initialization, one of two options is chosen for the calculation of the first sniff slot. A timing control flag in the message from the master indicates this. Note that only bit1 of the timing control flag is valid.

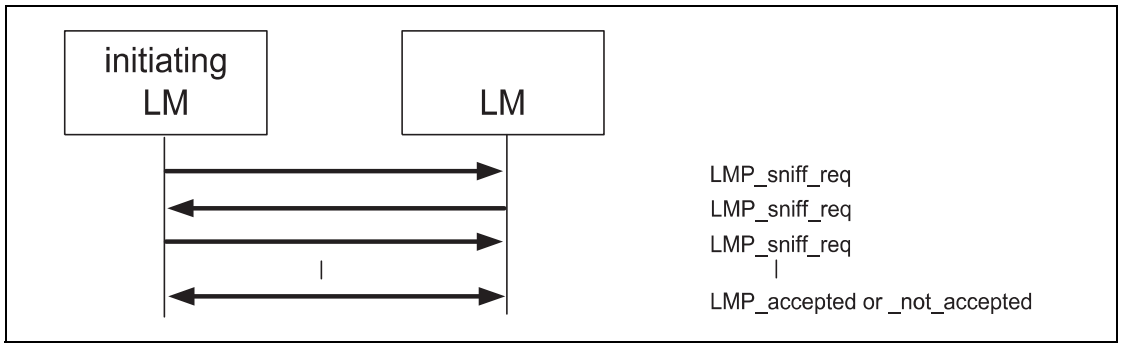
When the link is in sniff mode the master can only start a transmission in the sniff slot. Two parameters control the listening activity in the slave. The sniff attempt parameter determines for how many slots the slave shall listen, beginning at the sniff slot, even if it does not receive a packet with its own AM address. The sniff timeout parameter determines for how many additional slots the slave shall listen if it continues to receive only packets with its own AM address (see Table 57).

Table 57—PDUs used for sniff mode

M/O	PDU	Contents
O	LMP_sniff_req	timing control flags D_{sniff} T_{sniff} sniff attempt sniff timeout
O	LMP_unsniff_req	-

9.3.16.1 Master or slave requests sniff mode

The master or the slave can request to enter sniff mode. Upon receipt of the request, the same request with modified parameters can be returned or the negotiation can be terminated. If an agreement is seen LMP_accepted terminates the negotiation and the ACL link is placed in sniff mode. If no agreement is seen, LMP_not_accepted with the reason code *unsupported parameter value* terminates the negotiation and sniff mode is not entered.



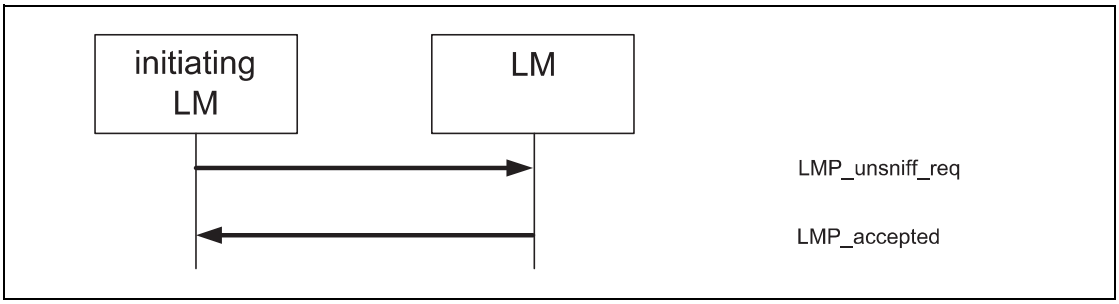
Sequence 30: Negotiation for sniff mode.

9.3.16.2 Moving a slave from sniff mode to active mode

Sniff mode is ended by sending the PDU LMP_unsniff_req. The requested device shall reply with LMP_accepted. If the slave requests, it will enter active mode after receiving LMP_accepted. If the master requests, the slave will enter active mode after receiving LMP_unsniff_req.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 192 of 1169



Sequence 31: Slave moved from sniff mode to active mode.

9.3.17 Park mode

If a slave does not need to participate in the channel, but still should be FH-synchronized, it can be placed in park mode. In this mode the device gives up its AM_ADDR but still re-synchronizes to the channel by waking up at the beacon instants separated by the beacon interval. The beacon interval, a beacon offset, and a flag indicating how the first beacon instant is calculated determine the first beacon instant. After this the beacon instants follow periodically at the predetermined beacon interval. At the beacon instant the parked slave can be activated again by the master, the master can change the park mode parameters, transmit broadcast information, or let the parked slaves request access to the channel.

All PDUs sent from the master to the parked slaves are broadcast. These PDUs (LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_addr_req and LMP_unpark_PM_addr_req) are the only PDUs that can be sent to a slave in park mode and the only PDUs that can be broadcast. To increase reliability for broadcast, the packets are made as short as possible. Therefore, the format for these LMP PDUs are somewhat different. The parameters are not always byte-aligned and the length of the PDUs is variable.

The messages for controlling the park mode include many parameters, which are all defined in 8.10.8.4. When a slave is placed in park mode it is assigned a unique PM_ADDR, which can be used by the master to unpark that slave. The all-zero PM_ADDR has a special meaning; it is not a valid PM_ADDR. If a device is assigned this PM_ADDR, it shall be identified with its BD_ADDR when it is unparked by the master (see Table 58).

9.3.17.1 Master requests slave to enter park mode

The master can request park mode. The master finalizes the transmission of the current ACL packet with L2CAP information, stops point-to-point L2CAP transmission, and then sends LMP_park_req. If the slave accepts to enter park mode, it finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission, and then responds with LMP_accepted.

When the slave queues LMP_accepted, it starts a timer for $6*T_{poll}$ slots. If the Baseband-level acknowledgement is received before this timer expires, it enters park mode immediately, otherwise it enters park mode when the timer expires.

When the master receives LMP_accepted, it starts a timer for $6*T_{poll}$ slots. When this timer expires the slave is in park mode and the AM_ADDR can be re-used.

Note that if the master never receives the LMP_accepted then a link supervision timeout will occur.

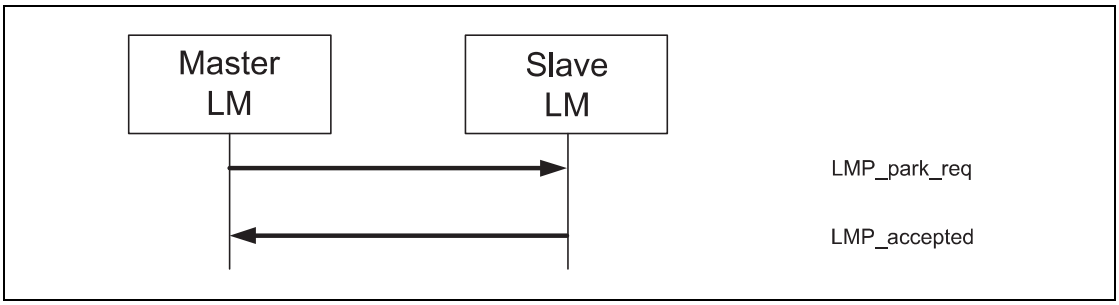
Copyright © 2002 IEEE. All rights reserved.

Table 58—PDUs used for park mode

M/O	PDU	Contents
O	LMP_park_req	timing control flags D_B T_B N_B Δ_B PM_ADDR AR_ADDR N_{Bsleep} D_{Bsleep} D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} access scheme
O	LMP_set_broadcast_scan_window	timing control flags D_B (optional) broadcast scan window
O	LMP_modify_beacon	timing control flags D_B (optional) T_B N_B Δ_B D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} access scheme
O	LMP_unpark_PM_ADDR_req	timing control flags D_B (optional) AM_ADDR PM_ADDR AM_ADDR (optional) PM_ADDR (optional) (totally 1–7 pairs of AM_ADDR, PM_ADDR)
O	LMP_unpark_BD_ADDR _req	timing control flags D_B (optional) AM_ADDR BD_ADDR AM_ADDR (optional) BD_ADDR (optional)

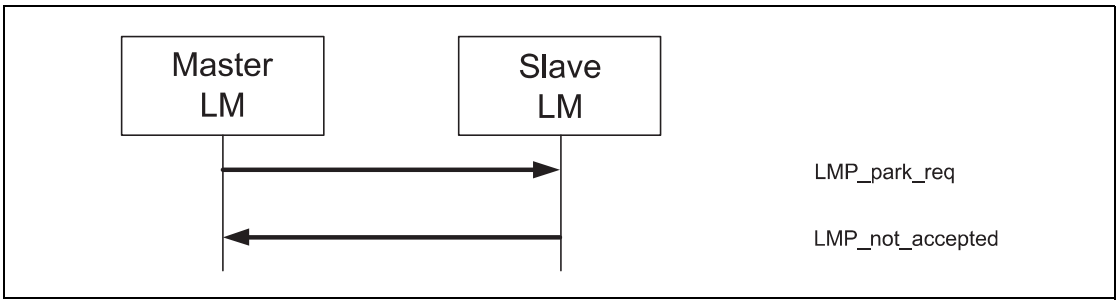
Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 194 of 1169



Sequence 32: Slave accepts to enter park mode.

If the slave rejects to enter park mode it responds with LMP_not_accepted and the master re-enables L2CAP transmission.



Sequence 33: Slave rejects to enter into park mode

9.3.17.2 Slave requests to enter park mode

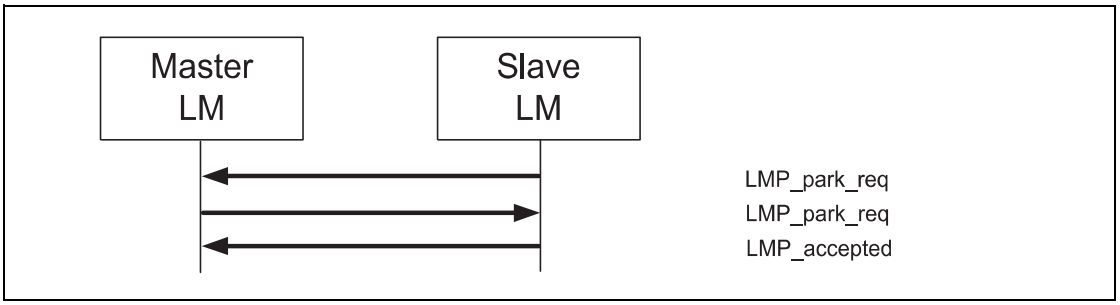
The slave can request park mode. The slave finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission, and then sends LMP_park_req. The parameters PM_ADDR and AR_ADDR are not valid and the other parameters represent suggested values. If the master wants the slave to enter park mode, it finalizes the transmission of the current ACL packet with L2CAP information, stops point-to-point L2CAP transmission, and then sends LMP_park_req, where the parameter values may be different from the values in the PDU sent from the slave. If the slave can accept these parameters, it responds with LMP_accepted.

When the slave queues LMP_accepted, it starts a timer for $6*T_{poll}$ slots. If the Baseband-level acknowledgement is received before this timer expires, it enters park mode immediately, otherwise it enters park mode when the timer expires.

When the master receives LMP_accepted it starts a timer for $6*T_{poll}$ slots. When this timer expires the slave is in park mode and the AM_ADDR can be re-used.

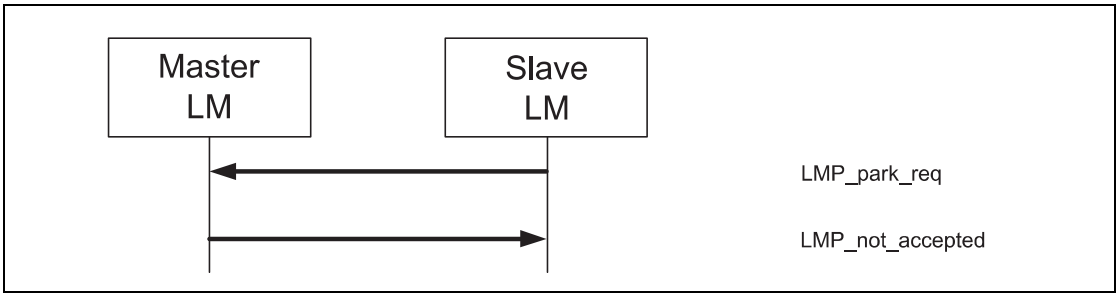
Note that if the master never receives the LMP_accepted then a link supervision timeout will occur.

Copyright © 2002 IEEE. All rights reserved.



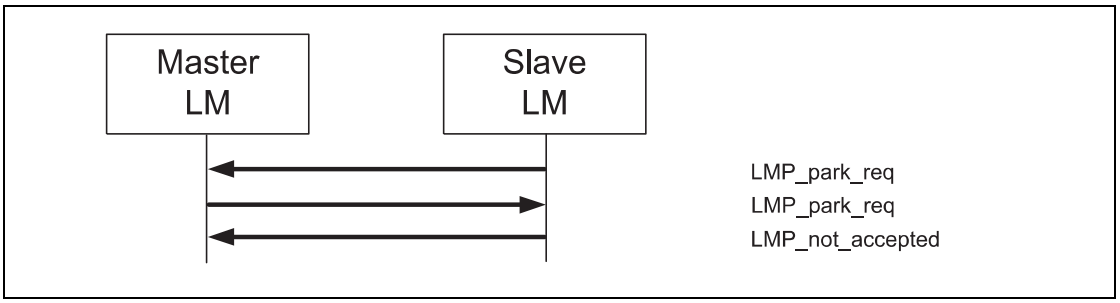
Sequence 34: Slave requests to enter park mode and accepts master's beacon parameters.

If the master does not accept that the slave enters park mode, it sends LMP_not_accepted. The slave then re-enables L2CAP transmission.



Sequence 35: Master rejects slave's request to enter park mode

If the slave does not accept the parameters in LMP_park_req sent from the master, it responds with LMP_not_accepted and both units re-enable L2CAP transmission.



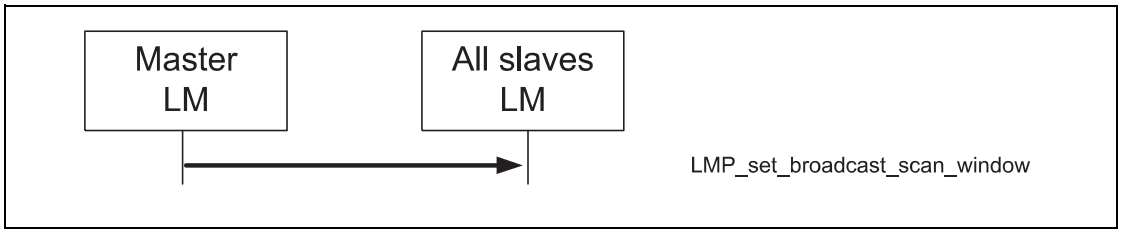
Sequence 36: Slave requests to enter park mode, but rejects master's beacon parameters.

9.3.17.3 Master sets up broadcast scan window

If more broadcast capacity is needed than the beacon train, the master can indicate to the slaves that more broadcast information will follow the beacon train by sending LMP_set_broadcast_scan_window. This message is always sent in a broadcast packet at the beacon slot(s). The scan window starts in the beacon instant and is only valid for the current beacon.

Copyright © 2002 IEEE. All rights reserved.

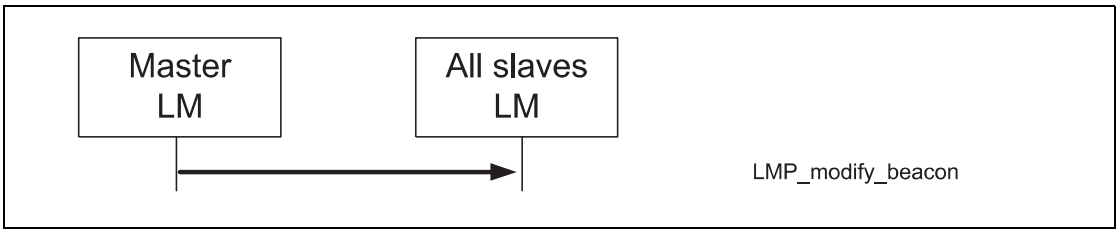
LINKPLAY EXHIBIT 1008
Page 196 of 1169



Sequence 37: Master notifies all slaves of increase in broadcast capacity.

9.3.17.4 Master modifies beacon parameters

When the beacon parameters change, the master notifies the parked slaves of this by sending LMP_modify_beacon. This message is always sent in a broadcast packet at the beacon slot(s).



Sequence 38: Master modifies beacon parameters.

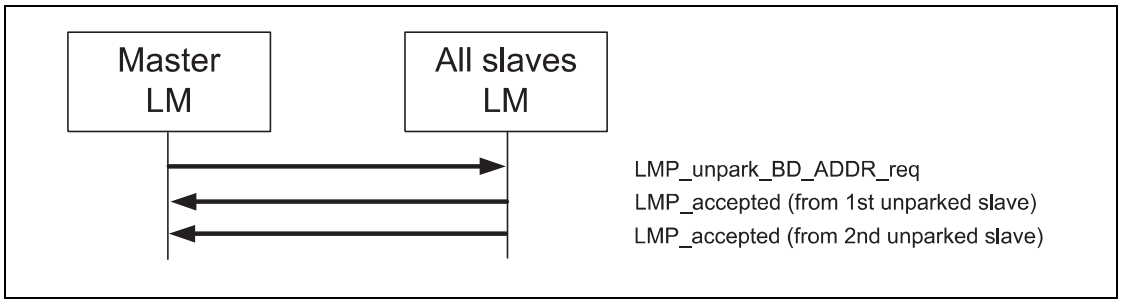
9.3.17.5 Unparking slaves

The master can unpark one or many slaves by sending a broadcast LMP message including the PM_ADDR or the BD_ADDR of the device(s) it wishes to unpark at the beacon slot(s). This message also includes the AM_ADDR that the master assigns to the slave(s). After sending this message, the master shall check the success of the unpark by polling each unparked slave, i.e., sending POLL packets, so that the slave is granted access to the channel. The unparked slave shall then send a response with LMP_accepted. If this message is not received from the slave within a certain time after the master sent the unpark message, the unpark failed and the master shall consider the slave as still being in park mode.

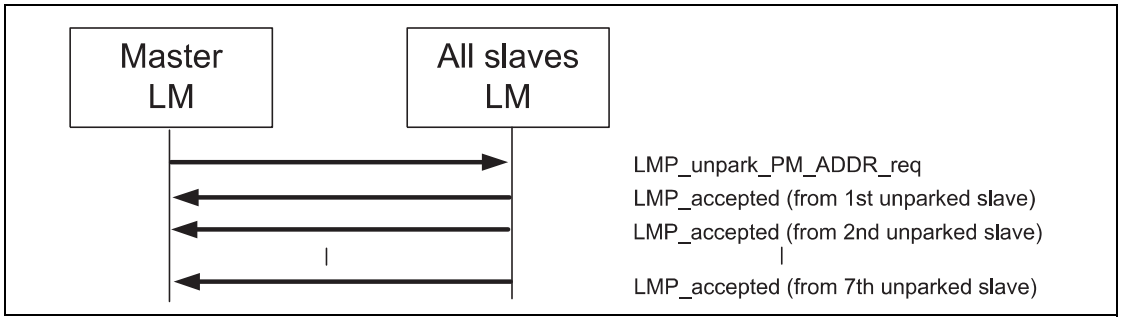
One message is used where the parked device is identified with the PM_ADDR, and another message is used where it is identified with the BD_ADDR. Both messages have variable length depending on the number of slaves the master unparks. For each slave the master wishes to unpark an AM_ADDR followed by the PM/BD_ADDR of the device that is assigned this AM_ADDR is included in the payload. If the slaves are identified with the PM_ADDR a maximum of seven slaves can be unparked with the same message. If they are identified with the BD_ADDR a maximum of two slaves can be unparked with the same message.

After a successful unparking, both units re-enable L2CAP transmission.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 197 of 1169



Sequence 39: Master unparks slaves addressed with their BD_ADDR.



Sequence 40: Master unparks slaves addressed with their PM_ADDR.

9.3.18 Power control

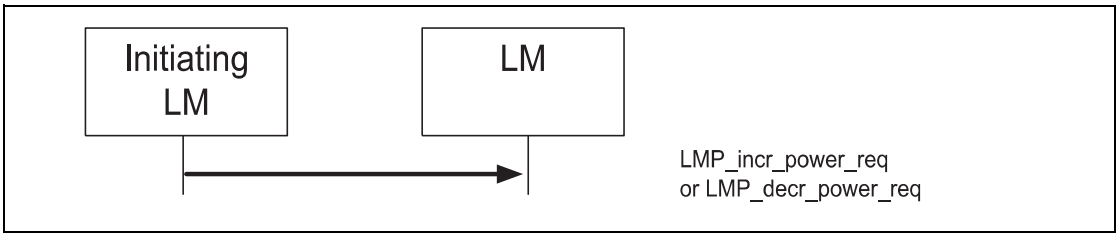
If the RSSI value differs too much from the preferred value of a Bluetooth device, it can request an increase or a decrease of the other device's TX power. The power adjustment requests can be made at anytime following a successful baseband paging procedure. If a device does not support power control requests this is indicated in the supported features list and thus no power control requests shall be sent after the supported features response has been processed. Prior to this time, a power control adjustment might be sent and if the recipient does not support power control it is allowed to send LMP_max_power in response to LMP_incr_power_req and LMP_min_power in response to LMP_decr_power_req. Another possibility is to send LMP_not_accepted with the reason unsupported LMP feature. Upon receipt of this message, the output power is increased or decreased one step. See 7.3 for the definition of the step size. At the master side the TX power is completely independent for different slaves; a request from one slave can only effect the master's TX power for that same slave (see Table 59).

