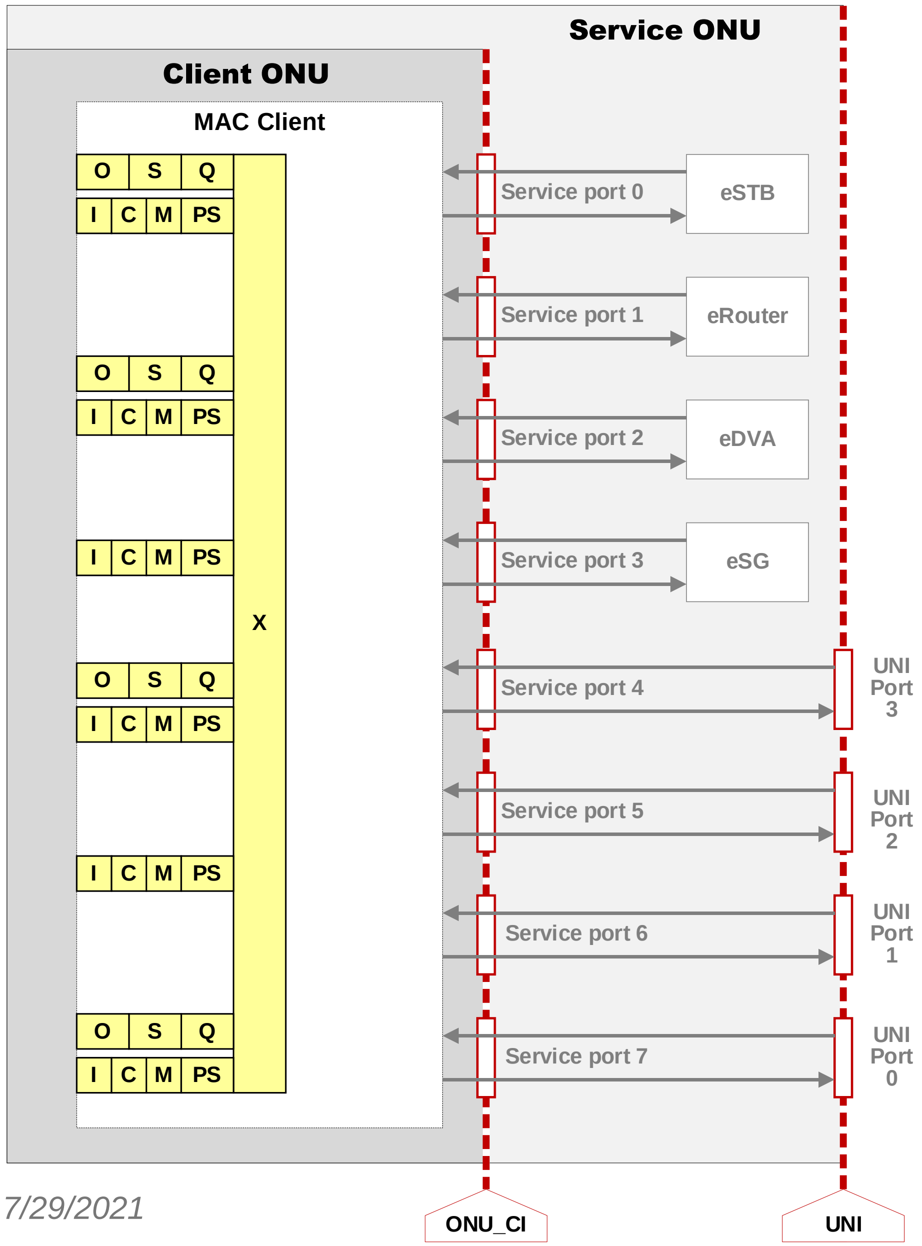
TLV field Value	TLV field Description
0xDB	Branch
0x01-21	Leaf
8	Length (2x4)
0	Type of service port [0] …
uni_port	… UNI #0
2	Type of service port [2] is …
uni_port	… UNI #2
5	Type of service port [5] is …
erouter	… eRouter
6	Type of service port [6] is …
edva	… eDVA

Issue #1 – ONU design constraint

- The Method #1 required Service Port indexes to match external marking of UNI ports.
- What if ONU is designed in such a way that Service Port indexes don’t correspond o UNI port indexes?
- Two solutions:

1. OAM client has to fake service port indexes that match UNU port indexes. Internally, map ‘external’ indexes to local hardware indexes.
2. Specify separate uni_port type code points for each UNI port index
   - `uni_port_0`
   - `uni_port_1`
   - `uni_port_2` …

