Provisioning of LLIDs, UNI Ports, and Queues

Basic categories of management operations

- There are several basic categories of device management operations:
  - **Query HW/SW capabilities and existing resources**
  - **Resource management**
    - Create an object instance (allocate/reserve the necessary resources)
    - Delete an object instance (release the allocated resources)
    - Query current resource allocation
  - **Operation management**
    - Enable object instance (enable operation)
    - Disable object instance (disable operation, but don’t release the resources)
    - Query object’s current operational state

Management of LLIDs, UNI Ports, and Queues

What should we be able to do in order to manage LLIDs, UNI ports, and Queues?

	LLID	UNI Port	Queue
Read HW capabilities	What is the max number of LLIDs ONU can support? Unidirectional and bidirectional	How many UNI ports (S interfaces) are there in the ONU? What eSAFE devices these ports are connected to?	What is the total memory available in ONU? What is the max number of queues ONU can support?
Create an instance	Provision/configure an LLID by allocating all the necessary resources	Provision/configure an UNI port by allocating all the necessary resources	Allocate a queue
Delete an instance	Delete an LLID and release all resources	Delete an UNI port and release all resources	Deallocate a queue
Query Objects	Returns a list of all existing (created) LLIDs	Returns a list of all existing (created) UNI ports	Returns the number of queues and their sizes
Enable	Enable LLID that was previously disabled.	Enable UNI port that was previously disabled.	Enable queue
Disable	Disable LLID without deleting it.	Disable UNI port without deleting it	Disable queue
Query State	Returns status of a given LLID	Returns status of a given UNI port	Returns status of a given queue

Basic management capabilities

Some management capabilities already exist in D0.3 and can be used as is

	LLID	UNI Port	Queue
Read HW capabilities	*aOnuLlidCount* (0xDB/0x00-07) RO	*aOnuUniPortType* (0xDB/0x00-09) RO	*aOnuInfoPacketBuffer* (0xDB/0x00-0A) RO
Create an instance	*aOnuPortConfig* (0xDB/0x01-14) RW + MPCP/OAM Registration	*aOnuPortConfig* (0xDB/0x01-14) RW	*aQueueConfig* (0xDB/0x01-15) RW
Delete an instance	*aOnuPortConfig* (0xDB/0x01-14) RW + MPCP/OAM Deregistration		
Query Objects	*aOnuPortConfig* (0xDB/0x01-14) RW		
Enable	*acEnableUserTraffic* (0xDD/0x06-01) WO	*acPhyAdminControl* (0x09/0x00-05) WO	N/A (controlled via parent object – LLID or UNI port)
Disable	*acDisableUserTraffic* (0xDD/0x06-02) WO		
Query State	*aLlidForwardState* (0xDB/0x00-0C) RO	*aPhyAdminState* (0x07/0x00-25) RO	

Everything in red is a target of this proposal

What needs to change and why

	LLID	UNI Port	Queue
Create an instance	LLIDs need to be provisioned by OAM, instead of being registered via MPCP	The *aOnuPortConfig* (0xDB/0x01-14) request simply tells ONU to enable N UNI ports. There is no mechanism to select specific ports connected to specific eSAFE devices.	The *aQueueConfig* (0xDB/0x01-15) request allocates queues to LLID and UNI ports after these objects were already created. Behavior is undefined if the parent object gets deleted. Some resources are allocated immediately when LLID or UNI is created (counters, lookup entries, etc.) But queues, which are also resources to LLID or UNI ports, are allocated through a separate step. What happens to parent object if queues cannot be allocated? Creating and deleting of LLIDs and UNIs, including allocation of memory, should be performed as an atomic operation (all or nothing, no partial success)
Delete an instance	LLIDs need to be deleted via OAM, instead of being deregistered via MPCP	Undefined behavior if *aOnuPortConfig* is issued dynamically with a different number of ports to enable. Are all ports deleted and created again? What happens to the queues? No mechanism to delete specific port connected to specific eSAFE device.	
Query Objects	Instead of simply reporting the total number of LLIDs, the ONU needs to report all the assigned LLID values and the type of each LLID	Instead of simply reporting the total number of UNI ports, the ONU needs to report index of each UNI port and the type of each port. The index and type should match the ONU HW capabilities reported by the *aOnuUniPortType* (0xDB/0x00-09) attribute	

