5 Universal Management Tunnel Protocol Data Units (UMTPDU)

5.1 UMTPDU Structure

5.2 UMTPDU *Subtype* encoding

5.2.1 UMT configuration subtype

5.2.2 OAM subtype

A UMTPDU with OAM subtype (*Subtype* field = 0x03) is an instantiation of a generic UMTPDU, as defined in 5.1, that carries an Operations, Administration, and Maintenance (OAM) payload (see IEEE Std 802.3, 57.4). The frame structure of UMTPDU with OAM subtype is defined in 7.2.1.

5.2.3 OMCI Subtype

<TBD>

5.2.4 L2 Subtype

A UMTPDU with L2 subtype (*Subtype* field = 0x05) is an instantiation of a generic UMTPDU, as defined in 5.1, that carries a complete L2 frame as its payload. The frame structure of UMTPDU with L2 subtype is defined in 7.3.1.

5.2.5 L3 Subtype

A UMTPDU with L3 subtype (*Subtype* field = 0x06) is an instantiation of a generic UMTPDU, as defined in 5.1, that carries an L3 packet as its payload. The frame structure of UMTPDU with L3 subtype is defined in 7.4.1.

5.2.6 Organization-specific extension subtypes

5.3 VLAN-Tagged UMTPDU

6 UMT sublayer

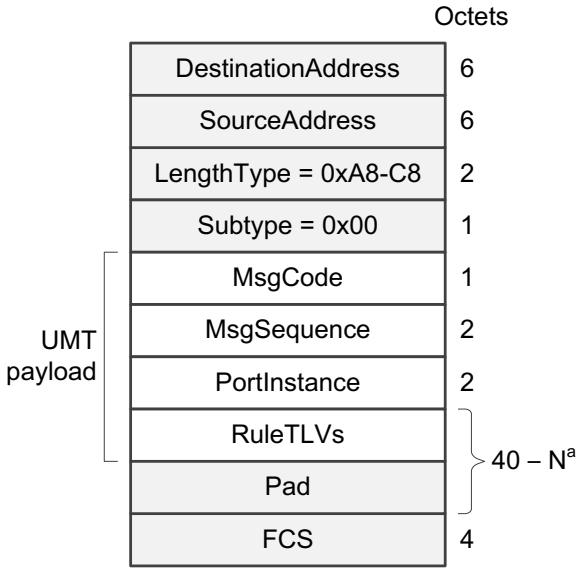
7 Per Protocol Specifications

7.1 Support for UMT Configuration

The tunnels originate and terminate in the UMT-aware devices. The tunnels are configured by means of provisioning specific CTE rules for the tunnel entry and exit points. These rules are provisioned by the operator using the *UMT_CONFIG* UMTPDUs, which carry a set of *condition-encoding* TLVs and a set of *action-encoding* TLVs.

7.1.1 Configuration UMTPDU

The *UMT_CONFIG* UMPTPDU format shall be as depicted in Figure 7-1. The *UMT_CONFIG* UMTPDU is used as both a request to configure a CTE rule as well as a response containing the result of the configuration request.



a – Maximum field length depends on frame type (see Figure 5-1).

Figure 7-1—*UMT_CONFIG* UMTPDU format

The *UMT_CONFIG* UMTPDU is an instantiation of the generic UMTPDU (see **Error! Reference source not found.**). It is identified by the *Subtype* field value of 0x00. The structure of the *UMT payload* is defined as follows:

—*MsgCode*:

The *MsgCode* field identifies whether the UMT_CONFIG message is a request message or a response. If the UMTPDU is a request, this field encodes the requested action. If the UMTPDU is a response, this field echoes the requested action and encodes the result code for this action. The format of the *MsgCode* field is shown in Table 7-1.

