Specification Volume 6

SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Experience More

Core System Package [Low Energy Controller volume]

Covered Core Package version: 4.0

Bluetooth®

Revision History

The Revision History is shown in the [Vol 0] Part C, Section on page 55.

Contributors

The persons who contributed to this specification are listed in the [Vol 0] Part C, Section on page 55.

Web Site

This specification can also be found on the official Bluetooth web site: http://www.bluetooth.com

Disclaimer and Copyright Notice

TABLE OF CONTENTS

Core System Package [Low Energy Controller volume]

Part A

PHYSICAL LAYER SPECIFICATION

This part of the specification describes the Bluetooth low energy physical layer.

CONTENTS

1 SCOPE

Bluetooth Low Energy (LE) devices operate in the unlicensed 2.4 GHz ISM (Industrial Scientific Medical) band. A frequency hopping transceiver is used to combat interference and fading.

An LE radio shall have a transmitter or a receiver, or both.

The LE radio shall fulfill the stated requirements for the operating conditions declared by the equipment manufacturer (see Section 5.1 and Section 5.2).

The radio parameters shall be measured according to the methods described in the LE RF PHY Test Specification.

This specification is based on the established regulations for Europe, Japan, North America, Taiwan, South Korea and China. The standard documents listed below are only for information, and are subject to change or revision at any time.

The Bluetooth SIG maintains an online database of regulations that apply to Bluetooth technology in the 2.4 GHz ISM band, posted at https://www.bluetooth.org/regulatory/newindex.cfm.

Europe:

Approval Standards: European Telecommunications Standards Institute, ETSI
Documents: EN 300 328, EN 301 489, ETS 300-826

Japan:

Approval Standards: Japanese Radio Law, JRL
Documents: Japanese Radio Law: Article 4.3, Article 28, Article 29, Article 38

Radio Equipment Regulations: Article 5, Article 6, Article 7, Article 14, Article 24, Article 9.4, Article 49.20.1.C.2, Article 49.20.1.E.3
Radio Law Enforcement Regulations: Article 6.2, Article 6.4.4.1, Article 7

North America:

Approval Standards: Federal Communications Commission, FCC, USA
Documents: CFR47, Part 15: Sections 15.205, 15.209 and 15.247

Approval Standards: Industry Canada, IC, Canada
Documents: RSS-210 and RSS139

Taiwan:

Approval Standards: National Communications Commission, NCC
Documents: Low Power 0002 (LP0002); Low-power Radio-frequency Devices Technical Regulations

South Korea:

Approval Standards: Korea Communications Commission, KCC
Documents: Rules on Radio Equipment 2008-116

China:

Approval Standards: Ministry of Industry and Information Technology, MIIT
Documents: MIIT regulation [2002]353

2 FREQUENCY BANDS AND CHANNEL ARRANGEMENT

The LE system operates in the 2.4 GHz ISM band at 2400-2483.5 MHz. The LE system uses 40 RF channels. These RF channels have center frequencies 2402 + k * 2 MHz, where k = 0, ..., 39.

Regulatory Range	RF Channels
2.400-2.4835 GHz	f=2402+k*2 MHz, k=0, … ,39

Table 2.1: Operating frequency bands

3 TRANSMITTER CHARACTERISTICS

The requirements stated in this section are given as power levels at the antenna connector of the LE device. If the device does not have a connector, a reference antenna with 0 dBi gain is assumed.

Due to the difficulty in making accurate radiated power measurements, systems with an integral antenna should provide a temporary antenna connector during LE RF qualification testing.

For a transmitter, the output power level at the maximum power setting shall be within the limits defined in Table 3.1.

Minimum Output Power	Maximum Output Power
0.01 mW (-20 dBm)	10 mW (+10 dBm)

Table 3.1: Transmission power

The output power control of a device may be changed locally, for example to optimize the power consumption or reduce interference to other equipment.

3.1 MODULATION CHARACTERISTICS

The modulation is Gaussian Frequency Shift Keying (GFSK) with a bandwidth-bit period product BT=0.5. The modulation index shall be between 0.45 and 0.55. A binary one shall be represented by a positive frequency deviation, and a binary zero shall be represented by a negative frequency deviation.

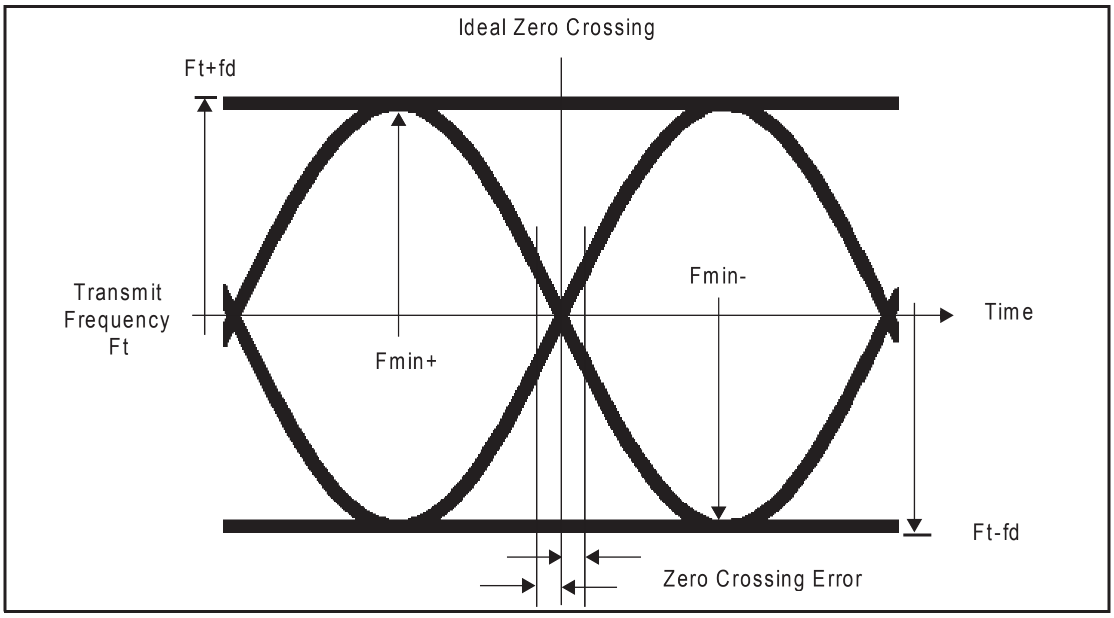

Figure 3.1: GFSK parameters definition

For each transmission the minimum frequency deviation,

$$F_{min} = \min \{ |F_{min+}| , F_{min-} \}$$

which corresponds to a 1010 sequence, shall be no smaller than ±80% of the frequency deviation with respect to the transmit frequency, which corresponds to a 00001111 sequence.

In addition, the minimum frequency deviation shall never be less than 185 kHz. The data transmitted has a symbol rate of 1 mega-symbols per second. The symbol timing accuracy shall be better than ±50 ppm.

The zero crossing error is the time difference between the ideal symbol period and the measured crossing time. This shall be less than ±1/8 of a symbol period.

3.2 SPURIOUS EMISSIONS

3.2.1 Modulation Spectrum

For products intended to comply with FCC part 15.247 rules, the minimum 6 dB bandwidth of the transmitter spectrum shall be at least 500 kHz using a resolution bandwidth of 100 kHz.

3.2.2 In-band Spurious Emission

An adjacent channel power is specified for channels at least 2 MHz from the carrier. This adjacent channel power is defined as the sum of the measured power in a 1 MHz bandwidth.

The spectrum measurement shall be performed with a 100 kHz resolution bandwidth and an average detector. The device shall transmit on an RF Channel with the center frequency M and the adjacent channel power shall be measured on a 1 MHz RF frequency N. The transmitter shall transmit a pseudo random data pattern in the payload throughout the test.

Frequency offset	Spurious Power
2 MHz (|M-N| = 2)	-20 dBm
3 MHz or greater (|M-N| ≥ 3)	-30 dBm

Table 3.2: Transmit Spectrum mask

Exceptions are allowed in up to three bands of 1 MHz width, centered on a frequency which is an integer multiple of 1 MHz. These exceptions shall have an absolute value of -20 dBm or less.

3.2.3 Out-of-band Spurious Emission

The equipment manufacturer is responsible for the ISM out-of-band spurious emissions requirements in the intended countries of sale.

3.3 RADIO FREQUENCY TOLERANCE

The deviation of the center frequency during the packet shall not exceed ±150 kHz, including both the initial frequency offset and drift. The frequency drift during any packet shall be less than 50 kHz. The drift rate shall be less than 400 Hz/μs.

The limits on the transmitter center frequency drift within a packet is shown in Table 3.3.

Parameter	**Frequency Drift**
Maximum drift	±50 kHz
Maximum drift rate[1]	400 Hz/μs

Table 3.3: Maximum allowable frequency drifts in a packet

1. The maximum drift rate is allowed anywhere in a packet.

4 RECEIVER CHARACTERISTICS

The reference sensitivity level referred to in this chapter is -70 dBm. The packet error rate corresponding to the defined bit error ratio (BER) shall be used in all receiver characteristic measurements.

4.1 ACTUAL SENSITIVITY LEVEL

The actual sensitivity level is defined as the receiver input level for which a BER of 0.1% is achieved.

The actual sensitivity level of the receiver shall be less than or equal to -70 dBm with any transmitter compliant to the transmitter specification specified in Section 3 together with any combination of the following allowed parameter variations:

- Initial frequency offset
- Frequency drift
- Symbol rate
- Frequency deviation

4.2 INTERFERENCE PERFORMANCE

The interference performance shall be measured with a wanted signal 3 dB over the reference sensitivity level. If the frequency of an interfering signal is outside of the band 2400-2483.5 MHz, the out-of-band blocking specification (see Section 4.3) shall apply. The measurement resolution shall be 1 MHz. Both the desired and the interfering signal shall be reference signals as specified in Section 4.6. The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 4.1:

Frequency of Interference	Ratio
Co-Channel interference, $C/I_{co\text{-}channel}$	21 dB
Adjacent (1 MHz) interference[1], $C/I_{1\ MHz}$	15 dB
Adjacent (2 MHz) interference[1], $C/I_{2\ MHz}$	-17 dB
Adjacent (≥3 MHz) interference[1], $C/I_{\geq 3\ MHz}$	-27 dB
Image frequency Interference[1][2][3], C/I_{Image}	-9 dB
Adjacent (1 MHz) interference to in-band image frequency[1], $C/I_{Image\pm 1MHz}$	-15 dB

Table 4.1: Interference performance

1. If two adjacent frequency specifications from Table 4.1 are applicable to the same frequency, the more relaxed specification applies.
2. In-band image frequency
3. If the image frequency ≠ n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency for integer n.

Any frequencies where the requirements are not met are called spurious response RF channels. Five spurious response RF channels are allowed with a distance of ≥2 MHz from the wanted signal excluding the image frequency and the image frequency ±1MHz. On these spurious response RF channels, a relaxed interference requirement C/I = -17 dB shall be met.

4.3 OUT-OF-BAND BLOCKING

The out-of-band blocking applies to interfering signals outside the band 2400-2483.5 MHz. The out-of-band suppression (or rejection) shall be measured with a wanted signal 3 dB over the reference sensitivity level. The interfering signal shall be a continuous wave signal. The desired signal shall be a reference signal as specified in Section 4.6, with a center frequency of 2440 MHz. The BER shall be ≤ 0.1%. The out-of-band blocking shall fulfill the following requirements:

Interfering Signal Frequency	Interfering Signal Power Level	Measurement resolution
30 MHz – 2000 MHz	-30 dBm	10 MHz
2003 – 2399 MHz	-35 dBm	3 MHz
2484 – 2997 MHz	-35 dBm	3 MHz
3000 MHz – 12.75 GHz	-30 dBm	25 MHz

Table 4.2: Out-of-band suppression (or rejection) requirements.

Up to 10 exceptions are permitted, which are dependent upon the given RF channel and are centered at a frequency which is an integer multiple of 1 MHz:

- For at least 7 of these spurious response frequencies, a reduced interference level of at least -50 dBm is allowed in order to achieve the required BER ≤ 0.1%.
- For a maximum of 3 of the spurious response frequencies, the interference level may be lower.

4.4 INTERMODULATION CHARACTERISTICS

The actual sensitivity performance, BER ≤ 0.1%, shall be met under the following conditions:

- The wanted signal shall be at a frequency f_0 with a power level 6 dB over the reference sensitivity level. The wanted signal shall be a reference signal as specified in Section 4.6.
- A static sine wave signal shall be at a frequency f_1 with a power level of -50 dBm.
- An interfering signal shall be at a frequency f_2 with a power level of -50 dBm. The interfering signal shall be a reference signal as specified in Section 4.6.

Frequencies f_0, f_1 and f_2 shall be chosen such that $f_0 = 2*f_1 - f_2$ and $| f_2-f_1 | = n*1$, where n can be 3, 4, or 5. The system shall fulfill at least one of the three alternatives (n=3, 4, or 5).

4.5 MAXIMUM USABLE LEVEL

The maximum usable input level the receiver can operate at shall be greater than -10 dBm, and the BER shall be less than or equal to 0.1% at -10 dBm input power. The input signal shall be a reference signal as specified in Section 4.6.

4.6 REFERENCE SIGNAL DEFINITION

A reference signal is defined as:

Modulation = GFSK

Modulation index = 0.5 ± 1%

BT = 0.5 ± 1%

Bit Rate = 1 Mbps ±1 ppm

Modulating Data for wanted signal = PRBS9

Modulating Data for interfering signal = PRBS15

Frequency accuracy better than ±1 ppm

5 APPENDIX A

5.1 NORMAL OPERATING CONDITIONS (NOC)

5.1.1 Normal Temperature and Air Humidity

The normal operating temperature shall be declared by the product manufacturer. The nominal test temperature shall be within ±10°C of the normal operating temperature. The nominal operating temperature ±10°C shall not exceed the extreme limits stated in Section 5.2.1.

Operating air humidity range shall be declared by the product manufacturer. The air humidity level for the nominal test condition tests shall be within the declared NOC range.

5.1.2 Nominal Supply Voltage

The nominal test voltage for the equipment under normal test conditions shall be the nominal supply voltage as declared by the product manufacturer.

5.2 EXTREME OPERATING CONDITIONS (EOC)

5.2.1 Extreme Temperature And Air Humidity

The extreme temperature limits are defined as the minimum and maximum temperatures of the operating temperature range declared by the product manufacturer.

For the extreme test condition, the air humidity shall be at a level within the operating air humidity range declared by the product manufacturer (see Section 5.1.1).

5.2.2 Extreme Supply Voltage

The extreme supply voltages are dependent on the characteristics of the product's power supply.

If the product is designed to be operated as a part of another system or a portion of a product, the extreme voltage limits for this product or system shall be used.

The applicable upper and lower extreme supply voltages shall be declared by the product manufacturer.

6 APPENDIX B - OPERATING CONDITIONS

The LE radio parameters shall be compliant in the following conditions.

Parameter	Temperature	Power supply
Output power	EOC	EOC
In-band emissions	EOC	EOC
Modulation characteristics	NOC	NOC
Carrier frequency offset and drift	EOC	EOC
Receiver sensitivity	EOC	EOC
C/I and selectivity performance	NOC	NOC
Blocking performance	NOC	NOC
Intermodulation performance	NOC	NOC
Maximum input signal level	NOC	NOC

EOC = Extreme Operating Conditions
NOC = Normal Operating Conditions

Note: Validation of the LE receiver parameters is performed using the Direct Test Mode (see [Vol. 6], Part F).

LINK LAYER SPECIFICATION

This part of the specification describes the Bluetooth low energy Link Layer.

CONTENTS

1 GENERAL DESCRIPTION

1.1 LINK LAYER STATES

The operation of the Link Layer can be described in terms of a state machine with the following five states:

- Standby State
- Advertising State
- Scanning State
- Initiating State
- Connection State

The Link Layer state machine allows only one state to be active at a time. The Link Layer shall have at least one Link Layer state machine that supports one of Advertising State or Scanning State. The Link Layer may have multiple instances of the Link Layer state machine. Certain combinations of states and roles within multiple state machines in the Link Layer are prohibited (see Section 1.1.1).

The Link Layer in the Standby State does not transmit or receive any packets. The Standby State can be entered from any other state.

The Link Layer in the Advertising State will be transmitting advertising channel packets and possibly listening to and responding to responses triggered by these advertising channel packets. A device in the Advertising State is known as an advertiser. The Advertising State can be entered from the Standby State.

The Link Layer in the Scanning State will be listening for advertising channel packets from devices that are advertising. A device in the Scanning State is known as a scanner. The Scanning State can be entered from the Standby State.

The Link Layer in the Initiating State will be listening for advertising channel packets from a specific device(s) and responding to these packets to initiate a connection with another device. A device in the Initiating State is known as an initiator. The Initiating State can be entered from the Standby State.

The Connection State can be entered either from the Initiating State or the Advertising State. A device in the Connection State is known as being in a connection.

Within the Connection State, two roles are defined:

- Master Role
- Slave Role

When entered from the Initiating State, the Connection State shall be in the Master Role. When entered from the Advertising State, the Connection State shall be in the Slave Role.

The Link Layer in the Master Role will communicate with a device in the Slave Role and defines the timings of transmissions.

The Link Layer in the Slave Role will communicate with a single device in the Master Role.

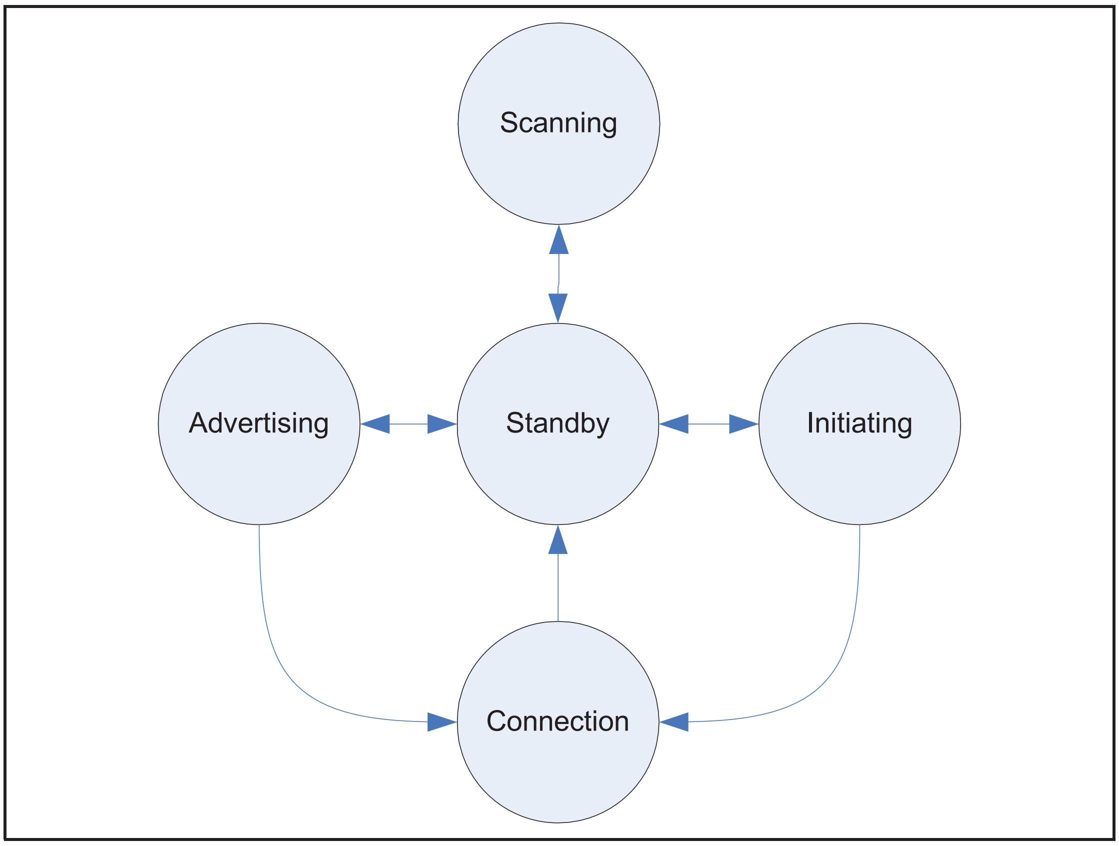

Figure 1.1: State diagram of the Link Layer state machine

1.1.1 State and Role Combination Restrictions

The Link Layer may optionally support multiple state machines. If it does support multiple state machines, the following state and role restrictions shall apply.

- The Link Layer in the Connection State shall not operate in the Master Role and Slave Role at the same time.
- The Link Layer in the Connection State operating in the Slave Role shall have only one connection.
- The Link Layer in the Connection State operating in the Master Role may have multiple connections.

- The Link Layer shall not operate in the Initiating State if the Link Layer is already operating in the Connection State in the Slave Role.
- If the Link Layer is already operating in the Connection State or Initiating State, the Link Layer shall not operate in the Advertising State with a type of advertising that could result in the Link Layer entering the Connection State in the Slave Role.

All other combinations of states may be supported. A Link Layer implementation is not required to support all the possible state combinations that are allowed by this specification.

The table below specifies the allowable and prohibited Link Layer state machine state combinations.

Multiple State Machine State and Roles		**Advertising**	**Scanning**	**Initiating**	**Connection**	
					Master Role	**Slave Role**
Advertising		Prohibited	Allowed	Allowed*	Allowed*	Allowed*
Scanning		Allowed	Prohibited	Allowed	Allowed	Allowed
Initiating		Allowed*	Allowed	Prohibited	Allowed	Prohibited
Connection	**Master Role**	Allowed*	Allowed	Allowed	Allowed	Prohibited
	Slave Role	Allowed*	Allowed	Prohibited	Prohibited	Prohibited

Table 1.1: Allowable and prohibited Link Layer state machine state combinations

* Only advertising packets that shall not result in the Link Layer entering Connection State in the Slave Role allowed.

1.2 BIT ORDERING

The bit ordering when defining fields within the packet or Protocol Data Unit (PDU) in the Link Layer specification follows the Little Endian format. The following rules apply:

- The Least Significant Bit (LSB) corresponds to b_0
- The LSB is the first bit sent over the air
- In illustrations, the LSB is shown on the left side

Furthermore, data fields defined in the Link Layer, such as the PDU header fields, shall be transmitted with the LSB first. For instance, a 3-bit parameter *X=3* is sent as:

$b_0 b_1 b_2 = 110$

Over the air, 1 is sent first, 1 is sent next, and 0 is sent last. This is shown as 110 in the specification.

Binary field values specified in this specification that follow the format 10101010b (e.g., pre-amble in Section 2.1.1 or advertising channel Access Address in Section 2.1.2) are written with the MSB to the left.

Multi-octet fields, with the exception of the Cyclic Redundancy Check (CRC) and the Message Integrity Check (MIC), shall be transmitted with the least significant octet first. Each octet within multi-octet fields, with the exception of the CRC (see Section 3.1.1), shall be transmitted in LSB first order. For example, the 48-bit addresses in the advertising channel PDUs shall be transmitted with the least significant octet first, followed by the remainder of the five octets in increasing order.

Multi-octet field values specified in this specification (e.g. the CRC initial value in Section 2.3.3.1) are written with the most significant octet to the left; for example in 0x112233445566, the octet 0x11 is the most significant octet.

1.3 DEVICE ADDRESS

Devices are identified using a device address. Device addresses may be either a public device address or a random device address. A public device address and a random device address are both 48 bits in length.

A device shall contain at least one type of device address and may contain both.

The public device address shall be created in accordance with section 9.2 ("48-bit universal LAN MAC addresses") of the IEEE 802-2001 standard (http://standards.ieee.org/getieee802/download/802-2001.pdf) and using a valid Organizationally Unique Identifier (OUI) obtained from the IEEE Registration Authority (see http://standards.ieee.org/regauth/oui/forms/ and sections 9 and 9.1 of the IEEE 802-2001 specification).

The public device address is divided into the following two fields:

- company_assigned field is contained in the 24 least significant bits
- company_id field is contained in the 24 most significant bits

LSB	MSB
company_assigned (24 bits)	company_id (24 bits)

Figure 1.2: Format of public device address

The random device address is divided into the following two fields:

- hash field is contained in the 24 least significant bits, as defined in [Vol. 3] Part C, Section 10.8.2.3.
- random field is contained in the 24 most significant bits, as defined in [Vol. 3] Part C, Section 10.8.2.2.

LSB	MSB
hash (24 bits)	random (24 bits)

Figure 1.3: Format of random device address

1.4 PHYSICAL CHANNEL

As specified in Part A, Section 2, 40 RF Channels are defined in the 2.4GHz ISM band. These RF Channels are allocated into two LE physical channels: advertising and data. The advertising physical channel uses three RF channels for discovering devices, initiating a connection and broadcasting data. The data physical channel uses up to 37 (see Section 4.5.8) RF channels for communication between connected devices. Each of these RF Channels is allocated a unique channel index (see Section 1.4.1).

Two devices that wish to communicate use a shared physical channel. To achieve this, their transceivers must be tuned to the same RF Channel at the same time.

Given that the number of RF Channels is limited, and that many Bluetooth devices may be operating independently within the same spatial and temporal area, there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF Channel, resulting in a physical channel collision. To mitigate the unwanted effects of this collision, each transmission on a physical channel starts with an Access Address that is used as a correlation code by devices tuned to the physical channel. This Access Address is a property of the physical channel. The Access Address is present at the start of every transmitted packet.

The Link Layer uses one physical channel at a given time.

Whenever the Link Layer is synchronized to the timing, frequency and Access Address of a physical channel it is said to be 'connected' to this channel (whether or not it is actively involved in communications over the channel).

1.4.1 Advertising and Data Channel Indexes

Table 1.2 shows the mapping from RF Channel to Data Channel Index and Advertising Channel Index. It also shows the allocation of channel type to each RF Channel.

RF Channel	RF Center Frequency	Channel Type	Data Channel Index	Advertising Channel Index
0	2402 MHz	Advertising channel		37
1	2404 MHz	Data channel	0	
2	2406 MHz	Data channel	1	
...	…	Data channels	…	
11	2424 MHz	Data channel	10	
12	2426 MHz	Advertising channel		38
13	2428 MHz	Data channel	11	
14	2430 MHz	Data channel	12	
...	…	Data channels	…	
38	2478 MHz	Data channel	36	
39	2480 MHz	Advertising channel		39

Table 1.2: Mapping of RF Channel to Data Channel Index and Advertising Channel Index

2 AIR INTERFACE PACKETS

LE devices shall use the packets as defined in the following sections.

2.1 PACKET FORMAT

The Link Layer has only one packet format used for both advertising channel packets and data channel packets.

The packet format is shown in Figure 2.1. Each packet consists of four fields: the preamble, the Access Address, the PDU, and the CRC.

LSB MSB

Preamble	Access Address	PDU	CRC
(1 octet)	(4 octets)	(2 to 39 octets)	(3 octets)

Figure 2.1: Link Layer packet format

The preamble is 1 octet and the Access Address is 4 octets. The PDU range is from 2 to a maximum of 39 octets. The CRC is 3 octets.

The Preamble is transmitted first, followed by the Access Address, followed by the PDU followed by the CRC.

The shortest packet is 80 bits in length. The longest packet is 376 bits in length.

2.1.1 Preamble

All Link Layer packets have an eight bit preamble. The preamble is used in the receiver to perform frequency synchronization, symbol timing estimation, and Automatic Gain Control (AGC) training.

Advertising channel packets shall have 10101010b as the preamble.

The data channel packet preamble is either 10101010b or 01010101b, depending on the LSB of the Access Address. If the LSB of the Access Address is 1, the preamble shall be 01010101b, otherwise the preamble shall be 10101010b.

2.1.2 Access Address

The Access Address for all advertising channel packets shall be 10001110100010011011111011010110b (0x8E89BED6).

The Access Address in data channel packets shall be different for each Link Layer connection between any two devices with certain restrictions as defined below. The Access Address shall be a random 32-bit value, generated by the device in the Initiating State and sent in a connection request as defined in Section 2.3.3.1. The initiator shall ensure that the Access Address meets the following requirements:

- It shall have no more than six consecutive zeros or ones.
- It shall not be the advertising channel packets' Access Address.
- It shall not be a sequence that differs from the advertising channel packets' Access Address by only one bit.
- It shall not have all four octets equal.
- It shall have no more than 24 transitions.
- It shall have a minimum of two transitions in the most significant six bits.

2.1.3 PDU

The preamble and Access Address are followed by a PDU.

When a packet is transmitted in an advertising physical channel, the PDU shall be the Advertising Channel PDU as defined in Section 2.3. When a packet is transmitted in a data physical channel, the PDU shall be the Data Channel PDU as defined in Section 2.4.

2.1.4 CRC

At the end of every Link Layer packet there is a 24-bit CRC. It shall be calculated over the PDU. The CRC polynomial is defined in Section 3.1.1.

2.2 RESERVED FOR FUTURE USE (RFU)

Any field marked as RFU is reserved for future use. It shall be set to zero on transmission and ignored upon receipt.

2.3 ADVERTISING CHANNEL PDU

The advertising channel PDU has a 16-bit header and a variable size payload. Its format is as shown in Figure 2.2. The 16 bit Header field of the advertising channel PDU is as shown in Figure 2.3.