Unidirectional vs. Bidirectional LLIDs

Bidirectional LLID

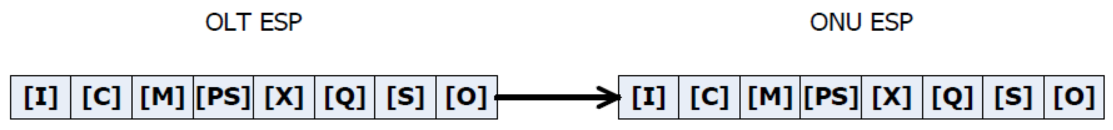
- Carries traffic in both directions
- Is bound to an upstream queue in ONU
- Must be reported and granted

Unidirectional LLID

- Carries only downstream traffic
- Is never reported or granted
- Consumes less resources than a bidirectional LLID
  - No upstream queue
  - Half of the statistic counters (RX only, no TX)
  - No SAR buffers (downstream is not fragmented)
- Multicast connections are created by provisioning the same unidirectional LLID value into multiple ONUs

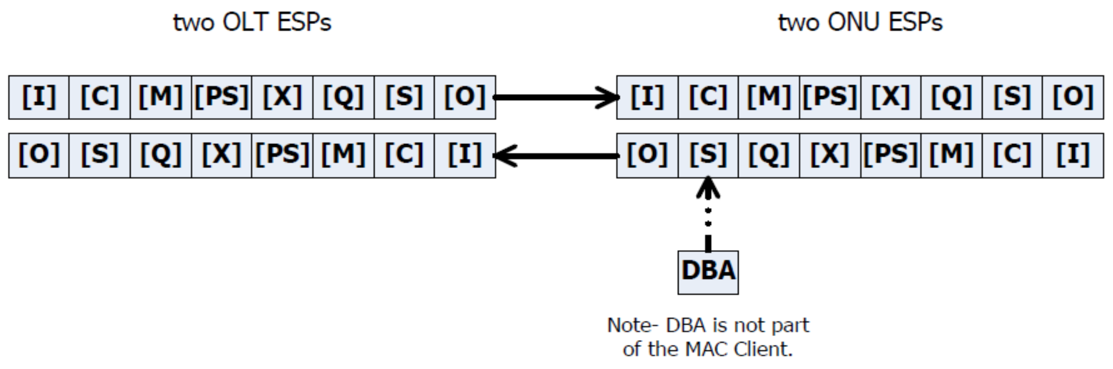
Per EPON Reference Model in 1904.1, the EPON Service Path (ESP) consists of the following logical blocks:

[I] -- Input block	[X] -- CrossConnect block
[C] -- Classifier block	[Q] -- Queues block
[M] -- Modifier block	[S] -- Scheduler block
[PS] -- Policer/Shaper block	[O] -- Output block



OLT ESP
ONU ESP
[I] [C] [M] [PS] [X] [Q] [S] [O]
[I] [C] [M] [PS] [X] [Q] [S] [O]

Figure 6-2—Unidirectional (downstream) unicast connectivity



two OLT ESPs
two ONU ESPs
[I] [C] [M] [PS] [X] [Q] [S] [O]
[O] [S] [Q] [X] [PS] [M] [C] [I]
[I] [C] [M] [PS] [X] [Q] [S] [O]
[O] [S] [Q] [X] [PS] [M] [C] [I]
DBA
Note- DBA is not part of the MAC Client.

Figure 6-4—Bidirectional unicast connectivity

Overview of the proposal

- Queues are referenced via an LLID or UNI port instance (as they are now)
- LLIDs and UNI ports may be added and deleted dynamically, one instance at a time
- There are no separate commands to allocate or free queues.
  - Queues are allocated when an LLID or UNI port instance is added
  - Queues are freed when an LLID or UNI port instance is deleted
- Rules may exist without queues, LLIDs, or UNI ports
  - If a rule has the destination set to a non-existent queue, a drop queue is assumed, i.e., the frame is dropped.
  - A rule may use a non-existent LLID value / UNI index in its Condition Clause. Such rule is valid, but it will never have a match.

LLID Provisioning Basics

- Only one bidirectional PLID and one bidirectional MLID are allowed per ONU
  - These are the primary PLID and MLID assigned during registration
- Multiple downstream-only PLIDs and MLIDs may be added to allow control and management of groups of ONUs (multicast)
  - ONUs always respond on primary PLID or MLID

	Bidirectional LLID	Unidirectional LLID
PLID	• Not allowed to be provisioned • Assigned during registration • RX/TX queues are predetermined	• BCAST_PLID is hardcoded (0x00-01) • Other values may be added/deleted by OAM • Uses the same RX queue as bidirectional PLID • No TX queue
MLID	• Not allowed to be provisioned • Assigned during registration • RX/TX queues are predetermined	• BCAST_MLID is hardcoded (0x00-02) • Other values may be added/deleted by OAM • Uses the same RX queue as bidirectional MLID • No TX queue
ULID	• Added/deleted by OAM • Downstream - forwards to Classifier • Upstream - TX queue is bound via OAM	• Added/deleted by OAM • Downstream - forwards to Classifier • No TX (upstream) queue

