13 Extended OAM for Nx25G-EPON

13.1 Requirements

13.1.1 Functional requirements

The EPON system supports all OAM functions as specified in IEEE Std 802.3, Clause 57, and an eOAM-based management suite providing the following functions and features:

- eOAM device and device capability discovery and notification
- Management of the FEC status and mode for the ONU EPON PHY
- Polling and setting DBA-related parameters, including the format of *REPORT* MPCPDU and associated queue structures
- Polling and setting QoS-related parameters, including flow marking, cataloguing, classification rules, etc.
- Configuration of user ports and their associated management
- Configuration and management of VLANs
- Configuration of multicast flows and associated management parameters
- Performing management actions on the ONU devices and subsystems, including port status control, ONU device status verification, etc.

13.1.2 Frame size requirements

The eOAMPDU supports the maximum frame size with a payload of 1500 octets. The size of OAMPDUs is specified in IEEE Std 802.3, 57A.2.

13.1.3 Frame rate requirements

The maximum frame rate for the sum of the IEEE 802.3 (Clause 57) OAMPDUs and eOAMPDUs per Line ONU is as specified in IEEE Std 802.3, Clause 57 and 57A.2.

13.1.4 Timing requirements

Some eOAMPDUs require a response from the ONU. The ONU shall generate an *eOAM_Get_Response* or *eOAM_Set_Response* eOAMPDU within 1 second of the reception of the corresponding *eOAM_Get_Request* or *eOAM_Set_Request* eOAMPDU from the OLT. This requirement covers all types of management requests issued by the OLT for the specific ONU: reading data from specific variable(s), setting values to specific variable(s), performing indicated action(s). The OLT may discard all ONU responses received after the expiration of the 1-second window without processing.

If an ONU cannot respond to the OLT request before the expiration of 1-second window, the ONU shall generate the ONU Busy alarm (see 13.2.4.2.6). The reception of the ONU Busy alarm at the OLT represents an error condition. The handling of this error is implementation specific. The raise of the ONU Busy alarm is generally not a reason to deregister that ONU.

13.2 eOAMPDU structure

This subclause defines the internal structure of the eOAMPDU frame, i.e., the size and meaning of individual fields, and describes a TLV-oriented approach to packaging data in the eOAMPDU. Two specific types of the TLVs are also specified, namely the Variable Descriptor and the Variable Container.

The eOAMPDU format is derived from the IEEE 802.3 (Clause 57) OAM frame format, through the use of Organization Specific Extension mechanism, as described in detail in IEEE Std 802.3, 57.4.2 (Figure 57-9), and further extended in the following subclauses.

13.2.1 Extended OAM organizationally-unique identifier (OUI)

EPON devices shall implement OAM-based management protocols per IEEE Std 802.3, Clause 57. This standard defines eOAM mechanisms for managing various features defined in this standard. The eOAM mechanisms utilize the *Organization Specific* OAMPDU, as defined in IEEE Std 802.3, 57.4.3.6. OAMPDUs defined are identified by the IEEE Std 1904.4-specific organizationally unique identifier (OUI) value, as defined in Table 13-1. Note that the OUI value is shared with Package A defined in IEEE Std 1904.1.

Table 13-1—OUI values for SIEPON.4 features

OUI	Description	Value
OUI_1904_4	OUI value identifying IEEE Std 1904.4 OAMPDUs	0x58-D0-8F

13.2.2 eOAMPDU frame format

Size of the individual fields in the eOAMPDU and their meanings shall be as shown in Table 13-2 and meet the requirements included in the following description.

Table 13-2—Structure of the eOAMPDU frame

Size (octets)	Field (name)	Value	Notes
6	Destination Address	0x01-80-C2-00-00-02	eOAMPDU header
6	Source Address	Varies	
2	Length/Type	0x88-09 (Slow Protocol)	
1	Subtype	0x03 (OAM)	
2	Flags	Varies	
1	Code	0xFE (Organization Specific extensions)	
3	OUI	OUI_1904_4	
1	Opcode	Varies	
38 to 1492	Data + Pad	Varies	
4	FCS	Varies	

The eOAMPDU comprises the following fields:

a) `Destination Address` (`DA`). The `DA` in the eOAMPDUs is the Slow_Protocols_Multicast address (i.e., 0x01-80-C2-00-00-02). Its use and encoding are specified in IEEE Std 802.3, Annex 57A.

b) `Source Address` (`SA`). The `SA` in eOAMPDUs carries the individual MAC address associated with the port through which the eOAMPDU is transmitted.

c) `Length/Type`. The eOAMPDUs are always Type encoded and carry the Slow_Protocols_Type field value.

d) `Subtype`. The `Subtype` field identifies the specific Slow Protocol being encapsulated.

e) `Flags`. The `Flags` field contains status bits as defined in IEEE Std 802.3, 57.4.2.1.

f) `Code`. The `Code` field identifies the specific type of the OAMPDU. The use and encoding of this field are specified in IEEE Std 802.3, Table 57–4.

g) `OUI`. This field carries the organizationally unique identifier assigned to given organization.

h) `Opcode`. This field carries the value of the operation code, identifying a type of the eOAMPDU ~~specified in this profile~~ (see 13.3.1).

i) `Data/Pad`. This field contains the eOAMPDU data and any necessary padding. The data portion of the `Data/Pad` field carries a number of TLVs used to encode specific operations/values.

The `Pad` field is as defined in IEEE Std 802.3, Clause 3.

j) `FCS`. This field is the frame check sequence, as defined in IEEE Std 802.3, Clause 4.

The fields described in item a) through item g) are jointly referred to as the eOAMPDU header.

13.2.3 TLV-oriented structure

The `Data/Pad` field in the eOAMPDU may carry at least one TLV, used to encode a specific set of values/operations. Individual TLV types for the eOAMPDU, i.e., Variable Descriptor and Variable Container, are defined in the following subclauses.

A series of TLVs carried in any of the *eOAM_Get_Request*, *eOAM_Get_Response*, *eOAM_Set_Request*, or *eOAM_Set_Response* eOAMPDUs shall be terminated with the Variable Descriptor with values carried in the `Branch` and `Leaf` fields equal to 0.

13.2.3.1 Variable Descriptor TLV

eOAM_Get_Request eOAMPDUs (see 13.3.2) use the list of Variable Descriptor TLVs and permit the management system to request the value of one or more managed objects hosted on the ONU, both defined in IEEE Std 802.3, Clause 30~~, and defined by this profile~~.

