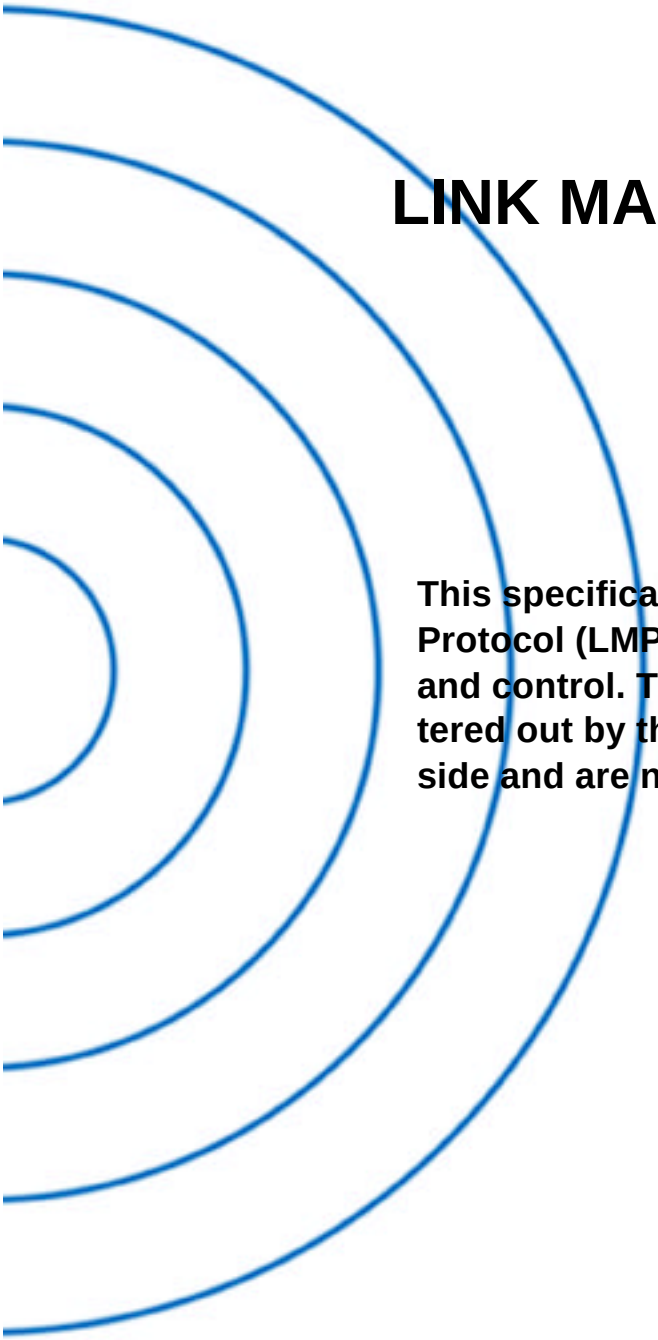
Part C



LINK MANAGER PROTOCOL

This specification describes the Link Manager Protocol (LMP) which is used for link set-up and control. The signals are interpreted and filtered out by the Link Manager on the receiving side and are not propagated to higher layers.

CONTENTS

1 GENERAL

LMP messages are used for link set-up, security and control. They are transferred in the payload instead of L2CAP and are distinguished by a reserved value in the L_CH field of the payload header. The messages are filtered out and interpreted by LM on the receiving side and are not propagated to higher layers.

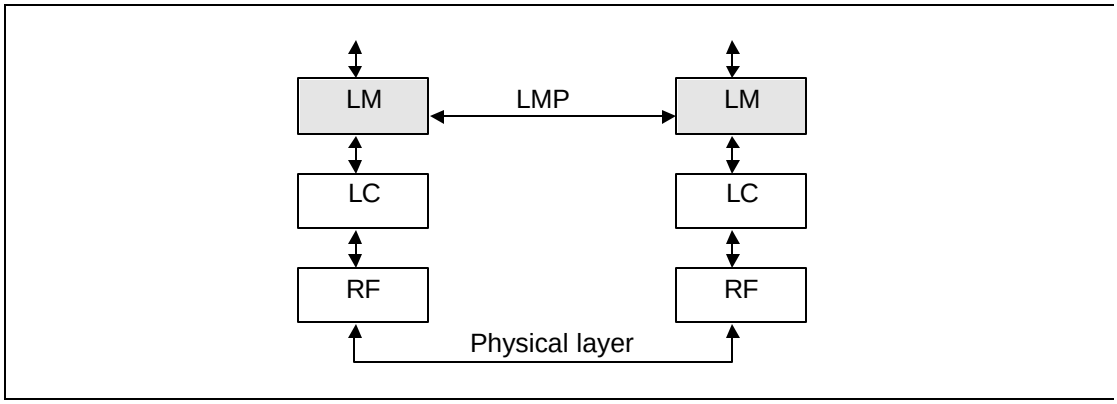


Figure 1.1: Link Manager's place on the global scene.

Link Manager messages have higher priority than user data. This means that if the Link Manager needs to send a message, it shall not be delayed by the L2CAP traffic, although it can be delayed by many retransmissions of individual baseband packets.

We do not need to explicitly acknowledge the messages in LMP since LC (see Baseband Specification Section 5, on page 66) provides us with a reliable link.

LC does not guarantee either the time taken to deliver a message to the remote device or the delay between the delivery of the message to the remote device and the reception of the corresponding ACK by the sender. This means that we must be aware of the underlying LC mechanism's limitations to synchronize state changes between master and slave. The criteria for determining when the master can reuse an AM_ADDR following the detach or park of a slave is based on the reception of the Baseband-level acknowledgement. Synchronization of a master-slave switch or the starting of hold mode utilizes the Bluetooth master clock, which the LM reads from the LC.

LC only guarantees that it will attempt to communicate with each slave once per T_{poll} slots.

T_{poll} is the poll interval as described in section 3.20 on page 225.

The time between receiving a baseband packet carrying an LMP PDU and sending a baseband packet carrying a valid response PDU, according to the

procedure rules in Section 3 on page 193, must shall be less than the LMP Response Timeout. The value of this timeout is 30 seconds. Note that the LMP Response Timeout is applied not only to sequences described in Section 3 on page 193, but also to the series of the sequences defined as the transactions in Section 3 on page 193. It is also applied to the series of the transactions, as long as no L2CAP PDUs are allowed, for example, any transactions until the PDUs LMP_setup_complete are exchanged.

2 FORMAT OF LMP

LM PDUs are always sent as single-slot packets and the payload header is therefore one byte. The two least significant bits in the payload header determine the logical channel. For LM PDUs these bits are set.

L_CH code	Logical Channel	Information
00	NA	undefined
01	UA/I	Continuing L2CAP message
10	UA/I	Start L2CAP message
11	LM	LMP message

Table 2.1: Logical channel L_CH field contents.

The FLOW bit in the payload header is always one and is ignored on the receiving side. Each PDU is assigned a 7-bit opcode used to uniquely identify different types of PDUs, see Table 5.1 on page 234. The opcode and a one-bit transaction ID are positioned in the first byte of the payload body. The transaction ID is positioned in the LSB. It is 0 if the PDU belongs to a transaction initiated by the master and 1 if the PDU belongs to a transaction initiated by the slave. If the PDU contains one or more parameters these are placed in the payload starting at the second byte of the payload body. The number of bytes used depends on the length of the parameters. If an SCO link is present using HV1 packets and length of *content* is less than 9 bytes the PDUs can be transmitted in DV packets. Otherwise DM1 packets must be used. All parameters have little endian format, i.e. the least significant byte is transmitted first.

The source/destination of the PDUs is determined by the AM_ADDR in the packet header.

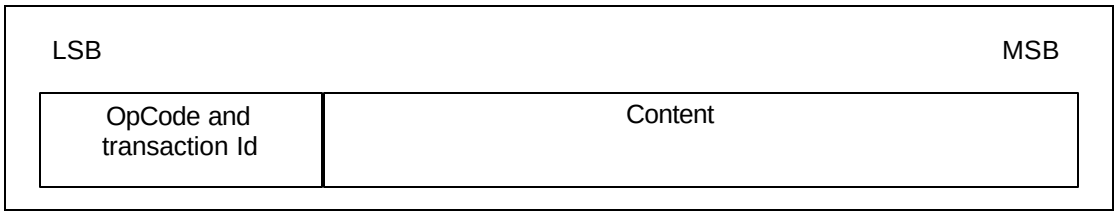


Figure 2.1: Payload body when LM PDUs are sent.

Each PDU is either mandatory or optional. The M/O field in the tables of Heading 1 indicates this. The LM does not need to be able to transmit a PDU that is optional. The LM must shall recognize all optional PDUs that it receives and, if a response is required, send a valid response according to the procedure rules in Heading 1. The reason that should be used in this case is *unsupported LMP feature*. If the optional PDU that is received does not require a response, no response is sent. Which of the optional PDUs a device supports can be requested, see Section 3.11 on page 207.

Each sequence described in section 3 is normally defined as a transaction. For pairing, see section 3.3, or encryption, see section 3.6, all sequences belonging to each section are counted as one transaction and shall use the same transaction ID. For connection establishment, see section 4, LMP_host_connection_req and the response with LMP_accepted or LMP_not_accepted form one transaction and have the transaction ID of 0. LMP_setup_complete is a stand-alone PDU, which forms a transaction by itself. For error handling, see section 7, the PDU to be rejected and LMP_not_accepted form a single transaction. Therefore the LMP_not_accepted shall have the same transaction ID as the PDU which is being rejected.

3 THE PROCEDURE RULES AND PDUs

Each procedure is described and depicted with a sequence diagram. The following symbols are used in the sequence diagrams:

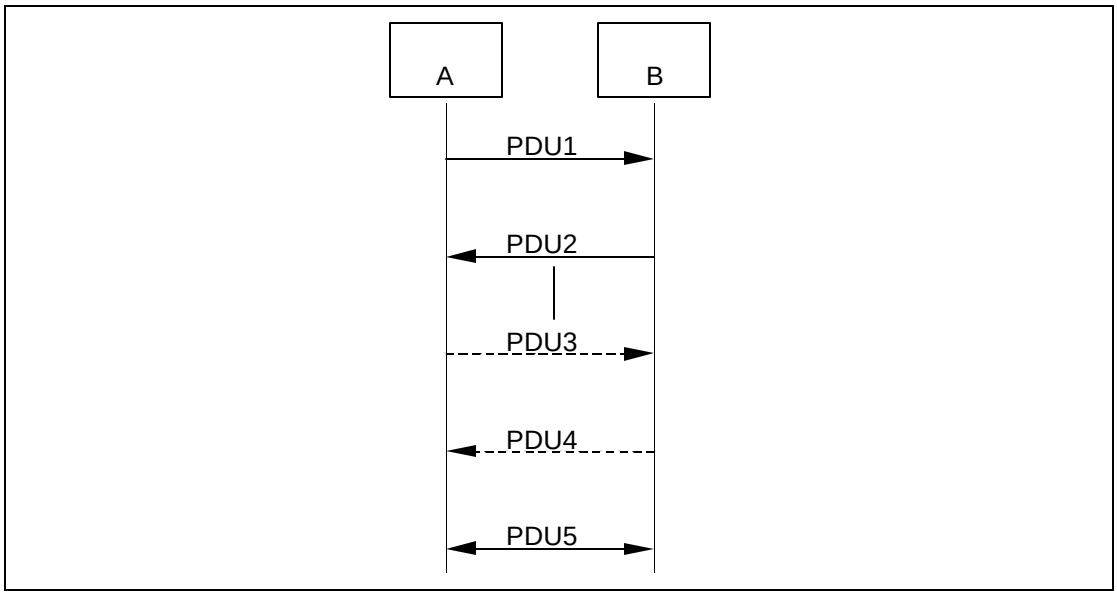


Figure 3.1: Symbols used in sequence diagrams.

PDU1 is a PDU sent from A to B. PDU2 is a PDU sent from B to A. PDU3 is a PDU that is optionally sent from A to B. PDU4 is a PDU that is optionally sent from B to A. PDU5 is a PDU sent from either A or B. A vertical line indicates that more PDUs can optionally be sent.

3.1 GENERAL RESPONSE MESSAGES

The PDUs LMP_accepted and LMP_not_accepted are used as response messages to other PDUs in a number of different procedures. The PDU LMP_accepted includes the opcode of the message that is accepted. The PDU LMP_not_accepted includes the opcode of the message that is not accepted and the reason why it is not accepted.