Table 59—PDUs used for power control

M/O	PDU	Contents
O	LMP_incr_power_req	for future use (1 Byte)
O	LMP_decr_power_req	for future use (1 Byte)
O	LMP_max_power	—
O	LMP_min_power	—

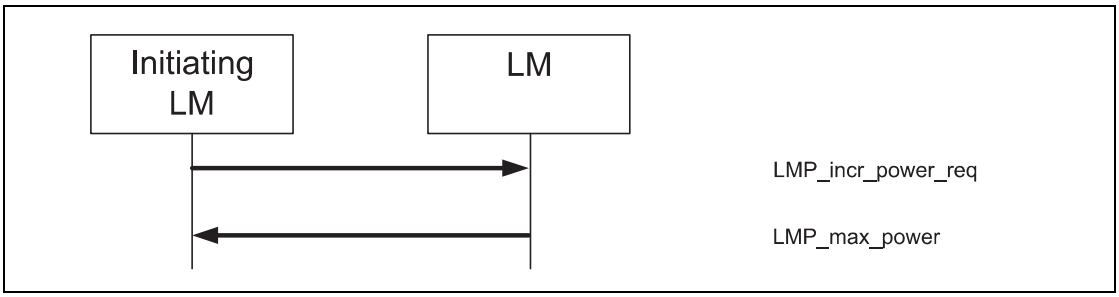
Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 198 of 1169

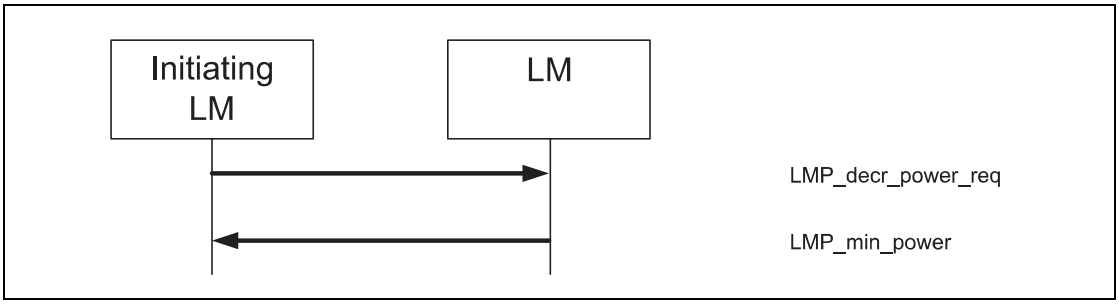


Sequence 41: A device requests a change of the other device's TX power.

If the receiver of LMP_incr_power_req already transmits at maximum power, LMP_max_power is returned. The device may then only request an increase again after having requested a decrease at least once. Similarly, if the receiver of LMP_decr_power_req already transmits at minimum power then LMP_min_power is returned and the device may only request a decrease again after having requested an increase at least once.



Sequence 42: The TX power cannot be increased.



Sequence 43: The TX power cannot be decreased.

One byte is reserved in LMP_incr/decr_power_req for future use. It could, for example, be the mismatch between preferred and measured RSSI. The receiver of LMP_incr/decr_power_req could then use this value to adjust to the correct power at once, instead of only changing it one step for each request. The parameter value shall be 0x00 for all versions of LMP where this parameter is not yet defined.

9.3.19 Channel quality-driven change between DM and DH

The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. If a device A wants the remote device B to have this control it sends LMP_auto_rate once. The device B can then send back LMP_preferred_rate to

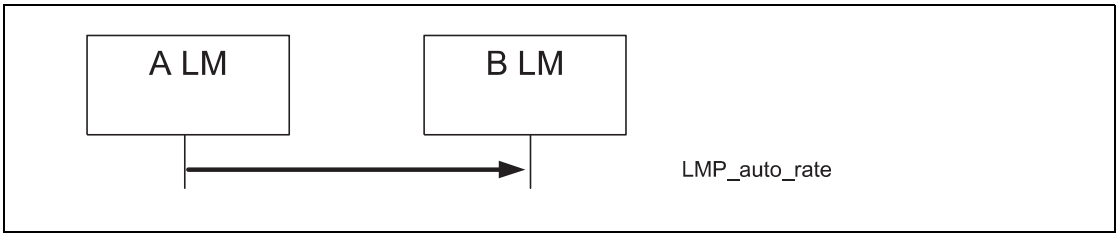
Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 199 of 1169

device A whenever it wishes to change the packet type that A transmits. This PDU has a parameter which determines the preferred coding (with or without 2/3FEC) and the preferred size (in slots) of the packets. Device A is not required to change to the packet type specified by this parameter and may never send a packet that is larger than the maximum allowed number of slots even if the preferred size is greater than this value.

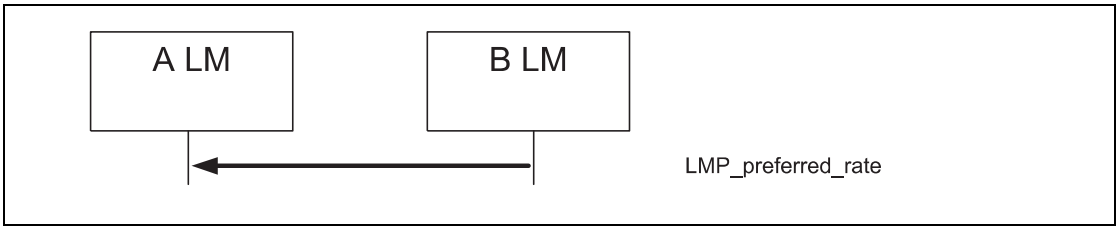
These PDUs can be sent at anytime after connection setup is completed. These PDUs shall not be sent to a given Bluetooth device if the supported features list of the device shows that it does not support the channel quality driven change requests (see Table 60).

Table 60—PDUs used for quality driven change of the data rate

M/O	PDU	Contents
O	LMP_auto_rate	—
O	LMP_preferred_rate	data rate



Sequence 44: A wants B to control A's packet type



Sequence 45: B changes A's packet type.

9.3.20 Quality of service (QoS)

The Link Manager provides QoS capabilities. A poll interval, which is defined as the maximum time between subsequent transmissions from the master to a particular slave on the ACL link, is used to support bandwidth allocation and latency control. The poll interval is guaranteed in the active mode except when there are collisions with page, page scan, inquiry, and inquiry scan. The poll interval is also known as T_{poll}. These PDUs can be sent at anytime after connection setup is completed.

In addition, master and slave negotiate the number of repetitions for broadcast packets (N_{BC}); see 8.5.3.5 and Table 61.

Copyright © 2002 IEEE. All rights reserved.

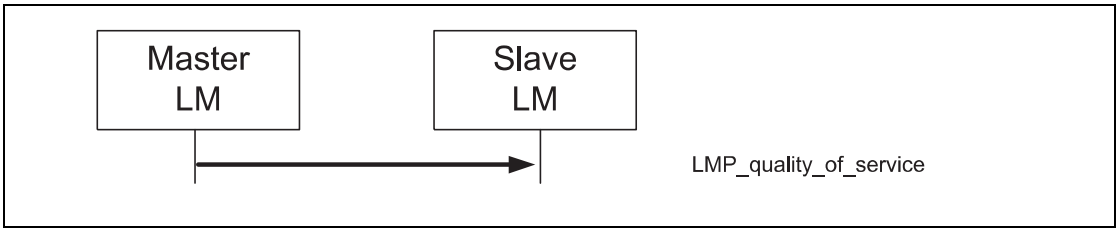
LINKPLAY EXHIBIT 1008
Page 200 of 1169

Table 61— PDUs used for QoS

M/O	PDU	Contents
M	LMP_quality_of_service	poll interval N_{BC}
M	LMP_quality_of_service_req	poll interval N_{BC}

9.3.20.1 Master notifies slave of the QoS

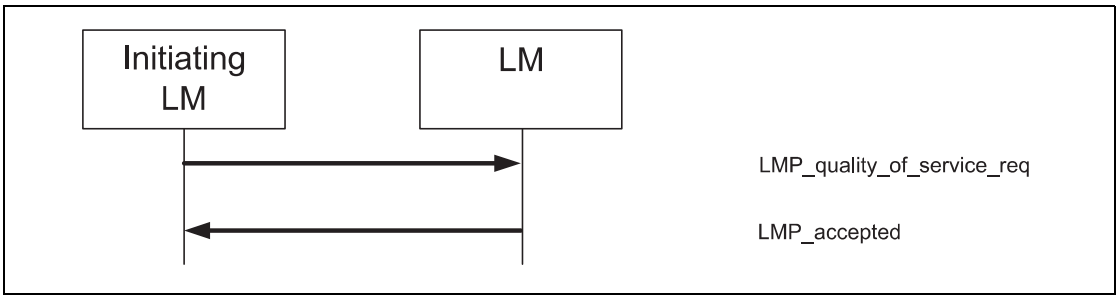
In this case the master notifies the slave of the new poll interval and N_{BC}. The slave cannot reject the notification.



Sequence 46: Master notifies slave of the QoS.

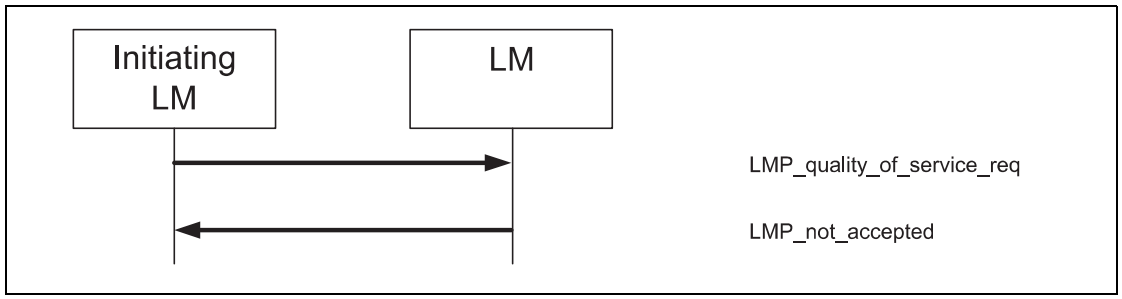
9.3.20.2 Device requests new QoS

In this case the master or slave requests a new poll interval and N_{BC}. The parameter N_{BC} is meaningful only when it is sent by a master to a slave. For transmission of LMP_quality_of_service_req PDUs from a slave, this parameter is ignored by the master. The request can be accepted or rejected. This will allow the master and slave to dynamically negotiate the QoS as needed.



Sequence 47: Device accepts new QoS.

Copyright © 2002 IEEE. All rights reserved.



Sequence 48: Device rejects new QoS.

9.3.21 SCO Links

When a connection has been established between two Bluetooth devices, the connection consists of an ACL link. One or more SCO links can then be established. The SCO link reserves slots separated by the SCO interval, T_{sco}. The first slot reserved for the SCO link is defined by T_{sco} and the SCO offset, D_{sco}. After that the SCO slots follows periodically with the SCO interval. To avoid problems with a wrap-around of the clock during initialization of the SCO link, a flag indicating how the first SCO slot should be calculated is included in a message from the master. Note: Only bit 0 and bit 1 of this field is valid. Each SCO link is distinguished from all other SCO links by an SCO handle. The SCO handle zero is never used (see Table 62).

Table 62—PDUs used for managing the SCO links

M/O	PDU	Contents
O	LMP_SCO_link_req	SCO handle timing control flags D_{sco} T_{sco} SCO packet air mode
O	LMP_remove_SCO_link_req	SCO handle reason

9.3.21.1 Master initiates an SCO link

When establishing an SCO link the master sends a request with parameters that specify the timing, packet type and coding that will be used on the SCO link. For each of the SCO packets, Bluetooth supports three different voice coding formats on the air-interface: μ-law log PCM, A-law log PCM, and CVSD. The air coding by log PCM or CVSD can be deactivated to achieve a transparent synchronous data link at 64 kbits/s.

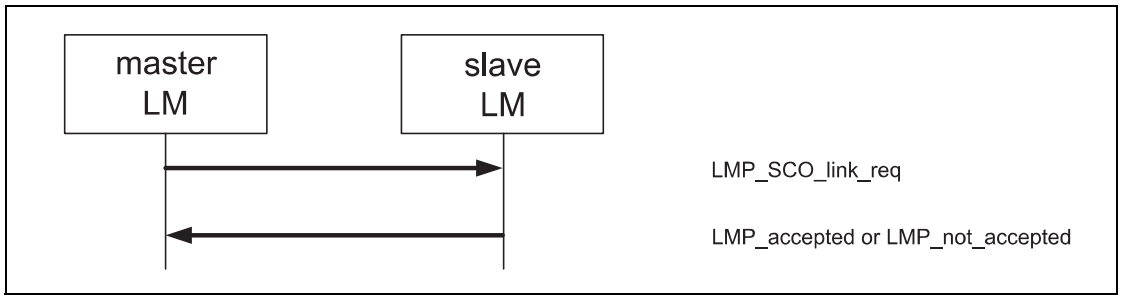
The slots used for the SCO links are determined by three parameters controlled by the master: T_{sco}, D_{sco}, and a flag indicating how the first SCO slot should be calculated. After the first slot, the SCO slots follows periodically with the T_{sco}.

If the slave does not accept the SCO link, but is willing to consider another possible set of SCO parameters, it can indicate what it does not accept in the error reason field of LMP_not_accepted. The master then has the possibility to issue a new request with modified parameters.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 202 of 1169

The SCO handle in the message shall be different from any already existing SCO link(s). Note that if the SCO packet type is HV1 the LMP_accepted shall be sent using the DM1 packet.

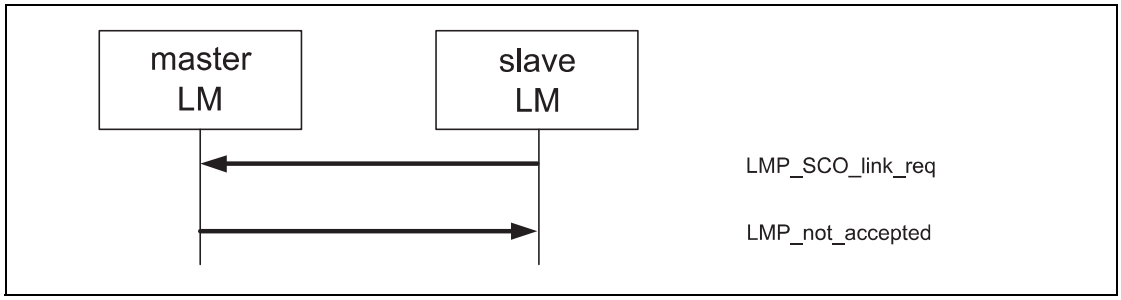


Sequence 49: Master requests an SCO link.

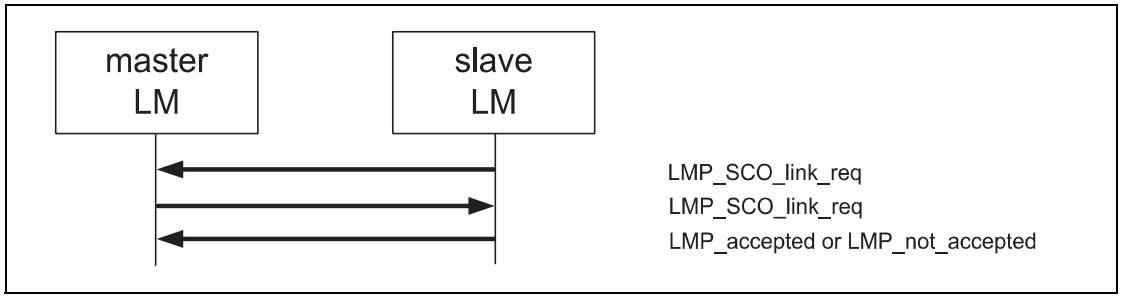
9.3.21.2 Slave initiates an SCO link

The slave can also initiate the establishment of an SCO link. The slave sends LMP_SCO_link_req, but the parameters timing control flags and D_{sco} are invalid as well as the SCO handle, which shall be zero. If the master is not capable of establishing an SCO link, it replies with LMP_not_accepted. Otherwise, it sends back LMP_SCO_link_req. This message includes the assigned SCO handle, $D_{sco,}$ and the timing control flags. For the other parameters, the master should try to use the same parameters as in the slave request; if the master cannot meet that request, it is allowed to use other values. The slave shall then reply with LMP_accepted or LMP_not_accepted.

Note that if the SCO packet type is HV1 the LMP_accepted shall be sent using the DM1 packet.



Sequence 50: Master rejects slave's request for an SCO link.



Sequence 51: Master accepts slave's request for an SCO link.

Copyright © 2002 IEEE. All rights reserved.

9.3.21.3 Master requests change of SCO parameters

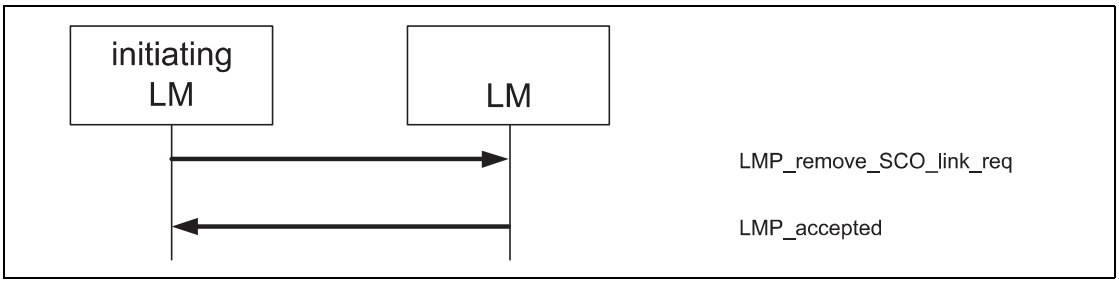
The master sends LMP_SCO_link_req, where the SCO handle is the handle of the SCO link the master wishes to change parameters for. If the slave accepts the new parameters, it replies with LMP_accepted and the SCO link will change to the new parameters. If the slave does not accept the new parameters, it replies with LMP_not_accepted and the SCO link is left unchanged. When the slave replies with LMP_not_accepted, it shall indicate in the error reason parameter what it does not accept. The master can then try to change the SCO link again with modified parameters. The sequence is the same as in 9.3.21.1.

9.3.21.4 Slave requests change of SCO parameters

The slave sends LMP_SCO_link_req, where the SCO handle is the handle of the SCO link the slave wishes to change parameters for. The parameters timing control flags and D_{sco} are not valid in this message. If the master does not accept the new parameters, it replies with LMP_not_accepted and the SCO link is left unchanged. If the master accepts the new parameters, it replies with LMP_SCO_link_req, where it shall use the same parameters as in the slave request. When receiving this message, the slave replies with LMP_not_accepted if it does not accept the new parameters. The SCO link is then left unchanged. If the slave accepts the new parameters, it replies with LMP_accepted and the SCO link will change to the new parameters. These sequences are the same as in 9.3.21.2.

9.3.21.5 Remove an SCO link

Master or slave can remove the SCO link by sending a request including the SCO handle of the SCO link to be removed and a reason indicating why the SCO link is removed. The receiving party shall respond with LMP_accepted.



Sequence 52: SCO link removed.

9.3.22 Control of multislot packets

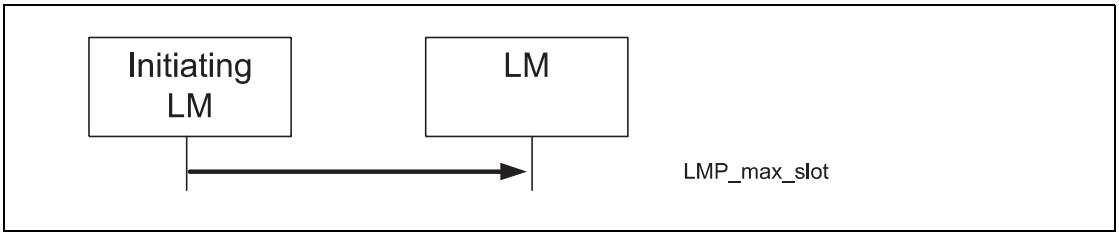
The number of slots used by a device can be limited. A device allows the remote device to use a maximal number of slots by sending the PDU LMP_max_slot providing max slots as parameter. Each device can request to use a maximal number of slots by sending the PDU LMP_max_slot_req providing max slots as parameter. After a new connection, as a result of page, page scan, master-slave switch, or unpark, the default value is 1 slot. Two PDUs are used for the control of multislot packets.These PDUs can be sent at anytime after connection setup is completed (see Table 63).

Copyright © 2002 IEEE. All rights reserved.

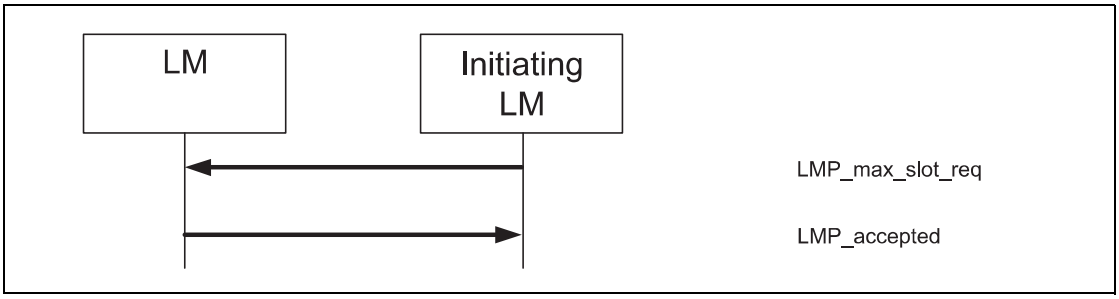
LINKPLAY EXHIBIT 1008
Page 204 of 1169

Table 63—PDUs used to control the use of multislot packets

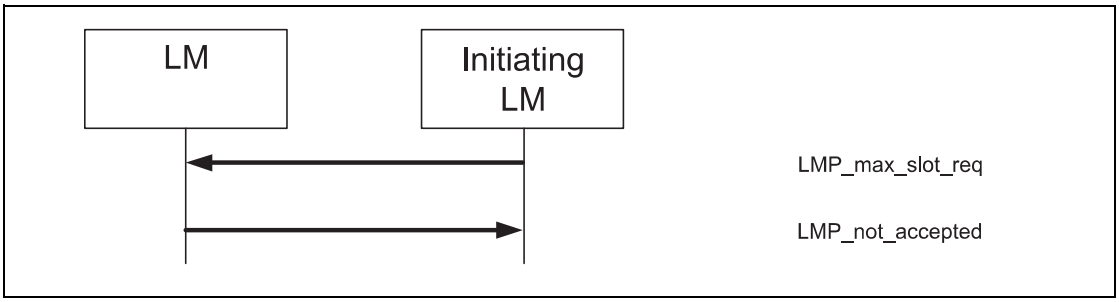
M/O	PDU	Contents
M	LMP_max_slot	max slots
M	LMP_max_slot_req	max slots



Sequence 53: Device allows Remote Device to use a maximum number of slots.



Sequence 54: Device requests a maximum number of slots. Remote Device accepts.



Sequence 55: Device requests a maximum number of slots. Remote Device rejects.

9.3.23 Paging scheme

In addition to the mandatory paging scheme, Bluetooth defines optional paging schemes; see Annex D. LMP provides a means to negotiate the paging scheme, which is to be used the next time a unit is paged (see Table 64).

