# Annex B UMT Peer Discovery and Tunnel Auto-Configuration

## B.1 Introduction

IEEE Std. 1904.2 Clause 4 defines a method for delivering service data units (SDU) for higher layer protocols across a layer-2 network in which those protocols would not normally be forwarded due to addressing conflicts or other factors. The described architecture consists of UMT peers that perform appropriate encapsulation of the UMT Client SDUs into UMTPDUs which are transmitted across a layer-2 network and received, decapsulated and the resulting UMT Client SDUs delivered to the desired UMT Client.

IEEE Std. 1904.2 requires that a UMT Peer be configured to know of the presence and functionality of another UMT Peer before they are able to transfer UMTPDUs between one another. IEEE Std. 1904.2 Clause 4, however, does not specify a method for automatically discovering the presence and capabilities of UMT Peers on a network.

This annex defines an architecture and system for automatic UMT Peer Discovery and for automatically configuring unicast tunnels between peers.

This annex is normative. Implementation of this annex is optional.

## B.2 Overview of UMT Peer Discovery Protocol

Figure B-1 depicts the topology of a network over which a set of UMT peers wish to discover one another for the purpose of transferring UMTPDUs.

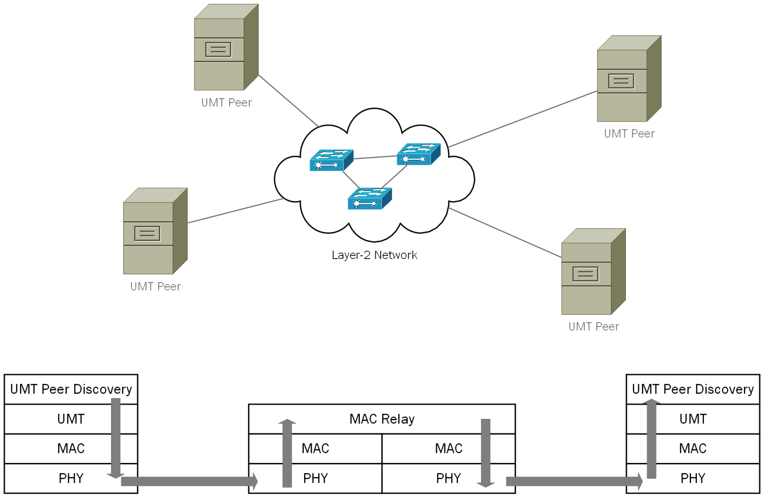


**Figure B-1 - Topology of UMT Peer Discovery**

In this generalized topology, a UMT peer wishes to discover and communicate with another UMT peer that is located one or more MAC Relay hops away. IEEE Std. 1904.2 Clause 4 allows multicast and broadcast operation for UMT. UMT Peer Discovery uses UMT broadcast operation to advertise the presence of a UMT peer and solicit neighboring UMT peers to respond to that advertisement to alert the UMT peer of their individual presence.

The UMT Peer Discovery function is, in fact, a UMT Client that uses the UMT Peer Maintenance UMT Subtype, referred to as a UMT Maintenance Service Data Unit (UMTMSDU), over a broadcast UMT tunnel adapter. A UMT Peer Discovery entity can be configured in *Active* mode or in *Passive* mode.

The packet flow of UMT Peer Discovery is shown in Figure B-2

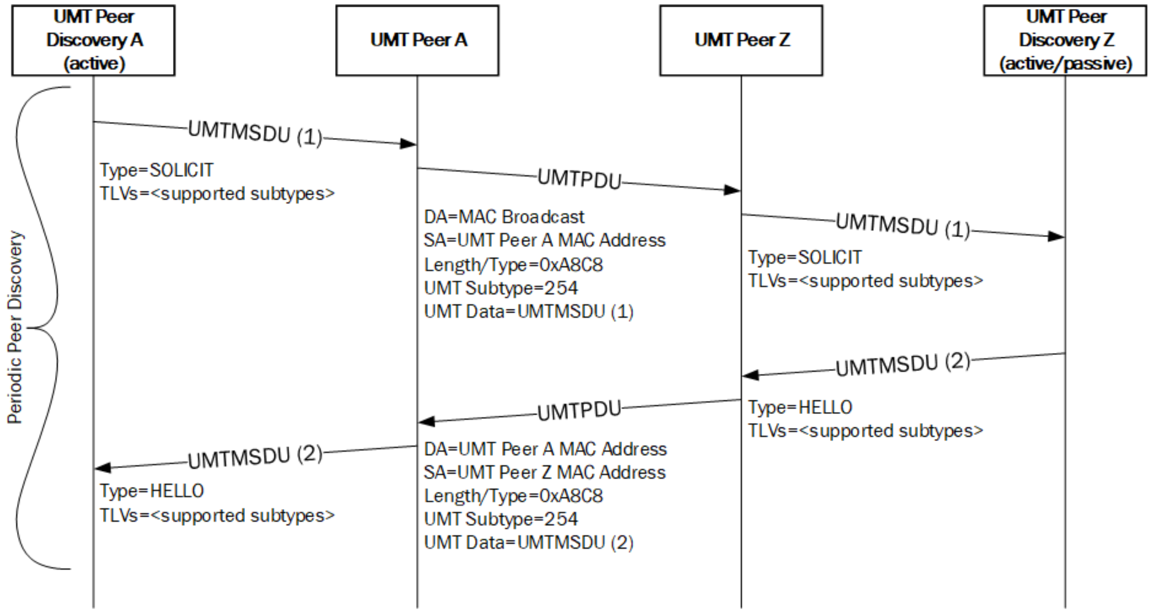


**Figure B-2 – Packet Flow for UMT Peer Discovery**

Periodically, a UMT Peer Discovery entity in Active mode forms a UMTPDU and transmits it as a MAC broadcast. The broadcast UMTPDU solicits neighboring UMT Peer Discovery entities (Active or Passive) to respond to the sending UMT Peer Discovery entity.

Upon receipt of the UMT Peer Discovery solicitation message, the receiving UMT Peer Discovery entity will, if local policy permits, form a UMTPDU and send it as a MAC unicast to the soliciting peer. This solicitation/response action allows the Active UMT Peer Discovery entity to build a table of neighboring UMT Peers and the capabilities of each.

The packet flow for automatic UMT tunnel configuration is show in Figure B-3.

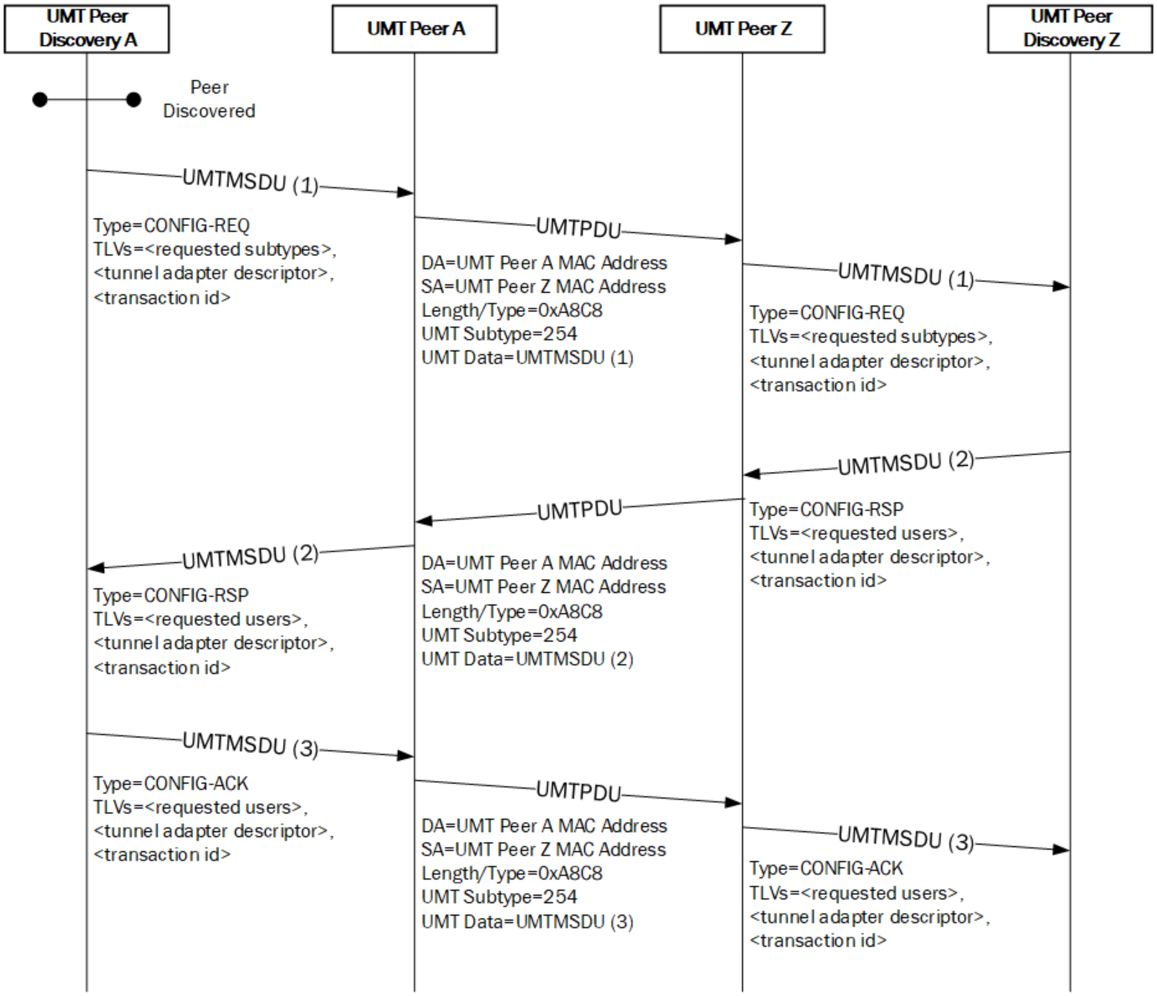


**Figure B-3 - Packet Flow for UMT Tunnel Automatic Configuration**

After the UMT Peers have discovered one another through manual configuration or through UMT Peer Discovery, a tunnel can be established automatically by the peers. A UMT tunnel is initiated when a UMT Peer entity sends a unicast message to another peer requesting that a tunnel adapter be created. The remote peer, if local policy permits, responds to indicate that the UMT peer is able and willing to create the tunnel adapter. That peer waits for the requesting peer to complete the tunnel configuration by sending an acknowledgement. In this exchange of configuration messages, the two UMT peers also send and negotiate tunnel parameters (for example, supported UMT Client protocols).

