13 Extended OAM for Nx25G-EPON

13.1 Introduction

13.2 Requirements

13.3 Device discovery and capability discovery

13.3.1 MPCP/OAM discovery process

Figure 13-1 shows the relationship between the process of registration, initialization, and negotiation in EPON prior to establishing the data plane connectivity. First, the MPCP discovery and registration process is executed, as defined in IEEE Std 802.3ca, 144.3.7. Next, the process of OAM discovery, as defined in IEEE Std 802.3, Clause 57, and eOAM discovery, as defined in the following subclauses, is executed.

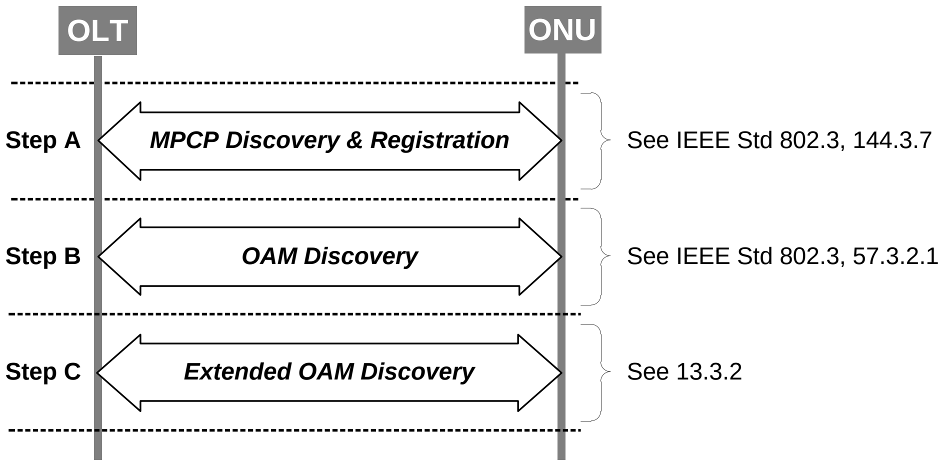
OLT
ONU
Step A
MPCP Discovery & Registration
See IEEE Std 802.3, 144.3.7
Step B
OAM Discovery
See IEEE Std 802.3, 57.3.2.1
Step C
Extended OAM Discovery
See 13.3.2

Figure 13-1—MPCP/OAM discovery process

13.3.2 eOAM discovery process

The eOAM discovery process in the EPON is used to determine whether the given connected ONU supports the specific subtype of the Organization Specific OAM extensions (as identified by the OUI and major and minor versions) in order to verify the capabilities of such an ONU device in terms of the supported OAM functions.

The eOAM discovery process is executed once per ONU. The eOAM discovery process shall be executed on the primary MLID.

13.3.2.1 Requirements

The ONU and OLT shall implement the eOAM discovery process by exchanging the *Organization Specific Information* TLV, as defined in IEEE Std 802.3, 57.5.2.3, and further specified in 13.4.4.1, henceforth referred to as *Extended Information* TLV. The *Extended Information* TLV is embedded in the *Information* OAMPDU, as defined in IEEE Std 802.3, 57.4.3.1.

The OLT starts the eOAM discovery process immediately after the successful completion of the OAM discovery process, as specified in IEEE Std 802.3, 57.3.2.1.

The OLT shall disable all data services for the given ONU until the successful completion of the OAM discovery process (see IEEE Std 802.3, 57.3.2.1), the eOAM discovery process, and the authentication process (see 11.2.2).

The OLT shall deregister any ONU that failed to complete the eOAM discovery process, as defined in 13.3, within five seconds of the time when the OLT sends the first *Extended Information* TLV to this ONU.

13.3.2.2 Ordering of *Organization Specific Information* TLVs

13.3.2.2.1 Source OAM Client requirements

A single IEEE Std 802.3, Clause 57, compliant *Information* OAMPDU may carry more than one *Information* TLV. To simplify both the reception and transmission processes, a specific order of transmission of such TLVs is required. In such a case, the *Local Information* TLV (IEEE Std 802.3, 57.5.2.1) and *Remote Information* TLV (IEEE Std 802.3, 57.5.2.2) shall be transmitted first, followed by the series of *Organization Specific Information* TLVs.