A Variable Descriptor TLV has the form of a 3-octet 2-tuple, comprising a 1-octet `Branch` code and a 2-octet `Leaf` code (together comprising the `Type` identifier), which unequivocally identifies the target attribute/action.

The structure of a Variable Descriptor shall be as presented in Table 13-3.

Table 13-3—Structure of the Variable Descriptor

Size (octets)	Field (name)		Value range
1	Branch	Type	0x00: End of list of TLVs 0x01 to 0xFF
2	Leaf		0x00-00 to 0xFF-FF

13.2.3.2 Variable Container TLV

eOAM_Get_Response eOAMPDUs (see 13.3.3), *eOAM_Set_Request* eOAMPDUs (see 13.3.4), and *eOAM_Set_Response* eOAMPDUs (see 13.3.5) use the list of Variable Container TLVs and permit

— The ONU to respond to the *eOAM_Get_Request* eOAMPDU; or

— The management system to set the value of one or more target managed attributes hosted on the ONU; or

— The ONU to respond to the *eOAM_Set_Request* eOAMPDU.

A Variable Container TLV has the form of a variable-length 4-tuple, comprising the following fields:

— A 1-octet-wide `Branch` field

— A 2-octet-wide `Leaf` field

— A 1-octet-wide `Length` field

— A variable-length `Value` field, the size of which is defined by the value carried in the `Length` field

The `Branch`/`Leaf` 2-tuple represents the `Type` field and unequivocally identifies the target attribute/action.

The `Length` code identifies the size of the following `Value` field, with additional restrictions:

— When the `Length` field value is in the range of 0x00 to 0x7F, it represents the length of the `Value` field, expressed in units of octets, where the value of 0x00 represents the length of the field equal to 128 octets and length in the range of 1 to 127 octets is mapped directly into the range of 0x01 to 0x7F (see IEEE Std 802.3, 57.6.2.1).

— When the `Length` field value is in the range of 0x80 to 0xFF, the value carried in this field represents the eOAMPDU return code and implies a zero length of the `Value` field (no additional value is carried in that case). The eOAMPDU return codes are defined in 13.4.

The `Value` field is present only if the `Length` field value is in the range of 0x00 to 0x7F and represents the following:

— In the case of *eOAM_Get_Response* eOAMPDUs, the value stored in the requested managed attribute

— In the case of *eOAM_Set_Request* eOAMPDUs, the value to be written into the target managed attribute

The structure of a Variable Container TLV shall be as presented in Table 13-4.

Table 13-4—Structure of the Variable Container

Size (octets)	Field (name)		Value	Comments
1	Branch	Type	0x00 to 0xFF	—
2	Leaf		0x00-00 to 0xFF-FF	—
1	Length		0x00 0x01 to 0x7F 0x80 to 0xFF	`Value` field is 128 octets long. `Value` field is 1 to 127 octets long. `Value` field is zero octets long, `Length` field represents the return code per 13.4.
Varies	Value		Variable	Present only when the value in the `Length` field is greater than zero; format as defined for the branch/leaf code

Variable Containers may contain data of a few common types, as defined below:

— **Integer**: An integer carried in a Variable Container shall be represented in the two's-complement form, with the Most Significant Octet (MSO) first. Note that Variable Containers are of variable length; as a result, attributes that are integers do not have a fixed width. The source OAM client may suppress leading zero octets of integers. The target OAM client shall accept an integer in a Variable Container of any legal width (1–128 octets). If a Variable Container is smaller (shorter) than the representation used by the target OAM client, the value is extended to match the representation used by the target OAM client as necessary. If the Variable Container is larger than the representation used by the target OAM client, the resulting action on the received value is implementation dependent.

— **Enumerated value**: An enumerated value carried in a Variable Container is a set of values with predefined meanings. Enumerated values always have a fixed length, regardless of the number of trailing zeros—effectively, enumerated values always have the maximum possible size anticipated for the particular managed attribute. Examples of valid enumerated values include (in binary format) 0b0000-0001-1000 or 0b0000-0000-0000. The source OAM client shall not suppress trailing zeros for enumerated values, and the target OAM client shall not add trailing zeros to the received enumerated values.

— **Sequence list**: A sequence list carried in a Variable Container has the form of a series (sequence) of values, typically of the enumerated value type. All elements in the sequence list shall be of the same length. The number of elements in the sequence list shall be determined from the size of the given Variable Container (value of the `Length` field).

13.2.3.3 TLVs carrying large values

The maximum length of data that can fit into a single Variable Container is equal to 128 octets. Some attribute values may be larger than the 128 octets, requiring a series of TLVs to transfer them between the source OAM client and the target OAM client, using a repeated branch/leaf tuple for the attribute in question. Such a series of TLVs is terminated with a TLV with the same branch/leaf tuple, and a length of zero, to indicate the end of multi-TLV value.

The value of such a large attribute is segmented into a number of individual TLVs, where each TLV in such a sequence of TLV carries a fragment of the large attribute and has the size meeting the requirements stipulated in 13.2.3.2. As a result, the value of such a large attribute is broken into a number of blocks, each with the size of at most 128 octets, and each such block is then carried in a dedicated TLV. For example, assume that the ONU has 23 MAC addresses stored in the dynamic MAC address table. The OLT requests the current list of MAC addresses learned by the ONU, in response to which the ONU needs to list all such MAC addresses using the respective TLVs. A single TLV can hold at most 21 whole MAC addresses (21 × 6 = 126 octets). In this case the first TLV would carry 21 MAC address, the second TLV would carry the remaining 23−21 = 2 MAC addresses, and this series of TLVs would be followed by a TLV with the same branch/leaf tuple but of zero size to indicate the end of the large value sequence.

13.2.4 TLVs for 802.3 OAMPDUs

13.2.4.1 *Extended Information* TLV

The *Information* OAMPDU defined in IEEE Std 802.3, Clause 57, can contain the Organization Specific Information as *Information* TLV (IEEE Std 802.3, 57.5.2.3). Presence of this *Extended Information* TLV in the *Information* OAMPDU during the OAM discovery process indicates that the OLT or the ONU supports the extended OAM.

The format of the *Extended Information* TLV shall be as specified in Table 13-5 and described in the following text.