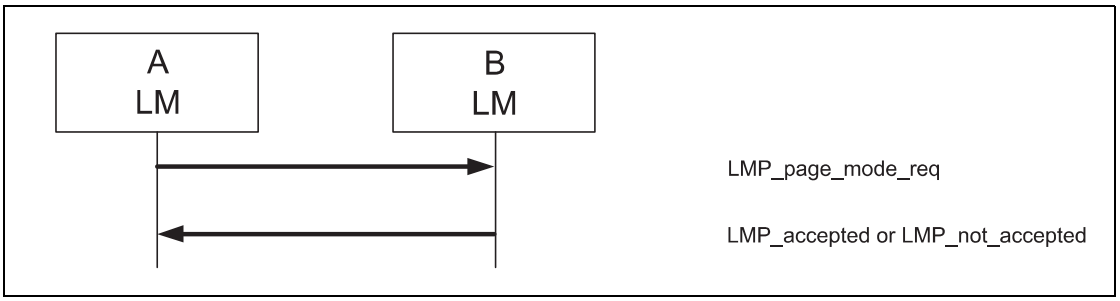
Copyright © 2002 IEEE. All rights reserved.

Table 64—PDUs used to request paging scheme

M/O	PDU	Contents
O	LMP_page_mode_req	paging scheme paging scheme settings
O	LMP_page_scan_mode_req	paging scheme paging scheme settings

9.3.23.1 Page mode

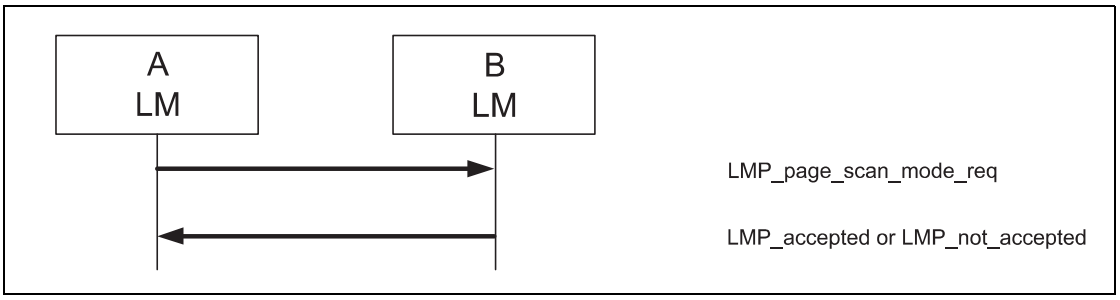
This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting is not changed. A request to switch back to the mandatory scheme may be rejected.



Sequence 56: Negotiation for page mode.

9.3.23.2 Page scan mode

This procedure is initiated from device A and negotiates the paging scheme used when device B pages device A. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting is not changed. A request to switch to the mandatory scheme shall be accepted.



Sequence 57: Negotiation for page scan mode.

Copyright © 2002 IEEE. All rights reserved.

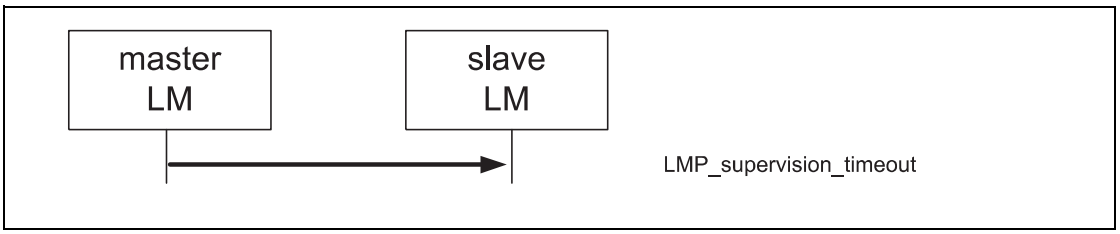
LINKPLAY EXHIBIT 1008
Page 206 of 1169

9.3.24 Link supervision

Each Bluetooth link has a timer that is used for link supervision. This timer is used to detect link loss caused by devices moving out of range, a device's power-down, or other similar failure cases. The scheme for link supervision is described in 8.10.11. An LMP procedure is used to set the value of the supervision timeout (see Table 65).

Table 65—PDU used to set the supervision timeout

M/O	PDU	Contents
M	LMP_supervision_timeout	supervision timeout



Sequence 58: Setting the link supervision timeout.

9.4 Connection Establishment

After the paging procedure, the master shall poll the slave with a max poll interval as defined in Table 71. LMP procedures for clock offset request, LMP version, supported features, name request and detach can then be carried out.

When the paging device wishes to create a connection involving layers above LM, it sends LMP_host_connection_req (see Table 66). When the other side receives this message, the host is informed about the incoming connection. The remote device can accept or reject the connection request by sending LMP_accepted or LMP_not_accepted. Alternatively, if the slave needs a master-slave switch (see 9.3.12), it sends LMP_slot_offset and LMP_switch_req after it has received LMP_host_connection_req. When the master-slave switch has been successfully completed, the old slave will reply with LMP_accepted or LMP_not_accpted to LMP_host_connection_req (with the transaction ID set to 0).

If LMP_host_connection_req is accepted, LMP security procedures (pairing, authentication, and encryption) can be invoked. When a device is not going to initiate any more security procedures during connection establishment, it sends LMP_setup_complete (see Table 66). When both devices have sent LMP_setup_complete the first packet on a logical channel different from LMP can then be transmitted.

Note that the transaction ID shall be 0 if LMP_setup_complete is sent from the master and 1 if it is sent from the slave. Figure 93 summarizes the LMP transactions for establishing a connection following the baseband page procedure.

Copyright © 2002 IEEE. All rights reserved.

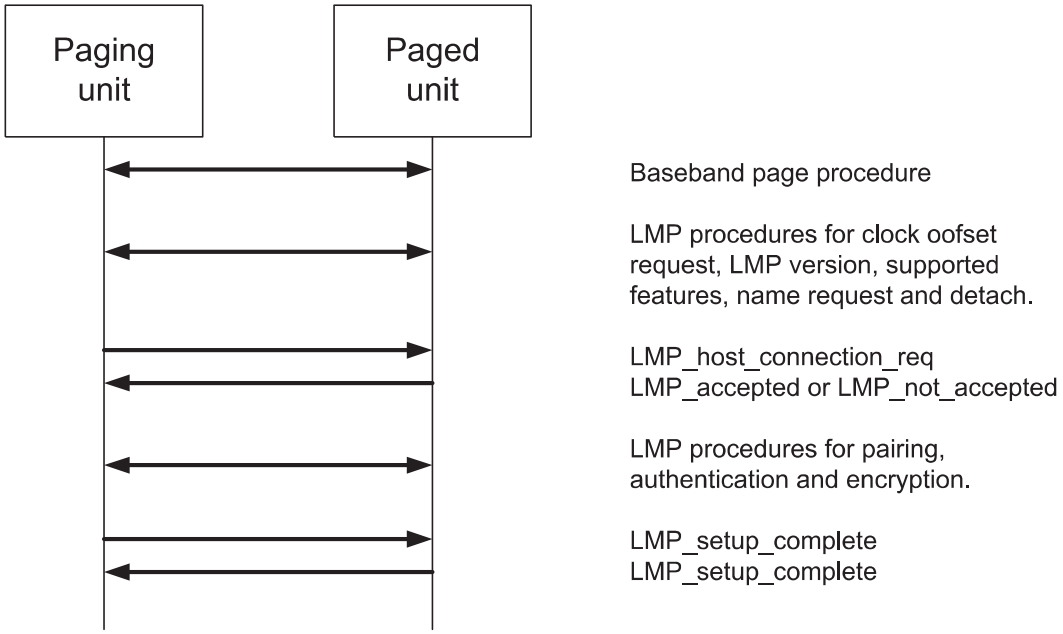


Figure 93—Connection establishment

Table 66—PDUs used for connection establishment

M/O	PDU	Contents
M	LMP_host_connection_req	—
M	LMP_setup_complete	—

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 208 of 1169

9.5 Summary of PDUs

Table 67 provides the coding of the different LM PDUs.

Table 67—Coding of the different LM PDUs

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_accepted	2	3	DM1/ DV	m ↔ s	op code	2
LMP_au_rand	17	11	DM1	m ↔ s	random number	2–17
LMP_auto_rate	1	35	DM1/ DV	m ↔ s	—	
LMP_clkoffset_req	1	5	DM1/ DV	m → s	—	
LMP_clkoffset_res	3	6	DM1/ DV	m ← s	clock offset	2–3
LMP_comb_key	17	9	DM1	m ↔ s	random number	2–17
LMP_decr_power_req	2	32	DM1/ DV	m ↔ s	for future use	2
LMP_detach	2	7	DM1/ DV	m ↔ s	reason	2
LMP_encryption_key_size_req	2	16	DM1/ DV	m ↔ s	key size	2
LMP_encryption_mode_ req	2	15	DM1/ DV	m ↔ s	encryption mode	2
LMP_features_req	9	39	DM1/ DV	m ↔ s	features	2–9
LMP_features_res	9	40	DM1/ DV	m ↔ s	features	2–9
LMP_host_connection_req	1	51	DM1/ DV	m ↔ s	—	
LMP_hold	7	20	DM1/ DV	m ↔ s	hold time, hold instant	4–7
LMP_hold_req	7	21	DM1/ DV	m ↔ s	hold time, hold instant	4–7
LMP_incr_power_req	2	31	DM1/ DV	m ↔ s	for future use	2
LMP_in_rand	17	8	DM1	m ↔ s	random number	2–17
LMP_max_power	1	33	DM1/ DV	m ↔ s	—	
LMP_max_slot	2	45	DM1/ DV	m ↔ s	max slots	2

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 209 of 1169

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_max_slot_req	2	46	DM1/ DV	m ↔ s	max slots	2
LMP_min_power	1	34	DM1/ DV	m ↔ s	—	
LMP_modify_beacon	11 or 13	28	DM1	m → s	timing control flags	2
					D_B	3–4
					T_B	5–6
					N_B	7
					Δ_B	8
					D_{access}	9
					T_{access}	10
					$N_{acc\text{-}slots}$	11
					N_{poll}	12
					M_{access}	13:0–3
					access scheme	13:4–7
LMP_name_req	2	1	DM1/ DV	m ↔ s	name offset	2
LMP_name_res	17	2	DM1	m ↔ s	name offset	2
					name length	3
					name fragment	4–17
LMP_not_accepted	3	4	DM1/ DV	m ↔ s	op code	2
					reason	3
LMP_page_mode_req	3	53	DM1/ DV	m ↔ s	paging scheme	2
					paging scheme settings	3
LMP_page_scan_mode_req	3	54	DM1/ DV	m ↔ s	paging scheme	2
					paging scheme settings	3

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 210 of 1169

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_park_req	17	25	DM	m → s	timing control flags	2
					D_B	3–4
					T_B	5–6
					N_B	7
					Δ_B	8
					PM_ADDR	9
					AR_ADDR	10
					N_{Bsleep}	11
					D_{Bsleep}	12
					D_{access}	13
					T_{access}	14
					$N_{acc\text{-}slots}$	15
					N_{poll}	16
					M_{access}	17:0–3
					access scheme	17:4–7
LMP_preferred_rate	2	36	DM1/ DV	m ↔ s	data rate	2
LMP_quality_of_service	4	41	DM1/ DV	m → s	poll interval	2–3
					N_{BC}	4
LMP_quality_of_service_req	4	42	DM1/ DV	m ↔ s	poll interval	2–3
					N_{BC}	4
LMP_remove_SCO_link_req	3	44	DM1/ DV	m ↔ s	SCO handle	2
					reason	3
LMP_SCO_link_req	7	43	DM1/ DV	m ↔ s	SCO handle	2
					timing control flags	3
					D_{sco}	4
					T_{sco}	5
					SCO packet	6
					air mode	7
LMP_set_broadcast_ scan_window	4 or 6	27	DM1	m → s	timing control flags	2
					D_B	3–4
					broadcast scan window	5–6

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 211 of 1169

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_setup_complete	1	49	DM1	m ↔ s	—	
LMP_slot_offset	9	52	DM1/ DV	m ↔ s	slot offset	2–3
					BD_ADDR	4–9
LMP_sniff_req	10	23	DM1	m ↔ s	timing control flags	2
					D_{sniff}	3–4
					T_{sniff}	5–6
					sniff attempt	7–8
					sniff timeout	9–10
LMP_sres	5	12	DM1/ DV	m ↔ s	authentication response	2–5
LMP_start_encryption_req	17	17	DM1	m → s	random number	2–17
LMP_stop_encryption_req.	1	18	DM1/ DV	m → s	—	
LMP_supervision_timeout	3	55	DM1/ DV	m →s	supervision timeout	2–3
LMP_switch_req	5	19	DM1/ DV	m ↔ s	switch instant	2–5
LMP_temp_rand	17	13	DM1	m → s	random number	2–17
LMP_temp_key	17	14	DM1	m → s	key	2–17
LMP_timing_accuracy_req	1	47	DM1/ DV	m ↔ s	—	
LMP_timing_accuracy_res	3	48	DM1/ DV	m ↔ s	drift	2
					jitter	3
LMP_unit_key	17	10	DM1	m ↔ s	key	2–17
LMP_unpark_BD_ADDR_req	variable	29	DM1	m → s	timing control flags	2
					D_B	3–4
					AM_ADDR 1st unpark	5:0–2
					AM_ADDR 2nd unpark	5:4–6
					BD_ADDR 1st unpark	6–11
					BD_ADDR 2nd unpark	12–17

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 212 of 1169

Table 67—Coding of the different LM PDUs *(continued)*

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_unpark_PM_ADDR_req	variable	30	DM1	m → s	timing control flags	2
					D_B	3–4
					AM_ADDR 1st unpark	5:0–3
					AM_ADDR 2nd unpark	5:4–7
					PM_ADDR 1st unpark	6
					PM_ADDR 2nd unpark	7
LMP_unsniff_req	1	24	DM1/ DV	m ↔ s	—	
LMP_use_semi_ permanent_key	1	50	DM1/ DV	m → s	—	
LMP_version_req	6	37	DM1/ DV	m ↔ s	VersNr	2
					CompId	3–4
					SubVersNr	5–6
LMP_version_res	6	38	DM1/ DV	m ↔ s	VersNr	2
					CompId	3–4
					SubVersNr	5–6

Note 1—For LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_ADDR_req, and LMP_unpark_PM_ADDR_req, the parameter D_B is optional. This parameter is only present if bit 0 of *timing control flags* is 1. If the parameter is not included, the position in payload for all parameters following D_B are decreased by 2.

Note 2—For LMP_unpark_BD_ADDR, the AM_ADDR and the BD_ADDR of the 2nd unparked slave are optional. If only one slave is unparked, "AM_ADDR 2nd unpark" should be zero and "BD_ADDR 2nd unpark" is left out.

Note 3—For LMP_unpark_PM_ADDR, the AM_ADDR and the PM_ADDR of the 2nd through 7th unparked slaves are optional. If N slaves are unparked, the fields up to and including the N^{th} unparked slave are present. If N is odd, the "AM_ADDR (N+1)th unpark" shall be zero. The length of the message is $x + 3N/2$ if N is even and $x + 3(N+1)/2 - 1$ if N is odd, where $x = 2$ or 4 depending on if the D_B is included or not (see Note 1).

Copyright © 2002 IEEE. All rights reserved.

9.5.1 Description of parameters

Table 68 contains the parameters that appear in the various LM_PDUs shown in Table 67.

Table 68—Parameters in LM PDUs

Name	Length (bytes)	Type	Unit	Detailed
access scheme	1	u_int4		0: polling technique 1–15: Reserved
air mode	1	u_int8		0: μ-law log 1: A-law log 2: CVSD 3: transparent data 4–255: Reserved
AM_ADDR	1	u_int4		
AR_ADDR	1	u_int8		
authentication response	4	multiple bytes		
BD_ADDR	6	multiple bytes		
broadcast scan window	2	u_int16	slots	
clock offset	2	u_int16	1.25ms	($CLKN_{16-2}$ slave - $CLKN_{16-2}$ master) mod 2^{15} (MSbit of second byte not used)
CompId	2	u_int16		See 2.4.3
D_{access}	1	u_int8	slots	
D_B	2	u_int16	slots	
D_{Bsleep}	1	u_int8	slots	
data rate	1	u_int8		bit0 = 0: use FEC bit0 = 1: do not use FEC bit1-2=0: No packet-size preference available bit1-2=1: use 1-slot packets bit1-2=2: use 3-slot packets bit1-2=3: use 5-slot packets bit3-7: Reserved
drift	1	u_int8	ppm	
D_{sco}	1	u_int8	slots	
D_{sniff}	2	u_int16	slots	
encryption mode	1	u_int8		0: no encryption 1: point-to-point encryption 2: point-to-point and broadcast encryption 3–255: Reserved

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 214 of 1169

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed
features	8	multiple bytes		See Table 69
hold instant	4	u_int32	slots	Bits 27:1 of the master Bluetooth clock value
hold time	2	u_int16	slots	
jitter	1	u_int8	µs	
key	16	multiple bytes		
key size	1	u_int8	byte	
M_{access}	1	u_int4		number of access windows
max slots	1	u_int8	slots	
$N_{acc\text{-}slots}$	1	u_int8	slots	
name fragment	14	multiple bytes		UTF-8 characters.
name length	1	u_int8	bytes	
name offset	1	u_int8	bytes	
N_B	1	u_int8		
N_{BC}	1	u_int8		
N_{Bsleep}	1	u_int8	slots	
N_{poll}	1	u_int8	slots	
op code	1	u_int8		
paging scheme	1	u_int8		0: mandatory scheme 1: optional scheme I 2: optional scheme II 3: optional scheme III 4–255: Reserved
paging scheme settings	1	u_int8		For mandatory scheme: 0: R0 1: R1 2: R2 3–255: Reserved For optional scheme 1: 0: Reserved 1: R1 2: R2 3–255: Reserved
PM_ADDR	1	u_int8		
poll interval	2	u_int16	slots	
random number	16	multiple bytes		
reason	1	u_int8		See Table 70
SCO handle	1	u_int8		

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 215 of 1169

Table 68—Parameters in LM PDUs *(continued)*

Name	Length (bytes)	Type	Unit	Detailed
SCO packet	1	u_int8		0: HV1 1: HV2 2: HV3 3–255: Reserved
slot offset	2	u_int16	µs	0 ≤ slot offset < 1250
sniff attempt	2	u_int16	slots	Number of receive slots
sniff timeout	2	u_int16	slots	Number of receive slots
SubVersNr	2	u_int16		Defined by each company
supervision timeout	2	u_int16	slots	0 means an infinite timeout
switch instant	4	u_int32	slots	Bits 27:1 of the master Bluetooth clock value
T_{access}	1	u_int8	slots	
T_B	2	u_int16	slots	
timing control flags	1	u_int8		bit0 = 0: no timing change bit0 = 1: timing change bit1 = 0: use initialization 1 bit1 = 1: use initialization 2 bit2 = 0: access window bit2 = 1: no access window bit3–7: Reserved
T_{sco}	1	u_int8	slots	
T_{sniff}	2	u_int16	slots	
VersNr	1	u_int8		See 2.4.3
Δ_B	1	u_int8	slots	

9.5.1.1 Coding of features

This parameter is a bitmap with information about the Bluetooth radio, baseband, and LMP features which a device supports. The bit shall be one if the feature is supported. In addition to the bitmap information the feature parameter has a 3-bit field denoted flow control lag. This is defined as the total amount of L2CAP data that can be sent following the receipt of a valid payload header with the payload header flow bit set to 0 and is in units of 256 bytes (see 8.4.5.2). The feature parameter bits that are not defined in Table 69 shall be 0.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 216 of 1169

Table 69—Coding of the parameter features

Byte	Bit	Supported feature
0	0	3–slot packets
	1	5–slot packets
	2	encryption
	3	slot offset
	4	timing accuracy
	5	switch
	6	hold mode
	7	sniff mode
1	0	park mode
	1	RSSI
	2	channel quality driven data rate
	3	SCO link
	4	HV2 packets
	5	HV3 packets
	6	u-law log
	7	A-law log
2	0	CVSD
	1	paging scheme
	2	power control
	3	transparent SCO data
	4	Flow control lag (bit0)
	5	Flow control lag (bit1)
	6	Flow control lag (bit2)

9.5.1.2 List of error reasons

Table 70 contains the codes of the different error reasons used in LMP.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 217 of 1169

Table 70—List of error reasons

Reason	Description
0x05	Authentication failure
0x06	Key missing
0x0A	Max number of SCO connections to a device. (The maximum number of SCO connections to a particle device has been reached. All allowed SCO connection handles to that device are used.)
0x0D	Host rejected due to limited resources. (The host at the remote side has rejected the connection because the remote host did not have enough additional resources to accept the connection.)
0x0E	Host rejected due to security reasons. (The host at the remote side has rejected the connection because the remote host determined that the local host did not meet its security criteria.)
0x0F	Host rejected due to remote device is only a personal device. (The host at the remote side has rejected the connection because the remote host is a personal device and will only accept the connection from one particle remote host.)
0x10	Host timeout. (Used at connection accept timeout, the host did not respond to an incoming connection attempt before the connection accept timer expired.)
0x13	Other end terminated connection: User ended connection
0x14	Other end terminated connection: Low resources
0x15	Other end terminated connection: About to power off
0x16	Connection terminated by Local Host
0x17	Repeated attempts. (An authentication or pairing attempt is made too soon after a previously failed authentication or pairing attempt.)
0x18	Pairing not allowed.
0x19	Unknown LMP PDU
0x1A	Unsupported LMP feature
0x1B	SCO offset rejected
0x1C	SCO interval rejected
0x1D	SCO air mode rejected
0x1E	Invalid LMP parameters
0x1F	Unspecified error
0x20	Unsupported parameter value
0x21	Switch not allowed
0x23	LMP error transaction collision
0x24	PDU not allowed
0x25	Encryption mode not acceptable
0x26	Unit key used
0x27	QoS not supported
0x28	Instant passed
0x29	Pairing with unit key not supported

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 218 of 1169

9.5.2 Default values

The Bluetooth device shall use the values in Table 71 before anything else has been negotiated:

Table 71—Default values

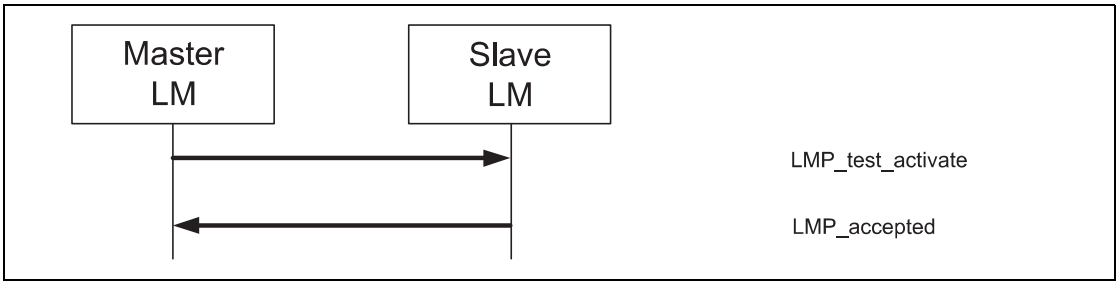
Parameter	Value
drift	250
jitter	10
max slots	1
poll interval	40

9.6 Test modes

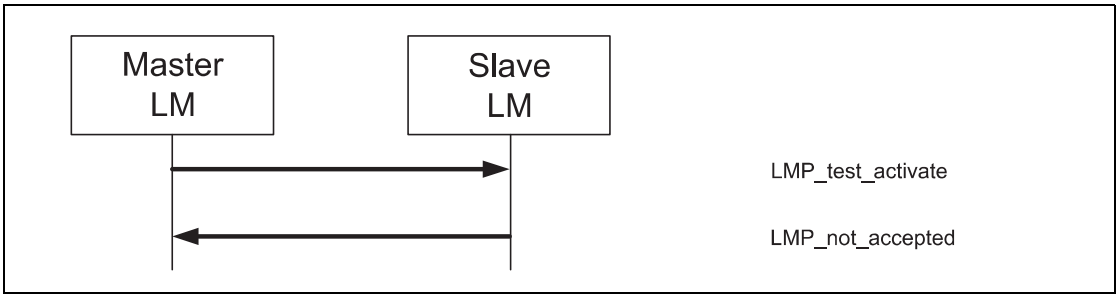
LMP has PDUs to support different Bluetooth test modes, which are used for certification and compliance testing of the Bluetooth radio and baseband. See Annex E for a detailed description of these test modes.