Method #2: Multiple code-points



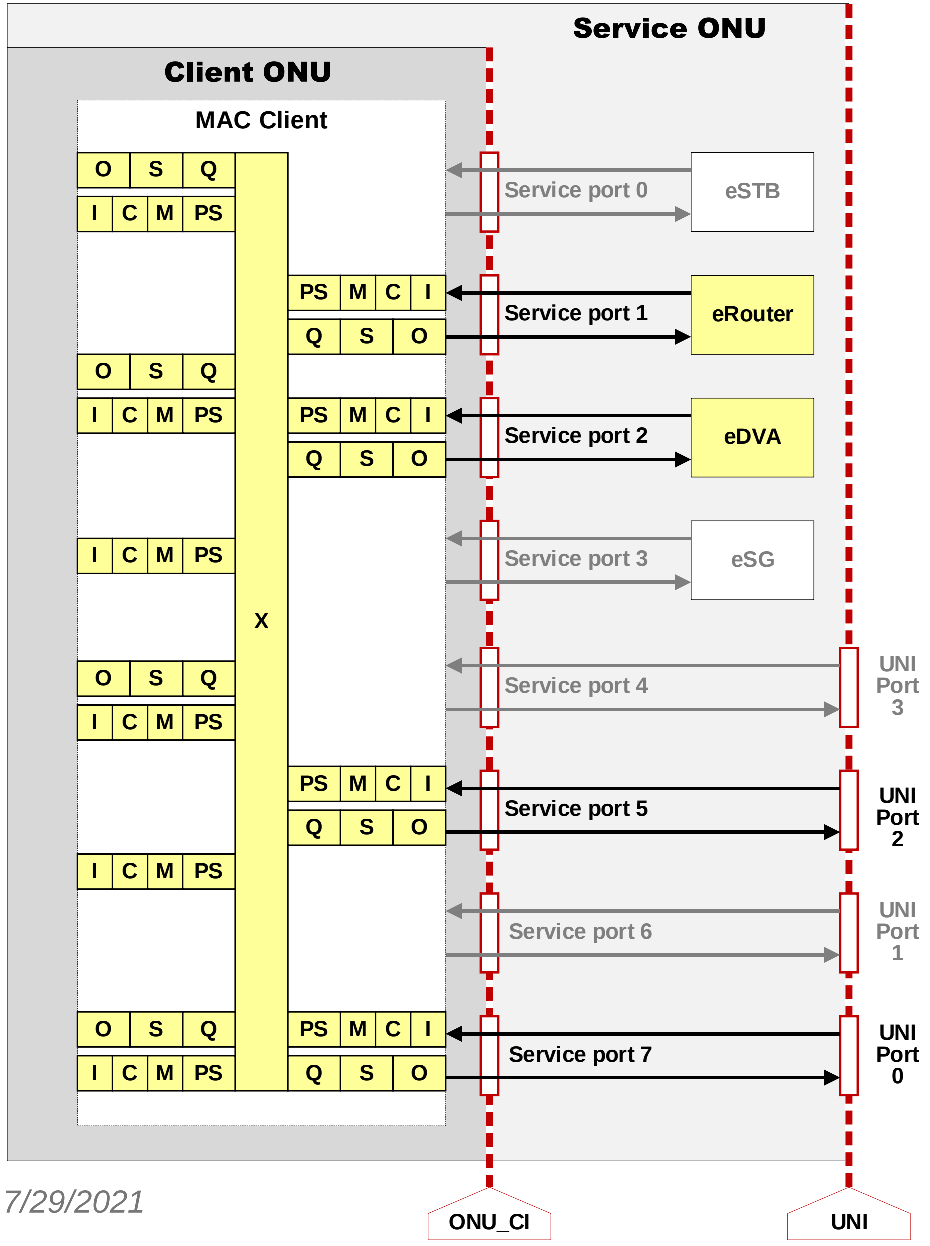
← This ONU is capable of supporting 8 service ports.

← Service port indexes don't match UNI indexes

***aOnuSrvPortCapability* attribute response**

TLV field Value	TLV field Description
0xDB	Branch
0x00-10	Leaf
8	Length
`estb`	Type of service port [0] is sSTB
`erouter`	Type of service port [1] is eRouter
`edva`	Type of service port [2] is eDVA
`esg`	Type of service port [3] is eSG
`uni_port_3`	Type of service port [4] is UNI #3
`uni_port_2`	Type of service port [5] is UNI #2
`uni_port_1`	Type of service port [6] is UNI #1
`uni_port_0`	Type of service port [7] is UNI #0

Method #2 (continued)



← Service ports 1, 2, 5, and 7 were provisioned.