→ The only distinction between unidirectional and bidirectional ULID provisioning is the upstream queue binding for the bidirectional LLID

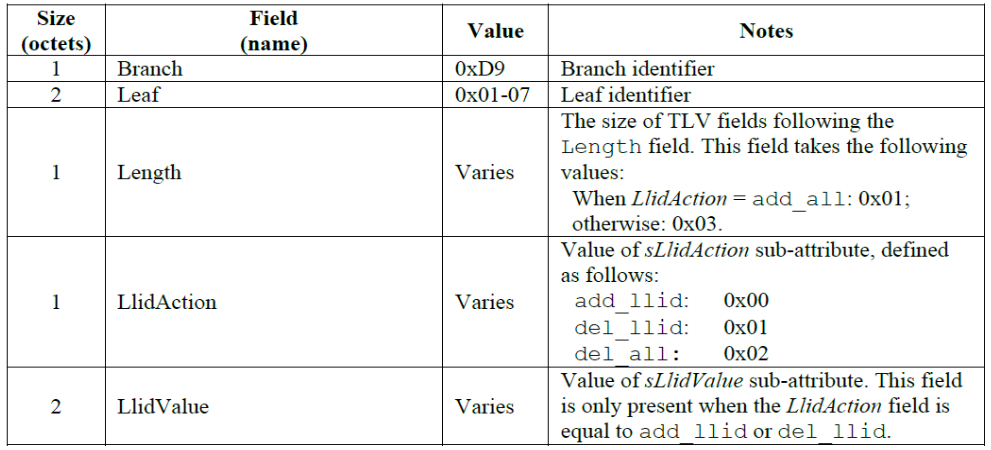
Proposal #1a – TLV for *acConfigLlid* Action

- A single ***acConfigLlid*** action is used to add or delete one LLID, or delete all LLIDs

Field	Size (bytes)	Description
Branch	1	Branch (0xDD)
Leaf	2	Leaf (TBD)
Length	2	TLV Length
Action	1	**0xA1** – `add_llid` **0xD1** – `delete_llid` **0xDA** – `delete_all` (see note on next slide)
LlidValue	2	LLID value in range 0x10-00 to 0xFF-FF (per 802.3ca)
LlidType	1	**0xB0** – Bidirectional ULID **0xD0** – Unidirectional (downstream-only) ULID **0xD1** – Unidirectional (downstream-only) PLID **0xD2** – Unidirectional (downstream-only) MLID
QueueSize	4	Size of the upstream queue in units of 1KB. This field is only present if *LlidType* == 0xB0

- Similar to ***acConfigMulticastLlid*** action, but adds LLID type and upstream queue binding

Table 14-333—*Config Multicast LLID* TLV (0xD9/0x01-07)



Size (octets)	Field (name)	Value	Notes
1	Branch	0xD9	Branch identifier
2	Leaf	0x01-07	Leaf identifier
1	Length	Varies	The size of TLV fields following the `Length` field. This field takes the following values: When *LlidAction* = `add_all`: 0x01; otherwise: 0x03.
1	LlidAction	Varies	Value of *sLlidAction* sub-attribute, defined as follows: `add_llid:` 0x00 `del_llid:` 0x01 `del_all:` 0x02
2	LlidValue	Varies	Value of *sLlidValue* sub-attribute. This field is only present when the *LlidAction* field is equal to `add_llid` or `del_llid`.

LlidValue, *LlidType*, and *QueueSize* fields are only present when needed

Add bidirectional LLID

Branch
Leaf
Length = **8**
Action = 0xA1
LlidValue
LlidType
QueueSize

Add unidirectional LLID

Branch
Leaf
Length = **4**
Action = 0xA1
LlidValue
LlidType

Delete one LLID

Branch
Leaf
Length = **3**
Action = 0xD1
LlidValue

Delete all LLIDs

Branch
Leaf
Length = **1**
Action = 0xDA

LLID provisioning behavior

- Adding a bidirectional ULID also creates/allocates an upstream queue for it. Deleting a bidirectional ULID also destroys/frees the associated upstream queue.
  - Adding or deleting a ULID shall not affect queues/data of other LLIDs

- The `delete_all` request deletes only the LLIDs that were provisioned using the `add_llid` request. It shall not delete the "system" LLIDs:
  1) The primary PLID and MLID assigned during the registration
  2) The pre-configured BCAST_PLID and BCAST_MLID

