1 Annex B UMT Peer Discovery and Tunnel Auto-Configuration

2 B.1 Introduction

3 IEEE Std. 1904.2 Clause 4 defines a method for delivering service data units (SDU) for higher layer 4 protocols across a layer-2 network in which those protocols would not normally be forwarded due to 5 addressing conflicts or other factors. The described architecture consists of UMT peers that perform 6 appropriate encapsulation of the UMT Client SDUs into UMTPDUs which are transmitted across a layer-2 7 network and received, decapsulated and the resulting UMT Client SDUs delivered to the desired UMT 8 Client.

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

12 UMT Peers on a network.

13 This annex defines an architecture and system for automatic UMT Peer Discovery and for automatically 14 configuring unicast tunnels between peers.

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

16 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.



19

20

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



Periodically, a UMT Peer Discovery entity in Active mode forms a UMTPDU and transmits it as a MAC

4 5 broadcast. The broadcast UMTPDU solicits neighboring UMT Peer Discovery entities (Active or Passive)

6 to respond to the sending UMT Peer Discovery entity.

7 Upon receipt of the UMT Peer Discovery solicitation message, the receiving UMT Peer Discovery entity 8 will, if local policy permits, form a UMTPDU and send it as a MAC unicast to the soliciting peer. This

9 solicitation/response action allows the Active UMT Peer Discovery entity to build a table of neighboring

10 UMT Peers and the capabilities of each.

2 3

11 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).

10 **B.3 Functional Specifications**

11 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.



- 5 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.
- 8 c) Automatic UMT Peer Discovery is only responsible for building a database of neighboring UMT
 9 peers.
- 10 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.

17 B.3.3 UMT Peer Maintenance

1 2

3

4

18 The UMT Peer Maintenance entity is a multi-functional and extensible entity. In the context of this annex, 19 the UMT Peer Maintenance entity is the context in which UMT Peer Discovery and UMT Tunnel Auto-20 Configuration is defined.

1 B.3.3.1 UMT Peer Maintenance Interactions

2 **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:
- 5 UMTCLT.request
- 6 UMTCLT.indication
- The UMTCLT.request and UMTCLT.indication service primitives are described in IEEE Std. 1904.2
 Clause 4.

9 **B.3.3.1.2** Intralayer Interactions

10 The UMT Peer Maintenance entity contains an abstract control process that communicates with the Auto-11 Configuration function using the following service interfaces:

12 UMTAC.request

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

15 **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 aUMT Peer.

18 **B.3.3.1.2.1.1 Function**

- 19 The semantics of the primitive are as follows:
- 20 UMTAC. request (

21	action,
22	tunnel_adapter_descriptor
23)

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.

26 **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.

30 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:

34 (
 35 Indicated DA = <Local UMT Peer MAC Address> (DA of UMTPDU.indication)

1	Indicated SA = <any></any>	(SA of UMTPDU.indicati	on)
2	Requested $DA = MAC Br$	oadcast	(DA of UMTPDU.request)
3	Requested SA = <local th="" u<=""><th>MT Peer MAC Address></th><th>(SA of UMTPDU.request)</th></local>	MT Peer MAC Address>	(SA of UMTPDU.request)
4	Transmission Method = B	broadcast	
5)		

6 The Active Peer Discovery Tunnel Adapter shall be configured prior to or during initialization of the 7 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.

13 B.3.3.4 Passive Peer Discovery Receive Tunnel Adapter

14 The Passive Peer Discovery Receive Tunnel Adapter is used by the Passive Peer Discovery process to 15 receive SOLICIT messages via MAC broadcast from any possible MAC source address. It shall not be used 16 to transmit messages. The Passive Peer Discovery Receive Tunnel Adapter is a receive-only tunnel adapter. 17 The Passive Peer Discovery Receive Tunnel Adapter is defined by the tuple:

18	(
19	Indicated DA = MAC Broadcast	(DA of UMTPDU.indication)
20	Indicated SA = <any></any>	(SA of UMTPDU.indication)
21	Requested $DA = \langle N/A \rangle$	(DA of UMTPDU.request)
22	Requested $SA = \langle N/A \rangle$	(SA of UMTPDU.request)
23	Transmission Method = Receive Only	
24)	

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

27 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:

33	(
34	Indicated $DA = \langle N/A \rangle$ (DA of UMTPDU.indication)	
35	Indicated $SA = \langle N/A \rangle$ (SA of UMTPDU.indication)	
36	Requested DA = <remote address="" mac="" peer="" umt=""> (DA of UMTPDU.request)</remote>	
37	Requested SA = <local address="" mac="" peer="" umt=""> (SA of UMTPDU.request)</local>	
38	Transmission Method = Unicast	
39)	

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.

1 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:

8	(
9	Indicated DA = <local address="" mac="" peer="" umt=""></local>
10	Indicated SA = <remote address="" mac="" peer="" umt=""></remote>
11	Requested DA = <remote address="" mac="" peer="" umt=""></remote>
12	Requested SA = <local address="" mac="" peer="" umt=""></local>
13	Transmission Method = Unicast
14	

15 The AutoConfig Tunnel Adapter shall be configured prior to or during initialization of the Auto-16 Configuration process.