9.6.1 Activation and deactivation of test mode

The test mode is activated by sending LMP_test_activate to the device under test (DUT). The DUT is always the slave. The link manager shall be able to receive this message anytime. If entering test mode is locally enabled in the DUT it responds with LMP_accepted and test mode is entered. Otherwise, the DUT responds with LMP_not_accepted and the DUT remains in normal operation. The reason code in LMP_not_accepted shall be *PDU not allowed*.



Sequence 59: Activation of test mode successful.



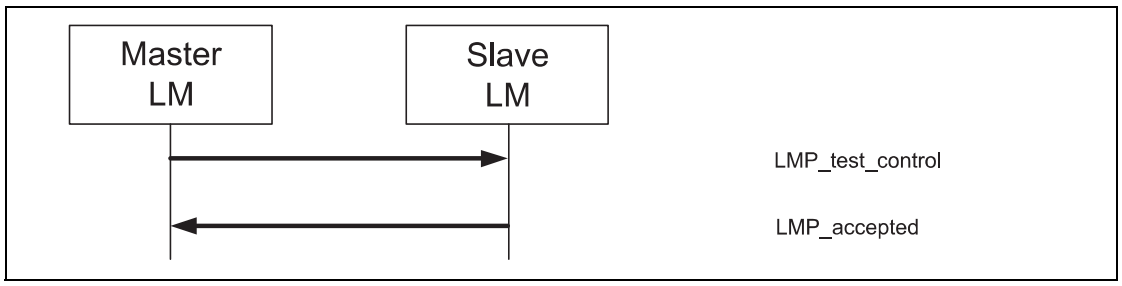
Sequence 60: Activation of test mode fails. Slave is not allowed to enter test mode.

Copyright © 2002 IEEE. All rights reserved.

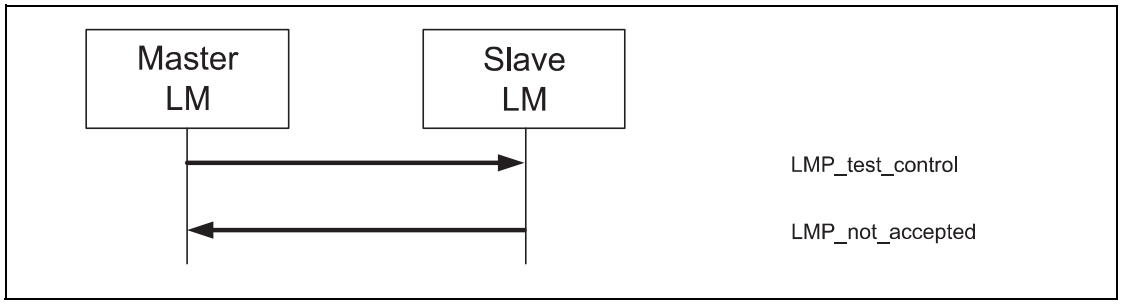
The test mode can be deactivated in two ways. Sending LMP_test_control with the test scenario set to "exit test mode" exits the test mode and the slave returns to normal operation still connected to the master. Sending LMP_detach to the DUT ends the test mode and the connection.

9.6.2 Control of test mode

When the DUT has entered test mode, the PDU LMP_test_control can be sent to the DUT to start a specific test. This PDU is acknowledged with LMP_accepted. If a device that is not in test mode receives LMP_test_control it responds with LMP_not_accepted, where the reason code shall be *PDU not allowed*.



Sequence 61: Control of test mode successful.



Sequence 62: Control of test mode rejected since slave is not in test mode.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 220 of 1169

9.6.3 Summary of test mode PDUs

The PDUs used for test purposes are summarized in Table 72. For a detailed description of the parameters, see Table E.2.

Table 72—Test mode PDUs

M/O	LMP PDU	Length	op code	Packet type	Possible direction	Contents	Position in payload
M	LMP_test_activate	1	56	DM1/DV	m → s	—	
M	LMP_test_control	10	57	DM1	m → s	test scenario	2
						hopping mode	3
						TX frequency	4
						RX frequency	5
						power control mode	6
						poll period	7
						packet type	8
						length of test data	9–10

9.7 Error Handling

If the Link Manager receives a PDU with unrecognized opcode, it responds with LMP_not_accepted with the reason code *unknown LMP PDU.* The opcode parameter that is echoed back is the unrecognized opcode.

If the Link Manager receives a PDU with invalid parameters, it responds with LMP_not_accepted with the reason code *invalid LMP parameters*.

If the maximum response time (see 9.1), is exceeded or if a link loss is detected (see 8.10.11), the party that waits for the response shall conclude that the procedure has terminated unsuccessfully.

Erroneous LMP messages can be caused by errors on the channel or systematic errors at the transmit side. To detect the latter case, the LM should monitor the number of erroneous messages and disconnect if it exceeds a threshold, which is implementation-dependent.

Since LMP PDUs are not interpreted in real time, collision situations can occur where both LMs initiate the same procedure and both cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending LMP_not_accepted with the reason code “LMP Error Transaction Collision”. The master-initiated procedure shall then be completed.

When the Link Manager receives a PDU that is not allowed, if the PDU normally expects a PDU reply, for example LMP_host_connection_req or LMP_unit_key, the PDU LMP_not_accepted with the reason code “PDU not allowed” will be returned. If the PDU normally does not expect a reply, for example LMP_sres or LMP_temp_key, the PDU will be ignored.

Copyright © 2002 IEEE. All rights reserved.

10. Logical Link Control and Adaptation Protocol Specification

Figure 94 indicates the relationship of the Bluetooth protocol stack to this clause. This clause describes the Bluetooth logical link control and adaptation protocol (L2CAP). This protocol supports higher level protocol multiplexing, packet segmentation, and reassembly, and the conveying of QoS information. It describes the protocol state machine, packet format, and composition.

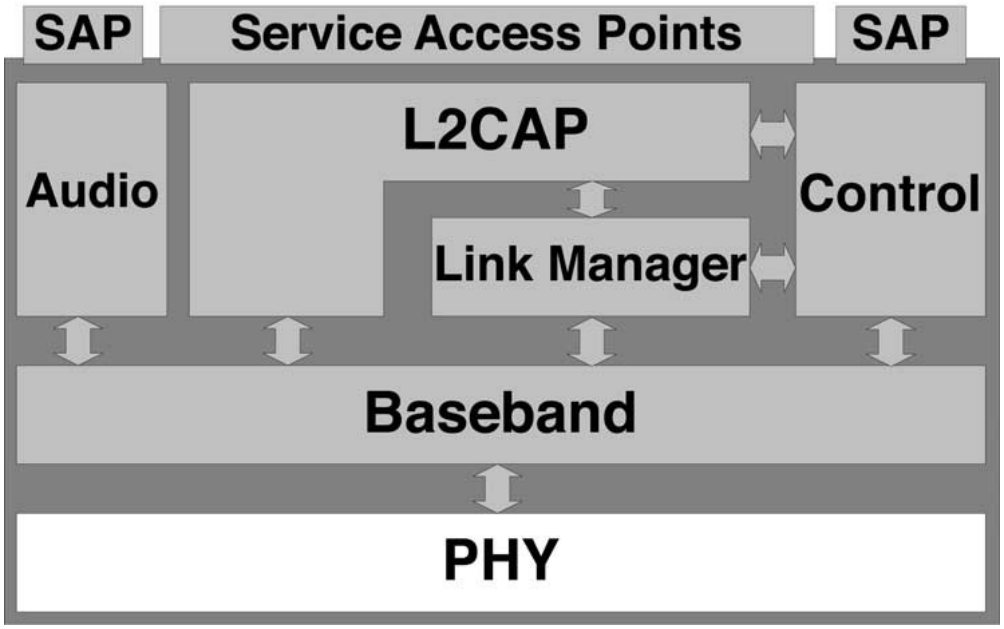


Figure 94—L2CAP interface relationships

10.1 Introduction

This section of the Bluetooth Specification defines the logical link control and adaptation layer protocol, referred to as L2CAP. L2CAP is layered over the baseband protocol and resides in the data link layer as shown in Figure 95. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets up to 64 kilobytes in length.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 222 of 1169

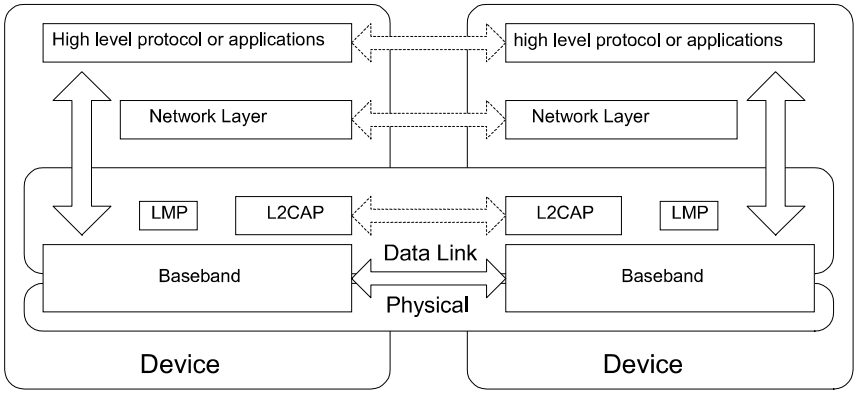


Figure 95—L2CAP within protocol layers

The Baseband specification (Clause 8) defines two link types: SCO links and ACL links. SCO links support real-time voice traffic using reserved bandwidth. ACL links support best effort traffic. The L2CAP specification is defined for only ACL links and no support for SCO links is planned.

For ACL links, use of the AUX1 packet on the ACL link is prohibited. This packet type supports no data integrity checks (no CRC). Because L2CAP depends on integrity checks in the Baseband to protect the transmitted information, AUX1 packets shall never be used to transport L2CAP packets.

The format of the ACL payload header is shown below. Figure 96 displays the payload header used for single-slot packets, and Figure 97 displays the header used in multislot packets. The only difference is the size of the length field. The packet type (a field in the baseband header) distinguishes single-slot packets from multislot packets.

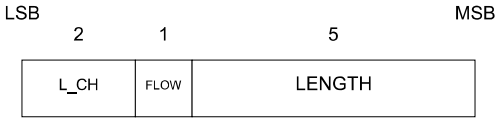


Figure 96—ACL payload header for single-slot packets

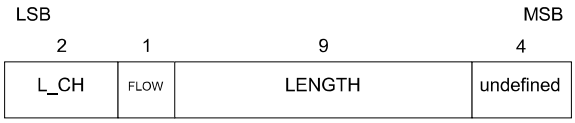


Figure 97—ACL payload header for multislot packets

The 2-bit logical channel (L_CH) field, defined in Table 73, distinguishes L2CAP packets from Link Manager Protocol () packets. The remaining code is reserved for future use.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 223 of 1169

Table 73—Logical channel L_CH field contents

L_CH code	Logical Channel	Information
00	RESERVED	Reserved for future use
01	L2CAP	Continuation of L2CAP packet
10	L2CAP	Start of L2CAP packet
11	LMP	Link Manager Protocol

The FLOW bit in the ACL header is managed by the LC, a baseband implementation entity, and is normally set to1 (“flow on”). It is set to 0 (“flow off”) when no further L2CAP traffic shall be sent over the ACL link. Sending an L2CAP packet with the FLOW bit set to 1 resumes the flow of incoming L2CAP packets. This is described in more detail in Clause 8.

10.1.1 L2CAP functional requirements

The functional requirements for L2CAP include protocol multiplexing, segmentation, and reassembly (SAR), and group management. Figure 98 illustrates how L2CAP fits into the Bluetooth Protocol Stack. L2CAP lies above the Baseband Protocol in Clause 8 and interfaces with other communication protocols such as the Bluetooth Service Discovery Protocol (SDP), RFCOMM, and Telephony Control (TCS). Voice-quality channels for audio and telephony applications are usually run over Baseband SCO links. Packetized audio data, such as IP Telephony, may be sent using communication protocols running over L2CAP.

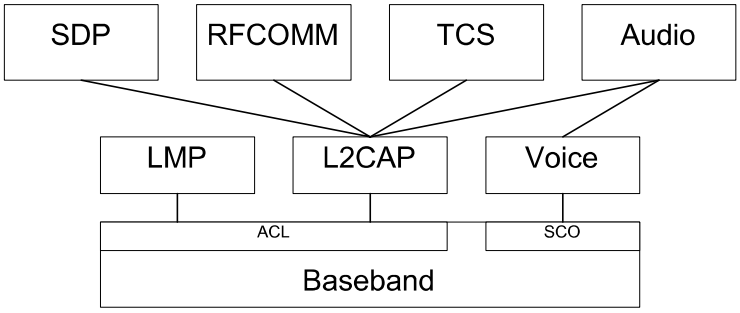


Figure 98—L2CAP in Bluetooth protocol architecture

Essential protocol requirements for L2CAP include simplicity and low overhead. Implementations of L2CAP should be applicable for devices with limited computational resources. L2CAP should not consume excessive power since that significantly sacrifices power efficiency achieved by the Bluetooth radio. Memory requirements for protocol implementation should also be kept to a minimum.

The protocol complexity should be acceptable to personal computers, PDAs, digital cellular phones, wireless headsets, joysticks and other wireless devices supported by Bluetooth. Furthermore, the protocol should be designed to achieve reasonably high bandwidth efficiency.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 224 of 1169

— *Protocol Multiplexing*

L2CAP supports protocol multiplexing because the Baseband Protocol does not support any "type" field identifying the higher layer protocol being multiplexed above it. L2CAP distinguishes between upper layer protocols such as the Service Discovery Protocol (2.4.2), RFCOMM (2.4.2), and Telephony Control (2.4.2).

— *Segmentation and Reassembly*

Compared to other wired physical media, the data packets defined by the Baseband Protocol (Clause 8) are limited in size. Exporting a maximum transmission unit (MTU) associated with the largest Baseband payload (341 bytes for DH5 packets) limits the efficient use of bandwidth for higher layer protocols that are designed to use larger packets. Large L2CAP packets must be segmented into multiple smaller Baseband packets prior to their transmission over the air. Similarly, multiple received Baseband packets may be reassembled into a single larger L2CAP packet following a simple integrity check (described in 10.2.4.2). The segmentation and reassembly (SAR) functionality is absolutely necessary to support protocols using packets larger than those supported by the Baseband.

— *Quality of Service*

The L2CAP connection establishment process allows the exchange of information regarding the QoS expected between two Bluetooth units. Each L2CAP implementation must monitor the resources used by the protocol and ensure that QoS contracts are honored.

— *Groups*

Many protocols include the concept of a group of addresses. The Baseband Protocol supports the concept of a piconet, a group of devices synchronously hopping together using the same clock. The L2CAP group abstraction permits implementations to efficiently map protocol groups on to piconets. Without a group abstraction, higher level protocols would need to be exposed to the Baseband Protocol and Link Manager functionality in order to manage groups efficiently.

10.1.2 Assumptions

The protocol is designed based on the following assumptions:

1) The ACL link between two units is set up using the Link Manager Protocol (Clause 9). The Baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than one ACL link exists between any two devices.

2) The Baseband always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.

3) L2CAP provides a reliable channel using the mechanisms available at the Baseband layer. The Baseband always performs data integrity checks when requested and resends data until it has been successfully acknowledged or a timeout occurs. Because acknowledgements may be lost, timeouts may occur even after the data has been successfully sent. The Baseband protocol uses a 1-bit sequence number that removes duplicates. Note that the use of Baseband broadcast packets is prohibited if reliability is required since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.

Copyright © 2002 IEEE. All rights reserved.

10.1.3 Features not supported

The following features are outside the scope of L2CAP's responsibilities:

— L2CAP does not transport audio designated for SCO links.

— L2CAP does not enforce a reliable channel or ensure data integrity, that is, L2CAP performs no retransmissions or checksum calculations.

— L2CAP does not support a reliable multicast channel (see 10.4.2).

— L2CAP does not support the concept of a global group name.

10.2 General operation

The L2CAP is based around the concept of *"channels"*. Each one of the end-points of an L2CAP channel is referred to by a *channel identifier.*

10.2.1 Channel identifiers

Channel identifiers (CIDs) are local names representing a logical channel end-point on the device. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. The null identifier (0x0000) is defined as an illegal identifier and shall never be used as a destination end-point. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that the same CID is not reused as a local L2CAP channel endpoint for multiple simultaneous L2CAP channels between a local device and some remote device. Table 74 summarizes the definition and partitioning of the CID name space.

CID assignment is relative to a particular device and a device can assign CIDs independently from other devices (unless it needs to use any of the reserved CIDs shown in Table 74). Thus, even if the same CID value has been assigned to (remote) channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device.

Table 74—CID definitions

CID	Description
0x0000	Null identifier
0x0001	Signalling channel
0x0002	Connectionless reception channel
0x0003–0x003F	Reserved
0x0040–0xFFFF	Dynamically allocated

10.2.2 Operation between devices

Figure 99 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel "group" where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signalling

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 226 of 1169

channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of these channels. Support for a signalling channel within an L2CAP entity is mandatory. Another CID is reserved for all incoming connectionless data traffic. In the example below, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this channel ID is directed to the remote channel reserved for connectionless data traffic.

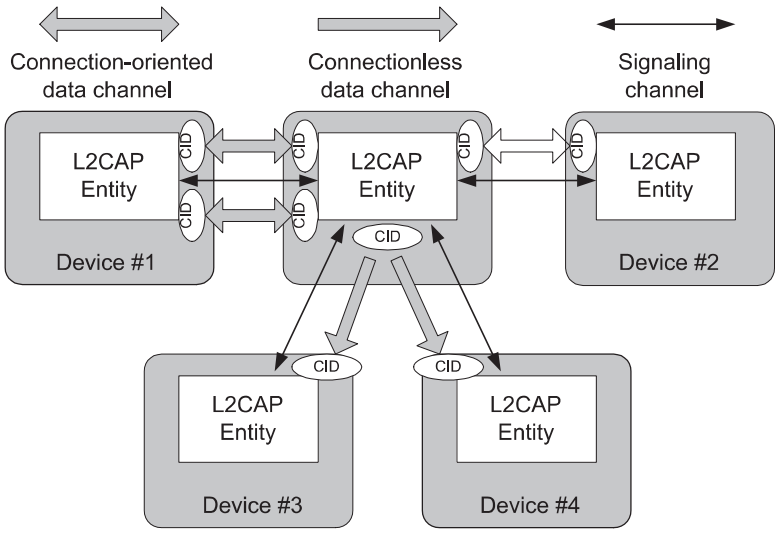


Figure 99—Channels between devices

Table 75 describes the various channels and their source and destination identifiers. An "allocated" channel is created to represent the local endpoint and should be in the range 0x0040 to 0xFFFF. 10.3 describes the state machine associated with each connection-orientedchannel. 10.4.1 describes the packet format associated with bi-directional channels and 10.4.2 describes the packet format associated with uni-directional channels.

Table 75—Types of channel identifiers

Channel type	Local CID	Remote CID
Connection-oriented	Dynamically allocated	Dynamically allocated
Connectionless data	Dynamically allocated	0x0002 (fixed)
Signalling	0x0001 (fixed)	0x0001 (fixed)

10.2.3 Operation between layers

L2CAP implementations should follow the general architecture described in Figure 100. L2CAP implementations must transfer data between higher layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation must also support a set of signalling commands for use between L2CAP implementations. L2CAP implementations should also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is an implementation-dependent process.

Copyright © 2002 IEEE. All rights reserved.

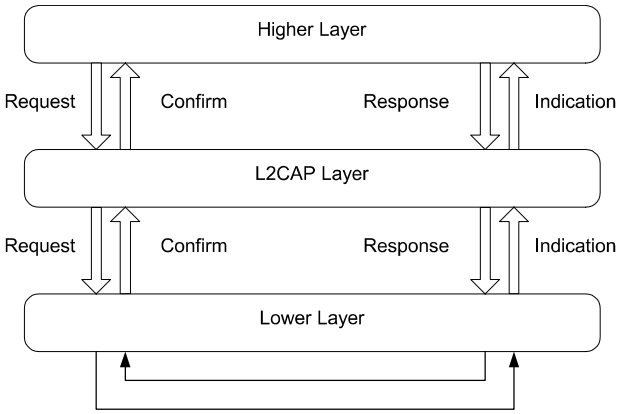


Figure 100—L2CAP architecture

10.2.4 Segmentation and reassembly

SAR operations are used to improve efficiency by supporting a maximum transmission unit (MTU) size larger than the largest baseband packet. This reduces overhead by spreading the network and transport packets used by higher layer protocols over several baseband packets. All L2CAP packets may be segmented for transfer over baseband packets. The protocol does not perform any segmentation and reassembly operations but the packet format supports adaptation to smaller physical frame sizes. An L2CAP implementation exposes the outgoing (i.e., the remote host's receiving) MTU and segments higher layer packets into "chunks" that can be passed to the Link Manager via the HCI, whenever one exists. On the receiving side, an L2CAP implementation receives "chunks" from the HCI and reassembles those chunks into L2CAP packets using information provided through the HCI and from the packet header (see Figure 101).