## B.3 Functional Specifications

### B.3.1 UMT Peer Discovery and Tunnel Auto-Configuration Service Interfaces

Figure B-4 depicts the usage of interlayer interfaces by the Discovery and Auto-Configuration processes in the UMT Peer Maintenance entity.

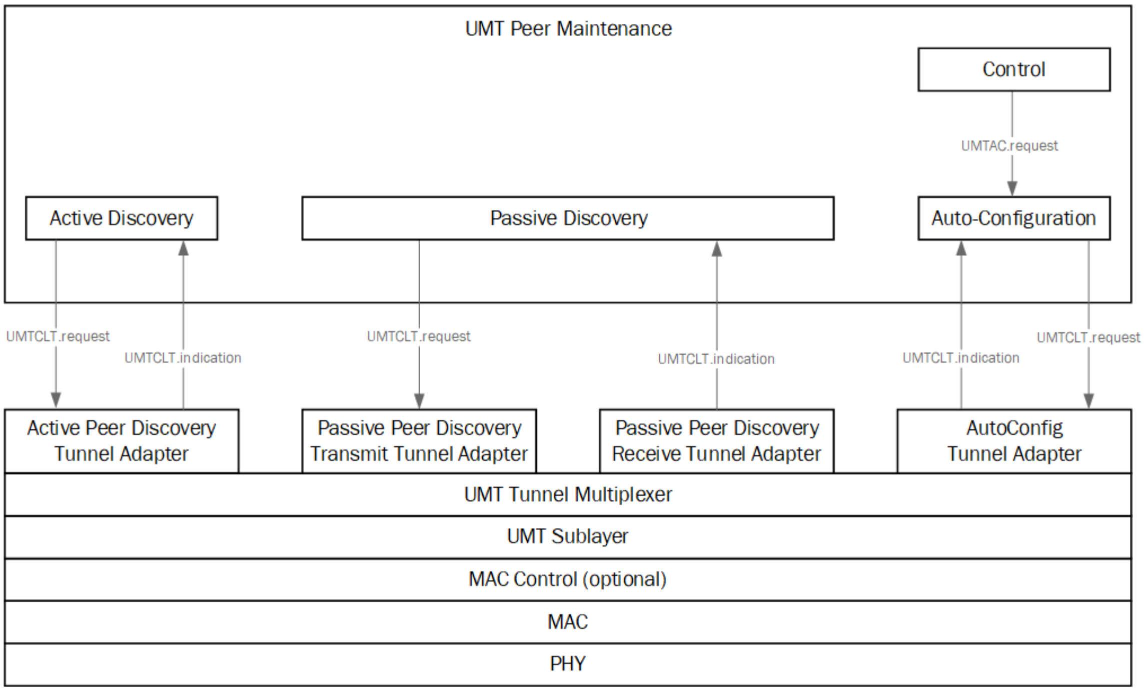


**Figure B-4 – UMT Discovery and Auto-Configuration service interfaces**

### B.3.2 Principles of Operation

UMT Peer Discovery employs the following principles and concepts:

a) Only an active UMT Peer Discovery entity may send unsolicited peer discovery messages.

b) A passive UMT Peer Discovery entity must remain silent unless it receives a solicitiation from an active peer.

c) Automatic UMT Peer Discovery is only responsible for building a database of neighboring UMT peers.

UMT Tunnel Auto-Configuration employs the following principles and concepts:

a) UMT Tunnel Auto-Configuration is not dependent upon automatic UMT Peer Discovery. Neighboring UMT peers may be configured manually by an administrator.

b) Any peer in the UMT network can initiate a tunnel configuration.

c) Since UMT tunnels are stateless, UMT Tunnel Auto-Configuration is not a method for establishing a tunnel. UMT Tunnel Auto-configuration is a method for requesting that a neighboring UMT peer create a Tunnel Adapter.

### B.3.3 UMT Peer Maintenance

The UMT Peer Maintenance entity is a multi-functional and extensible entity. In the context of this annex, the UMT Peer Maintenance entity is the context in which UMT Peer Discovery and UMT Tunnel Auto-Configuration is defined.

### B.3.3.1 UMT Peer Maintenance Interactions

#### B.3.3.1.1 Interlayer Interactions

All processes and functions within the UMT Peer Maintenance entity communicate with lower UMT layers using the following interlayer service interfaces:

UMTCLT.request

UMTCLT.indication

The UMTCLT.request and UMTCLT.indication service primitives are described in IEEE Std. 1904.2 Clause 4.

#### B.3.3.1.2 Intralayer Interactions

The UMT Peer Maintenance entity contains an abstract control process that communicates with the Auto-Configuration function using the following service interfaces:

UMTAC.request

The UMTAC.request service primitive described in this subclause is mandatory if the Auto-Configuration function is implemented.

##### B.3.3.1.2.1 UMTAC.request

This primitive triggers the Auto-Configuration function to initiate a request to create or delete a tunnel on a UMT Peer.

###### B.3.3.1.2.1.1 Function

The semantics of the primitive are as follows:

```
UMTAC. request (
                        action,
                        tunnel_adapter_descriptor
                        )
```

The action parameter indicates the action to be taken – create or delete. The tunnel_adapter_descriptor parameter specifies the tunnel adapter to be created on or deleted from the UMT Peer.

### B.3.3.2 Use of UMT Tunnel Adapters

As shown in Figure B-4, the Active Peer Discovery process, the Passive Peer Discovery process and the Auto-Configuration process are all clients to the UMT layers. Therefore, it is necessary for them to interact with one or more UMT Tunnel Adapters to enable them to operate.

### B.3.3.3 Active Peer Discovery Tunnel Adapter

The Active Peer Discovery Tunnel adapter is used by the Active Peer Discovery process to send SOLICIT messages via MAC broadcast and to receive HELLO messages via MAC unicast from any possible MAC source address. The Active Peer Discovery Tunnel Adapter is defined by the tuple:

```
        (
        Indicated DA = <Local UMT Peer MAC Address>    (DA of UMTPDU.indication)
```

Indicated SA = <any> (SA of UMTPDU.indication)
Requested DA = MAC Broadcast (DA of UMTPDU.request)
Requested SA = <Local UMT Peer MAC Address> (SA of UMTPDU.request)
Transmission Method = Broadcast
)

The Active Peer Discovery Tunnel Adapter shall be configured prior to or during initialization of the Active Peer Discovery process.

Editor’s Note: The tuple above is intended to represent a “filter” definition of the Tunnel Adapter. Requested DA corresponds to the destination_address parameter of UMTPDU.request primitive. Requested SA corresponds to the source_address parameter of the UMTPDU.request primitive. Indicated_DA and Indicated_SA correspond to the destination_address and source_address parameters (respectively) of the UMTPDU.indication primitive.

#### B.3.3.4 Passive Peer Discovery Receive Tunnel Adapter

The Passive Peer Discovery Receive Tunnel Adapter is used by the Passive Peer Discovery process to receive SOLICIT messages via MAC broadcast from any possible MAC source address. It shall not be used to transmit messages. The Passive Peer Discovery Receive Tunnel Adapter is a receive-only tunnel adapter. The Passive Peer Discovery Receive Tunnel Adapter is defined by the tuple:

(
Indicated DA = MAC Broadcast (DA of UMTPDU.indication)
Indicated SA = <any> (SA of UMTPDU.indication)
Requested DA = <N/A> (DA of UMTPDU.request)
Requested SA = <N/A> (SA of UMTPDU.request)
Transmission Method = Receive Only
)

The Passive Peer Discovery Receive Tunnel Adapter shall be configured prior to or during initialization of the Passive Peer Discovery process.

#### B.3.3.5 Passive Peer Discovery Transmit Tunnel Adapter

The Passive Peer Discovery Transmit Tunnel Adapter is a transient entity that is used by the Passive Peer Discovery process to transmit HELLO messages via unicast to the UMT peer from which a SOLICIT is received. It shall not be used to receive messages. The Passive Peer Discovery Transmit Tunnel Adapter is a transmit-only tunnel adapter. The Passive Peer Discovery Transmit Tunnel Adapter is defined by the tuple:

(
Indicated DA = <N/A> (DA of UMTPDU.indication)
Indicated SA = <N/A> (SA of UMTPDU.indication)
Requested DA = <Remote UMT Peer MAC Address> (DA of UMTPDU.request)
Requested SA = <Local UMT Peer MAC Address> (SA of UMTPDU.request)
Transmission Method = Unicast
)

The Passive Peer Discovery Transmit Tunnel Adapter shall be configured immediately prior to sending a HELLO message and shall be deleted immediately after transmitting the HELLO message.

#### B.3.3.6 AutoConfig Tunnel Adapter

The AutoConfig Tunnel adapter is used by the Passive Peer Discovery process to send HELLO messages in response to SOLICIT messages. The AutoConfig Tunnel Adapater is also used by the Auto-Configuration process to exchange configuration messages via MAC unicast between two UMT peers. The Auto-Configuration process requires multiple AutoConfig Tunnel adapters. A unique AutoConfig Tunnel adapter is required for each UMT peer wishing to participate in the Auto-Configuration process. The AutoConfig Tunnel Adapter is defined by the tuple:

```
(
Indicated DA = <Local UMT Peer MAC Address>
Indicated SA = <Remote UMT Peer MAC Address>
Requested DA = <Remote UMT Peer MAC Address>
Requested SA = <Local UMT Peer MAC Address>
Transmission Method = Unicast
)
```

The AutoConfig Tunnel Adapter shall be configured prior to or during initialization of the Auto-Configuration process.