LSB	MSB
Header (16 bits)	Payload (as per the Length field in the Header)

Figure 2.2: Advertising channel PDU

LSB					MSB
PDU Type (4 bits)	RFU (2 bits)	TxAdd (1 bit)	RxAdd (1 bit)	Length (6 bits)	RFU (2 bits)

Figure 2.3: Advertising channel PDU Header

The PDU Type field of the advertising channel PDU that is contained in the header indicates the PDU type as defined in Table 2.1.

The TxAdd and RxAdd fields of the advertising channel PDU that are contained in the header contain information specific to the PDU type defined for each advertising channel PDU separately. If the TxAdd or RxAdd fields are not defined as used in a given PDU then they shall be considered Reserved for Future Use.

The Length field of the advertising channel PDU header indicates the payload field length in octets. The valid range of the Length field shall be 6 to 37 octets.

The Payload fields in the advertising channel PDUs are specific to the PDU Type and are defined in Section 2.3.1 through Section 2.3.3. The PDU Types marked as Reserved shall not be sent and shall be ignored upon receipt.

Within advertising channel PDUs, Advertising Data from the Host may be included in the Payload in some PDU Types. The format of this data is defined in [Vol. 3] Part C, Section 17.

PDU Type $b_3b_2b_1b_0$	**Packet Name**
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

Table 2.1: Advertising channel PDU Header's PDU Type field encoding

2.3.1 Advertising PDUs

The following advertising channel PDU Types are called advertising PDUs and are used in the specified events:

- ADV_IND: connectable undirected advertising event
- ADV_DIRECT_IND: connectable directed advertising event
- ADV_NONCONN_IND: non-connectable undirected advertising event
- ADV_SCAN_IND: scannable undirected advertising event

These PDUs are sent by the Link Layer in the Advertising State and received by a Link Layer in the Scanning State or Initiating State.

2.3.1.1 ADV_IND

The ADV_IND PDU has the Payload as shown in Figure 2.4. The PDU shall be used in connectable undirected advertising events. The TxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1).

Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.4: ADV_IND PDU Payload

The Payload field consists of AdvA and AdvData fields. The AdvA field shall contain the advertiser's public or random device address as indicated by

TxAdd. The AdvData field may contain Advertising Data from the advertiser's Host.

2.3.1.2 ADV_DIRECT_IND

The ADV_DIRECT_IND PDU has the Payload as shown in Figure 2.5. The PDU shall be used in connectable directed advertising events. The TxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the Flags field indicates whether the initiator's address in the InitA field is public (RxAdd = 0) or random (RxAdd = 1).

Payload	
AdvA (6 octets)	InitA (6 octets)

Figure 2.5: ADV_DIRECT_IND PDU Payload

The Payload field consists of AdvA and InitA fields. The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The InitA field is the address of the device to which this PDU is addressed. The InitA field shall contain the initiator's public or random device address as indicated by RxAdd.

Note: This packet does not contain any Host data.

2.3.1.3 ADV_NONCONN_IND

The ADV_NONCONN_IND PDU has the Payload as shown in Figure 2.6. The PDU shall be used in non-connectable undirected advertising events. The TxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1).

Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.6: ADV_NONCONN_IND PDU Payload

The Payload field consists of AdvA and AdvData fields. The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The AdvData field may contain Advertising Data from the advertiser's Host.

2.3.1.4 ADV_SCAN_IND[1]

The ADV_SCAN_IND PDU has the Payload as shown in Figure 2.7. The PDU shall be used in scannable undirected advertising events. The TxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1).

Payload	
AdvA (6 octets)	AdvData (0-31 octets)

Figure 2.7: ADV_SCAN_IND PDU Payload

The Payload field consists of AdvA and AdvData fields. The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The AdvData field may contain Advertising Data from the advertiser's Host.

2.3.2 Scanning PDUs

The following advertising channel PDU Types are called scanning PDUs. They are used in the following states:

- SCAN_REQ: sent by the Link Layer in the Scanning State, received by a Link Layer in the Advertising State
- SCAN_RSP: sent by the Link Layer in the Advertising State, received by a Link Layer in the Scanning State

2.3.2.1 SCAN_REQ

The SCAN_REQ PDU has the Payload as shown in Figure 2.8. The TxAdd in the Flags field indicates whether the scanner's address in the ScanA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (RxAdd = 0) or random (RxAdd = 1).

Payload	
ScanA (6 octets)	AdvA (6 octets)

Figure 2.8: SCAN_REQ PDU Payload

The Payload field consists of ScanA and AdvA fields. The ScanA field shall contain the scanner's public or random device address as indicated by TxAdd. The AdvA field is the address of the device to which this PDU is addressed.

1. ADV_DISCOVER-IND was renamed to ADV_SCAN_IND

The AdvA field shall contain the advertiser's public or random device address as indicated by RxAdd.

Note: This packet does not contain any Host Data.

2.3.2.2 SCAN_RSP

The SCAN_RSP PDU has a format as shown in Figure 2.9. The TxAdd in the Flags field indicates whether the advertiser's address in the AdvA field is public (TxAdd = 0) or random (TxAdd = 1). The Length field indicates the size of the payload (AdvA and ScanRspData) in octets.

Payload	
AdvA (6 octets)	ScanRspData (0-31 octets)

Figure 2.9: SCAN_RSP PDU payload

The Payload field consists of AdvA and ScanRspData fields. The AdvA field shall contain the advertiser's public or random device address as indicated by TxAdd. The ScanRspData field may contain any data from the advertiser's Host.

2.3.3 Initiating PDUs

The following advertising channel PDU Type is called the initiating PDU:

- CONNECT_REQ

This PDU is sent by the Link Layer in the Initiating State and received by the Link Layer in the Advertising State.

2.3.3.1 CONNECT_REQ

The CONNECT_REQ PDU has the Payload as shown in Figure 2.10. TxAdd in the Flags field indicates whether the initiator's device address in the InitA field is public (TxAdd = 0) or random (TxAdd = 1). The RxAdd in the Flags field indicates whether the advertiser's device address in the AdvA field is public (RxAdd = 0) or random (RxAdd = 1).

Payload		
InitA (6 octets)	AdvA (6 octets)	LLData (22 octets)

Figure 2.10: CONNECT_REQ PDU payload

The format of the LLData field is shown in Figure 2.11.

LLData									
AA (4 octets)	CRCInit (3 octets)	WinSize (1 octet)	WinOffset (2 octets)	Interval (2 octets)	Latency (2 octets)	Timeout (2 octets)	ChM (5 octets)	Hop (5 bits)	SCA (3 bits)

Figure 2.11: LLData field structure in CONNECT_REQ PDU's payload

The Payload field consists of InitA, AdvA and LLData fields. The InitA field shall contain the Initiator's public or random device address as indicated by TxAdd. The AdvA field shall contain the advertiser's public or random device address as indicated by RxAdd.

The LLData consists of 10 fields:

- The AA field shall contain the Link Layer connection's Access Address determined by the Link Layer following the rules specified in Section 2.1.2.
- The CRCInit field shall contain the initialization value for the CRC calculation for the Link Layer connection, as defined in Section 3.1.1. It shall be a random value, generated by the Link Layer.
- The WinSize field shall be set to indicate the *transmitWindowSize* value, as defined in Section 4.5.3 in the following manner: *transmitWindowSize* = WinSize * 1.25 ms.
- The WinOffset field shall be set to indicate the *transmitWindowOffset*value, as defined in Section 4.5.3 in the following manner:.*transmitWindowOffset* = WinOffset * 1.25 ms.
- The Interval field shall be set to indicate the *connInterval* as defined in Section 4.5.1 in the following manner:*connInterval* = Interval * 1.25 ms.
- The Latency field shall be set to indicate the *connSlaveLatency*value, as defined in Section 4.5.1 in the following manner: *connSlaveLatency* = Latency.
- The Timeout field shall be set to indicate the *connSupervisionTimeout*value, as defined in Section 4.5.2, in the following manner: *connSupervisionTimeout* = Timeout * 10 ms.
- The ChM field shall contain the channel map indicating *Used* and *Unused* data channels. Every channel is represented with a bit positioned as per the data channel index as defined in Section 1.4.1. The LSB represents data channel index 0 and the bit in position 36 represents data channel index 36. A bit value of '0' indicates that the channel is *Unused*. A bit value of '1' indicates that the channel is *Used*. The bits in positions 37, 38 and 39 are Reserved for Future Use. Note: When mapping from RF Channels to data channel index, care should be taken to remember that there is a gap where the second advertising channel is placed.
- The Hop field shall be set to indicate the *hopIncrement* used in the data channel selection algorithm as defined in Section 4.5.8.2. It shall have a random value in the range of 5 to 16.

- The SCA field shall be set to indicate the *masterSCA* used to determine the worst case Master's sleep clock accuracy as defined in Section 4.2.2. The value of the SCA field shall be set as defined in Table 2.2.

SCA	***masterSCA***
0	251 ppm to 500 ppm
1	151 ppm to 250 ppm
2	101 ppm to 150 ppm
3	76 ppm to 100 ppm
4	51 ppm to 75 ppm
5	31 ppm to 50 ppm
6	21 ppm to 30 ppm
7	0 ppm to 20 ppm

Table 2.2: SCA field encoding

2.4 DATA CHANNEL PDU

The Data Channel PDU has a 16 bit header, a variable size payload, and may include a Message Integrity Check (MIC) field.

The Data Channel PDU is as shown in Figure 2.12.

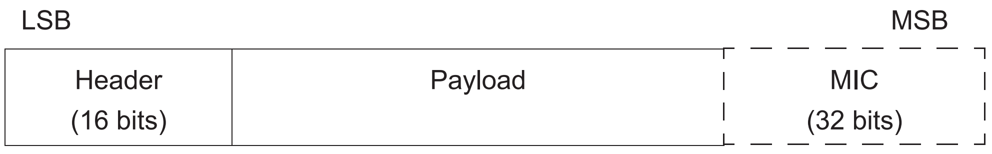

Figure 2.12: Data Channel PDU

The Header field of the Data Channel PDU is as shown in Figure 2.13.

Header						
LLID (2 bits)	NESN (1 bit)	SN (1 bit)	MD (1 bit)	RFU (3 bits)	Length (5 bits)	RFU (3 bits)

Figure 2.13: Data channel PDU header

The 16 bit Header field consists of 5 fields that are specified in Table 2.3.

The MIC field shall not be included in an un-encrypted Link Layer connection, or in an encrypted Link Layer connection with a data channel PDU with a zero length Payload.

The MIC field shall be included in an encrypted Link Layer connection, with a data channel PDU with a non-zero length Payload and shall be calculated as specified in [Vol. 6] Part E, Section 1.

The payload format depends on the LLID field of the Header. If the LLID field is 01b or 10b, the Data Channel PDU Payload field contains an LL Data PDU as defined in Section 2.4.1. If the LLID field is 11b then the Data Channel PDU Payload field contains an LL Control PDU as defined in Section 2.4.2. An LLID field of 00b is reserved.

The NESN bit of the Header is defined in Section 4.5.9.

The SN bit of the Header is defined in Section 4.5.9.

The MD bit of the Header is defined in Section 4.5.6.

The Length field of the Header indicates the length of the Payload and MIC if included. The length field has the range of 0 to 31 octets. The Payload field shall be less than or equal to 27 octets in length. The MIC is 4 octets in length.

Field name	Description
LLID	The LLID indicates whether the packet is an LL Data PDU or an LL Control PDU. 00b = Reserved 01b = LL Data PDU: Continuation fragment of an L2CAP message, or an Empty PDU. 10b = LL Data PDU: Start of an L2CAP message or a complete L2CAP message with no fragmentation. 11b = LL Control PDU
NESN	Next Expected Sequence Number
SN	Sequence Number
MD	More Data
Length	The Length field indicates the size, in octets, of the Payload and MIC, if included.

Table 2.3: Data channel PDU Header field

2.4.1 LL Data PDU

An LL Data PDU is a data channel PDU that is used to send L2CAP data. The LLID field in the Header shall be set to either 01b or 10b.

An LL Data PDU with the LLID field in the Header set to 01b, and the Length field set to 00000b, is known as an Empty PDU. The master’s Link Layer may send an Empty PDU to the slave to allow the slave to respond with any Data Channel PDU, including an Empty PDU.

An LL Data PDU with the LLID field in the Header set to 10b shall not have the Length field set to 00000b.

2.4.2 LL Control PDU

An LL Control PDU is a Data Channel PDU that is used to control the Link Layer connection.

The LL Control PDU Payload is as shown in Figure 2.14.

Payload	
Opcode (1 octet)	CtrData (0 – 22 octets)

Figure 2.14: LL control PDU payload

An LL Control PDU shall not have the Length field set to 00000b. All LL Control PDUs have a fixed length, depending on the Opcode.

The Payload field consists of Opcode and CtrData fields.

The Opcode field identifies different types of LL Control PDU, as defined in Table 2.4.

The CtrData field in the LL Control PDU is specified by the Opcode field and is defined in Section 2.4.2.1 through Section 2.4.2.14.

Opcode	**Control PDU Name**
0x00	LL_CONNECTION_UPDATE_REQ
0x01	LL_CHANNEL_MAP_REQ
0x02	LL_TERMINATE_IND
0x03	LL_ENC_REQ
0x04	LL_ENC_RSP
0x05	LL_START_ENC_REQ
0x06	LL_START_ENC_RSP
0x07	LL_UNKNOWN_RSP
0x08	LL_FEATURE_REQ
0x09	LL_FEATURE_RSP
0x0A	LL_PAUSE_ENC_REQ
0x0B	LL_PAUSE_ENC_RSP
0x0C	LL_VERSION_IND
0x0D	LL_REJECT_IND
0x0E-0xFF	Reserved for Future Use

Table 2.4: LL Control PDU Opcodes

If an LL Control PDU is received that is not used or not supported, the Link Layer shall respond with an LL_UNKNOWN_RSP PDU. The UnknownType field of the LL_UNKNOWN_RSP PDU shall be set to the value of the not used or not supported Opcode.

If an LL Control PDU is received with an invalid Opcode, i.e. the Opcode field is set to a value that is Reserved for Future Use, or with invalid CtrData fields, the Link Layer shall respond with an LL_UNKNOWN_RSP PDU. The Unknown-Type field of the LL_UNKNOWN_RSP PDU shall be set to the value of the invalid Opcode.

2.4.2.1 LL_CONNECTION_UPDATE_REQ

The format of the CtrData field is as shown in Figure 2.15.

CtrData					
WinSize (1 octet)	WinOffset (2 octets)	Interval (2 octets)	Latency (2 octets)	Timeout (2 octets)	Instant (2 octets)

Figure 2.15: CtrData field of the LL_CONNECTION_UPDATE_REQ PDU

The LL_CONNECTION_UPDATE_REQ CtrData consists of six fields:

- The WinSize field shall be set to indicate the *transmitWindowSize* value, as defined in Section 4.5.3 in the following manner: *transmitWindowSize* = WinSize * 1.25 ms.
- The WinOffset field shall be set to indicate the *transmitWindowOffset value,* as defined in Section 4.5.3, in the following manner: *transmitWindowOffset* = WinOffset * 1.25 ms.
- The Interval field shall be set to indicate the *connInterval* value, as defined in Section 4.4.4, in the following manner: *connInterval* = Interval * 1.25 ms.
- The Latency field shall be set to indicate the *connSlaveLatency* value, as defined by Section 4.5.1, in the following manner: *connSlaveLatency* = Latency.
- The Timeout field shall be set to indicate the *connSupervisionTimeout* value, as defined by Section 4.5.1, in the following manner*:* -connSupervisionTimeout= Timeout * 10 ms.
- The Instant field shall be set to indicate the *connInstant value, as* defined by Section 5.1.1. The Instant Field shall have a value in the range of 1 to 32767.

2.4.2.2 LL_CHANNEL_MAP_REQ

The format of the CtrData field is shown in Figure 2.16.

CtrData	
ChM (5 octets)	Instant (2 octets)

Figure 2.16: CtrData field of the LL_CHANNEL_MAP_REQ PDU

The LL_CHANNEL_MAP_REQ CtrData consists of two fields:

- The ChM field shall contain the channel map indicating *Used* and *Unused* data channels. Every channel is represented with a bit positioned as per the data channel index defined by Section 4.5.8. The format of this field is identical to the ChM field in the CONNECT_REQ PDU (see Section 2.3.3.1).

- The Instant field shall be set to indicate the *connInstant* value, as defined by Section 5.1.2. The Instant field shall have a value in the range of 1 to 32767.

2.4.2.3 LL_TERMINATE_IND

The format of the CtrData field is shown in Figure 2.17.

CtrData
Error Code (1 octet)

Figure 2.17: CtrData field of the LL_TERMINATE_IND PDU

The LL_TERMINATE_IND CtrData consists of one field:

- The Error Code field shall be set to inform the remote device why the connection is about to be terminated. See [Vol 2] Part D for details.

2.4.2.4 LL_ENC_REQ

The format of the CtrData field is shown in Figure 2.18.

CtrData			
Rand (8 octets)	EDIV (2 octets)	SKDm (8 octets)	IVm (4 octets)

Figure 2.18: CtrData field of the LL_ENC_REQ PDU

The LL_ENC_REQ CtrData consists of four fields:

- The Rand field contains a random number that is provided by the Host and used with EDIV (see [Vol. 3] Part H, Section 2.4.4).
- The EDIV field contains the encrypted diversifier.
- The SKDm field contains the master's portion of the session key identifier.
- The IVm field contains the master's portion of the initialization vector.

2.4.2.5 LL_ENC_RSP

The format of the CtrData field is shown in Figure 2.19.

CtrData	
SKDs (8 octets)	IVs (4 octets)

Figure 2.19: CtrData field of the LL_ENC_ RSP PDU

The LL_ENC_RSP CtrData consists of two fields.

- The SKDs field shall contain the slave's portion of the session key identifier.

- The IVs field shall contain the slave’s portion of the initialization vector.

2.4.2.6 LL_START_ENC_REQ

The LL_START_ENC_REQ PDU does not have a CtrData field.

2.4.2.7 LL_START_ENC_RSP

The LL_START_ENC_RSP PDU does not have a CtrData field.

2.4.2.8 LL_UNKNOWN_RSP

The format of the CtrData field is shown in Figure 2.20.

CtrData
UnknownType (1 octet)

Figure 2.20: CtrData field of the LL_UNKNOWN_RSP PDU

The LL_UNKNOWN_RSP CtrData consists of one field:

- UnknownType shall contain the Opcode field value of the received LL Control PDU.

2.4.2.9 LL_FEATURE_REQ

The format of the CtrData field is shown in Figure 2.21.

CtrData
FeatureSet (8 octets)

Figure 2.21: CtrData field of the LL_FEATURE_REQ PDU

The LL_FEATURE_REQ CtrData consists of one field:

- FeatureSet shall contain the set of supported features of the master’s Link Layer.

2.4.2.10 LL_FEATURE_RSP

The format of the CtrData field is shown in Figure 2.22.

CtrData
FeatureSet (8 octets)

Figure 2.22: CtrData field of the LL_FEATURE_RSP PDU

The LL_FEATURE_RSP CtrData consists of one field:

- FeatureSet shall contain the set of used features of the slave's Link Layer.

2.4.2.11 LL_PAUSE_ENC_REQ

The LL_PAUSE_ENC_REQ packet does not have a CtrData field.

2.4.2.12 LL_PAUSE_ENC_RSP

The LL_PAUSE_ENC_RSP packet does not have a CtrData field.

2.4.2.13 LL_VERSION_IND

The format of the CtrData field is shown in Figure 2.23.

CtrData		
VersNr (1 octet)	CompId (2 octets)	SubVersNr (2 octets)

Figure 2.23: CtrData field of the LL_VERSION_IND PDU

The LL_VERSION_IND CtrData consists of three fields:

- VersNr field shall contain the version of the Bluetooth Controller specification (see Bluetooth Assigned Numbers).
- CompId field shall contain the company identifier of the manufacturer of the Bluetooth Controller (see Bluetooth Assigned Numbers).
- SubVersNr field shall contain a unique value for each implementation or revision of an implementation of the Bluetooth Controller.

2.4.2.14 LL_REJECT_IND

The format of the CtrData field is shown in Figure 2.24.

CtrData
Error Code (1 octet)

Figure 2.24: CtrData field of the LL_ REJECT_IND

Error Code shall contain the reason a request was rejected; see [Vol 2] Part D.

3 BIT STREAM PROCESSING

Bluetooth devices shall use the bitstream processing schemes as defined in the following sections.

Figure 3.1 shows the processes that may have to be carried out on the PDU.

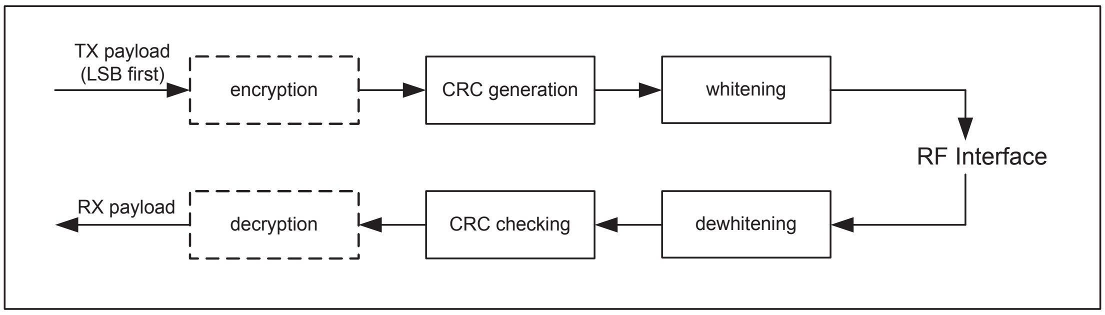

Figure 3.1: Payload bit processes

3.1 ERROR CHECKING

At packet reception, the Access Address shall be checked first. If the Access Address is incorrect, the packet shall be rejected, otherwise the packet shall be considered received. If the CRC is incorrect, the packet shall be rejected, otherwise the packet shall be considered valid. A packet shall only be processed if the packet is considered valid. A packet with an incorrect CRC may cause a connection event to continue, as specified in Section 4.5.1.

3.1.1 CRC Generation

The CRC shall be calculated on the PDU field in all Link Layer packets. If the PDU is encrypted, then the CRC shall be calculated after encryption of the PDU has been performed.

The CRC polynomial is a 24-bit CRC and all bits in the PDU shall be processed in transmitted order starting from the least significant bit. The polynomial has the form of $x^{24} + x^{10} + x^9 + x^6 + x^4 + x^3 + x + 1$. For every Data Channel PDU, the shift register shall be preset with the CRC initialization value set for the Link Layer connection and communicated in the CONNECT_REQ PDU. For every Advertising Channel PDU the shift register shall be preset with 0x555555.

Position 0 shall be set as the least significant bit and position 23 shall be set as the most significant bit of the initialization value. The CRC is transmitted most significant bit first, i.e. from position 23 to position 0 (see Section 1.2).

Figure 3.2 shows an example linear feedback shift register (LFSR) to generate the CRC.

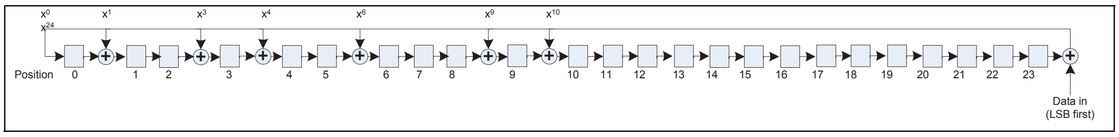

Figure 3.2: The LFSR circuit generating the CRC

3.2 DATA WHITENING

Data whitening is used to avoid long sequences of zeros or ones, e.g. 0000000b or 1111111b, in the data bit stream. Whitening shall be applied on the PDU and CRC fields of all Link Layer PDUs and is performed after the CRC in the transmitter. De-whitening is performed before the CRC in the receiver (see Figure 3.1).

The whitener and de-whitener are defined the same way, using a 7-bit linear feedback shift register with the polynomial $x^7 + x^4 + 1$. Before whitening or de-whitening, the shift register is initialized with a sequence that is derived from the channel index (data channel index or advertising channel index) in which the packet is transmitted in the following manner:

- Position 0 is set to one.
- Positions 1 to 6 are set to the channel index of the channel used when transmitting or receiving, from the most significant bit in position 1 to the least significant bit in position 6.

For example, if the channel index = 23 (0x17), the positions would be set as follows:

Position 0 = 1

Position 1 = 0

Position 2 = 1

Position 3 = 0

Position 4 = 1

Position 5 = 1

Position 6 = 1

Figure 3.3 shows an example linear feedback shift register (LFSR) to generate data whitening.

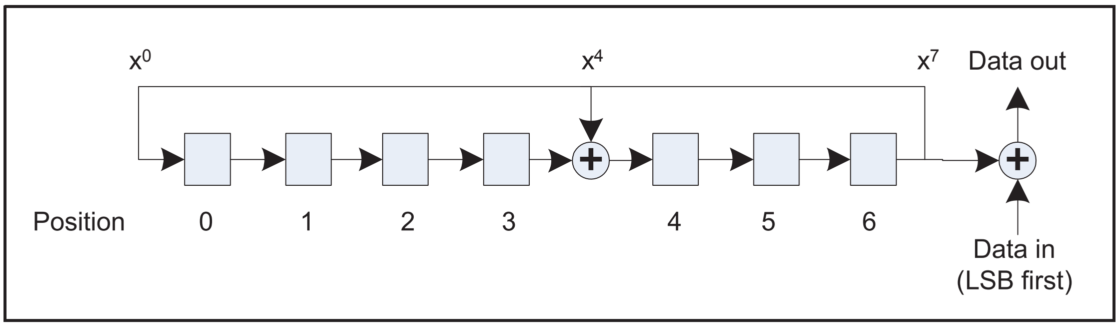

Figure 3.3: The LFSR circuit to generate data whitening

4 AIR INTERFACE PROTOCOL

The air interface protocol consists of the multiple access scheme, device discovery and link layer connection methods.

4.1 INTER FRAME SPACE

The time interval between two consecutive packets on the same channel index is called the Inter Frame Space. It is defined as the time from the end of the last bit of the previous packet to the start of the first bit of the subsequent packet. The Inter Frame Space is designated “T_IFS” and shall be 150 μs.

4.2 TIMING REQUIREMENTS

The Link Layer shall use one of two possible clock accuracies. During a connection event or advertising event the Link Layer shall use the active clock accuracy; otherwise it shall use the sleep clock accuracy.

4.2.1 Active Clock Accuracy

The average timing of packet transmission during a connection event is determined using the active clock accuracy, with a drift less than or equal to ±50 ppm. All instantaneous timings shall not deviate more than 2 μs from the average timing.

Note: This means that the start of a packet shall be transmitted 150±2 μs after the end of the previous packet.

4.2.2 Sleep Clock Accuracy

The timing of advertising events (see Section 4.4.2.2) and connection events (see Section 4.5.7) is determined using the sleep clock accuracy, with a drift less than or equal to ±500 ppm.

The instantaneous timing of the anchor point (see Section 4.5.7) shall not deviate more than 16 μs from the average timing.

Note: This means that a 1 sec connection interval with a total ±1000ppm sleep clock accuracy will give a window widening either side of the anchor point of 1ms plus 16us, assuming that the slave controller was using its sleep clock for almost the complete connection interval.

4.3 LINK LAYER DEVICE FILTERING

The Link Layer may perform device filtering based on the device address of the peer device. Link Layer Device Filtering is used by the Link Layer to minimize the number of devices to which it responds.

A Link Layer shall support Link Layer Device Filtering unless it only supports non-connectable advertising.

The filter policies for the Advertising State, Scanning State and Initiating State are independent of each other. When the Link Layer is in the Advertising State, the advertising filter policy shall be used. When the Link Layer is in the Scanning State, the scanning filter policy shall be used. When the Link Layer is in the Initiating State, the initiator filter policy shall be used. If the Link Layer does not support the Advertising State, Scanning State, or Initiating State, the corresponding filter policy is not required to be supported.

4.3.1 White List

The set of devices that the Link Layer uses for device filtering is called the White List.

A White List contains a set of White List Records used for Link Layer Device Filtering. A White List Record contains both the device address and the device address type (public or random). All Link Layers supporting Link Layer Device Filtering shall support a White List capable of storing at least one White List Record.

On reset, the White List shall be empty.

The White List is configured by the Host and is used by the Link Layer to filter advertisers, scanners or initiators. This allows the Host to configure the Link Layer to act on a request without awakening the Host.

All the device filter policies shall use the same White List.

4.3.2 Advertising Filter Policy

The advertising filter policy determines how the advertiser’s Link Layer processes scan and/or connection requests.

When the Link Layer is using connectable directed advertising the advertising filter policy shall be ignored, otherwise the Link Layer shall use one of the following advertising filter policy modes which are configured by the Host:

- The Link Layer shall process scan and connection requests only from devices in the White List.
- The Link Layer shall process scan and connection requests from all devices (i.e. the White List is not in use). This is the default on reset.
- The Link Layer shall process scan requests from all devices and shall only process connection requests from devices that are in the White List.
- The Link Layer shall process connection requests from all devices and shall only process scan requests from devices that are in the White List.

Only one advertising filter policy mode shall be supported at a time.

4.3.3 Scanner Filter Policy

The scanner filter policy determines how the scanner's Link Layer processes advertising packets. The Link Layer shall use one of the following scanner filter policy modes which are configured by the Host:

- The Link Layer shall process advertising packets only from devices in the White List.
- The Link Layer shall process all advertising packets (i.e., the White List is not used). This is the default on reset.

In addition to the scanner filter policy, a connectable directed advertising packet not containing the scanner's device address shall be ignored.