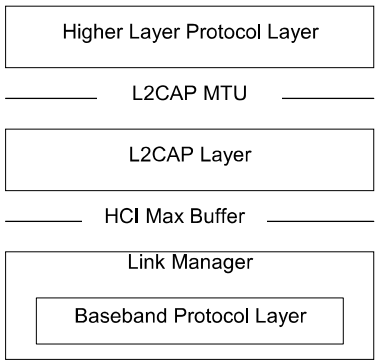


Figure 101—L2CAP SAR variables

Segmentation and reassembly is implemented using very little overhead in baseband packets. The two L_CH bits defined in the first byte of baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP packets. L_CH shall be "10" for the first segment in an L2CAP packet and "01"' for a continuation segment. An example use of SAR is shown in Figure 102.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 228 of 1169

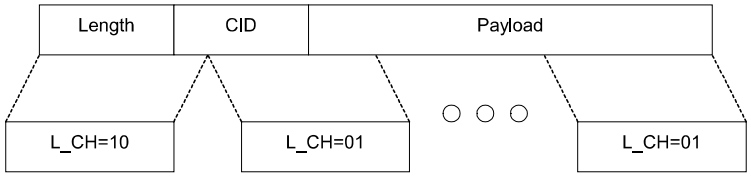


Figure 102—L2CAP segmentation

10.2.4.1 Segmentation procedures

The L2CAP maximum transmission unit (MTU) will be exported using an implementation-specific service interface. It is the responsibility of the higher layer protocol to limit the size of packets sent to the L2CAP layer below the MTU limit. An L2CAP implementation will segment the packet into protocol data units (PDUs) to send to the lower layer. If L2CAP runs directly over the Baseband Protocol, an implementation may segment the packet into Baseband packets for transmission over the air. If L2CAP runs above the host controller interface (typical scenario), an implementation may send block-sized chunks to the host controller where they will be converted into Baseband packets. All L2CAP segments associated with an L2CAP packet shall be passed through to the Baseband before any other L2CAP packet destined to the same unit may be sent.

10.2.4.2 Reassembly procedures

The Baseband Protocol delivers ACL packets in sequence and protects the integrity of the data using a 16-bit CRC. The Baseband also supports reliable connections using an automatic repeat request (ARQ) mechanism. As the Baseband controller receives ACL packets, it either signals the L2CAP layer on the arrival of each Baseband packets, or accumulates a number of packets before the receive buffer fills up or a timer expires before signalling the L2CAP layer.

L2CAP implementations shall use the length field in the header of L2CAP packets (see 10.4), as a consistency check and discard any L2CAP packets that fail to match the length field. If channel reliability is not needed, packets with improper lengths may be silently discarded. For reliable channels, L2CAP implementations shall indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as specified in 10.6.2.

Figure 103 illustrates the use of segmentation and reassembly operations to transmit a single higher layer PDU.[17] Note that while there is a one-to-one mapping between a high layer PDU and an L2CAP packet, the segment size used by the segmentation and reassembly routines is left to the implementation and may differ from the sender to the receiver.

[17]For simplicity, the stripping of any additional HCI and USB specific information fields prior to the creation of the baseband packets (Air_1, Air_2, etc.) is not shown in Figure 103.

Copyright © 2002 IEEE. All rights reserved.

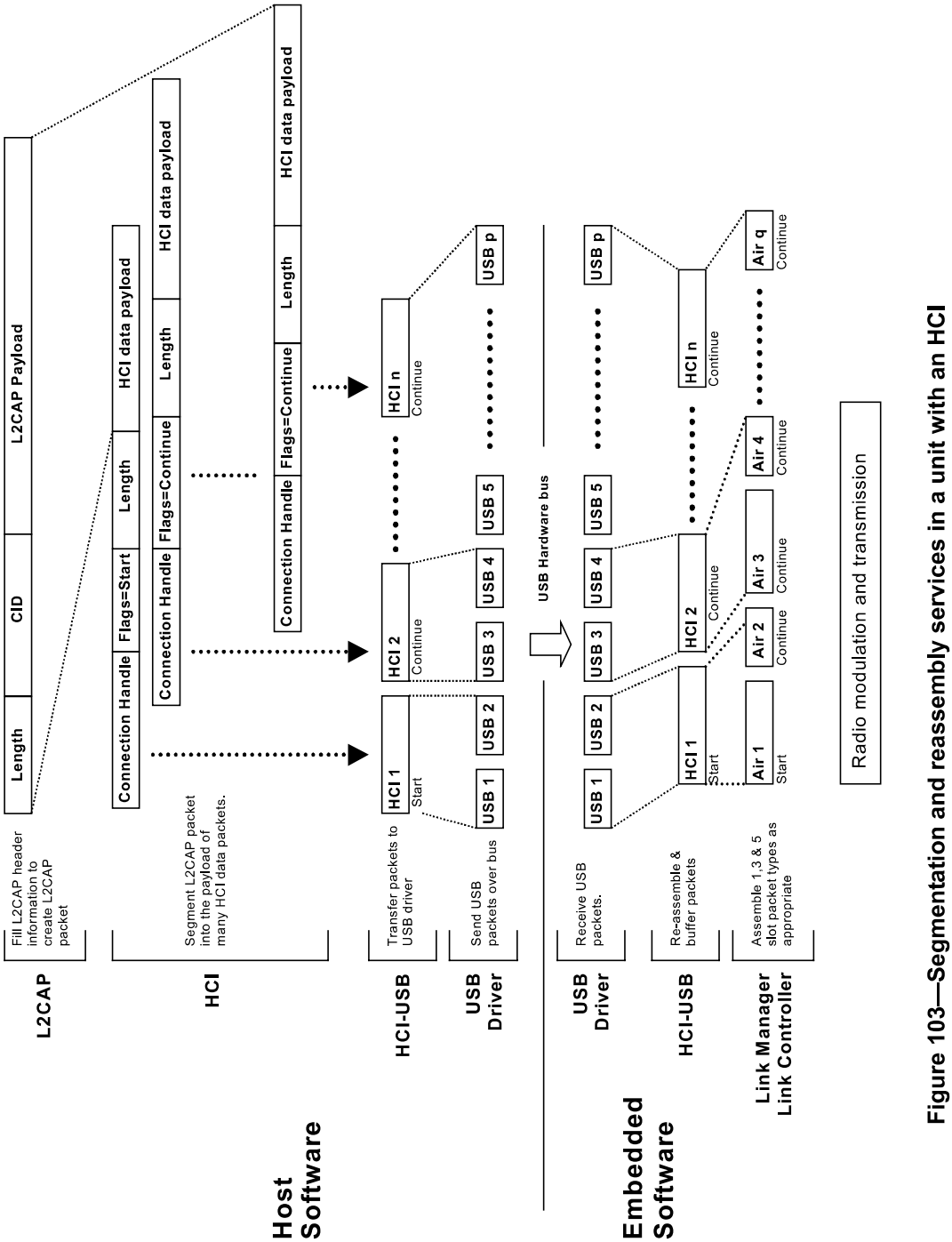


Figure 103—Segmentation and reassembly services in a unit with an HCI

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 230 of 1169

10.3 State machine

This subclause describes the L2CAP connection-oriented channel state machine. The subclause defines the states, the events causing state transitions, and the actions to be performed in response to events. This state machine is only pertinent to bi-directional CIDs and is not representative of the signalling channel or the uni-directional channel.

Figure 104 illustrates the events and actions performed by an implementation of the L2CAP layer. Client and Server simply represent the initiator of the request and the acceptor of the request respectively. An application-level Client would both initiate and accept requests. The naming convention is as follows. The interface between two layers (vertical interface) uses the prefix of the lower layer offering the service to the higher layer, e.g., L2CA. The interface between two entities of the same layer (horizontal interface) uses the prefix of the protocol (adding a P to the layer identification), e.g., L2CAP. Events coming from above are called Requests (Req) and the corresponding replies are called Confirms (Cfm). Events coming from below are called Indications (Ind) and the corresponding replies are called Responses (Rsp). Responses requiring further processing are called Pending (Pnd). The notation for Confirms and Responses assumes positive replies. Negative replies are denoted by a "Neg" suffix such as L2CAP_ConnectCfmNeg.

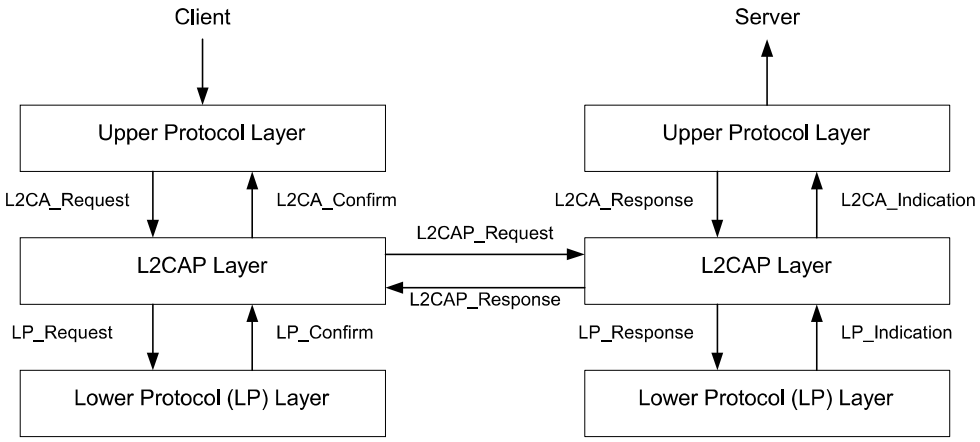


Figure 104—L2CAP layer interactions

While Requests for an action always result in a corresponding Confirmation (for the successful or unsuccessful satisfaction of the action), Indications do not always result into corresponding Responses. The latter is especially true, if the Indications are informative about locally triggered events, e.g., seeing the *LP_QoSViolationInd* in 10.3.1.1, or *L2CA_TimeOutInd* in 10.3.2.4.

Figure 105 uses a message sequence chart (MSC) to illustrate the normal sequence of events. The two outer vertical lines represent the L2CA interface on the initiator (the device issuing a request) and the acceptor (the device responding to the initiator's request). Request commands at the L2CA interface result in Requests defined by the protocol. When the protocol communicates the request to the acceptor, the remote L2CA entity presents the upper protocol with an Indication. When the acceptor's upper protocol responds, the response is packaged by the protocol and communicated back to the initiator. The result is passed back to the initiator's upper protocol using a Confirm message.

Copyright © 2002 IEEE. All rights reserved.

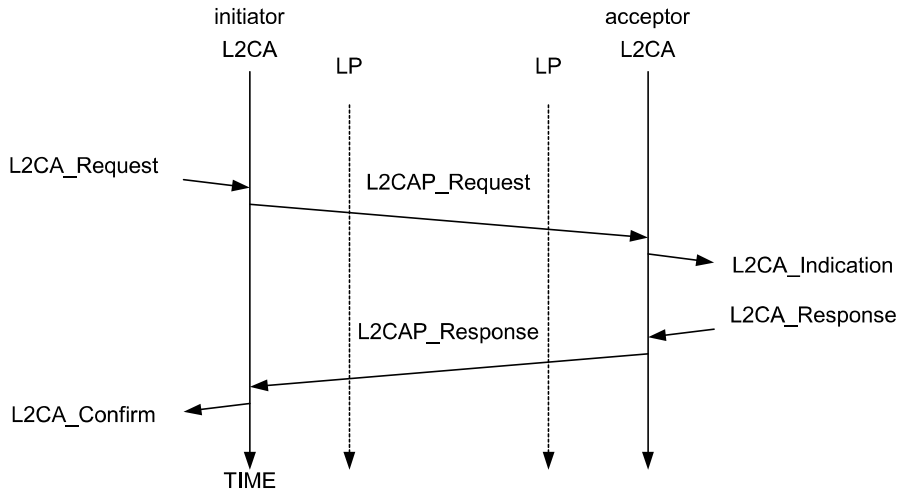


Figure 105—MSC of layer interactions

10.3.1 Events

Events are all incoming messages to the L2CAP layer along with timeouts. Events are partitioned into five categories: Indications and Confirms from lower layers, Requests and Responses from higher layers, data from peers, signal Requests and Responses from peers, and events caused by timer expirations.

10.3.1.1 Lower-layer protocol (LP) to L2CAP events

— *LP_ConnectCfm*

Confirms the request (see *LP_ConnectReq* in 10.3.2.1) to establish a lower layer (Baseband) connection. This includes passing the authentication challenge if authentication is required to establish the physical link.

— *LP_ConnectCfmNeg*

Confirms the failure of the request (see *LP_ConnectReq* in 10.3.2.1) to establish a lower layer (Baseband) connection failed. This could be because the device could not be contacted, refused the request, or the LMP authentication challenge failed.

— *LP_ConnectInd*

Indicates the lower protocol has successfully established connection. In the case of the Baseband, this will be an ACL link. An L2CAP entity may use this information to keep track of what physical links exist.

— *LP_DisconnectInd*

Indicates the lower protocol (Baseband) has been shut down by LMP commands or a timeout event.

— *LP_QoSCfm*

Confirms the request (see *LP_QoSReq* in 10.3.2.1) for a given QoS.

— *LP_QoSCfmNeg*

Confirms the failure of the request (see *LP_QoSReq* in 10.3.2.1) for a given QoS.

— *LP_QoSViolationInd*

Indicates the lower protocol has detected a violation of the QoS agreement specified in the previous *LP_QoSReq* (see 10.3.2.1).

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 232 of 1169

10.3.1.2 L2CAP to L2CAP signalling events

L2CAP to L2CAP signalling events are generated by each L2CAP entity following the exchange of the corresponding L2CAP signalling PDUs (see 10.5). L2CAP signalling PDUs, like any other L2CAP PDUs, are received from a lower layer via a lower protocol indication event. For simplicity of the presentation, we avoid a detailed description of this process, and we assume that signalling events are exchanged directly between the L2CAP peer entities as shown in Figure 104.

— *L2CAP_ConnectReq*

A Connection Request packet has been received.

— *L2CAP_ConnectRsp*

A Connection Response packet has been received with a positive result indicating that the connection has been established.

— *L2CAP_ConnectRspPnd*

A Connection Response packet has been received indicating the remote endpoint has received the request and is processing it.

— *L2CAP_ConnectRspNeg*

A Connection Response packet has been received, indicating that the connection could not be established.

— *L2CAP_ConfigReq*

A Configuration Request packet has been received indicating the remote endpoint wishes to engage in negotiations concerning channel parameters.

— *L2CAP_ConfigRsp*

A Configuration Response packet has been received indicating the remote endpoint agrees with all the parameters being negotiated.

— *L2CAP_ConfigRspNeg*

A Configuration Response packet has been received indicating the remote endpoint does not agree to the parameters received in the response packet.

— *L2CAP_DisconnectReq*

A Disconnection Request packet has been received and the channel shall initiate the disconnection process. Following the completion of an L2CAP channel disconnection process, an L2CAP entity should return the corresponding local CID to the pool of “unassigned” CIDs.

— *L2CAP_DisconnectRsp*

A Disconnection Response packet has been received. Following the receipt of this signal, the receiving L2CAP entity may return the corresponding local CID to the pool of unassigned CIDs. There is no corresponding negative response because the Disconnect Request must succeed.

10.3.1.3 L2CAP to L2CAP data events

— *L2CAP_Data*

A Data packet has been received.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 233 of 1169

10.3.1.4 Upper-layer to L2CAP events

— *L2CA_ConnectReq*

Request from upper layer for the creation of a channel to a remote device.

— *L2CA_ConnectRsp*

Response from upper layer to the indication of a connection request from a remote device (see L2CA_ConnectInd in 10.3.2.4).

— *L2CA_ConnectRspNeg*

Negative response (rejection) from upper layer to the indication of a connection request from a remote device (see *L2CA_ConnectInd* in 10.3.2.4).

— *L2CA_ConfigReq*

Request from upper layer to (re)configure the channel.

— *L2CA_ConfigRsp*

Response from upper layer to the indication of a (re) configuration request (see *L2CA_ConfigInd* in 10.3.2.4).

— *L2CA_ConfigRspNeg*

A negative response from upper layer to the indication of a (re) configuration request (see *L2CA_ConfigInd* in 10.3.2.4).

— *L2CA_DisconnectReq*

Request from upper layer for the immediate disconnection of a channel.

— *L2CA_DisconnectRsp*

Response from upper layer to the indication of a disconnection request (see *L2CA_DisconnectInd* in 10.3.2.4). There is no corresponding negative response, the disconnect indication must always be accepted.

— *L2CA_DataRead*

Request from upper layer for the transfer of received data from L2CAP entity to upper layer.

— *L2CA_DataWrite*

Request from upper layer for the transfer of data from the upper layer to L2CAP entity for transmission over an open channel.

10.3.1.5 Timer events

— *RTX*

The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signalling requests. This timer is started when a signalling request (see 10.5) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value shall be reset to a new value at least double the previous value.

Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before terminating the channel identified by the Requests. The one exception is the signalling CID that should never be terminated. The decision should be based on the flush timeout of the signalling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level. For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level. When terminating the channel, it is not

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 234 of 1169

necessary to send a L2CAP DisconnectReq and enter disconnection state.Channels should be transitioned directly to the Closed state.

The value of this timer is implementation-dependent but the minimum initial value is 1 s and the maximum initial value is 60 s. One RTX timer shall exist for each outstanding signalling request, including each Echo Request. The timer disappears on the final expiration, when the response is received, or the physical link is lost. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 60 s.

— *ERTX*

The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing additional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an *L2CAP_ConnectRspPnd* event is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be disconnected. If a duplicate Request is sent, the particular ERTX timer disappears, replaced by a new RTX timer and the whole timing procedure restarts as described previously for the RTX timer.

The value of this timer is implementation-dependent but the minimum initial value is 60 s and the maximum initial value is 300 s. Similar to RTX, there shall be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 300 s. When terminating the channel, it is not necessary to send a L2CAP DisconnectReq and enter disconnection state. Channels should be transitioned directly to the Closed state.

10.3.2 Actions

Actions are partitioned into five categories: Confirms and Indications to higher layers, Request and Responses to lower layers, Requests and Responses to peers, data transmission to peers, and setting timers.

10.3.2.1 L2CAP to lower layer actions

— *LP_ConnectReq*

L2CAP requests the lower protocol to create a connection. If a physical link to the remote device does not exist, this message shall be sent to the lower protocol to establish the physical connection. Since no more than a single ACL link between two devices is assumed, see 10.1.2, additional L2CAP channels between these two devices shall share the same baseband ACL link.

Following the processing of the request, the lower layer returns with an *LP_ConnectCfm* or an *LP_ConnectCfmNeg* to indicate whether the request has been satisfied or not, respectively.

— *LP_QoSReq*

L2CAP requests the lower protocol to accommodate a particular QoS parameter set. Following the processing of the request, the lower layer returns with an *LP_QoSCfm* or an *LP_QoSCfmNeg* to indicate whether the request has been satisfied or not, respectively.

— *LP_ConnectRsp*

A positive response accepting the previous connection indication request (see *LP_ConnectInd* in 10.3.1.1).

— *LP_ConnectRspNeg*

A negative response denying the previous connection indication request (see *LP_ConnectInd* in 10.3.1.1).

Copyright © 2002 IEEE. All rights reserved.

10.3.2.2 L2CAP to L2CAP signalling actions

This section contains the same names identified in 10.3.1.2 except the actions refer to the transmission, rather than reception, of these messages.

10.3.2.3 L2CAP to L2CAP data actions

This section is the counterpart of 10.3.1.3. Data transmission is the action performed here.

10.3.2.4 L2CAP to upper layer actions

— *L2CA_ConnectInd*

Indicates a Connection Request has been received from a remote device (see *L2CA_ConnectReq* in 10.3.1.4).

— *L2CA_ConnectCfm*

Confirms that a Connection Request has been accepted (see (*L2CAP_ConnectReq* in 10.3.1.4) following the receipt of a Connection message from the remote device.

— *L2CA_ConnectCfmNeg*

Negative confirmation (failure) of a Connection Request (see *L2CA_ConnectReq* in 10.3.1.4). An RTX timer expiration (see 10.3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

— *L2CA_ConnectPnd*

Confirms that a Connection Response (pending) has been received from the remote device.

— *L2CA_ConfigInd*

Indicates a Configuration Request has been received from a remote device.

— *L2CA_ConfigCfm*

Confirms that a Configuration Request has been accepted (see *L2CA_ConfigReq* in 10.3.1.4) following the receipt of a Configuration Response from the remote device.

— *L2CA_ConfigCfmNeg*

Negative confirmation (failure) of a Configuration Request (see *L2CA_ConfigReq* in 10.3.1.4). An RTX timer expiration (see 10.3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Connect Request can substitute for a negative Connect Response and result in this action.

— *L2CA_DisconnectInd*

Indicates a Disconnection Request has been received from a remote device or the remote device has been disconnected because it has failed to respond to a signalling request. See 10.3.1.5

— *L2CA_DisconnectCfm*

Confirms that a Disconnect Request has been processed by the remote device (see *L2CA_DisconnectReq* in 10.3.1.4) following the receipt of a Disconnection Response from the remote device. An RTX timer expiration (see 10.3.1.5 and *L2CA_TimeOutInd* below) for an outstanding Disconnect Request can substitute for a Disconnect Response and result in this action. Upon receiving this event the upper layer knows the L2CAP channel has been terminated. There is no corresponding negative confirm.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 236 of 1169

— *L2CA_TimeOutInd*

Indicates that a RTX or ERTX timer has expired. This indication will occur an implementation-dependant number of times before the L2CAP implementation will give up and send a L2CA_DisconnectInd.

— *L2CA_QoSViolationInd*

Indicates that the quality of service agreement has been violated.

10.3.3 Channel operational states

— *CLOSED*

In this state, there is no channel associated with this CID. This is the only state when a link level connection (Baseband*)* may not exist. Link disconnection forces all other states into the CLOSED state.

— *W4_L2CAP_CONNECT_RSP*

In this state, the CID represents a local end-point and an L2CAP_ConnectReq message has been sent referencing this endpoint and it is now waiting for the corresponding L2CAP_ConnectRsp message.

— *W4_L2CA_CONNECT_RSP*

In this state, the remote end-point exists and an *L2CAP_ConnectReq* has been received by the local L2CAP entity. An L2CA_ConnectInd has been sent to the upper layer and the part of the local L2CAP entity processing the received *L2CAP_ConnectReq* waits for the corresponding response. The response may require a security check to be performed.