M/O	PDU	Contents
M	LMP_accepted	op code
M	LMP_not_accepted	op code reason

Table 3.1: General response messages.

3.2 AUTHENTICATION

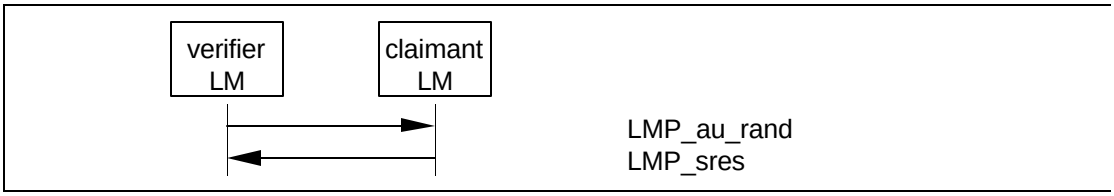
The authentication procedure is based on a challenge-response scheme as described in Baseband Specification Section 14.4, on page 168. The verifier sends an LMP_au_rand PDU which contains a random number (the challenge) to the claimant. The claimant calculates a response, which is a function of the challenge, the claimant's BD_ADDR and a secret key. The response is sent back to the verifier, which checks if the response was correct or not. How the response should be calculated is described in Baseband Specification Section 14.5.1, on page 170. A successful calculation of the authentication response requires that two devices share a secret key. How this key is created is described in Section 3.3 on page 195. Both the master and the slave can be verifiers. The following PDUs are used in the authentication procedure:

M/O	PDU	Contents
M	LMP_au_rand	random number
M	LMP_sres	authentication response

Table 3.2: PDUs used for authentication.

3.2.1 Claimant has link key

If the claimant has a link key associated with the verifier, it calculates the response and sends it to the verifier with LMP_sres. The verifier checks the response. If the response is not correct, the verifier can end the connection by sending LMP_detach with the reason code *authentication failure,* see Section 3.14 on page 211.

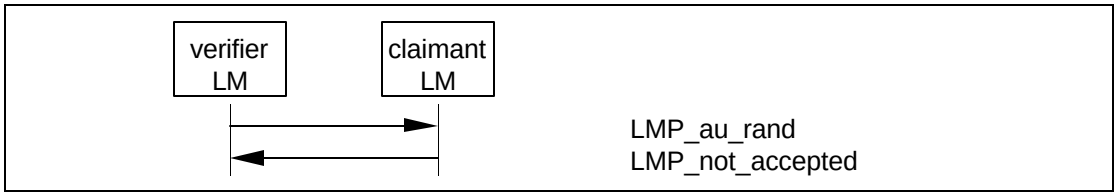


Sequence 1: Authentication. Claimant has link key.

If an LM receives LMP_au_rand and also wants to initiate an authentication it must shall first reply with LMP_sres before starting its own challenge. There can however be concurrent requests caused by master and slave simultaneously initiating an authentication. To avoid that this results in different ACOs in the units, this situation is resolved by the rule outlined in Section 7: If the master sends LMP_au_rand and receives another LMP_au_rand before receiving LMP_sres it shall respond with LMP_not_accepted with the reason code *LMP Error Transaction Collision;* in that case the slave LM shall complete the master's challenge by sending LMP_sres and may then initiate its authentication again.

3.2.2 Claimant has no link key

If the claimant does not have a link key associated with the verifier it sends LMP_not_accepted with the reason code *key missing* after receiving LMP_au_rand.



Sequence 2: Authentication fails. Claimant has no link key.

3.2.3 Repeated attempts

The scheme described in Baseband Specification Section 14.4.1, on page 169 shall be applied when an authentication fails. This will prevent an intruder from trying a large number of keys in a relatively short time.

3.3 PAIRING

When two devices do not have a common link key an initialization key (K_{init}) is created based on a PIN and a random number and a BD address. How the K_{init} is calculated is described in Baseband Specification Section 14.5.3, on page 174. When both devices have calculated K_{init} the link key is created, and finally a mutual authentication is made. The pairing procedure starts with a device sending LMP_in_rand; this device is referred to as "initiating LM" or "initiator" in Section 3.3.1 on page 196 - Section 3.3.5 on page 197. The other device is referred to as "responding LM" or "responder". The PDUs used in the pairing procedure are:

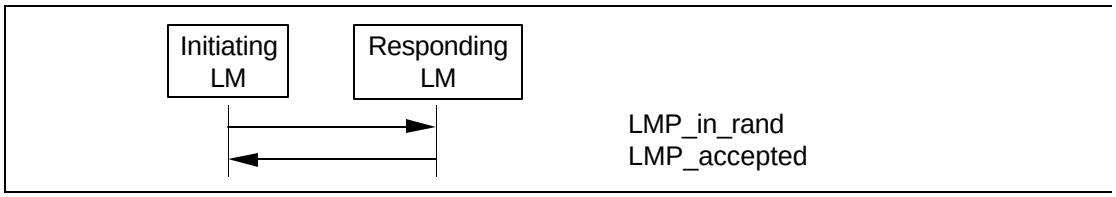
M/O	PDU	Contents
M	LMP_in_rand	random number
M	LMP_au_rand	random number
M	LMP_sres	authentication response
M	LMP_comb_key	random number
M	LMP_unit_key	key

Table 3.3: PDUs used for pairing

Note: all sequences described in section 3.3, including the mutual authentication after the link key has been created, form a single transaction. The transaction ID from the first LMP_in_rand will be used for all subsequent sequences.

3.3.1 Responder accepts pairing

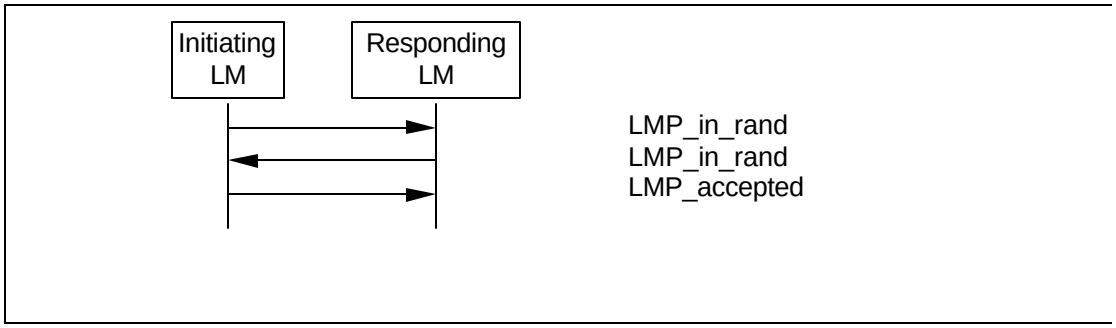
The initiator sends LMP_in_rand and the responder replies with LMP_accepted. Both devices calculate K_{init} based on the BD address of the responder and the procedure continues with creation of the link key; see Section 3.3.4 on page 197.



Sequence 3: Pairing accepted. Responder has a variable PIN.Initiator has a variable or fixed PIN.

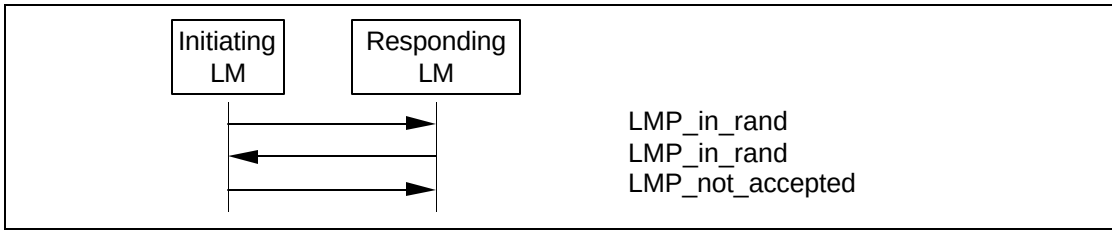
3.3.2 Responder has a fixed PIN

If the responder has a fixed PIN it shall generate a new random number and send it back in LMP_in_rand. If the initiator has a variable PIN it must shall accept this and respond with LMP_accepted. Both sides then calculate K_{init} based on the last IN_RAND and the BD address of the initiator. Thereafter the procedure continues with creation of the link key; see Section 3.3.4 on page 197.



Sequence 4: Responder has a fixed PIN and initiator has a variable PIN.

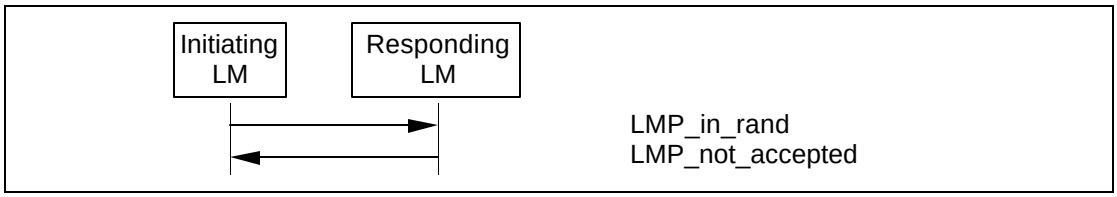
If the responder has a fixed PIN and the initiator also has a fixed PIN, the second LMP_in_rand is rejected by the initiator sending LMP_not_accepted with the reason code *pairing not allowed*; the pairing procedure is then ended.



Sequence 5: Both devices have a fixed PIN.

3.3.3 Responder rejects pairing

If the responder rejects pairing it sends LMP_not_accepted with the reason code *pairing not allowed* after receiving LMP_in_rand.



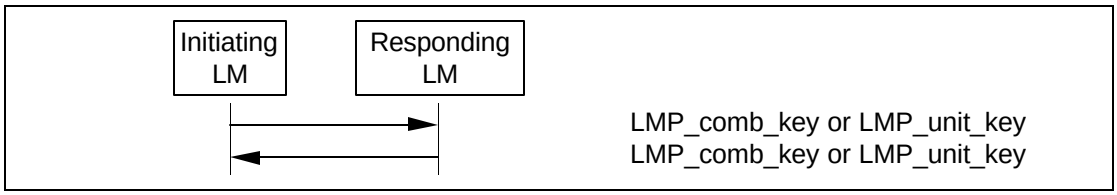
Sequence 6: Responder rejects pairing.

3.3.4 Creation of the link key

When K_{init} is calculated in both units the link key must be created. This link key will be used in the authentication between the two units for all subsequent connections until it is changed; see section 3.4 on page 198 and section 3.5 on page 199. The link key created in the pairing procedure will either be a combination key or one of the unit's unit keys. The following rules apply to the selection of the link key:

- if one unit sends LMP_unit_key and the other unit sends LMP_comb_key,the unit key will be the link key
- if both units send LMP_unit_key, the master's unit key will be the link key
- if both units send LMP_comb_key, the link key is calculated as described in Baseband Specification Section 14.2.2, on page 152.

The content of LMP_unit_key is the unit key bitwise XORed with K_{init}. The content of LMP_comb_key is LK_RAND bitwise XORed with K_{init}. Any device configured to use a combination key will store the link key.



Sequence 7: Creation of the link key.

3.3.5 Repeated attempts

When the authentication after creation of the link key fails because of a wrong authentication response, the same scheme as in Section 3.2.3 on page 195 is applied. This prevents an intruder from trying a large number of different PINs in a relatively short time.

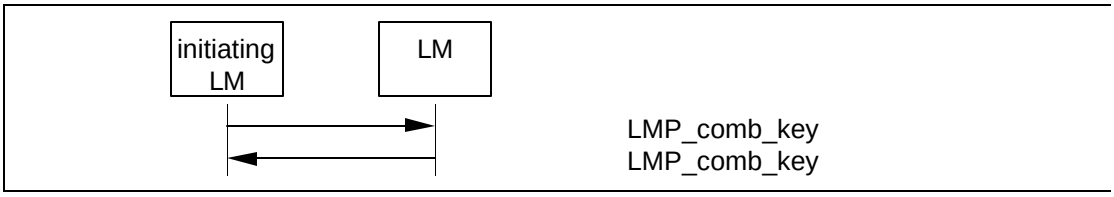
3.4 CHANGE LINK KEY

If the link key is derived from combination keys and the current link is the semi-permanent link key, the link key can be changed. If the link key is a unit key, the units must shall go through the pairing procedure in order to change the link key. The contents of LMP_comb_key is protected by a bitwise XOR with the current link key.