17 **B.4 Detailed functions and state diagrams**

18 **B.4.1 State diagram variables**

19 B.4.1.1 Constants

- 20 UMTM Subtype
- 21 The value of the UMT Subtype field for UMT Maintenance SDUs (See Table 4-2).
- 22 ta_unicast_mode
- The value of the Tunnel Adapter Transmission Method that indicates unicast transmission mode.
 (See B.5.3.4.1)

25 ta_rxonly_mode

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

28 NULL

29

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

30 B.4.1.2 Variables

31 BEGIN

32	A variable that resets the functions within a UMT Peer Maintenance process.
33	Values: TRUE; when any of the component UMT sublayers is reset.
34	FALSE; When (re-)initialization has completed.
35	

36 req_umt_subtype

- The value of the umt_subtype parameter passed to the UMT Client in the UMTCLT.request primitive.
- 39Value: Integer (See Table 4-2)
- 40 req_umtm_message_type

1 2 3 4 5	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
6	real revision
0 7	The value of the Revision field in a requested UMT Peer Maintenance SDU and passed to the
8	UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req umt client sdu
9	parameter.
10	Values: See B.5.1
11	
12	reg sequence number
13	The value of the Sequence Number field in a requested UMT Peer Maintenance SDU and passed
14	to the UMT Tunnel Adapter via the UMTCLT request primitive as part of the request solution of the requ
15	narameter
16	Values: See B 5 1
17	
18	rea supported unt subtypes the
19	The value of the Supported LIMT Subtypes TLV in a requested LIMT Peer Maintenance SDU and
20	nassed to the IIMT Tunnel Adapter via the IIMTCLT request primitive as part of the
20	requint client sdu parameter
21	Values: See B 5 3 1
22	values. See D.J.J.1
23	rad requested unit subtypes the
2 4 25	The value of the Dequested LIMT Subtypes TLV in a requested LIMT Deer Mointenance SDU and
25 26	new and of the LIMT Tunnel. Adapter via the LIMTCLT request primitive as part of the
20	rag unt alignt adu personator
21	Valuar See D 5 2 2
20	values: see B.5.5.2
29	
30 21	requinipeer identifier itv
22	The value of the UMT Turnel. Adapter significant the UMTCLT respect of the second to the UMT Turnel.
32 22	passed to the UNIT runner Adapter via the UNITELT.request primitive as part of the
33	req_umt_client_sdu parameter.
34 25	values: See B.5.5.5
35	
30	req_transaction_id_tiv
3/	The value of the Transaction Identifier ILV in a requested UNIT Peer Maintenance SDU and
38	passed to the UMI lunnel Adapter via the UMICLI request primitive as part of the
39	req_umt_client_sdu parameter.
40	Values: See B.5.3.3
41	
42	req_tunnel_adapter_descriptor_tiv
43	The value of the Tunnel Adapter Descriptor TLV in a requested UMT Peer Maintenance SDU and
44	passed to the UMT Tunnel Adapter via the UMTCLT.request primitive as part of the
45	req_umt_client_sdu parameter.
46	Values: See B.5.3.4
47	
48	req_reason_code
49	The value of the Reason Code TLV in a requested UMT Peer Maintenance SDU and passed to the
50	UMT Tunnel Adapter via the UMTCLT.request primitive as part of the req_umt_client_sdu
51	parameter.
52	Values: See B.5.3.6
53	
54	req_umt_client_sdu
55	The value of the umt_client_sdu parameter passed to the UMT Tunnel Adapter in the
56	UMTCLT.request primitive.

1	ind_SA	
2		The value of the source address parameter received in a UMTCLT.indication primitive.
3		
4	ind DA	
5	_	The value of the destination address parameter received in a UMTCLT indication primitive.
6		
7	ind umt	subtra
0	Ind_unit	_subuye The value of the Subture field in a received UMT metagel frome (see Table 4.2) and is used to
0		The value of the subtype field in a feetived Own protocol frame (see Fable 4-2) and is used to
9		determine the UM I Client to which the UM I payload is destined.
10		Value: Integer (See Table 4-2)
11		
12	ind_umt	m_message_type
13		The value of the UMTM Message Type field in a received UMT Peer Maintenance SDU and
14		passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the
15		ind umt client sdu parameter.
16		Values: See Table B-1
17		
18	ind revi	sion
10		The value of the Devision field in a received UMT Deer Maintenance SDU and passed to the UMT.
20		The value of the Keylston herd in a received of the certaintenance shot and passed to the OMT
20		Peer Maintenance entry via the OMTCLT.indication primitive as part of the ind_unit_chent_sdu
21		parameter.
22		Values: See B.5.1
23		
24	ind_sequ	uence_number
25		The value of the Sequence Number field in a received UMT Peer Maintenance SDU and passed to
26		the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the
27		ind umt client sdu parameter.
28		Values: See B.5.1
29		
30	ind sum	norted unit subtypes the
31	ina_sapi	The value of the Supported UMT Subtypes TLV in a received UMT Deer Maintenance SDU and
22		needed to the UMT Door Mointenance antity vie the UMTCLT indication primitive as part of the
32 22		passed to the OWI Feel Maintenance entry via the OWICLI.indication primitive as part of the
33		ind umi chent sou parameter.
34		Values: See B.5.3.1
35		
36	ind_requ	uested_umt_subtypes_tlv
37		The value of the Requested UMT Subtypes TLV in a received UMT Peer Maintenance SDU and
38		passed to the UMT Peer Maintenance entity via the UMTCLT.indication primitive as part of the
39		ind umt client sdu parameter.
40		Values: See B.5.3.2
41		
42	ind umt	neer identifier tly
/3	ina_uint	
43		to the LIMT Door Neistenenge origin the LIMTCH request primitive as part of the
44		to the OMT real mannehance entry via the OMTCLT equest primitive as part of the
45		ind umt client sou parameter.
46		Values: See B.5.3.5
47		
48	ind_tran	saction_id_tlv
49		The value of the Transaction Identifier TLV in a received UMT Peer Maintenance SDU and
50		passed to the UMT Peer Maintenance entity via the UMTCLT.request primitive as part of the
51		ind umt client sdu parameter.
52		Values: See B.5.3.3
53		
54	ind turr	nel adapter descriptor tly
. .		rr