***aSrvPortType* attribute response**

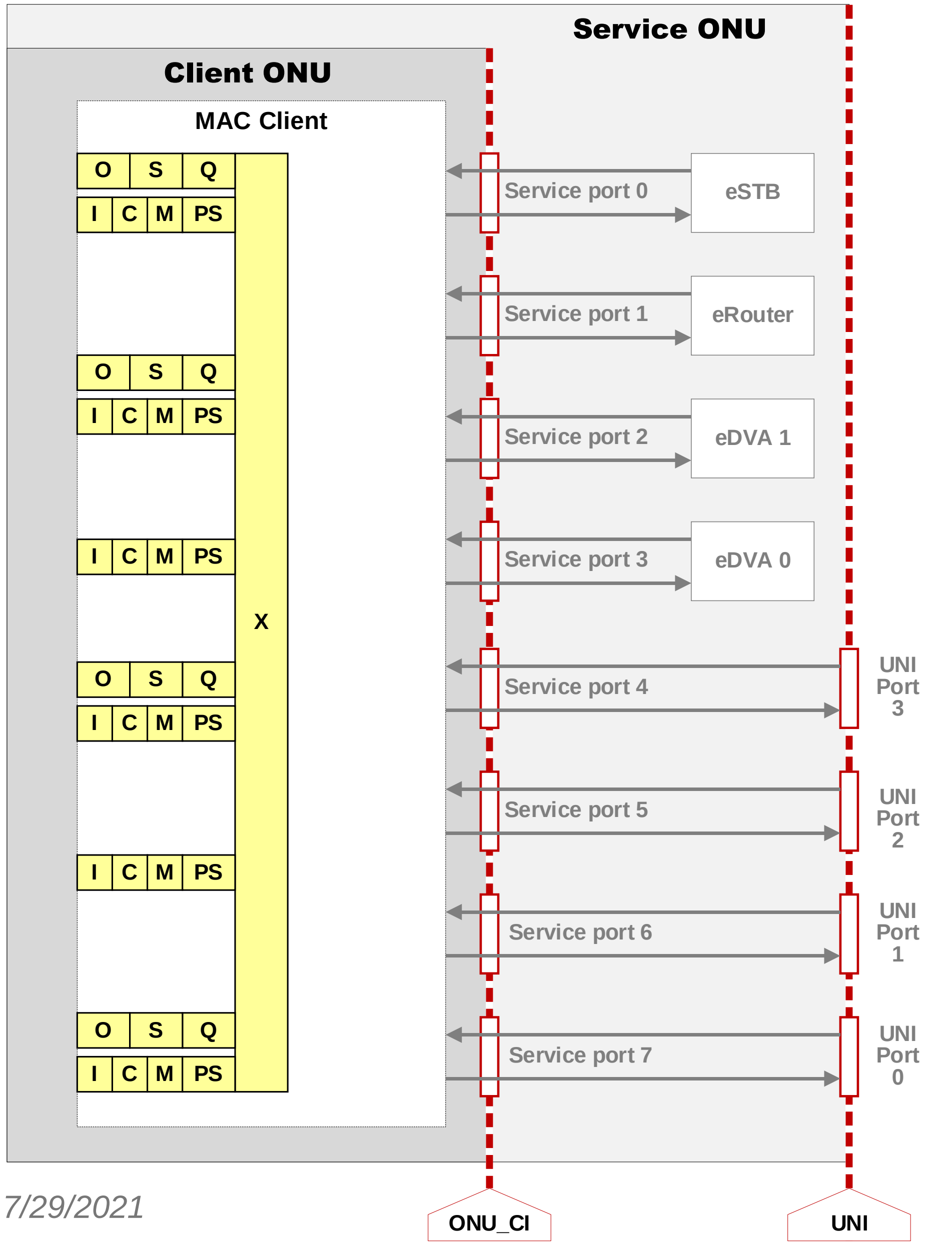
TLV field Value	TLV field Description
0xDB	Branch
0x01-21	Leaf
8	Length (2x4)
1	Type of service port [1] is …
erouter	… eRouter
2	Type of service port [2] is …
edva	… eDVA
5	Type of service port [5] is …
uni_port_2	… UNI #2
7	Type of service port [7] is …
uni_port_0	… UNI #0

Issue #2 –Multiple instances of other types

- The above example can identify UNI port instances without requiring service port indexes to match UNI port indexes.

- What if we have other port types (now or in the future) that may exist in multiple instances?
  - Method #1 doesn’t work. Different port types may use the same index (i.e., UNI[1] and eDVA[1]), but there is only a single service port with the index 1.
  - Method #2 may work, but would need to add new code-points every time we decide a type can be multi-instance

Method #3: Independent 'Instance' field



← This ONU is capable of supporting 8 service ports.

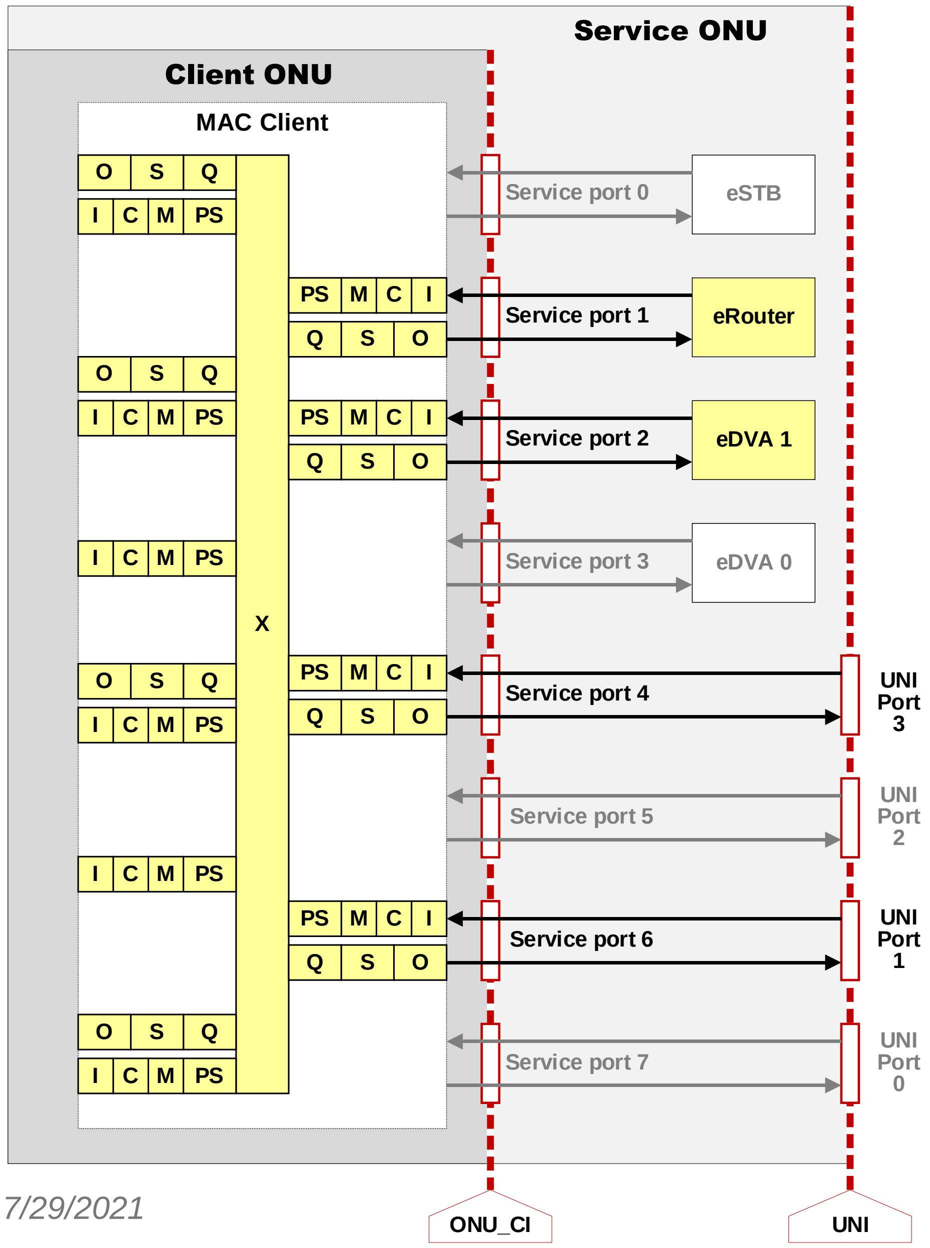
← Service port indexes don't match UNI indexes