There are no specific transmission order requirements for *Organization Specific Information* TLVs. The *Extended Information* TLV as defined in 13.4.4.1 may be transmitted as the first *Organization Specific Information* TLV, followed by other *Organization Specific Information* TLVs, if present.

13.3.2.2.2 Destination OAM Client requirements

The destination OAM Client shall support the processing of multiple *Information* TLVs in a single *Information* OAMPDU, including *Local Information* TLV, *Remote Information* TLV, and at least one *Organization Specific Information* TLV.

The destination OAM Client shall process all received *Information* TLVs in the order of their reception, discarding any *Information* TLVs that are either malformed or unsupported. A malformed *Information* TLV is considered to have an invalid length and/or unexpected type value. An unsupported *Information* TLV follows the *Information* TLV format requirements, but is marked with an OUI not supported by the given destination OAM Client.

13.3.2.3 Message flow during eOAM discovery process

Figure 13-2 illustrates the message flow during the eOAM discovery process for compliant devices. The eOAM discovery process operates by exchanging the Extended Information TLV between the OLT and the ONU. The OLT and the ONU may send additional Organization Specific Information TLVs if they support other versions of management software identified by other OUI values. The eOAM discovery process comprises two steps described below:

Step 1 — Discovery of OLT and ONU capabilities:

The OLT starts the eOAM discovery process immediately after the completion of the OAM discovery process, by sending the *Information* OAMPDU (`eOAM_Discovery`) with the *Extended Information* TLV, as defined in 13.4.4.1. This message #1 contains the list of all supported versions of the management software associated with the OUI value of the given *Extended Information* TLV.

The OLT may also send additional *Organization Specific Information* TLVs if it supports other versions of management software identified by other OUI values. The order in which individual *Information* TLVs are transmitted within the *Information* OAMPDU is defined in 13.3.2.2.

The ONU responds to the received `eOAM_Discovery` message #1 by sending the `eOAM_Discovery` message #2 with the *Extended Information* TLV containing the list of its supported versions of the management software associated with the OUI of the given *Extended Information* TLV. The ONU may also send additional *Organization Specific Information* TLVs if it supports other versions of management software identified by other OUI values.

Step 2 - Configuration of OLT and ONU eOAM versions:

Once the OLT receives the `eOAM_Discovery` message #2 from the ONU with the list of supported software versions, the OLT decides which version of the management software is to be used. The OLT then notifies the ONU about the selected version using the `eOAM_DiscoveryAck` message #3, carrying the single *eOAMversion* value.

The ONU confirms the selected extended OAM version by sending the `eOAM_DiscoveryAck` message #4 with the same software version.

This concludes the eOAM discovery process, at which time both the ONU and the OLT know what software version to use for further communication across the management channel.