$\frac{1}{2}$	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
3	ind umt client sdu parameter
4	Values: See B 5.3.4
5	Values. See D.5.5.4
6	ind reason code
7	The value of the Desson Code TLV in a received UMT Deer Maintenance SDU and passed to the
8	LIMT Deer Maintenance entity via the LIMTCLT request primitive as part of the
0	ind unt client sdu neremeter
9	Maluni Chen Sau parameter.
10	values: See D.3.5.0
11	in Lange of Land and La
12	ind_umt_client_sau
13 14	Maintenance entity in the umt_client_sdu paremeter of the UMTPDU.indication primitive.
15	req_action
16	req_tunnel_adapter_descriptor
17 18	The parameters of the UMTAC.request service primitive as defined in B.3.3.1.2.1
19	reg umtm data
20	The fields contained in a UMT Peer Maintenance SDU and passed to the UMT Tunnel Adapter in
21	the UMTCLT.request service primitive.
	1 1
22	reg action create
$\frac{-}{23}$	The action parameter of UMTAC request, as defined in B.3.3.1.2.1, with a value indicating a
24	create action.
25	reg action delete
26	The action parameter of LIMTAC request as defined in B 3 3 1 2 1 with a value indicating a
$\frac{20}{27}$	delete action.
_ /	
28	rea tunnel adapter descriptor
29	The value of the tunnel adapter descriptor parameter of the UMTAC request service primitive as
30	defined in B 3 3 1 2 1
50	
31	max retries
32	This variable defines the maximum number of times a LIMT Peer Maintenance process will send a
32	duplicate message in an attempt to communicate with a neer entity
55	duplicate message in an attempt to communicate with a peer entity.
3/	naram list
25	param_nst
35	The values returned from the check of a request function
50	The values returned from the check_crg_request function.
27	dal rag
20	The velve returned from the check del request function
30	The value returned from the check_del_request function.
20	tunnal descriptor
39 40	This variable corresponds the correspondence that define a typical adapter on a LIMT many The
40 41	number and the define a tunnel adapter are specified by the Type Adapter Descriptor
41	TIV defined in D.5.2.4. These permeters are specified by the state discrete and the state d
42 42	TL v defined in D.J.J.4. Those paramiters are represented in the state diagrams as:
43 14	to indicated dou Tunnal Adaptan Indicated Destination Address Subtrass (See D.5.2.4.2)
44	ta indicated da: Tunnel Adapter Indicated Destination Address Subtype (See B.5.3.4.3)
45	ta_indicated_sa: Tunnet Adapter indicated Source Address Subtype (See B.5.3.4.2)
46	ta_requested_da: Tunnel Adapter Requested Destination Address Subtype (See B.5.3.4.5)
47	ta_requested_sa: Tunnel Adapter Requested Source Address Subtype (See B.5.3.4.4)
48	ia_ix_method: Tunnel Adapter Transmission Method Subtype (See B.5.3.4.1)

3

4

6

- 2 requested umt subtypes
 - This variable represents the value contained in the Requested UMT Subtypes TLV (See B.5.3.2)
- 5 indicated umt mac address
 - The \overline{MAC} address of the local UMT peer.

7 **B.4.1.3 Counters**

- 8 retry counter
- 9 A counter used to limit the number of duplicate UMT Maintenance SDUs sent during a Peer 10 Discovery or Auto-Configuration negotiation.

11 **B.4.1.4 Timers**

- 12 discovery_tx_timer
- 13 Timer used to regulate the frequency that peer discovery SOLICIT messages are sent.

14 retry_timer

15 Timer used to regulate the frequency that auto-configuration SDUs are sent when no response is 16 received to a corresponding request.