Table 13-5—Structure of the *Extended Information* TLV

Size (octets)	Field (name)	Value
1	Type	0xFE (*Organization Specific Information* TLV)
1	Length	0x05
3	OUI	OUI_1904_4
1	InfoType	0x00
1	Version	This field identifies the version of the eOAM ~~used by this profile~~. Bits [7:4] represent the major version number Bits [3:0] represent the minor version number The following values are defined: ~~0x01: reserved for backward compatibility, same as 0x10~~ ~~0x02: pre-DPoE OAM, without Certificate Authority support~~ ~~0x03: pre-DPoE OAM, with Certificate Authority support~~ ~~0x10: OAM compliant with DPoE-SP-OAMv1.0-I04 and previous versions~~ ~~0x11: OAM compliant with DPoE-SP-OAMv1.0-I05 and subsequent versions of DPoE-SP-OAMv1.0~~ ~~0x20: OAM compliant with IEEE Std 1904.1-2013 Package A and DPoE-SP-OAMv2.0-I01 and through DPoE-SP-OAMv2.0-I05~~ ~~0x21: OAM compliant with DPoE-SP-OAMv2.0-I06 and DPoE-SP-OAMv2.0-I07~~ ~~0x22~~0x30: OAM compliant with IEEE Std 1904.~~1~~4-~~2016~~202x, ~~Package A and DPoE-SP-OAMv2.0-I09 and subsequent versions~~ Other values are reserved and ignored on reception.

The following fields comprise the *Extended Information* TLV:

a) `Type`: this field is used to indicate the data type held in the given TLV. In the case of the *Extended Information* TLV, this field carries the value of 0xFE, according to IEEE Std 802.3, Table 57–6, indicating the *Organization Specific Information* TLV.

b) `Length`: this field is used to indicate the length of the TLV, expressed in units of octets.

c) `OUI`: this field is used to identify the organization to which the given *Information* TLV belongs. At least one of the *Organization Specific Information* TLVs exchanged between the ONU and the OLT during the eOAM discovery process shall be of *Extended Information* TLV type, containing the OUI_1904_4.

d) `InfoType`: this field is used to identify the subtype of the *Extended Information* TLV.

e) `Version`: this field is used to indicate the version of the eOAM supported by the given device. The internal format of this field is as follows: aaaa.bbbb (4 bits followed by 4 bits), where “aaaa” represents the major version number, and “bbbb” represents the minor version number. For example, a `Version` field carrying the value of 0b0010.0000 represents a major version 2, and a minor version 0.

13.2.4.2 *Event Notification* TLV

The basic structure of the *Organization Specific Event* TLV shall be as specified in IEEE Std 802.3, 57.5.3.5. Specific fields in the *Organization Specific Event* TLV shall be as shown in Figure 13-1 and specified below.

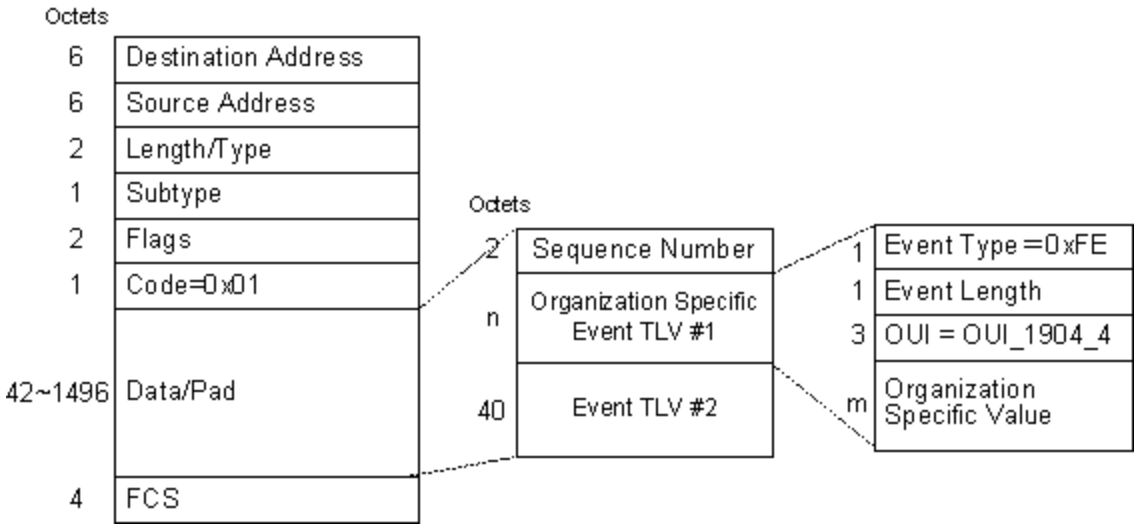

Figure 13-1—Relationship between *Organization Specific Event* TLV and the *Event Notification* OAMPDU

a) `Event Type` = 0xFE, according to the encoding of this field as defined in IEEE Std 802.3, Table 57–12.

b) `Event Length`. This one-octet field indicates the length (in octets) of this TLV-tuple.

c) `OUI` value, equal to OUI_1904_4.

d) `Organization Specific Value` carries the specific set of event-associated information. Further, the structure of the `Organization Specific Value` shall be as specified in Table 13-6 and described below.

Table 13-6—Internal structure of the `Organization Specific Value` field

Octet(s)	Field	Notes
1	EventCode	This field identifies the type of alarm that was identified by the source OAM client. See Table 13-7 for definition of individual values for the `EventCode` field. These alarm codes are grouped into link faults, critical events, and Dying Gasp alarm types, with code values numbered accordingly. Only the values listed in the table are supported. Other values are reserved and ignored on reception.
1	EventRaised	This field indicates whether the given event was raised. The following values are supported: 0x00: The given event was cleared. 0x01: The given event was raised. Other values are reserved and ignored on reception.
2	ObjectType	This field identifies the object element generating the alarm in question.
2 or 4	ObjectInstance	This field identifies the object element instance generating the alarm in question.

— `ObjectType` field identifies the object that generated the given event, as defined in 14.2.1.1. Other values of the `ObjectType` are reserved and ignored on reception.

— `ObjectInstance` field identifies the specific instance of the object that generated the given event, as defined in 14.2.1.2.

Table 13-7—Code points for the `EventCode` field

Event Code	Value	Description
Link Fault Alarms		
LoS	11	Loss of received optical power by the transceiver (ONU EPON port). Link down on Ethernet PHY (ONU UNI port).
Key Exchange Failure	12	ONU did not observe a switch to a new key after key exchange.
Critical Event Alarms		
Port Disabled	21	Ethernet port is disabled by management action.
Dying Gasp Alarms		
Power Failure	41	Loss of power at the ONU (Dying Gasp).
Other Alarms		
Statistics Alarm	81	Statistic has crossed defined alarm thresholds.
ONU Busy	82	ONU is busy and unable to acknowledge or process further OAM until alarm clears.
MAC Table Overflow	83	ONU MAC Table has seen more addresses than it can hold.
PON_IF_Switch	84	PON interface on the ONU was switched to backup.