- The ONU shall respond with the "Bad Parameters" (0x86) code to
  1) `add_llid` request containing an LLID value that already exists in this ONU
  2) `delete_llid` request containing an LLID value that does not exist in this ONU
  3) `delete_llid` request with an LLID value equal to primary PLID, primary MLID, BCAST_PLID, or BCAST_MLID

- The ONU shall respond with the "Insufficient Resources" (0x87) code to
  1) `add_llid` request if the maximum number of LLIDs already has been provisioned
  2) `add_llid` request with the value of *QueueSize* exceeding the ONU's remaining unallocated memory

Proposal #1b – TLV for *aLlidInfo* Attribute

- A single ***aLlidInfo*** attribute is used to query one or all LLIDs (depending on the Object context).

Query Request

- If context = ONU, then query all LLIDs
- If context = LLID, then query this LLID only

Variable Descriptor TLV

Field	Size (bytes)	Description
Branch	1	Branch (0xDB)
Leaf	2	Leaf (TBD)

Query Response (Variable Container TLV)

Field	Size (bytes)	Description
Branch	1	Branch (0xDB)
Leaf	2	Leaf (TBD)
Length	2	Length = 3*N*
LlidValue[i]	2	LLID value
LlidType[i]	1	**0xB0** – Bidirectional ULID **0xB1** – Bidirectional PLID **0xB2** – Bidirectional MLID **0xD0** – Unidirectional (downstream-only) ULID **0xD1** – Unidirectional (downstream-only) PLID **0xD2** – Unidirectional (downstream-only) MLID

×*N*

LLID Querying details

- Even if no LLIDs were provisioned by OAM, the Query Response would contain 4 entries for system LLIDs →
- One TLV may report up to 42 LLIDs. To report more LLIDs, multiple TLVs are used.
- Order of LLIDs in Query Response TLV is implementation-dependent

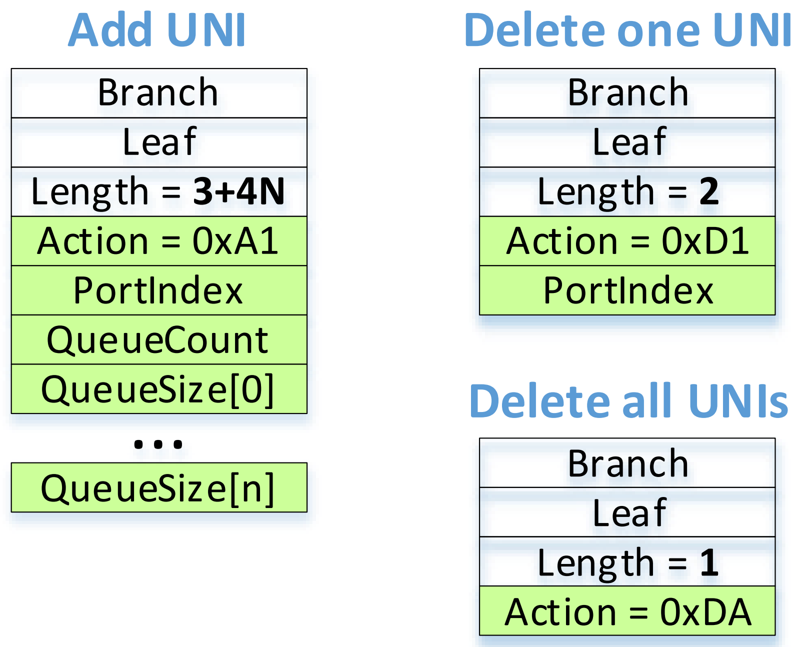
Query Response

Field	Size	Value	Description
Branch	1	0xDB	Branch
Leaf	2	TBD	Leaf
Length	2	12	Length
LlidValue[0]	2	0x00-01	BCAST_PLID value
LlidType[0]	1	0xD1	Code point for unidirectional PLID
LlidValue[1]	2	0x00-02	BCAST_MLID value
LlidType[1]	1	0xD2	Code point for unidirectional MLID
LlidValue[2]	2	??	Primary PLID value
LlidType[2]	1	0xB1	Code point for bidirectional PLID
LlidValue[3]	2	??	Primary MLID value
LlidType[3]	1	0xB2	Code point for bidirectional MLID

Proposal #2a – TLV for *acConfigUniPort* Action

- A single ***acConfigUniPort*** action is used to add or delete one UNI port, or delete all UNI ports
- ***UniIndex***, ***QueueCount***, *and* ***QueueSize[]*** fields are only present when needed

Field	Size (bytes)	Description
Branch	1	Branch (0xDD)
Leaf	2	Leaf (TBD)
Length	2	TLV Length = 3 + 4N
Action	1	**0xA1** – `add_uni` **0xD1** – `delete_uni` **0xDA** – `delete_all`
PortIndex	1	UNI Port index shall be one of the available indices reported by *aOnuUniPortType* (0xDB/0x00-09) attribute
QueueCount	1	Number of queues associated with the given UNI
QueueSize[n]	4 x N	Sizes of queues associated with the given UNI. The value is in units of 1KB.