## B.4 Detailed functions and state diagrams

### B.4.1 State diagram variables

#### B.4.1.1 Constants

UMTM_Subtype
The value of the UMT Subtype field for UMT Maintenance SDUs (See Table 4-2).

ta_unicast_mode
The value of the Tunnel Adapter Transmission Method that indicates unicast transmission mode. (See B.5.3.4.1)

ta_rxonly_mode
The value of the Tunnel Adapter Transmission Method that indicates receive-only transmission mode. (See B.5.3.4.1)

NULL
This constant is used to indicate that no value is assigned or an empty value is assigned.

#### B.4.1.2 Variables

BEGIN
A variable that resets the functions within a UMT Peer Maintenance process.
Values: TRUE; when any of the component UMT sublayers is reset.
FALSE; When (re-)initialization has completed.

req_umt_subtype
The value of the umt_subtype parameter passed to the UMT Client in the UMTCLT.request primitive.
Value: Integer (See Table 4-2)

req_umtm_message_type

The value of the UMTM Message Type field in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See Table B-1

req_revision
The value of the Revision field in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.1

req_sequence_number
The value of the Sequence Number field in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.1

req_supported_umt_subtypes_tlv
The value of the Supported UMT Subtypes TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.1

req_requested_umt_subtypes_tlv
The value of the Requested UMT Subtypes TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.2

req_umt_peer_identifier_tlv
The value of the UMT Peer Identifier TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.5

req_transaction_id_tlv
The value of the Transaction Identifier TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.3

req_tunnel_adapter_descriptor_tlv
The value of the Tunnel Adapter Descriptor TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.4

req_reason_code
The value of the Reason Code TLV in a requested UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu parameter.
Values: See B.5.3.6

req_umt_client_sdu
The value of the umt_client_sdu parameter passed to the UMT Tunnel Adapter in the UMTCLT.request primitive.

ind_SA

The value of the source address parameter received in a UMTCLT.indication primitive.

ind_DA

The value of the destination address parameter received in a UMTCLT.indication primitive.

ind_umt_subtype

The value of the Subtype field in a received UMT protocol frame (see Table 4-2) and is used to determine the UMT Client to which the UMT payload is destined.
Value: Integer (See Table 4-2)

ind_umtm_message_type

The value of the UMTM Message Type field in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the ind_umt_client_sdu parameter.
Values: See Table B-1

ind_revision

The value of the Revision field in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.1

ind_sequence_number

The value of the Sequence Number field in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.1

ind_supported_umt_subtypes_tlv

The value of the Supported UMT Subtypes TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.1

ind_requested_umt_subtypes_tlv

The value of the Requested UMT Subtypes TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.2

ind_umt_peer_identifier_tlv

The value of the UMT Peer Identifier TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.request primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.5

ind_transaction_id_tlv

The value of the Transaction Identifier TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.request primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.3

ind_tunnel_adapter_descriptor_tlv

The value of the Tunnel Adapter Descriptor TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.request primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.4

ind_reason_code

The value of the Reason Code TLV in a received UMT Peer Maintenance SDU and passed to the UMT Peer Maintenance entity via the UMTCLT.request primitive as part of the ind_umt_client_sdu parameter.
Values: See B.5.3.6

ind_umt_client_sdu

The value of the Data field in a received UMT protocol frame and is passed to the UMT Peer Maintenance entity in the umt_client_sdu paremeter of the UMTPDU.indication primitive.

req_action
req_tunnel_adapter_descriptor

The parameters of the UMTAC.request service primitive as defined in B.3.3.1.2.1

req_umtm_data

The fields contained in a UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter in the UMTCLT.request service primitive.

req_action_create

The action parameter of UMTAC.request, as defined in B.3.3.1.2.1, with a value indicating a create action.

req_action_delete

The action parameter of UMTAC.request, as defined in B.3.3.1.2.1, with a value indicating a delete action.

req_tunnel_adapter_descriptor

The value of the tunnel_adapter_descriptor parameter of the UMTAC.request service primitive, as defined in B.3.3.1.2.1.

max_retries

This variable defines the maximum number of times a UMT Peer Maintenance process will send a duplicate message in an attempt to communicate with a peer entity.

param_list
cfg_req

The values returned from the check_cfg_request function.

del_req

The value returned from the check_del_request function.

tunnel_descriptor

This variable represents the parameters that define a tunnel adapter on a UMT peer. The parameters required to define a tunnel adapter are specified by the Tunnel Adapter Descriptor TLV defined in B.5.3.4. Those paramters are represented in the state diagrams as:

ta_indicated_da: Tunnel Adapter Indicated Destination Address Subtype (See B.5.3.4.3)
ta_ indicated _sa: Tunnel Adapter Indicated Source Address Subtype (See B.5.3.4.2)
ta_requested_da: Tunnel Adapter Requested Destination Address Subtype (See B.5.3.4.5)
ta_requested_sa: Tunnel Adapter Requested Source Address Subtype (See B.5.3.4.4)
ta_tx_method: Tunnel Adapter Transmission Method Subtype (See B.5.3.4.1)

requested_umt_subtypes

This variable represents the value contained in the Requested UMT Subtypes TLV (See B.5.3.2)

indicated_umt_mac_address

The MAC address of the local UMT peer.

### B.4.1.3 Counters

retry_counter

A counter used to limit the number of duplicate UMT Maintenance SDUs sent during a Peer Discovery or Auto-Configuration negotiation.

### B.4.1.4 Timers

discovery_tx_timer

Timer used to regulate the frequency that peer discovery SOLICIT messages are sent.

retry_timer

Timer used to regulate the frequency that auto-configuration SDUs are sent when no response is received to a corresponding request.

### B.4.1.5 Functions

save_peer_info(source_address, umt_client_sdu)

This function parses received SOLICIT and HELLO messages and saves the received data for use by other processes in the UMT Maintenance entity (e.g. Auto-Configuration). This function requires as its arguments, the source address parameter and a UMT Client Service Data Unit as received via the UMTCLT.indication primitive.

create_tunnel_adapter(tunnel_descriptor, requested_umt_subtypes)

This function creates a UMT Tunnel Adapter on the local UMT peer, if it does not already exist, and makes the tunnel accessible by the UMT clients indicated by the requested_umt_subtypes parameter. The function requires a tunnel descriptor and list of UMT Subtypes (see Table 4-2) as its arguments.

delete_tunnel_adapter(tunnel_descriptor, requested_umt_subtypes)

This function removes access to the tunnel adapter indicated by tunnel_descriptor for the UMT clients indicated by requested_umt_subtypes and deletes the tunnel adapter from the local UMT peer if there are no remaining clients. The function requires a tunnel descriptor and list of UMT Subtypes (see Table 4-2) as its arguments.

(cfg_req, param_list) ⇐ check_cfg_request(umt_client_sdu)

This function parses received CONFIG-REQ messages and returns a value, in the cfg_req variable, indicating the status of the CONFIG-REQ. This function requires as its only argument, a UMT Client Service Data Unit as received via the UMTCLT.indication primitive. A return value of ACK indicates that the request is acceptable. A return value of REJ indicates that the request contains unacceptable fields or TLVs. A return value of NAK indicates that the request contains acceptable fields and TLVs but the value of of one or more of the fields or TLVs is unacceptable. If this function returns NAK, it will also return a list of the fields, in the param_list variable, and TLVs containing unacceptable values along with values for each that are acceptable to the local peer and a reason code to indicate the reason the request is unacceptable. If this function returns REJ, it will also return a list of the unacceptable fields and TLVs along with values for each that are acceptable to the local peer and a reason code to indicate the reason the request is unacceptable.

del_req ⇐ check_del_request(umt_client_sdu)

This function parses received DELETE-REQ messages and returns a value, in the del_req variable, indicating the status of the DELETE-REQ. A return value of ACK indicates that the request is acceptable. This function requires as its only argument, a UMT Client Service Data Unit as received via the UMTCLT.indication primitive. A return value of REJ indicates that the message contains unacceptable fields or TLVs. A return value of NAK indicates that the message contains acceptable fields and TLVs but the value of one or more of the fields or TLVs is unacceptable. If this function returns NAK, it will also return a reason code to indicate the reason the request is unacceptable. If this function returns REJ, it will also return a reason code to indicate the reason the request is unacceptable. This function shall not return a list of unacceptable or acceptable fields, TLVs or values.

valueof(tlv)

This function returns the value contained in a TLV.

### B.4.1.6 Messages

UMTCLTREQ_SOLICIT
Alias for the request for a peer discovery SOLICIT message to be sent via the UMTCLT.request primitive. The requested SOLICIT message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ SOLICIT (see Table B-1)
req_revision
req_sequence_number
req_supported_umt_subtypes_tlv
req_umt_peer_identifier_tlv

UMTCLTIND_HELLO
Alias for the receipt of a peer discovery HELLO message via the UMTCLT.indication primitive. The received HELLO message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ HELLO (see Table B-1)
ind_revision
ind_sequence_number
ind_supported_umt_subtypes_tlv
ind_umt_peer_identifier_tlv

UMTCLTIND_SOLICIT
Alias for the receipt of a peer discovery SOLICIT message via the UMTCLT.indication primitive. The received SOLICIT message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ SOLICIT (see Table B-1)
ind_revision
ind_sequence_number
ind_supported_umt_subtypes_tlv
ind_umt_peer_identifier_tlv

UMTCLTREQ_HELLO

Alias for the request for a peer discovery HELLO message to be sent via the UMTCLT.request primitive. The requested HELLO message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ HELLO (see Table B-1)
req_revision
req_sequence_number
req_supported_umt_subtypes_tlv
req_umt_peer_identifier_tlv

UMTAC_CREATE

Alias for UMTAC.request(req_action, req_tunnel_adapter_descriptor), where req_action contains the value indicating a create action.

UMTAC_DELETE

Alias for UMTAC.request(req_action, req_tunnel_adapter_descriptor), where req_action contains the value indicating a delete action.