17 **B.4.1.5 Functions**

- 18 save_peer_info(source_address, umt_client_sdu)
- 19 This function parses received SOLICIT and HELLO messages and saves the received data for use 20 by other processes in the UMT Maintenance entity (e.g. Auto-Configuration). This function 21 requires as its arguments, the source address parameter and a UMT Client Service Data Unit as 22 received via the UMTCLT.indication primitive.
- 23 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.
- 28 delete_tunnel_adapter(tunnel_descriptor, requested_umt_subtypes)
- 29This function removes access to the tunnel adapter indicated by tunnel_descriptor for the UMT30clients indicated by requested_umt_subtypes and deletes the tunnel adapter from the local UMT31peer if there are no remaining clients. The function requires a tunnel descriptor and list of UMT32Subtypes (see Table 4-2) as its arguments.
- 33 $(cfg_req, param_list) \leftarrow check_cfg_request(umt_client_sdu)$
- 34 This function parses received CONFIG-REQ messages and returns a value, in the cfg req variable, 35 indicating the status of the CONFIG-REQ. This function requires as its only argument, a UMT 36 Client Service Data Unit as received via the UMTCLT.indication primitive. A return value of 37 ACK indicates that the request is acceptable. A return value of REJ indicates that the request 38 contains unacceptable fields or TLVs. A return value of NAK indicates that the request contains 39 acceptable fields and TLVs but the value of of one or more of the fields or TLVs is unacceptable. 40 If this function returns NAK, it will also return a list of the fields, in the param list variable, and 41 TLVs containing unacceptable values along with values for each that are acceptable to the local

1	peer and a reason code to indicate the reason the request is unacceptable. If this function returns
2	REJ, it will also return a list of the unacceptable fields and TLVs along with values for each that
3	are acceptable to the local peer and a reason code to indicate the reason the request is unacceptable.

4 $del_req \leftarrow check_del_request(umt_client_sdu)$

5 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 6 7 acceptable. This function requires as its only argument, a UMT Client Service Data Unit as 8 received via the UMTCLT.indication primitive. A return value of REJ indicates that the message 9 contains unacceptable fields or TLVs. A return value of NAK indicates that the message contains 10 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 11 12 unacceptable. If this function returns REJ, it will also return a reason code to indicate the reason 13 the request is unacceptable. This function shall not return a list of unacceptable or acceptable 14 fields, TLVs or values.

- 15 valueof(tlv)
- 16 This function returns the value contained in a TLV.

17 B.4.1.6 Messages

- 18 UMTCLTREQ SOLICIT
- 19 Alias for the request for a peer discovery SOLICIT message to be sent via the UMTCLT.request 20 primitive. The requested SOLICIT message contains the following fields, parameters and values:

 \Leftarrow SOLICIT (see Table B-1)

- 21 req_umt_subtype
- 22 req umtm message type
- 23 req_revision
- 24 req_sequence_number
- 25 req_supported_umt_subtypes_tlv
- 26 req umt peer identifier tlv

28 UMTCLTIND HELLO

27

- 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:
- 31 ind umt subtype 32 ind umtm message type \leftarrow HELLO (see Table B-1) 33 ind revision 34 ind sequence number 35 ind supported umt subtypes tlv 36 ind umt peer identifier tlv 37 38 UMTCLTIND SOLICIT 39 Alias for the receipt of a peer discovery SOLICIT message via the UMTCLT.indication primitive. 40 The received SOLICIT message contains the following fields, parameters and values: 41 ind umt subtype 42 ind umtm message type \Leftarrow SOLICIT (see Table B-1) 43 ind revision 44 ind sequence number ind supported umt subtypes tlv 45 46 ind umt peer identifier tlv

1 2 3 4	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:
5	req_umt_subtype
6	req_umtm_message_type \Leftarrow HELLO (see Table B-1)
7	req_revision
8	req_sequence_number
9	req_supported_umt_subtypes_tlv
10	req_umt_peer_identifier_tlv
12	UMTAC_CREATE
13	Alias for UMTAC.request(req_action, req_tunnel_adapter_descriptor), where req_action contains
14	the value indicating a create action.
15	UMTAC_DELETE
16	Alias for UMTAC.request(req_action, req_tunnel_adapter_descriptor), where req_action contains
17	the value indicating a delete action.
18	UMTCLTREQ_CFGREQ
19	Alias for the request for a Auto-Configuration CONFIG-REQ message to be sent via the
20	UMTCLT.request primitive. The requested CONFIG-REQ message contains the following fields,
21	parameters and values:
22 23 24 25 26 27 28 29 20	req_umt_subtype req_umt_message_type \Leftarrow 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
30 31 32 33 34	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:
35 36 37 38 39 40 41 42 43 44	<pre>ind_umt_subtype ind_umtm_message_type</pre>