Only one scanner filter policy mode shall be supported at a time.

4.3.4 Initiator Filter Policy

The initiator filter policy determines how an initiator's Link Layer processes advertising packets. The Link Layer shall use one of the following initiator filter policy modes which are configured by the Host:

- The Link Layer shall process connectable advertising packets from all devices in the White List.
- The Link Layer shall ignore the White List and process connectable advertising packets from a specific single device specified by the Host.

If the Link Layer receives a connectable directed advertising packet from an advertiser that is not contained in the White List or the single address specified by the Host, the connectable directed advertising packet shall be ignored.

Only one initiator filter policy mode shall be supported at a time.

4.4 NON-CONNECTED STATES

4.4.1 Standby State

The Standby State is the default state in the Link Layer. The Link Layer shall not send or receive packets in the Standby State. The Link Layer may leave the Standby State to enter the Advertising State, Scanning State or Initiator State.

4.4.2 Advertising State

The Link Layer shall enter the Advertising State when directed by the Host. When placed in the Advertising State, the Link Layer shall send advertising PDUs (see Section 2.3.1) in advertising events.

Each advertising event is composed of one or more advertising PDUs sent on used advertising channel indexes. The advertising event shall be closed after one advertising PDU has been sent on each of the used advertising channel indexes (see Section 4.4.2.1) or the advertiser may close an advertising event earlier to accommodate other functionality.

The time between two consecutive advertising events is defined in Section 4.4.2.2.

An advertising event can be one of the following types:

- a connectable undirected event
- a connectable directed event
- a non-connectable undirected event
- a scannable undirected event

For each advertising event type, a corresponding Advertising Channel PDU is used.

The first PDU of each advertising event shall be transmitted in the used advertising channel with the lowest advertising channel index.

The advertising event type determines the allowable response PDUs. The table below specifies the allowable responses for each advertising event.

Advertising Event Type	**PDU used in this advertising event type**	**Allowable response PDUs for advertising event**	
		SCAN_REQ	CONNECT_REQ
Connectable Undirected Event	ADV_IND	YES	YES
Connectable Directed Event	ADV_DIRECT_IND	NO	YES*
Non-connectable Undirected Event	ADV_NONCONN_IND	NO	NO
Scannable Undirected Event	ADV_SCAN_IND	YES	NO

Table 4.1: Advertising event types, PDUs used and allowable response PDUs

* Only the correctly addressed initiator may respond.

If the advertiser receives a PDU for the advertising event that is not explicitly allowed it shall be ignored. If no PDU is received or the received PDU was ignored, the advertiser shall either send an advertising PDU on the next used advertising channel index or close the advertising event.

4.4.2.1 Advertising Channel Selection

Advertising events use three predefined advertising channels. Advertising channel indexes are either used or unused.

The Link Layer shall use the advertising channel indexes as specified by the Host, and the used advertising channel indexes shall take effect when the Advertising State is entered.

4.4.2.2 Advertising Interval

For all undirected advertising events, the time between the start of two consecutive advertising events (*T_advEvent*) is computed as follows for each advertising event:

T_advEvent = advInterval + advDelay

The *advInterval* shall be an integer multiple of 0.625 ms in the range of 20 ms to 10.24 s. If the advertising event type is either a scannable undirected event type or a non-connectable undirected event type, the *advInterval* shall not be less than 100 ms. If the advertising event type is a connectable undirected event type, the *advInterval* can be 20 ms or greater.

The *advDelay* is a pseudo-random value with a range of 0 ms to 10 ms generated by the Link Layer for each advertising event.

As illustrated in Figure 4.1, the advertising events are perturbed in time using the advDelay.

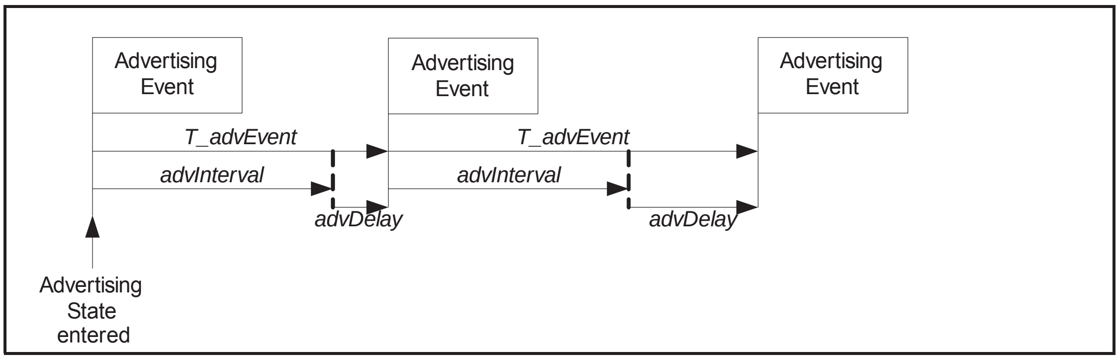

Figure 4.1: Advertising events perturbed in time using advDelay

4.4.2.3 Connectable Undirected Event Type

When the connectable undirected advertising event type is used, advertising indications (ADV_IND PDU) are sent by the Link Layer.

The connectable undirected advertising event type allows a scanner or initiator to respond with either a scan request or connect request. A scanner may send a scan request (SCAN_REQ PDU) to request additional information about the advertiser. An initiator may send a connect request (CONNECT_REQ PDU) to request the Link Layer to enter the Connection State.

The Link Layer shall listen on the same advertising channel index for requests from scanners or initiators.

If the advertiser receives a SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy, it shall reply with a SCAN_RSP PDU on the same advertising channel index. After the SCAN_RSP PDU is sent, or if the advertising filter policy prohibited processing the SCAN_REQ PDU, the advertiser shall either move to the next used advertising channel index to send another ADV_IND PDU, or close the advertising event.

If the advertiser receives a CONNECT_REQ PDU that contains its device address, from an initiator allowed by the advertising filter policy, the Link Layer shall exit the Advertising State and transition to the Connection State in the Slave Role as defined in Section 4.5.5. If the advertising filter policy prohibited processing the received CONNECT_REQ PDU, the advertiser shall either move to the next used advertising channel index to send another ADV_IND PDU, or close the advertising event.

The time between the beginning of two consecutive ADV_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising state shall be closed within the advertising interval.

An illustration of an advertising event using all the advertising channel indexes and in which no SCAN_REQ or CONNECT_REQ PDUs are received is shown in Figure 4.2.

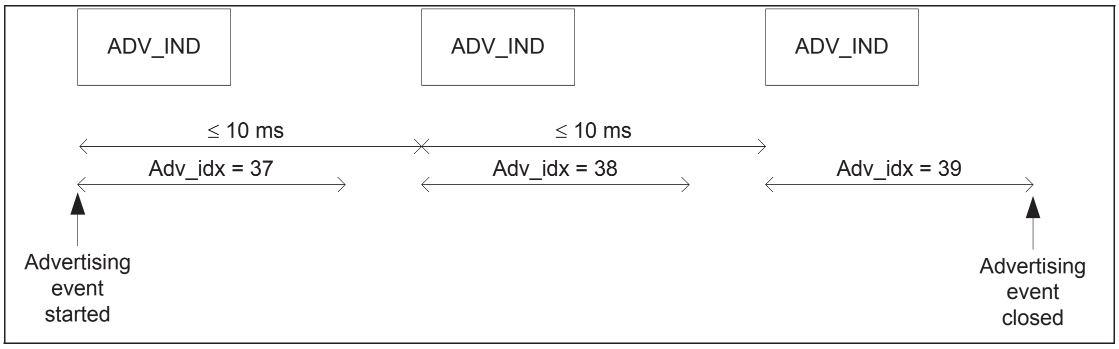

Figure 4.2: Connectable undirected advertising event with only advertising PDUs

Two illustrations of advertising events using all the advertising channel indexes during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.3 and in Figure 4.4.

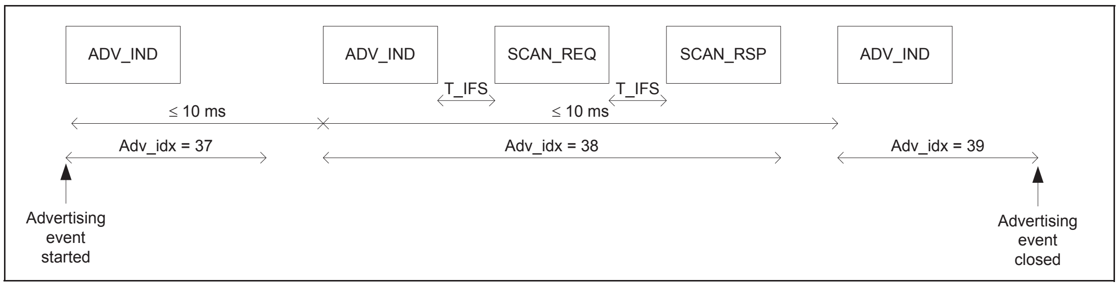

Figure 4.3: Connectable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event

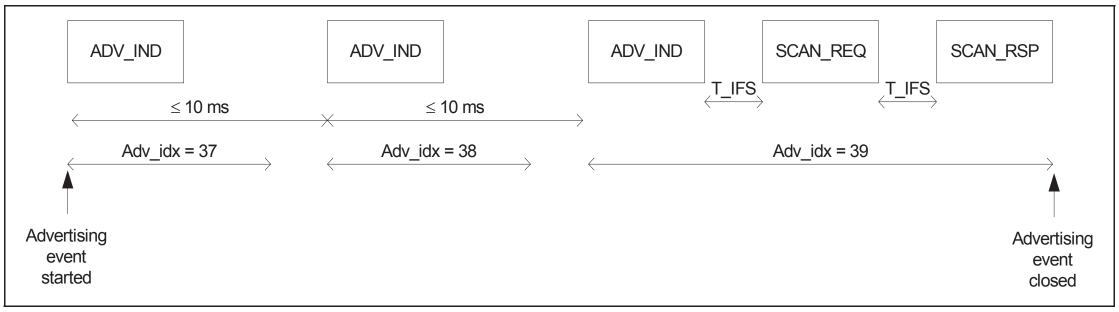

Figure 4.4: Connectable undirected advertising event with SCAN_REQ and SCAN_RSP packets at the end of an advertising event

Figure 4.5 illustrates an advertising event during which a CONNECT_REQ PDU is received on the second advertising channel index.

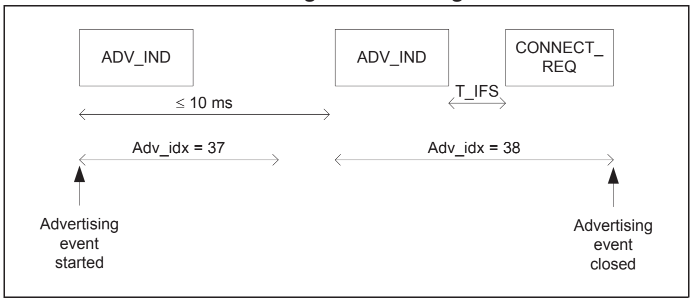

Figure 4.5: Connectable undirected advertising event during which a CONNECT_REQ PDU is received

4.4.2.4 Connectable Directed Event Type

When the connectable directed advertising event type is used, directed advertising indications (ADV_DIRECT_IND PDUs) are sent by the Link Layer.

The connectable directed advertising event type allows an initiator to respond with a connect request. An initiator may send a connect request (CONNECT_REQ PDU) to request the Link Layer to enter the Connection State.

The ADV_DIRECT_IND PDU contains both the initiator's device address and the advertiser's device address. Only the addressed initiator may initiate a Link

Layer connection with the advertiser by sending a CONNECT_REQ PDU to the advertiser.

After every ADV_DIRECT_IND PDU sent by the advertiser, the advertiser shall listen for CONNECT_REQ PDUs on the same advertising channel index. Any SCAN_REQ PDUs received shall be ignored.

If the advertiser receives a CONNECT_REQ PDU that contains its device address and the initiator device address is contained in the ADV_DIRECT_IND PDU, the Link Layer shall exit the Advertising State and transition to the Connection State in the Slave Role as defined in Section 4.5.5. Otherwise, the advertiser shall either move to the next used advertising channel index to send another ADV_DIRECT_IND PDU, or close the advertising event.

The time between the start of two consecutive ADV_DIRECT_IND PDUs sent on the same advertising channel index shall be less than or equal to 3. 75 ms.

The Link Layer shall exit the Advertising State no later than 1.28 s after the Advertising State was entered.

A sequence of five ADV_DIRECT_IND PDUs in two advertising events without CONNECT_REQ PDUs is shown in Figure 4.6 for the case in which all the advertising channels are used.

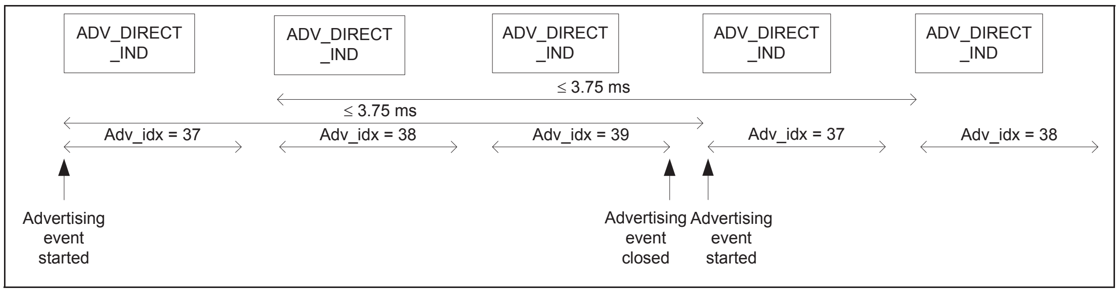

Figure 4.6: Connectable directed advertising event type with only advertising PDUs

Note: Connectable directed advertising is designed for cases in which fast Link Layer connection setup is essential (e.g., a reconnection). It is a power and bandwidth intensive advertising scheme that should only be used when fast connection setup is required.

4.4.2.5 Scannable Undirected Event Type[1]

When the scannable undirected advertising event type is used, scannable advertising indications (ADV_SCAN_IND PDUs) packets are sent by the Link Layer.

1. Discoverable Undirected Event Type was renamed to Scannable Undirected Event Type

The scannable undirected event type allows a scanner to respond with a scan request (SCAN_REQ PDU) to request additional information about the advertiser.

The Link Layer shall listen on the same advertising channel index for requests from scanners.

If the advertiser receives a SCAN_REQ PDU that contains its device address from a scanner allowed by the advertising filter policy it shall reply with a SCAN_RSP PDU on the same advertising channel index. After the SCAN_RSP PDU is sent or if the advertising filter policy prohibited processing the SCAN_REQ PDU the advertiser shall either move to the next used advertising channel index to send another ADV_SCAN_IND PDU, or close the advertising event.

The time between the beginning of two consecutive ADV_SCAN_IND PDUs within an advertising event shall be less than or equal to 10ms. The advertising event shall be closed within the advertising interval.

The structure of an advertising event in which no SCAN_REQ PDU was received is shown in Figure 4.7 for the case in which all the advertising channels are used.

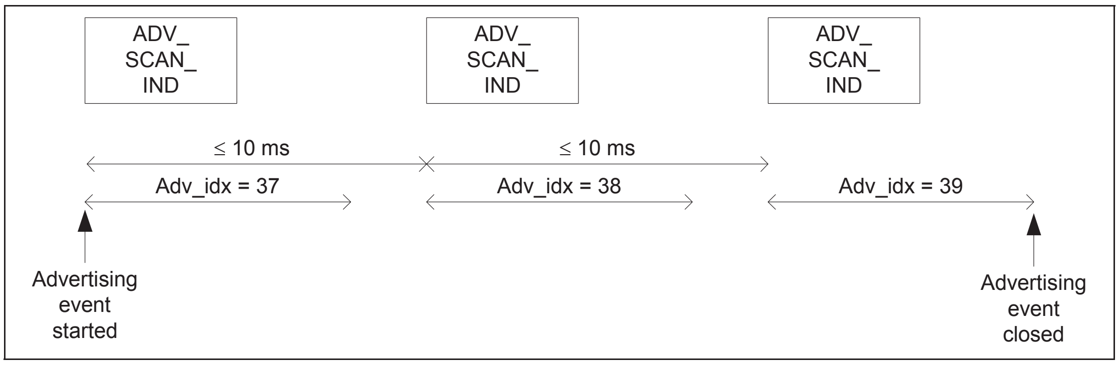

Figure 4.7: Scannable undirected advertising event with only advertising PDUs

Two example advertising events during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.8 and in Figure 4.9 for the case in which all the advertising channels are used.

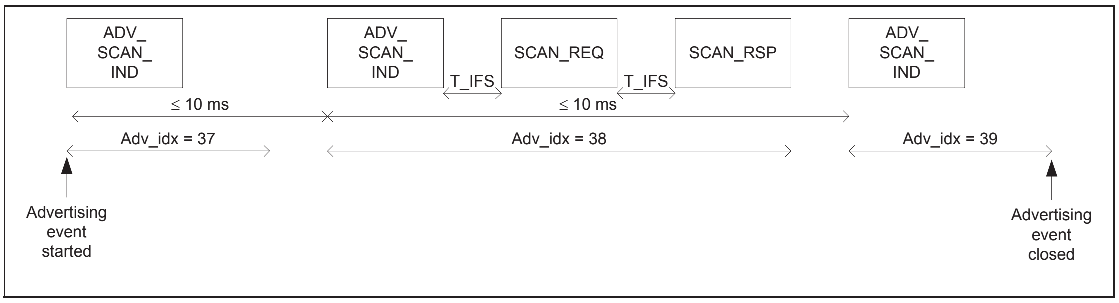

Figure 4.8: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs in the middle of an advertising event

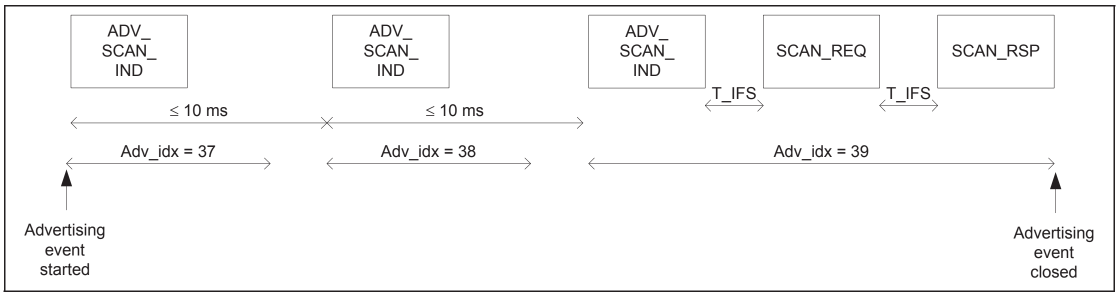

Figure 4.9: Scannable undirected advertising event with SCAN_REQ and SCAN_RSP PDUs at the end of an advertising event

4.4.2.6 Non-connectable Undirected Event Type

When the non-connectable undirected event type is used, non-connectable advertising indications (ADV_NONCONN_IND PDU) packets are sent by the Link Layer.

The non-connectable undirected event type allows a scanner to receive information contained in the ADV_NONCONN_IND PDU from the advertiser.

The advertiser shall either move to the next used advertising channel index or close the advertising event after each ADV_NONCONN_IND PDU that is sent. The Link Layer does not listen, and therefore cannot receive any requests from scanners or initiators.

The time between the beginning of two consecutive ADV_NONCONN_IND PDUs within an advertising event shall be less than or equal to 10 ms. The advertising event shall be closed within the advertising interval.

An illustration of a non-connectable advertising event is shown in Figure 4.10 for the case in which all the advertising channels are used.

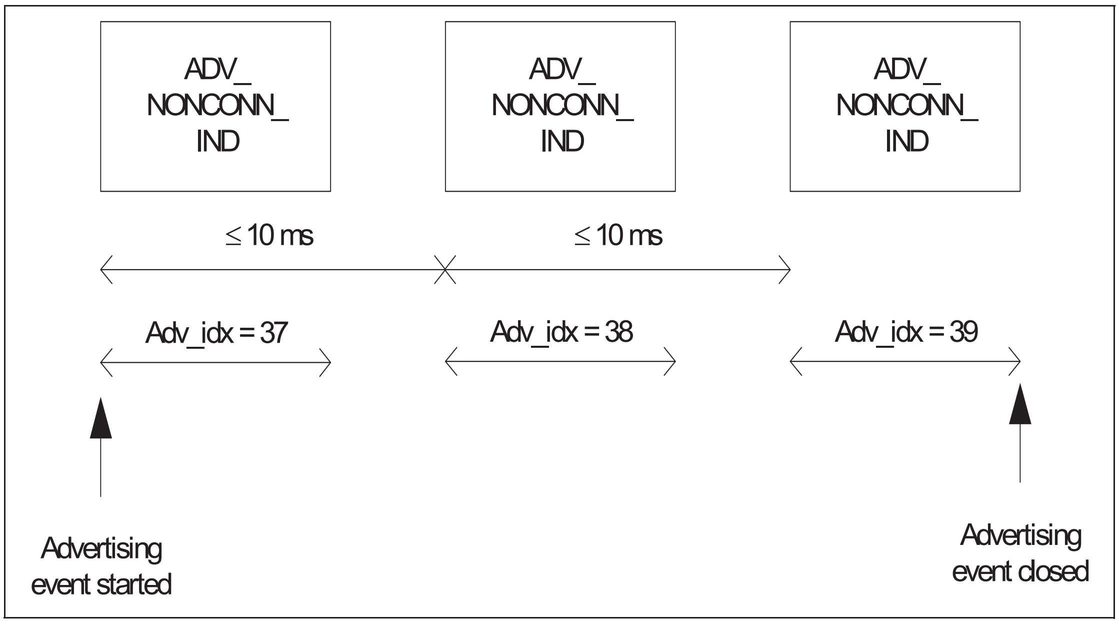

Figure 4.10: Non-connectable undirected advertising event

4.4.3 Scanning State

The Link Layer shall enter the Scanning State when directed by the Host. When scanning, the Link Layer shall listen on the advertising channel indices. There are two types of scanning, determined by the Host: passive and active.

There are no strict timing or advertising channel index selection rules for scanning.

During scanning, the Link Layer listens on an advertising channel index for the duration of the scan window, *scanWindow*. The scan interval, *scanInterval*, is defined as the interval between the start of two consecutive scan windows.

The Link Layer should listen for the complete *scanWindow* every *scanInterval* as directed by the Host unless there is a scheduling conflict. In each scan window, the Link Layer should scan on a different advertising channel index. The Link Layer shall use all the advertising channel indices.

The *scanWindow* and *scanInterval* parameters shall be less than or equal to 10.24 s. The *scanWindow* shall be less than or equal to the *scanInterval*. If the *scanWindow* and the *scanInterval* parameters are set to the same value by the Host, the Link Layer should scan continuously.

The scanner filter policy shall apply when receiving an advertising PDU when scanning.

For each non-duplicate ADV_DIRECT_IND PDU received that contains this Link Layer's device address from an advertiser the Link Layer shall send an advertising report to the Host. For each non-duplicate ADV_IND, ADV_SCAN_IND, ADV_NONCONN_IND, or SCAN_RSP PDU from an advertiser, the Link Layer shall send an advertising report to the Host. The advertising report shall contain at least the advertiser's device address and advertising data or scan response data if present. Duplicate advertising reports are not required to be sent to the Host. A duplicate advertising report is an advertising report for the same device address while the Link Layer stays in the Scanning State. The advertising data may change; advertising data or scan response data is not considered significant when determining duplicate advertising reports.

4.4.3.1 Passive Scanning

When in passive scanning, the Link Layer will only receive packets; it shall not send any packets.

4.4.3.2 Active Scanning

In active scanning, the Link Layer shall listen for advertising PDUs and depending on the advertising PDU type it may request an advertiser to send additional information.

The Link Layer shall not send a SCAN_REQ PDU to an advertiser from which an ADV_DIRECT_IND PDU or ADV_NONCONN_IND PDU is received.

The Link Layer shall send at least one SCAN_REQ PDU after entering the Scanning State to advertisers from which ADV_IND or ADV_SCAN_IND PDUs are received, The Link Layer shall send further SCAN_REQ PDUs to advertisers from which ADV_IND or ADV_SCAN_IND PDUs have been received. The Link Layer should interleave SCAN_RSP PDUs to multiple advertisers.

The scanner shall run a backoff procedure to minimize collisions of SCAN_REQ PDUs from multiple scanners.

The backoff procedure uses two parameters, *backoffCount* and *upperLimit* to restrict the number of SCAN_REQ PDUs sent when collisions occur on SCAN_RSP PDUs.

Upon entering Scanning State, the *upperLimit* shall be set to one and the *backoffCount* shall be set to one.

On every received ADV_IND PDU or ADV_SCAN_IND PDU that is allowed by the scanner filter policy and a SCAN_REQ PDU is to be sent the *backoffCount* shall be decremented by one until it reaches the value of zero. The SCAN_REQ PDU shall only be sent when *backoffCount* becomes zero.

After sending a SCAN_REQ PDU the Link Layer shall listen for a SCAN_RSP PDU from that advertiser. If the SCAN_RSP PDU was not received from that advertiser, it is considered a failure otherwise it is considered a success. On every two consecutive failures, the *upperLimit* shall be doubled until it reaches the value of 256. On every two consecutive successes, the *upperLimit* shall be halved until it reaches the value of one. After success or failure of receiving the SCAN_RSP PDU, the Link Layer shall set *backoffCount* to a new pseudo-random integer between one and *upperLimit*.

Two illustrations of advertising events using all the advertising channel indexes during which a SCAN_REQ PDU is received and a SCAN_RSP PDU is sent are shown in Figure 4.3 and in Figure 4.4.

4.4.4 Initiating State

The Link Layer shall enter the Initiating State when directed by the Host. When initiating, the Link Layer shall listen on the advertising channel indices.

There are no strict timing or advertising channel index selection rules for initiators.

During initiating, the Link Layer listens on an advertising channel index for the duration of the scan window, *scanWindow*. The scan interval, *scanInterval*, is defined as the interval between the start of two consecutive scan windows.

The Link Layer should listen for the complete *scanWindow* every *scanInterval* as directed by the Host unless there is a scheduling conflict. In each scan window, the Link Layer should listen on a different advertising channel index. The Link Layer shall use all the advertising channel indexes.

The *scanWindow* and *scanInterval* parameters shall be less than or equal to 10.24 s. The *scanWindow* shall be less than or equal to the *scanInterval*. If the *scanWindow* and the *scanInterval* parameters are set to the same value by the Host, the Link Layer should listen continuously.

If an ADV_IND PDU is received that is allowed by the initiator filter policy, the initiator shall send a CONNECT_REQ PDU to the advertiser. If an ADV_DIRECT_IND PDU with this Link Layer's device address is received that is allowed by the initiator filter policy, the initiator shall send a CONNECT_REQ PDU to the advertiser; otherwise it shall be ignored.

After sending the CONNECT_REQ PDU, the Link Layer shall exit the Initiating State, and shall transition to the Connection State in the Master Role as defined in Section 4.5.4.

4.5 CONNECTION STATE

The Link Layer enters the Connection State when an initiator sends a CONNECT_REQ PDU to an advertiser or an advertiser receives a CONNECT_REQ PDU from an initiator.

After entering the Connection State, the connection is considered to be created. The connection is not considered to be established at this point. A connection is only considered to be established once a data channel packet has been received from the peer device. The only difference between a connection that is created and a connection that is established is the Link Layer connection supervision timeout value that is used (see Section 4.5.2).

When two devices are in a connection, the two devices act in different roles. A Link Layer in the Master Role is called a master. A Link Layer in the Slave Role is called a slave. The master controls the timing of a connection event. A connection event is a point of synchronization between the master and the slave.

4.5.1 Connection Events

The Link Layer in the Connection State shall only transmit Data Channel PDUs (see Section 2.4) in connection events. The master and slave shall determine the data channel index for each connection event as defined in Section 4.5.8. The same data channel index shall be used for all packets in the connection event. Each connection event contains at least one packet sent by the master.

During a connection event, the master and slave alternate sending and receiving packets. The connection event is considered open while both devices continue to send packets. The slave shall always send a packet if it receives a packet from the master regardless of a valid CRC match, except after multiple consecutive invalid CRC matches as specified in Section 4.5.6. The master may send a packet if it receives a packet from the slave regardless of a valid CRC match. The Length field of the Header is assumed to be correct even if the CRC match was invalid. If the master does not receive a packet from the slave, the master shall close the connection event.

The connection event can be closed by either device, as defined in Section 4.5.6.

The timing of connection events is determined by two parameters: connection event interval (*connInterval*), and slave latency (*connSlaveLatency*).

The start of a connection event is called an anchor point. At the anchor point, a master shall start to transmit a Data Channel PDU to the slave. The start of connection events are spaced regularly with an interval of *connInterval* and shall not overlap. The master shall ensure that a connection event closes at least T_IFS before the anchor point of the next connection event. The slave listens for the packet sent by its master at the anchor point.

The *connInterval* shall be a multiple of 1.25 ms in the range of 7.5 ms to 4.0 s. The *connInterval* is set by the Initiator's Link Layer in the CONNNECT_REQ PDU from the range given by the Host.