UMTCLTREQ_CFGREQ

Alias for the request for a Auto-Configuration CONFIG-REQ message to be sent via the UMTCLT.request primitive. The requested CONFIG-REQ message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ CONFIG-REQ (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv

RX_CFGNAK

Alias for the receipt of an Auto-Configuration CONFIG-NAK message via the UMTCLT.indication primitive. The received CONFIG-NAK message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ CONFIG-NAK (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_requested_umt_subtypes_tlv (optional per B.4.3.2.4)
ind_tunnel_adapter_descriptor_tlv (optional per B.4.3.2.4)
ind_reason_code

RX_CFGREJ

Alias for the receipt of an Auto-Configuration CONFIG-REJ message via the UMTCLT.indication primitive. The received CONFIG-REJ message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ CONFIG-REJ (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_requested_umt_subtypes_tlv (optional per B.4.3.2.5)
ind_tunnel_adapter_descriptor_tlv (optional per B.4.3.2.5)
ind_reason_code

RX_CFGRSP

Alias for the receipt of an Auto-Configuration CONFIG-RSP message via the UMTCLT.indication primitive. The received CONFIG-RSP message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ CONFIG-RSP (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_requested_umt_subtypes_tlv
ind_tunnel_adapter_descriptor_tlv

UMTCLTREQ_CFGACK

Alias for the request for a Auto-Configuration CONFIG-ACK message to be sent via the UMTCLT.request primitive. The requested CONFIG-ACK message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ CONFIG-ACK (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv

UMTCLTIND_CFGREQ

Alias for the receipt of a Auto-Configuration CONFIG-REQ message via the UMTCLT.indication primitive. The received CONFIG-REQ message contains the following fields, parameters and values:

```
ind_umt_subtype
ind _umtm_message_type                  ⇐ CONFIG-REQ (see Table B-1)
ind _revision
ind _sequence_number
ind _transaction_id_tlv
ind _umt_peer_identifier_tlv
ind _requested_umt_subtypes_tlv
ind _tunnel_adapter_descriptor_tlv
```

UMTCLTREQ_CFGRSP

Alias for the request for a Auto-Configuration CONFIG-RSP message to be sent via the UMTCLT.request primitive. The requested CONFIG-RSP message contains the following fields, parameters and values:

```
req_umt_subtype
req_umtm_message_type                   ⇐ CONFIG-RSP (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv
```

UMTCLTREQ_CFGNAK

Alias for the request for a Auto-Configuration CONFIG-NAK message to be sent via the UMTCLT.request primitive. The requested CONFIG-NAK message contains the following fields, parameters and values:

```
req_umt_subtype
req_umtm_message_type                   ⇐ CONFIG-NAK (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
param_list
req_reason_code
```

UMTCLTREQ_CFGREJ

Alias for the request for a Auto-Configuration CONFIG-REJ message to be sent via the UMTCLT.request primitive. The requested CONFIG-REJ message contains the following fields, parameters and values:

```
req_umt_subtype
req_umtm_message_type                   ⇐ CONFIG-REJ (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
param_list
req_reason_code
```

UMTCLTIND_CFGACK

Alias for the receipt of a Auto-Configuration CONFIG-ACK message via the UMTCLT.indication primitive. The received CONFIG-ACK message contains the following fields, parameters and values:

ind_umt_subtype
ind _umtm_message_type ⇐ CONFIG-ACK (see Table B-1)
ind _revision
ind _sequence_number
ind _transaction_id_tlv
ind _umt_peer_identifier_tlv
ind _requested_umt_subtypes_tlv
ind _tunnel_adapter_descriptor_tlv

UMTCLTREQ_DELREQ

Alias for the request for a Auto-Configuration DELETE-REQ message to be sent via the UMTCLT.request primitive. The requested DELETE-REQ message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ DELETE-REQ (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv

RX_DELREJ

Alias for the receipt of an Auto-Configuration DELETE-REJ message via the UMTCLT.indication primitive. The received DELETE-REJ message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ DELETE-REJ (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_reason_code

RX_DELRSP

Alias for the receipt of an Auto-Configuration DELETE-RSP message via the UMTCLT.indication primitive. The received DELETE-RSP message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ DELETE-RSP (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_requested_umt_subtypes_tlv
ind_tunnel_adapter_descriptor_tlv

UMTCLTREQ_DELACK

Alias for the request for a Auto-Configuration DELETE-ACK message to be sent via the UMTCLT.request primitive. The requested DELETE-ACK message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ DELETE-ACK (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv

UMTCLTIND_DELREQ

Alias for the receipt of a Auto-Configuration DELETE-REQ message via the UMTCLT.indication primitive. The received DELETE-REQ message contains the following fields, parameters and values:

ind_umt_subtype
ind_umtm_message_type ⇐ DELETE-REQ (see Table B-1)
ind_revision
ind_sequence_number
ind_transaction_id_tlv
ind_umt_peer_identifier_tlv
ind_requested_umt_subtypes_tlv
ind_tunnel_adapter_descriptor_tlv

UMTCLTREQ_DELRSP

Alias for the request for a Auto-Configuration DELETE-RSP message to be sent via the UMTCLT.request primitive. The requested DELETE-RSP message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ DELETE-RSP (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv

UMTCLTREQ_DELREJ

Alias for the request for a Auto-Configuration DELETE-REJ message to be sent via the UMTCLT.request primitive. The requested DELETE-REJ message contains the following fields, parameters and values:

req_umt_subtype
req_umtm_message_type ⇐ DELETE-REJ (see Table B-1)
req_revision
req_sequence_number
req_transaction_id_tlv
req_umt_peer_identifier_tlv
req_reason_code

UMTCLTIND_DELACK

Alias for the receipt of a Auto-Configuration DELETE-ACK message via the UMTCLT.indication primitive. The received DELETE-ACK message contains the following fields, parameters and values:

| | |
|---|---|
| ind_umt_subtype | |
| ind_umtm_message_type | ⇐ DELETE-ACK (see Table B-1) |
| ind_revision | |
| ind_sequence_number | |
| ind_transaction_id_tlv | |
| ind_umt_peer_identifier_tlv | |
| ind_requested_umt_subtypes_tlv | |
| ind_tunnel_adapter_descriptor_tlv | |

### B.4.2 UMT Peer Discovery

As depicted in Figure B-4, the UMT Discovery function is contained in the UMT Peer Maintenance entity and consists of:

a) ***Active Discovery***. This function is responsible for soliciting discovery responses from neighboring UMT peers.

b) ***Passive Discovery***. This function is responsible for listening for UMT discovery solicitations and responding accordingly to received solicitations.

#### B.4.2.1 Active Discovery

A UMT Maintenance entity may implement the Active Discovery process. If the Active Discovery process is implemented, it shall implement the active discovery state diagram shown in Figure B-5.

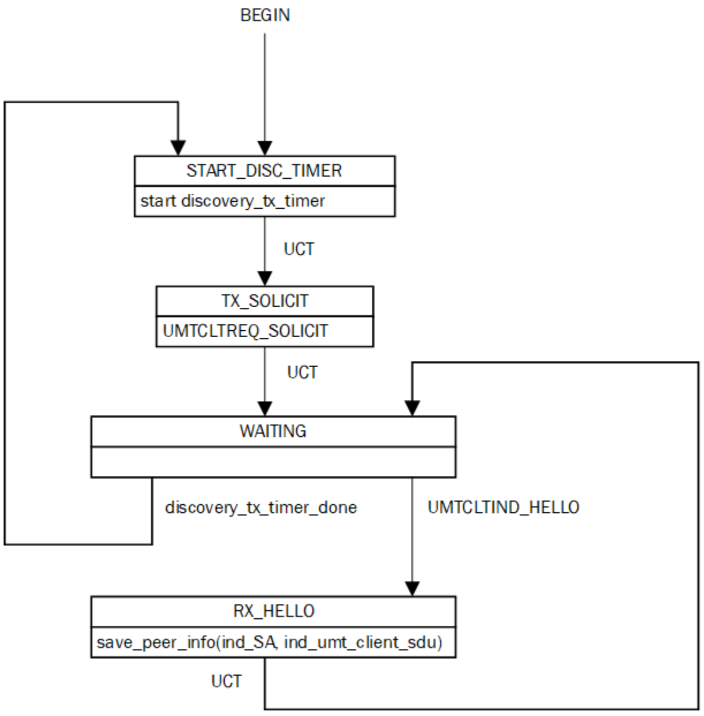


**Figure B-5 - Active Discovery Process State Diagram**

### B.4.2.1.1 START_DISC_TIMER State

Upon initialization, the START_DISC_TIMER state is entered. In the START_DISC_TIMER state, the Active Discovery process starts the discovery_tx_timer. Upon completion of the START_DISC_TIMER state, the Active Discovery process transitions to the TX_SOLICIT state.

### B.4.2.1.2 TX_SOLICIT State

When the Active Discovery process enters the TX_SOLICIT state, the Active Discovery process asserts the UMTCLT.request primitive with the required parameters to send a UMT Maintenance SOLICIT message. The UMTCLT.request primitive is asserted toward the Active Peer Discovery Tunnel Adapter (See B.3.3.3) so that the SOLICIT message is sent as a MAC broadcast.

### B.4.2.1.3 WAITING State

The Active Discovery process enters the WAITING state after completing the TX_SOLICIT state. In the WAITING state, the Active Discovery process waits for a UMT Maintenance HELLO message to arrive via the UMTCLT.indication primitive or for the discovery_tx_timer to expire.

If the discovery_tx_timer expires, the Active Discovery process moves back to the TX_SOLICIT state to send another UMT Maintenance SOLICIT message.