UNI provisioning behavior

- Adding a UNI port also creates/allocates downstream queue(s) for it.
  Deleting a UNI port also destroys/frees the associated downstream queue(s).
  - Adding or deleting a UNI port shall not affect queues/data of other ports

- UNI ports may be added with non-consecutive indexes. Deleting a UNI port does not cause re-indexing of existing ports

- The ONU shall respond with the “Bad Parameters” (0x86) code to
  1) `add_uni` request containing a UNI port index exceeding the maximum index for this ONU (as reported by *aOnuUniPortType* (0xDB/0x00-09) attribute)
  2) `add_uni` request containing a UNI port index that was already added to this ONU
  3) `delete_uni` request containing an UNI port index that was not previously added to this ONU

- The ONU shall respond with the “Insufficient Resources” (0x87) code to
  1) `add_uni` request if the maximum number of UNI ports already has been allocated
  2) `add_uni` request with the sum of *QueueSize* values exceeding the remaining unallocated memory

Proposal #2b – TLV for *aUniPortInfo* Attribute

- A single ***aUniPortInfo*** attribute is used to query one or all UNI Ports (depending on the Object context).

Query Request

- If context = ONU, then query all UNI Ports
- If context = UNI Port, then query this UNI Port only

Variable Descriptor TLV

Field	Size (bytes)	Description
Branch	1	Branch (0xDB)
Leaf	2	Leaf (TBD)

Query Response (Variable Container TLV)

	Field	Size (bytes)	Description
	Branch	1	Branch (0xDB)
	Leaf	2	Leaf (TBD)
	Length	2	Length = 2*N*
×*N*	***PortIndex[i]***	1	Index of the UNI Port. This index matches the port index reported in *aOnuUniPortType* (0xDB/0x00-09) for the same UNI port instance.
×*N*	***PortType[i]***	1	Port type is determined by the type of the embedded/external device connected to it (see the definition of *aOnuUniPortType*): `unspecified`: port is not connected to a known device `emta`: port is connected to a PacketCable/eMTA `estb_ip`: port is connected to an eSTB-IP `estb_dsg`: port is connected to an eSTB-DSG. `etea`: port is connected to an eTEA `esg`: port is connected to an ESG `erouter`: port is connected to an eRouter `edva`: port is connected to an eDVA `seb_estp_ip`: port is connected to an SEB eSTB-IP

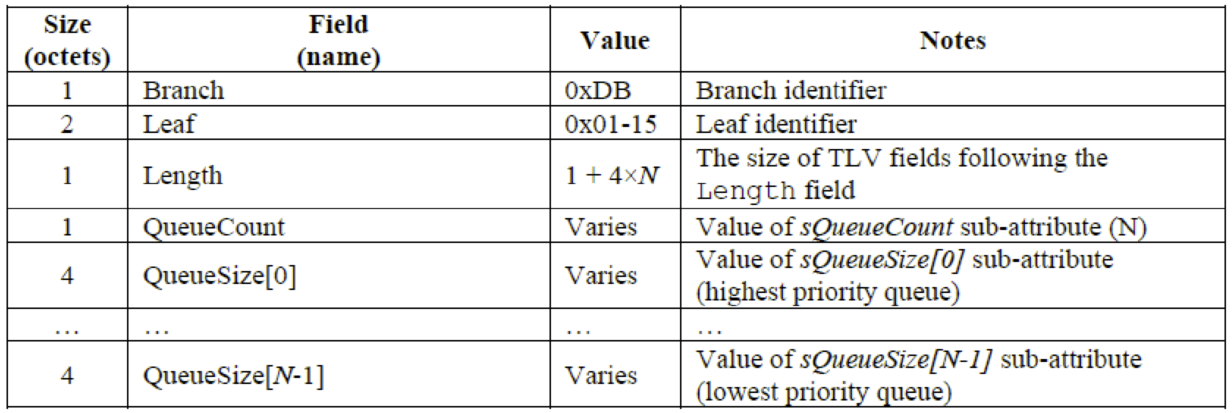
Proposal #3 – TLV for *aQueueInfo* Attribute

- A single ***aQueueInfo*** attribute is used to query the number and sizes of queues allocated to an LLID or a UNI port (depending on the Object context).