An ONU may transmit any alarm via any L-ONU, i.e., on any bi-directional LLID registered at that ONU.

13.2.4.2.1 LoS (0x11)

For the PON port, a loss of signal (LoS) condition is detected by lack of incoming optical power or loss of clock and data recovery lock to the downstream bit clock. The transceiver status monitoring for the ONU and the OLT is as specified in 9.1.3. On any of the UNI ports, the LoS condition corresponds to the Link Down condition detected by the UNI port PHY.

13.2.4.2.2 Key Exchange Failure (0x12)

The Key Exchange Failure alarm indicates that a scheduled key exchange has failed. Encryption continues with the previous key for another key exchange interval. Another key exchange is attempted at the next key exchange time.

13.2.4.2.3 Port Disabled (0x21)

The Port Disabled event indicates that one of the ONU ports has been disabled by management action. If the PON port is disabled, then this event notification is not transmitted, and this alarm is visible only locally on the ONU.

13.2.4.2.4 Power Failure (0x41)

A Power Failure alarm indicates that the ONU lost power and is imminently going to be removed from the EPON. An ONU makes every attempt to send this *Event Notification* TLV when it detects loss of power. An ONU may not be able to actually send this *Event Notification* TLV if the required transmission grants are not allocated by the OLT before the ONU runs out of power.

13.2.4.2.5 Statistics Alarm (0x81)

The Statistics Alarm indicates a crossing of predefined thresholds on a specific statistic, as indicated by the *Alarm* TLV, as defined in Table 13-8. Typically, these thresholds would be set for counters for error conditions such as CRC errors.

Table 13-8—*Alarm* TLV structure

Size (octets)	Field (name)	Value
1	Branch	Branch of statistic that crossed threshold
2	Leaf	Leaf of statistic that crossed threshold

13.2.4.2.6 ONU Busy (0x82)

The ONU Busy alarm may be raised by an ONU to inform the OLT that it has been busy for an extended period and may have problems responding to any further OAM requests in the usual timely fashion.

13.2.4.2.7 MAC Table Overflow (0x83)

The MAC Table Overflow alarm is raised by an ONU to inform the OLT that an ingress MAC address has not been learned because the total number of MAC addresses has been exceeded. For example, if the ONU was provisioned to allow four MAC addresses on a particular UNI port, then the first four addresses seen would be learned; the fifth address would cause this alarm to be raised.

13.2.4.2.8 PON_IF_Switch (0x84)

The PON_IF_Switch alarm is raised by the ONU to inform the OLT that the PON interface on the ONU was switched from the active interface to backup interface, according to the tree protection mechanism defined in 9.3.4.

13.2.5 Multipart eOAMPDU response sequence

In some cases, responses to a single eOAMPDU transmitted by the OLT, received from the ONU, may not fit into a single eOAMPDU, especially when reading values from tables with the total size exceeding the payload of an eOAMPDU, e.g., MAC address tables. In this case, the ONU splits its response across multiple eOAMPDUs. The ONU shall inform the OLT that the complete response to the original request was not sent in a single eOAMPDU, but rather in a series of eOAMPDUs. In addition, the OLT shall be able to detect any missing eOAMPDUs in the series of eOAMPDUs comprising a complete response from the ONU.

To indicate that additional eOAMPDUs comprising a complete response from the ONU are forthcoming, the ONU shall add an instance of the *Sequence* TLV (0xDB/0x00-01) to the response eOAMPDU to denote the response sequence. The ONU should not add the *Sequence* TLV (0xDB/0x00-01) to an eOAMPDU not being part of the response sequence.

To send a multiple part response requiring *N* eOAMPDUs, the ONU does the following:

- For the first eOAMPDU in the response sequence:
  - Set the value in the Sequence# field to 0x00.
- For the last eOAMPDU in the response sequence:
  - Set bit 15 in the Sequence# field to 1.
- For all eOAMPDUs in the response sequence:
  - Add the *Sequence* TLV (0xDB/0x00-01) to the eOAMPDU.
  - Increment the value of the Sequence# field.

Figure 13-2 presents examples of various eOAMPDU exchange sequences.

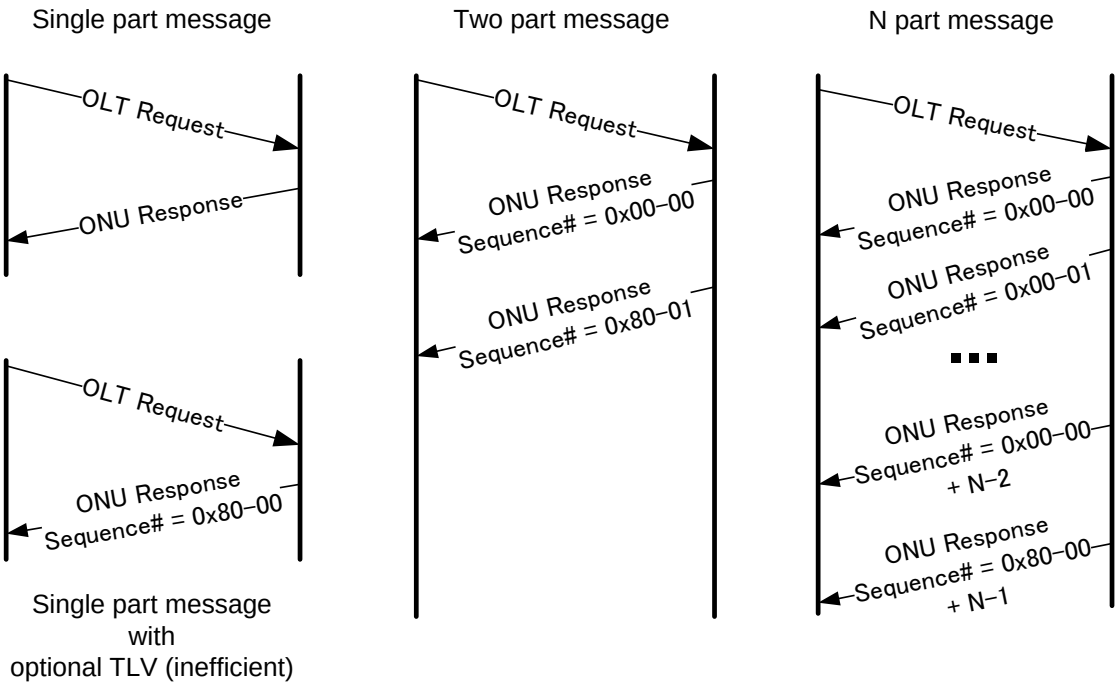

Figure 13-2—Example of various eOAMPDU exchange sequences

13.3 eOAMPDU

Most management functions required for the proper operation of EPON are carried out through the process of reading and writing individual attributes of the managed objects hosted in the ONU. As an example, setting the operating speed for an UNI port requires writing an appropriate value into the speed attribute of the proper port object. Likewise, information can be read from the respective managed objects hosted on the ONU using dedicated eOAMPDUs.