Table 7-1—Format of the *MsgCode* field

Bits	Field name	Value	Description
3:0	*MsgType*	0x0	The message is a request
		0x1	The message is a response indicating successful action

		0x2	The message is a response indicating failed action
		0x3	The message is a response indicating that no action was necessary
		0x4	The message is a response indicating invalid request
		0x5 to 0xF	Reserved, ignored on reception
7:4	*RequestCode*	0x0	Query all rules
		0x1	Add a rule
		0x2	Remove a rule
		0x4 to 0xF	Reserved, ignored on reception

—*MsgSequence*:

In situations when a UMT configuration request or a response consists of multiple messages, this field identifies the message sequence number. The field is represented by a decrementing counter, with the last message in a sequence having the *MsgSequence* value of zero. When a request or a response consists of a single UMTPDU, this field has the value of zero.

Note that even when a UMT configuration request or a response consists of multiple messages, a single rule is not split across multiple messages and as such – no reassembly mechanism is necessary to reconstruct any rule. An example scenario where the response consists of multiple messages with decrementing *MsgSequence* values would be a UMT configuration response to a 'Query all rules' request, where multiple rules are being reported.

—*PortInstance*:

This field identifies a port instance in the UMT-aware device to which the given *UMT_CONFIG* UMTPDU applies. The format of the *PortInstance* field is shown in Table 7-2.

Table 7-2—Format of the *PortInstance* field

Bits	Field name	Value	Description
14:0	*PortIndex*	0x00-00 to 0x7F-FF	Index of a port (UMT sublayer) to which the requested action is to be applied.
15	*Direction*	0	The rule is to be applied to the transmit path of UMT sublayer (i.e., an egress rule)
		1	The rule is to be applied to the receive path of UMT sublayer (i.e., an ingress rule)

In the UMT response message, this field reflects the *PortInstance* field value from the corresponding UMT request message.

— *RuleTLVs*:

This field includes one or more CTE rule TLV(s) as defined in 7.1.2. The combined size of the *RuleTLV* and *Pad* fields ranges between 40 and *N*, where *N* is defined in **Error! Reference source not found.**.

7.1.2 CTE rule TLV structure

The structure of a CTE rule TLV is shown in Table 7-3. Each *UMT_CONFIG* UMTPDU shall contain at least one CTE rule TLV.

Table 7-3—CTE rule TLV structure

Field Size (octets)	Field Name	Value	Description
1	*Type*	0xC0	Type code identifying the condition-encoding TLV
		0xAC	Type code identifying the action-encoding TLV
		0x00	Type code indicating that there are no more TLVs to process. The Length field and other fields (if present) are ignored. The TLV with Type = 0x00 shall be the last TLV in every *UMT_CONFIG* UMTPDU and it may be the only TLV in the *UMT_CONFIG* UMTPDU.
1	*Length*	*V*+*M*+4	The *Length* field encompasses the entire TLV, including the *Type* and *Length* fields. A TLV with length of 0x00 through 0x03 is invalid.
1	*Operation*[a]	per Table 6-1	Comparison operator code, if the TLV *Type* = 0xC0
		per Table 6-3	Action code, if the TLV *Type* = 0xAC
V	*FieldCode*[a]	per Table 6-2	Identifies a field to be used in a comparison, or to be modified by an action.
L	*Value*	Various	The value to be used in a comparison or by an Add/Change action. Some TLVs may omit this field.
M[b]	*Mask*	various	The mask pattern to be used in a comparison condition. The mask pattern is applied as a bitwise-AND operation to both the value to be used in a comparison (see the *Value* field above) as well the value of the field identified by the *FieldCode* parameter of this TLV. Some TLVs may omit this field[c]. When *Mask* is omitted, the comparison applies to the entire field.

[a)] Fields *Operation* and *FieldCode* shall be present in all TLVs, even if they are not used. When these fields are not used, they should be set to the value of zero.

[b)] The length *M* of *Mask* field shall be the same as the length of *Value* field, if mask field is present. Otherwise, the length *M* is considered to be equal to zero.

[c)] If a CTE rule TLV omits the *Value* field, the *Mask* field shall also be omitted.