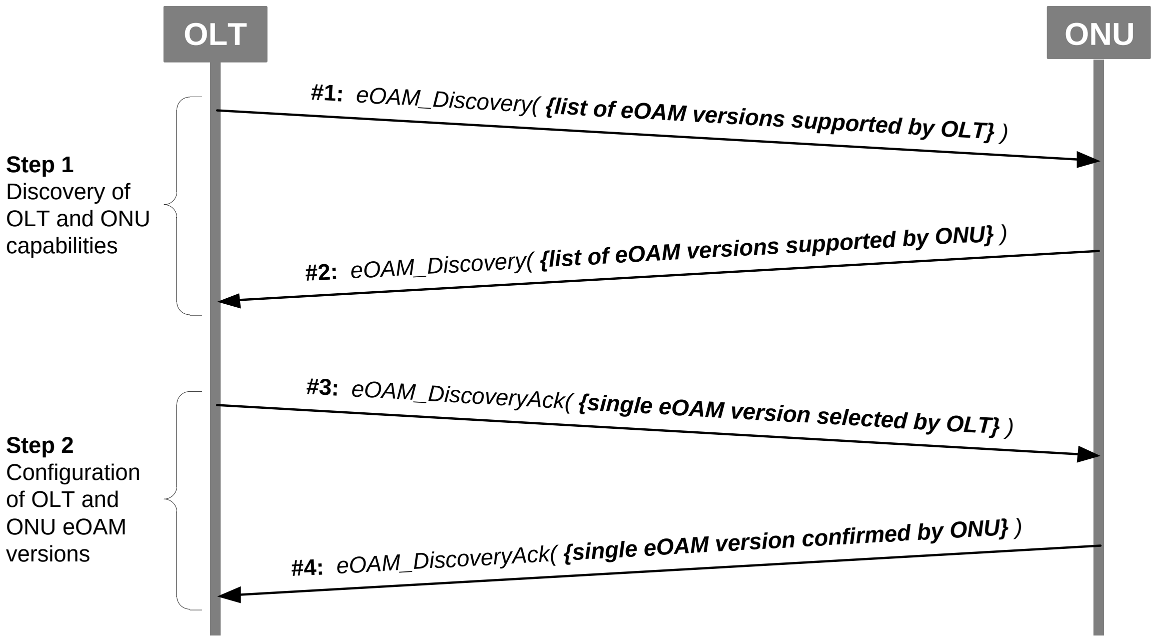
OLT
ONU
#1: eOAM_Discovery({list of eOAM versions supported by OLT})
Step 1
Discovery of OLT and ONU capabilities
#2: eOAM_Discovery({list of eOAM versions supported by ONU})
#3: eOAM_DiscoveryAck({single eOAM version selected by OLT})
Step 2
Configuration of OLT and ONU eOAM versions
#4: eOAM_DiscoveryAck({single eOAM version confirmed by ONU})

Figure 13-2—Illustration of the eOAM discovery process

The step 1 above is necessary when a new ONU has been discovered in the network or the ONU underwent a software/firmware update. In other situations, such as when a previously-discovered ONU is restarted, the step 1 may be unnecessary. If the OLT is aware of the exact set of eOAM versions supported by the ONU, it may omit the step 1 and directly proceed to step 2 to instruct the ONU to use a specific eOAM version.

The following subclauses specify the state diagrams for the eOAM discovery process for the ONU and OLT, including the message flow for both devices, timeout conditions for the OLT, and failure indications.

13.3.2.3.1 Constants

`timeoutOLT`

TYPE: time interval

VALUE: 1 second

This constant identifies the duration of the ONU response timeout period, during which the OLT expects to receive response from the ONU as part of the extended OAM discovery process. If the OLT fails to receive a response from the given ONU within this period of time, it retransmits the previous *Information* OAMPDU carrying the *Extended Information* TLV.

13.3.2.3.2 Variables

The variable type “eOAM version” is represented by a tuple `{MajorVersion, MinorVersion}` as defined in 13.4.4.1.

`commonList`

TYPE: Sequence of eOAM versions

This variable represents the list of extended OAM versions supported by both the OLT and the ONU. When there are no OAM versions supported by the OLT and the ONU simultaneously, this variable is an empty list. No particular ordering of OAM versions in the list is assumed.

`confirmedVersion`

TYPE: eOAM version

This variable represents the supported extended OAM version confirmed by the ONU.

`eOAMDiscoveryComplete`

TYPE: Boolean

This variable indicates whether the extended OAM discovery process has been completed successfully (when set to `true`) or not (when set to `false`). It is set to the default value of `false` upon the ONU initialization.

`retryCount`

TYPE: 8-bit unsigned integer

This variable represents the count of retransmission attempts performed by the OLT.

`selectedVersion`

TYPE: eOAM version

This variable represents the extended OAM version selected by the OLT from the list of versions supported by the ONU.

`versionListOLT`

TYPE: Sequence of eOAM versions

This variable represents the list of extended OAM versions supported by the OLT. The value of this variable is assigned by the OLT manufacturer. No particular ordering of OAM versions in the list is assumed.

`versionListONU`

TYPE: Sequence of eOAM versions

This variable represents the list of extended OAM versions supported by the ONU. In the ONU, the value of this variable is assigned by the manufacturer. In the OLT, the value of this variable is extracted from the received *Extended Information* TLV, as defined in 13.4.4.1. No particular ordering of OAM versions in the list is assumed.