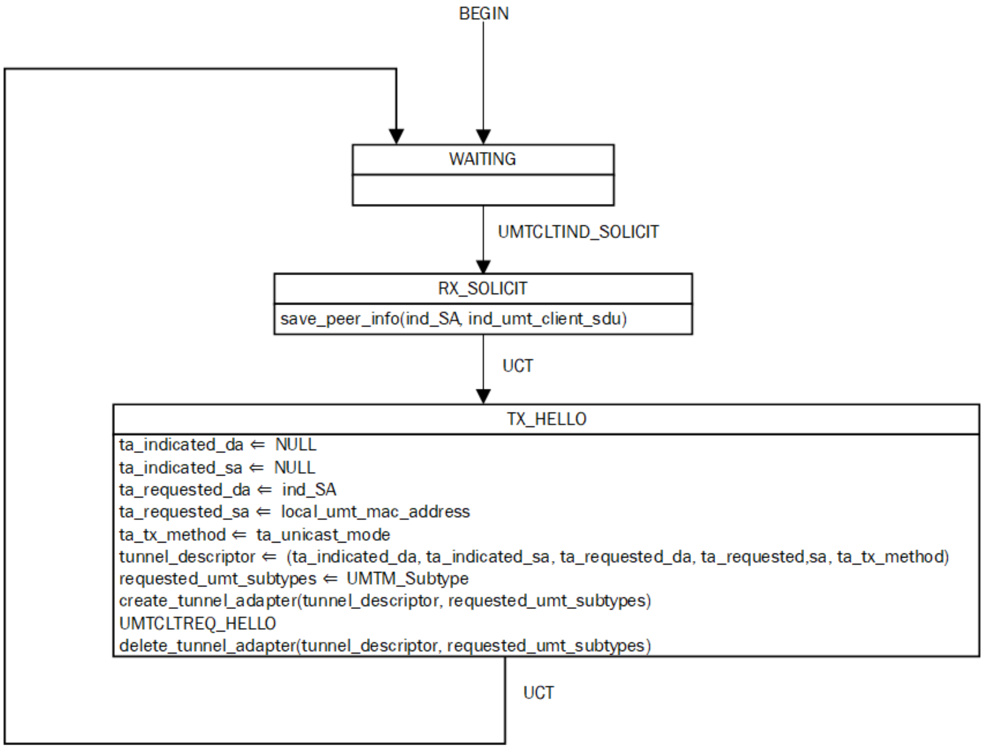
If the Active Discovery process receives a UMTCLT.indication containing a UMT Maintenance HELLO message, the Active Discovery process moves to the RX_HELLO state. All other received message types are silently ignored by the Active Discovery process.

#### B.4.2.1.4 RX_HELLO State

Upon entering the RX_HELLO state, the Active Discovery process calls the save_peer_info function to store the information received in the UMT Maintenance HELLO message. Upon completion, the Active Discovery Process moves to the WAITING state.

### B.4.2.2 Passive Discovery Process

A UMT Maintenance entity may implement the Passive Discovery process. If the Passive Discovery process is implemented, it shall implement the passive discovery state diagram shown in Figure B-6.



BEGIN
WAITING
UMTCLTIND_SOLICIT
RX_SOLICIT
save_peer_info(ind_SA, ind_umt_client_sdu)
UCT
TX_HELLO
ta_indicated_da ⇐ NULL
ta_indicated_sa ⇐ NULL
ta_requested_da ⇐ ind_SA
ta_requested_sa ⇐ local_umt_mac_address
ta_tx_method ⇐ ta_unicast_mode
tunnel_descriptor ⇐ (ta_indicated_da, ta_indicated_sa, ta_requested_da, ta_requested,sa, ta_tx_method)
requested_umt_subtypes ⇐ UMTM_Subtype
create_tunnel_adapter(tunnel_descriptor, requested_umt_subtypes)
UMTCLTREQ_HELLO
delete_tunnel_adapter(tunnel_descriptor, requested_umt_subtypes)
UCT

**Figure B-6 - Passive Discovery Process State Diagram**

#### B.4.2.2.1 WAITING State

Upon initialization the Passive Discovery process enters the WAITING state. In the WAITING state, the Passive Discovery process waits to receive a UMT Maintenance SOLICIT message via the UMTCLT.indication primitive asserted by the Passive Peer Discovery Receive Tunnel Adapter. Upon receipt of the SOLICIT message, the Passive Discovery process moves to the RX_SOLICIT state.

#### B.4.2.2.2 RX_SOLICIT State

Upon entry to the RX_SOLICIT state, the Passive Discovery process calls the save_peer_info function to store the information received in the UMT Maintenance SOLICIT message. Upon completion, the Passive Discovery process moves to the TX_HELLO state.

#### B.4.2.2.3 TX_HELLO State

When the the Passive Discovery process enters the TX_HELLO state, the Passive Discovery process calls the create_tunnel_adapter function to create the Passive Peer Discovery Transmit Tunnel Adapter. The Passive Discovery process then asserts the UMTCLT.request primitive with the required parameters to send a UMT Maintenance HELLO message. After UMTCLT.request primitive is asserted, the Passive Discovery process calls the delete_tunnel_adapter function to remove the Passive Peer Discovery Transmit Tunnel Adapter from operation.

## B.4.3 UMT Auto-Configuration

As depicted in Figure B-4, the UMT Auto-Configuration function is contained in the UMT Peer Maintenance entity. The Auto-Configuration process is responsible for communicating with peer Auto-Configuration entities to negotiate the creation and deletion of UMT Tunnel Adapters. The Auto-Configuration process is comprised of the following subprocesses:

a) ***Configuration Initiator***. This function intitiates a request to a peer Auto-Configuration entity to request that a new Tunnel Adapter be created on the peer.

b) ***Configuration Init Receiver***. This function receives configuration requests from peer Auto-Configuration entities and negotiates with the peer entity to agree on the parameters for configuring a new Tunnel Adapter.

c) ***Delete Initiator***. This function intitiates a request to a peer Auto-Configuration entity to request that a Tunnel Adapter be deleted from the peer.

d) ***Delete Receiver***. This function receives requests for tunnel adapter deletion from peer Auto-Configuration entities and negotiates with the peer entity to agree on the deletion of the Tunnel Adapter.

### B.4.3.1 Configuration Initiator

A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration process is implemented, it shall implement the Configuration Initiator state diagram shown in Figure B-7.

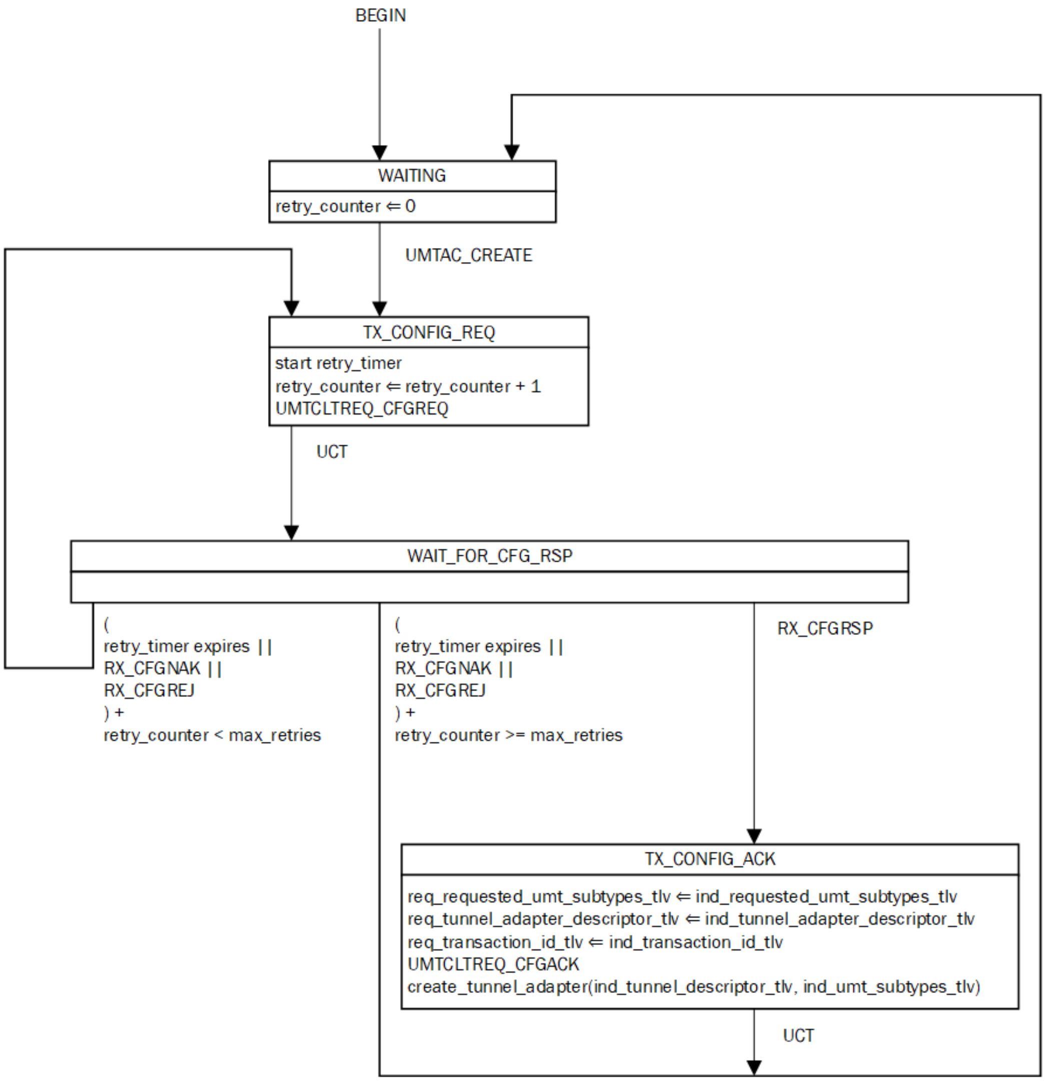

BEGIN
WAITING
retry_counter ⇐ 0
UMTAC_CREATE
TX_CONFIG_REQ
start retry_timer
retry_counter ⇐ retry_counter + 1
UMTCLTREQ_CFGREQ
UCT
WAIT_FOR_CFG_RSP
(
retry_timer expires | |
RX_CFGNAK | |
RX_CFGREJ
) +
retry_counter < max_retries
(
retry_timer expires | |
RX_CFGNAK | |
RX_CFGREJ
) +
retry_counter >= max_retries
RX_CFGRSP
TX_CONFIG_ACK
req_requested_umt_subtypes_tlv ⇐ ind_requested_umt_subtypes_tlv
req_tunnel_adapter_descriptor_tlv ⇐ ind_tunnel_adapter_descriptor_tlv
req_transaction_id_tlv ⇐ ind_transaction_id_tlv
UMTCLTREQ_CFGACK
create_tunnel_adapter(ind_tunnel_descriptor_tlv, ind_umt_subtypes_tlv)
UCT

**Figure B-7 - Configuration Initiator State Diagram**

#### B.4.3.1.1 WAITING State

Upon initialization, the WAITING state is entered. In the WAITING state, the Configuration Initiator subprocess sets the retry_counter to zero and waits for assertion of the UMTAC.request primitive with the action parameter set to indicate a create action.