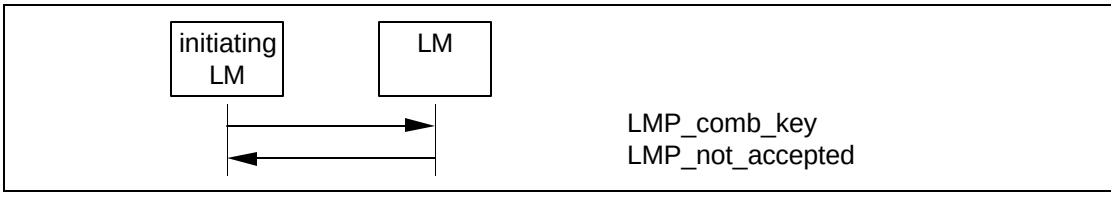
M/O	PDU	Contents
M	LMP_comb_key	random number

Table 3.4: PDUs used for change of link key.

Note: all sequences described in Section 3.4 on page 198, including the mutual authentication after the link key has been changed, form a single transaction. The transaction ID from the first LMP_comb_key will be used for all subsequent sequences.



Sequence 8: Successful change of the link key.



Sequence 9: Change of the link key not possible since the other unit uses a unit key.

If the change of link key is successful the new link key is stored and the old link key is discarded. The new link key will be used as link key for all the following connections between the two devices until the link key is changed again. The new link key also becomes the current link key. It will remain the current link key until the link key is changed again, or until a temporary link key is created, see Section 3.5 on page 199.

When the new link key has been created mutual authentication must shall be made to confirm that the same link key has been created in both units. The first authentication in the mutual authentication is made with the unit that initiated change link key as verifier. When finalized an authentication in the reversed direction is made.

3.5 CHANGE THE CURRENT LINK KEY

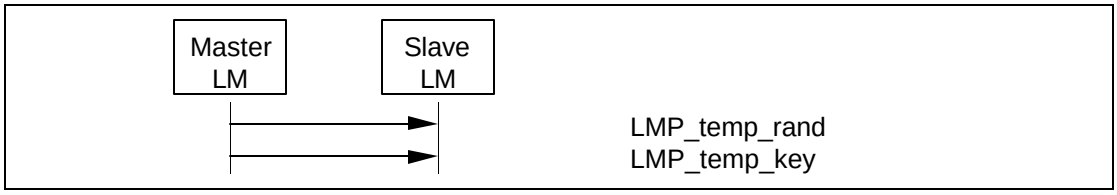
The current link key can be a semi-permanent link key or a temporary link key key. It can be changed temporarily, but the change is only valid for the session, see Baseband Specification Section 14.2.1, on page 150. Changing to a temporary link key is necessary if the piconet is to support encrypted broadcast.

M/O	PDU	Contents
M	LMP_temp_rand	random number
M	LMP_temp_key	key
M	LMP_use_semi_permanent_key	-

Table 3.5: PDUs used to change the current link key.

3.5.1 Change to a temporary link key

In the following, we use the same terms as in Baseband Specification Section 14.2.2.8, on page 157. The master starts by creating the master key K_{master} as described in Baseband Specification (EQ 27), on page 157. Then the master issues a random number RAND and sends it to the slave in LMP_temp_rand. Both sides can then calculate an overlay denoted OVL as OVL= E_{22}(current link key, RAND, 16). Then the master sends K_{master} protected by a modulo-2 addition with OVL to the slave in LMP_temp_key. The slave, who knows OVL, calculates K_{master}. After this, K_{master} becomes the current link key. It will be the current link key until the units fall back to the semi-permanent link key, see section 3.5.2 on page 200.



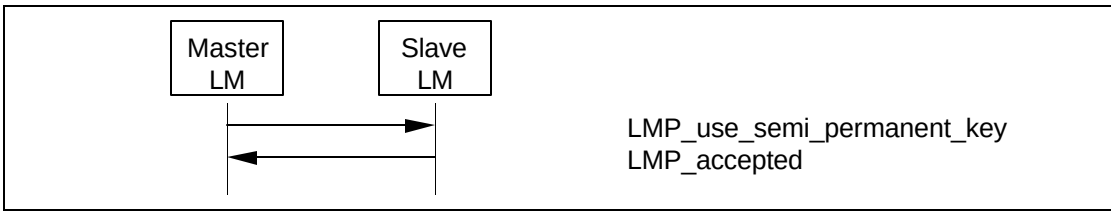
Sequence 10: Change to a temporary link key.

Note: all sequences described in section 3.4.1, including the mutual authentication after K_{master} has been created, form a single transaction. The transaction ID is set to 0.

When the units have changed to the temporary key a mutual authentication must shall be made to confirm that the same link key has been created in both units. The first authentication in the mutual authentication is made with the master as verifier. When finalized an authentication in the reversed direction is made.

3.5.2 Make the semi-permanent link key the current link key

After the current link key has been changed to K_{master}, this change can be undone and the semi-permanent link key becomes the current link key again. If encryption is used on the link, the procedure of going back to the semi-permanent link key must shall be immediately followed by a stop of the encryption by the master invoking the procedure described in Section 3.6.4 on page 203. Encryption can then be started again by the master according to the procedures in section 3.6.1 on page 201 subsection 3. This is to assure that encryption with encryption parameters known by other devices in the piconet is not used when the semi-permanent link key is the current link key.



Sequence 11: Link key changed to the semi-permanent link key.

3.6 ENCRYPTION

If at least one authentication has been performed encryption may be used. If the master wants all slaves in the piconet to use the same encryption parameters it must issue a temporary key (K_{master}) and make this key the current link key for all slaves in the piconet before encryption is started, see Section 3.5 on page 199. This is necessary if broadcast packets should be encrypted.

M/O	PDU	Contents
O	LMP_encryption_mode_req	encryption mode
O	LMP_encryption_key_size_req	key size
O	LMP_start_encryption_req	random number
O	LMP_stop_encryption_req	-

Table 3.6: PDUs used for handling encryption.

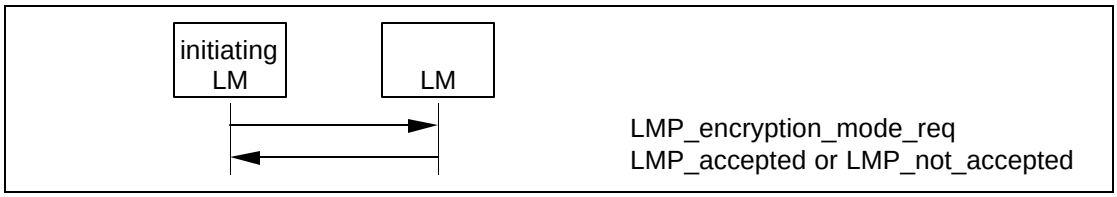
Note: all sequences described in Section 3.6 on page 201 form a single transaction. The transaction ID from the LMP_encryption_mode_req will be used for all subsequent sequences.

3.6.1 Encryption mode

First of all the master and the slave must agree upon whether to use encryption or not and if encryption shall only apply to point-to-point packets or if encryption shall apply to both point-to-point packets and broadcast packets. If master and slave agree on the encryption mode, the master continues to give more detailed information about the encryption.

The initiating LM finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and sends LMP_encryption_mode_req. If the change in encryption mode is accepted then the other device finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and responds with LMP_accepted.

L2CAP transmission is re-enabled when the attempt to encrypt or decrypt the link is completed i.e. at the end of Sequence 14, 15 or 16.

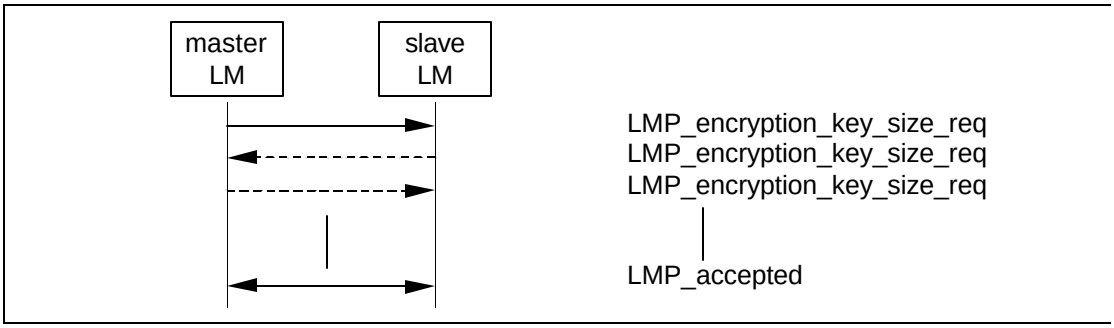


Sequence 12: Negotiation for encryption mode.

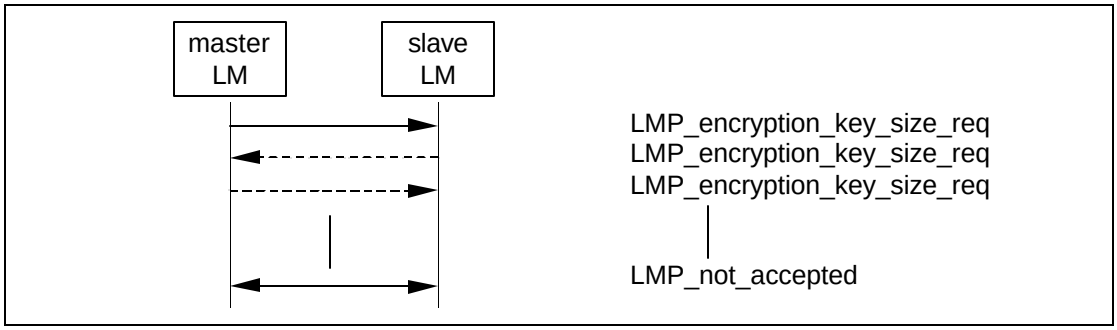
After a unit has sent LMP_encryption_mode_req it is not allowed to send LMP_au_rand before encryption is actually switched on. After a unit has received LMP_encryption_mode_req and sent LMP_accepted it is not allowed to send LMP_au_rand before encryption is actually switched on. If an LMP_au_rand is still sent violating these rules, the claimant shall respond with LMP_not_accepted with the reason code *PDU not allowed*. This is to avoid that the units have different ACOs when they calculate the encryption key. If the encryption mode is not accepted or the encryption key size negotiation results in disagreement the units are allowed to send LMP_au_rand again.

3.6.2 Encryption key size

The next step is to determine the size of the encryption key. In the following we use the same terms as in Baseband Specification Section 14.3.1, on page 159. The master sends LMP_encryption_key_size_req including the suggested key size Lsug, m, which is initially equal to $L_{max,\,m}$. If $L_{min,\,s} \leq L_{sug,\,m}$ and the slave supports $L_{sug,\,m}$ it responds with LMP_accepted and $L_{sug,\,m}$ will be used as the key size. If both conditions are not fulfilled the slave sends back LMP_encryption_key_size_req including the slave's suggested key size $L_{sug,\,s}$. This value is the slave's largest supported key size that is less than $L_{sug,\,m}$. Then the master performs the corresponding test on the slave's suggestion. This procedure is repeated until a key size agreement is reached or it becomes clear that no such agreement can be reached. If an agreement is reached a unit sends LMP_accepted and the key size in the last LMP_encryption_key_size_req will be used. After this, the encryption is started; see Section 3.6.3 on page 203. If an agreement is not reached a unit sends LMP_not_accepted with the reason code *Unsupported parameter value* and the units are not allowed to communicate using Bluetooth link encryption.".



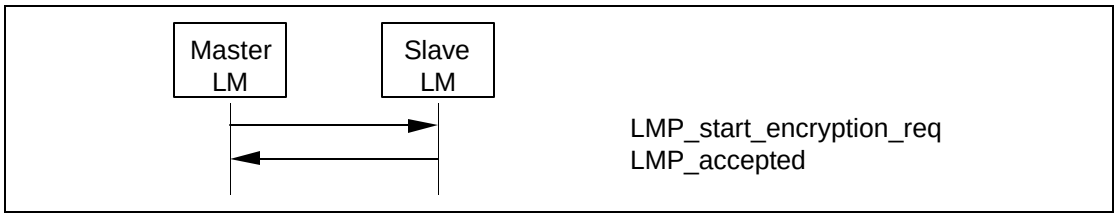
Sequence 13: Encryption key size negotiation successful.



Sequence 14: Encryption key size negotiation failed.