13.3.2.3.3 Timers

`timerTimeout`

This timer measures the response timeout period, during which the OLT awaits the response from the ONU with the specific *Extended Information* TLV.

13.3.2.3.4 Functions

`selectOAMVersion( versionList )`

This function selects and returns a single version of extended OAM from the list `versionList`.

13.3.2.3.5 Primitives

`eOAMI_Discovery`

Acronym for reception of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym is equivalent to the following logical condition:

```
OPI(source_address, flags, code, RxLocalInfoTLV | RxRemoteInfoTLV |
RxExtendedInfoTLV) AND
code                          == 0x00 AND
RxExtendedInfoTLV.Type        == 0xFE AND
RxExtendedInfoTLV.OUI         == OUI_1904_4 AND
RxExtendedInfoTLV.Opcode      == 0x02 AND
RxExtendedInfoTLV.Revision    == 0x01
```

`eOAMI_DiscoveryAck`

Acronym for reception of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym is equivalent to the following logical condition:

```
OPI(source_address, flags, code, RxLocalInfoTLV | RxRemoteInfoTLV |
RxExtendedInfoTLV) AND
code                          == 0x00 AND
RxExtendedInfoTLV.Type        == 0xFE AND
RxExtendedInfoTLV.OUI         == OUI_1904_4 AND
RxExtendedInfoTLV.Opcode      == 0x03 AND
RxExtendedInfoTLV.Revision    == 0x01
```

`eOAMI_RevisionNack`

Acronym for reception of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym replaces the following logical condition:

```
OPI(source_address, flags, code, RxLocalInfoTLV | RxRemoteInfoTLV |
RxExtendedInfoTLV)AND
code                          == 0x00 AND
RxExtendedInfoTLV.Type        == 0xFE AND
RxExtendedInfoTLV.OUI         == OUI_1904_4 AND
RxExtendedInfoTLV.Opcode      == 0x00 AND
RxExtendedInfoTLV.Revision    == 0x01
```

`eOAMI_UnknownRevision`

Acronym for reception of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym replaces the following logical condition:

```
OPI(source_address, flags, code, RxLocalInfoTLV | RxRemoteInfoTLV |
RxExtendedInfoTLV)AND
code                          == 0x00 AND
RxExtendedInfoTLV.Type        == 0xFE AND
RxExtendedInfoTLV.OUI         == OUI_1904_4 AND
(RxExtendedInfoTLV.Opcode     == 0x02 OR
RxExtendedInfoTLV.Opcode      == 0x03) AND
RxExtendedInfoTLV.Revision    != 0x01
```

`eOAMR_Discovery( versionList )`

Acronym for transmission of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym replaces the following sequence of operations:

```
code                          = 0x00
TxExtendedInfoTLV.Type        = 0xFE
TxExtendedInfoTLV.Length      = 7+sizeof(versionList)
TxExtendedInfoTLV.OUI         = OUI_1904_4
TxExtendedInfoTLV.Opcode      = 0x02
TxExtendedInfoTLV.Revision    = 0x01
TxExtendedInfoTLV.versionList = versionList
OPR(source_address, flags, code, TxLocalInfoTLV | TxRemoteInfoTLV |
TxExtendedInfoTLV)
```

The argument `versionList` represents an array (sequence) of one or more tuples `{MajorVersion, MinorVersion}`, representing individual eOAM versions supported by the given device.

`eOAMR_DiscoveryAck( eOAMver )`

Acronym for transmission of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. The argument `eOAMver` is a tuple `{MajorVersion, MinorVersion}` representing eOAM version being assigned by the OLT or being confirmed by the ONU. This acronym replaces the following sequence of operations:

```
code                                = 0x00
TxExtendedInfoTLV.Type              = 0xFE
TxExtendedInfoTLV.Length            = 8
TxExtendedInfoTLV.OUI               = OUI_1904_4
TxExtendedInfoTLV.Opcode            = 0x03
TxExtendedInfoTLV.Revision          = 0x01
TxExtendedInfoTLV.eOAMversion[0]    = eOAMver
OPR(source_address, flags, code, TxLocalInfoTLV | TxRemoteInfoTLV |
TxExtendedInfoTLV)
```

If the ONU failed to select (activate) the eOAM version requested by the OLT, it generates the `eOAMR_DiscoveryAck` message with the `eOAMver` parameter value equal to 0x00, i.e., `{MajorVersion = 0x0, MinorVersion = 0x0}`.