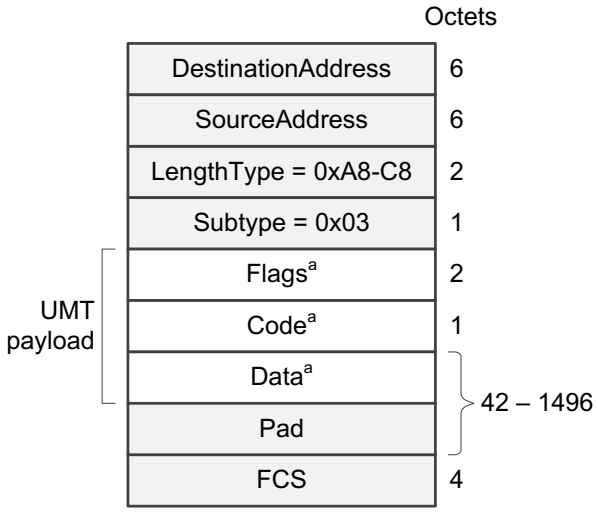
7.1.3 Receive Path Specification

7.1.4 Transmit Path Specification

7.2 Support for OAM

7.2.1 OAM_subtype UMTPDU Encoding

The frame structure of UMTPDU with OAM subtype shall be as depicted in Figure 7-2.



a – This field is defined in IEEE 802.3, 57.4

Figure 7-2—Format of UMTPDU with OAM subtype

The structure of the *UMT payload* in the UMTPDU with OAM subtype is defined as follows:

—*Flags*:
This field carries the value of the *Flags* field as defined in IEEE Std 802.3, 57.4.

—*Code*:
This field carries the value of the *Code* field as defined in IEEE Std 802.3, 57.4.

—*Data*:
This field carries the payload portion of the OAMPDU as defined IEEE Std 802.3, 57.4.

7.2.2 Receive Path Specification

7.2.3 Transmit Path Specification

7.2.4 Support for OAM remote loopback

7.2.4.1 Overview

OAM defined in IEEE Std 802.3, 57.2.11 provides an optional data link layer frame-level loopback mode, which can be used for fault localization and link performance testing.

The OAM entity that initiates the loopback mode is called the *local* OAM entity. The OAM entity on the opposite end of a link is called the *remote* OAM entity. In the OAM remote loopback mode, the local and remote OAM entities operate as follows:

a) The local OAM entity transmits frames from the MAC client and OAMPDUs from the local OAM client or OAM sublayer.
b) Within the OAM sublayer of the remote OAM entity, every received OAMPDU is passed to the OAM client, while non-OAMPDUs, including other Slow Protocol frames, are looped back without altering any field of the frame.
c) Frames received by the local OAM entity are parsed by the OAM sublayer. OAMPDUs are passed to the OAM client and all other frames are discarded.

Both OAM entities continue exchanging OAMPDUs in order to keep the OAM discovery process from restarting and to perform other management tasks.

7.2.4.2 OAM loopback over UMT tunnel

When the OAM loopback is initiated over a UMT tunnel, the behavior of the local and remote OAM entities remains as it is described in 7.2.4.1. Specifically, the remote OAM sublayer loops back all non-OAMPDUs (i.e., generates an *MA_DATA.request()* primitive in response to every *MA_DATA.indication()* primitive that does not contain an OAMPDU). The local OAM sublayer discards all received non-OAMPDU frames.

However, to ensure that the non-OAMPDUs transmitted by the local MAC client are delivered to the remote OAM sublayer, an additional UMT tunnel needs to be established from the local DTE to the remote DTE. Similarly, to deliver the looped-back frames from the remote DTE back to the local DTE, a UMT tunnel operating in the opposite direction also needs to be established.

Since the OAM is a link-level protocol (i.e., operates over a single-span link), either a DTE itself or a bridge immediately adjacent to that DTE must be UMT-aware. A network configuration with both the local and the remote DTE being UMT-unaware is illustrated in Figure 7-3.