#### B.4.3.1.2 TX_CONFIG_REQ State

When the TX_CONFIG_REQ state is entered, the Configuration Initiator subprocess starts the retry_timer, increments the retry_counter, and asserts the UMTCLT.request primitive with the required parameters to send a CONFIG-REQ message.

#### B.4.3.1.3 WAIT_FOR_CFG_RSP State

In the WAIT_FOR_CFG_RSP state, the Configuration Initiator subprocess waits for any of the following events:

retry_timer expires

If the retry_timer expires, the Configuration Initiator subprocess will compare the value of retry_counter to the value of max_retries. If retry_timer is less than max_retries, then the Configuration Initiator subprocess moves to the TX_CONFIG_REQ state. If retry_timer equals or exceeds max_retries then the Configuration Initiator subprocess moves to the WAITING state.

Receive CONFIG-NAK

If the UMTCLT.indication primitive is asserted and contains a CONFIG-NAK, the Configuration Initiator subprocess will compare the value of retry_counter to the value of max_retries. If retry_timer is less than max_retries, then the Configuration Initiator subprocess moves to the TX_CONFIG_REQ state where the Configuration Initiator subprocess shall adjust the values of the parameters, fields, and TLVs to be sent in the CONFIG-REQ in a way to achieve agreement with the remote peer's configuration as sent in the CONFIG-NAK. If retry_timer equals or exceeds max_retries then the Configuration Initiator subprocess moves to the WAITING state.

Receive CONFIG-REJ

If the UMTCLT.indication primitive is asserted and contains a CONFIG-REJ, the Configuration Initiator subprocess will compare the value of retry_counter to the value of max_retries. If retry_timer is less than max_retries, then the Configuration Initiator subprocess moves to the TX_CONFIG_REQ state where the Configuration Initiator subprocess shall adjust the parameters, fields, and TLVs to be sent in the CONFIG-REQ in a way to achieve agreement with the remote peer's configuration as sent in the CONFIG-REJ. If retry_timer equals or exceeds max_retries then the Configuration Initiator subprocess moves to the WAITING state.

Receive CFG-RSP

If the UMTCLT.indication primitive is asserted and contains a CONFIG-RSP, indicating that the remote peer agrees with the configuration sent in the CONFIG_REQ message, the Configuration Initiator subprocess will move to the TX_CONFIG_ACK state.

#### B.4.3.1.4 TX_CONFIG_ACK State

When the Configuration Initiator subprocess enters the TX_CONFIG_ACK state, the Configuration Initiator subprocess assert the UMTCLT.request primitive with the parameters required to send a CONFIG-ACK message. The Configuration Initiator subprocess will then call the create_tunnel_adapter function with the tunnel descriptor and UMT subtypes specified in the CONFIG-REQ message.

### B.4.3.2 Configuration Init Receiver

A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration process is implemented, it shall implement the Configuration Init Receiver state diagram shown in Figure B-8.

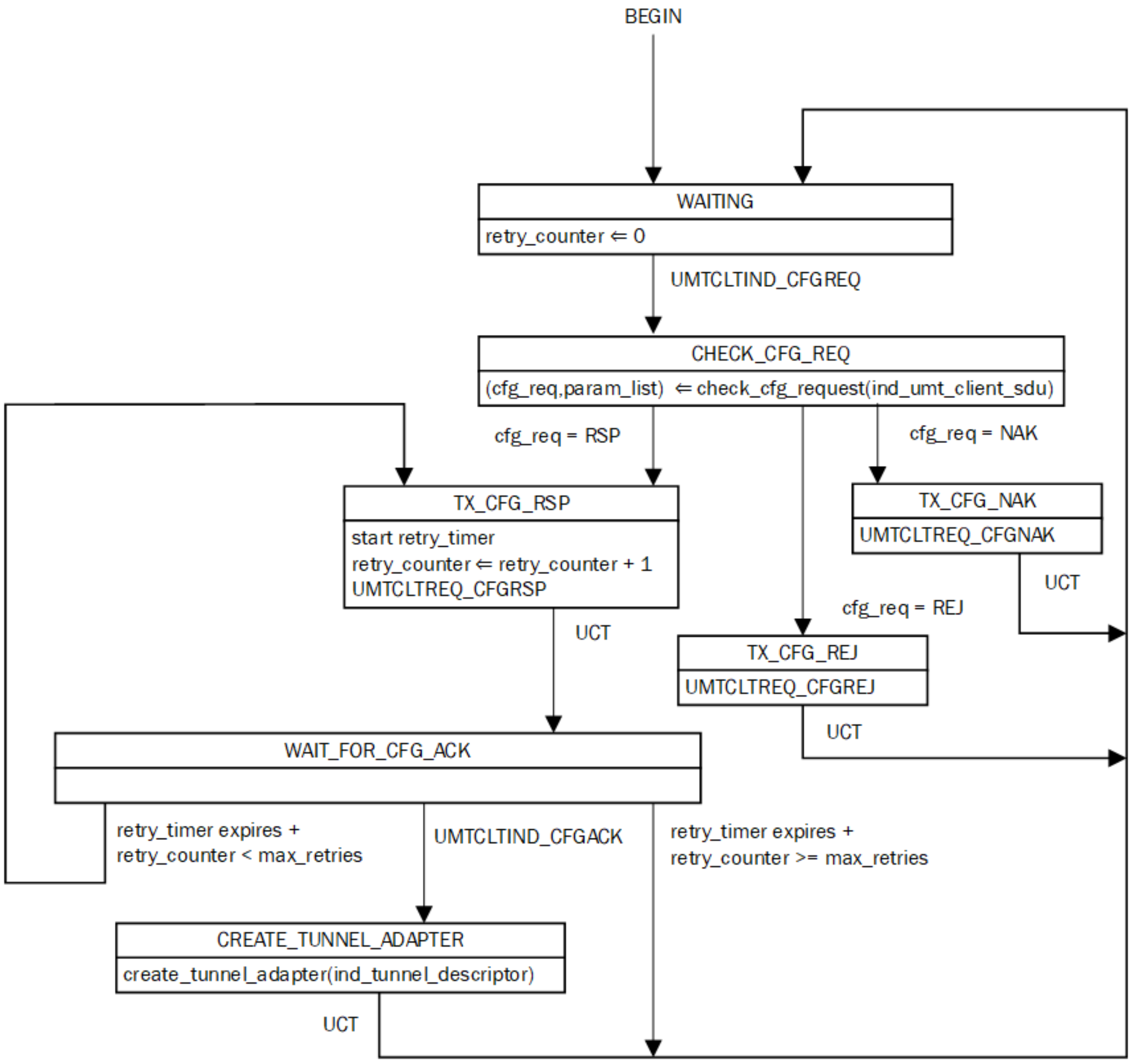


**Figure B-8 - Configuration Init Receiver State Diagram**

### B.4.3.2.1 WAITING State

Upon initialization, the WAITING state is entered. In the WAITING state, the Configuration Init Receiver subprocess sets the retry_counter to zero and waits for assertion of the UMTCLT.indication primitive with the umt_client_sdu containing a CONFIG-REQ message.

### B.4.3.2.2 CHECK_CFG_REQ State

Upon entering the CHECK_CFG_REQ state, the Configuration Init Receiver subprocess calls the check_cfg_request function to check if the received CONFIG-REQ fields, paramaters, TLVs and values are acceptable.

If the check_cfg_request returns a cfg_req indicating the request is acceptable (cfg_req=RSP), then the Configuration Init Receiver subprocess moves to the TX_CFG_RSP state. If the check_cfg_request returns a cfg_req indicating the request contains fields, parameters or TLVs that are unacceptable (cfg_req=REJ), then the Configuration Init Receiver subprocess moves to the TX_CFG_REJ state. If the check_cfg_request returns a cfg_req indicating the request indicating that the values of the fields, parameters or TLVs are unacceptable (cfg_req=NAK), then the Configuration Init Receiver subprocess moves to the TX_CFG_NAK state.

#### B.4.3.2.3 TX_CFG_RSP State

In the TX_CFG_RSP state the the Configuration Init Receiver subprocess starts retry_timer and increments retry_counter. The Configuration Init Receiver subprocess copies ind_requested_umt_subtypes_tlv into req_requested_umt_subtypes_tlv, ind_tunnel_adapter_descriptor_tlv into req_tunnel_adapter_descriptor, and ind_transaction_id_tlv into req_tranaction_id_tlv and then asserts the UMTCLT.request service primitive with the parameters required to send a CONFIG-RSP message.

#### B.4.3.2.4 TX_CFG_NAK State

#### B.4.3.2.5 TX_CFG_REJ State

#### B.4.3.2.6 WAIT_FOR_CFG_ACK State

In the WAIT_FOR_CFG_ACK state, the Configuration Init Receiver subprocess waits for assertion of the UMTCLT.indication primitive with the umt_client_sdu containing a CONFIG-ACK message. If retry_timer expires before a CONFIG-ACK is received

#### B.4.3.2.7 CREATE_TUNNEL_ADAPTER State

### B.4.3.3 Delete Initiator

A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration process is implemented, it shall implement the Delete Initiator state diagram shown in Figure B-9.