Field	Size (bytes)	Description
Branch	1	Branch (0xDB)
Leaf	2	Leaf (TBD)
Length	2	Length = 1+4*N*
QueueCount	1	This field represents the number of queues associated with the given LLID or UNI port object
QueueSize[0]	4	Size of the queue with index 0 (highest priority queue)
	…	
QueueSize[N-1]	4	Size of the queue with index N-1 (lowest priority queue)

- Similar to 'Read' part of the ***aQueueConfig*** attribute

Table 14-94—*Queue Configuration* TLV (0xDB/0x01-15)



Size (octets)	Field (name)	Value	Notes
1	Branch	0xDB	Branch identifier
2	Leaf	0x01-15	Leaf identifier
1	Length	$1 + 4 \times N$	The size of TLV fields following the `Length` field
1	QueueCount	Varies	Value of *sQueueCount* sub-attribute (N)
4	QueueSize[0]	Varies	Value of *sQueueSize[0]* sub-attribute (highest priority queue)
…	…	…	…
4	QueueSize[*N*-1]	Varies	Value of *sQueueSize[N-1]* sub-attribute (lowest priority queue)

- This TLV is valid under the LLID or UNI Port object contexts
  - If the object context is a bidirectional LLID, the ONU shall return the *QueueCount* value of 1 and a single *QueueSize* field
  - If the object context is a unidirectional (downstream-only) LLID, the ONU shall return the *QueueCount* value of 0 and no *QueueSize* fields

Summary of management capabilities

- Management of LLIDs and UNI Ports is done in a consistent manner
  - All device capabilities, resource allocation, and operational mode queries use read-only attributes
  - All changes in resource allocations or in operational modes are done via write-only actions

	LLID	**UNI Port**	**Queue**
Read HW capabilities	*aOnuLlidCount* (0xDB/0x00-07) **RO** attribute	*aOnuUniPortType* (0xDB/0x00-09) **RO** attribute	*aOnuInfoPacketBuffer* (0xDB/0x00-0A) **RO** attribute
Create an instance	*acConfigLlid* (Proposal #1a) **WO** action	*acConfigUniPort* (Proposal #2a) **WO** action	N/A. Queues are allocated when LLID or UNI port instance is created
Delete an instance			N/A. Queues are deallocated when LLID or UNI port instance is deleted
Query Objects	*aLlidInfo* (Proposal #1b) **RO** attribute	*aUniPortInfo* (Proposal #2b) **RO** attribute	*aQueueInfo* (Proposal #3) **RO** attribute
Enable	*acEnableUserTraffic* (0xDD/0x06-01) **WO** action	*acPhyAdminControl* (0x09/0x00-05) **WO** action	N/A. Queues are always enabled
Disable	*acDisableUserTraffic* (0xDD/0x06-02) **WO** action		N/A. Queues are never disabled
Query State	*aLlidForwardState* (0xDB/0x00-0C) **RO** attribute	*aPhyAdminState* (0x07/0x00-25) **RO** attribute	N/A. Nothing to query

Attribute/Action change summary

- **Affected management attributes**
  - ~~*aOnuUniPortCount* (0xDB/0x00-09)~~ – delete, redundant with *aOnuUniPortType* (action item #22)
  - ~~*aOnuPortConfig* (0xDB/0x01-14)~~ – delete, superseded by new actions (action item #25)
  - ~~*aQueueConfig* (0xDB/0x01-15)~~ – delete, superseded by new actions (action item #26)
  - ~~*aOnuMulticastLlid* (0xDB/0x01-10)~~ – delete, superseded by *aLlidInfo* (action item #24)
  - *aLlidInfo* (0xDB/TBD) - add new attribute to query provisioned LLIDs (action item #3)
  - *aUniPortInfo* (0xDB/TBD) - add new attribute to query provisioned UNI ports
  - *aQueueInfo* (0xDB/TBD) – add a new attribute to query queue sizes (action item #26)

- **Affected management actions**
  - ~~*acConfigMulticastLlid* (0xDD/0x01-07)~~ - delete, superseded by *acConfigLlid* (action item #5)
  - *acConfigLlid* (0xDD/TBD) – add new action to configure LLID (action item #3)
  - *acConfigUniPort* (0xDD/TBD) – add new action to configure UNI