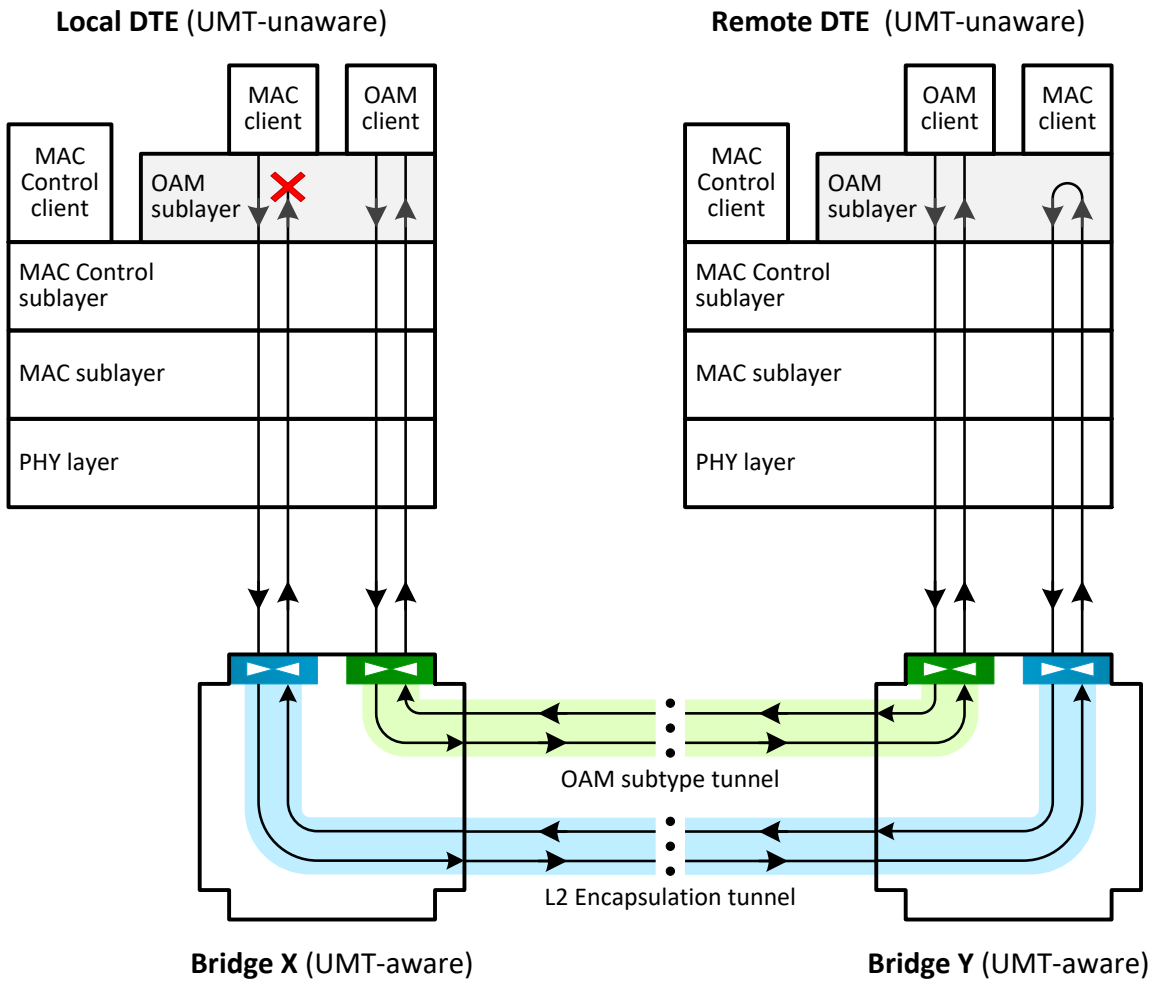


Figure 7-3— Remote OAM loopback over UMT tunnel with UMT-unaware local DTE and UMT-unaware remote DTE.