Slave latency allows a slave to use a reduced number of connection events. The connSlaveLatency parameter defines the number of consecutive connection events that the slave device is not required to listen for the master. The value of connSlaveLatency should not cause a Supervision Timeout (see Section 4.5.2). connSlaveLatency shall be an integer in the range of 0 to ((connSupervisionTimeout / connInterval) - 1). connSlaveLatency shall also be less than 500. When connSlaveLatency is set to zero the slave device shall listen at every anchor point. If the slave does not receive a packet from the master after applying slave latency, it should listen at each anchor point and not apply slave latency until it receives a packet from the master.

Both the master and the slave shall have a 16-bit connection event counter (*connEventCounter*) for each Link Layer connection. It shall be set to zero on the first connection event sent by the master of the connection. It shall be incremented by one for each new connection event sent by the master; the *connEventCounter* shall wrap from 0xFFFF to 0x0000. This counter is used to synchronize Link Layer control procedures.

The slave shall increment *connEventCounter* for all connection events, even if it may not be listening to the master due to slave latency in those events.

4.5.2 Supervision Timeout

A connection can break down due to various reasons such as a device moving out of range, encountering severe interference or a power failure condition. Since this may happen without any prior warning, it is important for both the master and the slave to monitor the status of the connection.

To be able to detect link loss, both the master and the slave shall use a Link Layer connection supervision timer, $T_{LLconnSupervision}$. Upon reception of a valid packet, the timer shall be reset.

If the Link Layer connection supervision timer reaches 6 * *connInterval* before the connection is established (see Section 4.5), the connection shall be considered lost. This enables fast termination of connections that fail to establish.

Connection supervision timeout (*connSupervisionTimeout*) is a parameter that defines the maximum time between two received Data Packet PDUs before the connection is considered lost. The *connSupervisionTimeout* shall be a multiple of 10 ms in the range of 100 ms to 32.0 s and it shall be larger than (1 + *connSlaveLatency*) * *connInterval*.

If at any time in Connection State after the connection has been established and the timer reaches the *connSupervisionTimeout* value, the connection shall be considered lost.

If the connection is considered lost, the Link Layer shall not send any further packets. The Link Layer exits the Connection State and shall transition to the Standby State. The Host shall be notified of the loss of connection.

4.5.3 Connection Event Transmit Window

To allow the master to efficiently schedule connection events for multiple connections or other activities it may be involved in, the master has the flexibility to schedule the first connection event anchor point at a time of its choosing. The CONNECT_REQ PDU includes parameters to determine when the master can send its first packet in the Connection State to set the anchor point and when the slave must listen.

The CONNECT_REQ PDU includes three parameters used to determine the transmit window. The transmit window starts at *transmitWindowOffset* + 1.25 ms after the end of the CONNECT_REQ PDU, and the *transmitWindowSize* parameter shall define the size of the transmit window. The *connInterval* is used in the calculation of the maximum offset and size of the transmit window. *transmitWindowOffset* and *transmitWindowSize* are determined by the Link Layer.

The *transmitWindowOffset* shall be a multiple of 1.25 ms in the range of 0 ms to *connInterval*. The *transmitWindowSize* shall be a multiple of 1.25 ms in the range of 1.25 ms to the lesser of 10 ms and (*connInterval* - 1.25 ms).

Therefore the start of the first packet will be no earlier than 1.25 ms + *transmitWindowOffset* and no later than 1.25 ms + *transmitWindowOffset* + *transmitWindowSize* after the end of the CONNECT_REQ PDU transmitted in the advertising channel.

4.5.4 Connection Setup – Master Role

After the initiator sends the CONNECT_REQ PDU the Link Layer is in Connection State in the Master Role. The master shall reset the Link Layer connection supervision timer $T_{LLconnSupervision}$. The Link Layer shall notify the Host that the connection has been created. The first connection event shall use the data channel index as specified in Section 1.4.1.

The master shall start to send the first packet within the transmit window as defined in Section 4.5.3. It is permitted that the master's first packet can extend beyond the transmit window.

The first packet sent in the Connection State by the master determines the anchor point for the first connection event, and therefore the timings of all future connection events in this connection.

The second connection event anchor point shall be *connInterval* after the first connection event anchor point. All the normal connection event transmission rules specified in Section 4.5.1 shall apply.

Two examples of the LL connection setup procedure timing from master's perspective are shown in Figure 4.11 and in Figure 4.12.

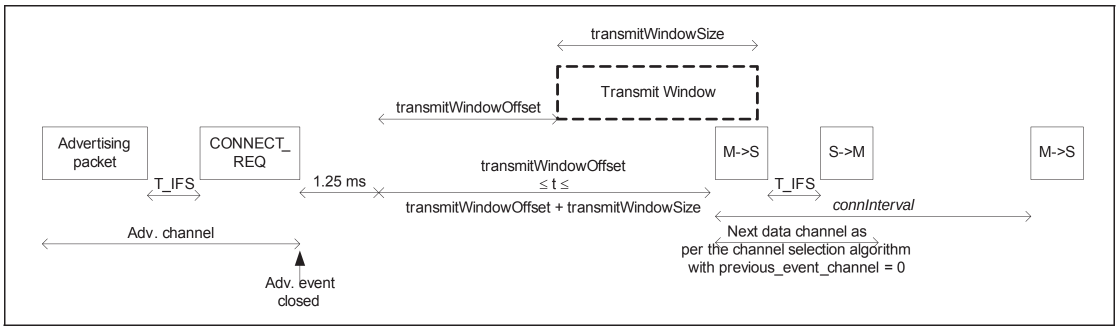
transmitWindowSize
Transmit Window
transmitWindowOffset
Advertising packet
CONNECT_ REQ
T_IFS
1.25 ms
transmitWindowOffset ≤ t ≤ transmitWindowOffset + transmitWindowSize
M->S
S->M
M->S
T_IFS
connInterval
Adv. channel
Next data channel as per the channel selection algorithm with previous_event_channel = 0
Adv. event closed

Figure 4.11: Master's view on LL connection setup with a non-zero transmitWindowOffset

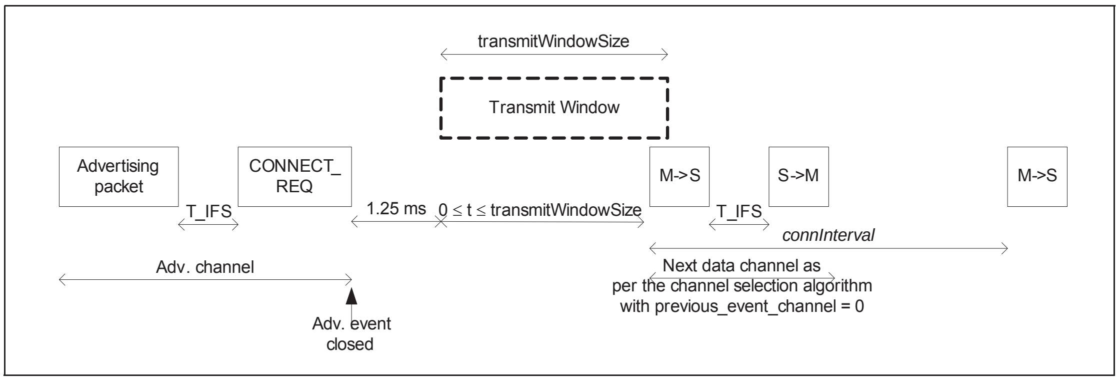

Figure 4.12: Master's view on LL connection setup with transmitWindowOffset set to zero

4.5.5 Connection Setup – Slave Role

After the advertiser receives a CONNECT_REQ PDU the Link Layer is in Connection State in the Slave Role. The slave shall reset the Link Layer connection supervision timer $T_{LLconnSupervision}$. The Link Layer shall notify the Host that the connection has been created. The first connection event shall use the data channel index as specified in Section 1.4.1.

The slave shall start to listen for the first packet within the transmit window as defined in Section 4.5.3. It is permitted that the master's first packet can extend beyond the transmit window, and therefore the slave must take this into account.

The first packet received, regardless of a valid CRC match (i.e., only the access code matches), in the Connection State by the slave determines the anchor point for the first connection event, and therefore the timings of all future connection events in this connection.

If a packet is not received in a transmit window, the slave shall attempt to receive a packet in a subsequent transmit window. A subsequent transmit window shall start *connInterval* after the start of the previous transmit window, with the same *transmitWindowSize*. The data channel index shall be the next data channel index as specified in Section 1.4.1. The *connEventCount* shall also be incremented by one.

An example of the procedure from the slave's perspective is shown in Figure 4.13 in which the slave fails to receive any part of the first packet (i.e., connEventCount = 0) from the master and acquires anchor point timing from the second packet (i.e., connEventCount = 1).

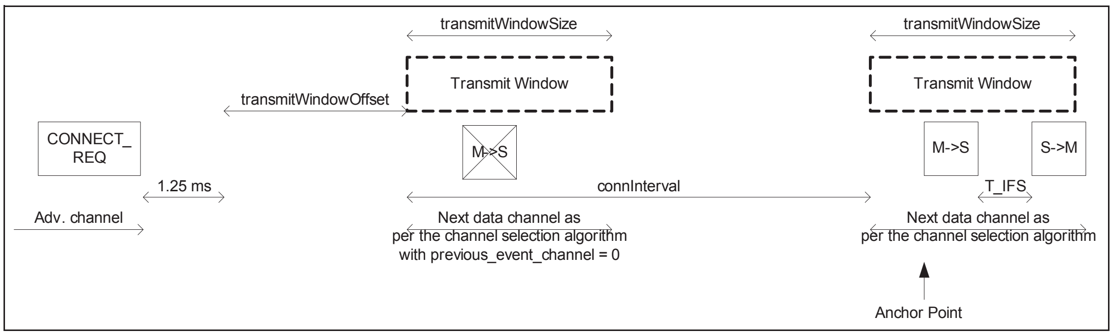

Figure 4.13: Slave closing LL connection setup in the second LL connection event

The slave shall be active in every connection event until the NESN is set to one in a master's packet at which point the slave can use slave subrating as defined in Section 4.5.1.

4.5.6 Closing Connection Events

The MD bit of the Header of the Data Channel PDU is used to indicate that the device has more data to send. If neither device has set the MD bit in their packets, the packet from the slave closes the connection event. If either or both of the devices have set the MD bit, the master may continue the connection event by sending another packet, and the slave should listen after sending its packet. If a packet is not received from the slave by the master, the master will close the connection event. If a packet is not received from the master by the slave, the slave will close the connection event.

Two consecutive packets received with an invalid CRC match within a connection event shall close the event.

MD bit usage is summarized in Table 4.2.

		Master	
		MD = 0	MD = 1
Slave	MD = 0	Master shall not send another packet, closing the connection event. Slave does not need to listen after sending its packet.	Master may continue the connection event. Slave should listen after sending its packet.
	MD = 1	Master may continue the connection event. Slave should listen after sending its packet.	Master may continue the connection event. Slave should listen after sending its packet.

Table 4.2: MD bit usage for closing connection events

4.5.7 Window Widening

Because of sleep clock accuracies (see Section 4.2.2), there is uncertainty in the slave of the exact timing of the master's anchor point. Therefore the slave is required to re-synchronize to the master's anchor point at each connection event where it listens for the master. If the slave receives a packet from the master regardless of a CRC match, the slave shall update its anchor point.

The slave calculates the time when the master will send the first packet of a connection event (*slaveExpectedAnchorPoint*) taking clock jittering, and in the case of connection setup or a connection parameter update the transmit window, into account. The slave shall also use the masters sleep clock accuracy (*masterSCA*) from the CONNECT_REQ PDU, together with its own sleep clock accuracy (*slaveSCA*) and the anchor point of the last connection event where it received a packet from the master (*timeSinceLastAnchor*) to calculate the time it needs to receive.

The increase in listening time is called the window widening. Assuming the clock inaccuracies are purely given in parts per million (ppm), it is calculated as follows:

$$windowWidening = ((masterSCA + slaveSCA) / 1000000) * timeSinceLastAnchor$$

During connection setup or during a connection parameter update, the slave should listen for windowWidening before the start of the transmit window and until windowWidening after the end of the transmitWindow for the master's anchor point.

At each subsequent connection event, the slave should listen for *windowWidening* before the start of the *slaveExpectedAnchorPoint* and until *windowWidening* after *slaveExpectedAnchorPoint* for the master's anchor point.

The *windowWidening* shall be smaller than ((*connInterval*/2) - T_IFS us). If the *windowWidening* reaches ((connInterval/2) - T_IFS us) in magnitude, the connection should be considered lost.

4.5.8 Data Channel Index Selection

4.5.8.1 Channel Classification

The master's Link Layer shall classify data channels into *used channels* (used for the connection) and *unused channels* (not used for the connection). This is called the channel map. The minimum number of used channels shall be 2.

The Host may provide channel classification information to the Link Layer. The Link Layer may use the information provided by the Host. The slave shall receive the channel map from the master in the CONNECT_REQ PDU. If the master changes the channel map it shall notify the slave as specified in Section 5.1.2.

4.5.8.2 Channel Selection

The channel selection algorithm consists of two stages: calculation of the unmapped channel index followed by mapping this index to a data channel index from the set of *used channels*.

The *unmappedChannel* and *lastUnmappedChannel* are the unmapped channel indices of two consecutive connection events. The *unmappedChannel* is the unmapped channel index for the current connection event. The *lastUnmappedChannel* is the unmapped channel index of the previous connection event. The *lastUnmappedChannel* shall be '0' for the first connection event of a connection.

At the start of a connection event, *unmappedChannel* shall be calculated using the following basic algorithm:

unmappedChannel = (*lastUnmappedChannel* + *hopIncrement*) mod 37

When a connection event closes, the *lastUnmappedChannel* shall be set to the value of the *unmappedChannel*.

If the *unmappedChannel* is a *used channel* according to the channel map, the channel selection algorithm shall use the *unmappedChannel* as the data channel index for the connection event.

If the *unmappedChannel* is an *unused channel* according to the channel map, the *unmappedChannel* shall be re-mapped to one of the used channels in the channel map using the following algorithm:

remappingIndex = *unmappedChannel* mod *numUsedChannels* where *numUsedChannels* is the number of used channels in the channel map.

A remapping table is built that contains all the *used channels* in ascending order, indexed from zero. The *remappingIndex* is then used to select the data channel index for the connection event from the remapping table.

The complete procedure is as shown in Figure 4.14.

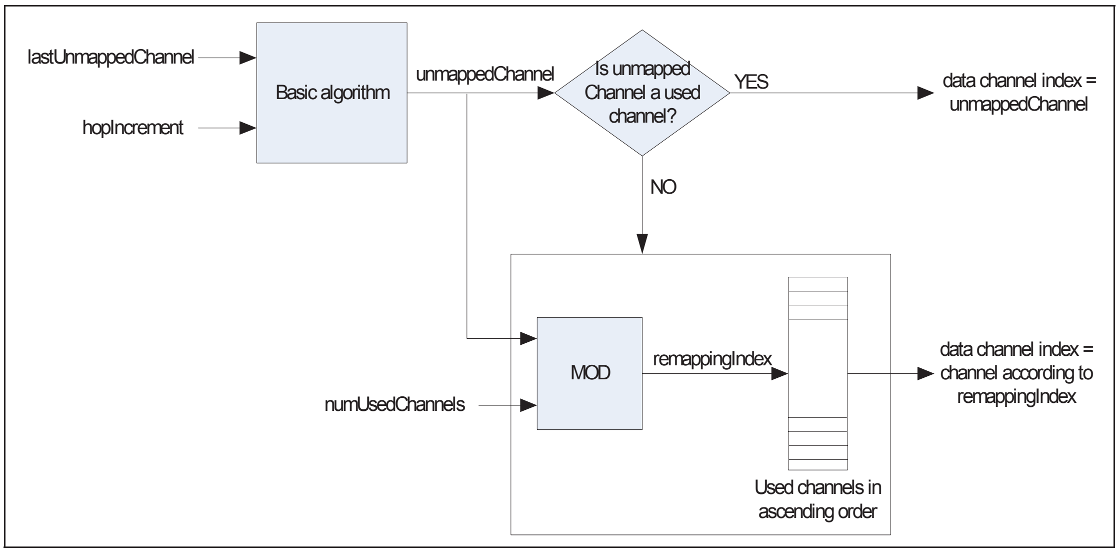
lastUnmappedChannel
hopIncrement
Basic algorithm
unmappedChannel
Is unmapped Channel a used channel?
YES
data channel index = unmappedChannel
NO
MOD
numUsedChannels
remappingIndex
data channel index = channel according to remappingIndex
Used channels in ascending order

Figure 4.14: Block diagram of data channel selection algorithm

4.5.9 Acknowledgement and Flow Control

The Link Layer acknowledgement and flow control scheme shall be used in all Link Layer connections.

For each connection the Link Layer has two parameters, *transmitSeqNum* and *nextExpectedSeqNum*, each one bit in size. *transmitSeqNum* is used to identify packets sent by the Link Layer. *nextExpectedSeqNum* is used by the peer to either acknowledge the last Data Channel PDU sent, or to request resending of the last Data Channel PDU sent.

transmitSeqNum and *nextExpectedSeqNum* shall be set to zero upon entering the Connection State.

A new Data Channel PDU is a Data Channel PDU sent for the first time by the Link Layer. A last Data Channel PDU is a Data Channel PDU that is resent by the Link Layer. When resending a Data Channel PDU, the LLID field, the SN field and the payload of the sent Data Channel PDU shall be equal to those of the last Data Channel PDU sent by the Link Layer.

For each new Data Channel PDU that is sent, the SN bit of the Header shall be set to *transmitSeqNum*. If a Data Channel PDU is resent, then the SN bit shall not be changed.

Upon reception of a Data Channel PDU, the SN bit shall be compared to *next-ExpectedSeqNum*. If the bits are different, then this is a resent Data Channel PDU, and *nextExpectedSeqNum* shall not be changed. If the bits are the same, then this is a new Data Channel PDU, and *nextExpectedSeqNum* may be incremented by one (see Section 4.5.9.1).

When a Data Channel PDU is sent, the NESN bit of the Header shall be set to *nextExpectedSeqNum*.

Upon receiving a Data Channel PDU, if the NESN bit of that Data Channel PDU is the same as *transmitSeqNum*, then the last sent Data Channel PDU has not been acknowledged and shall be resent. If the NESN bit of the Data Channel PDU is different from *transmitSeqNum*, then the last sent Data Channel PDU has been acknowledged, *transmitSeqNum* shall be incremented by one, and a new Data Channel PDU may be sent.

The above process is illustrated in Figure 4.15.

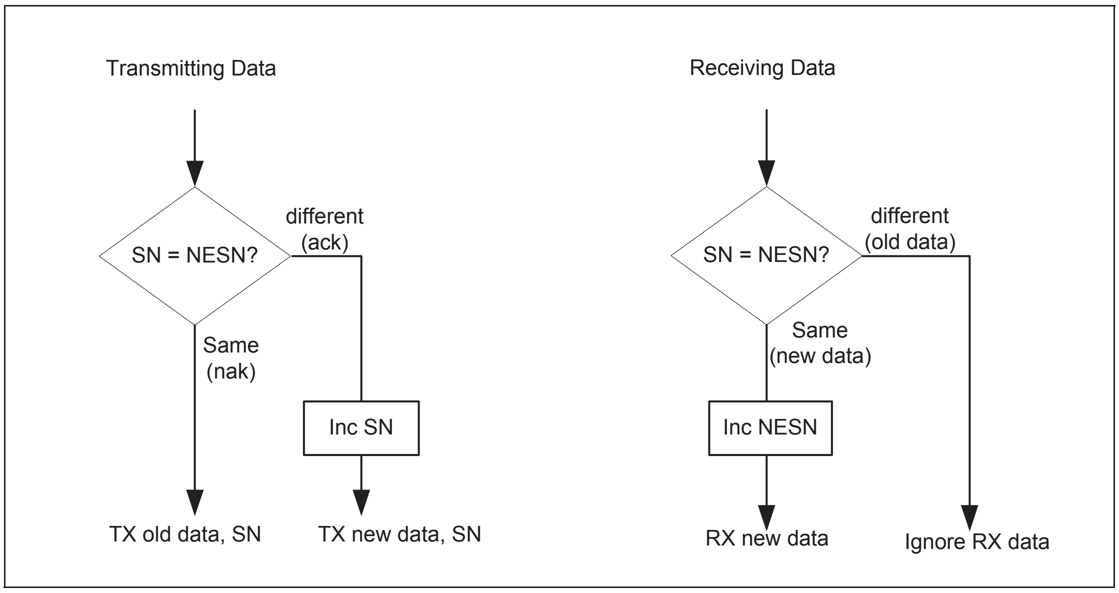

Figure 4.15: Transmit and Receive SN and NESN flow diagram

If a Data Channel PDU is received with an invalid CRC match, nextExpected-SeqNum shall not be changed; this means that the Data Channel PDU will not be acknowledged, causing the peer to resend the Data Channel PDU. Since the received Data Channel PDU has been rejected, the *nextExpectedSeqNum* from the peer device cannot be trusted, and therefore the last sent Data Channel PDU from this device was not acknowledged and must be retransmitted.

SN, NESN and MD bits shall be used from every received Data Channel PDU which has passed the CRC check. The Data Channel PDU payload shall be ignored on every received Data Channel PDU that has the same SN value as the previously received Data Channel PDU.

4.5.9.1 Flow Control

A Link Layer may not update nextExpectedSeqNum for reasons, including, but not limited to, lack of receive buffer space. This will cause the peer to resend the Data Channel PDU at a later time, thus enabling flow control.

4.6 FEATURE SUPPORT

When this information is sent from a Controller to a Host, a bit set to ‘0’ indicates that the Link Layer Feature is not supported in this Controller; a bit set to ‘1’ indicates that the Link Layer Feature is supported in this Controller.

When this information is sent from a Controller to a peer Controller, a bit set to ‘0’ indicates that the Link Layer Feature shall not be used by the Controllers; a bit set to ‘1’ indicates that the Link Layer Feature may be used by the Controllers.

The bit positions for each Link Layer Feature shall be as shown in Table 4.3. This table also shows if these bits are valid for the intended destination. If a bit is shown as not valid, using ‘N’, then this bit shall be ignored upon receipt.

Bit position	Link Layer Feature	Valid from Controller to Host	Valid from Host to Controller	Valid from Controller to Controller
0	LE Encryption	Y	Y	Y
1 – 63	RFU			

Table 4.3: FeatureSet field’s bit mapping to Controller features

4.6.1 LE Encryption

A controller that supports LE Encryption shall support the following sections within this document:

- LL_ENC_REQ (Section 2.4.2.4)
- LL_ENC_RSP (Section 2.4.2.5)
- LL_START_ENC_REQ (Section 2.4.2.6)
- LL_START_ENC_RSP (Section 2.4.2.7)
- Encryption Start Procedure (Section 5.1.3.1)
- Encryption Pause Procedure (Section 5.1.3.2)

5 LINK LAYER CONTROL

The Link Layer Control Protocol (LLCP) is used to control and negotiate aspects of the operation of a connection between two Link Layers. This includes procedures for control of the connection, starting and pausing encryption and other link procedures.

Procedures have specific timeout rules as defined in Section 5.2. The Termination Procedure may be initiated at any time, even if any other Link Layer Control Procedure is currently active. For all other Link Layer Control Procedures, only one Link Layer Control Procedure shall be initiated in the Link Layer at a time per connection per device. A new Link Layer Control Procedure can be initiated only after a previous Link Layer Control Procedure has completed.

The prioritization of LL Control PDUs and LL Data PDUs is implementation specific. For example, a Host cannot assume that pending data will be sent when a termination of the link is requested without waiting for those data PDUs to be completed and indicated to the Host.

5.1 LINK LAYER CONTROL PROCEDURES

5.1.1 Connection Update Procedure

The Link Layer parameters for a connection (*connInterval*, *connSlaveLatency* and *connSupervisionTimeout*) may be updated after entering the Connection State. The master can update the connection parameters by sending an LL_CONNECTION_UPDATE_REQ PDU. The slave shall not send this PDU; the slave may request a change to the connection parameters using the L2CAP LE signaling channel.

The Link Layer of the master shall determine the *connInterval* from the interval range given by the Host ($connInterval_{min}$ and $connInterval_{max}$). The Link Layer shall indicate to the Host the selected interval value.

The Instant field of the LL_CONNECTION_UPDATE_REQ PDU shall be used to indicate the *connEventCount* when the updated parameters shall be applied; this is known as the instant. The master should allow a minimum of 6 connection events that the slave will be listening for before the instant occurs.

The connection interval used before the instant is known as $connInterval_{OLD}$. The connection interval contained in the LL_CONNECTION_UPDATE_REQ PDU and used at the instant and after, is known as $connInterval_{NEW}$.

The connection slave latency used before the instant is known as $connSlaveLatency_{OLD}$. The connection slave latency contained in the LL_CONNECTION_UPDATE_REQ PDU and used at the instant and after, is known as $connSlaveLatency_{NEW}$.

The connection supervision timeout used before the instant is known as *connSupervisionTimeout*$_{OLD}$. The connection supervision timeout contained in the LL_CONNECTION_UPDATE_REQ PDU and used at the instant and after, is known as *connSupervisionTimeout*$_{NEW}$. The connection supervision timer shall be reset at the instant.

For example, the interval between the previous connection event and the connection event at the instant will be *connInterval*$_{OLD}$. The interval between the connection event at the instant and the next connection event will be *connInterval*$_{NEW}$.

When a slave receives an LL_CONNECTION_UPDATE_REQ PDU where (Instant – connEventCount) modulo 65536 is less than 32767 and Instant is not equal to connEventCount, the slave shall listen to all the connection events until it has confirmation that the master has received its acknowledgement of the LL_CONNECTION_UPDATE_REQ PDU or connEventCount equals Instant. The slave shall also listen to the connection event where connEventCount equals Instant and the connection event before it.

When a slave receives an LL_CONNECTION_UPDATE_REQ PDU where (Instant – *connEventCount*) modulo 65536 is greater than or equal to 32767 (because the instant is in the past), the Link Layer of the slave shall consider the connection to be lost, the Link Layer shall exit the Connection State, and shall transition to the Standby State and shall notify the Host.

Note: The comparison of the *connEventCount* and the received Instant field is performed using modulo 65536 math (only values from 0 to 65535 are allowed), to handle the situation when the *connEventCount* field has wrapped.

The master may adjust the anchor point when deciding the timing of the first packet transmitted with new connection parameters. A transmit window is used, as defined in Section 4.5.3. The transmit window starts at *connInterval*$_{OLD}$ + *transmitWindowOffset* after the anchor point of the connection event before the instant. The *transmitWindowOffset* shall be a multiple of 1.25 ms in the range of 0 ms to *connInterval*$_{NEW}$. The *transmitWindowSize* shall be a multiple of 1.25 ms in the range of 1.25 ms to the lesser of 10 ms and (*connInterval*$_{NEW}$ - 1.25 ms).

The master shall start to send the first packet within the transmit window as defined in Section 4.5.3. It is permitted that the master's first packet can extend beyond the transmit window.

The first packet sent at the instant by the master determines the new anchor point for the connection events, and therefore the timings of all future connection events in this connection.

The next connection event anchor point shall be *connInterval*$_{NEW}$ after the connection event anchor point at the Instant. All the normal connection event transmission rules specified in Section 1.4.1, shall apply.

An example of the connection update procedures is shown in Figure 5.1.

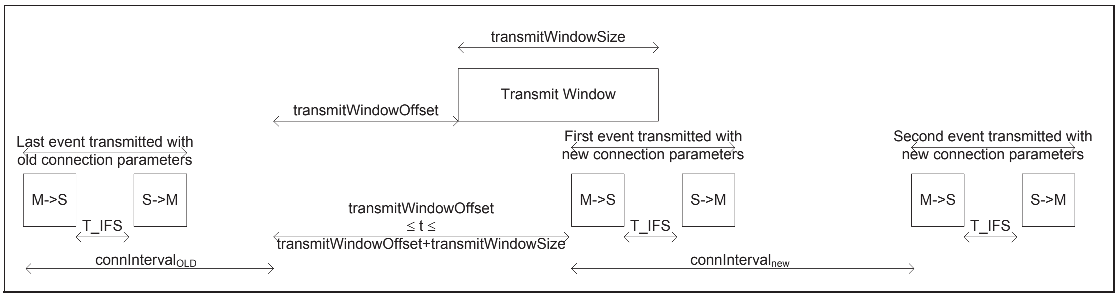
transmitWindowSize
Transmit Window
transmitWindowOffset
Last event transmitted with old connection parameters
First event transmitted with new connection parameters
Second event transmitted with new connection parameters
M->S
S->M
T_IFS
transmitWindowOffset ≤ t ≤ transmitWindowOffset+transmitWindowSize
connInterval_OLD
connInterval_new

Figure 5.1: Connection event timing in the case of connection parameter update

At the start of the transmit window, the Link Layer shall reset $T_{LLconnSupervision}$.

If the Link Layer of the master transmits an LL_CONNECTION_UPDATE_REQ PDU autonomously, for example without being requested to by the Host, the Latency and Timeout parameters shall not be changed and shall remain the same as in the last LL_CONNECTION_UPDATE_REQ or CONNECT_REQ PDU, any of the other parameters (*transmitWindowSize*, *transmitWindowOffset*, *connInterval*, Instant) may be changed within the restrictions given above. Note: Autonomous updates can be used to change the anchor points to allow the master to change the scheduling of the connection due to other activities.

The Link Layer shall notify its Host if any of the three connection parameters have changed. If no connection parameters are changed, the Host would not be notified; this is called an anchor point move.

The procedure is complete when the instant has passed, and the new connection event parameters have been applied.