3.6.3 Start encryption

Finally, encryption is started. The master issues the random number EN_RAND and calculates the encryption key as $K_c=E_3$(current link key, EN_RAND, COF). See Baseband Specification Section 14.2.2.5, on page 155 and 14.2.2.2 for the definition of the COF. The random number must be the same for all slaves if the piconet should support encrypted broadcast. Then the master sends LMP_start_encryption_req, which includes EN_RAND. The slave calculates K_c when this message is received and acknowledges with LMP_accepted.



Sequence 15: Start of encryption.

The start of encryption will be done in three steps:

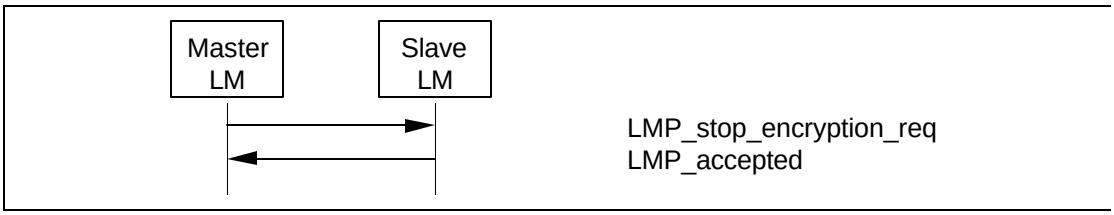
1. Master is configured to transmit unencrypted packets, but to receive encrypted packets.
2. Slave is configured to transmit and receive encrypted packets.
3. Master is configured to transmit and receive encrypted packets.

Between step 1 and step 2, master-to-slave transmission is possible. This is when LMP_start_encryption_req is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3, slave-to-master transmission is possible. This is when LMP_accepted is transmitted. Step 3 is triggered when the master receives this message.

3.6.4 Stop encryption

If a unit wants to stop encryption it sends LMP_encryption_mode_req with the parameter encryption mode equal to 0 (no encryption). The other device

responds with LMP_accepted or LMP_not_accepted (the procedure is described in Sequence 12 in section 3.6.1 on page 201). If accepted encryption is stopped by the master sending LMP_stop_encryption_req and the slave responding with LMP_accepted according to Sequence 16.



Sequence 16: Stop of encryption.

Stopping of encryption is then done in three steps, similar to the procedure for starting encryption.

1. Master is configured to transmit encrypted packets, but to receive unencrypted packets.
2. Slave is configured to transmit and receive unencrypted packets.
3. Master is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 master to slave transmission is possible. This is when LMP_stop_encryption_req is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3 slave to master transmission is possible. This is when LMP_accepted is transmitted. Step 3 is triggered when the master receives this message.

3.6.5 Change encryption mode, key or random number

If the encryption key or encryption random number need to be changed or if the encryption mode needs to be changed between 1 (point-to-point) and 2(point-to-point and broadcast), encryption must first be stopped and then re-started with the new parameters according to the procedures in section 3.6 on page 201, subsections 1-3.

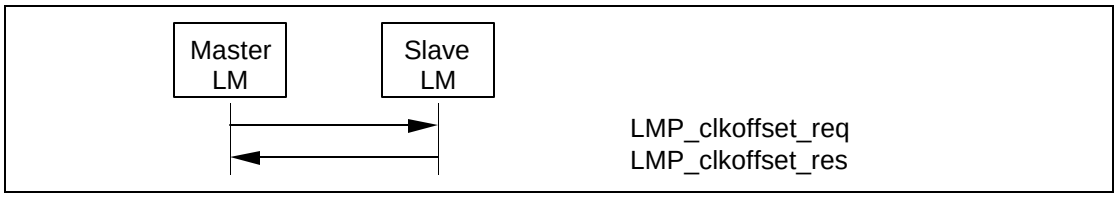
3.7 CLOCK OFFSET REQUEST

When a slave receives the FHS packet, the difference is computed between its own clock and the master's clock included in the payload of the FHS packet. The clock offset is also updated each time a packet is received from the master. The master can request the clock offset at anytime following a successful baseband paging procedure (i.e., before, during or after connection setup).

By saving this clock offset the master knows on what RF channel the slave wakes up to PAGE SCAN after it has left the piconet. This can be used to speed up the paging time the next time the same device is paged.

M/O	PDU	Contents
M	LMP_clkoffset_req	-
M	LMP_clkoffset_res	clock offset

Table 3.7: PDUs used for clock offset request.



Sequence 17: Clock offset requested.

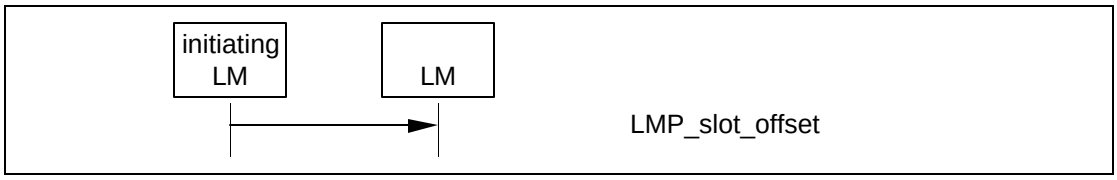
3.8 SLOT OFFSET INFORMATION

With LMP_slot_offset the information about the difference between the slot boundaries in different piconets is transmitted. This PDU carries the parameters slot offset and BD_ADDR. The slot offset is the subtraction of time in µs of the start of the master's TX slot in the piconet where the PDU is transmitted from the time in µs of the start of the master's TX slot in the piconet where the BD_ADDR device is master modulo 1250.

See Section 3.12 on page 208 for the use of LMP_slot_offset in the context of the master-slave switch..

M/O	PDU	Contents
O	LMP_slot_offset	slot offset BD_ADDR

Table 3.8: PDU used for slot offset information.



Sequence 18: Slot offset information is sent.

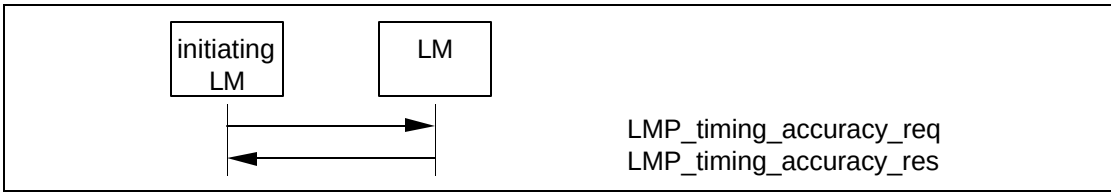
3.9 TIMING ACCURACY INFORMATION REQUEST

LMP supports requests for the timing accuracy. This information can be used to minimize the scan window for a given hold time when returning from hold and to extend the maximum hold time. It can also be used to minimize the scan window when scanning for the sniff mode slots or the park mode beacon packets. The timing accuracy parameters returned are the long term drift measured in ppm and the long term jitter measured in µs of the clock used during hold,

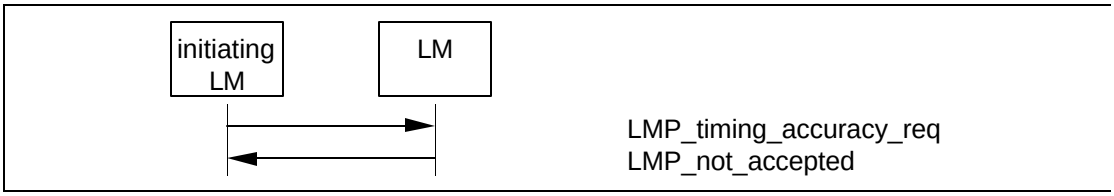
sniff and park mode. These parameters are fixed for a certain device and must shall be identical when requested several times. Timing accuracy can be requested at anytime following a successful baseband paging procedure, provided this PDU is shown as supported in the supported features list. If the timing accuracy request is not supported, the requesting device must shall assume worst case values (drift=250ppm and jitter=10?s).

M/O	PDU	Contents
O	LMP_timing_accuracy_req	-
O	LMP_timing_accuracy_res	drift jitter

Table 3.9: PDUs used for requesting timing accuracy information.



Sequence 19: The requested device supports timing accuracy information.



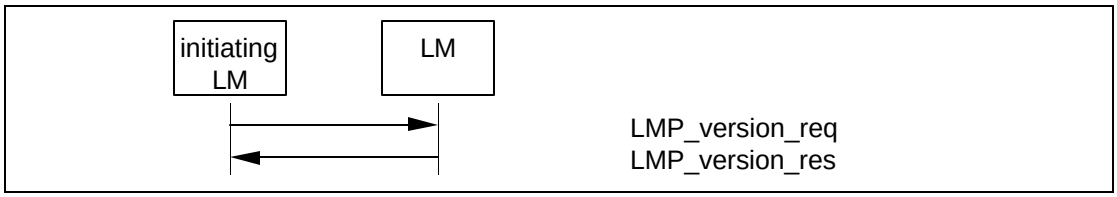
Sequence 20: The requested device does not support timing accuracy information.

3.10 LMP VERSION

LMP supports requests for the version of the LM protocol. The requested device will send a response with three parameters: VersNr, CompId and SubVersNr. VersNr specifies the version of the Bluetooth LMP specification that the device supports. CompId is used to track possible problems with the lower Bluetooth layers. All companies that create a unique implementation of the Link Manager shall have their own CompId. The same company is also responsible for the administration and maintenance of the SubVersNr. It is recommended that each company has a unique SubVersNr for each RF/BB/LM implementation. For a given VersNr and CompId, the values of the SubVersNr must shall increase each time a new implementation is released. For both CompId and SubVersNr the value 0xFFFF means that no valid number applies. There is no ability to negotiate the version of the LMP. The sequence below is only used to exchange the parameters. LMP version can be requested at anytime following a successful baseband paging procedure.

M/O	PDU	Contents
M	LMP_version_req	VersNr CompId SubVersNr
M	LMP_version_res	VersNr CompId SubVersNr

Table 3.10: PDUs used for LMP version request.



Sequence 21: Request for LMP version.

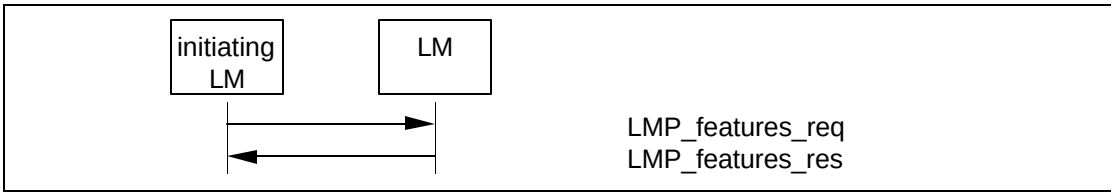
3.11 SUPPORTED FEATURES

The Bluetooth radio and link controller may support only a subset of the packet types and features described in Baseband Specification and Radio Specification. The PDU LMP_features_req and LMP_features_res are used to exchange this information. The supported features can be requested at anytime following a successful baseband paging procedure. A device may not send any packets other than ID, FHS, NULL, POLL, DM1 or DH1 before it is aware of the supported features of the other device. After the features request has been carried out, the intersection of the supported packet types for both sides may also be transmitted. Whenever a request is issued, it must shall be compatible with the supported features of the other device. For instance, when establishing an SCO link the initiator may not propose to use HV3 packets if

that packet type is not supported by the other device. Exceptions to this rule are LMP_switch_ req and LMP_slot_offset, which can be sent before the requesting side is aware of the other side's features (switch is an optional feature)

M/O	PDU	Contents
M	LMP_features_req	features
M	LMP_features_res	features

Table 3.11: PDUs used for features request.



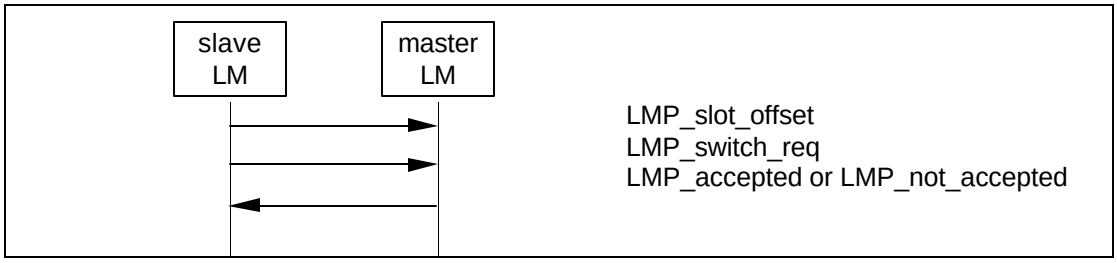
Sequence 22: Request for supported features.