Consistent management approach

	Element	Query	Provisioning
Device Capabilities	**LLID**	*aOnuLlidCount* (0xDB/0x00-07) - **RO** attribute	n/a
	UNI Port	*aOnuUniPortType* (0xDB/0x00-09) - **RO** attribute	n/a
Resource Allocation	**LLID**	*aOnuPortConfig* (0xDB/0x01-14) - **RW** attribute	*aOnuPortConfig* (0xDB/0x01-14) - **RW** attribute
	UNI Port	*aOnuPortConfig* (0xDB/0x01-14) - **RW** attribute	*aOnuPortConfig* (0xDB/0x01-14) - **RW** attribute
Operational Status	**LLID**	*aLlidForwardState* (0xDB/0x00-0C) - **RO** attribute	*acEnableUserTraffic* (0xDD/0x06-01) - **WO** action *acDisableUserTraffic* (0xDD/0x06-02) – **WO** action
	UNI Port	*aPhyAdminState* (0x07/0x00-25) - **RO** attribute	*acPhyAdminControl* (0x09/0x00-05) - **WO** action

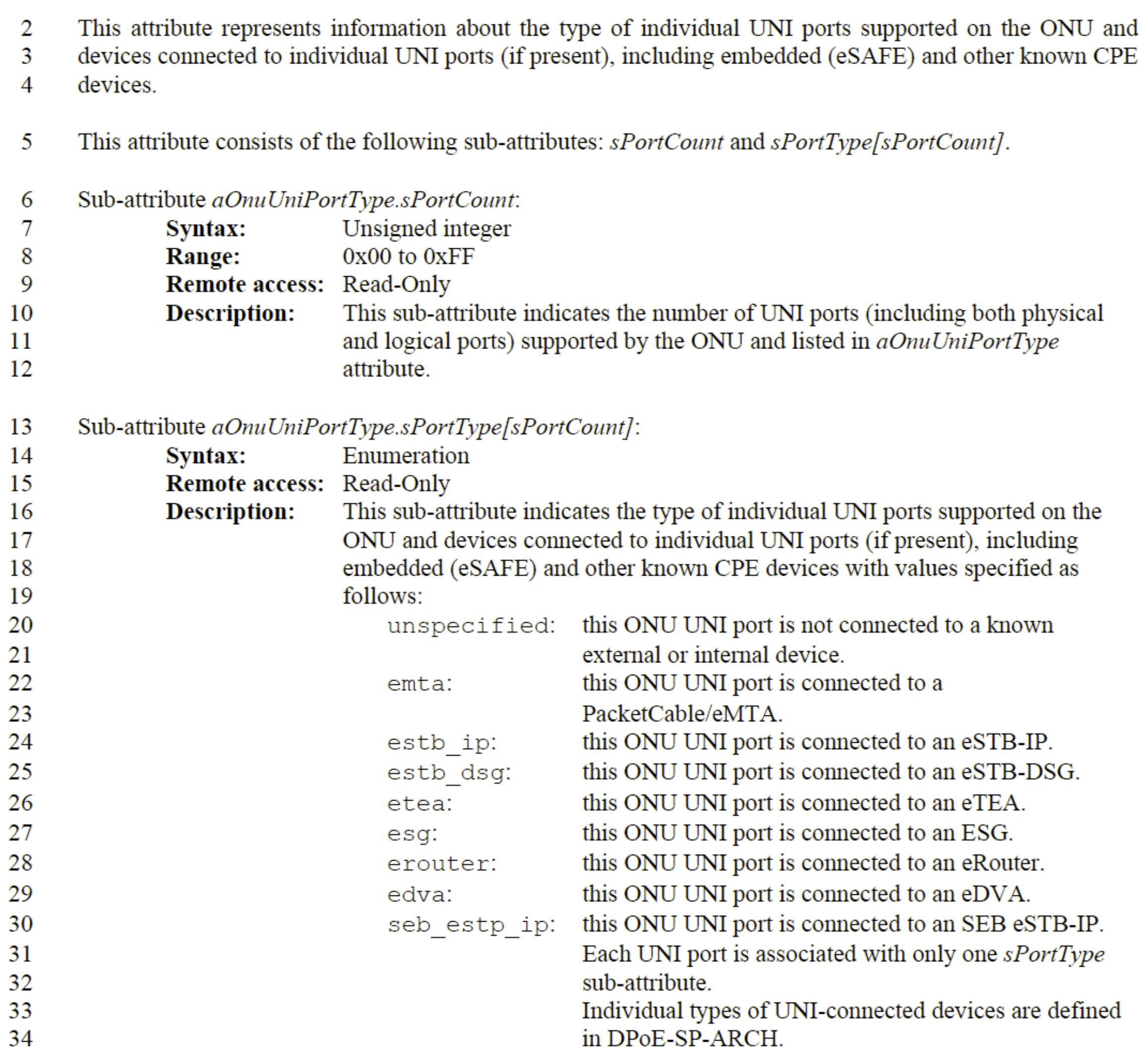
All device capabilities, resource allocation, and operational mode queries use read-only attributes