— *CONFIG*

In this state, the connection has been established but both sides are still negotiating the channel parameters. The Configuration state may also be entered when the channel parameters are being renegotiated. Prior to entering the CONFIG state, all outgoing data traffic should be suspended since the traffic parameters of the data traffic are to be renegotiated. Incoming data traffic shall be accepted until the remote channel endpoint has entered the CONFIG state.

In the CONFIG state, both sides shall issue L2CAP_ConfigReq messages – if only defaults are being used, a null message should be sent (see 10.5.4). If a large amount of parameters need to be negotiated, multiple messages may be sent to avoid any MTU limitations and negotiate incrementally. See 10.6 for more details.

Moving from the CONFIG state to the OPEN state requires both sides to be ready. An L2CAP entity is ready when it has received a positive response to its final request and it has positively responded to the final request from the remote device.

— *OPEN*

In this state, the connection has been established and configured, and data flow may proceed.

— *W4_L2CAP_DISCONNECT_RSP*

In this state, the connection is shutting down and an L2CAP_DisconnectReq message has been sent. This state is now waiting for the corresponding response.

— *W4_L2CA_DISCONNECT_RSP*

In this state, the connection on the remote endpoint is shutting down and an L2CAP_DisconnectReq message has been received. An L2CA_DisconnectInd has been sent to the upper layer to notify the owner of the CID that the remote endpoint is being closed. This state is now waiting for the corresponding response from the upper layer before responding to the remote endpoint.

Copyright © 2002 IEEE. All rights reserved.

10.3.4 Mapping events to actions

Table 76 defines the actions taken in response to events that occur in a particular state. Events that are not listed in the table, nor have actions marked N/C (for no change), are assumed to be errors and silently discarded.

Data input and output events are only defined for the Open and Configuration states. Data may not be received during the initial Configuration state, but may be received when the Configuration state is re-entered due to a reconfiguration process. Data received during any other state should be silently discarded.

Table 76—L2CAP Channel State Machine

Event	Current state	Action	New state
LP_ConnectCfm	CLOSED	Flag physical link as up and initiate the L2CAP connection.	CLOSED
LP_ConnectCfmNeg	CLOSED	Flag physical link as down and fail any outstanding service connection requests by sending an L2CA_ConnectCfmNeg message to the upper layer.	CLOSED
LP_ConnectInd	CLOSED	Flag link as up.	CLOSED
LP_DisconnectInd	CLOSED	Flag link as down.	CLOSED
LP_DisconnectInd	Any except CLOSED	Send upper layer L2CA_DisconnectInd message.	CLOSED
LP_QoSViolationInd	Any but OPEN	Discard	N/C
LP_QoSViolationInd	OPEN	Send upper layer L2CA_QoSViolationInd message. If service level is guaranteed, terminate the channel.	OPEN or W4_L2CA_ DISCONNECT_ RSP
L2CAP_ConnectReq	CLOSED. (CID dynamically allocated from free pool.)	Send upper layer L2CA_ConnectInd. Optionally: Send peer L2CAP_ConnectRspPnd	W4_L2CA_ CONNECT_ RSP
L2CAP_ConnectRsp	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectCfm message. Disable RTX timer.	CONFIG
L2CAP_ConnectRspPnd	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectPnd message. Disable RTX timer and start ERTX timer.	N/C
L2CAP_ConnectRspNeg	W4_L2CAP_CONNECT_RSP	Send upper layer L2CA_ConnectCfmNeg message. Return CID to free pool. Disable RTX/ERTX timers.	CLOSED
L2CAP_ConfigReq	CLOSED	Send peer L2CAP_Reject message.	N/C

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 238 of 1169

Table 76—L2CAP Channel State Machine *(continued)*

Event	Current state	Action	New state
L2CAP_ConfigReq	CONFIG	Send upper layer L2CA_ConfigInd message.	N/C
L2CAP_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send upper layer L2CA_ConfigInd message.	CONFIG
L2CAP_ConfigRsp	CONFIG	Send upper layer L2CA_ConfigCfm message. Disable RTX timer. If an L2CAP_ConfigReq message has been received and positively responded to, then enter OPEN state, otherwise remain in CONFIG state.	N/C or OPEN
L2CAP_ConfigRsp Neg	CONFIG	Send upper layer L2CA_ConfigCfmNeg message. Disable RTX timer.	N/C
L2CAP_Disconnect Req	CLOSED	Send peer L2CAP_DisconnectRsp message.	N/C
L2CAP_Disconnect Req	Any except CLOSED	Send upper layer L2CA_ DisconnectInd message.	W4_L2CA_ DISCONNECT_ RSP
L2CAP_Disconnect Rsp	W4_L2CAP_ DISCONNECT_ RSP	Send upper layer L2CA_DisconnectCfm message. Disable RTX timer.	CLOSED
L2CAP_Data	OPEN or CONFIG	If complete L2CAP packet received, send upper layer L2CA_Read confirm.	N/C
L2CA_ConnectReq	CLOSED (CID dynamically allocated from free pool)	Send peer L2CAP_ConnectReq message. Start RTX timer.	W4_L2CAP_ CONNECT_RSP
L2CA_ConnectRsp	W4_L2CA_CONNEC T_RSP	Send peer L2CAP_ ConnectRsp message.	CONFIG
L2CA_ConnectRsp Neg	W4_L2CA_CONNEC T_RSP	Send peer L2CAP_ConnectRspNeg message. Return CID to free pool.	CLOSED
L2CA_ConfigReq	CLOSED	Send upper layer L2CA_ ConfigCfmNeg message.	N/C
L2CA_ConfigReq	CONFIG	Send peer L2CAP_ConfigReq message. Start RTX timer.	N/C
L2CA_ConfigReq	OPEN	Suspend data transmission at a convenient point. Send peer L2CAP_ConfigReq message. Start RTX timer.	CONFIG

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 239 of 1169

Table 76—L2CAP Channel State Machine *(continued)*

Event	Current state	Action	New state
L2CA_ConfigRsp	CONFIG	Send peer L2CAP_ConfigRsp message. If all outstanding L2CAP_ConfigReq messages have received positive responses then move in OPEN state. Otherwise, remain in CONFIG state.	N/C or OPEN
L2CA_ConfigRspNeg	CONFIG	Send peer L2CAP_ ConfigRspNeg message.	N/C
L2CA_Disconnect Req	OPEN or CONFIG	Send peerF L2CAP_DisconnectReq message. Start RTX timer.	W4_L2CAP_ DISCONNECT_ RSP
L2CA_DisconnectRsp	W4_L2CA_ DISCONNECT_ RSP	Send peer L2CAP_DisconnectRsp message. Return CID to free pool.	CLOSED
L2CA_DataRead	OPEN	If payload complete, transfer payload to InBuffer.	OPEN
L2CA_DataWrite	OPEN	Send peer L2CAP_Data message.	OPEN
Timer_RTX	Any	Send upper layer L2CA_TimeOutInd message. If final expiration, return CID to free pool and go to CLOSE state, else re-send Request.	N/C or CLOSED
Timer_ERTX	Any	Send upper layer L2CA_TimeOutInd message. If final expiration, return CID to free pool and go to CLOSE state, else re-send Request.	N/C or CLOSED

Figure 106 illustrates a simplified state machine and typical transition path taken by an initiator and acceptor. The state machine shows what events cause state transitions and what actions are also taken while the transitions occur. Not all the events listed in Table 76 are included in the simplified State Machine to avoid cluttering the figure. Also, W2_L2CA_DISCON_RSP stands for W2_L2CA_DISCONNECT_RSP; W2_L2CAP_DISCON_RSP stands for W2_L2CAP_DISCONNECT_RSP.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 240 of 1169

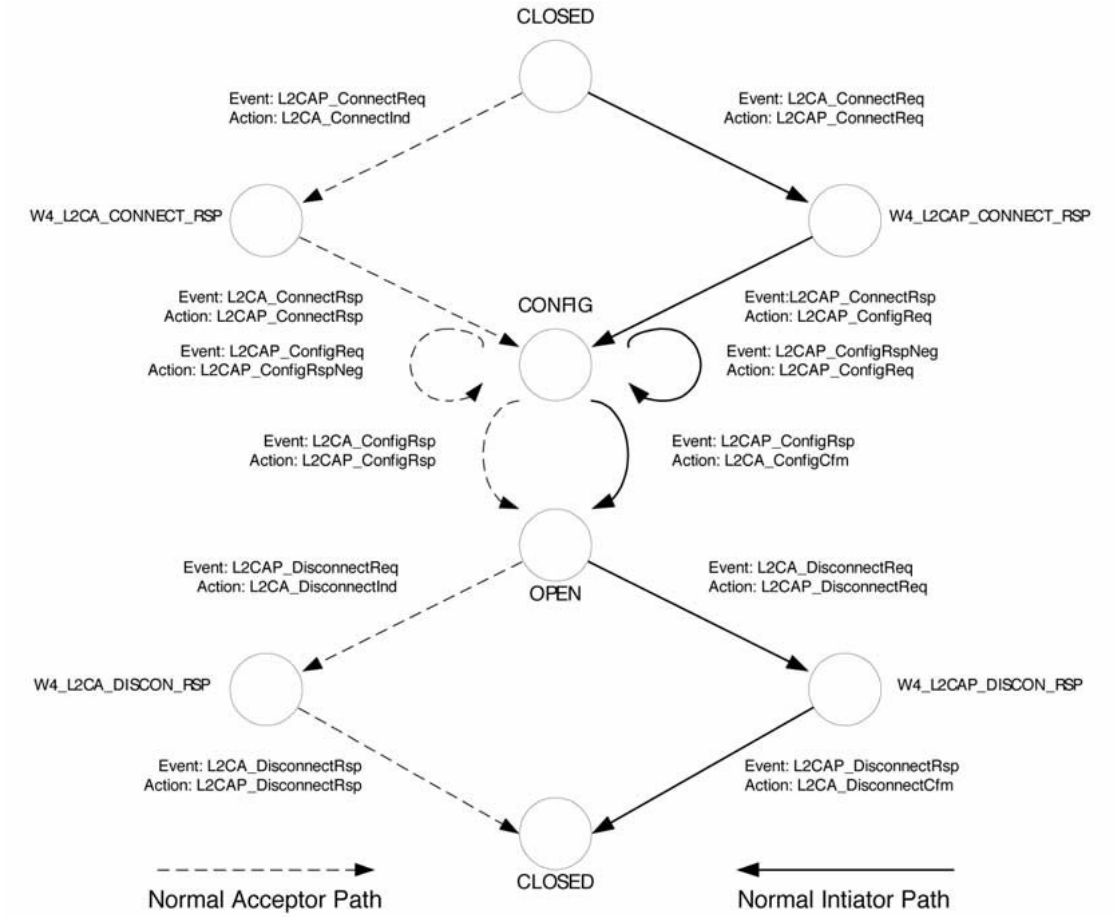


Figure 106—State machine example

Figure 107 presents another illustration of the events and actions based around the messages sequences being communicated between two devices. In this example, the initiator is creating the first L2CAP channel between two devices. Both sides start in the CLOSED state. After receiving the request from the upper layer, the entity requests the lower layer to establish a physical link. If no physical link exists, LMP commands are used to create the physical link between the devices. Once the physical link is established, L2CAP signals may be sent over it.

Figure 107 is an example and not all setup sequences will be identical to the one illustrated in the figure.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 241 of 1169

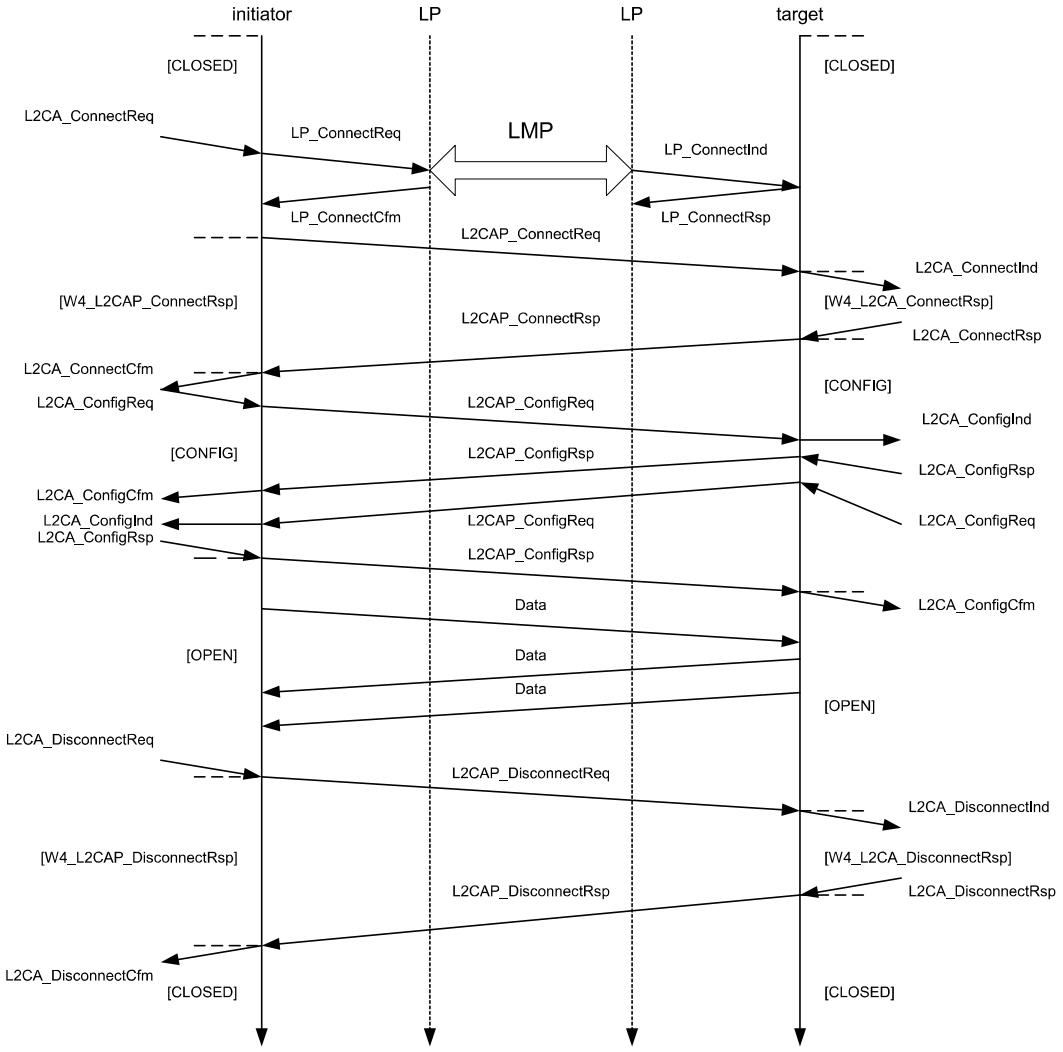


Figure 107—Message sequence chart of basic operation

10.4 Data packet format

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless. All packet fields use Little Endian byte order.

10.4.1 Connection-oriented channel

Figure 108 illustrates the format of the L2CAP packet (also referred to as the L2CAP PDU) within a connection-oriented channel.

Copyright © 2002 IEEE. All rights reserved.

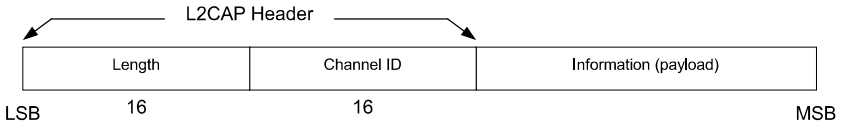


Figure 108—L2CAP packet (field sizes in bits)

The fields shown in Figure 108 are as follows:

— *Length: 2 octets (16 bits)*

Length indicates the size of information payload in bytes, excluding the length of the L2CAP header. The length of an information payload can be up to 65535 bytes. The Length field serves as a simple integrity check of the reassembled L2CAP packet on the receiving end.

— *Channel ID: 2 octets*

The channel ID identifies the destination channel endpoint of the packet. The scope of the channel ID is relative to the device the packet is being sent to.

— *Information: 0 to 65535 octets*

This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The minimum supported MTU for connection-oriented packets (MTU_{cno}) is negotiated during channel configuration (see 10.6.1). The minimum supported MTU for the signalling packet (MTU_{sig}) is 48 bytes (see 10.5).

10.4.2 Connectionless data channel

In addition to connection-oriented channels, L2CAP also exports the concept of a group-oriented channel. Data sent to the "group" channel is sent to all members of the group in a best-effort manner. Groups have no QoS associated with them. Group channels are unreliable; L2CAP makes no guarantee that data sent to the group successfully reaches all members of the group. If reliable group transmission is required, it must be implemented at a higher layer.

Transmissions to a group must be nonexclusively sent to all members of that group. The local device cannot be a member of the group, and higher layer protocols are expected to loopback any data traffic being sent to the local device. Nonexclusive implies non-group members may receive group transmissions and higher level (or link level) encryption can be used to support private communication. Figure 109 illustrates the format of the L2CAP PDU within a connectionless channel.

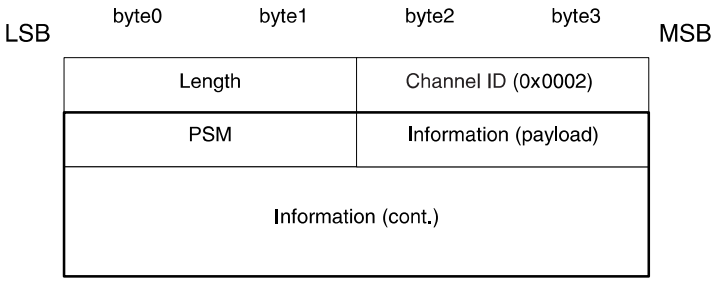


Figure 109—Connectionless packet

Copyright © 2002 IEEE. All rights reserved.

The fields shown in Figure 109 are as follows:

— *Length: 2 octets*

Length indicates the size of information payload plus the PSM field in bytes, excluding the length of the L2CAP header.

— *Channel ID: 2 octets*

Channel ID (0x0002) reserved for connectionless traffic.

— *Protocol/Service Multiplexer (PSM): 2 octets (minimum)*

The PSM field is based on the ISO 3309 extension mechanism for address fields. All content of the PSM field, referred to as the PSM value, shall be ODD, that is, the least significant bit of the least significant octet must be “1”. Also, all PSM values shall be assigned such that the least significant bit of the most significant octet equals “0”. This allows the PSM field to be extended beyond 16 bits. The PSM value definitions are specific to L2CAP and assigned by the Bluetooth SIG. For more information on the PSM field see 10.5.2.

— *Information: 0 to 65533 octets*

The payload information to be distributed to all members of the group. Implementations shall support a minimum connectionless MTU (MTU_{cnl}) of 670 octets, unless explicitly agreed upon otherwise, e.g., for single operation devices that are built to comply to a specific Bluetooth profile that dictates the use of a specific MTU for connectionless traffic that is less than MTU_{cnl}.

The L2CAP group service interface provides basic group management mechanisms including creating a group, adding members to a group, and removing members from a group. There are no predefined groups such as “all radios in range.”

10.5 Signalling

This subclause describes the signalling commands passed between two L2CAP entities on remote devices. All signalling commands are sent to CID 0x0001. The L2CAP implementation must be able to determine the Bluetooth address (BD_ADDR) of the device that sent the commands. Figure 110 illustrates the general format of all L2CAP packets containing signalling commands. Multiple commands may be sent in a single (L2CAP) packet and packets are sent to CID 0x0001. Commands take the form of Requests and Responses. All L2CAP implementations shall support the reception of signalling packets whose MTU (MTU_{sig}) does not exceed 48 bytes. L2CAP implementations should not use signalling packets beyond this size without first testing whether the implementation can support larger signalling packets. Implementations must be able to handle the reception of multiple commands in an L2CAP packet as long as the MTU is not exceeded.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 244 of 1169

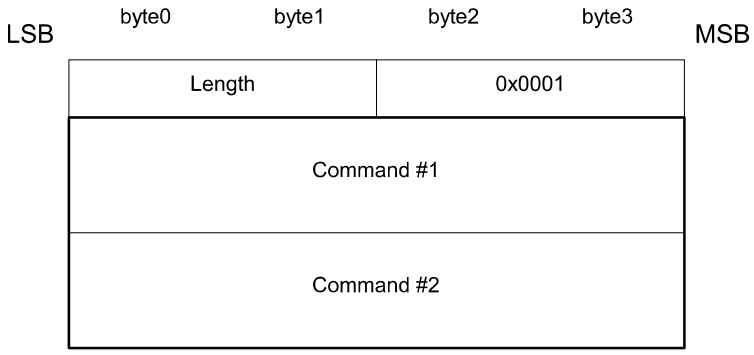


Figure 110—Signalling command packet format

Figure 111 displays the general format of all signalling commands.

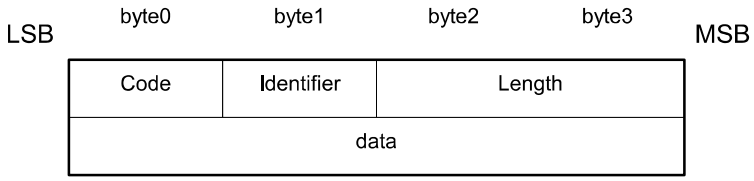


Figure 111—Command format

The fields shown in Figure 111 are as follows:

— *Code: 1 octet*

The Code field is one octet long and identifies the type of command. When a packet is received with an unknown Code field, a Command Reject packet (defined in 10.5.1) is sent in response. Up-to-date values of assigned Codes are specified in the latest Bluetooth Assigned Numbers document 2.4.3. Table 77 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

— *Identifier: 1 octet*

The Identifier field is one octet long and helps matching a request with the reply. The requesting device sets this field and the responding device uses the same value in its response. A different Identifier shall be used for each original command. Identifiers should not be recycled until a period of 360 s has elapsed from the initial transmission of the command using the identifier. On the expiration of a RTX or ERTX timer, the same identifier should be used if a duplicate Request is resent as stated in 10.3.1.5. A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded. Signalling identifier 0x0000 is defined to be an illegal identifier and shall never be used in any command.

— *Length: 2 octets*

The Length field is two octets long and indicates the size in octets of the data field of the command only, i.e., it does not cover the Code, Identifier, and Length fields.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 245 of 1169

— *Data: 0 or more octets*

The Data field is variable in length and discovered using the Length field. The Code field determines the format of the Data field.

Table 77—Signalling command codes

Code	Description
0x00	RESERVED
0x01	Command reject
0x02	Connection request
0x03	Connection response
0x04	Configure request
0x05	Configure response
0x06	Disconnection request
0x07	Disconnection response
0x08	Echo request
0x09	Echo response
0x0A	Information request
0x0B	Information response

10.5.1 Command reject (code 0x01)

A Command Reject packet is sent in response to a command packet with an unknown command code or when sending the corresponding Response is inappropriate. Figure 112 displays the format of the packet. The Identifier should match the Identifier of the packet containing the unidentified code field. Implementations shall always send these packets in response to unidentified signalling packets. Command Reject packets should not be sent in response to an identified Response packet.