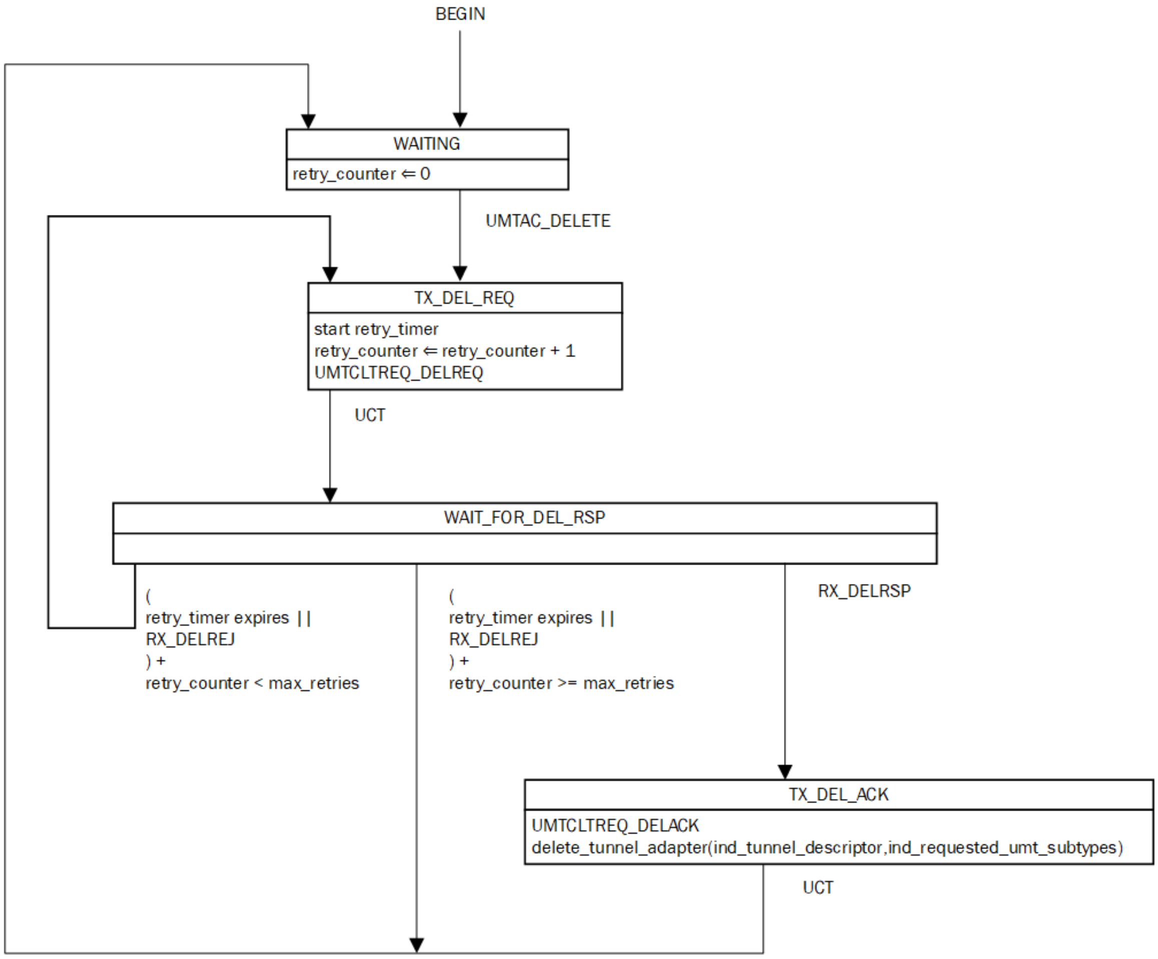


**Figure B-9 - Delete Initiator State Diagram**

#### B.4.3.3.1 WAITING State

#### B.4.3.3.2 TX_DEL_REQ State

#### B.4.3.3.3 WAIT_FOR_DEL_RSP State

#### B.4.3.3.4 TX_DEL_ACK State

### B.4.3.4 Delete Receiver

A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration process is implemented, it shall implement the Delete Receiver state diagram shown in Figure B-10

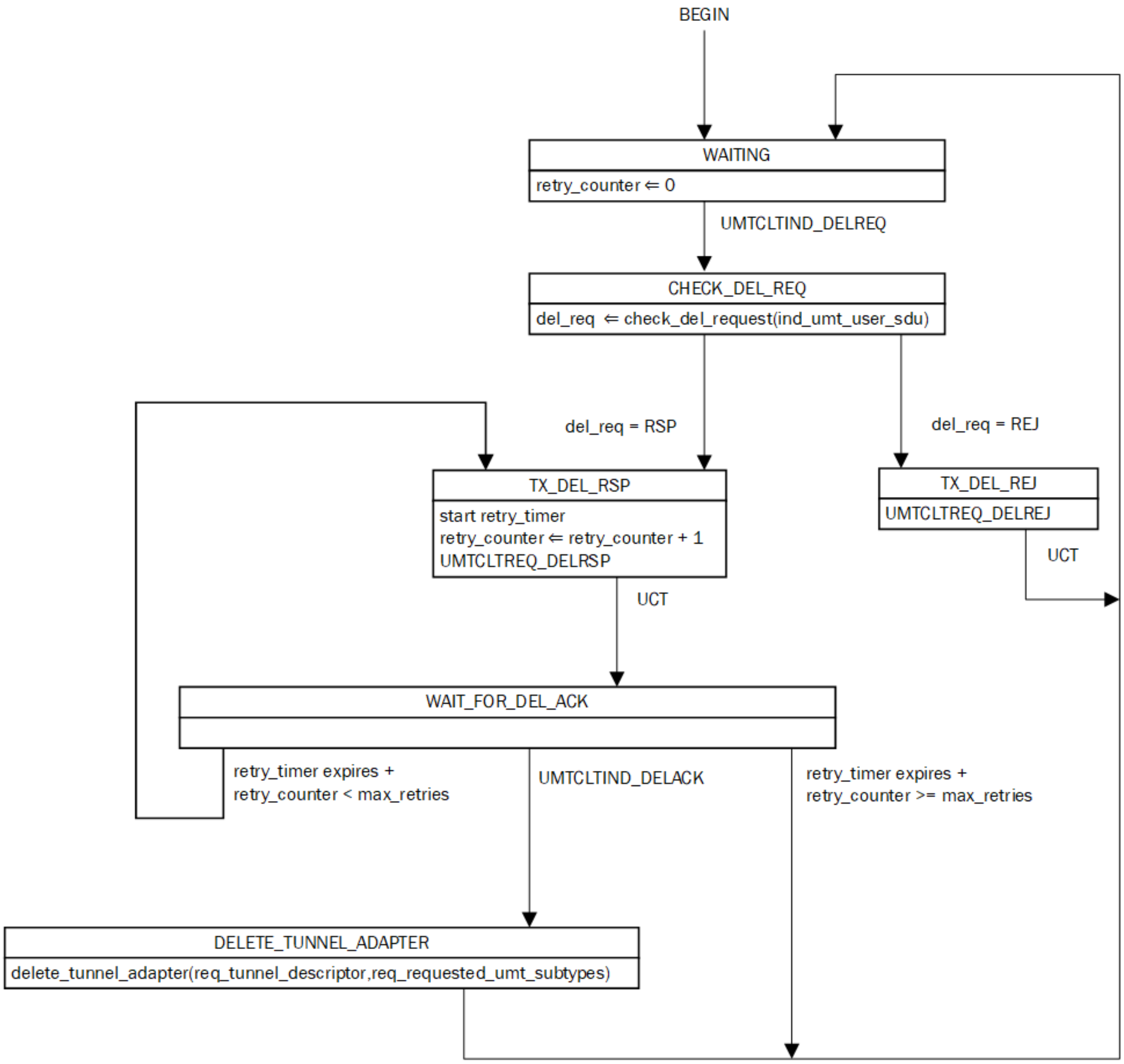


**Figure B-10 - Delete Receiver State Diagram**

#### B.4.3.4.1 WAITING State

#### B.4.3.4.2 CHECK_DEL_REQ State

#### B.4.3.4.3 TX_DEL_RSP State

#### B.4.3.4.4 TX_DEL_REJ State

#### B.4.3.4.5 WAIT_FOR_DEL_ACK State

#### B.4.3.4.6 DELETE_TUNNEL_ADAPTER State

## B.5 UMT Peer Maintenance SDU Format

UMT Peer Maintenance SDUs are encapsulated in UMTPDUs under the UMT Peer Maintenance subtype (See IEEE Std. 1904.2 Table 4-2).

UMT Peer Maintenance SDUs may be fragmented and span multiple UMTPDUs. It is up to the UMT Client to manage SDU fragmentation and reassembly. The Sequence Number field is present in the UMT Peer Maintenance SDU to aid the UMT Client in managing the fragmentation process.

### B.5.1 Structure

The UMT Peer Maintenance PDU structure shall be as shown in Figure B-11.

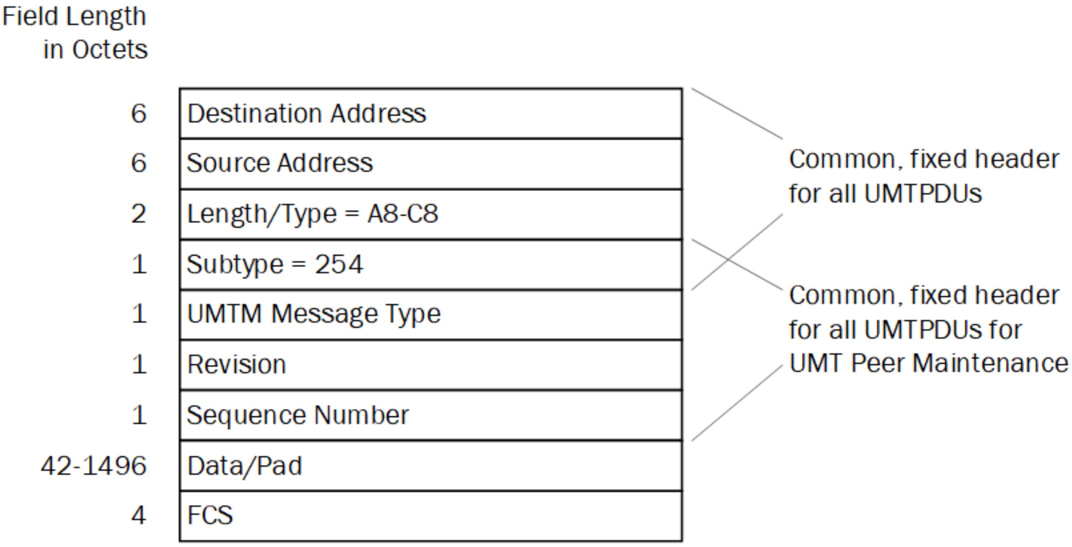


**Figure B-11 - UMT Peer Maintenance PDU Structure**

UMT Peer Maintenance PDUs shall have the following fields

a) ***Destination Address*** (DA). This is the Destination Address field. Its use in the context of UMT is specified in IEEE Std 1904.2 Clause 4.