5.1.2 Channel Map Update Procedure

The Link Layer parameter for channel map (*channelMap*) may be updated after entering the Connection State. The master can update the channel map by sending an LL_CHANNEL_MAP_REQ PDU. The slave shall not send this PDU.

The Instant field of the LL_CHANNEL_MAP_REQ PDU shall be used to indicate the *connEventCount* when the *channelMap*$_{NEW}$ shall be applied; this is known as the instant. The master should allow a minimum of 6 connection events that the slave will be listening for before the instant occurs.

The channel map used before the instant is known as *channelMap*$_{OLD}$. The channel map contained in the LL_CHANNEL_MAP_REQ PDU and used at the instant and after, is known as *channelMap*$_{NEW}$.

When a slave receives an LL_CHANNEL_MAP_REQ PDU where (Instant – *connEventCount*) modulo 65536 is less than 32767, the slave shall listen to all the connection events until it has confirmation that the master has received its acknowledgement of the LL_CHANNEL_MAP_REQ PDU or *connEventCount* equals Instant.

When a slave receives an LL_CHANNEL_MAP_REQ where (Instant – *connEventCount*) modulo 65536 is greater than or equal to 32767 (because the instant is in the past), the Link Layer of the slave shall consider the connection to be lost and shall notify the Host.

Note: The comparison of the *connEventCount* and the received Instant field is performed using modulo 65536 math (only values from 0 to 65535 are allowed), to cope with the cases when the *connEventCount* field has wrapped.

When *connEventCount* is equal to the Instant field, the *channelMap*$_{NEW}$ shall be the current *channelMap*. The *lastUnmappedChannel* shall not be reset. If the *unmappedChannel* is an unused channel, then the *channelMap*$_{NEW}$ will be used when remapping. The only parameter that changes is the *channelMap*.

For example:

At connection set-up:

- initial *channelMap*$_{OLD}$: 0x1FFFFFFFFF (i.e., all channels enabled)
- initial *hopIncrement*: 10 (decimal)

An LL_CHANNEL_MAP_REQ PDU with the following parameters is then issued:

- Instant: 100 (decimal). Assume that no connection event count wrap-around occurred since the start of the connection.
- *channelMap*$_{NEW}$: 0x1FFFFFF7FF (i.e. all channels enabled except channel 11)

Channels used:

- *connEventCount* 99 --> data channel index 1 (*channelMap*$_{OLD}$)
- *connEventCount* 100 --> data channel index 12 (remapped from 11) (*channelMap*$_{NEW}$)
- *connEventCount* 101 --> data channel index 21 (*channelMap*$_{NEW}$)

The procedure is complete when the instant has passed, and the new channel map has been applied.

5.1.3 Encryption Procedure

The Link Layer, upon request from the Host, can enable the encryption of packets after entering the Connection State.

If the connection is not encrypted, the Link Layer shall only use the encryption start procedure.

If the connection is encrypted, the Link Layer shall first use the encryption pause procedure followed by the encryption start procedure.

5.1.3.1 Encryption Start Procedure

To enable encryption, two parameters must be exchanged, IV and SKD. Both are composed of two parts, a master part and a slave part, and exchanged in LL_ENC_REQ and LL_ENC_RSP PDUs. After these are exchanged, and the Host has notified the Link Layer of the Long Term Key to be used on this connection, encryption can be started using a three way handshake, using LL_START_ENC_REQ and LL_START_ENC_RSP PDUs.

To start encryption, the Link Layer of the master shall generate the master's part of the initialization vector (IVm) and the master's part of the session key diversifier (SKDm).

The Link Layer of the master shall finalize the sending of the current Data Channel PDU and may finalize the sending of additional Data Channel PDUs queued in the Controller. After these Data Channel PDUs are acknowledged, the Link Layer of the master shall only send Empty PDUs or LL_ENC_REQ, LL_START_ENC_REQ, LL_START_ENC_RSP, LL_TERMINATE_IND or LL_REJECT_IND PDUs.

The Link Layer of the master shall then send an LL_ENC_REQ PDU; the Rand and EDIV fields are provided by the Host.

If encryption is not supported by the Link Layer of the slave, the Link Layer of the slave shall send an LL_REJECT_IND PDU with the error code set to "Unsupported Remote Feature / Unsupported LMP Feature" (0x1A). The Link Layer of the master receiving the LL_REJECT_IND PDU shall notify the Host. The Link Layer of the master can now send LL Data Packets and LL Control Packets; these packets will not be encrypted. This procedure is complete in the master when the master receives the LL_REJECT_IND PDU from the slave. The procedure is complete in the slave when the acknowledgement for the LL_REJECT_IND PDU is received from the master.

Otherwise, when the Link Layer of the slave receives an LL_ENC_REQ PDU it shall generate the slave's part of the initialization vector (IVs) and the slave's part of the session key diversifier (SKDs), and notify the Host with the Rand and EDIV fields.

The Link Layer of the slave shall finalize the sending of the current Data Channel PDU and may finalize the sending of additional Data Channel PDUs queued in the Controller. After these Data Channel PDUs are acknowledged, the Link Layer of the slave is only allowed to send Empty PDUs or LL_ENC_RSP, LL_START_ENC_REQ, LL_START_ENC_RSP, LL_TERMINATE_IND or LL_REJECT_IND PDUs.

The Link Layer of the slave shall then send an LL_ENC_RSP PDU.

Each Link Layer shall combine the initialization vector parts and session key diversifier parts in the following manner:

SKD = SKDm || SKDs

IV = IVm || IVs

The SKDm is concatenated with the SKDs. The least significant octet of SKDm becomes the least significant octet of SKD. The most significant octet of SKDs becomes the most significant octet of SKD.

The IVm is concatenated with the IVs. The least significant octet of IVm becomes the least significant octet of IV. The most significant octet of IVs becomes the most significant octet of IV.

The Long Term Key is provided by the Host to the Link Layer in the master and slave, and one of the following three actions shall occur:

- If this procedure is being performed after a Pause Encryption Procedure, and the Host does not provide a Long Term Key, the slave shall perform the Termination Procedure with the error code “PIN or key Missing.”
- If the Host does not provide a Long Term Key, either because the event to the Host was masked out or if the Host indicates that a key is not available, the slave shall send an LL_REJECT_IND PDU with the error code set to “PIN or key Missing.” Upon receiving an LL_REJECT_IND PDU, the Link Layer shall notify the Host. The Link Layer can now send LL Data PDUs and LL Control PDUs; these packets will not be encrypted. This procedure is complete in the master when the master receives the LL_REJECT_IND PDU from the slave. The procedure is completed in the slave when the acknowledgement has been received for the LL_REJECT_IND PDU from the master.
- If the Host does provide a Long Term Key, the Link Layer of the slave shall respond to the LL_ENC_REQ PDU from the master with an LL_ENC_RSP PDU. The Link Layer shall also calculate *sessionKey* using the encryption engine with LTK as the key, and SKD as the plain text input. *sessionKey* shall be set to the output of the encryption engine.

The *sessionKey* shall be used as the key for the encryption engine for all encrypted packets.

After *sessionKey* has been calculated, the Link Layer of the slave shall send an LL_START_ENC_REQ PDU. This packet shall be sent unencrypted, and the Link Layer shall be set up to receive an encrypted packet in response.

When the Link Layer of the master receives an LL_START_ENC_REQ PDU it shall send an LL_START_ENC_RSP PDU. This PDU shall be sent encrypted and set up to receive encrypted.

When the Link Layer of the slave receives an LL_START_ENC_RSP PDU it shall transmit an LL_START_ENC_RSP PDU. This packet shall be sent encrypted.

When the Link Layer of the master receives the LL_START_ENC_RSP PDU, the connection is encrypted. The Link Layer can now send LL Data PDUs and LL Control PDUs; these PDUs will be encrypted.

The Link Layers shall notify the Hosts that the connection is encrypted.

The procedure is complete in the master when the master receives the LL_START_ENC_RSP PDU from the slave. The procedure is complete in the slave when the slave receives the LL_START_ENC_RSP PDU from the master.

5.1.3.2 Encryption Pause Procedure

To enable a new encryption key to be used without disconnecting the link, encryption must be disabled and then enabled again. During the pause, data PDUs shall not be sent unencrypted to protect the data.

The Link Layer of the master shall finalize the sending of the current Data Channel PDU and may finalize the sending of additional Data Channel PDUs queued in the Controller. After these Data Channel PDUs are acknowledged, the Link Layer of the master shall only send Empty PDUs or LL_PAUSE_ENC_REQ or LL_TERMINATE_IND PDUs.

The Link Layer of the master shall then send an LL_PAUSE_ENC_REQ PDU.

When the Link Layer of the slave receives an LL_PAUSE_ENC_REQ PDU it shall finalize the sending of the current Data Channel PDU and may finalize the sending of additional Data Channel PDUs queued in the Controller. After these Data Channel PDUs are acknowledged, the Link Layer of the slave is only allowed to send Empty PDUs or LL_PAUSE_ENC_RSP or LL_TERMINATE_IND PDUs.

The Link Layer of the slave shall then send an LL_PAUSE_ENC_RSP PDU. This packet shall be sent encrypted, and Link Layer shall be set up to receive unencrypted.

When the Link Layer of the master receives an LL_PAUSE_ENC_RSP PDU it shall set up to send and receive unencrypted. It shall then send an LL_PAUSE_ENC_RSP PDU to the slave unencrypted.

When the Link Layer of the slave receives an LL_PAUSE_ENC_RSP PDU it shall set up to also send unencrypted.

The encryption start procedure shall now be used to re-enable encryption using a new session key.

5.1.4 Feature Exchange Procedure

The Link Layer parameter for the current supported feature set (featureSet) may be exchanged after entering the Connection State. The master can initiate this procedure with an LL_FEATURE_REQ PDU, and the slave responds with an LL_FEATURE_RSP PDU. The slave cannot initiate this procedure.

The featureSet information may be cached. A Link Layer should not request this information on every connection if the information has been cached for this device. Cached information for a device may not be authoritative, and therefore an implementation must be able to accept the LL_UNKNOWN_RSP PDU if use of a feature is attempted that is not currently supported or used by the peer.

featureSet_M is the feature capabilities of the Link Layer of the master. When the Link Layer of the master sends an LL_FEATURE_REQ PDU the FeatureSet field shall be set to featureSet_M.

featureSet_S is the feature capabilities of the Link Layer of the Slave.

The featureSet_{USED} is the logical AND of featureSet_M and featureSet_S. When the Link Layer of the slave sends an LL_FEATURE_RSP PDU the FeatureSet field shall be set to featureSet_{USED}.

The Link Layer of the master sends an LL_FEATURE_REQ PDU. This can be sent on request from the Host or autonomously.

When the Link Layer of the slave receives an LL_FEATURE_REQ PDU it shall send an LL_FEATURE_RSP PDU. The Link Layer of the slave shall only use procedures that are indicated in featureSet_{USED}.

When the Link Layer of the master receives an LL_FEATURE_RSP PDU it shall only use procedures that are indicated in featureSet_{USED}.

An example of feature exchange is shown in Figure 5.2.

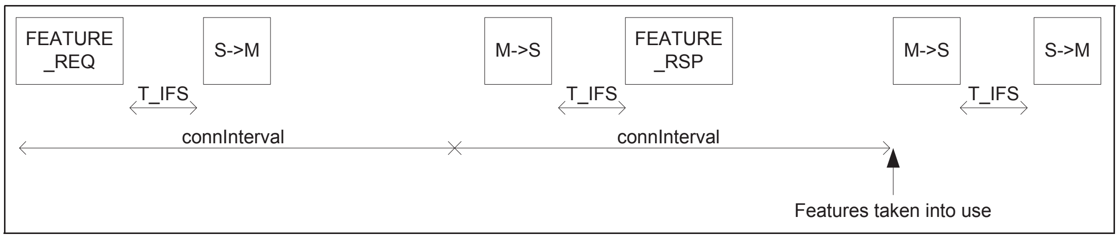

Figure 5.2: Feature Exchange Procedure

The procedure is complete in the master when the master receives the LL_FEATURE_RSP PDU from the slave.

5.1.5 Version Exchange

The Link Layer parameters for version information (*companyID*, *subVerNum*, *linkLayerVer,* as defined in Section 2.4.2.13) may be exchanged after entering the Connection State. Either the Link Layer of the master or slave can initiate this procedure by sending an LL_VERSION_IND PDU. This procedure should be used when requested by the Host. This procedure may be initiated autonomously by the Link Layer.

The Link Layer shall only queue for transmission a maximum of one LL_VERSION_IND PDU during a connection.

If the Link Layer receives an LL_VERSION_IND PDU and has not already sent an LL_VERSION_IND then the Link Layer shall send an LL_VERSION_IND PDU to the peer device.

If the Link Layer receives an LL_VERSION_IND PDU and has already sent an LL_VERSION_IND PDU then the Link Layer shall not send another LL_VERSION_IND PDU to the peer device.

The procedure has completed when an LL_VERSION_IND PDU has been received from the peer device.

5.1.6 Termination Procedure

This procedure is used for voluntary termination of a connection while in the Connection State. Voluntary termination occurs when the Host requests the Link Layer to terminate the connection. Either the Link Layer of the master or slave can initiate this procedure by sending an LL_TERMINATE_IND PDU. The termination procedure is not used in the event of the loss of the connection, for example after link supervision timeout or after a procedure timeout.

The Link Layer shall start a timer, $T_{terminate}$, when the LL_TERMINATE_IND PDU has been queued for transmission. The initiating Link Layer shall send LL_TERMINATE_IND PDUs until an acknowledgement is received or until the timer, $T_{terminate}$, expires. The initial value for $T_{terminate}$ shall be set to value of the *connSupervisionTimeout*.

When the Link Layer receives an LL_TERMINATE_IND PDU it shall send the acknowledgement, exit the Connection State and shall transition to the Standby State.

The procedure has completed when the acknowledgement has been received.

5.2 PROCEDURE RESPONSE TIMEOUT

This section specifies procedure timeout rules that shall be applied to all the Link Layer control procedures specified in Section 5.1, except for the Connection Update and Channel Map Update procedures for which there are no timeout rules.

To be able to detect a non-responsive Link Layer Control Procedure, both the master and the slave shall use a procedure response timeout timer, T_{PRT}. Upon the initiation of a procedure, the procedure response timeout timer shall be reset and started.

Each LL Control PDU that is queued for transmission resets the procedure response timeout timer.

When the procedure completes, the procedure response timeout timer shall be stopped.

If the procedure response timeout timer reaches 40 seconds, the connection is considered lost. The Link Layer exits the Connection State and shall transition to the Standby State. The Host shall be notified of the loss of connection.

SAMPLE DATA

This part of the specification contains sample data for Bluetooth low energy. All sample data are provided for reference purpose only. They can be used to check the behavior of an implementation and avoid misunderstandings.

CONTENTS

1 ENCRYPTION SAMPLE DATA

This section contains sample data for the Low Energy encryption process.

The following scenario describes the start of encryption, followed by the transfer of an encrypted data channel data packet in each direction. It describes:

- how the derived values are calculated (fixed values are given in red)
- which HCI command and events are exchanged (given in italic)
- which LL messages are exchanged over the air (given in green).

Note: CRCs are not shown because they depend on a random CRC init value. Scrambling is disabled.

The following parameters are set to the fixed values below:

```
LTK = 0x4C68384139F574D836BCF34E9DFB01BF (MSO to LSO)
EDIV = 0x2474 (MSO to LSO)

RAND = 0xABCDEF1234567890 (MSO to LSO)
SKDm = 0xACBDCEDFE0F10213 (MSO to LSO)
SKDs = 0x0213243546576879 (MSO to LSO)
IVm = 0xBADCAB24 (MSO to LSO)
IVs = 0xDEAFBABE (MSO to LSO)

HCI_LE_Start_Encryption (length 0x1C) – master HCI command
  Pars (LSO to MSO)    00 08 90 78 56 34 12 ef cd ab 74 24 bf 01 fb 9d 4e f3 bc 36
d8 74 f5 39 41 38 68 4c
    Handle (2-octet value MSO to LSO) 0x8000
    Random (8-octet value MSO to LSO) 0xabcdef1234567890
    Encrypted Diversifier (2-octet value MSO to LSO) 0x2474
    Long Term Key (16-octet value MSO to LSO) 0x4c68384139f574d836bcf34e9dfb01bf

SKDm (LSO to MSO)    :0x13:0x02:0xF1:0xE0:0xDF:0xCE:0xBD:0xAC:

IVm (LSO to MSO)     :0x24:0xAB:0xDC:0xBA

LL_ENC_REQ 03 17 03 90 78 56 34 12 ef cd ab 74 24 13 02 f1 e0 df ce bd ac 24
ab dc ba
  Length 0x17
  Control Type 0x03
  Rand 90 78 56 34 12 ef cd ab
  EDIV 74 24
  SKDm 13 02 f1 e0 df ce bd ac
  IVm 24 ab dc ba


SKDs (LSO to MSO)    :0x79:0x68:0x57:0x46:0x35:0x24:0x13:0x02:

IVs (LSO to MSO)     :0xBE:0xBA:0xAF:0xDE
```

```
LL_ENC_RSP 0b 0d 04 79 68 57 46 35 24 13 02 be ba af de
  Length 0x0D
  Control Type 0x04
  SKDs 79 68 57 46 35 24 13 02
  IVs be ba af de

IV = IVm || IVs
IV (LSO to MSO)      :0x24:0xAB:0xDC:0xBA:0xBE:0xBA:0xAF:0xDE

HCI_Long_Term_Key_Requested(length 0x0D) – slave event
  Pars (LSO to MSO)05 01 08 90 78 56 34 12 ef cd ab 74 24
    LE_Event_Code 0x05
    Handle (2-octet value MSO to LSO) 0x0801
    Random (8-octet value MSO to LSO) 0xabcdef1234567890
    Encrypted Diversifier (2-octet value MSO to LSO) 0x2474

HCI_LE_Long_Term_Key_Request_Reply (length 0x12) – slave command
  Pars (LSO to MSO)    01 08 bf 01 fb 9d 4e f3 bc 36 d8 74 f5 39 41 38 68 4c
    Handle (2-octet value MSO to LSO) 0x0801
    Key (16-octet value MSO to LSO) 0x4C68384139F574D836BCF34E9DFB01BF


SKD = SKDm || SKDs
SKD (LSO to MSO)
:0x13:0x02:0xF1:0xE0:0xDF:0xCE:0xBD:0xAC:0x79:0x68:0x57:0x46:0x35:0x24:0x13:0x02:

SK = Encrypt(LTK, SKD)
SK (LSO to MSO)
:0x66:0xC6:0xC2:0x27:0x8E:0x3B:0x8E:0x05:0x3E:0x7E:0xA3:0x26:0x52:0x1B:0xAD:0x99:

LL_START_ENC_REQ 07 01 05
  Length 0x0D
  Control Type 0x05

LL_START_ENC_RSP1 0f 05 9f cd a7 f4 48
  Length 0x05
  Control Type Encrypted:0x9F Clear:0x06
  MIC (32-bit value MSO to LSO) 0xCDA7F448 (note that MICs are sent MSO first on
the air)

LL_START_ENC_RSP2 07 05 a3 4c 13 a4 15
  Length 0x05
  Control Type Encrypted:0xA3 Clear:0x06
  MIC (32-bit value MSO to LSO) 0x4C13A415

HCI_ACL_Data_Packet Master host to controller
  00 08 1b 00 17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33 34 35 36
37 38 39 30
    Handle (12-bit value MSO to LSO) 0x8000
    Data Total Length (16-bit value MSO to LSO) 0x001B (27 dec)
    Data (LSO to MSO) 17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33
34 35 36 37 38 39 30

LL_DATA1 0e 1f 7a 70 d6 64 15 22 6d f2 6b 17 83 9a 06 04 05 59 6b d6 56 4f 79 6b 5b
9c e6 ff 32 f7 5a 6d 33
  Length 0x1F (i.e. 27 + 4 = 31 dec)
  Data (LSO to MSO)
```

```
    Clear     17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31 32 33 34 35 36
37 38 39 30
    Encrypted 7a 70 d6 64 15 22 6d f2 6b 17 83 9a 06 04 05 59 6b d6 56 4f 79 6b 5b
9c e6 ff 32
  MIC (32-bit value MSO to LSO)   0xF75A6D33

HCI_ACL_Data_Packet Slave host to controller
  01 08 1b 00 17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d
4e 4f 50 51
    Handle (12-bit value MSO to LSO) 0x8001
    Data Total Length (16-bit value MSO to LSO) 0x001B (27 dec)
    Data (LSO to MSO) 17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a
4b 4c 4d 4e 4f 50 51


LL_DATA2 06 1f f3 88 81 e7 bd 94 c9 c3 69 b9 a6 68 46 dd 47 86 aa 8c 39 ce 54 0d 0d
ae 3a dc df 89 b9 60 88
  Length 0x1F (i.e. 27 + 4 = 31 dec)
  Data (LSO to MSO)
    Clear     17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d
4e 4f 50 51
    Encrypted f3 88 81 e7 bd 94 c9 c3 69 b9 a6 68 46 dd 47 86 aa 8c 39 ce 54 0d 0d
ae 3a dc df
  MIC (32-bit value MSO to LSO)   0x89B96088
```

1.1 ENCRYPT COMMAND

```
HCI_LE_Encrypt (length 0x20) – command
  Pars (LSO to MSO) bf 01 fb 9d 4e f3 bc 36 d8 74 f5 39 41 38 68 4c 13 02 f1 e0 df
ce bd ac 79 68 57 46 35 24 13 02
    Key (16-octet value MSO to LSO):             0x4C68384139F574D836BCF34E9DFB01BF
    Plaintext_Data (16-octet value MSO to LSO): 0x0213243546576879acbdcedfe0f10213

HCI_Command_Complete (length 0x14) – event
  Pars (LSO to MSO) 02 17 20 00 66 c6 c2 27 8e 3b 8e 05 3e 7e a3 26 52 1b ad 99
    Num_HCI_Commands_Packets: 0x02
    Command_Opcode (2-octet value MSO to LSO): 0x2017
    Status: 0x00
    Encrypted_Data (16-octet value MSO to LSO): 0x99ad1b5226a37e3e058e3b8e27c2c666
```

1.2 DERIVATION OF THE MIC AND ENCRYPTED DATA

All B/X/A/S values below follow notation of AES-CCM: MSbyte to LSbyte & msbit to lsbit.

```
IV = DEAFBABEBADCAB24
SK = 99AD1B5226A37E3E058E3B8E27C2C666

1.START_ENC_RSP1 (packet 0, M --> S)
----------------

B0 = 49000000008024ABDCBABEBAAFDE0001
B1 = 00010300000000000000000000000000
B2 = 06000000000000000000000000000000
```

```
X1 = 712eaaaae60603521d245e50786eefe4
X2 = debc43782a022675fca0aa6f0854f1ab
X3 = 6399913fede5fa111bdb993bbfb9be06
=> MIC = 6399913f

A0 = 01000000008024ABDCBABEBAAFDE0000
A1 = 01000000008024ABDCBABEBAAFDE0001

S0 = ae3e6577f64a8f25408c9c10d53acf8e
S1 = 99190d88f4aa1b60b97ecfe6f5fee777

So, encrypted packet payload = 9F
    encrypted MIC = CDA7F448

Which results in the following packet:

LL_START_ENC_RSP1 - 0f 05 9f cd a7 f4 48
  Length: 05
  Control Type:
    Clear:     06
    Encrypted: 9f
  MIC: CD A7 F4 48


2.START_ENC_RSP2 (packet 0, S --> M)
----------------

B0 = 49000000000024ABDCBABEBAAFDE0001
B1 = 00010300000000000000000000000000
B2 = 06000000000000000000000000000000

X1 = ddc86e3094f0c29cf341ef4c2c1e0088
X2 = fe960f5c93fba45a53959842ea8a0c0a
X3 = db403db3a32f39156faf6a6b472e1010
=> MIC = db403db3

A0 = 01000000000024ABDCBABEBAAFDE0000
A1 = 01000000000024ABDCBABEBAAFDE0001

S0 = 975399a66acdc39124886930d7bca95f
S1 = a5add4127b2f43788ddc9cd86b0b89d2

So, encrypted packet payload = A3
    encrypted MIC = 4c13a415

Which results in the following packet:

LL_START_ENC_RSP2 07 05 a3 4c 13 a4 15
  Length: 05
  Control Type:
    Clear:     06
    Encrypted: A3
  MIC: 4c 13 a4 15


3. Data packet1 (packet 1, M --> S)
---------------

B0 = 49010000008024ABDCBABEBAAFDE001B
B1 = 00010200000000000000000000000000
B2 = 1700636465666768696A6B6C6D6E6F70
B3 = 71313233343536373839300000000000

X1 = 7c688612996de101f3eacb68b443969c
X2 = e3f1ef5c30161c0a9ec07274a0757fc8
X3 = e7e346f5b7c8a6072890a60dcf4ec20a
X4 = 3db113320b182f9fed635db14cac2df0
=> MIC = 3db11332

A0 = 01010000008024ABDCBABEBAAFDE0000
A1 = 01010000008024ABDCBABEBAAFDE0001
A2 = 01010000008024ABDCBABEBAAFDE0002

S0 = caeb7e017296dd2fa9a2ce789179501a
S1 = 6d70b50070440a9a027de8f66b6a6a29
S2 = 1ae7647c4d5e6dabdec602404c302341


So, encrypted packet payload =
7A70D66415226DF26B17839A060405596BD6564F796B5B9CE6FF32
    encrypted MIC = F75A6D33


which results in the following packet:

LL_DATA1 0E 1F 7A 70 D6 64 15 22 6D F2 6B 17 83 9A 06 04 05 59 6B D6
56 4F 79 6B 5B 9C E6 FF 32 F7 5A 6D 33
  Length: 1F
  Data:
    Clear:     17 00 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f 70 71 31
32 33 34 35 36 37 38 39 30
    Encrypted: 7A 70 D6 64 15 22 6D F2 6B 17 83 9A 06 04 05 59 6B D6
56 4F 79 6B 5B 9C E6 FF 32
  MIC: F7 5A 6D 33

4. Data packet2 (packet 1, S --> M)
---------------

B0 = 49010000000024ABDCBABEBAAFDE001B
B1 = 00010200000000000000000000000000
B2 = 17003736353433323130414243444546
```

```
B3 = 4748494A4B4C4D4E4F50510000000000

X1 = 714234d50d6f1da5663be3e78460ad87
X2 = 96df1d97959e6176ac215c7baf90c674
X3 = 6cc52c3dcecdc2fa81eb347887960673
X4 = a776a26be617366496c391e36f6374a1 => MIC = a776a26b

A0 = 01010000000024ABDCBABEBAAFDE0000
A1 = 01010000000024ABDCBABEBAAFDE0001
A2 = 01010000000024ABDCBABEBAAFDE0002

S0 = 2ecfc2e31e01875653c0f306fc7bfb96
S1 = e488b6d188a0faf15889e72a059902c0
S2 = edc470841f4140e0758c8e8f708399bd


So, encrypted packet payload =
F38881E7BD94C9C369B9A66846DD4786AA8C39CE540D0DAE3ADCDF
    encrypted MIC = 89B96088


Which results in the following packet:

LL_DATA2 06 1F F3 88 81 E7 BD 94 C9 C3 69 B9 A6 68 46 DD 47 86 AA 8C
39 CE 54 0D 0D AE 3A DC DF 89 B9 60 88
  Length: 1F
  Data:
    Clear:      17 00 37 36 35 34 33 32 31 30 41 42 43 44 45 46 47 48
49 4a 4b 4c 4d 4e 4f 50 51
    Encrypted: F3 88 81 E7 BD 94 C9 C3 69 B9 A6 68 46 DD 47 86 AA 8C
39 CE 54 0D 0D AE 3A DC DF

MIC: 89 B9 60 88
```

MESSAGE SEQUENCE CHARTS

Examples of message sequence charts showing the interactions of the Host Controller Interface with the Link Layer.

CONTENTS

1 INTRODUCTION

This section shows typical interactions between Host Controller Interface (HCI) Commands and Events and the Link Layer (LL). It focuses on the message sequence charts (MSCs) for the procedures specified in “Bluetooth Host Controller Interface Functional Specification” with regard to Link Layer Control Procedures from “Link Layer”. This section illustrates only the most useful scenarios; it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or host interface. In all message sequence charts it is assumed that all events are not masked, so the Host Controller will not filter out any events.

The sequence of messages in these message sequence charts is for illustrative purposes. The messages may be sent in a different order where allowed by the Link Layer or HCI sections. If any of these charts differ with text in the Link Layer or HCI sections, the text in those sections shall be considered normative. This section is informative.

1.1 NOTATION

The notation used in the message sequence charts (MSCs) consists of ovals, elongated hexagons, boxes, lines, and arrows. The vertical lines terminated on the top by a shadow box and at the bottom by solid oval indicate a protocol entity that resides in a device. MSCs describe interactions between these entities and states those entities may be in.

The following symbols represent interactions and states:

Oval	**Defines the context for the message sequence chart**
Hexagon	Indicates a condition needed to start the transactions below this hexagon. The location and width of the Hexagon indicates which entity or entities make this decision.
Box	Replaces a group of transactions. May indicate a user action, or a procedure in the baseband.
Dashed Box	Optional group of transactions.
Solid Arrow	Represents a message, signal or transaction. Can be used to show Link Layer and HCI traffic. Some baseband packet traffic is also shown. These are prefixed by BB followed by either the type of packet, or an indication that there is an ACK signal in a packet.
Dashed Arrow	Represents a optional message, signal or transaction. Can be used to show Link Layer and HCI traffic.