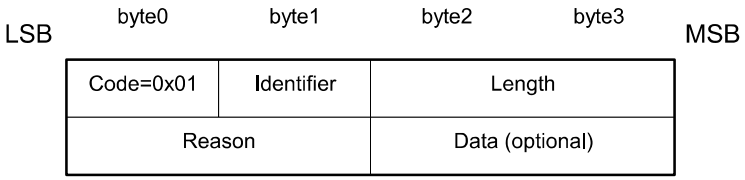


Figure 112—Command reject packet

When multiple commands are included in an L2CAP packet and the packet exceeds the MTU of the receiver, a single Command Reject packet is sent in response. The identifier should match the first Request command in the L2CAP packet. If only Responses are recognized, the packet shall be silently discarded.

— *Length = 0x0002 or more octets*

— *Reason: 2 octets*

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 246 of 1169

The Reason field describes why the Request packet was rejected (see Table 78).

Table 78—Reason code descriptions

Reason value	Description
0x0000	Command not understood
0x0001	Signalling MTU exceeded
0x0002	Invalid CID in request
Other	Reserved

— *Data: 0 or more octets*

The length and content of the Data field depends on the Reason code. If the Reason code is 0x0000, “Command not understood”, no Data field is used. If the Reason code is 0x0001, “Signalling MTU Exceeded”, the 2-octet Data field represents the maximum signalling MTU the sender of this packet can accept.

If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. A 4-octet data field on the command reject will contain the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The latter endpoints are obtained from the corresponding rejected command. If the rejected command contains only one of the channel endpoints, the other one is replaced by the null CID 0x0000 (see Table 79).

Table 79—Reason data values

Reason value	Data Length	Data value
0x0000	0 octets	N/A
0x0001	2 octets	Actual MTU
0x0002	4 octets	Requested CID

10.5.2 Connection request (code 0x02)

Connection request packets are sent to create a channel between two devices. The channel connection shall be established before configuration may begin. Figure 113 illustrates a minimum size Connection Request packet.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 247 of 1169

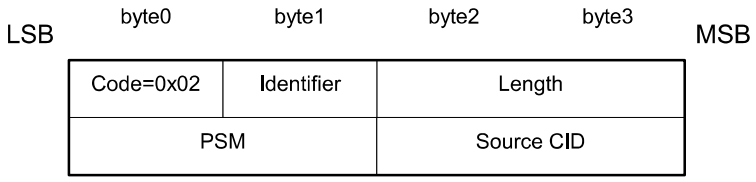


Figure 113—Connection request packet

— *Length = 0x0004 or more octets*

— *Protocol/Service Multiplexor (PSM): 2 octets (minimum)*

The PSM field is two octets (minimum) in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values shall be ODD, that is, the least significant bit of the least significant octet must be "1". Also, all PSM values shall be assigned such that the least significant bit of the most significant octet equals "0". This allows the PSM field to be extended beyond 16 bits. PSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the SDP; see 2.4.1, Part E, of the Bluetooth specification. The dynamically assigned values may be used to support multiple implementations of a particular protocol (see Table 80).

Table 80—Defined PSM values

PSM value	Description
0x0001	Service Discovery Protocol
0x0003	RFCOMM
0x0005	Telephony Control Protocol
<0x1000	RESERVED
[0x1001-0xFFFF]	DYNAMICALLY ASSIGNED

— *Source CID (SCID): 2 octets*

The source local CID is two octets in length and represents a channel end-point on the device sending the request. Once the channel has been configured, data packets flowing tothe sender of the request shall be sent to this CID. In this section, the Source CID represents the channel endpoint on the device sending the request and receiving the response.

10.5.3 Connection response (code 0x03)

When a unit receives a Connection Request packet, it shall send a Connection Response packet. The format of the connection response packet is shown in Figure 114.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 248 of 1169

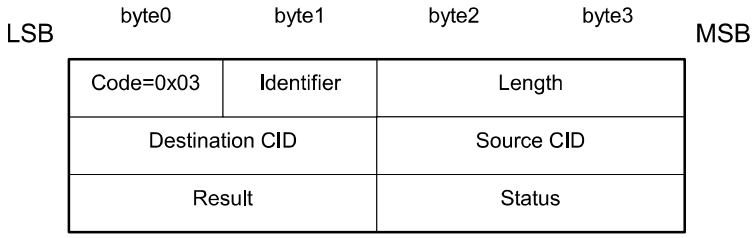


Figure 114—Connection response packet

— *Length = 0x0008 octets*

— *Destination Channel Identifier (DCID): 2 octets*

The field contains the channel end-point on the device sending this Response packet. In this section, the Destination CID represents the channel endpoint on the device receiving the request and sending the response.

— *Source Channel Identifier (SCID): 2 octets*

The field contains the channel end-point on the device receiving this Response packet.

— *Result: 2 octets*

The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a nonzero value indicates the connection request failed or is pending. A logical channel is established on the receipt of a successful result. Table 81 defines values for this field. If the result field is not zero. The DCID and SCID fields should be ignored when the result field indicates the connection was refused.

Table 81—Result values

Value	Description
0x0000	Connection successful
0x0001	Connection pending
0x0002	Connection refused – PSM not supported
0x0003	Connection refused – security block
0x0004	Connection refused – no resources available
Other	Reserved

— *Status: 2 octets*

Only defined for Result = Pending. Indicates the status of the connection (see Table 82).

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 249 of 1169

Table 82—Status values

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorization pending
Other	Reserved

10.5.4 Configuration request (code 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to renegotiate this contract whenever appropriate. During a renegotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request is related exclusively either with the outgoing or the incoming data traffic but not both of them. In 10.6, the various configuration parameters and their relation to the outgoing or incoming data traffic are presented. If an L2CAP entity receives a Configuration Request while it is waiting for a response it shall not block sending the Configuration Response, otherwise the configuration process may deadlock.

If no parameters need to be negotiated, no options need to be inserted and the C-bit should be cleared. L2CAP entities in remote devices shall negotiate all parameters defined in this document whenever the default values are not acceptable. Any missing configuration parameters are assumed to have their most recently (mutually) explicitly or implicitly accepted values. Even if all default values are acceptable, a Configuration Request packet with no options shall be sent. Implicitly accepted values are any default values for the configuration parameters specified in this document that have not been explicitly negotiated for the specific channel under configuration.

Each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. If a device needs to establish the value of a configuration parameter in the opposite direction than the one implied by a Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Configuration Request.

The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation but it shall not last more than 120 s.

Figure 115 defines the format of the Configuration Request packet.

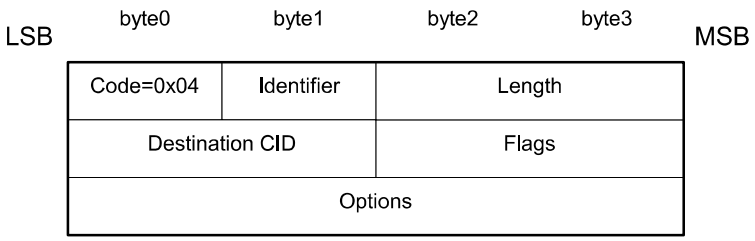


Figure 115—Configuration request packet

Copyright © 2002 IEEE. All rights reserved.

— *Length = 0x0004 or more octets*

— *Destination CID (DCID): 2 octets*

The field contains the channel end-point on the device receiving this Request packet.

— *Flags: 2 octets*

Figure 116 displays the two-octet Flags field. Note the most significant bit is shown on the left.

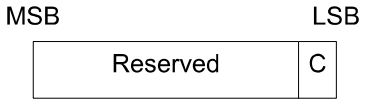


Figure 116—Configuration request flags field format

In Figure 116, C represents the continuation flag. When all configuration options cannot fit into the receiver's MTU_{sig}, the options must be passed in multiple configuration command packets. If all options fit into the receiver's MTU, then the continuation bit should not be used. Each Configuration Request should contain in integral number of options, partially formed options shall not be sent. Each Request will be tagged with a different Identifier and be matched with a Response with the same Identifier.

When used in the Configuration Request, the continuation flag indicates the responder should expect to receive multiple request packets. The responder shall reply to each request packet. The responder may reply to each Configuration Request with a Configuration Response containing the same option(s) present in the Request, except for those error conditions more appropriate for a Command Reject, or the responder may reply with a "Success" Configuration Response packet containing no options, delaying those options until the full Request has been received. The Configuration Request packet with the configuration flag cleared shall be treated as the Configuration Request event in the channel state machine.

When used in the Configuration Response, the continuation flag shall be set if the flag is set in the Request. If the configuration flag is set in the Response when the matching Request does not set the flag, it indicates the responder has additional options to send to the requestor. In this situation, the requestor shall send null-option Configuration Requests (with cleared C-flag) to the responder until the responder replies with a Configuration Response where the continuation flag is clear. The Configuration Response packet with the configuration flag cleared shall be treated as the Configuration Response event in the channel state machine.

The result of the configuration transaction is the union of all the result values. All the result values shall succeed for the configuration transaction to succeed.

Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

— *Configuration Options*

The list of the parameters and their values to be negotiated. These are defined in 10.6. Configuration Requests may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the length field is set to 0x0004.

10.5.5 Configure response (code 0x05)

Configure Response packets shall be sent in reply to Configuration Request packets except when the error condition id covered by a Command Reject response. Each configuration parameter value (if any is present) in a Configuration Response reflects an "adjustment" to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request. Thus, for example, if a

Copyright © 2002 IEEE. All rights reserved.

configuration parameter in a Configuration Request relates to traffic flowing from device A to device B, the sender of the Configuration Response will only adjust (if needed) this value again for the same traffic flowing from device A to device B. The options sent in the Response depend on the value in the Result field. Figure 117 defines the format of the Configuration Response packet.

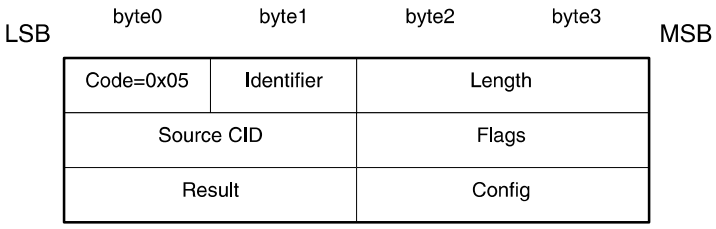


Figure 117—Configuration response packet

— *Length = 0x0006 or more octets*

— *Source CID (SCID): 2 octets*

The field contains the channel end-point on the device receiving this Response packet. The device receiving the Response shall check that the Identifier field matches the same field in the corresponding configuration request command and the SCID matches its local CID paired with the original DCID.

— *Flags: 2 octets*

Figure 118 displays the two-octet Flags field. Note that the most significant bit is shown on the left.

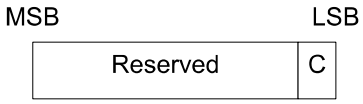


Figure 118—Configuration response flags field format

In Figure 118, C represents that more configuration responses will follow when set to 1. This flag indicates that the parameters included in the response are a partial subset of parameters being sent by the device sending the Response packet.
Other flags are reserved and should be cleared. L2CAP implementations should ignore these bits.

— *Result: 2 octets*

The Result field indicates whether or not the Request was acceptable. See Table 83 for possible result codes.

— *Configuration Options*

This field contains the list of parameters being negotiated. These are defined in 10.6. On a successful result, these parameters contain the return values for any wild card parameters (see 10.6.3) contained in the request.

On an unacceptable parameters failure (Result = 0x0001) the rejected parameters should be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters are assumed to have their most recently (mutually) accepted values and they too can be included in the Configuration Response if need to be changed. Recall that, each configuration parameter is one-directional and relative to the direction implied by the sender of a Configuration Request. Thus, if the sender of the Configuration Response needs to

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 252 of 1169

establish the value of a configuration parameter in the opposite direction than the one implied by an original Configuration Request, a new Configuration Request with the desired value of the configuration parameter in it needs to be sent in the direction opposite the one used for the original Connection Request.

On an unknown option failure (Result = 0x0003), the option types not understood by the recipient of the Request shall be included in the Response. Note that hints (defined in 10.6), those options in the Request that are skipped if not understood, shall not be included in the Response and shall not be the sole cause for rejecting the Request.

The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is left to the implementation.

Table 83—Configuration response result codes

Result	Description
0x0000	Success
0x0001	Failure – unacceptable parameters
0x0002	Failure – rejected (no reason provided)
0x0003	Failure – unknown options
Other	RESERVED

10.5.6 Disconnection request (code 0x06)

Terminating an L2CAP channel requires that a disconnection request packet be sent and acknowledged by a disconnection response packet. Disconnection is requested using the signalling channel since all other L2CAP packets sent to the destination channel automatically get passed up to the next protocol layer. Figure 119 displays a disconnection packet request. The receiver shall ensure both source and destination CIDs match before initiating a connection disconnection. Once a Disconnection Request is issued, all incoming data in transit on this L2CAP channel will be discarded and any new additional outgoing data is not allowed. Once a disconnection request for a channel has been received, all data queued to be sent out on that channel may be discarded.

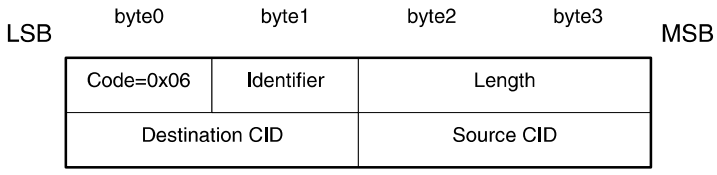


Figure 119—Disconnection request packet

Copyright © 2002 IEEE. All rights reserved.

— *Length = 0x0004 octets*

— *Destination CID (DCID): 2 octets*

This field specifies the end-point of the channel to be shutdown on the device receiving this request.

— *Source CID (SCID): 2 octets*

This field specifies the end-point of the channel to be shutdown on the device sending this request. The SCID and DCID are relative to the sender of this request and shall match those of the channel to be disconnected. If the DCID is not recognized by the receiver of this message, a Command Reject message with “invalid CID” result code shall be sent in response. If the receivers finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.

10.5.7 Disconnection response (code 0x07)

Disconnection responses should be sent in response to each disconnection request (see Figure 120).



Figure 120—Disconnection response packet

— *Length = 0x0004 octets*

— *Destination CID (DCID): 2 octets*

This field identifies the channel end-point on the device sending the response.

— *Source CID (SCID): 2 octets*

This field identifies the channel end-point on the device receiving the response.

The DCID and the SCID (which are relative to the sender of the request), and the Identifier fields shall match those of the corresponding disconnection request command. If the CIDs do not match, the response should be silently discarded at the receiver.

10.5.8 Echo request (code 0x08)

Echo requests are used to solicit a response from a remote L2CAP entity. These requests may be used for testing the link or passing vendor specific information using the optional data field. L2CAP entities shall respond to well-formed Echo Request packets with an Echo Response packet. The Data field is optional and implementation-dependent. L2CAP entities should ignore the contents of this field (see Figure 121).

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 254 of 1169

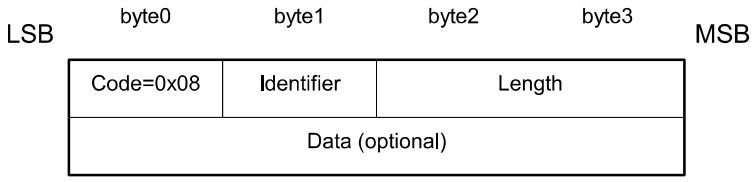


Figure 121—Echo request packet

10.5.9 Echo response (code 0x09)

Echo responses are sent upon receiving Echo Request packets. The identifier in the response shall match the identifier sent in the Request. The optional and implementation-dependent data field may contain the contents of the data field in the Request, different data, or no data at all (see Figure 122).

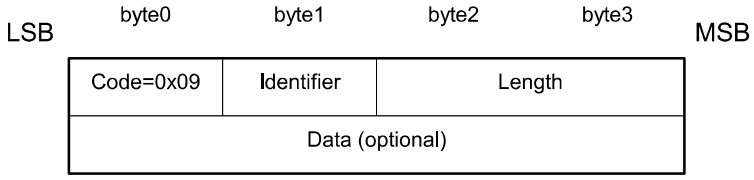


Figure 122—Echo response packet

10.5.10 Information Request (CODE 0x0A)

Information requests are used to solicit implementation-specific information from a remote L2CAP entity. L2CAP entities shall respond to well-formed Information Request packets with an Information Response packet (see Figure 123).

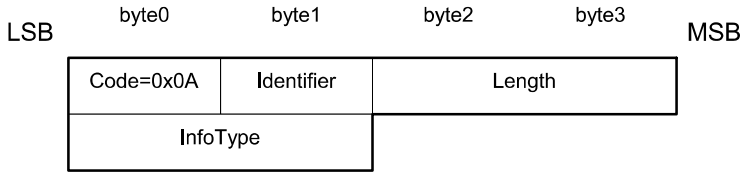


Figure 123—Information request packet

— *Length = 0x0002 octets*

— *InfoType: 2 octets*

The InfoType defines the type of implementation-specific information being solicited (see Table 84).

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 255 of 1169

Table 84—InfoType definitions

Value	Description
0x0001	Connectionless MTU
Other	Reserved

10.5.11 Information response (CODE 0x0B)

Information responses are sent upon receiving Information Request packets. The identifier in the response shall match the identifier sent in the Request. The optional data field may contain the contents of the data field in the Request, different data, or no data at all (see Figure 124).

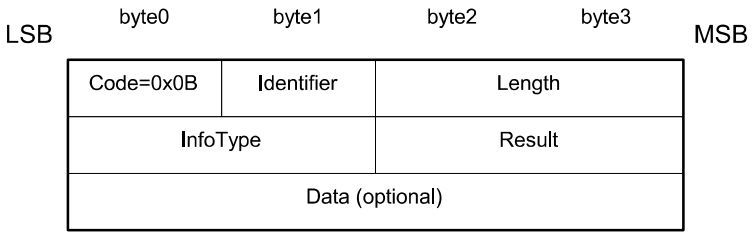


Figure 124—Information response packet

— *Length = 0x0004 or more octets*

— *InfoType: 2 octets*

Same value sent in the request.

— *Result: 2 octets*

The Result contains information about the success of the request. If result is “Success”, the data field contains the information as specified in Table 85. If result is “Not supported”, no data should be returned.

Table 85—Information response result values

Value	Description
0x0000	Success
0x0001	Not supported
Other	Reserved

— *Data: 0 or more octets*

The contents of the Data field depends on the InfoType. For the Connection MTU request, the data field contains the remote entity’s 2-octet acceptable connectionless MTU (see Table 86).

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 256 of 1169

Table 86—Information response data fields

InfoType	Data	Data length (in octets)
0x0001	Connectionless MTU	2

10.6 Configuration parameter options

Options are a mechanism to extend the ability to negotiate different connection requirements. Options are transmitted in the form of information elements comprises an option type, an option length, and one or more option data fields. Figure 125 illustrates the format of an option.

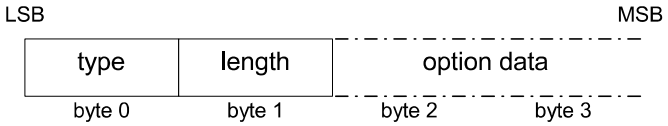


Figure 125—Configuration option format

— *Type: 1 octet*

The option type field defines the parameters being configured. The most significant bit of the type determines the action taken if the option is not recognized. The semantics assigned to the bit are defined as follows:

0–Option shall be recognized; refuse the configuration request

1–Option is a hint; skip the option and continue processing

— *Length: 1 octet*

The length field defines the number of octets in the option payload. So an option type with no payload has a length of 0.

— *Option data*

The contents of this field are dependent on the option type.

10.6.1 Maximum transmission unit (MTU)

This option specifies the payload size the sender is capable of accepting. The type is 0x01, and the payload length is 2 bytes, carrying the two-octet MTU size value as the only information element (see Figure 126).

Since all L2CAP implementations are capable to support a minimum L2CAP packet size (see 10.4), MTU is not really a negotiated value but rather an informational parameter to the remote device that the local device can accommodate in this channel an MTU larger than the minimum required. In the unlikely case that the remote device is only willing to send L2CAP packets in this channel that are larger than the MTU announced by the local device, then this Configuration Request will receive a negative response in which the remote device will include the value of MTU that is indented to transmit. In this case, it is implementation specific on whether the local device will continue the configuration process or even maintain this channel.

The remote device in its positive Configuration Response will include the actual MTU to be used on this channel for traffic flowing into the local device which is minimum{ MTU in configReq, outgoing MTU capability of remote device }. The MTU to be used on this channel but for the traffic flowing in the opposite

Copyright © 2002 IEEE. All rights reserved.

direction will be established when the remote device (with respect to this discussion) sends its own Configuration Request as explained in 10.5.4.

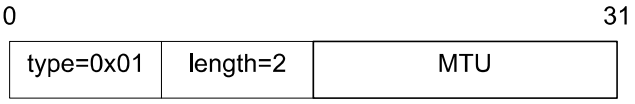


Figure 126—MTU Option Format

— MTU Size: 2 octets

The MTU field represents the largest L2CAP packet payload, in bytes, that the originator of the Request can accept for that channel. The MTU is asymmetric and the sender of the Request shall specify the MTU it can receive on this channel if it differs from the default value. L2CAP implementations shall support a minimum MTU size of 48 bytes. The default value is 672 bytes.[18]

10.6.2 Flush timeout option

This option is used to inform the recipient of the amount of time the originator's link controller/link manager will attempt to successfully transmit an L2CAP segment before giving up and flushing the packet. The type is 0x02 and the payload size is 2 octets (see Figure 127).

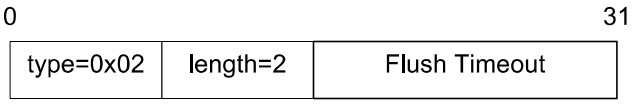


Figure 127—Flush timeout

— *Flush Timeout*

This value represents units of time measured in milliseconds. The value of 1 implies no retransmissions at the Baseband level should be performed since the minimum polling interval is 1.25 ms. The value of all 1's indicates an infinite amount of retransmissions. This is also referred to as "reliable channel". In this case, the link manager shall continue retransmitting a segment until physical link loss occurs. This is an asymmetric value and the sender of the Request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

10.6.3 QoS option

This option specifies a flow specification (flowSpec) similar to RFC 1363 (see 2.5). If no QoS configuration parameter is negotiated the link should assume the default parameters discussed below. The QoS option is type 0x03.