`eOAMR_RevisionNack`

Acronym for transmission of the *Information* OAMPDU carrying the *Extended Information* TLV, as defined in 13.4.4.1. This acronym replaces the following sequence of operations:

```
code                          = 0x00
TxExtendedInfoTLV.Type        = 0xFE
TxExtendedInfoTLV.Length      = 7
TxExtendedInfoTLV.OUI         = OUI_1904_4
TxExtendedInfoTLV.Opcode      = 0x00
TxExtendedInfoTLV.Revision    = 0x01
OPR(source_address, flags, code, TxLocalInfoTLV | TxRemoteInfoTLV |
TxExtendedInfoTLV)
```

`NMSI( message, value )`

This primitive is used to notify the NMS about the result of the extended OAM discovery process. The argument `message` indicates to the NMS whether the eOAM discovery succeeded, and if not, it indicates the type of failure. This parameter could be any of the following:

- `MSG1`: Extended OAM discovery is successful. The argument `value` indicates the eOAM version selected by the OLT and confirmed by the ONU.
- `MSG2`: OLT timed out after three attempts to initiate extended OAM discovery.
- `MSG3`: ONU informed the OLT that it received an unrecognized revision of the *Extended Information* TLV.
- `MSG4`: OLT received an unrecognized revision of the *Extended Information* TLV. There are no common extended OAM versions supported by both OLT and ONU.
- `MSG5`: There are no common extended OAM versions supported by both OLT and ONU.
- `MSG6`: OLT timed out after three attempts requesting the ONU to select a specific extended OAM version.
- `MSG7`: ONU informed the OLT that it rejected the selected extended OAM version or it confirmed the extended OAM version is different from the version assigned to it by the OLT.

Format of the NMSI primitives is outside of scope of this standard.

`NMSR( extDiscovery )`

This primitive is used by the NMS to request the OLT to repeat the extended OAM discovery process.

13.3.2.3.6 State diagrams

The C-OLT shall instantiate the extended OAM discovery process as shown in Figure 13-3 for each newly discovered L-ONU. The C-ONU shall implement the extended OAM discovery process as shown in Figure 13-4.