The remote OAM loopback can also be established when one of the DTEs is UMT-aware and the other is not. Figure 7-4 illustrates a network configuration with the local DTE being UMT-aware and the remote DTE being UMT-unaware.

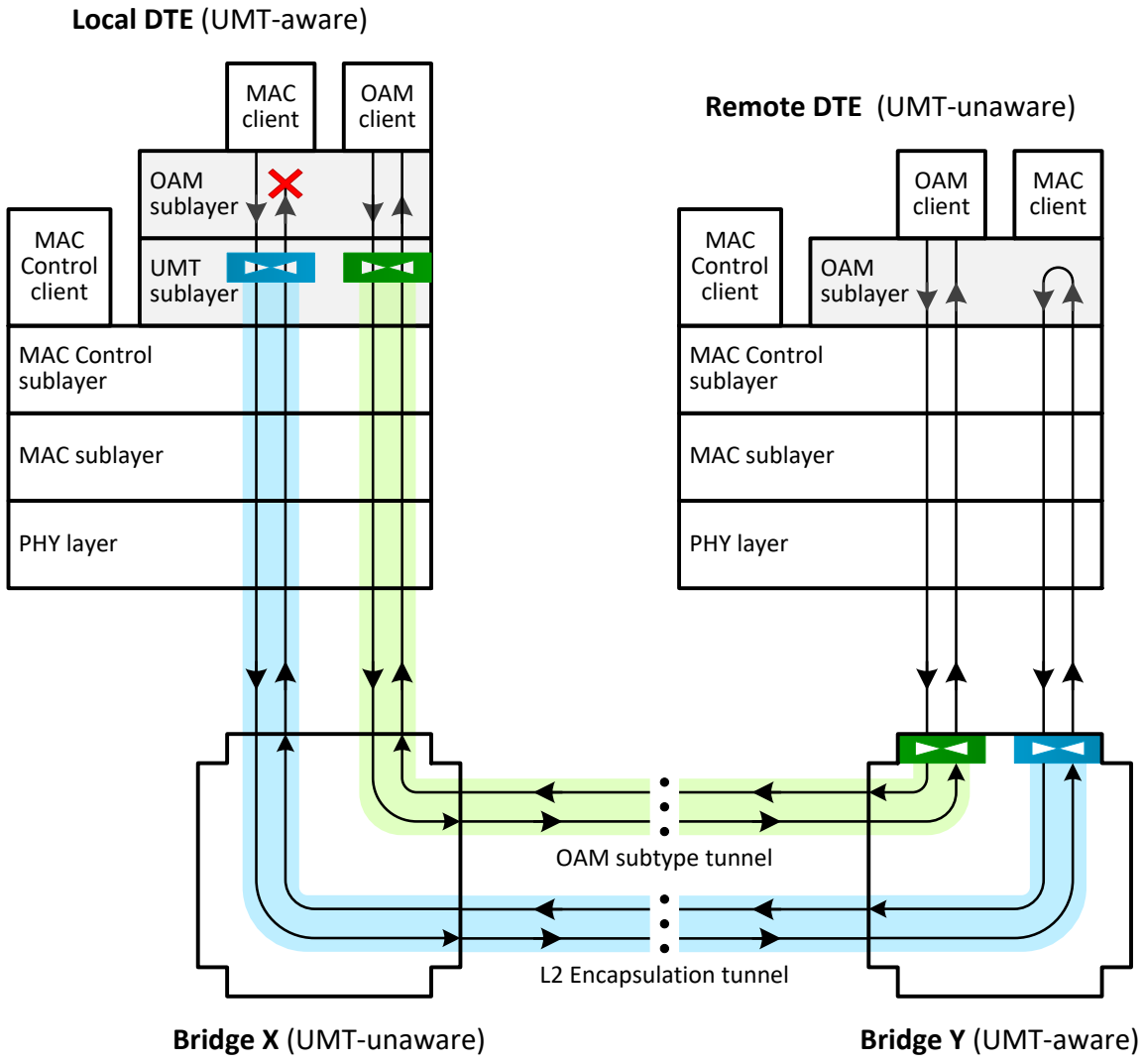


Figure 7-4— Remote OAM loopback over UMT tunnel with UMT-aware local DTE and UMT-unaware remote DTE.