It is also possible to cause the ONU to perform certain actions, e.g., disable specific UNI ports, reset counters, by setting appropriate values in the stored managed objects.

13.3.1 eOAMPDU codes

eOAMPDUs ~~specified for this profile~~ shall be as defined in Table 13-9. These eOAMPDUs use the Organization Specific Extension mechanisms defined in IEEE Std 802.3, Clause 57. Other values are reserved and ignored on reception.

Table 13-9—eOAMPDUs and assignment of Opcode values

Opcode	eOAMPDUs	Defined in
0x00	Reserved, ignored on reception	
0x01	*eOAM_Get_Request*	13.3.2
0x02	*eOAM_Get_Response*	13.3.3
0x03	*eOAM_Set_Request*	13.3.4
0x04	*eOAM_Set_Response*	13.3.5
0x05	Reserved, ignored on reception	
0x06	Reserved, ignored on reception	
0x07	Reserved, ignored on reception	
0x08	*eOAM_KeyExchange*	13.3.7
0x09	*eOAM_Software*	13.3.6
0x0A	Reserved, ignored on reception	

Opcode	eOAMPDUs	Defined in
0x0B	Reserved, ignored on reception	
0x0C	Reserved, ignored on reception	
0xFC	*eOAM_Early_WakeUpOLT*	13.3.8
0xFD	*eOAM_Early_WakeUpONU*	13.3.9
0xFE	*eOAM_Sleep_Allowed*	13.3.10

13.3.2 *eOAM_Get_Request* eOAMPDU

The *eOAM_Get_Request* eOAMPDU permits the management system to request the value of one or more attributes hosted on the ONU, both defined in IEEE Std 802.3, Clause 30~~, and defined by this profile~~. The `Data` field of the *eOAM_Get_Request* eOAMPDU contains a series of Variable Descriptors and *Object Context* TLVs, if needed. The size (and presence) of the `Pad` field depends on the number of individual Variable Descriptors and *Object Context* TLVs. The structure of the Variable Descriptor is defined in 13.2.3.1. The structure of the *Object Context* TLV is defined in 14.2.1.

Functionally, the *eOAM_Get_Request* eOAMPDU is identical to the Variable Request OAMPDU as defined in IEEE Std 802.3, 57.4.3.3.

The structure of the *eOAM_Get_Request* eOAMPDU shall be as specified in Table 13-10 and as described in more detail below.

Table 13-10—Structure of the *eOAM_Get_Request* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0x01
Varies	Data	Varies, a series of Variable Descriptors and *Object Context* TLV
Varies	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Data`, `Pad`, and `FCS` fields are defined in 13.2.2.

13.3.3 *eOAM_Get_Response* eOAMPDU

The *eOAM_Get_Response* eOAMPDU permits the ONU to respond to the *eOAM_Get_Request* eOAMPDU and contains a series of Variable Containers and *Object Context* TLVs, if needed.. The size (and presence) of the `Pad` field depends on the number of individual Variable Containers and *Object Context* TLVs. Each Variable Container may carry the value of the requested variable or a return code (per 13.4) if the variable reading process fails for any reason. The structure of the Variable Container is defined in 13.2.3.2. The structure of the *Object Context* TLV is defined in 14.2.1.

Functionally, the *eOAM_Get_Response* eOAMPDU is identical to the Variable Response OAMPDU as defined in IEEE Std 802.3, 57.4.3.4.

The structure of the *eOAM_Get_Response* eOAMPDU shall be as specified in Table 13-11 and as described in more detail below.

Table 13-11—Structure of the *eOAM_Get_Response* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0x02
Varies	Data	Varies, a series of Variable Containers and *Object Context* TLVs

Size (octets)	Field (name)	Value
Varies	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Data`, `Pad`, and `FCS` fields are defined in 13.2.2.

The size of individual Variable Container(s) ranges between 5 and 132 octets, with the maximum size limited by the `Length` field encoding used in the Variable Container, as defined in 13.2.3.2.

13.3.4 *eOAM_Set_Request* eOAMPDU

The *eOAM_Set_Request* eOAMPDU permits the management system to set the value of one or more attributes hosted on the ONU, both defined in IEEE Std 802.3, Clause 30~~, and defined by this profile~~. The `Data` field of the *eOAM_Set_Request* eOAMPDU contains a series of Variable Containers and *Object Context* TLVs, if needed. The size (and presence) of the `Pad` field depends on the number of individual Variable Containers and *Object Context* TLVs. The structure of the Variable Container is defined in 13.2.3.2. The structure of the *Object Context* TLV is defined in 14.2.1.

The *eOAM_Set_Request* eOAMPDU does not have a functional equivalent in the OAMPDU defined in IEEE Std 802.3, Clause 57. IEEE 802.3 OAM does not support operations related to setting attributes and actions.

The structure of the *eOAM_Set_Request* eOAMPDU shall be as specified in Table 13-12 and as described in more detail below.

Table 13-12—Structure of the *eOAM_Set_Request* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0x03
Varies	Data	Varies, a series of *M* Variable Containers and *Object Context* TLVs
Varies	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Data`, `Pad`, and `FCS` fields are defined in 13.2.2.

Each Variable Container included in the *eOAM_Set_Request* eOAMPDU may map into an attribute (e.g., for branches 0x07 or 0xDB) or an action (e.g., for branches 0x09 or 0xDD), as indicated by the `Branch`/`Leaf` 2-tuple. The `Value` field provides a new value to be assigned to the target attribute or a parameter for the target action.