All changes in resource allocations and in operational modes are done via write-only actions

	Element	Query	Provisioning
Device Capabilities	**LLID**	*aOnuLlidCount* (0xDB/0x00-07) - **RO** attribute	n/a
	UNI Port	*aOnuUniPortType* (0xDB/0x00-09) - **RO** attribute	n/a
Resource Allocation	**LLID**	*aLlidInfo* (Proposal #1b) - **RO** attribute	*acConfigLlid* (Proposal #1a) - **WO** action
	UNI Port	*aUniPortInfo* (Proposal #2b) - **RO** attribute	*acConfigUniPort* (Proposal #2a) - **WO** action
Operational Status	**LLID**	*aLlidForwardState* (0xDB/0x00-0C) - **RO** attribute	*acEnableUserTraffic* (0xDD/0x06-01) - **WO** action *acDisableUserTraffic* (0xDD/0x06-02) – **WO** action
	UNI Port	*aPhyAdminState* (0x07/0x00-25) - **RO** attribute	*acPhyAdminControl* (0x09/0x00-05) - **WO** action

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*:

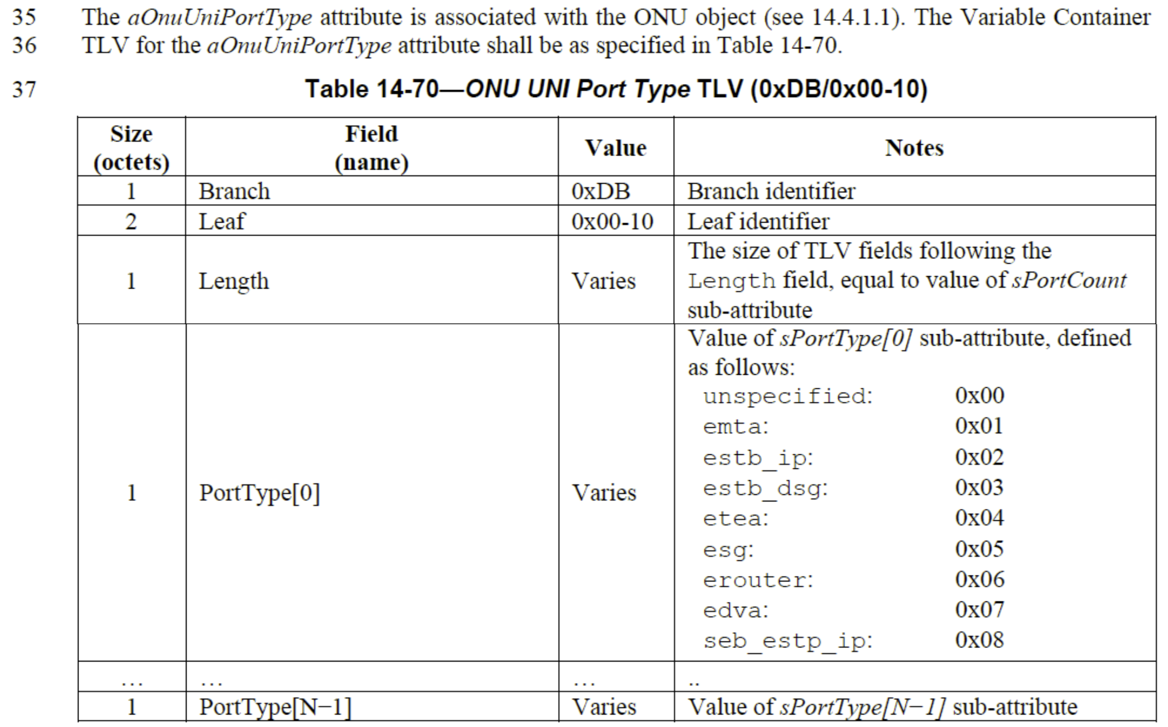
Syntax:	Unsigned integer
Range:	0x00 to 0xFF
Remote access:	Read-Only
Description:	This sub-attribute indicates the number of UNI ports (including both physical and logical ports) supported by the ONU and listed in *aOnuUniPortType* attribute.

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.