1	RX CFGREJ
2	Alias for the receipt of an Auto-Configuration CONFIG-REJ message via the
3	UMTCLT.indication primitive. The received CONFIG-REJ message contains the following fields,
4	parameters and values:
5	ind_umt_subtype
6	ind_umtm_message_type \Leftarrow CONFIG-REJ (see Table B-1)
7	ind_revision
8	ind_sequence_number
9	ind_transaction_id_tlv
10	ind_umt_peer_identifier_tlv
11	ind_requested_umt_subtypes_tlv (optional per B.4.3.2.5)
12	ind_tunnel_adapter_descriptor_tlv (optional per B.4.3.2.5)
13	ind_reason_code
14	DV CECDOD
15	RX_CFGRSP
16	Alias for the receipt of an Auto-Configuration CONFIG-RSP message via the
17	UMTCLT.indication primitive. The received CONFIG-RSP message contains the following fields,
18	parameters and values:
19	ind umt subtype
20	ind unity message type \leftarrow CONFIG-RSP (see Table B-1)
21	ind_revision
22	ind sequence number
23	ind transaction id thy
24	ind umt peer identifier thy
25	ind requested unt subtypes the
26	ind tunnel adapter descriptor the
27	ma_mmor_ampror_accord.or_a
28	UMTCLTREQ CFGACK
29	Alias for the request for a Auto-Configuration CONFIG-ACK message to be sent via the
30	UMTCLT.request primitive. The requested CONFIG-ACK message contains the following fields,
31	parameters and values:
32	reg unt subtype
22	req_unit_subtype \leftarrow CONEIC ACK (see Table P 1)
33	req_unun_message_type \leftarrow CONFIG-ACK (see Table D-1)
34 35	reg sequence number
35	req_sequence_number
30	requirement peer identifier the
38	req_unit_peet_lucitinet_uv
30	reg_tunnel_adapter_descriptor_tly
40	
-U	

1	UMICLIIND_CFGREQ				
2	Alias for the receipt of a Auto-Configuration CONFIG-REQ message via the UMTCLT.indication				
3	primitive. The received CONFIG-REQ message contains the following fields, parameters and				
4	values:				
•					
5	ind unt subture				
5					
6	ind _umtm_message_type \Leftrightarrow CONFIG-REQ (see Table B-1)				
7	ind _revision				
8	ind sequence number				
9	ind transaction id the				
10	ind umt peer identifier the				
11	ind requested unit subtypes the				
12	ind_tunnel_adapter_descriptor_thy				
12	nd_dumel_adapter_desemptor_uv				
13					
14	UMICLIREQ_CFGRSP				
15	Alias for the request for a Auto-Configuration CONFIG-RSP message to be sent via the				
16	UMTCLT.request primitive. The requested CONFIG-RSP message contains the following fields,				
17	parameters and values:				
	1				
18	rea umt subtyne				
10	(= CONEIC DSD (= T-h) - D)				
19	$red_umm_message_type$ $\Leftarrow CONFIG-RSP (see Table B-1)$				
20	req_revision				
21	req_sequence_number				
22	req transaction id tlv				
23	req umt peer identifier tly				
24	rea requested umt subtypes thy				
25	rea tunnel adapter descriptor fly				
25	req_uniter_adapter_descriptor_uv				
20					
27	LIMTCI TREO, CECNAR				
27	UMTCLTREQ_CFGNAK				
27 28	UMTCLTREQ_CFGNAK Alias for the request for a Auto-Configuration CONFIG-NAK message to be sent via the				
27 28 29	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,				
27 28 29 30	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:				
27 28 29 30	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:				
27 28 29 30 31	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				
27 28 29 30 31 32	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_umt_message type				
27 28 29 30 31 32 33	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 configuration configuration				
27 28 29 30 31 32 33 34	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 req_revision req_revision				
27 28 29 30 31 32 33 34 25	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 req_revision req_revision req_sequence_number				
27 28 29 30 31 32 33 34 35	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 req_revision req_sequence_number req_transaction_id_tlv				
27 28 29 30 31 32 33 34 35 36	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_umt_message_type req_revision req_sequence_number req_transaction_id_tlv req_umt_peer_identifier_tlv				
27 28 29 30 31 32 33 34 35 36 37	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				
27 28 29 30 31 32 33 34 35 36 37 38	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				
27 28 29 30 31 32 33 34 35 36 37 38 39	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				
27 28 29 30 31 32 33 34 35 36 37 38 39 40	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_umt_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_CEGREJ				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41	<pre>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_umt_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-REL message to be sent via the</pre>				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41	<pre>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_umt_message_type</pre>				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42	<pre>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</pre>				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43	<pre>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</pre>				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43	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_umt_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:				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44	<pre>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_umt_message_type</pre>				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45	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_umt_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 to be sent via the following fields, parameters and values: req_umt_subtype req_umt_message_type ← CONFIG-REJ (see Table B-1)				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46	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_umt_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 to be sent via the following fields, parameters and values: req_umt_subtype req_umt_message_type ← CONFIG-REJ (see Table B-1) req revision				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47	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_umt_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 to be sent via the following fields, parameters and values: req_umt_subtype req_umt_message_type ← CONFIG-REJ (see Table B-1) req_revision req sequence number				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48	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_umt_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 to be sent via 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_stansaction_id_tlv				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49	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_umt_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 to be sent via the UMTCLT.request primitive. The requested CONFIG-REJ message contains the following fields, parameters and values: req_umt_subtype req_umt_subtype req_umt_nessage_type ⇐ CONFIG-REJ (see Table B-1) req_revision req_sequence_number req_transaction_id_tlv req_umts_reprint the request for the request fo				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	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_umt_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 to be sent via the 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_umt_message_type ⇐ CONFIG-REJ (see Table B-1) req_revision req_sequence_number req_transaction_id_tlv req_umt_peer_identifier_tlv req_umt_peer_identifier_tlv req_umt_metr_dentifier_tlv				
27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	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_unt_subtype req_unt_message_type ← CONFIG-NAK (see Table B-1) req_revision req_sequence_number req_transaction_id_tlv req_unt_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 to be sent via the UMTCLT.request primitive. The requested CONFIG-REJ message contains the following fields, parameters and values: req_unt_subtype req_unt_message_type ⇐ CONFIG-REJ (see Table B-1) req_revision req_sequence_number req_transaction_id_tlv req_unt_peer_identifier_tlv param_list '				