Actions instruct the target eOAM client to execute a procedure, e.g., rebooting the ONU. The management actions specified in IEEE Std 802.3, Clause 30, are not supported in the IEEE 802.3 (Clause 57) OAMPDUs. The OAM extensions ~~specified for this profile~~ allow the source eOAM client to request execution of both actions defined by IEEE Std 802.3, Clause 30, and actions ~~specified in this profile~~. Some of the actions ~~specified in this profile~~ are expressed using the Variable Container, where the parameters for this action are carried in the body of the Variable Container. Actions that do not have parameters are represented with a Variable Container of zero length (`Length` value of 0x80).

13.3.5 *eOAM_Set_Response* eOAMPDU

The *eOAM_Set_Response* eOAMPDU permits the ONU to respond to management requests (read variable[s], set variable[s], or perform action[s]) and contains a series of Variable Containers and *Object Context* TLVs, if needed. The size (and presence) of the `Pad` field depends on the number of individual Variable Containers and *Object Context* TLVs. Each Variable Container carries a return code (see 13.4) together with the `Branch`/`Leaf` identification of the target attribute/action. The structure of the Variable Container is defined in 13.2.3.2. The structure of the *Object Context* TLV is defined in 14.2.1.

The structure of the *eOAM_Set_Response* eOAMPDU shall be as specified in Table 13-13 and as described in more detail below.

Table 13-13—Structure of the *eOAM_Set_Response* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0x04
Varies	Data	Varies, a series of Variable Containers and *Object Context* TLVs
Varies	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Data`, `Pad`, and `FCS` fields are defined in 13.2.2.

13.3.6 *eOAM_Software* eOAMPDU

This subclause provides the definition of the generic *eOAM_Software* eOAMPDU, together with the specific eOAMPDU subtypes required to implement the software update mechanism ~~specified for this profile~~. The software update mechanism ~~for this profile~~ is specified in 12.3.3.

13.3.6.1 *eOAM_Software* eOAMPDU structure

The *eOAM_Software* eOAMPDU is a specific type of the generic eOAMPDU, as defined in Table 13-8.

The generic structure of the *eOAM_Software* eOAMPDU shall be as presented in Table 13-14 and as described in more detail below.

Table 13-14—Structure of the *eOAM_Software* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x09
1	FileTransferOpcode	Indicates the type of the *eOAM_Software* eOAMPDU, per Table 13-15.
Varies	FileTransferBody	Carries the actual data portion of the *eOAM_Software* eOAMPDU, depending on the value of the `FileTransferOpcode` field.
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

Table 13-15—Values of the `FileTransferOpcode` field

`FileTransferOpcode` value	Value
Reserved	0x00
eOAM_Software_WriteRequest	0x01
eOAM_Software_FileTransferData	0x02
eOAM_Software_FileTransferAck	0x03

a) eOAMPDU header, as defined in 13.2.2.

b) `Opcode`, as defined in 13.2.2. This field carries the value of 0x09 for *eOAM_Software* eOAMPDU.

c) `FileTransferOpcode` indicates the type of the *eOAM_Software* eOAMPDU. Three types of *eOAM_Software* eOAMPDUs are defined ~~for this profile~~:

0x01: *eOAM_Software_WriteRequest* eOAMPDU is used by the OLT to initiate the ONU software image transfer process.

0x02: *eOAM_Software_FileTransferData* eOAMPDU is used by the OLT to transfer a fragment of the given ONU software image between the OLT and the ONU.

0x03: *eOAM_Software_FileTransferAck* eOAMPDU is used by the ONU to provide a return code to the OLT, indicating the current status of the ONU software image transfer process.

Other values are reserved and ignored on reception.

d) `FileTransferBody` carries the actual information related to the given software upgrade process. There are several supported messages types, as specified by the `FileTransferOpcode` field.

Individual *eOAM_Software* eOAMPDUs (*eOAM_Software_WriteRequest* eOAMPDU, *eOAM_Software_FileTransferData* eOAMPDU, and *eOAM_Software_FileTransferAck* eOAMPDU) are further defined in the following subclauses.

The size of this field is variable and depends on the eOAMPDU subtype as indicated in the `Type` field.

e) `Pad`, as defined in 13.2.2. The length of this field is variable and depends on the size of the total size of the `FileTransferOpcode` and `FileTransferBody` fields.

f) `FCS`, as defined in 13.2.2.

13.3.6.2 *eOAM_Software_WriteRequest* eOAMPDU

The *eOAM_Software_WriteRequest* eOAMPDUs is used by the OLT to initiate a file transfer from the OLT to the selected ONU and deliver the name of the ONU software filename to be stored in the *aOnuFwFileName* (0xDB/0x01-0E) attribute. After this eOAMPDU is received, the ONU prepares for the reception of the software image.

The structure of the *eOAM_Software_WriteRequest* eOAMPDU shall be as specified in Table 13-16 and as described in more detail below.

Table 13-16—Structure of the *eOAM_Software_WriteRequest* eOAMPDU

Size (octets)	Field	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x09
1	FileTransferOpcode	0x01

Size (octets)	Field	Value + notes
Varies	FileName	null-terminated ASCII string
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, `Opcode`, `FileTransferOpcode`, `Pad`, and `FCS` fields are defined in 13.3.6.1.

b) `FileTransferOpcode` identifies the *eOAM_Software_WriteRequest* eOAMPDU.

c) `FileName` represents the ONU software filename, to be stored at the ONU in the *aOnuFwFileName* (0xDB/0x01-0E) attribute.

13.3.6.3 *eOAM_Software_FileTransferData* eOAMPDU

The *eOAM_Software_FileTransferData* eOAMPDUs are used to carry individual fragments of the ONU software image file. Each eOAMPDU carries the block number (`BlockNumber` field) and data fragment size indicator (`BlockWidth` field), specifying the number of file data octets to follow.

The structure of the *eOAM_Software_FileTransferData* eOAMPDU shall be as specified in Table 13-17 and as described in more detail below.

Table 13-17—Structure of the *eOAM_Software_FileTransferData* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x09
1	FileTransferOpcode	0x02
2	BlockNumber	This field reflects the sequential number of the current ONU software image fragment carried in this eOAMPDU.
2	BlockWidth	This field reflects the size of the `BlockData` field. Its value is expressed in units of octets. When the value of this field is equal to 0x00-00, this eOAMPDU is used to keep the ONU software image transfer process alive.
Varies	BlockData	This field carries the actual fragment of the ONU software image.
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, `Opcode`, `FileTransferOpcode`, `Pad`, and `FCS` fields are defined in 13.3.6.1.

b) `FileTransferOpcode` identifies the *eOAM_Software_FileTransferData* eOAMPDU.

c) `BlockNumber` contains the sequential number of the current ONU software image fragment carried in the *eOAM_Software_FileTransferData* eOAMPDU.

d) `BlockWidth` represents the size of `BlockData`. Its value is expressed in units of octets. When the *eOAM_Software_FileTransferData* eOAMPDU is used to keep the ONU software image transfer process alive (keep-alive message), the value of this field is equal to 0x00-00.

e) `BlockData` carries the actual fragment of the ONU software image.