3.12 SWITCH OF MASTER-SLAVE ROLE

Since the paging device always becomes the master of the piconet, a switch of the master-slave role is sometimes needed, see Baseband Specification Section 10.9.3, on page 121.

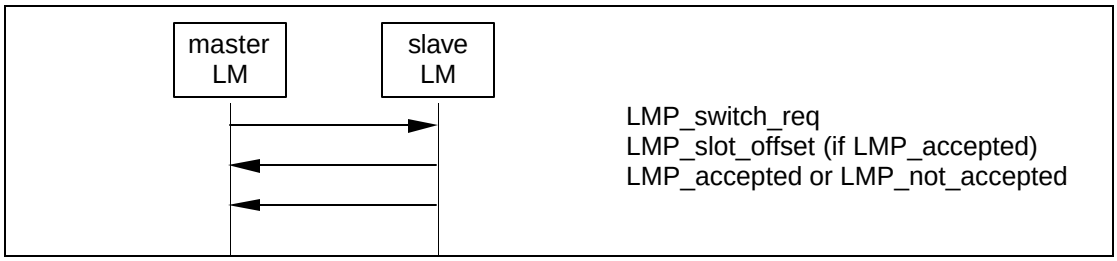
M/O	PDU	Contents
O	LMP_switch_req	switch instant
O	LMP_slot_offset	slot offset BD_ADDR

*PDUs used for master-slave switch.*If the slave initiates the master-slave switch it finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and sends LMP_slot_offset immediately followed by LMP_switch_req. If the master accepts the master-slave switch it finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and responds with LMP_accepted. When the master-slave switch has been completed on Baseband level (successfully or not) both units re-enable L2CAP transmission. If the master rejects the master-slave switch it responds with LMP_not_accepted and the slave re-enables L2CAP transmission. The transaction ID for all PDUs in the sequence is set to 1.



Sequence 23: Master-slave switch (slave initiated).

If the master initiates the master-slave switch it finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and sends LMP_switch_req. If the slave accepts the master-slave switch it finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and responds with LMP_slot_offset immediately followed by LMP_accepted. When the master-slave switch has been completed on Baseband level (successfully or not) both units re-enable L2CAP transmission. If the slave rejects the master-slave switch it responds with LMP_not_accepted and the master re-enables L2CAP transmission. The transaction ID for all PDUs in the sequence is set to 0.



Sequence 24: Master-slave switch (master initiated).

The LMP_switch_req PDU contains a parameter, switch instant, which specifies the instant at which the TDD switch is made. This is specified as a Bluetooth clock value of the master's clock, which is available to both devices. This instant is chosen by the sender of the message and should be at least $2*T_{poll}$ or 32 (whichever is greater) slots in the future. The assumption is that the switch instant is always within 12 hours of the current clock value, in order to accommodate clock wrap.

The sender of the LMP_switch_req selects the switch instant and queues the LMP_switch_req to LC for transmission and starts a timer to expire at the switch instant. When the timer expires it initiates the mode switch. In the case of a master initiated switch if the LMP_slot_offset has not been received by the switch instant the master slave switch is carried out without an estimate of the slave's slot offset. If LMP_not_accepted is received before the timer expires then the timer is stopped and the role switch is not initiated.

When the LMP_switch_req is received the switch instant is compared with the current master clock value. If it is in the past then the instant has been passed

and LMP_not_accepted with the reason code Instant passed shall be returned. If it is in the future then LMP_accepted is returned assuming the master-slave switch is allowed and a timer is started to expire at the switch instant. When this timer expires it initiates the mode switch.

Support for LMP_slot_offset is mandatory if LMP_switch_req is supported.

3.13 NAME REQUEST

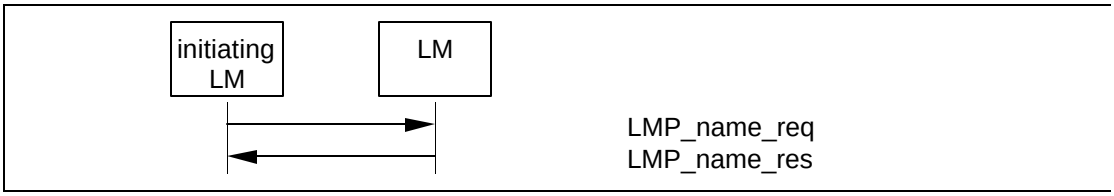
LMP supports name request to another Bluetooth device. The name is a user-friendly name associated with the Bluetooth device and consists of a maximum of 248 bytes coded according to the UTF-8 standard. The name is fragmented over one or more DM1 packets. When the LMP_name_req is sent, a name offset indicates which fragment is expected. The corresponding LMP_name_res carries the same name offset, the name length indicating the total number of bytes in the name of the Bluetooth device and the name fragment, where:

- name fragment(N) = name(N + name offset), if (N + name offset) < name length
- name fragment(N) = 0,otherwise.

Here 0 ??N ??13. In the first sent LMP_name_req, name offset=0. Sequence 25 is then repeated until the initiator has collected all fragments of the name. The name request can be made at anytime following a successful baseband paging procedure.

M/O	PDU	Contents
M	LMP_name_req	name offset
M	LMP_name_res	name offset name length name fragment

Table 3.12: PDUs used for name request.



Sequence 25: Device's name requested and it responses.

3.14 DETACH

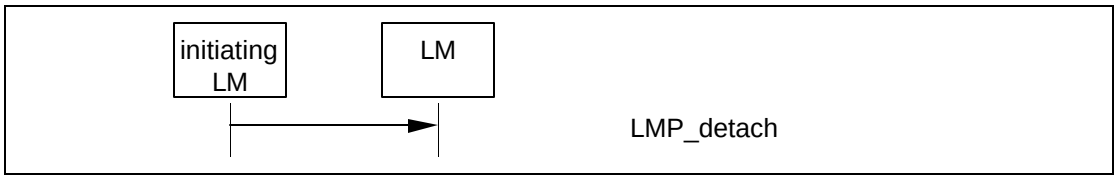
The connection between two Bluetooth devices can be closed anytime by the master or the slave. A reason parameter is included in the message to inform the other party of why the connection is closed.

M/O	PDU	Contents
M	LMP_detach	reason

Table 3.13: PDU used for detach.

The initiating LM first finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission. The initiating LM then queues the LMP_detach for transmission and starts a timer for $6*T_{poll}$ slots where T_{poll} is the poll interval for the connection. If the initiating LM receives the Baseband-level acknowledgement before the timer expires it now starts a timer for $3*T_{poll}$ slots. When this timer expires (and if the initiating LM is the master) the AM_ADDR can be re-used immediately. If the initial timer expires then the initiating LM drops the link and starts a timer for $T_{linksupervisiontimeout}$ slots after which the AM_ADDR can be re-used (if the initiating LM is the master).

When the receiving LM receives the LMP_detach, it starts a timer for $6*T_{poll}$ slots if it is the master and $3*T_{poll}$ if it is the slave. On timer expiration, the link is detached and (if the receiving LM is the master) the AM_ADDR can be re-used immediately. If the receiver never gets the LMP_detach then a link supervision timeout will occur and the link will be detached.



Sequence 26: Connection closed by sending LMP_detach.

3.15 HOLD MODE

The ACL link of a connection between two Bluetooth devices can be placed in hold mode for a specified hold time. During this time no ACL packets will be transmitted from the master. The hold mode is typically entered when there is no need to send data for a relatively long time. The transceiver can then be turned off in order to save power. But the hold mode can also be used if a device wants to discover or be discovered by other Bluetooth devices, or wants to join other piconets. What a device actually does during the hold time is not controlled by the hold message, but it is up to each device to decide.

M/O	PDU	Contents
O	LMP_hold	hold time, hold instant
O	LMP_hold_req	hold time, hold instant

Table 3.14: PDUs used for hold mode.

The LMP_hold and LMP_hold_req PDUs both contain a parameter, hold instant, which specifies the instant at which the hold will become effective. This is specified as a Bluetooth clock value of the master's clock, which is available to both devices. This instant is chosen by the sender of the message and should be at least $6*T_{poll}$ slots in the future. The assumption is that the hold instant is always within 12 hours of the current clock value, in order to accommodate clock wrap.

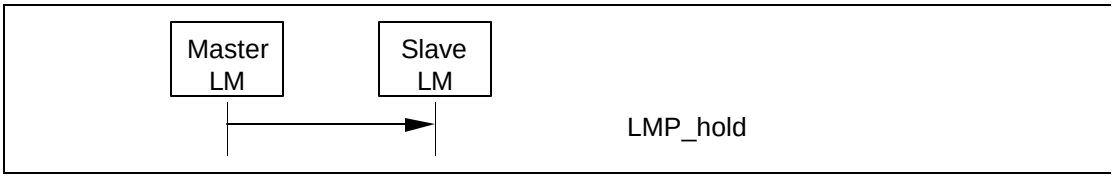
3.15.1 Master forces hold mode

The master can force hold mode if there has previously been a request for hold mode that has been accepted. The hold time included in the PDU when the master forces hold mode cannot be longer than any hold time the slave has previously accepted when there was a request for hold mode.

The master LM first finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission. It selects the hold instant and queues the LMP_hold to its LC for transmission. It then starts a timer to wait until the hold instant occurs. When this timer expires then the connection enters hold mode. Note that the Baseband-level acknowledgement is ignored in this mechanism.

When the slave LM receives the LMP_hold it compares the hold instant with the current master clock value. If it is in the future then it starts a timer to expire at this instant and enters hold mode when it expires.

When the master LM exits from Hold mode it re-enables L2CAP transmission.



Sequence 27: Master forces slave into hold mode.

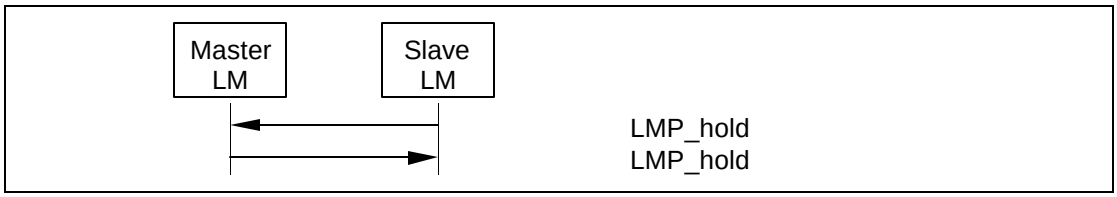
3.15.2 Slave forces hold mode

The slave can force hold mode if there has previously been a request for hold mode that has been accepted. The hold time included in the PDU when the slave forces hold mode cannot be longer than any hold time the master has previously accepted when there was a request for hold mode.

The slave LM first finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission. It selects the hold instant and queues the LMP_hold to its LC for transmission. It then waits for the LMP_hold from the master acting according to the procedure described in Section 3.15.1 .

When the master LM receives the LMP_hold it finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission. It then inspects the hold instant. If this is less than $6*T_{poll}$ slots in the future it should modify the instant so that it is at least $6*T_{poll}$ slots in the future. Then it sends the LMP_hold using the mechanism described in Section 3.15.1 .

When the master and slave LMs exit from Hold mode they re-enable L2CAP transmission.



Sequence 28: Slave forces master into hold mode.

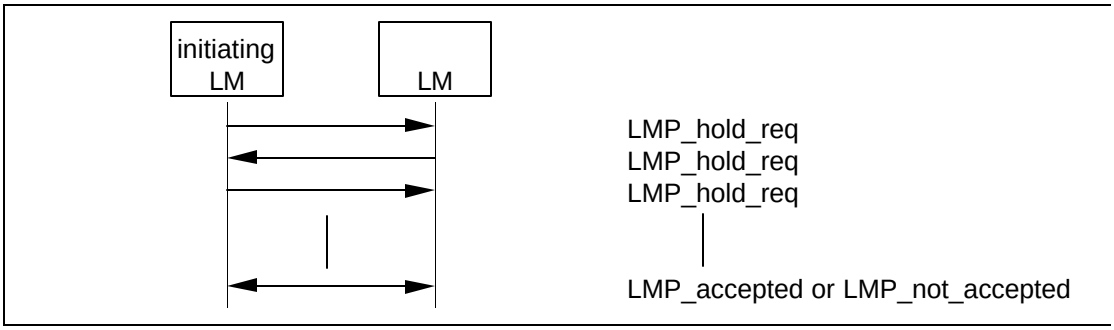
3.15.3 Master or slave requests hold mode

The master or the slave can request to enter hold 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 hold mode. If no agreement is seen, LMP_not_accepted with the reason code *unsupported parameter value* terminates the negotiation and hold mode is not entered.