← Multiple instances of various port types (eDVA, UNI port)

aOnuSrvPortCapability attribute response

TLV field Value	TLV field Description
0xDB	Branch
0x00-10	Leaf
16	Length (2x8)
estb	Type of service port [0] is sSTB …
0	… with instance index 0
erouter	Type of service port [1] is eRouter …
0	… with instance index 0
edva	Type of service port [2] is eDVA …
1	… with instance index 1
edva	Type of service port [3] is eDVA …
0	… with instance index 0
uni_port	Type of service port [4] is UNI …
3	… with instance index 3
uni_port	Type of service port [5] is UNI …
2	… with instance index 2
uni_port	Type of service port [6] is UNI …
1	… with instance index 1
uni_port	Type of service port [7] is UNI …
0	… with instance index 0

Method #3 (continued)



← Service ports 1, 2, 4, and 6 were provisioned.

aSrvPortType attribute response

TLV field Value	TLV field Description
0xDB	Branch
0x01-21	Leaf
12	Length (3x4)
1	Type of service port [1] is …
erouter	… eRouter with …
0	… index 0
2	Type of service port [2] is …
edva	… eDVA with …
1	… index 1
4	Type of service port [5] is …
uni_port	… UNI with …
3	… index 3
6	Type of service port [7] is …
uni_port	… UNI with …
1	… index 1

Method #4: Additional 'Instance' attribute

Assume the same ONU design as in Method #3

Step 1: Query Service Port Type

Query Request

Context = ONU

TLV field Value	TLV field Description
0xDA	Identification branch
0x00-00	Object is ONU
1	Length
0	ONU Instance

'Type' Request

TLV field Value	TLV field Description
0xDB	Identification branch
0x00-10	Leaf of 'Type' attribute

Query Response

Context = ONU

TLV field Value	TLV field Description
0xDA	Identification branch
0x00-00	Object is ONU
0x01	Length
0	ONU instance

'Type' Response

TLV field Value	TLV field Description
0xDB	Branch
0x00-10	Leaf
8	Length
`estb`	Type of service port [0] is sSTB
`erouter`	Type of service port [1] is eRouter
`edva`	Type of service port [2] is eDVA
`edva`	Type of service port [3] is eDVA
`uni_port`	Type of service port [4] is UNI
`uni_port`	Type of service port [5] is UNI
`uni_port`	Type of service port [6] is UNI
`uni_port`	Type of service port [7] is UNI

Method #4: Additional 'Instance' attribute

Step 2: Query 'Instance of Type'

Assume the same ONU design as in Method #3

This exchange needs to be repeated for every service port

Query Request

TLV field Value	TLV field Description
0xDA	Identification branch
0x00-03	Object is Service Port
1	Length
4	Service Port Instance

Context = SrvPort Instance

TLV field Value	TLV field Description
0xDB	Identification branch
TBD	Leaf of 'Instance of Type' attribute

'Instance of Type' Request

Query Response

Context = SrvPort Instance

TLV field Value	TLV field Description
0xDA	Identification branch
0x00-03	Object is Service Port
1	Length
4	Service Port Instance

'Instance of Type' Response

TLV field Value	TLV field Description
0xDB	Branch
TBD	Leaf of 'Instance of Type' attribute
1	Length
3	Instance of type of service port 4 is 3

Emerging consensus

1. Method #3 to query types and indexes by type
2. Method #4 to query text labels under the context of Service Port instance.
    1. Text labels reported by TLV “Description” in #2 should unambiguously identify a specific UNI port (location) exposed on the ONU or other types of embedded functions
        1. “UNI 1G (left most)”
        2. “UNI 1G”
        3. …
        4. “UNI 10G (right-most)”