When included in a Configuration Request, this option describes the outgoing traffic flow from the device sending the request to the device receiving it. When included in a positive Configuration Response, this option describes the incoming traffic flow agreement as seen from the device sending the response. When

[18] The default MTU was selected based on the payload carried by two Baseband DH5 packets (2 * 341 = 682) minus the Baseband ACL headers (2 * 2 = 4) and L2CAP header (6).

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 258 of 1169

included in a negative Configuration Response, this option describes the preferred incoming traffic flow from the perspective of the device sending the response.

L2CAP implementations are only required to support “Best Effort” service, support for any other service type is optional. Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort shall be assumed. If any QoS guarantees are required then a QoS configuration request shall be sent.

The remote device places information that depends on the value of the result field, see 10.5.5, in its Configuration Response. If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is “Failure – unacceptable parameters”, the response may include a list of outgoing flowspec parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response. Recall that any missing configuration parameters from a Configuration Request are assumed to have their most recently (mutually) accepted values. For both Best effort and Guaranteed service, when the QoS option appears in the Configuration Response, "do not cares" shall be present where they appeared in the Configuration Request (see Figure 128).

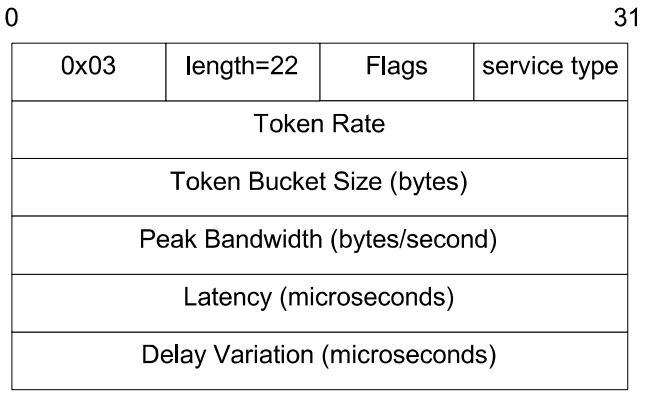


Figure 128—QoS flow specification

— *Flags: 1 octet*

Reserved for future use and shall be set to 0.

— *Service Type: 1 octet*

This field indicates the level of service required. Table 87 defines the different services available. If “No traffic” is selected, the remainder of the fields may be ignored because there is no data being sent across the channel in the outgoing direction.

If “Best effort,” the default value, is selected, the remaining fields should be treated as hints by the remote device. The remote device may choose to ignore the fields, try to satisfy the hint but provide no response (QoS option omitted in the Response message), or respond with the settings it will try to meet.

— *Token Rate: 4 octets*

The value of this field represents the rate at which traffic credits are granted in bytes per second. An application may send data at this rate continuously. Burst data may be sent up to the token bucket size (see below). Until that data burst has been drained, an application must limit itself to the token rate. The value 0x00000000 indicates no token rate is specified. This is the default value and implies

Copyright © 2002 IEEE. All rights reserved.

indifference to token rate. The value 0xFFFFFFFF represents a wild card matching the maximum token rate available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

— *Token Bucket Size: 4 octets*

The value of this field represents the size of the token bucket in bytes. If the bucket is full, then applications must either wait or discard data. The value of 0x00000000 represents no token bucket is needed; this is the default value. The value 0xFFFFFFFF represents a wild card matching the maximum token bucket available. The meaning of this value depends on the semantics associated with the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum buffer space available at the time of the request.

— *Peak Bandwidth: 4 octets*

The value of this field, expressed in bytes per second, limits how fast packets may be sent back-to-back from applications. Some intermediate systems can take advantage of this information, resulting in more efficient resource allocation. The value of 0x00000000 states that the maximum bandwidth is unknown, which is the default value.

— *Latency: 4 octets*

The value of this field represents the maximum acceptable delay between transmission of a bit by the sender and its initial transmission over the air, expressed in microseconds. The precise interpretation of this number depends on the level of guarantee specified in the Class of Service. The value 0xFFFFFFFF represents a do not care and is the default value.

— *Delay Variation: 4 octets*

The value of this field is the difference, in microseconds, between the maximum and minimum possible delay that a packet will experience. This value is used by applications to determine the amount of buffer space needed at the receiving side in order to restore the original data transmission pattern. The value 0xFFFFFFFF represents a “do not care” and is the default value.

Table 87—Service type definitions

Value	Description
0x00	No traffic
0x01	Best effort (Default)
0x02	Guaranteed
Other	Reserved

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 260 of 1169

10.6.4 Configuration process

Negotiating the channel parameters involves three steps:

1) Informing the remote side of the nondefault parameters that the local side will accept using a Configuration Request
2) Remote side responds, agreeing or disagreeing to these values, including the default ones, using a Configuration Response. The local and remote devices repeat steps 1) and 2) as needed.
3) Repeat steps 1) and 2) exactly once more for the reverse direction.

This process can be abstracted into the initial Request negotiation path and a Response negotiation path, followed by the reverse direction phase. Reconfiguration follows a similar two-phase process by requiring negotiation in both directions.

10.6.4.1 Request Path

The Request Path negotiates the incoming MTU, flush timeout, and outgoing flowspec. Table 88 defines the configuration options that may be placed in the Configuration Request message and their semantics.

Table 88—Parameters allowed in request

Parameter	Description
MTU	Incoming MTU information
FlushTO	Outgoing flush timeout
OutFlow	Outgoing flow information

10.6.4.2 Response path

The Response Path negotiates the outgoing MTU (remote side's incoming MTU), the remote side's flush timeout, and incoming flowspec (remote side's outgoing flowspec). If a request-oriented parameter is not present in the Request message (reverts to default value), the remote side may negotiate for a nondefault value by including the proposed value in a negative Response message (see Table 89).

Table 89—Parameters allowed in response

Parameter	Description
MTU	Outgoing MTU information
FlushTO	Incoming flush timeout
InFlow	Incoming flow information

Copyright © 2002 IEEE. All rights reserved.

10.6.4.3 Configuration state machine

The configuration state machine shown in Figure 129 depicts two paths. Before leaving the CONFIG state and moving into the OPEN state, both paths shall reach closure. The request path requires the local device to receive a positive response to reach closure while the response path requires the local device to send a positive response to reach closure.

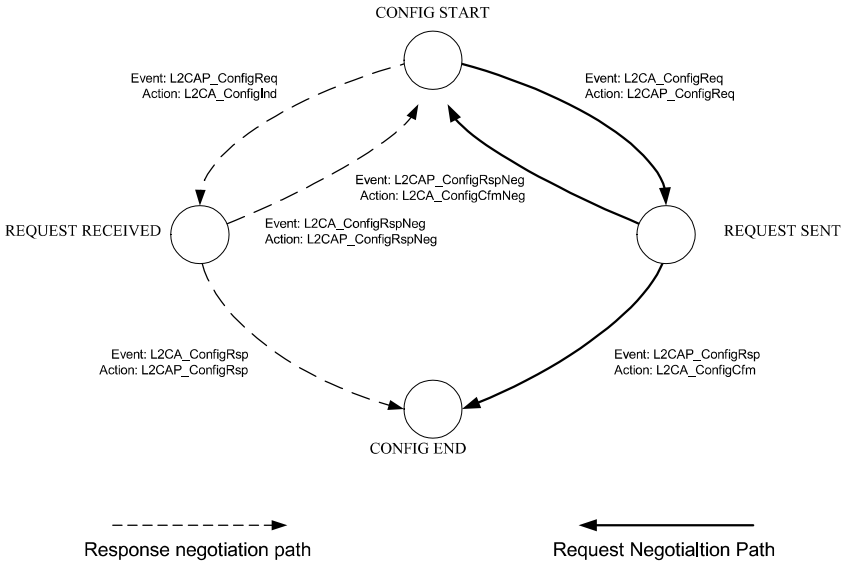


Figure 129—Configuration state machine

Subclause 10.8.1 provides some configuration examples.

10.7 Service primitives

This subclause presents an abstract description of the services offered by L2CAP in terms of service primitives and parameters. The service interface is required for testing. The interface is described independently of any platform specific implementation. All data values use Little Endian byte ordering.

10.7.1 Event indication

Table 90 provides the event indication service primative.

Table 90—Event indication

Service	Input parameters	Output parameters
EventIndication	Event, Callback	Result

Description:

The use of this primitive requests a callback when the selected indication Event occurs.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 262 of 1169

Input Parameters:

Event *Type: uint* *Size: 1 octet*

Value	Description
0x00	Reserved
0x01	L2CA_ConnectInd
0x02	L2CA_ConfigInd
0x03	L2CA_DisconnectInd
0x04	L2CA_QoSViolationInd
other	Reserved for future use

Callback *Type: function* *Size: N/A*

Event	Callback function input parameters
L2CA_ConnectInd	BD_ADDR, CID, PSM, Identifier
L2CA_ConfigInd	CID, OutMTU, InFlow, InFlushTO
L2CA_DisconnectInd	CID
L2CA_QoSViolationInd	BD_ADDR

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Event successfully registered
0x0001	Event registration failed

10.7.1.1 L2CA_ConnectInd callback

This callback function includes the parameters for the address of the remote device that issued the connection request, the local CID representing the channel being requested, the Identifier contained in the request, and the PSM value the request is targeting.

10.7.1.2 L2CA_ConfigInd callback

This callback function includes the parameters indicating the local CID of the channel the request has been sent to, the outgoing MTU size (maximum packet that can be sent across the channel) and the flowspec describing the characteristics of the incoming data. All other channel parameters are set to their default values if not provided by the remote device.

10.7.1.3 L2CA_DisconnectInd callback

This callback function includes the parameter indicating the local CID the request has been sent to.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 263 of 1169

10.7.1.4 L2CA_QoSViolationInd callback

This callback function includes the parameter indicating the address of the remote Bluetooth device where the QoS contract has been violated.

10.7.2 Connect

Table 91 shows the parameters associated with the L2CA connection request primitive.

Table 91—Connect request

Service	Input parameters	Output parameters
L2CA_ConnectReq	PSM, BD_ADDR	LCID, Result, Status

Description:

This primitive initiates the sending of an L2CAP_ConnectReq message and blocks until a corresponding L2CA_ConnectCfm(Neg) or L2CA_TimeOutInd eventis received.

The use of this primitive requests the creation of a channel representing a logical connection to a physical address. Input parameters are the target protocol (*PSM*) and remote device's 48-bit address (*BD_ADDR*). Output parameters are the local CID *(LCID)* allocated by the local L2CAP entity, and *Result* of the request. If the *Result* indicates success, the *LCID* value contains the identification of the local endpoint. Otherwise the *LCID* returned should be set to 0. If *Result* indicates a pending notification, the *Status* value may contain more information of what processing is delaying the establishment of the connection. Otherwise the *Status* value should be ignored.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Target PSM provided for the connection

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Output Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel if Result = 0x0000, otherwise set to 0.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 264 of 1169

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful and the CID identifies the local endpoint. Ignore Status parameter
0x0001	Connection pending. Check Status parameter for more information
0x0002	Connection refused because no service for the PSM has been registered
0x0003	Connection refused because the security architecture on the remote side has denied the request
0xEEEE	Connection timeout occurred. This is a result of a timer expiration indication being included in the connection confirm message

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information
0x0001	Authentication pending
0x0002	Authorization pending

10.7.3 Connect response

Table 92 shows the parameters associated with the L2CA connection response primitive.

Table 92—Connect response

Service	Input parameters	Output parameters
L2CA_ConnectRsp	BD_ADDR, Identifier, LCID, Response, Status	Result

Description:

This primitive represents the L2CAP_ConnectRsp.

The use of this primitive issues a response to a connection request event indication. Input parameters are the remote device's 48-bit address, Identifier sent in the request, local CID, the Response code, and the Status attached to the Response code. The output parameter is the Result of the service request.

This primitive shall be called no more than once after receiving the callback indication. This primitive returns once the local L2CAP entity has validated the request. A successful return does indicate the response has been sent over the air interface.

Copyright © 2002 IEEE. All rights reserved.

Input Parameters:

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device

Identifier *Type: uint* *Size: 1 octets*

Value	Description
0xXX.	This value shall match the value received in the L2CA_ConnectInd event described in 10.7.1.1

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Response *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Connection successful
0x0001	Connection pending
0x0002	Connection refused – PSM not supported
0x0003	Connection refused – security block
0x0004	Connection refused – no resources available
0xXXXX	Other connection response code

Status *Type: uint* *Size: 2 octets*

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorization pending
0xXXXX	Other status code

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response successfully sent
0x0001	Failure to match any outstanding connection request

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 266 of 1169

10.7.4 Configure

Table 93 shows the parameters associated with the L2CA configure primitive.

Table 93—Configure

Service	Input parameters	Output parameters
L2CA_ConfigReq	CID, InMTU, OutFlow, OutFlushTO, LinkTO	Result, InMTU, OutFlow, OutFlushTO

Description:

This primitive initiates the sending of an L2CAP_ConfigReq message and blocks until a corresponding L2CA_ConfigCfm(Neg) or L2CA_TimeOutInd eventis received.

The use of this primitive requests the initial configuration (or reconfiguration) of a channel to a new set of channel parameters. Input parameters are the local CID endpoint, new incoming receivable MTU (InMTU), new outgoing flow specification, and flush and link timeouts. Output parameters composing the L2CA_ConfigCfm(Neg) message are the Result, accepted incoming MTU(InMTU), the remote side's flow requests, and flush and link timeouts. Note that the output results are returned only after the local L2CAP entity transitions out of the CONFIG state (even if this transition is back to the CONFIG state).

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Local CID

InMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel can accept

OutFlow *Type: Flow* *Size: x octets*

Value	Description
flowspec	Quality of service parameters dealing with the traffic characteristics of the outgoing data flow

Copyright © 2002 IEEE. All rights reserved.

OutFlushTO *Size: 2 octets*

Value	Description
0xXXXX	Number of milliseconds to wait before an L2CAP packet that cannot be acknowledged at the physical layer is dropped
0x0000	Request to use the existing flush timeout value if one exists, otherwise the default value (0xFFFF) will be used
0x0001	Perform no retransmissions at the Baseband layer
0xFFFF	Perform retransmission at the Baseband layer until the link timeout terminates the channel

LinkTO *Size: 2 octets*

Value	Description
0xXXXX	Number of milliseconds to wait before terminating an unresponsive link

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values
0x0001	Failure – invalid CID
0x0002	Failure – unacceptable parameters
0x0003	Failure – signalling MTU exceeded
0x0004	Failure – unknown options
0xEEEE	Configuration timeout occurred. This is a result of a timer expiration indication being included in the configuration confirm

InMTU *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit that the remote unit will send across this channel (maybe less or equal to the InMTU input parameter).

OutFlow *Size: 2 octets*

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the agreed-upon out-going data flow if Result is successful. Otherwise this represents the requested Quality of Service

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 268 of 1169

OutFlushTO *Size: 2 octets*

Value	Description
0xXXXX	Number of milliseconds before an L2CAP packet that cannot be acknowledged at the physical layer is dropped. This value is informative of the actual value that will be used for outgoing packets. It may be less or equal to the OutFlushTO parameter given as input.

10.7.5 Configuration response

Table 94 shows the parameters associated with the L2CA configuration response primitive.

Table 94—Configuration response

Service	Input parameters	Output parameters
L2CA_ConfigRsp	CID, OutMTU, InFlow	Result

Description:

This primitive represents the L2CAP_ConfigRsp.

The use of this primitive issues a response to a configuration request event indication. Input parameters include the local CID of the endpoint being configured, outgoing transmit MTU (which may be equal or less to the OutMTU parameter in the L2CA_ConfigInd event) and the accepted flowspec for incoming traffic. The output parameter is the Result value.

Input Parameters:

LCID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Local channel identifier

OutMTU *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Maximum transmission unit this channel will send

InFlow *Type: Flow* *Size: x octets*

Value	Description
FlowSpec	Quality of service parameters dealing with the traffic characteristics of the incoming data flow

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 269 of 1169

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Configuration is successful. Parameters contain agreed upon values
0x0001	Configuration failed – unacceptable parameters
0x0002	Configuration failed – rejected
0x0003	Configuration failed – invalid CID
0x0004	Configuration failed – unknown options
0xXXXX	Reserved

10.7.6 Disconnect

Table 95 shows the parameters associated with the L2CA disconnect response primitive.

Table 95—Disconnect

Service	Input parameters	Output parameters
L2CA_DisconnectReq	CID	Result

Description:

This primitive represents the L2CAP_DisconnectReq and the returned output parameters represent the corresponding L2CAP_DisconnectRsp or the RTX timer expiration.

The use of this primitive requests the disconnection of the channel. Input parameter is the *CID* representing the local channel endpoint. Output parameter is *Result*. *Result* is zero if a L2CAP_DisconnectRsp is received, otherwise a non-zero value is returned. Once disconnection has been requested, no process will be able to successfully read or write from the CID. Writes in progress should continue to be processed.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Disconnection successful. This is a result of the receipt of a disconnection response message
0xEEEE	Disconnection timeout occurred.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 270 of 1169

10.7.7 Write

Table 96 shows the parameters associated with the L2CA write primitive.

Table 96—Write

Service	Input parameters	Output parameters
L2CA_DataWrite	CID, Length, OutBuffer	Size, Result

Description:

The use of this primitive requests the transfer of data across the channel. If the length of the data exceeds the OutMTU then only the first OutMTU bytes are sent This command may be used for both connection-oriented and connectionless traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where data to be transmitted are stored

OutBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the input buffer used to store the message

Output Parameters:

Size *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	The number of bytes transferred

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful write
0x0001	Error – Flush timeout expired
0x0002	Error – Link termination (perhaps this should be left to the indication)

Copyright © 2002 IEEE. All rights reserved.

10.7.8 Read

Table 97 shows the parameters associated with the L2CA read primitive.

Table 97—Read

Service	Input parameters	Output parameters
L2CA_DataRead	CID, Length, InBuffer	Result, N

Description:

The use of this primitive requests for the reception of data. This request returns when data is available or the link is terminated. The data returned represents a single L2CAP payload. If not enough data is available, the command will block until the data arrives or the link is terminated. If the payload is bigger than the buffer, only the portion of the payload that fits into the buffer will be returned, and the remainder of the payload will be discarded. This command may be used for both connection-oriented and connectionless traffic.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	CID

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the buffer where received data are to be stored

InBuffer *Type: pointer* *Size: N/A*

Value	Description
N/A	Address of the buffer used to store the message

Output parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Unsuccessful

N *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Number of bytes transferred to InBuffer

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 272 of 1169

10.7.9 Group create

Table 98 shows the parameters associated with the L2CA group create primitive.

Table 98—Group create

Service	Input parameters	Output parameters
L2CA_GroupCreate	PSM	CID

Description:

The use of this primitive requests the creation of a CID to represent a logical connection to multiple devices. Input parameter is the *PSM* value that the outgoing connectionless traffic is labelled with, and the filter used for incoming traffic. Output parameter is the *CID* representing the local endpoint. On creation, the group is empty but incoming traffic destined for the PSM value is readable.

Input Parameters:

PSM *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Protocol/service multiplexer value

Output Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

10.7.10 Group close

Table 99 shows the parameters associated with the L2CA group close primitive.

Table 99—Group close

Service	Input parameters	Output parameters
L2CA_GroupClose	CID	Result

Description:

The use of this primitive closes down a Group.

Copyright © 2002 IEEE. All rights reserved.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Successful closure of the channel
0x0001	Invalid CID

10.7.11 Group add member

Table 100 shows the parameters associated with the L2CA group add member primitive.

Table 100—Group add member

Service	Input parameters	Output parameters
L2CA_GroupAddMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the addition of a member to a group. The input parameter includes the CID representing the group and the BD_ADDR of the group member to be added. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Remote device address

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 274 of 1169

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure to establish connection to remote device
Other	Reserved

10.7.12 Group remove member

Table 101 shows the parameters associated with the L2CA group remove member primitive.

Table 101—Group remove member

Service	Input parameters	Output parameters
L2CA_GroupRemoveMember	CID, BD_ADDR	Result

Description:

The use of this primitive requests the removal of a member from a group. The input parameters include the CID representing the group and BD_ADDR of the group member to be removed. The output parameter Result confirms the success or failure of the request.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address device to be removed

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure – device not a member of the group
Other	Reserved

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 275 of 1169

10.7.13 Get group membership

Table 102 shows the parameters associated with the L2CA get group membership primitive.

Table 102—Get group membership

Service	Input parameters	Output parameters
L2CA_GroupMembership	CID	Result, N, BD_ADDR_Lst

Description:

The use of this primitive requests a report of the members of a group. The input parameter CID represents the group being queried. The output parameter Result confirms the success or failure of the operation. If the Result is successful, BD_ADDR_Lst is a list of the Bluetooth addresses of the N members of the group.

Input Parameters:

CID *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Channel ID representing local end-point of the communication channel.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Success
0x0001	Failure – group does not exist
Other	Reserved

N *Type: uint* *Size: 2 octets*

Value	Description
0x0000-0xFFFF	The number of devices in the group identified by the channel end-point CID. If Result indicates failure, N should be set to 0.

BD_ADDR_Lst *Type: pointer* *Size: N/A*

Value	Description
N/A	List of N unique Bluetooth addresses of the devices in the group identified by the channel end-point CID. If Result indicates failure, the all-zero address is the only address that should be returned.

Copyright © 2002 IEEE. All rights reserved.

LINKPLAY EXHIBIT 1008
Page 276 of 1169

10.7.14 Ping

Table 103 shows the parameters associated with the L2CA ping primitive.

Table 103—Ping

Service	Input parameters	Output parameters
L2CA_Ping	BD_ADDR, ECHO_DATA, Length	Result, ECHO_DATA, Size

Description:

This primitive represents the initiation of an L2CA_EchoReq message and the reception of the corresponding L2CAP_EchoRsp message.

Input Parameters:

BD_ADDR *Type: uint* *Size: 6 octets*

Value	Description
0xXXXXXXXXXXXX	Unique Bluetooth address of target device.

ECHO_DATA *Type: pointer* *Size: N/A*

Value	Description
N/A	The buffer containing the contents to be transmitted in the data payload of the Echo Request command.

Length *Type: uint* *Size: 2 octets*

Value	Description
0xXXXX	Size, in bytes, of the data in the buffer.

Output Parameters:

Result *Type: uint* *Size: 2 octets*

Value	Description
0x0000	Response received.
0x0001	Timeout occurred.

ECHO_DATA *Type: pointer* *Size: N/A*

Value	Description
N/A	The buffer containing the contents received in the data payload of the Echo Response command.

Copyright © 2002 IEEE. All rights reserved.
LINKPLAY EXHIBIT 1008
Page 277 of 1169