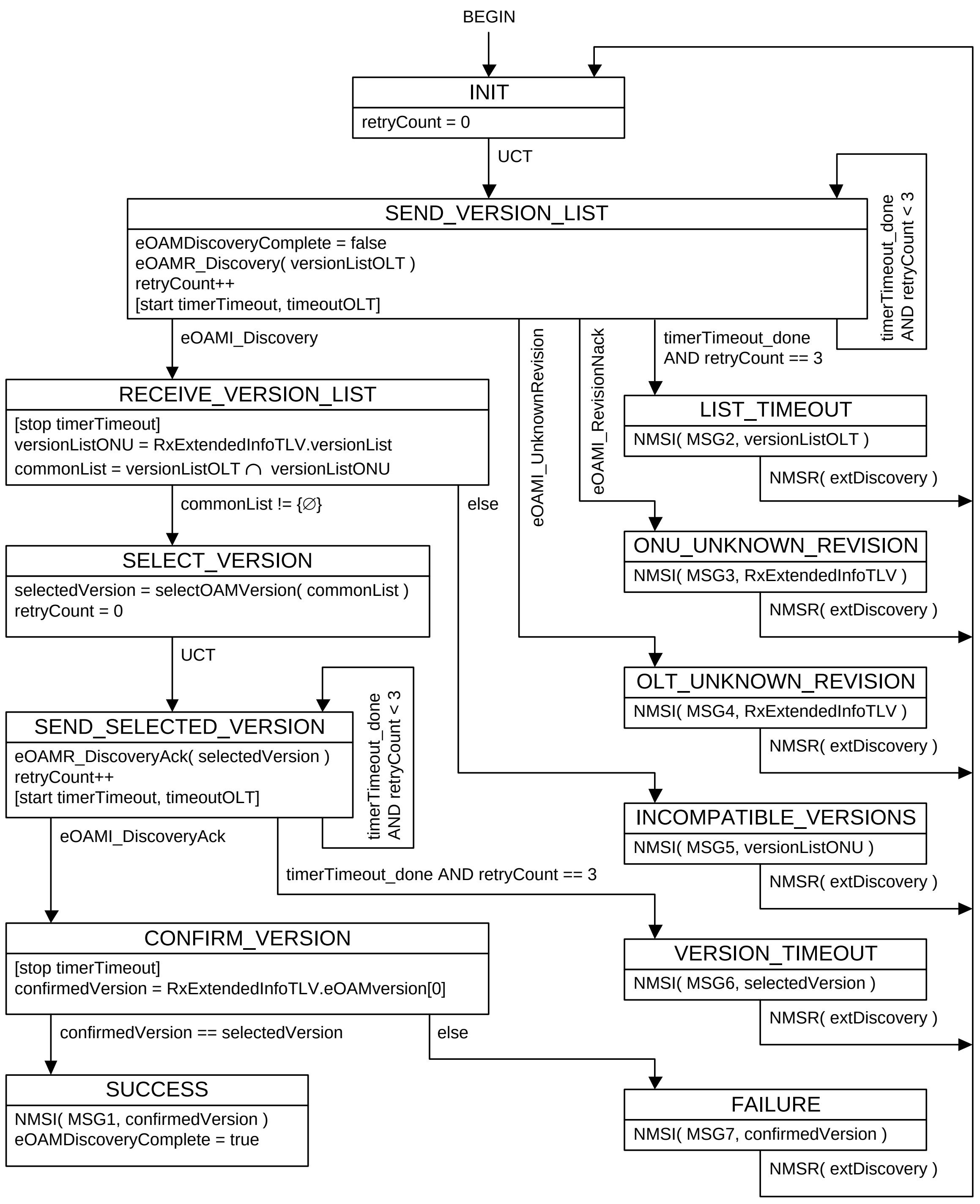
BEGIN
INIT
retryCount = 0
UCT
SEND_VERSION_LIST
eOAMDiscoveryComplete = false
eOAMR_Discovery(versionListOLT)
retryCount++
[start timerTimeout, timeoutOLT]
timerTimeout_done AND retryCount < 3
eOAMI_Discovery
eOAMI_UnknownRevision
eOAMI_RevisionNack
timerTimeout_done AND retryCount == 3
RECEIVE_VERSION_LIST
[stop timerTimeout]
versionListONU = RxExtendedInfoTLV.versionList
commonList = versionListOLT ∩ versionListONU
LIST_TIMEOUT
NMSI(MSG2, versionListOLT)
NMSR(extDiscovery)
commonList != {∅}
else
ONU_UNKNOWN_REVISION
NMSI(MSG3, RxExtendedInfoTLV)
NMSR(extDiscovery)
SELECT_VERSION
selectedVersion = selectOAMVersion(commonList)
retryCount = 0
UCT
OLT_UNKNOWN_REVISION
NMSI(MSG4, RxExtendedInfoTLV)
NMSR(extDiscovery)
SEND_SELECTED_VERSION
eOAMR_DiscoveryAck(selectedVersion)
retryCount++
[start timerTimeout, timeoutOLT]
timerTimeout_done AND retryCount < 3
INCOMPATIBLE_VERSIONS
NMSI(MSG5, versionListONU)
NMSR(extDiscovery)
eOAMI_DiscoveryAck
timerTimeout_done AND retryCount == 3
CONFIRM_VERSION
[stop timerTimeout]
confirmedVersion = RxExtendedInfoTLV.eOAMversion[0]
VERSION_TIMEOUT
NMSI(MSG6, selectedVersion)
NMSR(extDiscovery)
confirmedVersion == selectedVersion
else
SUCCESS
NMSI(MSG1, confirmedVersion)
eOAMDiscoveryComplete = true
FAILURE
NMSI(MSG7, confirmedVersion)
NMSR(extDiscovery)