1 2 3 4 5	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:
6 7 8 9 10 11 12 13 14	<pre>ind_umt_subtype ind_umtm_message_type</pre>
15 16 17 18	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:
19 20 21 22 23 24 25 26 27	req_umt_subtype req_umt_message_type
27 28 29 30 31	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:
32 33 34 35 36 37 38 20	ind_umt_subtype ind_umtm_message_type ind_revision ind_sequence_number ind_transaction_id_tlv ind_umt_peer_identifier_tlv ind_reason_code
 39 40 41 42 43 	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:
44 45 46 47 48 49 50 51	<pre>ind_umt_subtype ind_umtm_message_type</pre>

1		
2	UMTCLTREQ DELACK	
3	Alias for the request for a Auto-Configur	ation DELETE-ACK message to be sent via the
4	UMTCLT.request primitive. The requested D	ELETE-ACK message contains the following fields,
5	parameters and values:	6 6 7
U		
6	rea umt subtype	
7	req untry message type	\leftarrow DELETE-ACK (see Table B-1)
8	req_revision	\leftarrow DELETE-ACK (see Table D-1)
0	req_revision	
9	req_sequence_number	
10	req_transaction_id_tiv	
11	req_umi_peer_identifier_liv	
12	req_requested_umt_subtypes_tiv	
13	req_tunnel_adapter_descriptor_tlv	
14		
15	UMTCLTIND_DELREQ	
16	Alias for the receipt of a Auto-Configuration	DELETE-REQ message via the UMTCLT.indication
17	primitive. The received DELETE-REQ mes	ssage contains the following fields, parameters and
18	values:	
19	ind_umt_subtype	
20	ind umtm message type	\equiv DELETE-REQ (see Table B-1)
21	ind revision	
22	ind sequence number	
23	ind transaction id the	
24	ind umt peer identifier the	
25	ind requested unit subtypes the	
26	ind tunnel adapter descriptor the	
27		
28	LIMTCI TREO, DEL RSP	
20	Alias for the request for a Auto-Configu	ration DELETE-RSP message to be sent via the
30	LIMTCL T request primitive. The requested I	ELETE-RSP message contains the following fields
21	noremeters and values:	JEETE-KST message contains the following fields,
51	parameters and values.	
32	rea umt subtype	
22	req_unt_subtype	- DELETE DSD (coo Table D 1)
22	req_unun_message_type	~ DELETE-KSF (see Table B-1)
24 25		
33	req_sequence_number	
30	req_transaction_id_tiv	
3/	req_umt_peer_identifier_tiv	
38	req_requested_umt_subtypes_tiv	
39	req_tunnel_adapter_descriptor_tlv	
40		
41	UMTCLTREQ_DELREJ	
42	Alias for the request for a Auto-Configu	ration DELETE-REJ message to be sent via the
43	UMTCLT.request primitive. The requested I	DELETE-REJ message contains the following fields,
44	parameters and values:	
45	req_umt_subtype	
46	req_umtm_message_type	⇐ DELETE-REJ (see Table B-1)
47	req_revision	
48	req_sequence number	
49	req transaction id tlv	
50	req umt peer identifier tlv	
51	req reason code	
	·	

1	
2	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:
- 6 ind_umt_subtype
- 7
 ind_umtm_message_type
 ⇐ DELETE-ACK (see Table B-1)

 8
 ind_revision

 9
 ind_sequence_number

 10
 ind_transaction_id_tlv

 11
 ind_umt_peer_identifier_tlv

 12
 ind_requested_umt_subtypes_tlv

 13
 ind_tunnel_adapter_descriptor_tlv

14 B.4.2 UMT Peer Discovery

- As depicted in Figure B-4, the UMT Discovery function is contained in the UMT Peer Maintenance entityand 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.

21 **B.4.2.1** Active Discovery

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



2

Figure B-5 - Active Discovery Process State Diagram

3 B.4.2.1.1 START_DISC_TIMER State

4 Upon initialization, the START_DISC_TIMER state is entered. In the START_DISC_TIMER state, the 5 Active Discovery process starts the discovery_tx_timer. Upon completion of the START_DISC_TIMER 6 state, the Active Discovery process transitions to the TX_SOLICIT state.

7 B.4.2.1.2 TX_SOLICIT State

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

12 **B.4.2.1.3 WAITING State**

13 The Active Discovery process enters the WAITING state after completing the TX_SOLICIT state. In the 14 WAITING state, the Active Discovery process waits for a UMT Maintenance HELLO message to arrive 15 via the UMTCLT.indication primitive or for the discovery tx timer to expire.

16 If the discovery_tx_timer expires, the Active Discovery process moves back to the TX_SOLICIT state to 17 send another UMT Maintenance SOLICIT message.

18 If the Active Discovery process receives a UMTCLT.indication containing a UMT Maintenance HELLO

- 19 message, the Active Discovery process moves to the RX_HELLO state. All other received message types
- 20 are silently ignored by the Active Discovery process.