13.3.6.4 *eOAM_Software_FileTransferAck* eOAMPDU

The *eOAM_Software_FileTransferAck* eOAMPDU is used by the ONU to indicate the current status of the ONU software image transfer process. Each eOAMPDU of this type carries the block number (`BlockNumber` field) and the response code (`ResponseCode` field). The block number indicates the number of the next ONU software image data block expected by the ONU.

The *eOAM_Software_FileTransferAck* eOAMPDU is also used by the OLT to indicate the end of the ONU software image file. In this case, the *eOAM_Software_FileTransferAck* eOAMPDU carries the `BlockNumber` value of 0x00-00, together with the `ResponseCode` value indicating the end of the transfer process.

The structure of the *eOAM_Software_FileTransferAck* eOAMPDU shall be as specified in Table 13-18 and as described in more detail below.

Table 13-18—Structure of the *eOAM_Software_FileTransferAck* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x09
1	FileTransferOpcode	0x03
2	BlockNumber	This field carries the sequential number of the software image fragment (block) that the ONU expects to receive next.
1	ResponseCode	This field carries the response code generated by the sender entity (either ONU or OLT).
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, `Opcode`, `FileTransferOpcode`, `Pad`, and `FCS` fields are defined in 13.3.6.1.

b) `FileTransferOpcode` identifies the *eOAM_Software_FileTransferData* eOAMPDU.

c) `BlockNumber` contains the number of the software image block, as described in 13.3.6.3.

d) `ResponseCode` carries the response code, as defined in Table 13-19. Only the values specified in Table 13-19 are allowed. Other values are reserved and cause the *eOAM_Software_FileTransferAck* eOAMPDU to be ignored.

Table 13-19—Response Code values carried in `ResponseCode` field

Response Code	Value	Meaning
OK	0x00	No errors.
Undefined	0x01	Unknown error, or one not covered elsewhere.
Not Found	0x02	Read requested file that is not available.
No Access	0x03	Access permissions do not allow the requested read/write.
Full	0x04	Storage is full, and cannot hold the written file.
Illegal Operation	0x05	Cannot perform requested operation in current state.
Unknown ID	0x06	Requested file ID is not supported by this device.
Bad Block	0x07	Block received in error.
Timeout	0x08	No block received before timer expiration.
Busy	0x09	Cannot perform requested action due to other activity.
Incompatible File	0x0A	Received file is incompatible with this device. File incompatibility is determined by the device vendor.

Response Code	Value	Meaning
Corrupted File	0x0B	File was received corrupted and is unusable by this device. File integrity is determined by the device vendor.

13.3.7 *eOAM_KeyExchange* eOAMPDU

The *eOAM_KeyExchange* eOAMPDU is used to implement the key exchange protocol between the OLT and the ONU.

13.3.7.1 *eOAM_KeyExchange* eOAMPDU structure

The *eOAM_KeyExchange* eOAMPDU is a specific type of the generic eOAMPDU, as defined in Table 13-2.

The structure of the *eOAM_KeyExchange* eOAMPDU shall be as presented in Table 13-20 and as described in more detail below.

Table 13-20—Structure of the *eOAM_KeyExchange* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x08
1	KeyExchangeOpcode	Indicates the type of the *eOAM_KeyExchange* eOAMPDU.
Varies	KeyExchangeBody	Carries the actual data portion of the *eOAM_KeyExchange* eOAMPDU, depending on the value of the `Key Exchange Opcode` field.
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, as defined in 13.2.2.

b) `Opcode`, as defined in 13.2.2. This field carries the value of 0x08 for *eOAM_KeyExchange* eOAMPDU.

c) `KeyExchangeOpcode` indicates the type of the *eOAM_KeyExchange* eOAMPDU. Two types of *eOAM_KeyExchange* eOAMPDUs are defined ~~for this profile~~:

   0x00: *eOAM_KeyExchange_Assign* eOAMPDU is used to assign the encryption key.

   0x01: *eOAM_KeyExchange_ACK* eOAMPDU is used to acknowledge the assignment of the encryption key.

   Other values are reserved and ignored on reception.

d) `KeyExchangeBody` carries the actual information related key exchange process. There are several supported messages types, as specified by the `Key Exchange Opcode` field.

   Individual *eOAM_KeyExchange* eOAMPDUs (*eOAM_KeyExchange_Assign* and *eOAM_KeyExchange_ACK* eOAMPDU) are further defined in the following subclauses.

   The size of this field is variable and depends on the eOAMPDU subtype as indicated in the `Type` field.

e) `Pad`, as defined in 13.2.2. The length of this field is variable and depends on the size of the total size of the `KeyExchangeOpcode` and `KeyExchangeBody` fields.

f) `FCS`, as defined in 13.2.2.

13.3.7.2 *eOAM_KeyExchange_Assign* eOAMPDU

The *eOAM_KeyExchange_Assign* eOAMPDU is used to assign the new encryption key to the link peer.

The structure of the *eOAM_KeyExchange_Assign* eOAMPDU shall be as specified in Table 13-21 and as described in more detail below.

Table 13-21—Structure of the *eOAM_KeyExchange_Assign* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x08
1	KeyExchangeOpcode	0x00
2	LLID	This field carries the value of LLID (as in LLID carried in the frame preamble) for L-ONU to which this eOAMPDU applies. The supported range of values is 0x00-00 to 0x7F-FF. Other values are reserved and ignored on reception.
1	KeyNumber	This field indicates the key exchange phase. The supported range of value is 0x00 to 0x01. Other values are reserved and ignored on reception
1	KeyLength	This field indicates the length of the encryption key. The value is expressed in units of octets.
Varies	Key	This field carries the actual encryption key of the length indicates by the `Key Length` field.
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, `Opcode`, `Pad`, and `FCS` fields are defined in 13.2.2.

b) `KeyExchangeOpcode` identifies the *eOAM_KeyExchange_Assign* eOAMPDU.

c) `LLID` indicates the value of L-ONU LLID to which this *eOAM_KeyExchange_Assign* eOAMPDU refers.

d) `KeyNumber` indicates the key exchange phase, indicating to the receiving link peer whether the current or previous key is to be used.

e) `KeyLength` provides information on the length of the actual encryption key, expressed in units of octets.

f) `Key` carries the actual encryption key.