Figure 13-3—OLT eOAM discovery process state diagram

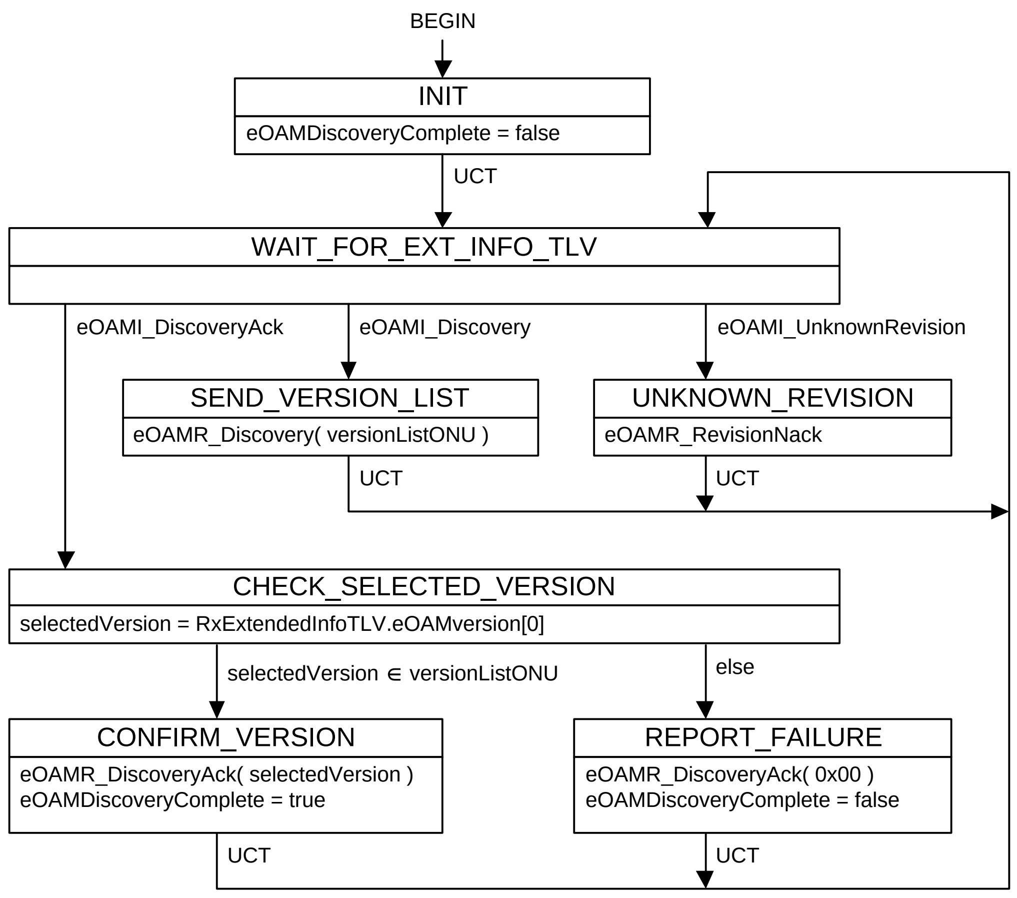
BEGIN
INIT
eOAMDiscoveryComplete = false
UCT
WAIT_FOR_EXT_INFO_TLV
eOAMI_DiscoveryAck
eOAMI_Discovery
eOAMI_UnknownRevision
SEND_VERSION_LIST
eOAMR_Discovery(versionListONU)
UNKNOWN_REVISION
eOAMR_RevisionNack
UCT
UCT
CHECK_SELECTED_VERSION
selectedVersion = RxExtendedInfoTLV.eOAMversion[0]
selectedVersion ∈ versionListONU
else
CONFIRM_VERSION
eOAMR_DiscoveryAck(selectedVersion)
eOAMDiscoveryComplete = true
REPORT_FAILURE
eOAMR_DiscoveryAck(0x00)
eOAMDiscoveryComplete = false
UCT
UCT