Service Port types

Missing Service Port types in 1904.1 and .4

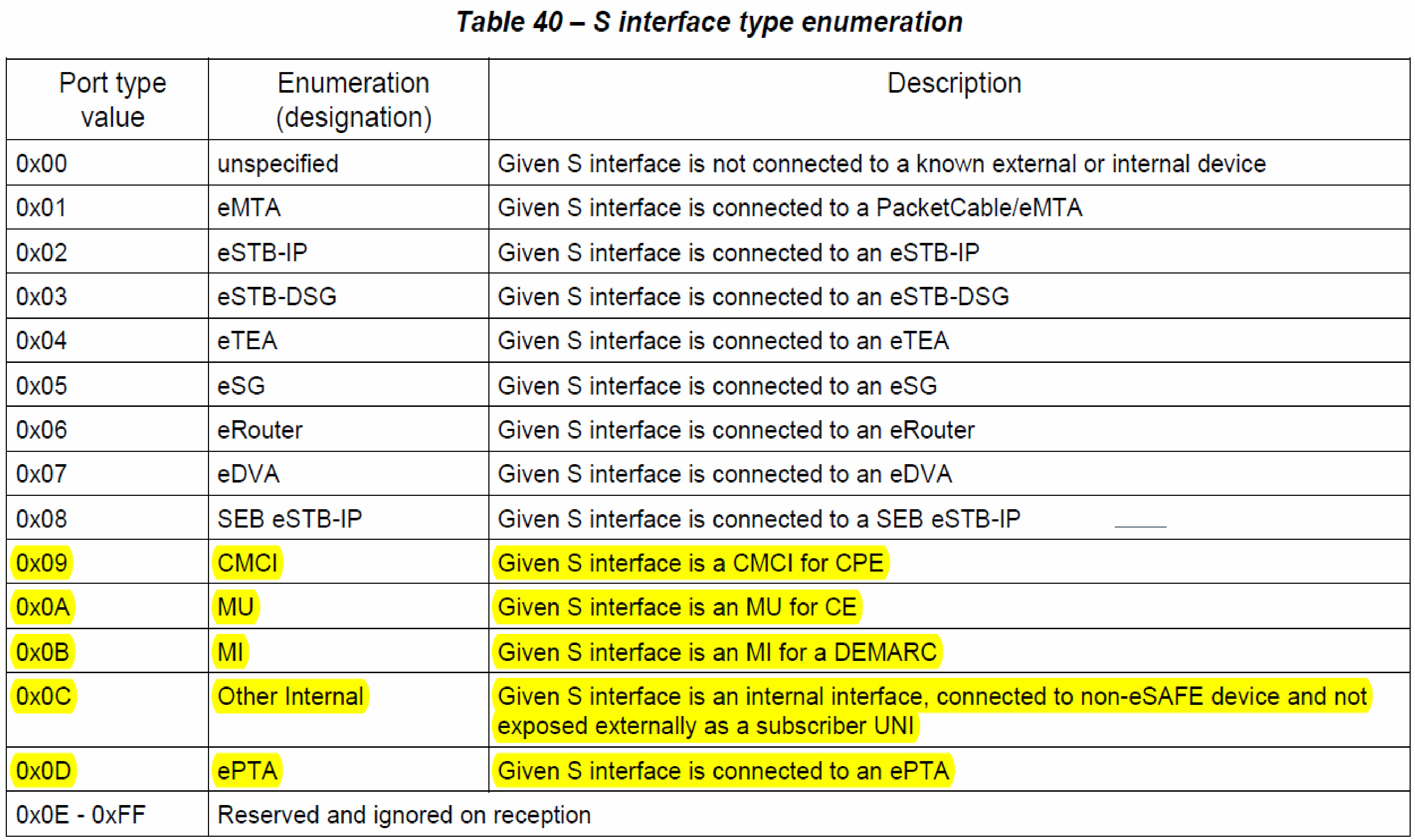


Table 40 – S interface type enumeration

Port type value	Enumeration (designation)	Description
0x00	unspecified	Given S interface is not connected to a known external or internal device
0x01	eMTA	Given S interface is connected to a PacketCable/eMTA
0x02	eSTB-IP	Given S interface is connected to an eSTB-IP
0x03	eSTB-DSG	Given S interface is connected to an eSTB-DSG
0x04	eTEA	Given S interface is connected to an eTEA
0x05	eSG	Given S interface is connected to an eSG
0x06	eRouter	Given S interface is connected to an eRouter
0x07	eDVA	Given S interface is connected to an eDVA
0x08	SEB eSTB-IP	Given S interface is connected to a SEB eSTB-IP
0x09	CMCI	Given S interface is a CMCI for CPE
0x0A	MU	Given S interface is an MU for CE
0x0B	MI	Given S interface is an MI for a DEMARC
0x0C	Other Internal	Given S interface is an internal interface, connected to non-eSAFE device and not exposed externally as a subscriber UNI
0x0D	ePTA	Given S interface is connected to an ePTA
0x0E - 0xFF	Reserved and ignored on reception	