13.3.7.3 *eOAM_KeyExchange_ACK* eOAMPDU

The *eOAM_KeyExchange_ACK* eOAMPDU is used by the link peer to confirm the assignment of the new encryption key.

The structure of the *eOAM_KeyExchange_ACK* eOAMPDU shall be as specified in Table 13-22 and as described in more detail below.

Table 13-22—Structure of the *eOAM_KeyExchange_ACK* eOAMPDU

Size (octets)	Field (name)	Value + notes
21	eOAMPDU header	Varies
1	Opcode	0x08
1	KeyExchangeOpcode	0x01

Size (octets)	Field (name)	Value + notes
2	LLID	This field carries the value of LLID (as in LLID carried in the frame preamble) for L-ONU to which this eOAMPDU applies. The supported range of values is 0x00-00 to 0x7F-FF. Other values are reserved and ignored on reception.
1	KeyNumber	This field indicates the key exchange phase. The supported range of value is 0x00 to 0x01. Other values are reserved and ignored on reception.
Varies	Pad (optional)	0x00-…-00
4	FCS	Varies

a) eOAMPDU header, `Opcode`, `Pad`, and `FCS` fields are defined in 13.2.2.

b) `KeyExchangeOpcode` identifies the *eOAM_KeyExchange_ACK* eOAMPDU.

c) `LLID` indicates the value of L-ONU LLID to which this *eOAM_KeyExchange_ACK* eOAMPDU refers.

d) `KeyNumber` indicates the key exchange phase, indicating to the receiving link peer whether the current or previous key is to be used.

13.3.8 *eOAM_Early_WakeUpOLT* eOAMPDU

The OLT with enabled support for early wake-up function sends the *eOAM_Early_WakeUpOLT* eOAMPDU to request the ONU to leave the sleep state and enter the active state.

The structure of the *eOAM_Early_WakeUpOLT* eOAMPDU shall be as specified in Table 13-23 and as described in more detail below.

Table 13-23—Structure of the *eOAM_Early_WakeUpOLT* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0xFC
38	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Pad`, and `FCS` fields are defined in 13.2.2.

13.3.9 *eOAM_Early_WakeUpONU* eOAMPDU

The ONU sends the *eOAM_Early_WakeUpONU* eOAMPDU to indicate to the OLT that it left the sleep state and entered the active state. This information allows the OLT to enable the downstream queues and resume downstream transmission to this particular ONU.

The structure of the *eOAM_Early_WakeUpONU* eOAMPDU shall be as specified in Table 13-24 and as described in more detail below.

Table 13-24—Structure of the *eOAM_Early_WakeUpONU* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0xFD
38	Pad	0x00-…-00

Size (octets)	Field (name)	Value
4	FCS	Varies

eOAMPDU header, `Opcode`, `Pad`, and `FCS` fields are defined in 13.2.2.

13.3.10 *eOAM_Sleep_Allowed* eOAMPDU

The *eOAM_Sleep_Allowed* eOAMPDU is used by the OLT to request the ONU to enter the specified sleep mode (indicated by the `SleepMode` field) for a specific duration of time (indicated by the `SleepDuration` field).

The structure of the *eOAM_Sleep_Allowed* eOAMPDU shall be as specified in Table 13-25.

Table 13-25—Structure of the *eOAM_Sleep_Allowed* eOAMPDU

Size (octets)	Field (name)	Value
21	eOAMPDU header	Varies
1	Opcode	0xFE
1	SleepMode	Sleep mode requested by the OLT
4	SleepDuration	The duration of the sleep state, expressed in units of time quanta
Varies	Pad	0x00-…-00
4	FCS	Varies

eOAMPDU header, `Opcode`, `Pad`, and `FCS` fields are defined in 13.2.2.

13.4 eOAMPDU return codes

The eOAMPDU generated by the ONU in response to OLT-side query eOAMPDU may carry return codes when the `Length` field value in the Variable Container is in the range of 0x80 to 0xFF. Specific return codes ~~required for compliance with this profile~~ shall be as specified in Table 13-26. Other values are reserved and ignored on reception.

Per the definition of the Variable Container (see 13.2.3.2), when bit 7 in the `Length` field is set, the `Value` field is not present in the Variable Container.

Table 13-26—Return codes

Code	Name	Description
0x80	No Error	The operation was successfully completed.
0x81	Too Long	Length of result exceeded eOAMPDU data field available.
0x86	Bad Parameters	Parameters for the requested action fail error checking.
0x87	No Resources	The device does not currently have the resources (table entries, memory, etc.) to perform the requested action.
0x88	System Busy	The device is not currently in the proper state to perform the requested action.
0xA0	Undetermined Error	Unknown or unlisted attribute error.
0xA1	Unsupported	An attribute requested is not supported on this device.
0xA2	May Be Corrupted	The value of an attribute counter may be invalid due to reset.
0xA3	Hardware Failure	An attribute hardware error prevented the operation from completing.
0xA4	Overflow	The requested attribute experienced overflow error.

NOTE—Specific return codes may be carried in either *eOAM_Set_Response* eOAMPDU or *eOAM_Get_Response* eOAMPDU.

The OLT at its own discretion may send multiple TLVs in a single *eOAM_Set_Request* eOAMPDU or *eOAM_Get_Request* eOAMPDU, covering multiple attributes and/or actions.

The ONU shall provide exactly one TLV with the return code for each attribute/action TLV included in the received *eOAM_Set_Request* eOAMPDU. The ONU shall provide either exactly one TLV with the return code or at least one TLV with the value of the requested attribute for each attribute TLV included in the received *eOAM_Get_Request* eOAMPDU. The number of *eOAM_Set_Response* or *eOAM_Get_Response* eOAMPDUs generated by the ONU depends on the number of response TLVs generated by the ONU in response to attribute/action TLVs in the received *eOAM_Set_Request* or *eOAM_Get_Request* eOAMPDU.

If a TLV in the *eOAM_Set_Request* eOAMPDU requires the accompanying *Object Context* TLV, the return code in the *eOAM_Set_Response* eOAMPDU shall be preceded by the same *Object Context* TLV. If the series of return codes to the given TLVs in the *eOAM_Set_Request* eOAMPDU does not fit into one *eOAM_Set_Response* eOAMPDU, the remaining part of the series of return codes shall be preceded by the appropriate *Object Context* TLV.