Figure 13-4—ONU eOAM discovery process state diagram

13.4 eOAMPDU structure

13.4.1 Extended OAM organizationally-unique identifier (OUI)

13.4.2 eOAMPDU frame format

13.4.3 TLV-oriented structure

13.4.4 TLVs for 802.3 OAMPDUs

13.4.4.1 *Extended Information* TLV

The *Information* OAMPDU may carry an *Organization Specific Information* TLV (see IEEE Std 802.3, 57.5.2.3). The OUI-dependent Value field of the *Organization Specific Information* TLV is further defined in this standard under the OUI `OUI_1904_4` (see Table 13-1). This TLV is refered to as the *Extended Information* TLV. The *Extended Information* TLV carries information used by the eOAM discovery process

The format of the *Extended Information* TLV shall be as specified in Table 13-5, depicted in Figure 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		7 + *N*, where *N* indicates the number of supported extended OAM versions
3		OUI		OUI_1904_4
1		Opcode		0x00: Unknown revision. 0x02: eOAM version discovery: the message contains a list of eOAM versions supported by the transmitting device. 0x03: eOAM version assignment/confirmation.
1		Revision		Revision of the given *Extended Information* TLV
N	1	versionList	eOAMversion[0]	Version of the 1st supported eOAM extension
	1		eOAMversion[1]	Version of the 2nd supported eOAM extension
	…		…	
	1		eOAMversion[N-1]	Version of the *N*th supported eOAM extension

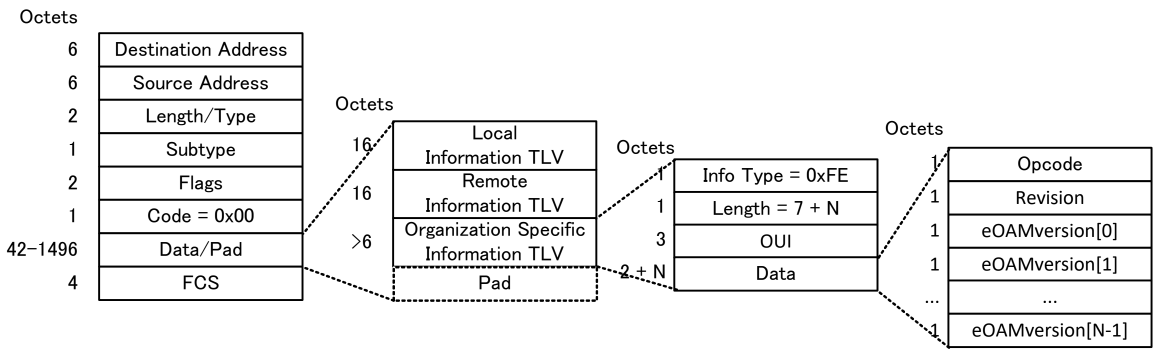

Figure 13-5—Structure of the *Information* OAMPDU with the *Extended Information* TLV

The following fields comprise the *Extended Information* TLV:

a) `Type`: this field represents the type of the given TLV. The *Extended Information* TLV is a specific version of the *Organization Specific Information* TLVs, as indicated by the Type value of 0xFE (see IEEE Std 802.3, Table 57–6).

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

c) `OUI`: this field represents the organizationally unique identifier of the organization-specific TLV. Compliant OLTs and ONUs shall set this value to OUI_1904_4.

d) `Opcode`: this field identifies the type of the message being conveyed by the given *Extended Information* TLV.

e) `Revision`: this field identifies the revision of the *Extended Information* TLV. Compliant OLTs and ONUs shall set this value to 0x01.

f) `versionList`: this field is an array of *N* `eOAMversion[i]` elements representing the eOAM versions supported by the given device. Each array element `eOAMversion[i]` is a tuple

`{MajorVersion, MinorVersion}`, where the `MajorVersion` and the `MinorVersion` are 4-bit integers denoting the major and minor version of the extended OAM respectively. The `MajorVersion` value is mapped into the 4 most-significant bits of `eOAMversion[i]` field, and the `MinorVersion` value is mapped into the 4 least-significant bits of the field. The `versionList` field of the compliant OLT and ONUs shall include the `eOAMversion` values as listed in Table 13-6 and may also include other values.

Table 13-6—Supported values for `eOAMversion` field

MajorVersion_ (bits 7:4)	**MinorVersion** (bits 3:0)	**Description**
0b0011 (3)	0b0000 (0)	eOAM version defined in IEEE Std 1904.4-2025