The initiating LM first finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission. On receiving the LMP_hold_req the receiving LM finalizes the transmission of the current ACL packet with L2CAP information and stops L2CAP transmission

The LM sending LMP_hold_req selects the hold instant, which should be at least $9*T_{poll}$ slots in the future. If this is a response to a previous LMP_hold_req and the contained hold instant is at least $9*T_{poll}$ slots in the future then this should be used. The LMP_hold_req is then queued to its LC for transmission and a timer is started to expire at this instant and the connection enters hold mode when it expires unless an LMP_not_accepted or LMP_hold_req is received by its LM before that point. If the LM receiving LMP_hold_req agrees to enter hold mode it returns LMP_accepted and starts a timer to expire at the hold instant. When this timer expires it enters hold mode.

When each LM exits from Hold mode it re-enables L2CAP transmission.



Sequence 29: Negotiation for hold mode.

3.16 SNIFF MODE

To enter sniff mode, master and slave negotiate a sniff interval T_{sniff} and a sniff offset, D_{sniff}, which specifies the timing of the sniff slots. The offset determines the time of the first sniff slot; after that the sniff slots follows periodically with the sniff interval T_{sniff}. To avoid problems with a clock wrap-around during the 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: 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 must 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 must shall listen if it continues to receive only packets with its own AM address.

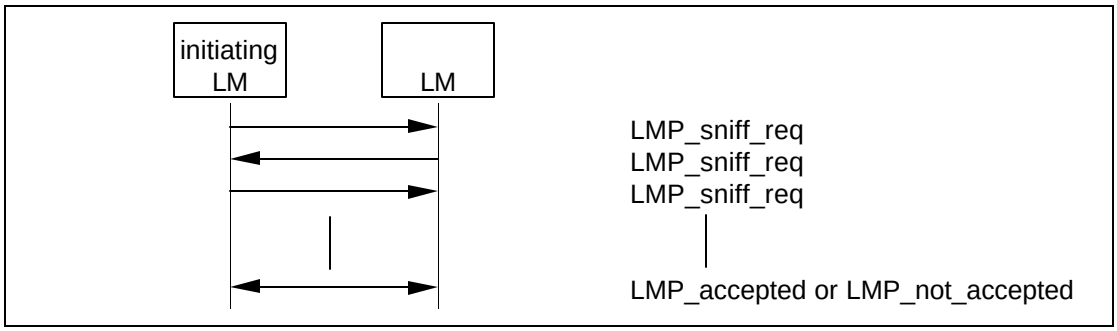
M/O	PDU	Contents
O	LMP_sniff_req	timing control flags D_{sniff} T_{sniff} sniff attempt sniff timeout
O	LMP_unsniff_req	-

Table 3.15: PDUs used for sniff mode.

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

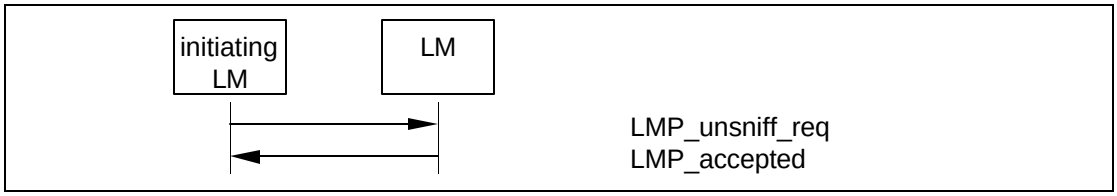
is seen, LMP_not_accepted with the reason code *unsupported parameter value* terminates the negotiation and sniff mode is not entered.



Sequence 30: Negotiation for sniff mode.

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 must 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.



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

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 Baseband Specification Section 10.8.4, on page 112. 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 must shall be identified with its BD_ADDR when it is unparked by the master.

M/O	PDU	Contents
O	LMP_park_req	timing control flags D_B T_B N_B $?_B$ 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 $?_B$ 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)

Table 3.16: PDUs used for park mode.

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: If the master never receives the LMP_accepted then a link supervision timeout will occur.

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

3.17.2 Slave requests to enter park mode

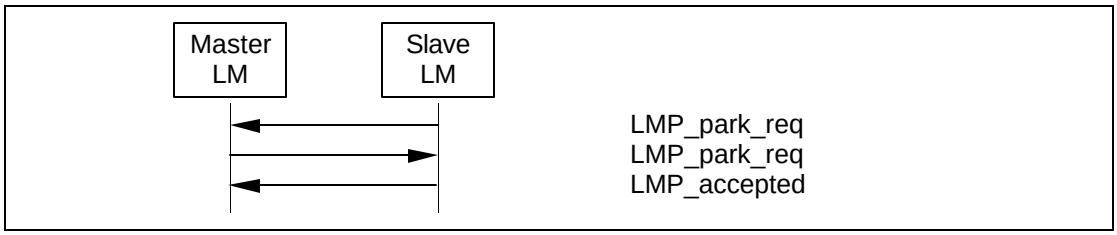
The slave can request park mode. The slave finalizes the transmission of the current ACL packet with L2CAP information, stops L2CAP transmission and then sends LMP_park_req. The parameters PM_ADDR and AR_ADDR are not valid and the other parameters represent suggested values. If the master wants the slave 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 parameter 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.

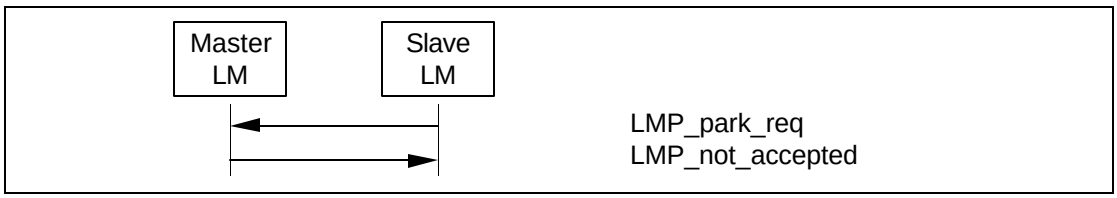
When the master receives LMP_accepted it starts a timer for $6*T_{poll}$ slots. When this timer expires the slave is in park mode and the AM_ADDR can be re-used.

Note: If the master never receives the LMP_accepted then a link supervision timeout will occur.



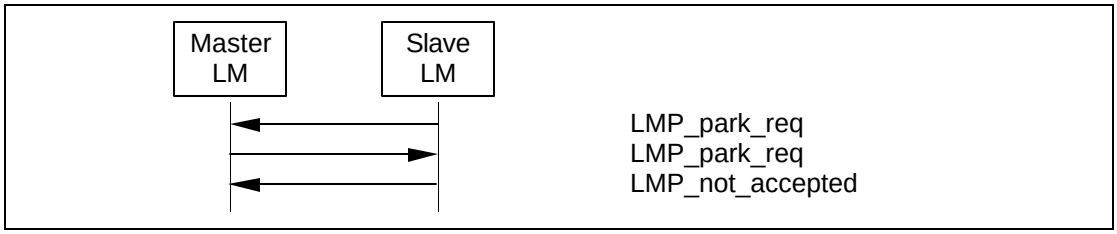
Sequence 34: Slave requests to enter park mode and accepts master's beacon parameters.

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

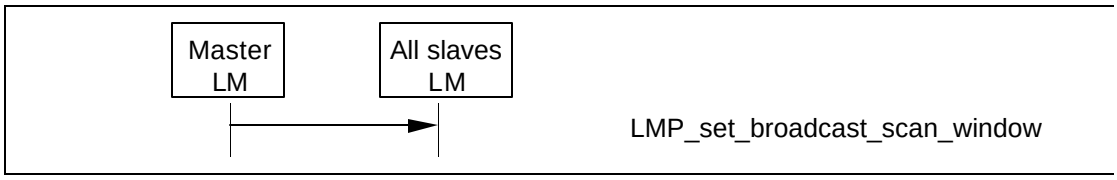
If the slave does not accept the parameters in LMP_park_req sent from the master it responds with LMP_not_accepted and both units re-enable L2CAP transmission.



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

3.17.3 Master sets up broadcast scan window

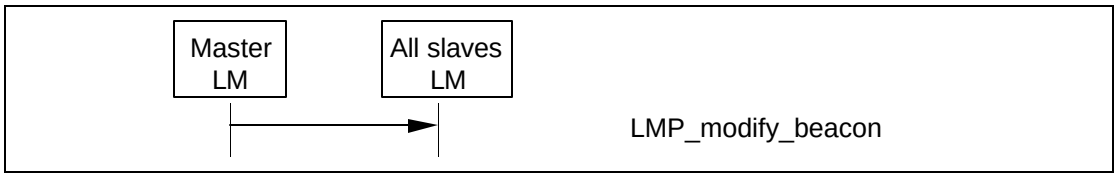
If more broadcast capacity is needed than the beacon train, the master can indicate to the slaves that more broadcast information will follow the beacon train by sending LMP_set_broadcast_scan_window. This message is always sent in a broadcast packet at the beacon slot(s). The scan window starts in the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



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

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.

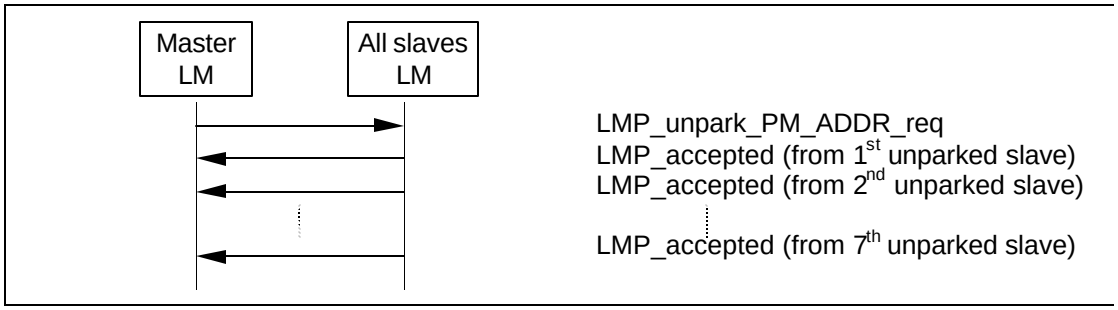
3.17.5 Unparking slaves

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 must 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 must 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 must 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 7 slaves can be unparked with the same message. If they are identified with the BD_ADDR a maximum of 2 slaves can be unparked with the same message.

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

Sequence 39: Master unparks slaves addressed with their BD_ADDR.



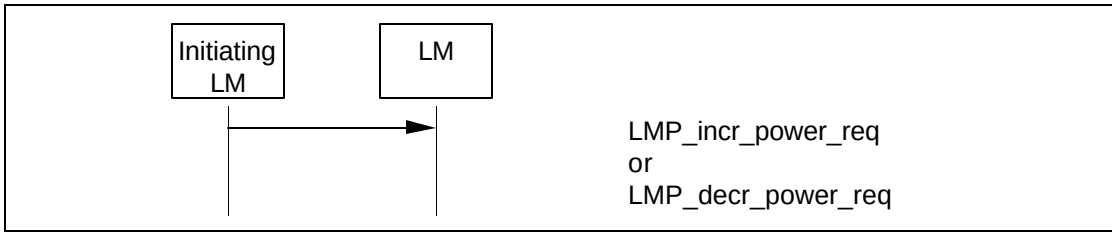
Sequence 40: Master unparks slaves addressed with their PM_ADDR.

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 Radio Specification Section 3.1, on page 22 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.

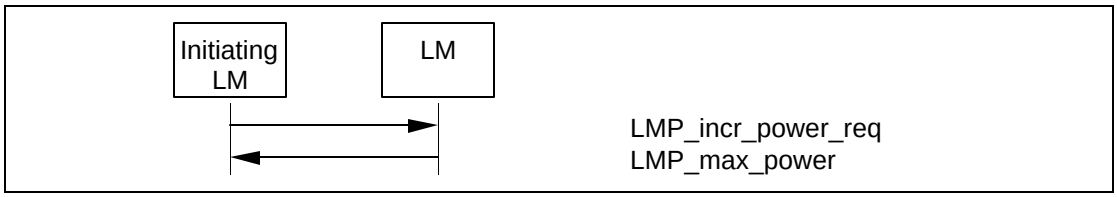
M/O	PDU	Contents
O	LMP_incr_power_req	for future use (1 Byte)
O	LMP_decr_power_req	for future use (1 Byte)
O	LMP_max_power	-
O	LMP_min_power	-

Table 3.17: PDUs used for power control.