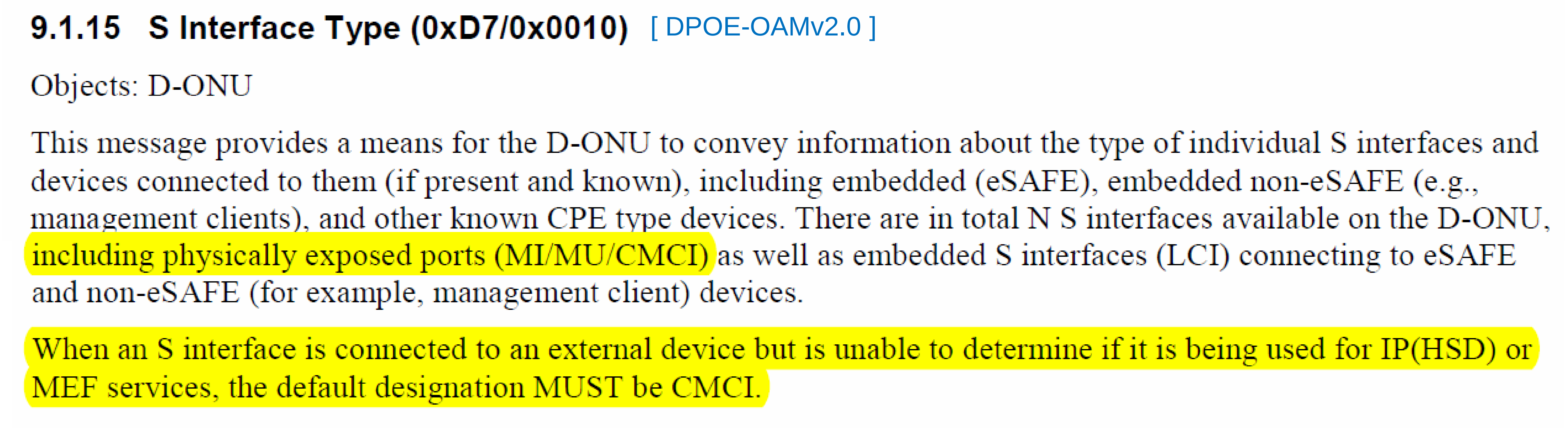
S Interface types in DPOE-OAMv2.0

Missing ports types are highlighted

eDOCSIS also includes ePS (embedded Portal Service) Should we add it?

MU and MI are problematic (see next slide)

MU/MI Port Types



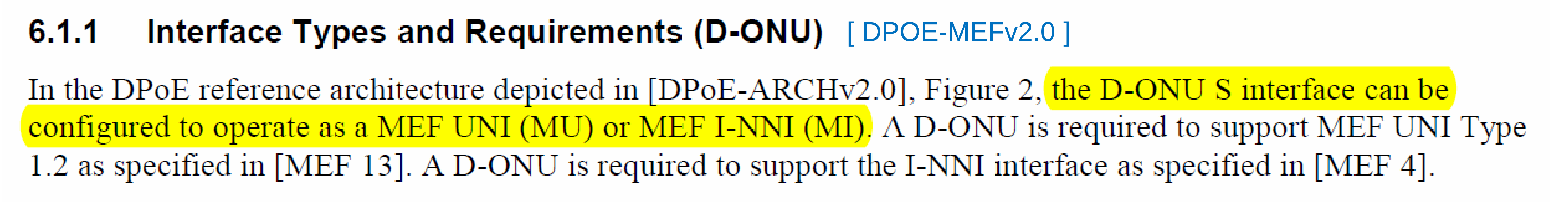
9.1.15 S Interface Type (0xD7/0x0010)

[DPOE-OAMv2.0]

Objects: D-ONU

This message provides a means for the D-ONU to convey information about the type of individual S interfaces and devices connected to them (if present and known), including embedded (eSAFE), embedded non-eSAFE (e.g., management clients), and other known CPE type devices. There are in total N S interfaces available on the D-ONU, including physically exposed ports (MI/MU/CMCI) as well as embedded S interfaces (LCI) connecting to eSAFE and non-eSAFE (for example, management client) devices.

When an S interface is connected to an external device but is unable to determine if it is being used for IP(HSD) or MEF services, the default designation MUST be CMCI.



6.1.1 Interface Types and Requirements (D-ONU)

[DPOE-MEFv2.0]

In the DPoE reference architecture depicted in [DPoE-ARCHv2.0], Figure 2, the D-ONU S interface can be configured to operate as a MEF UNI (MU) or MEF I-NNI (MI). A D-ONU is required to support MEF UNI Type 1.2 as specified in [MEF 13]. A D-ONU is required to support the I-NNI interface as specified in [MEF 4].

“Capability” vs “Configuration”

- The service port type represents immutable hardware implementation.
  - The Service Port Type Capability attribute describes how a port is wired (design-time choice), not how it has been configured at run time.
    - For example, if a service port is connected to eDVA, it is always connected to eDVA.
  - This is how ONU can report its capability even before any ports have been provisioned – port type values are hardcoded based on ONU design.
- But MI and MU according to DPoE MEF spec are just different configurations.
  - By changing classification rules that apply specific VLAN/tunneling mode to a service port, one can dynamically change the port type from MI to MU and vice versa.
  - How can ONU report its port type capability if it can change at runtime?
- **Proposal:**
  - CMCI, MI, and MU should all fall under the UNI Port type category.
  - The run time configuration of a UNI port is determined by (a) querying the rules provisioned for this UNI port and (b) analyzing these rules with respect to what VLAN/tunneling mode they represent.