Figure 7-5 represents a similar network configuration, but with both the local and the remote DTEs being UMT-aware.

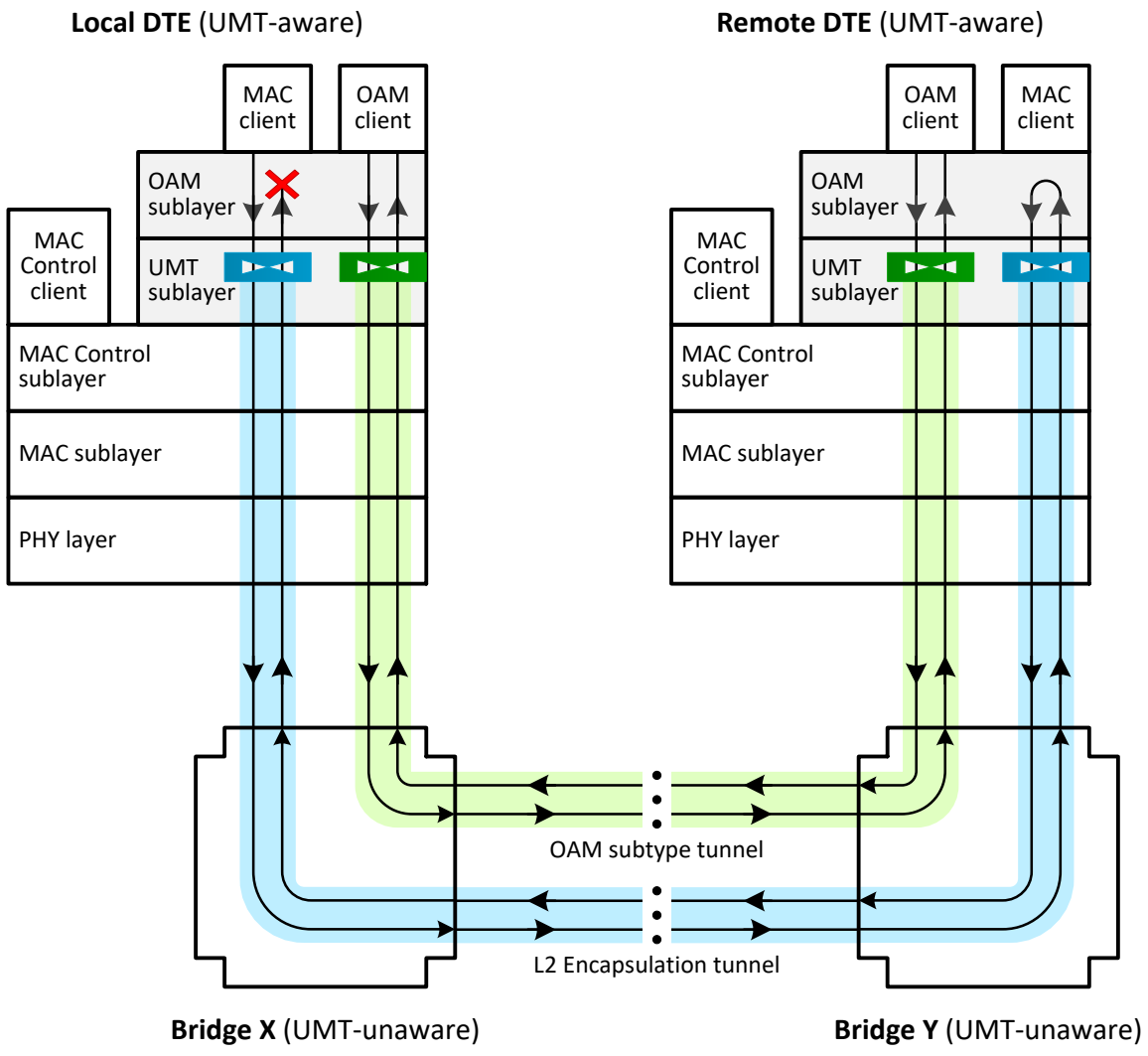


Figure 7-5— Remote OAM loopback over UMT tunnel with UMT-aware local DTE and UMT-aware remote DTE.