b) ***Source Address*** (SA). This is the Source Address field. Its use in the context of UMT is specified in IEEE Std 1904.2 Clause 4.

c) ***Length/Type***. This is the Length/Type field. Its use in the context of UMT is specified in IEEE Std 1904.2 Clause 4.

d) ***Subtype***. The Subtype field identifies the specific UMT Client layer being encapsulated. For UMT Peer Management PDUs, the Subtype field value carries the UMT Maintenance/Peer Management value as specified in IEEE Std. 1904.2 Table 4-2.

e) ***UMTM Message Type***. The UMTM Message Type field specifies the UMT Peer Maintenance Message Type. Valid values for the Message Type field are specified in Table B-1.

f) ***Revision***. The Revision field contains the revision number of the configuration contained in the UMT Peer Maintenance Message PDU. The revision number begins at 1 and increments each time a change occurs in the content of the data being transmitted in the UMT Peer Maintenance Message PDU. A change causing an increment of the revision can be a change in the value of a field or TLV, or the addition or deletion of a TLV.

g) ***Sequence Number***. The Sequence Number field provides a method to signal to the receiving UMT Maintenance peer that the UMT Peer Maintenance Message spans multiple UMTPDUs.

h) ***Data***. This field contains one or more UMT Peer Maintenance TLVs. Valid UMT Peer Maintenance TLVs are specified in B.5.2.

i) ***FCS***. This field is the Frame Check Sequence, as defined in IEEE Std. 802.3.

**Table B-1 - UMT Maintenance Message Types**

| UMT Maintenace Message Type | Message Name |
|---|---|
| 0 | Reserved |
| 1 | SOLICIT |
| 2 | HELLO |
| 3 | CONFIG-REQ |
| 4 | CONFIG-ACK |
| 5 | CONFIG-NAK |
| 6 | CONFIG-REJ |
| 7 | DELETE-REQ |
| 8 | DELETE-ACK |
| 9 | DELETE-REJ |
| 10-252 | Unassigned |
| 253 | Vendor-Specific |
| 254 | Unassigned |
| 255 | Reserved |

### B.5.2 UMT Peer Maintenance TLVs

### B.5.3 Encodings for UMT Maintenance TLVs

The following type/length/value encodings are used in UMT Maintenance messages.

#### B.5.3.1 Supported UMT SubTypes

This field describes the list of UMT SubTypes (Table 4-2) that are supported by the UMT peer. This list is structured as a series of 1-octet values. Each supported type is represented by its corresponding value found in Table 4-1. The Length parameter indicates the number of 1-octet values contained in the field.

| Type | Length | Value |
|---|---|---|
| 1 | n | List of UMT Subtypes supported by the UMT peer (values from Table 4-1) |

#### B.5.3.2 Requested UMT Subtypes

This field describes the list of UMT Subtypes (Table 4-1) being requested by the UMT peer for use on a UMT tunnel adapter. This list is structured as a series of 1-octet values. Each supported type is represented by its corresponding value found in Table 4-1. The Length parameter indicates the number of 1-octet values contained in the field.

| Type | Length | Value |
|---|---|---|
| 2 | n | List of UMT Subtypes supported by the UMT peer (values from Table 4-1) |

#### B.5.3.3 Transaction Identifier

The value of this TLV contains a 4-octet random number generated by the UMT Peer sending a CONFIG-REQ or DELETE-REQ. The transaction identifier is used by the requestor and requested UMT peers to correlate the messages sent between the two UMT peers.

| Type | Length | Value |
|---|---|---|
| 3 | 4 | 4-octet random number |

#### B.5.3.4 Tunnel Adapter Descriptor

This field describes the characteristics of a tunnel adapter. It is formatted as a set of encapsulated sub-TLVs. When used in a UMT Peer Maintenance SDU, the Tunnel Adapter Descriptor shall contain one and no more than one instance of each of the sub-TLVs defined in this subclause.

| Type | Length | Value |
|---|---|---|
| 4 | n | Encapsulated sub-TLVs |

##### B.5.3.4.1 Tunnel Adapter Transmission Method Subtype

This field specifies the tunnel adapter type. Valid values are Broadcast, Multicast, or Unicast.

| Type | Length | Value |
|---|---|---|
| 1 | 1 | 1 – Broadcast<br>2 – Unicast<br>3 – Multicast<br>4 – Receive Only |

##### B.5.3.4.2 Tunnel Adapter Indicated Source Address Subtype

This field specifies the Source Address of incoming UMTPDUs to be associated with the local tunnel adapter. This is the MAC Source Address that the local tunnel adapter expects in a received UMTPDU.

| Type | Length | Value |
|---|---|---|
| 2 | 6 | 48-bit MAC address |

##### B.5.3.4.3 Tunnel Adapter Indicated Destination Address Subtype

This field specifies the Destination Address of transmitted UMTPDUs to be associated with the local tunnel adapter. This is the MAC Destination Address that the local tunnel adapter expects in a received UMTPDU.

| Type | Length | Value |
|---|---|---|
| 3 | 6 | 48-bit MAC address |

##### B.5.3.4.4 Tunnel Adapter Requested Source Address Subtype

This field specifies the Source Address of transmitted UMTPDUs to be associated with the remote tunnel adapter. This is the MAC Source Address that the remote tunnel adapter expects in a received UMTPDU.

| Type | Length | Value |
|---|---|---|
| 4 | 6 | 48-bit MAC address |

#### B.5.3.4.5 Tunnel Adapter Requested Destination Address Subtype

This field specifies the Destination Address of transmitted UMTPDUs to be associated with the remote tunnel adapter. This is the MAC Destination Address that the remote tunnel adapter expects in a received UMTPDU.

| Type | Length | Value |
|---|---|---|
| 5 | 6 | 48-bit MAC address |

### B.5.3.5 UMT Peer Identifier

This field contains the 48-bit MAC address of the UMT peer that is sending the message.

| Type | Length | Value |
|---|---|---|
| 6 | 6 | 48-bit MAC address |

### B.5.3.6 Reason Code

This field contains a reason code encoded as an n-octet integer. The reason code indicates to a receiving entity the reason for an error associated with the parameter negotiation. Multiple instances of this field may be present in a UMT Maintenance SDU.

| Type | Length | Value |
|---|---|---|
| 7 | n | Reason Code (see Table B-2) |

**Table B-2 - Reason Codes**

| Code | Reason |
|---|---|
| 0 | Reserved. Do Not Use |
| 1 | No Tunnel Adapter Matches Requested Tunnel Adaptor Descriptor |
| 2 | Requested SubType does not exist on Requested Descriptor |
| 3 | Unsupported SubType |
| 4 | Unsupported Tunnel Adapter Descriptor |
| 5 | |
| | |

### B.5.3.7 Vendor-Specific Extension

The Vendor-Specific extension field may be used to extend the capabilities of a specific implementation of the UMT Peer Maintenance SDU. The format of this TLV is implementation-specific, but it is recommended that it be formatted as an encapsulated set of subTLVs.

| Type | Length | Value |
|---|---|---|
| 253 | n | Unspecified |

with the Subtype set to UMT Mainteance (see IEEE Std. 1904.2 Table 4-2) and requests that the UMT Peer transmit the PDU (referred to as a UMT Maintenance PDU) as a MAC broadcast.

A UMT Peer Discovery entity operating in Passive or Active mode that receives the broadcast UMT Maintenance PDU (UMTMPDU)

Upon initialization, a UMT Peer Discovery entity in Active mode configures the local UMT peer with a UMT Tunnel Adapter configured for broadcast operation in the transmit direction (req_SA=local MAC address, req_DA=MAC broadcast) and unicast operation in the receive direction (ind_SA=any MAC address, ind_DA=local MAC address). This Tunnel Adapter is the *Active Peer Discovery Tunnel Adapter*.

A UMT Peer Discovery entity operating in Passive or Active mode configures a UMT Tunnel Adapter for (req_SA=local MAC address, req_DA=MAC broadcast, ind_SA=any, ind_DA=MAC broadcast). This second UMT Tunnel Adapter is called the *Passive Peer Discovery Tunnel Adapter* and is never used by the UMT Peer Discovery entity to transmit a UMTPDU.

The Active UMT Peer Discovery entity generates an UMTPDU with the Subtype set to UMT Maintenance. This UMTPDU will be called a UMT Maintenace SDU (UMTMSDU). The UMT Peer Discovery entity requests that the UMTMSDU be transmitted through the Active Peer Discovery Tunnel Adapter.

Upon receipt of a broadcast UMTPDU containing a UMTMSDU, a UMT peer will deliver the UMTMSDU to the UMT Peer Discovery entity via the Passive Peer Discovery Tunnel Adapter, if the entity exists on the local UMT peer.

The receiving UMT Peer Discovery entity will determine whether to respond based on local policy configured by the administrator. If local policy allows it, the UMT Peer Discovery entity will

configure a new UMT Tunnel Adapter for unicast operation (req_SA=local MAC address, req_DA=SA from received UMTMSDU, ind_SA= SA from received UMTMSDU, ind_DA=local MAC address). The UMT Peer Discovery entity then forms a response UMTMSDU and transmits it through this newly configured UMT Tunnel Adapter.

Upon receipt of the response, the Active UMT Peer Discovery entity, if local policy allows it, configures a new UMT Tunnel Adapter for unicast operation (req_SA=local MAC address, req_DA=SA from received UMTMSDU, ind_SA=SA from received UMTMSDU, ind_DA=local MAC address). The Active UMT Peer Discovery entity then forms an UMTMSDU acknowledging the response, and sends it on the newly formed UMT Tunnel Adapter. The two UMT Peer Discovery Entities continue the exchange of UPDPSDUs until agreement is reached on the tunnel operational parameters or until one or both of the UMT Peer Discovery Entities give up.