1.2 CONTROL FLOW

Some message sequences are split into several charts. These charts are marked in sequence with different step numbers with multiple paths through with optional letters after the step numbers. Numbers indicate normal or required ordering. The letters represent alternative paths. For example, Step 4 is after Step 3, and Step 5a could be executed instead of Step 5b.

1.3 EXAMPLE MSC

The protocol entities represented in the example shown in Figure 1.1 illustrate the interactions of two devices named A and B. Note that each device includes a Host and a LL entity in this example. Other MSCs in this section may show the interactions of more than two devices.

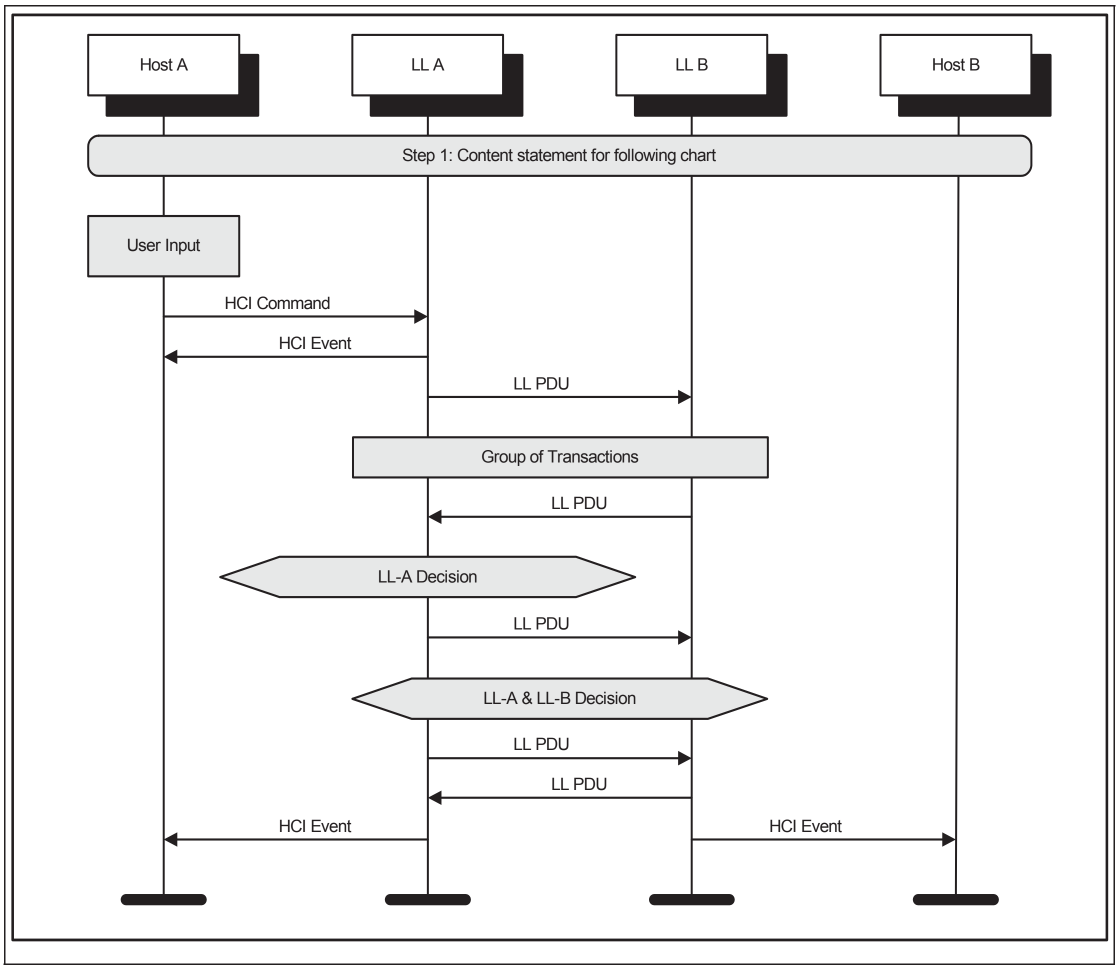

Figure 1.1: Example MSC

2 STANDBY STATE

2.1 INITIAL SETUP

To perform initial setup of a LE Controller, the following sequence of actions may be required.

First, the host would wait for the Controller to indicate the number of HCI Command Packets the Host is currently allowed to send using a Command Complete event on a No OPeration command opcode. Then it would reset the Controller to a known state. Then it needs to read the local supported features to check that low energy is supported on this Controller. It would then set the event mask and LE event mask to enable the events that it wants the Controller to generate to the Host. Next, it will check the buffers that are available for data flow, using the Read Buffer Size and LE Read Buffer Size commands. Then it would read the locally supported LE features and select the features that it wishes to use. Finally, it will read the public device address if the Controller has one (see Figure 2.1).

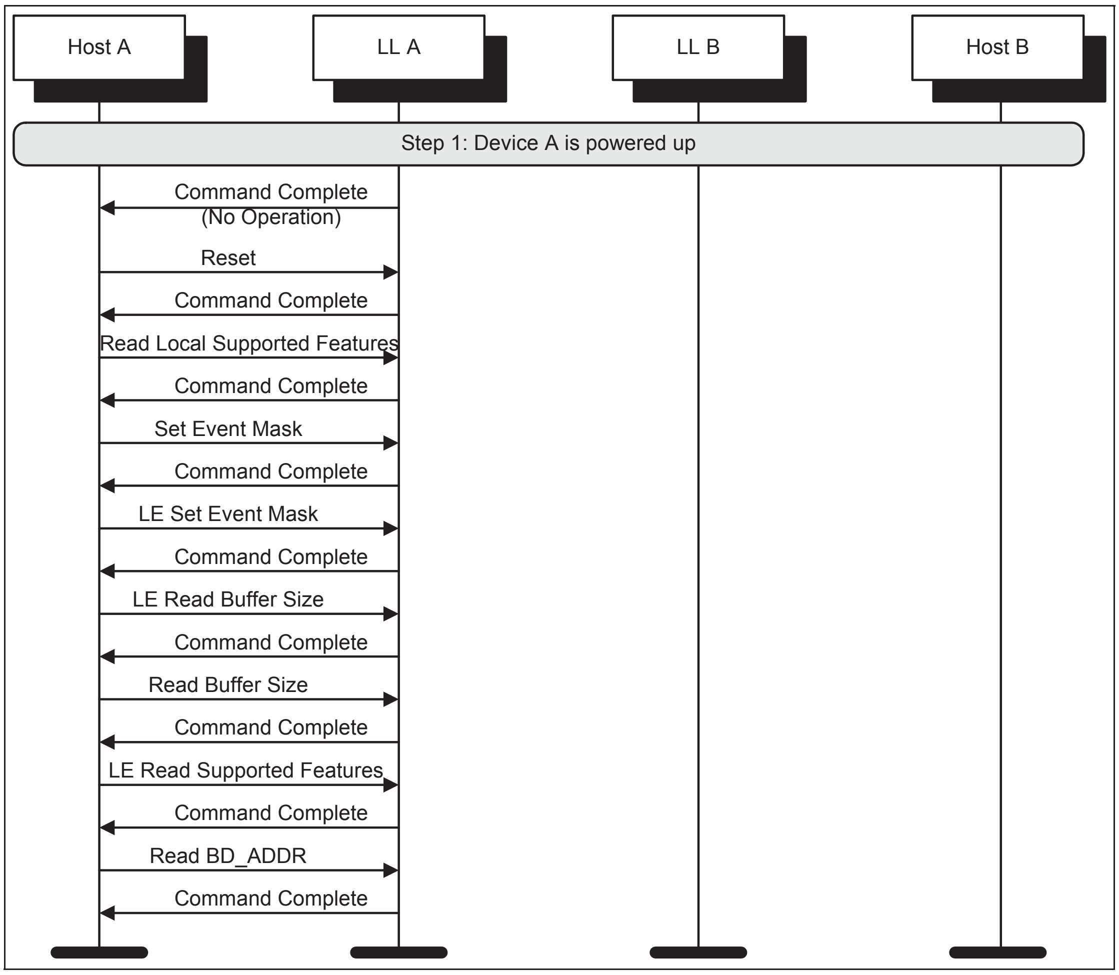

Figure 2.1: Initial Setup

2.2 RANDOM DEVICE ADDRESS

A device may use a random device address, but this address has to be configured before being used during advertising, scanning or initiating (see Figure 2.2).

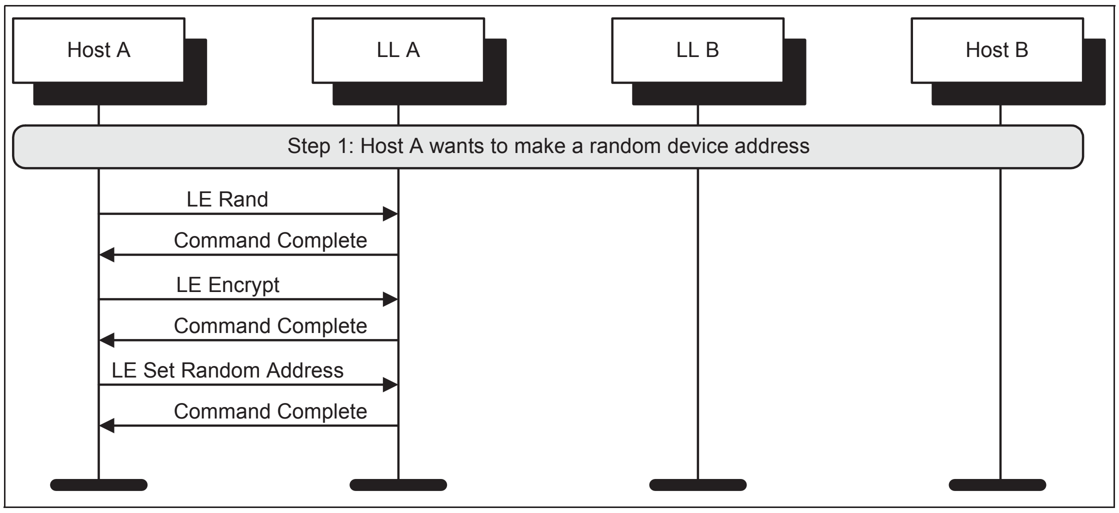

Figure 2.2: Random Device Address

2.3 WHITE LISTS

Before advertising, scanning or initiating can use white lists, the white list may be cleared and devices added in as required (see Figure 2.3).

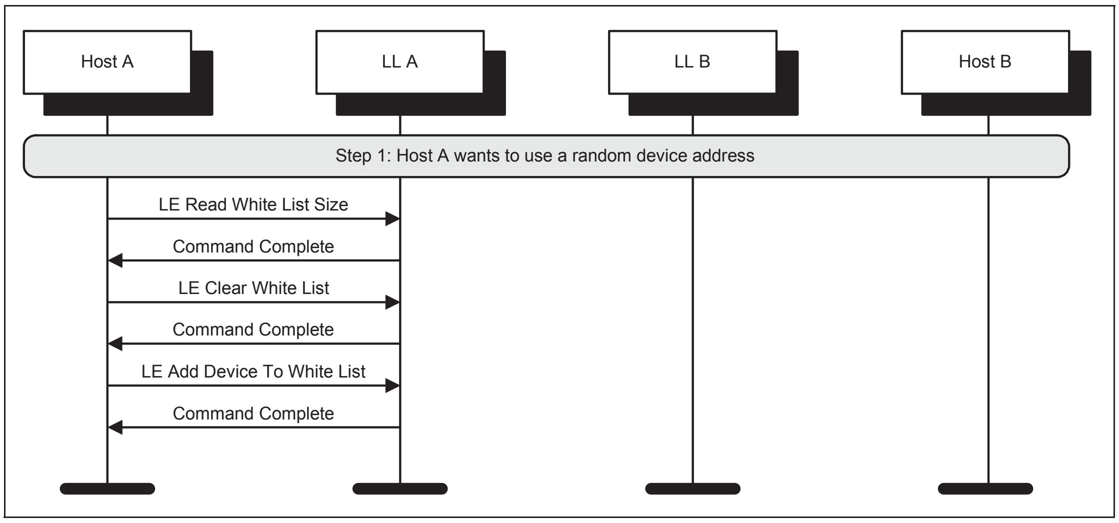

Figure 2.3: White Lists

3 ADVERTISING STATE

3.1 UNDIRECTED ADVERTISING

A device may enter the Advertising State by enabling advertising. It should also configure the advertising parameters before doing this (see Figure 3.1).

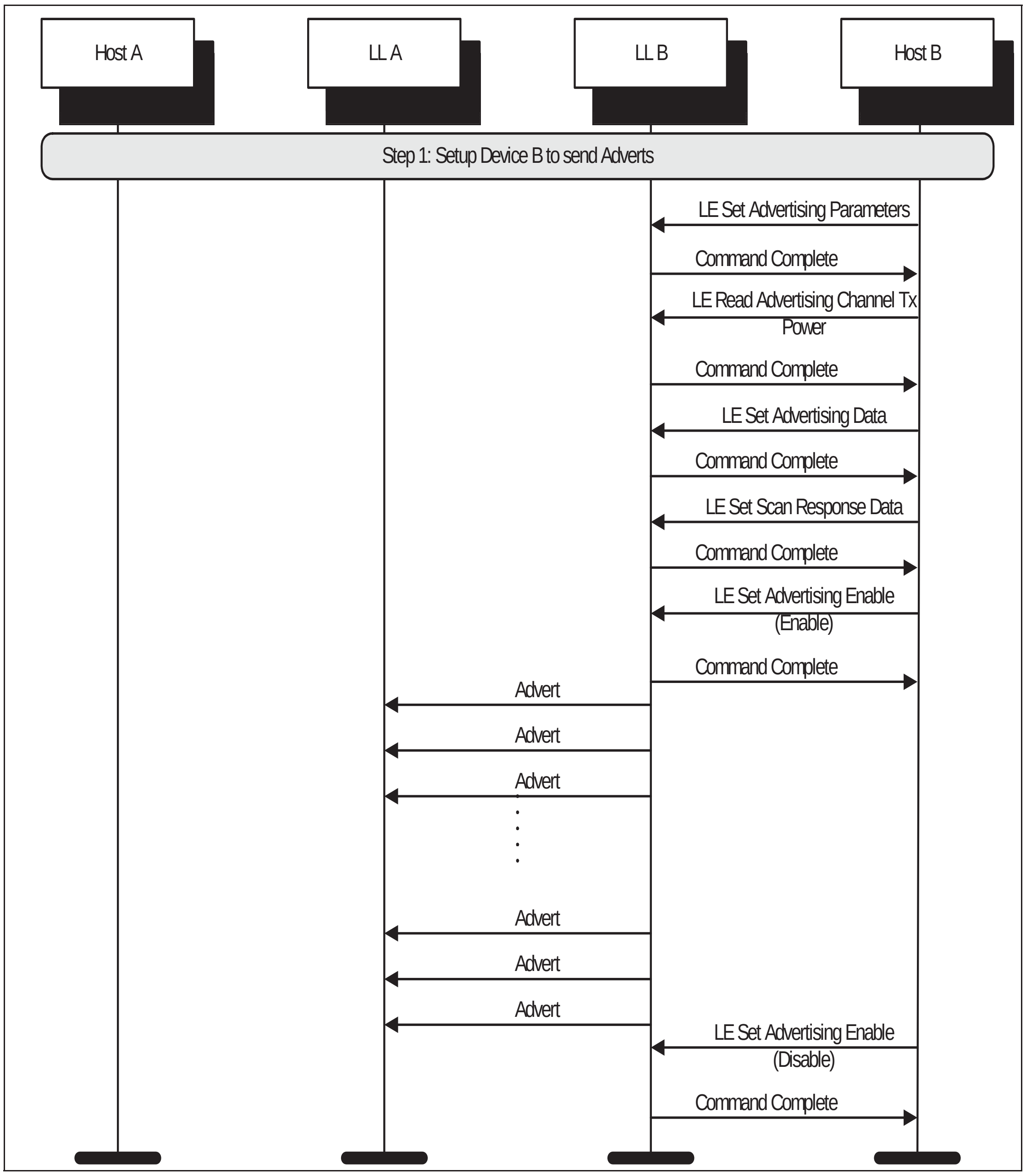
Host A
LL A
LL B
Host B
Step 1: Setup Device B to send Adverts
LE Set Advertising Parameters
Command Complete
LE Read Advertising Channel Tx Power
Command Complete
LE Set Advertising Data
Command Complete
LE Set Scan Response Data
Command Complete
LE Set Advertising Enable (Enable)
Command Complete
Advert
Advert
Advert
Advert
Advert
Advert
LE Set Advertising Enable (Disable)
Command Complete

Figure 3.1: Undirected Advertising

3.2 DIRECTED ADVERTISING

A device may use directed advertising to allow an initiator to connect to it. Directed advertising is time limited in the Controller and therefore this may fail before a connection is created. This example only shows the failure case (see Figure 3.2).

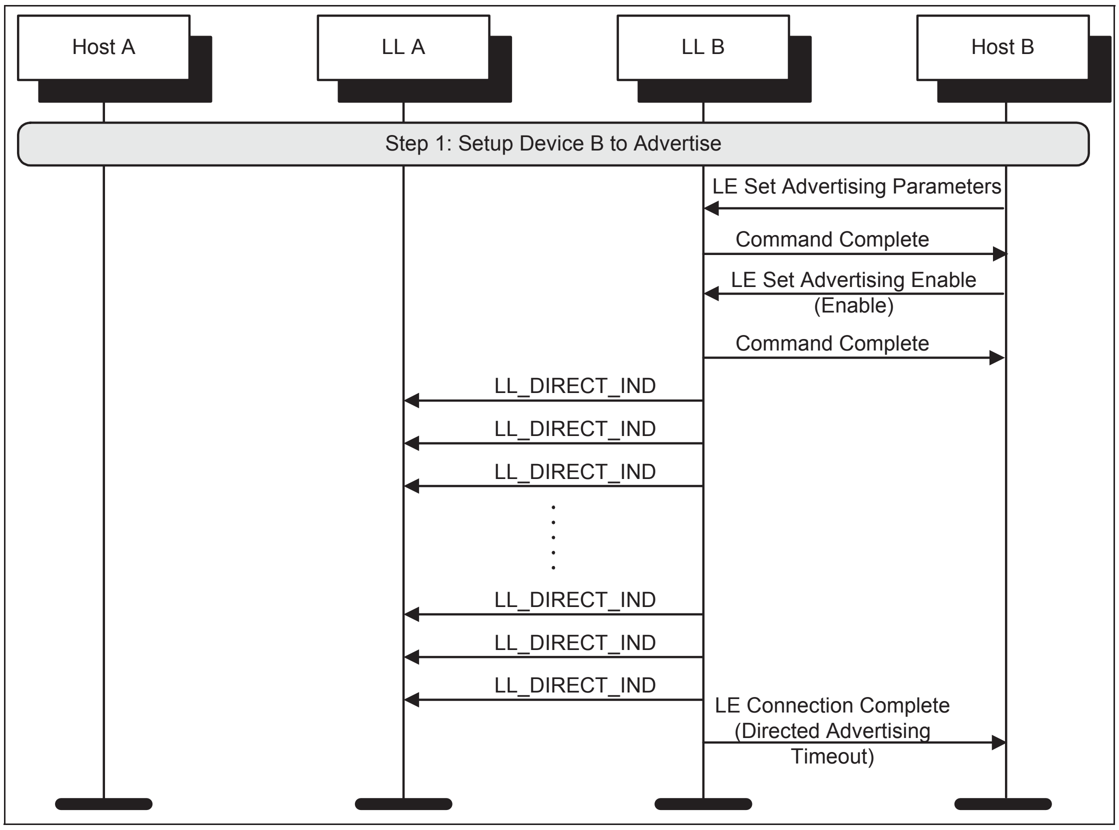
Host A
LL A
LL B
Host B
Step 1: Setup Device B to Advertise
LE Set Advertising Parameters
Command Complete
LE Set Advertising Enable (Enable)
Command Complete
LL_DIRECT_IND
LL_DIRECT_IND
LL_DIRECT_IND
LL_DIRECT_IND
LL_DIRECT_IND
LL_DIRECT_IND
LE Connection Complete (Directed Advertising Timeout)

Figure 3.2: Directed Advertising showing failure case

4 SCANNING STATE

4.1 PASSIVE SCANNING

A device can use passive scanning to find advertising devices in the area. This would receive advertising packets from peer devices and report these to the Host (see Figure 4.1).

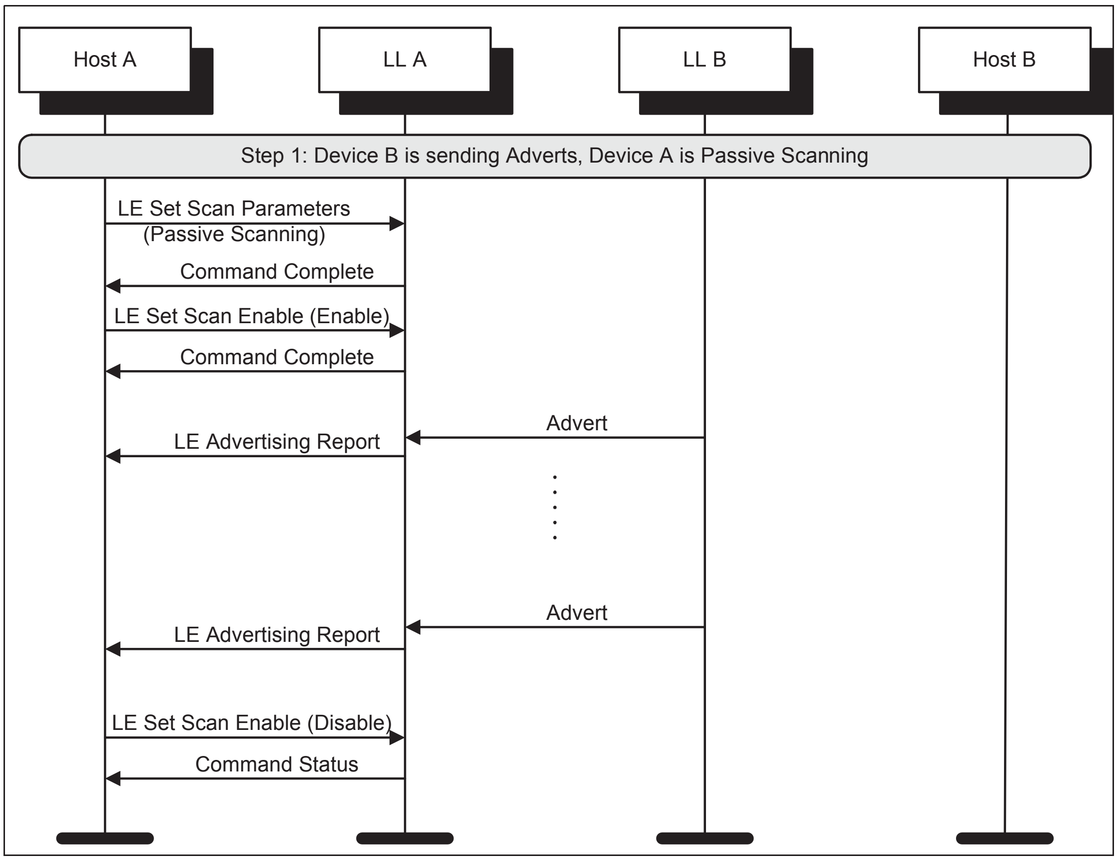
Host A
LL A
LL B
Host B
Step 1: Device B is sending Adverts, Device A is Passive Scanning
LE Set Scan Parameters (Passive Scanning)
Command Complete
LE Set Scan Enable (Enable)
Command Complete
Advert
LE Advertising Report
Advert
LE Advertising Report
LE Set Scan Enable (Disable)
Command Status

Figure 4.1: Passive Scanning

4.2 ACTIVE SCANNING

A device may use active scanning to obtain more information about devices that may be useful to populate a user interface. Active scanning involves more link layer advertising messages (see Figure 4.2).

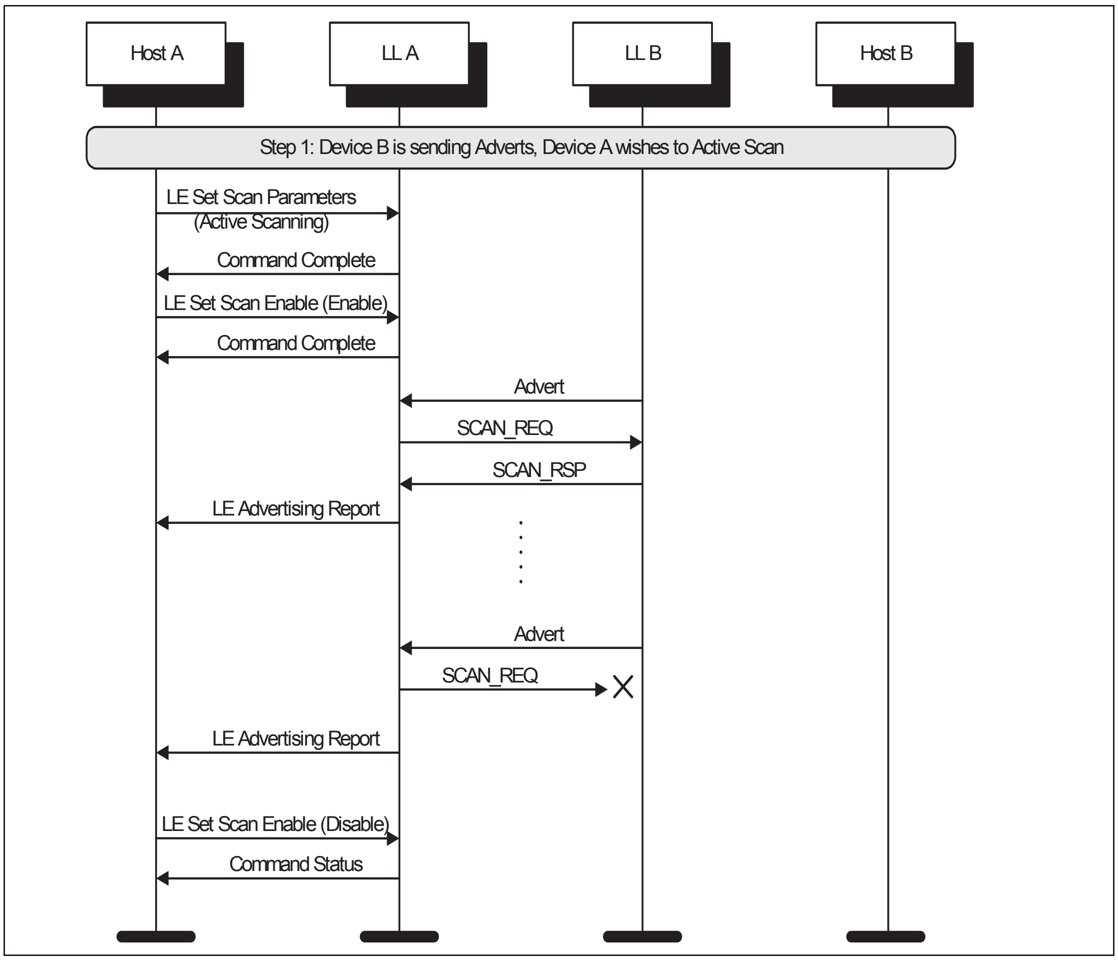
Host A
LL A
LL B
Host B
Step 1: Device B is sending Adverts, Device A wishes to Active Scan
LE Set Scan Parameters (Active Scanning)
Command Complete
LE Set Scan Enable (Enable)
Command Complete
Advert
SCAN_REQ
SCAN_RSP
LE Advertising Report
Advert
SCAN_REQ
LE Advertising Report
LE Set Scan Enable (Disable)
Command Status

Figure 4.2: Active Scanning

5 INITIATING STATE

5.1 INITIATING A CONNECTION

A device can initiate a connection to an advertiser. This example shows a successful initiation, resulting in both devices able to send application data (see Figure 5.1).