Proposed Port Types

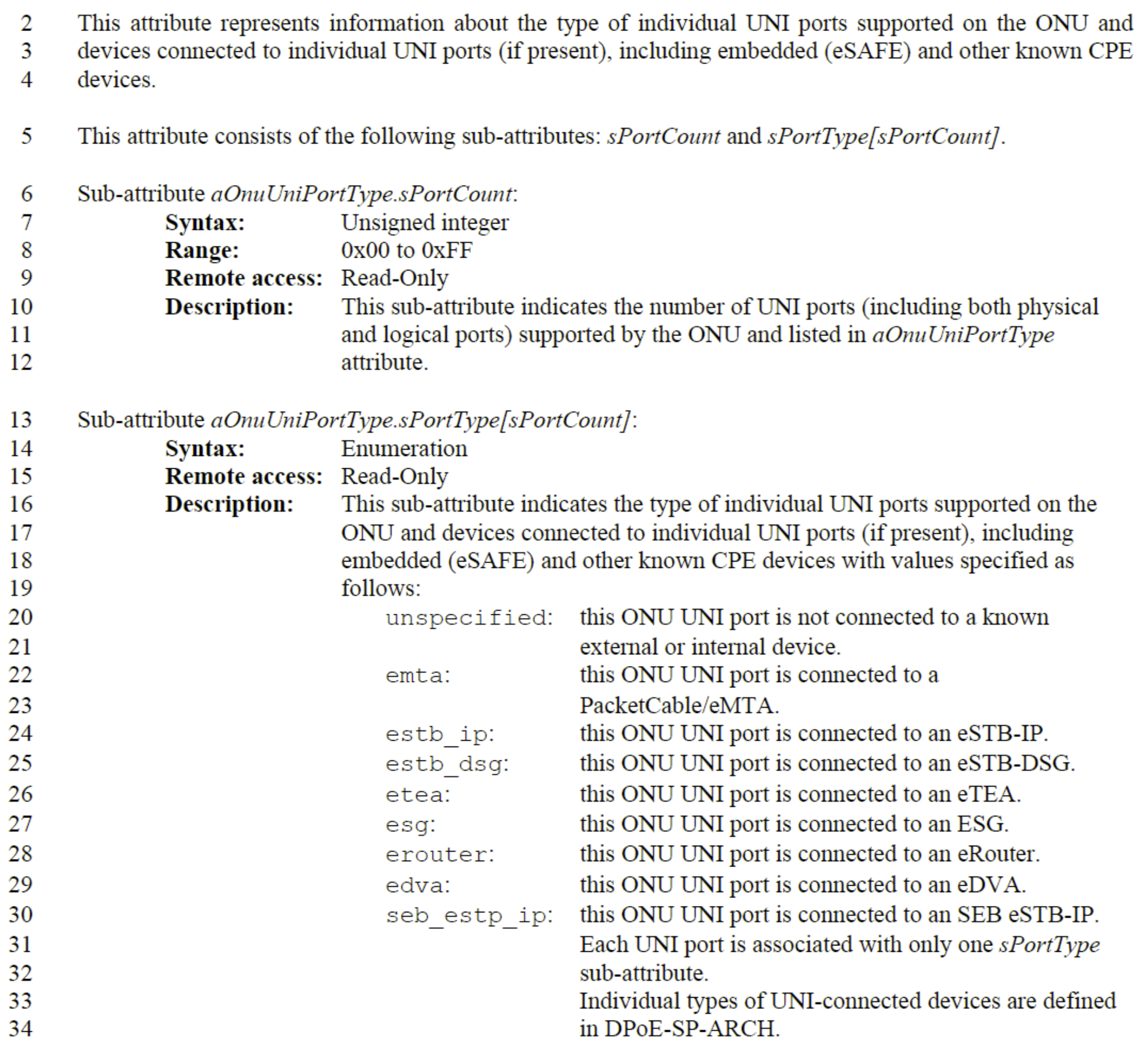
Value	Enumeration	Description
0x00	`unspecified`	service port is not connected to a known external or internal device
0x01	`emta`	service port is connected to an embedded PacketCable Multimedia Terminal Adapter (eMTA)
0x02	`estb_ip`	service port is connected to an IP interface of an embedded Set-Top Box (eSTB-IP)
0x03	`estb_dsg`	service port is connected to an embedded Set-Top Box compliant with DOCSIS Set-Top Gateway specification (eSTB-DSG)
0x04	`etea`	service port is connected to an embedded T1/E1 TDM Emulation Adapter (eTEA)
0x05	`esg`	service port is connected to an embedded Security, Monitoring, and Automation Gateway (eSG)
0x06	`erouter`	service port is connected to an embedded router (eRouter)
0x07	`edva`	service port is connected to an embedded PacketCable 2.0 Digital Voice Adaptor (eDVA).
0x08	`seb_estb_ip`	service port is connected to an embedded Set-Top Box with a Set-Top Extender Bridge (SEB eSTB-IP)
0x09	`uni_port`	service port is connected to an external UNI port. This port type may be equivalent to CMCI, MN, or MI port types defined in [DPoE-ARCHv2.0]
0x0C	`other_internal`	service port is connected to non-eSAFE device and not exposed externally as a subscriber UNI
0x0D	`epta`	service port is connected to an embedded Performance Test Agent (ePTA)
0x0E	`eps`	service port is connected to an embedded CableHome Portal Services Logical Element (ePS)

Type needed?

Description correct?

Thank you

Attribute *aOnuUniPortType* (0xDB/0x00-09)



This attribute represents information about the type of individual UNI ports supported on the ONU and devices connected to individual UNI ports (if present), including embedded (eSAFE) and other known CPE devices.

This attribute consists of the following sub-attributes: *sPortCount* and *sPortType[sPortCount]*.