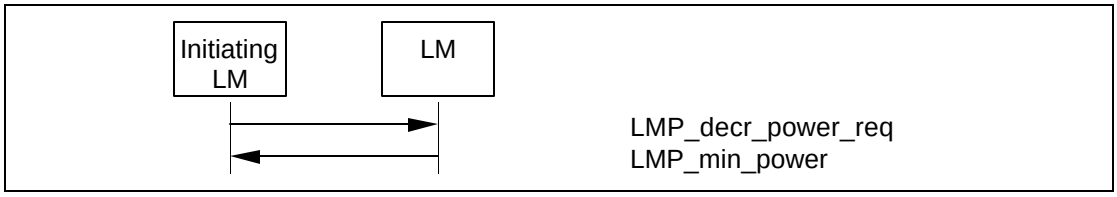


Sequence 41: A device requests a change of the other device's TX power.

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 must shall be 0x00 for all versions of LMP where this parameter is not yet defined.

3.19 CHANNEL QUALITY-DRIVEN CHANGE BETWEEN DM AND DH

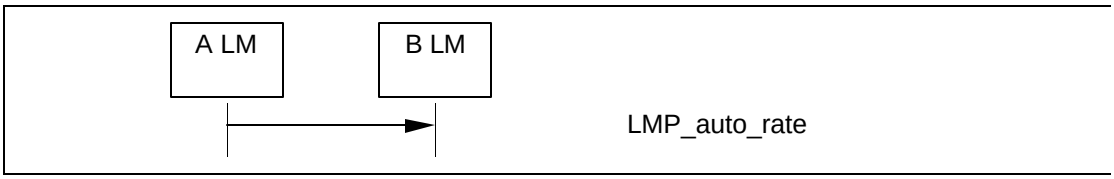
The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. If a device A wants the remote device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

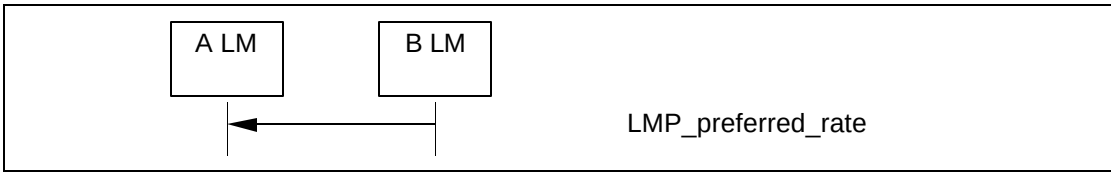
tures list of the device shows that it does not support the channel quality driven change requests.

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

Table 3.18: PDUs used for quality driven change of the data rate.



Sequence 44: The left-hand unit is configured to automatically change between DM and DH.



Sequence 45: A wants B to control A's packet type.

3.20 QUALITY OF SERVICE (QoS)

The Link Manager provides Quality of Service 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.

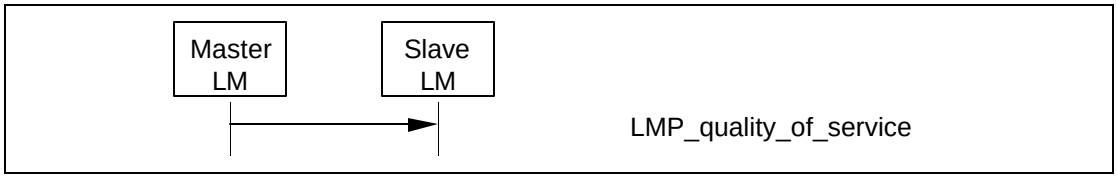
In addition, master and slave negotiate the number of repetitions for broadcast packets (N_{BC}), see Baseband Specification Section 5.3.5, on page 71.

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

Table 3.19: PDUs used for quality of service.

3.20.1 Master notifies slave of the quality of service

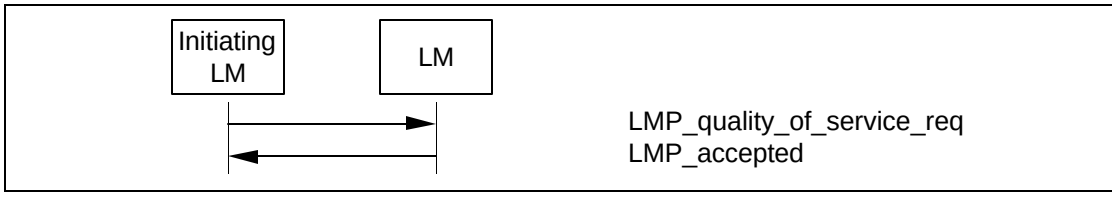
In this case the master notifies the slave of the new poll interval and N_{BC}. The slave cannot reject the notification.



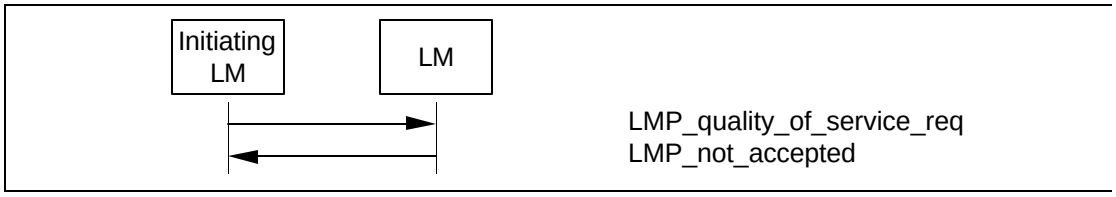
Sequence 46: B changes A's packet type.

3.20.2 Device requests new quality of service

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 quality of service as needed.



Sequence 47: Device accepts new quality of service



Sequence 48: Device rejects new quality of service.

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 bit0 and bit1 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.

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

Table 3.20: PDUs used for managing the SCO links.

3.21.1 Master initiates an SCO link

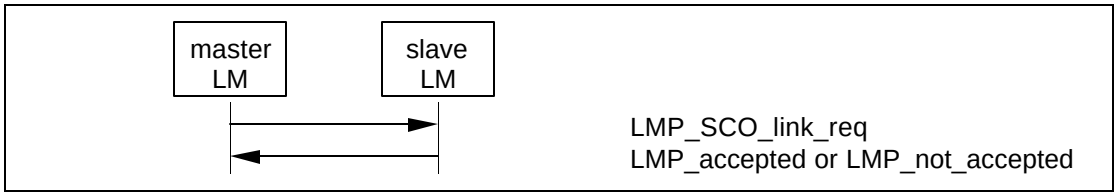
When establishing an SCO link the master sends a request with parameters that specify the timing, packet type and coding that will be used on the SCO link. For each of the SCO packets Bluetooth supports three different voice coding formats on the air-interface: ?-law log PCM, A-law log PCM and CVSD. 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.

The SCO handle in the message must shall be different from any already existing SCO link(s).

Note: If the SCO packet type is HV1 the LMP_accepted must shall be sent using the DM1 packet.

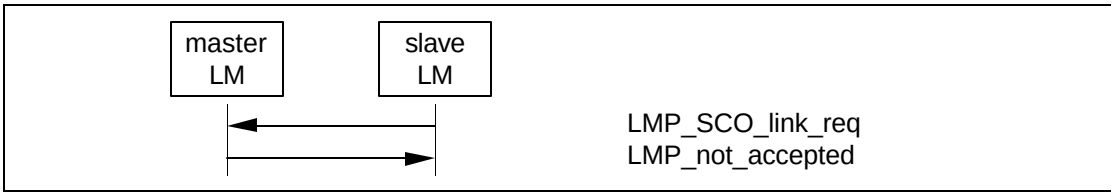


Sequence 49: Master requests an SCO link.

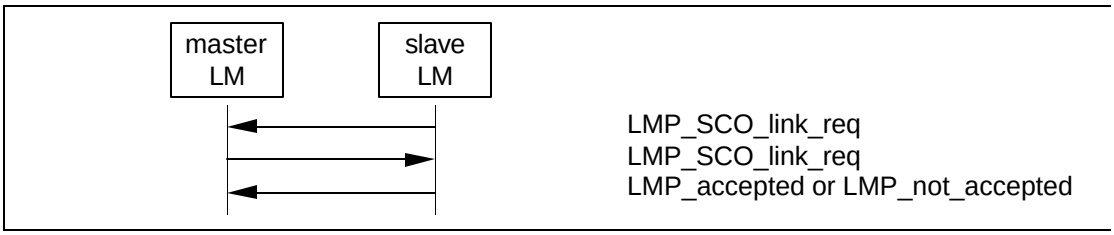
3.21.2 Slave initiates an SCO link

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 must 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 must shall then reply with LMP_accepted or LMP_not_accepted.

"Note: If the SCO packet type is HV1 the LMP_accepted must 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.

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 Section 3.21.1 on page 227.

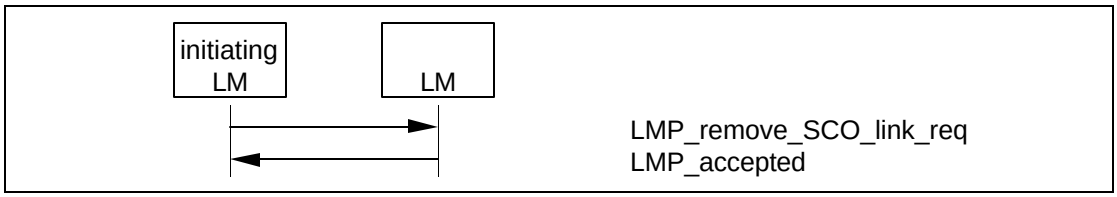
3.21.4 Slave requests change of SCO parameters

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 must 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. The sequence is the same as in Section 3.21.2 on page 227.

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 must shall respond with LMP_accepted.



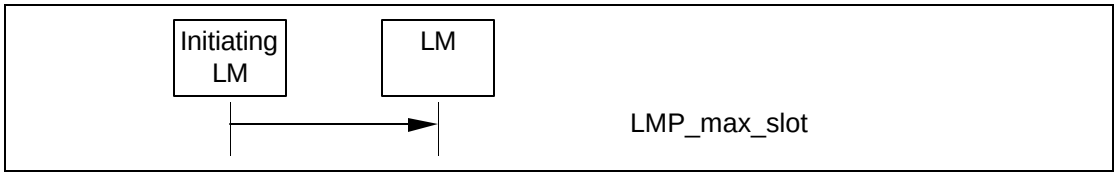
Sequence 52: SCO link removed.

3.22 CONTROL OF MULTI-SLOT PACKETS

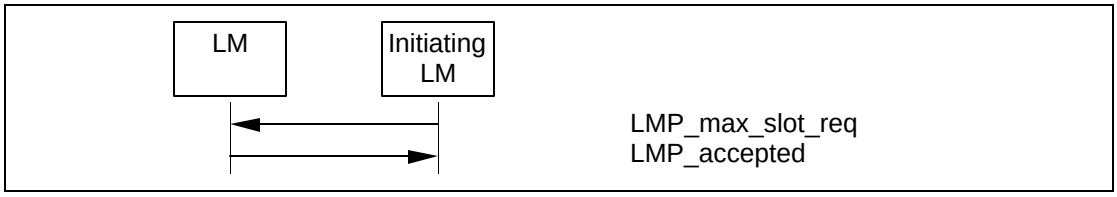
The number of slots used by a device can be limited. A device allows the remote device to use a maximal number of slots by sending the PDU LMP_max_slot providing max slots as parameter. Each device can request to use a maximal number of slots by sending the PDU LMP_max_slot_req providing max slots as 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 multi-slot packets.These PDUs can be sent at anytime after connection setup is completed.

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

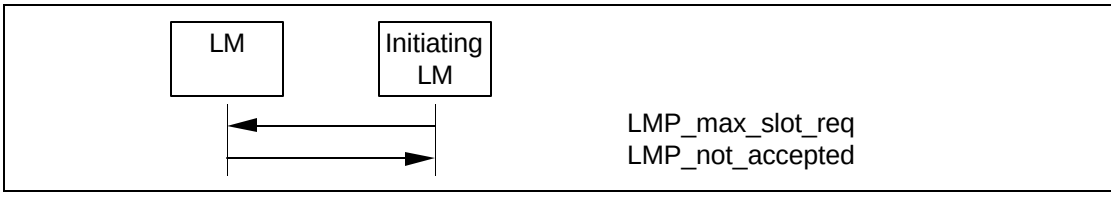
Table 3.21: PDUs used to control the use of multi-slot packets.