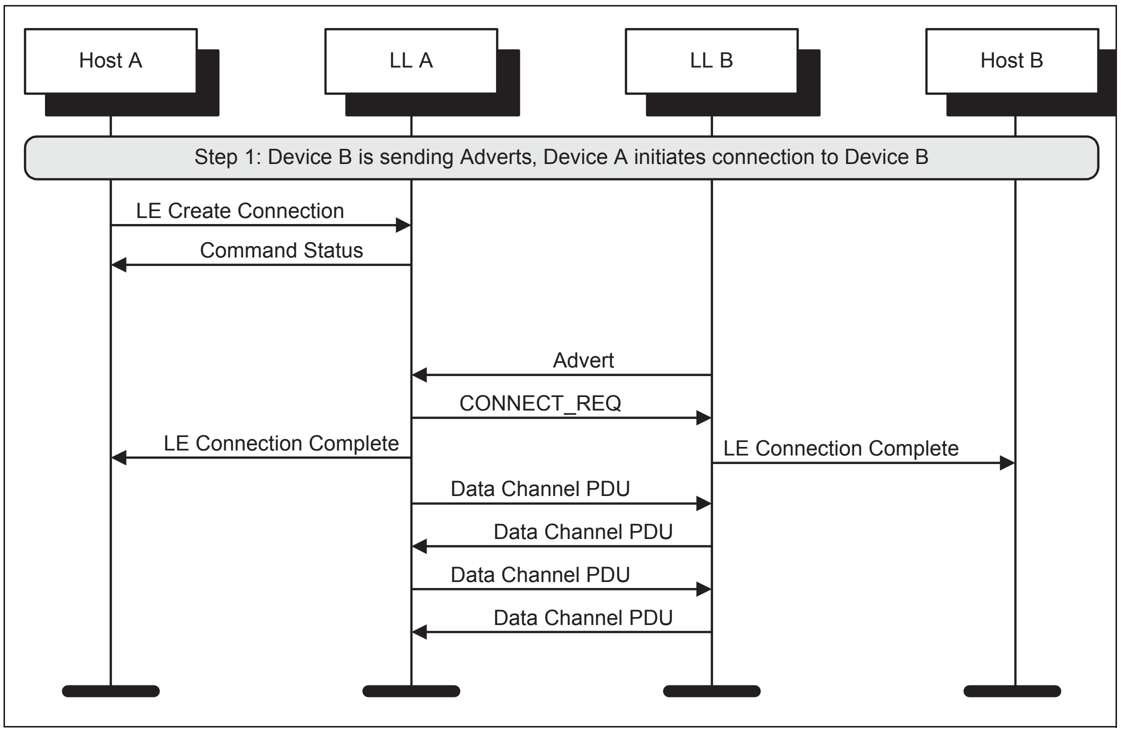

Figure 5.1: Initiating a Connection

5.2 CANCELING AN INITIATION

A device can cancel a pending connection creation. This example shows an unsuccessful initiation, followed by a cancellation of the initiation (see Figure 5.2).

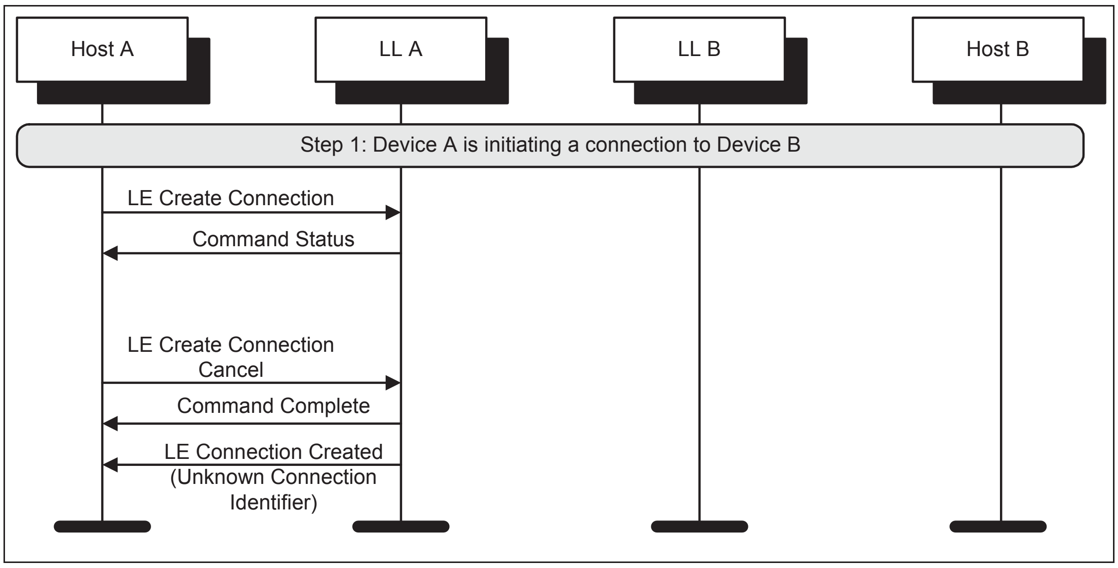
Host A
LL A
LL B
Host B
Step 1: Device A is initiating a connection to Device B
LE Create Connection
Command Status
LE Create Connection Cancel
Command Complete
LE Connection Created (Unknown Connection Identifier)

Figure 5.2: Canceling an Initiation

6 CONNECTION STATE

6.1 SENDING DATA

Once two devices are in a connection, either device can send data. This example shows both devices sending data, for example when the attribute protocol does a read request and a read response is returned (see Figure 6.1).

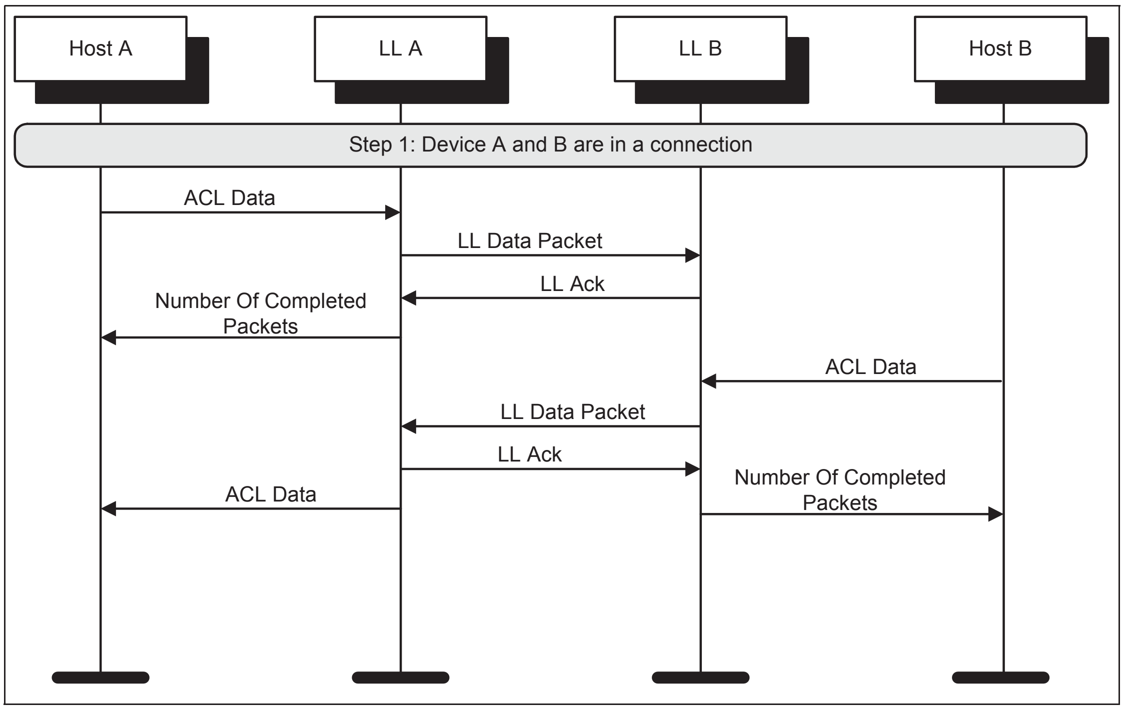
Host A
LL A
LL B
Host B
Step 1: Device A and B are in a connection
ACL Data
LL Data Packet
LL Ack
Number Of Completed Packets
ACL Data
LL Data Packet
LL Ack
ACL Data
Number Of Completed Packets

Figure 6.1: Sending Data

6.2 CONNECTION UPDATE

The master of the connection may request a connection update using a Link Layer Control Procedure (see Figure 6.2).

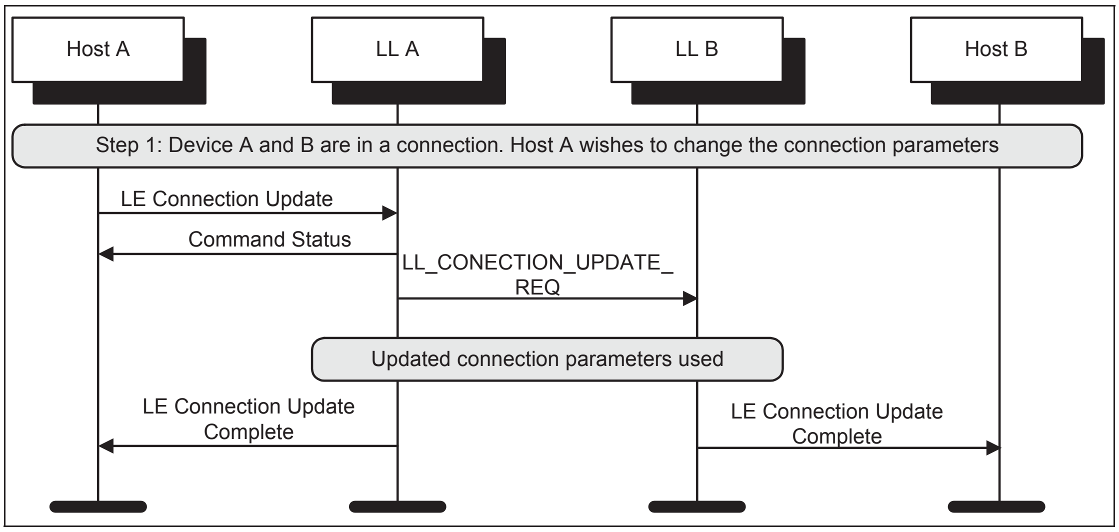
Host A
LL A
LL B
Host B
Step 1: Device A and B are in a connection. Host A wishes to change the connection parameters
LE Connection Update
Command Status
LL_CONECTION_UPDATE_REQ
Updated connection parameters used
LE Connection Update Complete
LE Connection Update Complete

Figure 6.2: Connection Update

6.3 CHANNEL MAP UPDATE

The Controller of the master may receive some channel classification data from the Host and then perform the Channel Update Link Layer Control Procedure (see Figure 6.3).

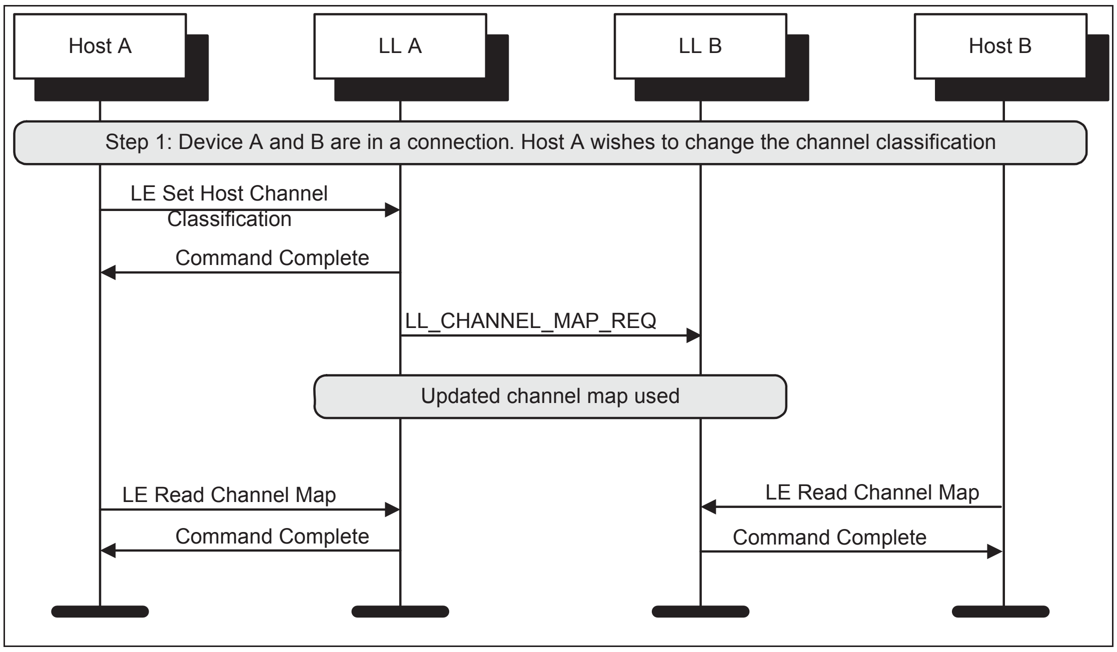

Figure 6.3: Channel Map Update

6.4 FEATURES EXCHANGE

The master device can discover the set of features available on the slave device. To achieve this, the Feature Exchange Link Layer Control Procedure is used (see Figure 6.4).

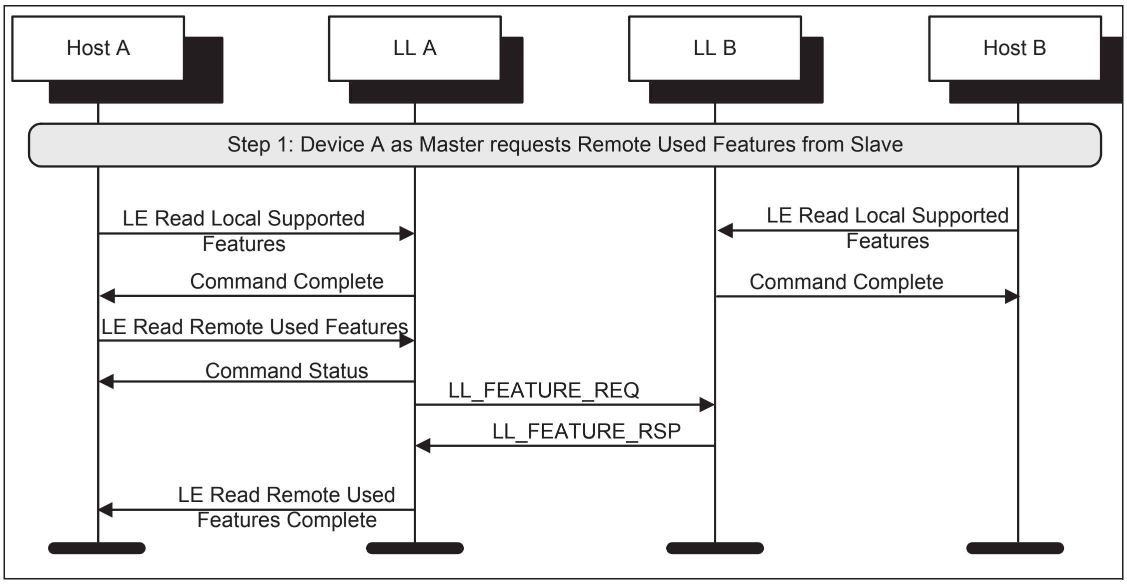
Host A
LL A
LL B
Host B
Step 1: Device A as Master requests Remote Used Features from Slave
LE Read Local Supported Features
LE Read Local Supported Features
Command Complete
Command Complete
LE Read Remote Used Features
Command Status
LL_FEATURE_REQ
LL_FEATURE_RSP
LE Read Remote Used Features Complete

Figure 6.4: Features Exchange

6.5 VERSION EXCHANGE

Either device may perform a version exchange (see Figure 6.5 and Figure 6.6).

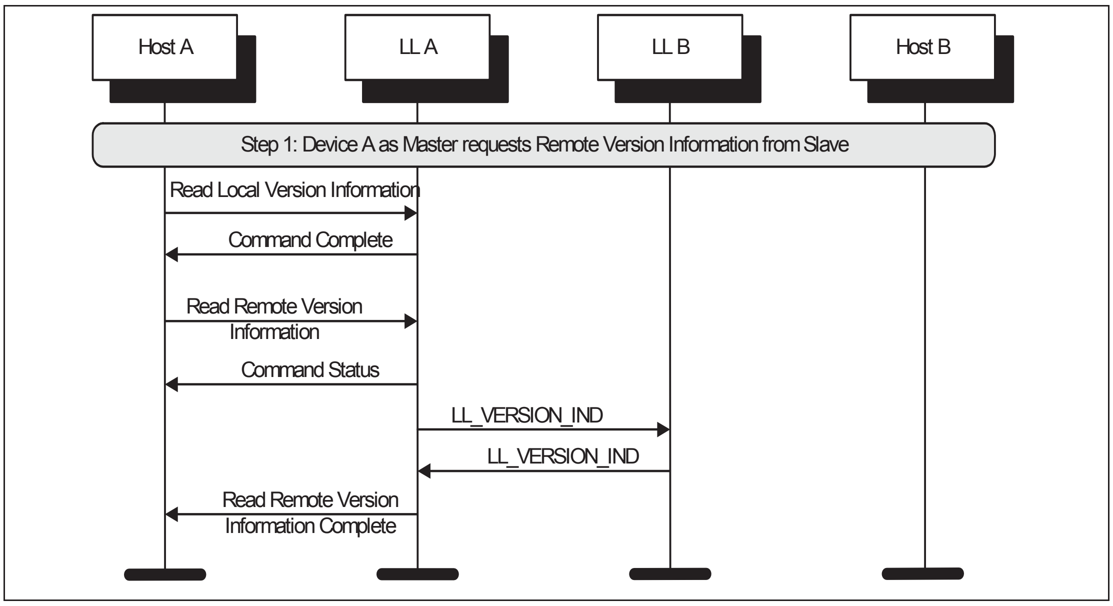
Host A
LL A
LL B
Host B
Step 1: Device A as Master requests Remote Version Information from Slave
Read Local Version Information
Command Complete
Read Remote Version Information
Command Status
LL_VERSION_IND
LL_VERSION_IND
Read Remote Version Information Complete

Figure 6.5: Version Exchange from Master

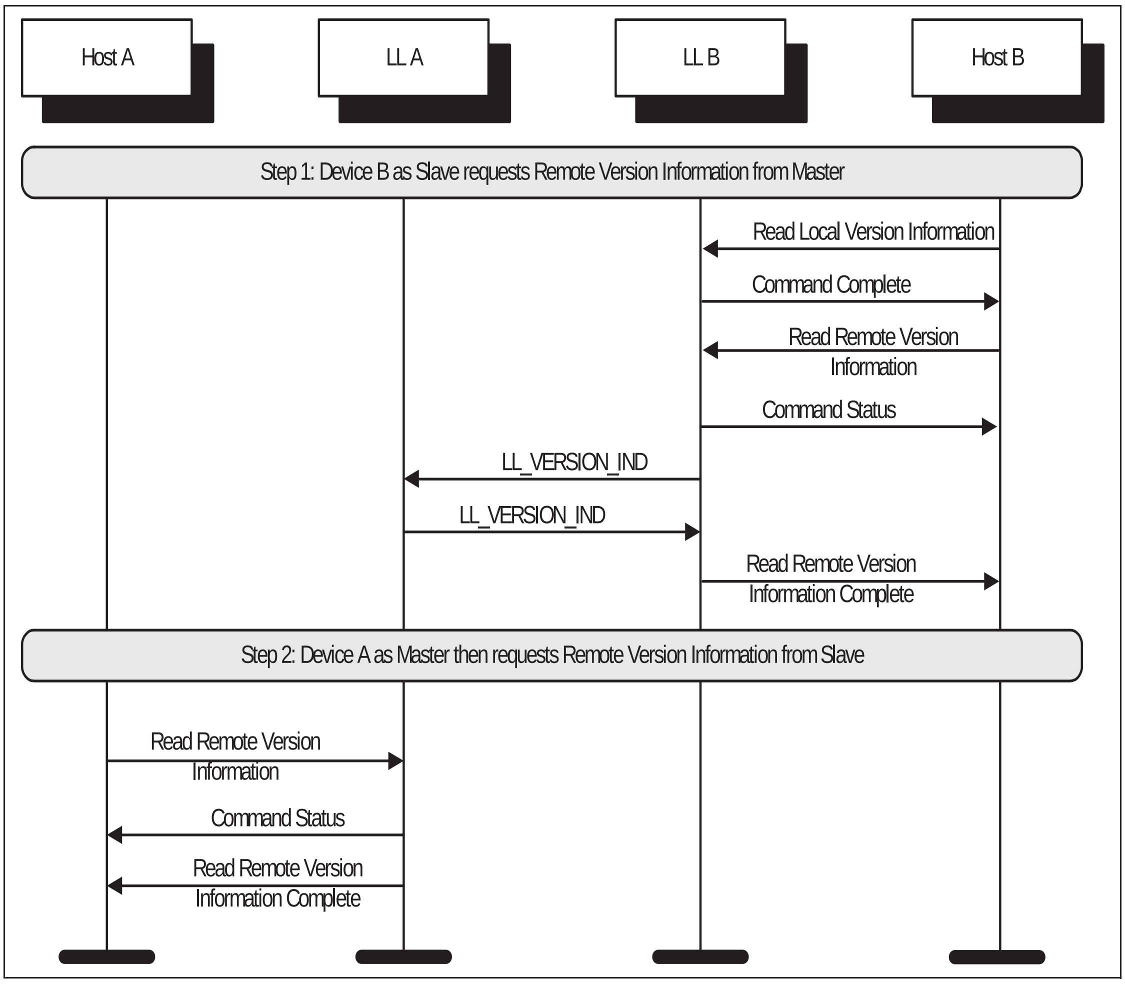
Host A
LL A
LL B
Host B
Step 1: Device B as Slave requests Remote Version Information from Master
Read Local Version Information
Command Complete
Read Remote Version Information
Command Status
LL_VERSION_IND
LL_VERSION_IND
Read Remote Version Information Complete
Step 2: Device A as Master then requests Remote Version Information from Slave
Read Remote Version Information
Command Status
Read Remote Version Information Complete

Figure 6.6: Version Exchange from Slave

6.6 START ENCRYPTION

If encryption has not been started on a connection, it may be started by the master (see Figure 6.7).

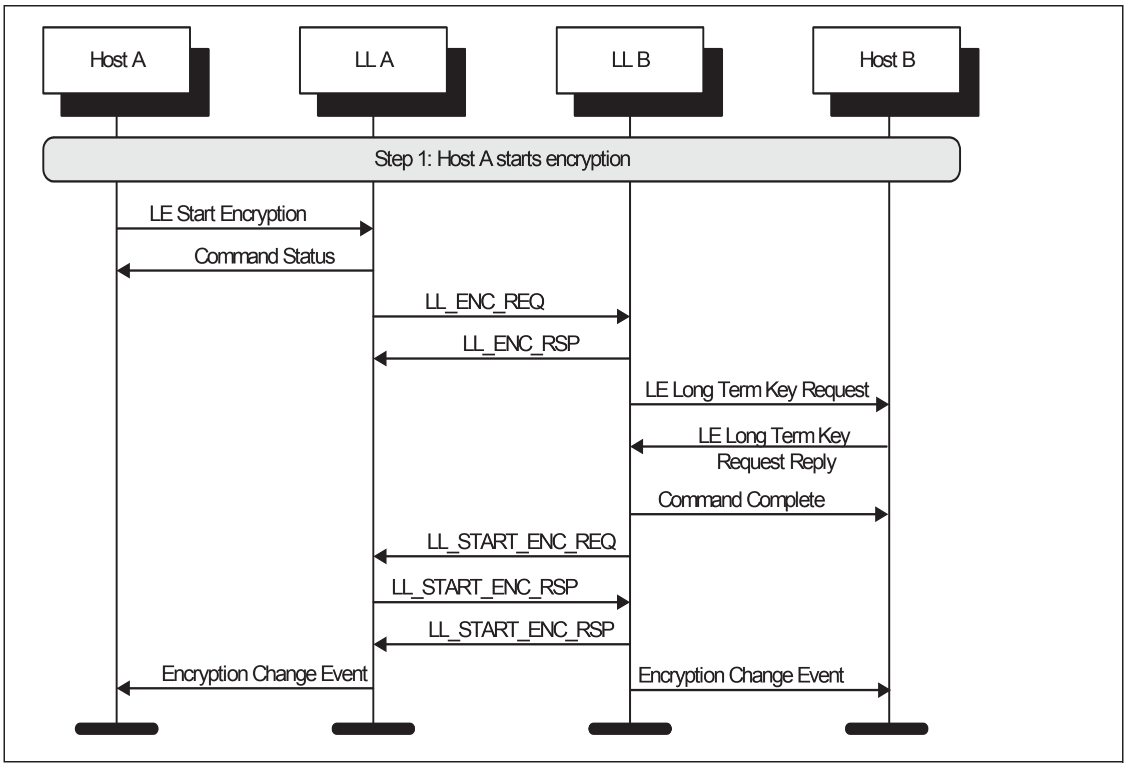

Figure 6.7: Start Encryption

6.7 START ENCRYPTION WITHOUT LONG TERM KEY

If encryption has not been started on a connection, it may be started by the master. Figure 6.8 shows the failure case of the slave not having the long term key for the master.

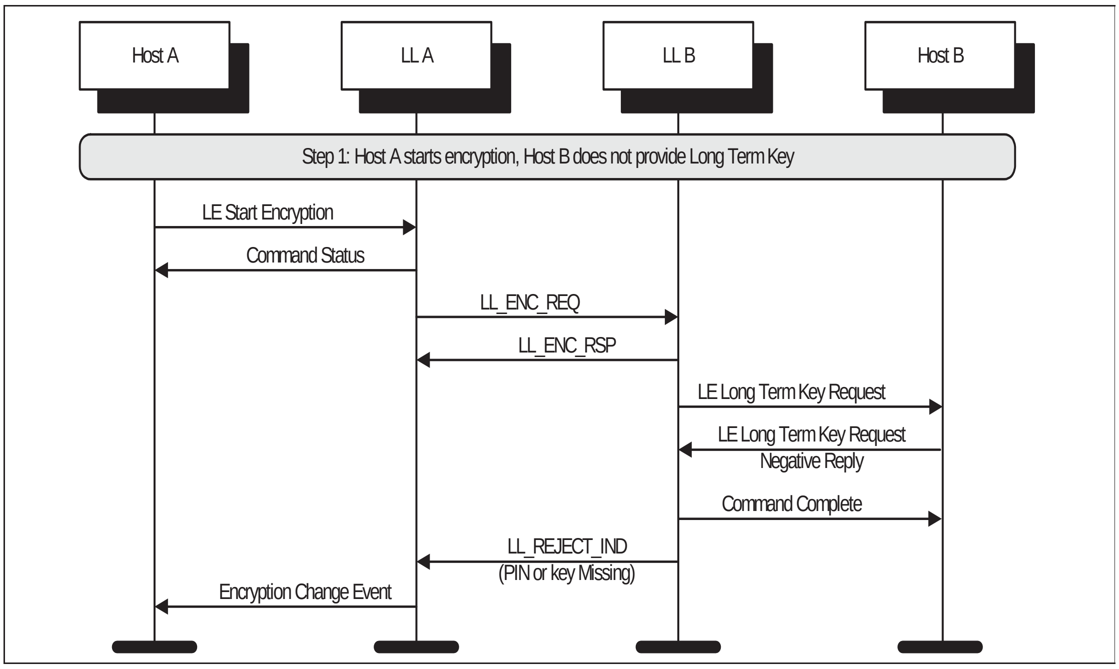
Host A
LL A
LL B
Host B
Step 1: Host A starts encryption, Host B does not provide Long Term Key
LE Start Encryption
Command Status
LL_ENC_REQ
LL_ENC_RSP
LE Long Term Key Request
LE Long Term Key Request Negative Reply
Command Complete
LL_REJECT_IND (PIN or key Missing)
Encryption Change Event

Figure 6.8: Start encryption without long-term key

6.8 START ENCRYPTION WITH EVENT MASKED

If encryption has not been started on a connection, it may be started by the master. Figure 6.9 shows the failure case when the slave has masked out the LE Long Term Key Request event.

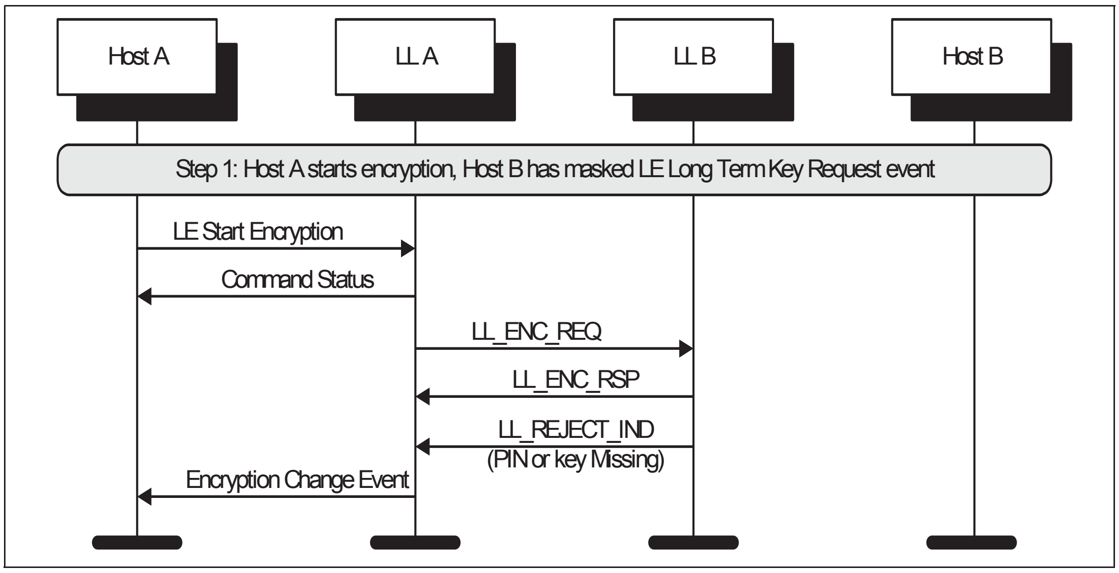

Figure 6.9: Start encryption with slave masking out event

6.9 START ENCRYPTION WITHOUT SLAVE SUPPORTING ENCRYPTION

If Encryption has not been started on a connection, it may be started by the master. Figure 6.10 shows the failure case of the slave that does not support the encryption feature.