1 **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.

5 **B.4.2.2 Passive Discovery Process**

6 A UMT Maintenance entity may implement the Passive Discovery process. If the Passive Discovery 7 process is implemented, it shall implement the passive discovery state diagram shown in Figure B-6.

8



9 10

Figure B-6 - Passive Discovery Process State Diagram

11 B.4.2.2.1 WAITING State

12 Upon initialization the Passive Discovery process enters the WAITING state. In the WAITING state, the 13 Passive Discovery process waits to receive a UMT Maintenance SOLICIT message via the 14 UMTCLT.indication primitive asserted by the Passive Peer Discovery Receive Tunnel Adapter. Upon 15 receipt of the SOLICIT message, the Passive Discovery process moves to the RX SOLICIT state.

16 **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.

1 **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.

8 **B.4.3 UMT Auto-Configuration**

9 As depicted in Figure B-4, the UMT Auto-Configuration function is contained in the UMT Peer 10 Maintenance entity. The Auto-Configuration process is responsible for communicating with peer Auto-11 Configuration entities to negotiate the creation and deletion of UMT Tunnel Adapters. The Auto-12 Configuration process is comprised of the following subprocesses:

- a) *Configuration Initiator*. This function initiates 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.
- 18 c) *Delete Initiator*. This function intitiates a request to a peer Auto-Configuration entity to request
 19 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.

23 **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.



2

Figure B-7 - Configuration Initiator State Diagram

3 B.4.3.1.1 WAITING State

4 Upon initialization, the WAITING state is entered. In the WAITING state, the Configuration Initiator 5 subprocess sets the retry_counter to zero and waits for assertion of the UMTAC.request primitive with the 6 action parameter set to indicate a create action.

7 B.4.3.1.2 TX_CONFIG_REQ State

8 When the TX_CONFIG_REQ state is entered, the Configuration Initiator subprocess starts the retry_timer, 9 increments the retry_counter, and asserts the UMTCLT.request primitive with the required parameters to 10 send a CONFIG-REQ message.

11 B.4.3.1.3 WAIT_FOR_CFG_RSP State

12 In the WAIT_FOR_CFG_RSP state, the Configuration Initiator subprocess waits for any of the following 13 events: 1 retry timer expires

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

6 Receive CONFIG-NAK

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

14 Receive CONFIG-REJ

15 If the UMTCLT.indication primitive is asserted and contains a CONFIG-REJ, the Configuration 16 Initiator subprocess will compare the value of retry_counter to the value of max_retries. If 17 retry_timer is less than max_retries, then the Configuration Initiator subprocess moves to the 18 TX_CONFIG_REQ state where the Configuration Initiator subprocess shall adjust the parameters, 19 fields, and TLVs to be sent in the CONFIG-REQ in a way to achieve agreement with the remote 20 peer's configuration as sent in the CONFIG-REJ. If retry_timer equals or exceeds max_retries 21 then the Configuration Initiator subprocess moves to the WAITING state.

22 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.

26 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.

31 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

3 B.4.3.2.1 WAITING State

4 Upon initialization, the WAITING state is entered. In the WAITING state, the Configuration Init Receiver 5 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. 6

7 B.4.3.2.2 CHECK CFG REQ State

8 Upon entering the CHECK CFG REQ state, the Configuration Init Receiver subprocess calls the 9 check cfg request function to check if the received CONFIG-REQ fields, paramaters, TLVs and values are 10 acceptable.

11 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 12 a cfg req indicating the request contains fields, parameters or TLVs that are unacceptable (cfg req=REJ), 13

14 then the Configuration Init Receiver subprocess moves to the TX CFG REJ state. If the check cfg request

15 returns a cfg req indicating the request indicating that the values of the fields, parameters or TLVs are 1 unacceptable (cfg_req=NAK), then the Configuration Init Receiver subprocess moves to the 2 TX_CFG_NAK state.

3 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.

9 B.4.3.2.4 TX_CFG_NAK State

10 **B.4.3.2.5 TX_CFG_REJ State**

11 B.4.3.2.6 WAIT_FOR_CFG_ACK State

12 In the WAIT_FOR_CFG_ACK state, the Configuration Init Receiver subprocess waits for assertion of the 13 UMTCLT.indication primitive with the umt_client_sdu containing a CONFIG-ACK message. If 14 retry timer expires before a CONFIG-ACK is received

15 **B.4.3.2.7 CREATE_TUNNEL_ADAPTER State**

16 **B.4.3.3 Delete Initiator**

17 A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration

18 process is implemented, it shall implement the Delete Initiator state diagram shown in Figure B-9.