Sub-attribute *aOnuUniPortType.sPortCount*:

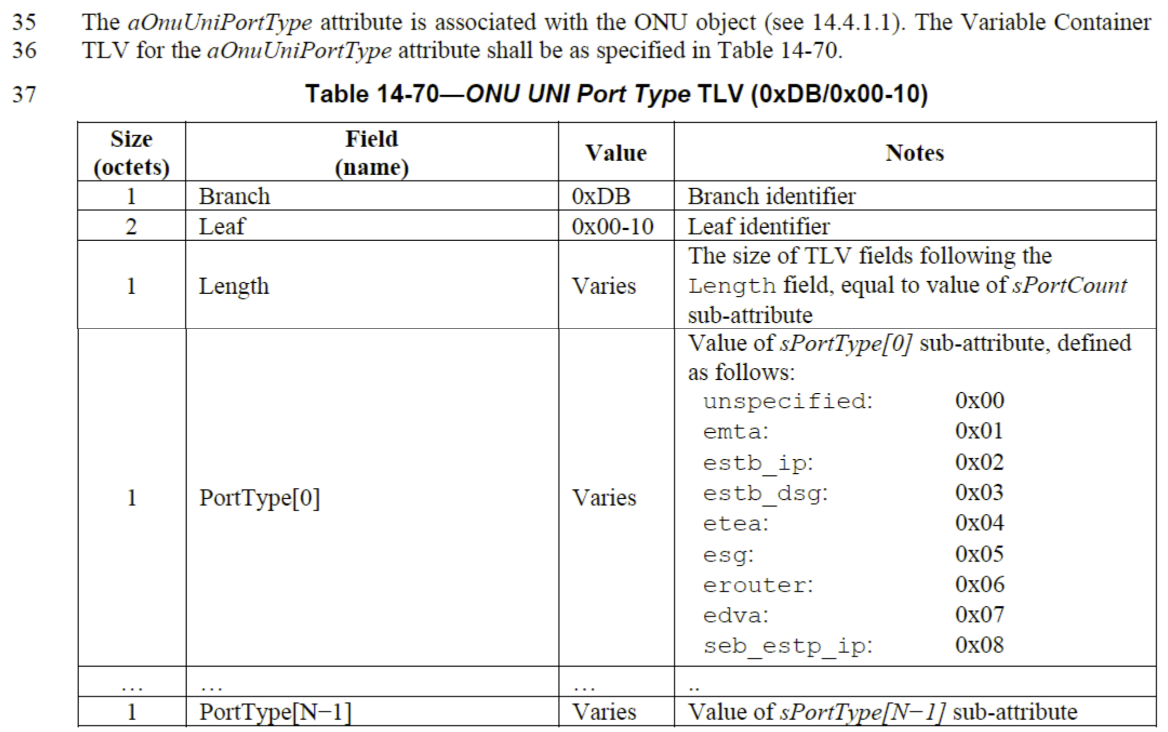
Syntax:	Unsigned integer
Range:	0x00 to 0xFF
Remote access:	Read-Only
Description:	This sub-attribute indicates the number of UNI ports (including both physical and logical ports) supported by the ONU and listed in *aOnuUniPortType* attribute.

Sub-attribute *aOnuUniPortType.sPortType[sPortCount]*:

Syntax: Enumeration
Remote access: Read-Only
Description: This sub-attribute indicates the type of individual UNI ports supported on the ONU and devices connected to individual UNI ports (if present), including embedded (eSAFE) and other known CPE devices with values specified as follows:

`unspecified:`	this ONU UNI port is not connected to a known external or internal device.
`emta:`	this ONU UNI port is connected to a PacketCable/eMTA.
`estb_ip:`	this ONU UNI port is connected to an eSTB-IP.
`estb_dsg:`	this ONU UNI port is connected to an eSTB-DSG.
`etea:`	this ONU UNI port is connected to an eTEA.
`esg:`	this ONU UNI port is connected to an ESG.
`erouter:`	this ONU UNI port is connected to an eRouter.
`edva:`	this ONU UNI port is connected to an eDVA.
`seb_estp_ip:`	this ONU UNI port is connected to an SEB eSTB-IP.

Each UNI port is associated with only one *sPortType* sub-attribute.
Individual types of UNI-connected devices are defined in DPoE-SP-ARCH.



The *aOnuUniPortType* attribute is associated with the ONU object (see 14.4.1.1). The Variable Container TLV for the *aOnuUniPortType* attribute shall be as specified in Table 14-70.

Table 14-70—*ONU UNI Port Type* TLV (0xDB/0x00-10)

Size (octets)	Field (name)	Value	Notes
1	Branch	0xDB	Branch identifier
2	Leaf	0x00-10	Leaf identifier
1	Length	Varies	The size of TLV fields following the `Length` field, equal to value of *sPortCount* sub-attribute
1	PortType[0]	Varies	Value of *sPortType[0]* sub-attribute, defined as follows: `unspecified:` 0x00 `emta:` 0x01 `estb_ip:` 0x02 `estb_dsg:` 0x03 `etea:` 0x04 `esg:` 0x05 `erouter:` 0x06 `edva:` 0x07 `seb_estp_ip:` 0x08
…	…	…	..
1	PortType[N−1]	Varies	Value of *sPortType[N−1]* sub-attribute

- Port indices 0 through N-1 and the type of the device connected to each port is fixed at manufacturing or at deployment (not configurable).
- Any of these ports can be "added" or "deleted". When port is added, it gets the necessary resources (queues, counters, etc.) to become operational.
- Operational ports do not need to have contiguous indices.