While the OAM subtype tunnel between the local and remote DTEs persists permanently to ensure that the two OAM entities are able to exchange OAMPDUs, the bidirectional tunnel for the looped-back data only needs to be established for the duration of the loopback mode. This tunnel has L2 encapsulation subtype in order to deliver any non-OAMPDU (regardless of their Source and destination MAC addresses) to from the local DTE to the remote DTE and in the reverse direction, from the remote DTE to the local DTE.

Table 7-4 illustrates the tunnel entrance rules for the UMT L2 encapsulation tunnel from the local DTE to the remote DTE. The table shows two rules that have different conditions, but identical actions. If these rules are provisioned in the bridge adjacent to the local DTE, as illustrated in Figure 7-3, these are ingress tunnel entrance rules. If the rules are provisioned in the local DTE itself, as illustrated in Figure 7-4 and Figure 7-5, these are egress tunnel entrance rules.

Table 7-4—Tunnel entrance rule for non-OAMPDU traffic from local DTE to remote DTE

Conditions	Actions
1. ETYPE_LEN != SP_TYPE	1. ADD(UMT_DST_ADD, <remote_MAC>) 2. ADD(UMT_SRC_ADD, <local_MAC>) 3. ADD(UMT_ETH_TYPE, UMT_TYPE) 4. ADD(UMT_SUBTYPE, L2_subtype)
1. ETYPE_LEN == SP_TYPE 2. XPDU_SUBTYPE != OAM_subtype	

NOTE:
<local_MAC > - MAC address associated with the loopback port in the local DTE
<remote_MAC > - MAC address associated with the loopback port in the remote DTE

SP_TYPE – Slow Protocol Ethertype value (see IEEE Std 802.3, 57A.4)
UMT_TYPE – Ethertype value identifying UMTPDUs (see 5.1)

OAM_subtype – UMT subtype value identifying OAMPDU payload (see 5.2)
L2_subtype – UMT subtype value identifying L2 encapsulation payload (see 5.2)

Table 7-5 illustrates the tunnel exit rule for the UMT L2 encapsulation tunnel from the local DTE to the remote DTE. If this rule is provisioned in the bridge adjacent to the remote DTE, as illustrated in Figure 7-3 and Figure 7-4, this rule is an egress tunnel exit rule. If the rule is provisioned in the remote DTE itself, as illustrated in Figure 7-5, this rule is an ingress tunnel exit rule.

Table 7-5—Tunnel exit rule for non-OAMPDU traffic from local DTE to remote DTE

Conditions	Actions
1. DST_ADDR == <remote_MAC> 2. SRC_ADDR == <local_MAC> 3. ETH_TYPE == UMT_TYPE 4. UMT_SUBTYPE == L2_subtype	1. REMOVE(UMT_DST_ADDR) 2. REMOVE(UMT_SRC_ADDR) 3. REMOVE(UMT_ETH_TYPE) 4. REMOVE(UMT_SUBTYPE)

NOTE:
<local_MAC > - MAC address associated with the loopback port in the local DTE
<remote_MAC > - MAC address associated with the loopback port in the remote DTE