1

Figure B-9 - Delete Initiator State Diagram

- 3 B.4.3.3.1 WAITING State
- 4 B.4.3.3.2 TX_DEL_REQ State
- 5 B.4.3.3.3 WAIT_FOR_DEL_RSP State
- 6 B.4.3.3.4 TX_DEL_ACK State
- 7 B.4.3.4 Delete Receiver
- 8 A UMT Maintenance entity may implement the Auto-Configuration process. If the Auto-Configuration
- 9 process is implemented, it shall implement the Delete Receiver state diagram shown in Figure B-10



Figure B-10 - Delete Receiver State Diagram

- 3 B.4.3.4.1 WAITING State
- 4 B.4.3.4.2 CHECK_DEL_REQ State
- 5 B.4.3.4.3 TX_DEL_RSP State
- 6 B.4.3.4.4 TX_DEL_REJ State
- 7 B.4.3.4.5 WAIT_FOR_DEL_ACK State
- 8 B.4.3.4.6 DELETE_TUNNEL_ADAPTER State
- 9 B.5 UMT Peer Maintenance SDU Format

10 UMT Peer Maintenance SDUs are encapsulated in UMTPDUs under the UMT Peer Maintenance subtype

11 (See IEEE Std. 1904.2 Table 4-2).

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

4 B.5.1 Structure

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



- 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.

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		

Table B-1 - UMT Maintenance Message Types

5

B.5.2 **UMT Peer Maintenance TLVs** 6

7 B.5.3 **Encodings for UMT Maintenance TLVs**

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

9 B.5.3.1 Supported UMT SubTypes

10 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 11 12 in Table 4-1. The Length parameter indicates the number of 1-octet values contained in the field.

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

13 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 14

15 UMT tunnel adapter. This list is structured as a series of 1-octet values. Each supported type is represented

16 by its corresponding value found in Table 4-1. The Length parameter indicates the number of 1-octet values contained in the field. 17

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

1

2

3

4

1 **B.5.3.3 Transaction Identifier**

2 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 3 4 correlate the messages sent between the two UMT peers.

Туре	Length	Value
3	4	4-octet random number

5

B.5.3.4 Tunnel Adapter Descriptor 6

7 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 8 9

more than one instance of each of the sub-TLVs defined in this subclause.

Туре	Length	Value
4	n	Encapsulated sub-TLVs

10 B.5.3.4.1 Tunnel Adapter Transmission Method Subtype

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

Туре	Length	Value
1	1	1 – Broadcast
		2 – Unicast
		3 – Multicast
		4 – Receive Only

12

13 B.5.3.4.2 Tunnel Adapter Indicated Source Address Subtype

14 This field specifies the Source Address of incoming UMTPDUs to be associated with the local tunnel

15 adapter. This is the MAC Source Address that the local tunnel adapter expects in a received UMTPDU.

Туре	Length	Value
2	6	48-bit MAC address

16 B.5.3.4.3 Tunnel Adapter Indicated Destination Address Subtype

17 This field specifies the Destination Address of transmitted UMTPDUs to be associated with the local tunnel 18 adapter. This is the MAC Destination Address that the local tunnel adapter expects in a received UMTPDU.

Туре	Length	Value
3	6	48-bit MAC address

19 B.5.3.4.4 Tunnel Adapter Requested Source Address Subtype

20 This field specifies the Source Address of transmitted UMTPDUs to be associated with the remote tunnel

21 adapter. This is the MAC Source Address that the remote tunnel adapter expects in a received UMTPDU.

Туре	Length	Value
4	6	48-bit MAC address

1 B.5.3.4.5 Tunnel Adapter Requested Destination Address Subtype

2 This field specifies the Destination Address of transmitted UMTPDUs to be associated with the remote 3 tunnel adapter. This is the MAC Destination Address that the remote tunnel adapter expects in a received

4 UMTPDU.

Туре	Length	Value
5	6	48-bit MAC address

5 **B.5.3.5 UMT Peer Identifier**

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

Туре	Length	Value
6	6	48-bit MAC address

7 **B.5.3.6 Reason Code**

8 This field contains a reason code encoded as an n-octet integer. The reason code indicates to a receiving

9 entity the reason for an error associated with the parameter negotiation. Multiple instances of this field may

10 be present in a UMT Maintenance SDU.

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

11

12

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			

13

14 B.5.3.7 Vendor-Specific Extension

15 The Vendor-Specific extension field may be used to extend the capabilities of a specific implementation of 16 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.

Туре	Length	Value
253	n	Unspecified

- 1
- 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)

6 Upon initialization, a UMT Peer Discovery entity in Active mode configures the local UMT peer with a 7 UMT Tunnel Adapter configured for broadcast operation in the transmit direction (req_SA=local MAC 8 address, req_DA=MAC broadcast) and unicast operation in the receive direction (ind_SA=any MAC 9 address, req_DA=MAC broadcast) and unicast operation in the receive direction (ind_SA=any MAC

9 address, ind_DA=local MAC address). This Tunnel Adapter is the *Active Peer Discovery Tunnel Adapter*.

10 A UMT Peer Discovery entity operating in Passive or Active mode configures a UMT Tunnel Adapter for 11 (req_SA=local MAC address, req_DA=MAC broadcast, ind_SA=any, ind_DA=MAC broadcast). This 12 second UMT Tunnel Adapter is called the *Passive Peer Discovery Tunnel Adapter* and is never used by the 13 UMT Peer Discovery entity to transmit a UMTPDU.

14 The Active UMT Peer Discovery entity generates an UMTPDU with the Subtype set to UMT Maintenance. 15 This UMTPDU will be called a UMT Maintenace SDU (UMTMSDU). The UMT Peer Discovery entity 16 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

22

23 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.

34