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.

3.23 PAGING SCHEME

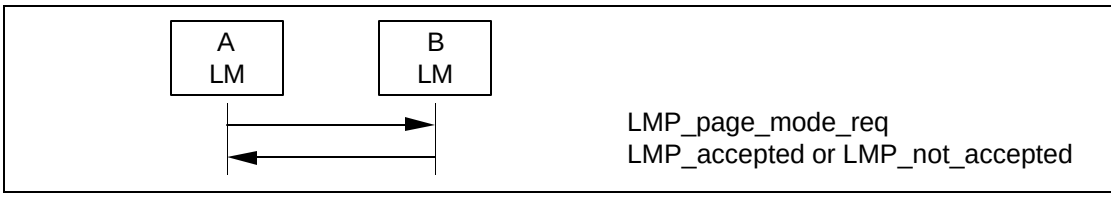
In addition to the mandatory paging scheme, Bluetooth defines optional paging schemes; see "Appendix VII" on page 1025. LMP provides a means to negotiate the paging scheme, which is to be used the next time a unit is paged.

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

Table 3.22: PDUs used to request paging scheme.

3.23.1 Page mode

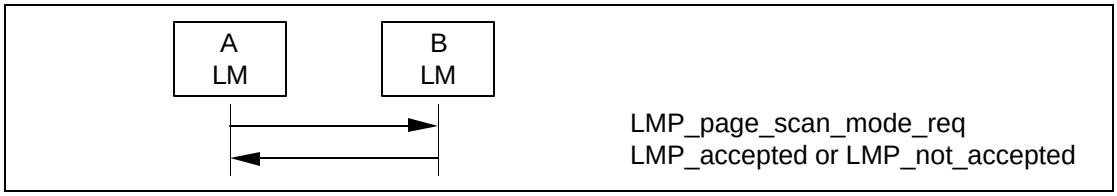
This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting is not changed. A request to switch back to the mandatory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



Sequence 56: Negotiation for page mode.

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 must shall be accepted.



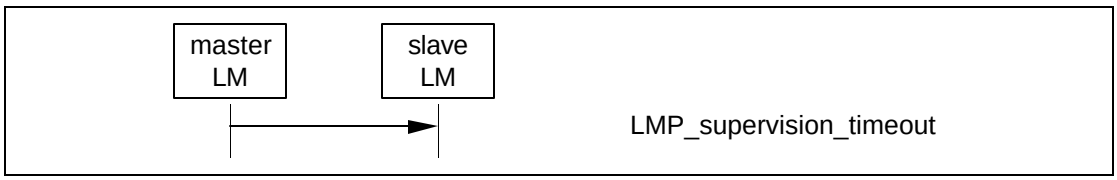
Sequence 57: Negotiation for page scan mode

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 Baseband Specification Section 10.11, on page 124. An LMP procedure is used to set the value of the supervision timeout.

M/O	PDU	Contents
M	LMP_supervision_timeout	supervision timeout

Table 3.23: PDU used to set the supervision timeout.



Sequence 58: Setting the link supervision timeout.

4 CONNECTION ESTABLISHMENT

After the paging procedure, the master must shall poll the slave with a max poll interval as defined in Table 5.5 on page 245. LMP procedures with for clock offset request, LMP version, supported features, name request and detach can then be carried out.

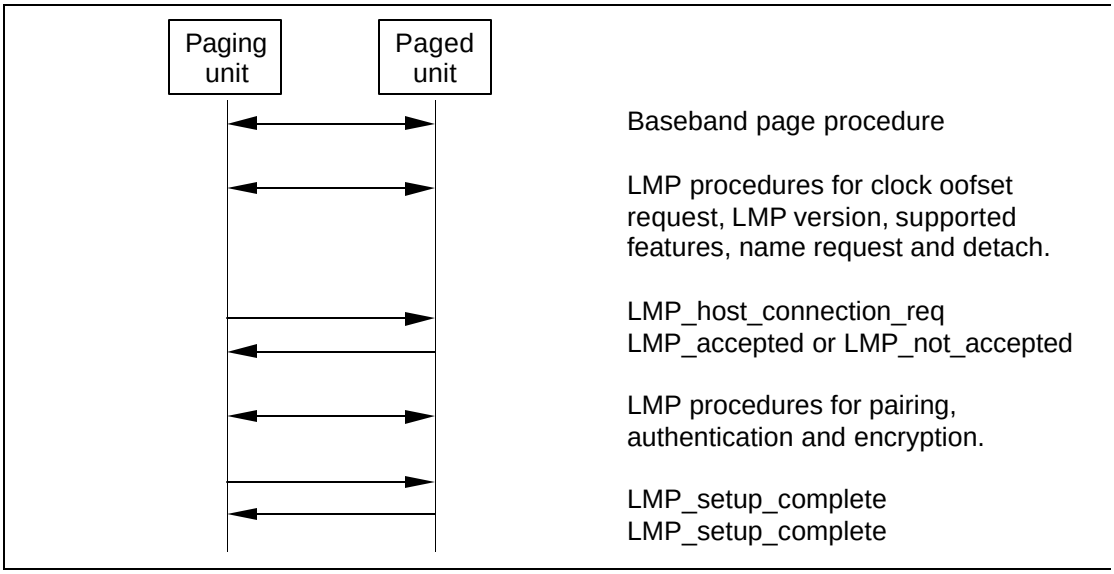


Figure 4.1: Connection establishment.

When the paging device wishes to create a connection involving layers above LM, it sends LMP_host_connection_req. 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 section 3.12 on page 208, 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. When both devices have sent LMP_setup_complete the first packet on a logical channel different from LMP can then be transmitted.

M/O	PDU	Contents
M	LMP_host_connection_req	-
M	LMP_setup_complete	-

Table 4.1: PDUs used for connection establishment.

Note: the transaction ID shall be 0 if LMP_setup_complete is sent from the master and 1 if it is sent from the slave.

5 SUMMARY OF 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	-	

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_max_slot	2	45	DM1/ DV	m ? s	max slots	2
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
					$?_B$	8
					D_{access}	9
					T_{access}	10
					$N_{acc-slots}$	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

Table 5.1: Coding of the different LM PDUs.

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
					?$_B$	8
					PM_ADDR	9
					AR_ADDR	10
					N_{Bsleep}	11
					D_{Bsleep}	12
					D_{access}	13
					T_{access}	14
					$N_{acc-slots}$	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

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
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
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_timeou t	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

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
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
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

Table 5.1: Coding of the different LM PDUs.

Note1: 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 bit0 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.

Note2: 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.

Note3: For LMP_unpark_PM_ADDR the AM_ADDR and the PM_ADDR of the 2^{nd} – 7^{th} 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* must 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 Note1).

5.1 DESCRIPTION OF PARAMETERS

Name	Length (bytes)	Type	Unit	Detailed
access scheme	1	u_int4		0: polling technique 1-15: Reserved
air mode	1	u_int8		0: ?-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 Bluetooth Assigned Numbers (http://www.bluetooth.org/assigned-numbers.htm)
D_{access}	1	u_int8	slots	
D_B	2	u_int16	slots	
D_{Bsleep}	1	u_int8	slots	

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed
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
features	8	multiple bytes		See Table 5.3 on page 243
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	?s	
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	
$N_{acc-slots}$	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		

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed
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 5.4 on page 244.
SCO handle	1	u_int8		
SCO packet	1	u_int8		0: HV1 1: HV2 2: HV3 3-255: Reserved
slot offset	2	u_int16	?s	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 time-out	2	u_int16	slots	0 means an infinite time-out

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed
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 Bluetooth Assigned Numbers, (http://www.bluetooth.org/assigned-numbers.htm)
$?_B$	1	u_int8	slots	

Table 5.2: Parameters in LM PDUs.

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 Byte. See further in Baseband Specification Section 4.5.2, on page 62. The feature parameter bits that are not defined in Table 5.3 shall be zero.

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)

Table 5.3: Coding of the parameter features.

5.1.2 List of error reasons

The following table contains the codes of the different error reasons used in LMP.

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

Table 5.4: List of error reasons.

Reason	Description
0x26	Unit key used
0x27	QoS not supported
0x28	Instant passed
0x29	Pairing with unit key not supported

Table 5.4: List of error reasons.

5.2 DEFAULT VALUES

The Bluetooth device must shall use these values before anything else has been negotiated:

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

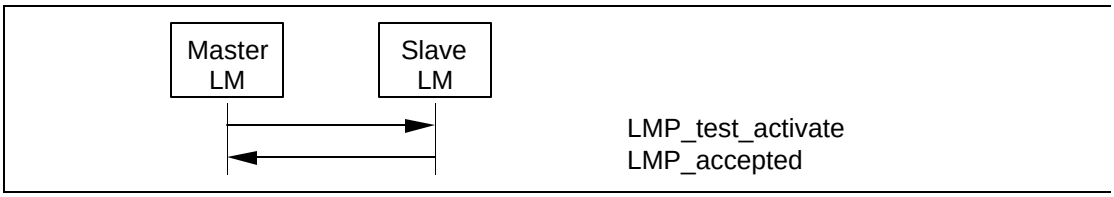
Table 5.5: Default values.

6 TEST MODES

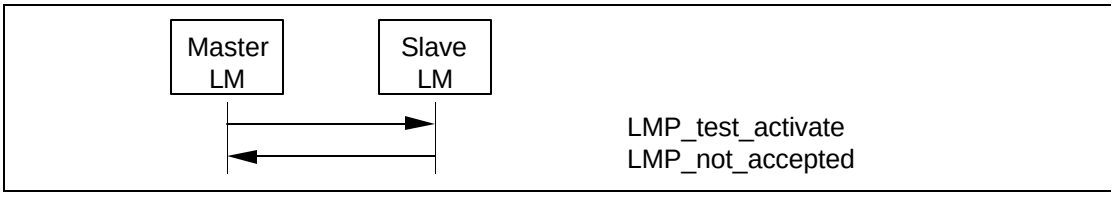
LMP has PDUs to support different Bluetooth test modes, which are used for certification and compliance testing of the Bluetooth radio and baseband. See "Bluetooth Test Mode" on page827 for a detailed description of these test modes.

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 must 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.

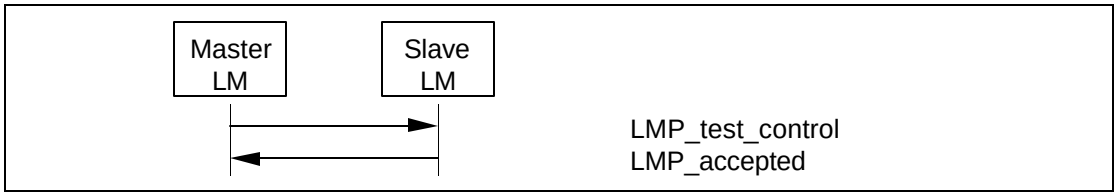


Sequence 60: Activation of test mode fails. Slave is not allowed to enter test mode.

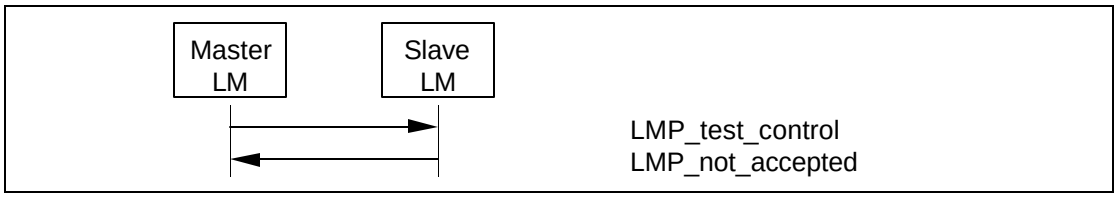
The test mode can be deactivated in two ways. Sending LMP_test_control with the test scenario set to "exit test mode" exits the test mode and the slave returns to normal operation still connected to the master. Sending LMP_detach to the DUT ends the test mode and the connection.

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.

6.3 SUMMARY OF TEST MODE PDUs

The PDUs used for test purposes are summarized in the following table. For a detailed description of the parameters, see Bluetooth Test Mode Table 3.2 on page 842.

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

Table 6.1: Test mode PDUs.

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 Section 1 on page 189, is exceeded or if a link loss is detected (see Baseband Specification Section 10.11, on page 124), 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 doesn’t expect a reply, for example LMP_sres or LMP_temp_key, the PDU will be ignored.

8 LIST OF FIGURES

9 LIST OF TABLES