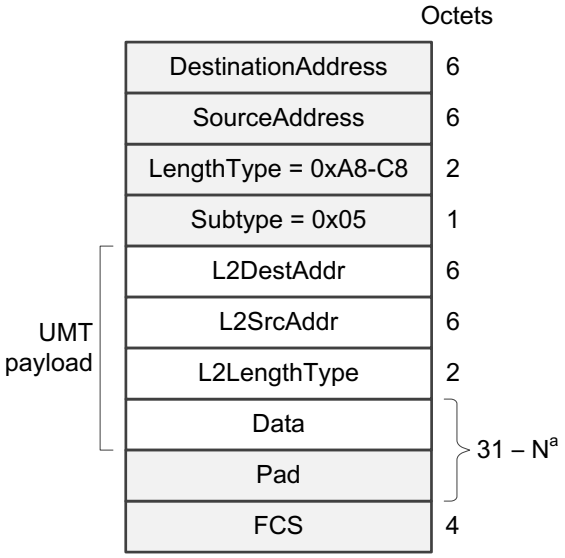
UMT_TYPE – Ethertype value identifying UMTPDUs (see 5.1)
L2_subtype – UMT subtype value identifying L2 encapsulation payload (see 5.2)

The entrance rules for the return tunnel (from the remote DTE back to the local DTE), the rules are similar to the rules shown in Table 6-8, but with <*local_MAC*> and <*remote_MAC*> values swapped. Similarly, the tunnel exit rule is as shown in Table 6-9, but also with <*local_MAC*> and <*remote_MAC*> values swapped.

7.3 Support for L2 subtype

7.3.1 L2_subtype UMTPDU Encoding

The frame structure of UMTPDU with L2 subtype shall be as depicted in Figure 7-6.



a – Maximum field length depends on frame type (see Figure 5-1).

Figure 7-6—Format of UMTPDU with L2 subtype

The structure of the *UMT payload* in the UMTPDU with L2 subtype is defined as follows:

—*L2DestAddr*:
This field carries the L2 destination address of the original L2 frame being tunneled using UMT.

—*L2SrcAddr*:
This field carries the L2 source address of the original L2 frame being tunneled using UMT.

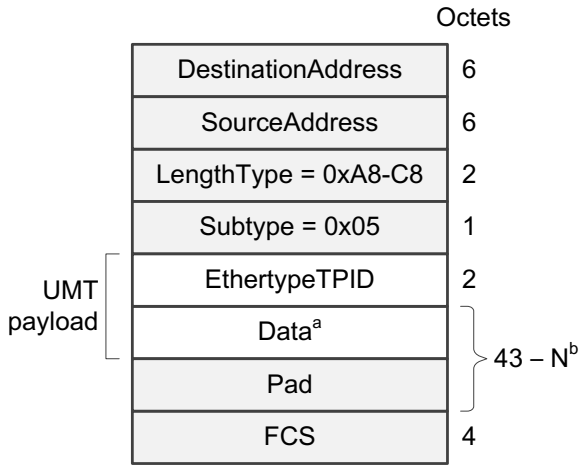
—*L2LengthType*:
This field carries the Length/Type value of the original L2 frame being tunneled using UMT.

—*Data*:
This field carries the L2 payload of the original L2 frame being tunneled using UMT. The combined size of the *Data* and *Pad* fields ranges between 31 and *N*, where *N* is defined in **Error! Reference source not found.**.

7.4 Support for L3 Subtype

7.4.1 L3_subtype UMTPDU Encoding

The frame structure of UMTPDU with L3 subtype shall be as depicted in Figure 7-7. The format of the Data/Pad field is dependent on the value of the *EthertypeTPID* field and is beyond the scope of this standard.



a – Field format depends on the value of *EthertypeTPID* field.
b – Maximum field length depends on frame type (see Figure 5-1).

Figure 7-7—Format of UMTPDU with L3 subtype

The structure of the *UMT payload* in the UMTPDU with L3 subtype is defined as follows:

—*EthertypeTPID*:
This field carries the L2 Ethertype/TPID value of the original L3 packet being tunneled using UMT.

—*Data*:
This field carries the L3 packet being tunneled using UMT. The combined size of the *Data* and *Pad* fields ranges between 43 and *N*, where *N* is defined in **Error! Reference source not found.**.

7.5 Support for OMCI