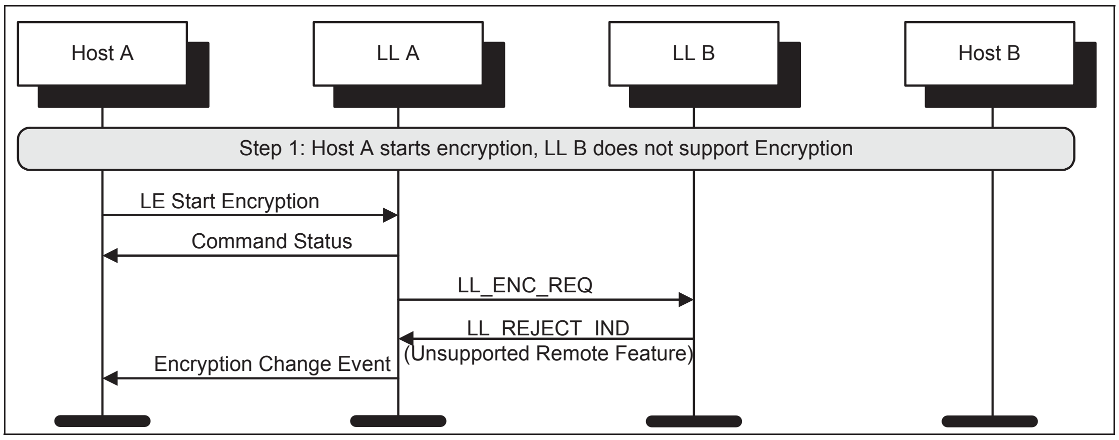

Figure 6.10: Start Encryption failure when slave does not support encryption

6.10 RESTART ENCRYPTION

If encryption has already been started on a connection, it may be restarted by the master. This may be required to use a stronger encryption as negotiated by the Security Manager Protocol (see Figure 6.11).

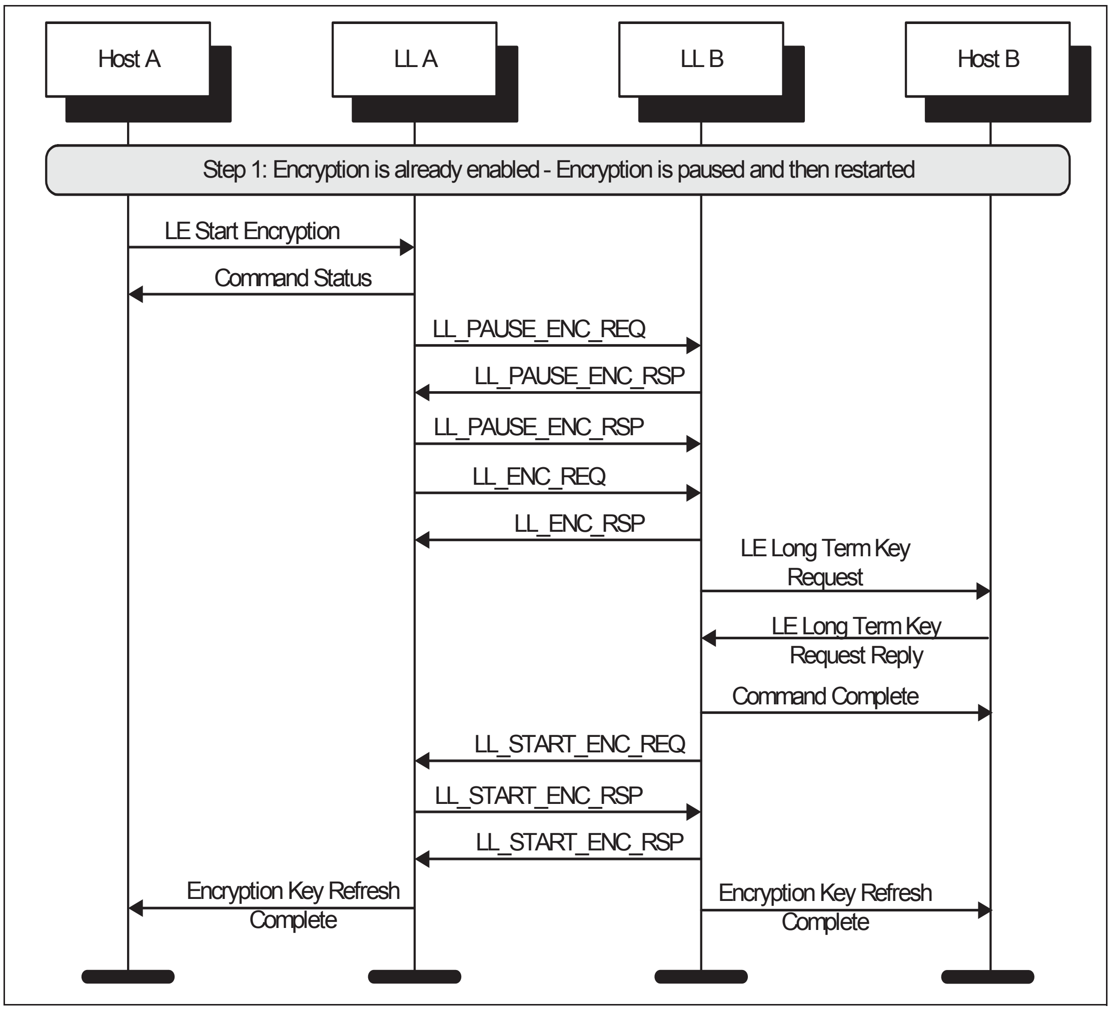

Figure 6.11: Restart Encryption

6.11 DISCONNECT

Once a connection has no need to be kept active, the Host can disconnect it. This can be done by either device (see Figure 6.12 and Figure 6.13).

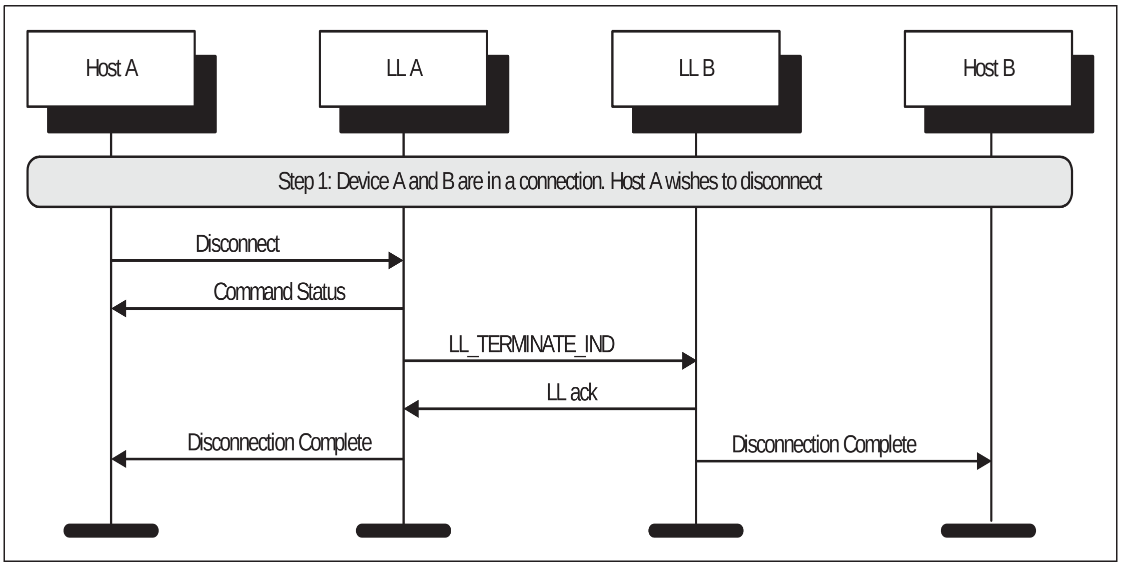

Figure 6.12: Disconnect from Master

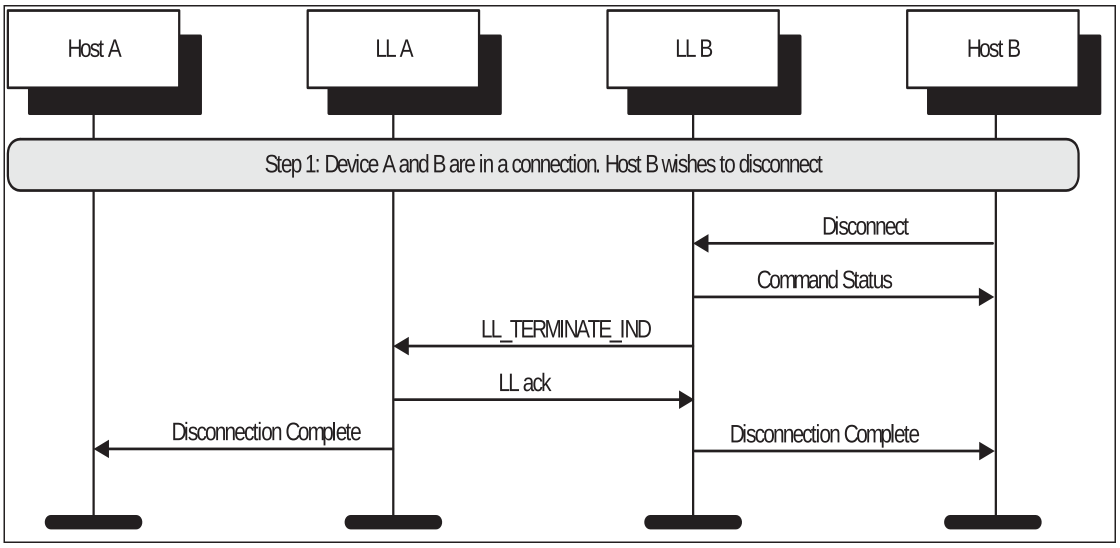

Figure 6.13: Disconnect from Slave

LOW ENERGY LINK LAYER SECURITY

This part of the specification describes the Link Layer security for Bluetooth low energy.

CONTENTS

1 ENCRYPTION AND AUTHENTICATION OVERVIEW

The Link Layer provides encryption and authentication using Counter with Cipher Block Chaining-Message Authentication Code (CCM) Mode, which shall be implemented consistent with the algorithm as defined in IETF RFC 3610 (http://www.ietf.org/rfc/rfc3610.txt) in conjunction with the AES-128 block cipher as defined in NIST Publication FIPS-197 (http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf). A description of the CCM algorithm can also be found in the NIST Special Publication 800-38C (http://csrc.nist.gov/publications/PubsSPs.html).

This specification uses the same notation and terminology as the IETF RFC except for the Message Authentication Code (MAC) that in this specification is called the Message Integrity Check (MIC) to avoid confusion with the term Media Access Controller.

CCM has two size parameters, M and L. The Link Layer defines these to be:

- M = 4; indicating that the MIC (authentication field) is 4 octets
- L = 2; indicating that the Length field is 2 octets

CCM requires a new temporal key whenever encryption is started. CCM also requires a unique nonce value for each Data Channel PDU protected by a given temporal key. The CCM nonce shall be 13 octets.

The Link Layer connection may be either encrypted and authenticated or unencrypted and unauthenticated. In an encrypted and authenticated connection, all the Data Channel PDUs with a non-zero length Payload shall be encrypted and authenticated. Authentication is performed by appending a MIC field to the Payload. The MIC shall be calculated over the Data Channel PDU's Payload field and the first octet of the header (Part B, Section 2.4) with the NESN, SN and MD bits masked to zero.

Encryption shall be applied to the Data Channel PDU's Payload field and MIC.

Each new Data Channel PDU with a non-zero length Payload shall be decrypted and authenticated before being sent to the Host or processed by the Link Layer. Authentication is unrelated to Link Layer acknowledgement scheme; authentication does not have to be performed before the packet is acknowledged by the Link Layer.

In the unlikely event of an authentication failure being detected, the connection shall be considered lost. The Link Layer shall not send or receive any further packets on that connection. The Link Layer shall exit the Connection State and transition to the Standby State. The Host shall be notified of the loss of connection due to an authentication failure. The peer Link Layer will detect this loss of connection through the supervision timeout procedure.

2 CCM

This section provides the details for using the CCM algorithm. As specified, the CCM algorithm requires the payload and some additional parameters to be formatted into the CCM nonce, counter-mode blocks and encryption blocks. The CCM nonce provides uniqueness to each packet. The counter-mode blocks are used to calculate the MIC. The encryption blocks provide the keystream that is used to encrypt the payload and the MIC of the Data Channel PDU.

Sample data of the blocks (see Section 2.2 and Section 2.3) can be found in [Vol 6] Part C, Section and Section 1.2.

2.1 CCM NONCE

The CCM nonce is constructed from a 39-bit *packetCounter*, 1-bit *directionBit* and an 8-octet IV (initialization vector). The format of the 13-octet nonce shall be as shown in Table 2.1.

Octet	**Field**	**Size (octets)**	**Value**	**Description**
0	Nonce0	1	variable	Octet0 (LSO) of *packetCounter*
1	Nonce1	1	variable	Octet1 of *packetCounter*
2	Nonce2	1	variable	Octet2 of *packetCounter*
3	Nonce3	1	variable	Octet3 of *packetCounter*
4	Nonce4	1	variable	Bit 6 – Bit 0: Octet4 (7 most significant bits of *packetCounter*, with Bit 6 being the most significant bit) Bit7:*directionBit*
5	Nonce5	1	variable	Octet0 (LSO) of IV
6	Nonce6	1	variable	Octet1 of IV
7	Nonce7	1	variable	Octet2 of IV
8	Nonce8	1	variable	Octet3 of IV
9	Nonce9	1	variable	Octet4 of IV
10	Nonce10	1	variable	Octet5 of IV
11	Nonce11	1	variable	Octet6 of IV
12	Nonce12	1	variable	Octet7 (MSO) of IV

Table 2.1: CCM nonce format

The Link Layer shall maintain one *packetCounter* per Role for each connection.

For each connection, the *packetCounter* shall be set to zero for the first encrypted Data Channel PDU sent during the encryption start procedure. The *packetCounter* shall then be incremented by one for each new Data Channel PDU that is encrypted. The *packetCounter* shall not be incremented for retransmissions.

The *directionBit* shall be set to ‘1’ for Data Channel PDUs sent by the master and set to ‘0’ for Data Channel PDUs sent by the slave.

The IV is common for both Roles of a connection. Whenever encryption is started or restarted, a new 8-octet IV shall be used for each pair of communicating devices. The IV is determined as specified in Part B, Section 5.1.3.1.

2.2 COUNTER MODE BLOCKS

For calculating the MIC, the multiple counter mode blocks are generated according to the CCM specification. These are referred to as blocks $B_0 – B_n$. Table 2.2 defines the format of block B_0. Table 2.3 defines the format of block B_1 that is devoted to the authentication of the additional authenticated data. Additional B blocks are generated as needed for authentication of the payload.

Offset (octets)	**Field**	**Size (octets)**	**Value**	**Description**
0	Flags	1	0x49	As per the CCM specification
1	Nonce	13	variable	The nonce as described in Table 2.1. Nonce0 shall have offset 1. Nonce12 shall have offset 13.
14	Length[MSO]	1	0x00	The most significant octet of the length of the payload
15	Length[LSO]	1	variable	The least significant octet of the length of the payload

Table 2.2: Block B_0 format

Offset	**Field**	**Size (octets)**	**Value**	**Description**
0	AAD_Length[MSO]	1	0x00	The most significant octet of the length of the additional authenticated data
1	AAD_Length[LSO]	1	0x01	The least significant octet of the length of the additional authenticated data
2	AAD	1	variable	The data channel PDU header’s first octet with NESN, SN and MD bits masked to 0

Table 2.3: Block B_1 format

Offset	Field	Size (octets)	Value	Description
3	Padding	13	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	These octets are only used to pad the block. They are not part of the packet and never transmitted.

Table 2.3: Block B_1 format

2.3 ENCRYPTION BLOCKS

The CCM algorithm uses the A_x blocks to generate keystream that is used to encrypt the MIC and the Data Channel PDU payload. Block A_0 is always used to encrypt and decrypt the MIC. Block A_1 is always used to encrypt and decrypt the first 16 octets of the Payload. Block A_2 is always used to encrypt and decrypt the rest of the Payload as needed.

Offset (octets)	Field	Size (octets)	Value	Description
0	Flags	1	0x01	As per the CCM specification
1	Nonce	13	variable	The nonce as described above Nonce0 shall have offset 1. Nonce12 shall have offset 13.
14	i[MSO]	1	variable	The most significant octet of the counter i
15	i[LSO]	1	variable	The least significant octet of the counter i

Table 2.4: Block A_x format

Core System Package [Low Energy Controller volume]

Part F

DIRECT TEST MODE

This part of the specification describes the Direct Test Mode for RF PHY testing of Bluetooth low energy devices.

CONTENTS

1 INTRODUCTION

Direct Test Mode is used to control the Device Under Test (DUT) and provides a report back to the Tester.

Direct Test Mode shall be set up using one of two alternate methods:

1. over HCI (as defined in Section 2) or
2. through a 2-wire UART interface (as defined in Section 3)

Each DUT shall implement one of the two Direct Test Mode methods in order to test the Low Energy PHY layer. Figure 1.1 illustrates the alternatives for Direct Test Mode setup.

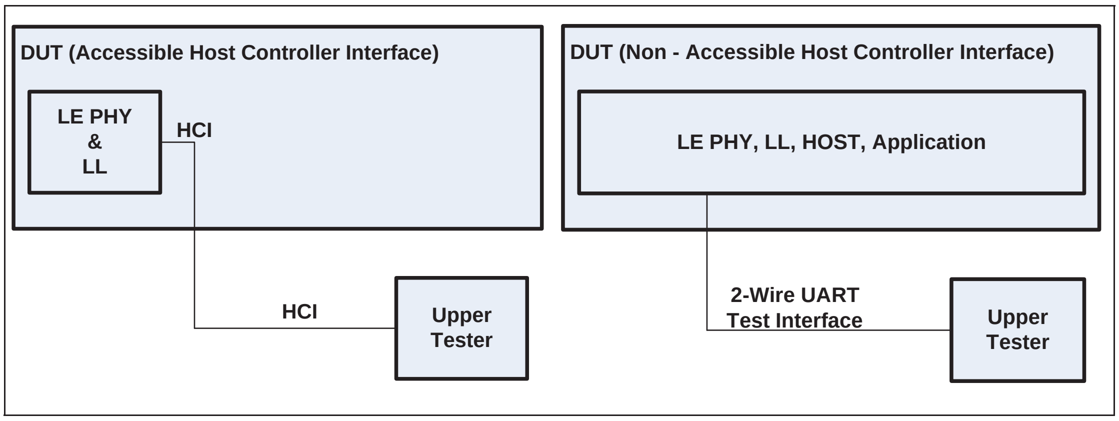
DUT (Accessible Host Controller Interface)
LE PHY & LL
HCI
HCI
Upper Tester
DUT (Non - Accessible Host Controller Interface)
LE PHY, LL, HOST, Application
2-Wire UART Test Interface
Upper Tester

Figure 1.1: Setup alternatives for LE Direct Test Mode: Designs with accessible HCI (left) and designs without accessible HCI (right)

Figure 1.2 illustrates the Bluetooth LE Direct Test Mode setup principle using a 2-wire UART interface.

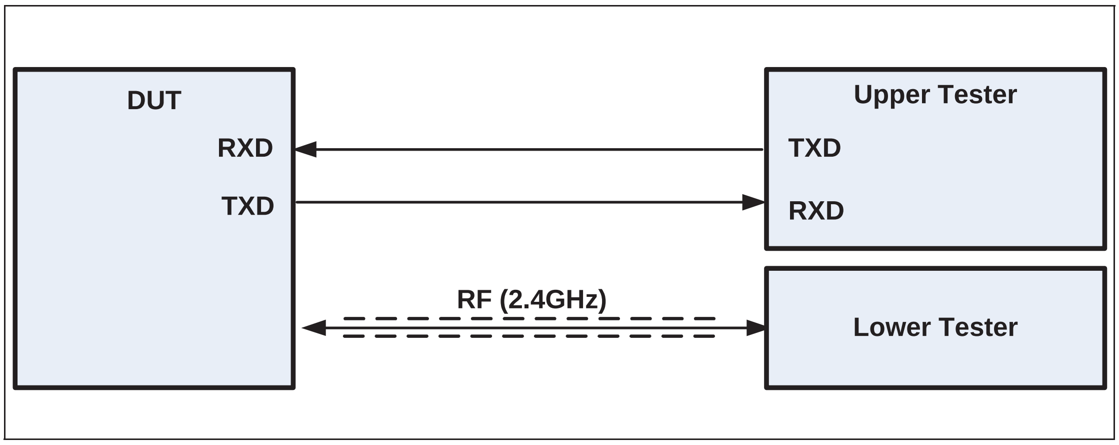

Figure 1.2: RF PHY test setup for Direct Test Mode (UART control)

2 LOW ENERGY TEST SCENARIOS

2.1 TEST SEQUENCES

These sequences are used as routines and used to control an LE DUT with an accessible HCI or a 2-wire UART interface for RF testing.

The following mapping shall be performed from the RF testing commands to HCI commands and events or 2-wire UART commands and events:

RF Test Command / Event	**HCI Command / Event**	**2-wire UART Command / Event**
LE_TRANSMITTER_TEST	LE Transmitter Test command	LE Transmitter Test
LE_RECEIVER_TEST	LE Receiver Test command	LE Receiver Test
LE_TEST_END	LE Test End command	LE Test End
LE_STATUS	Command Complete event	LE Test Status
LE_PACKET_REPORT	Command Complete event	LE Packet Report

Table 2.1: Mapping table of HCI / 2-wire Commands/Events

The HCI commands and events used in Direct Test Mode are defined in [Vol. 2] Part E, Section 7.8

2.2 MESSAGE SEQUENCE CHARTS

Transmitter Test

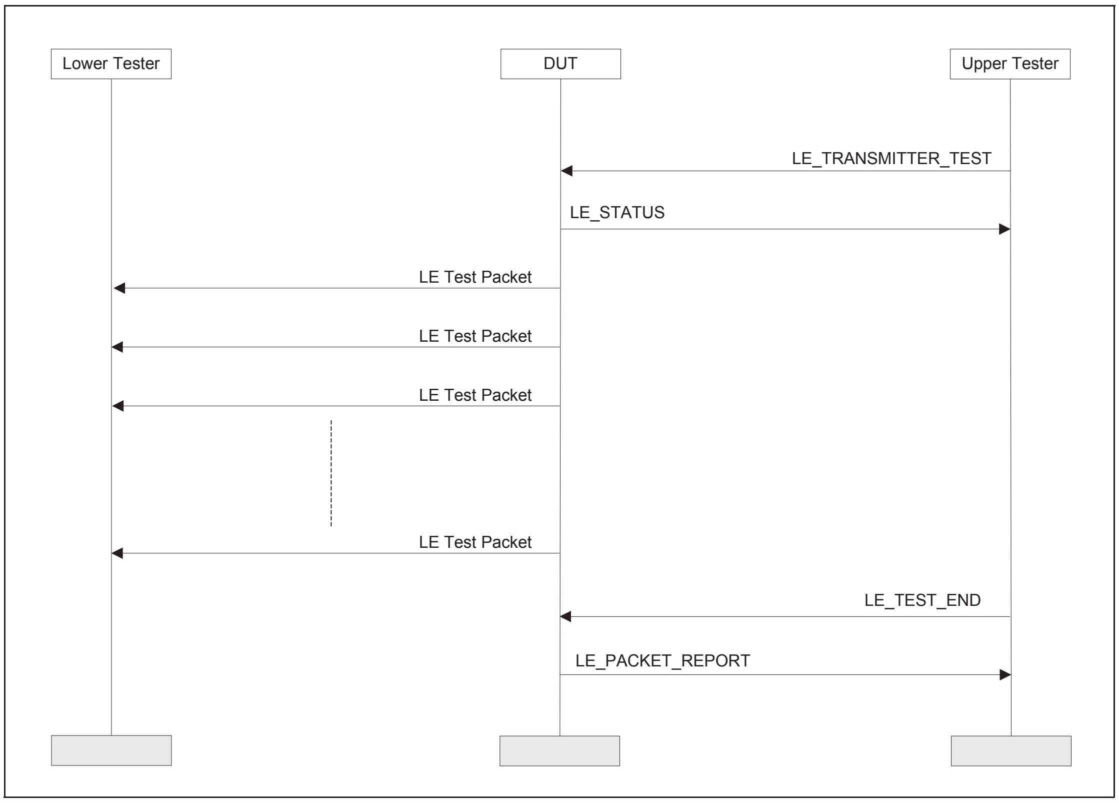

Figure 2.1: Transmitter Test MSC

Receiver Test

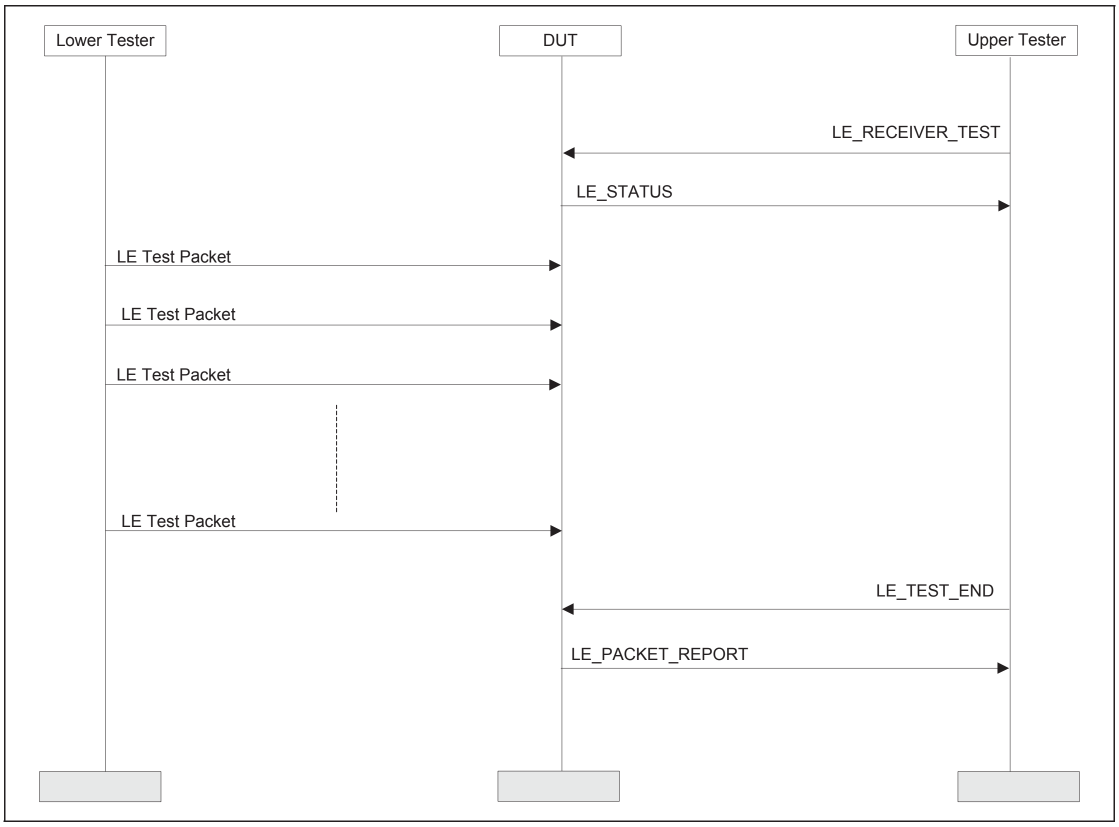

Figure 2.2: Receiver Test MSC

3 UART TEST INTERFACE

3.1 UART INTERFACE CHARACTERISTICS

The UART interface characteristics shall be set to use the following parameters:

- Baud rate: One of the following shall be supported by the DUT:
  1200, 2400, 9600, 14400, 19200, 38400, 57600, 115200
- Number of data bits: 8
- No parity
- 1 stop bit
- No flow control (RTS or CTS)

3.2 UART FUNCTIONAL DESCRIPTION

The Upper Tester shall always initiate any a test scenario using the UART interface. The DUT shall respond to the commands from the Upper Tester.

The Upper Tester sends test commands to the DUT. The DUT shall respond with a status event or the packet reporting event.

The Upper Tester shall not transmit further commands before it receives a response from the DUT. If the Upper Tester does not receive a response from the DUT within the time $t_{TIMEOUT}$, the Upper Tester shall transmit a reset command to the DUT and display an appropriate error message.

On reception of a reset command, the DUT shall reset all parameters to their default state.

Definitions

- All Commands and Events consist of 16 bits (2 octets).
- The most significant bit is bit number 15.
- The least significant bit is bit number 0.
- The most significant octet is from bit 15 to 8.
- The least significant octet is from bit 7 to 0.
- Commands and Events are sent most significant octet (MSO) first, followed by the least significant octet (LSO).

3.3 COMMANDS AND EVENTS

3.3.1 Command and Event Behavior

Table 3.1 outlines the set of commands which can be received by the DUT and the corresponding response events that can be transmitted by the DUT.

Command (DUT RXD)	Event (DUT TXD)
LE_Reset	LE_Test_Status SUCCESS LE_Test_Status FAIL
LE_Receiver_Test	LE_Test_Status SUCCESS LE_Test_Status FAIL
LE_Transmitter_Test	LE_Test_Status SUCCESS LE_Test_Status FAIL
LE_Test_End	LE_Packet_Report LE_Test_Status FAIL

Table 3.1: 2-Wire command and event behavior

3.3.2 Commands

Command packet format is as shown in Figure 3.1.

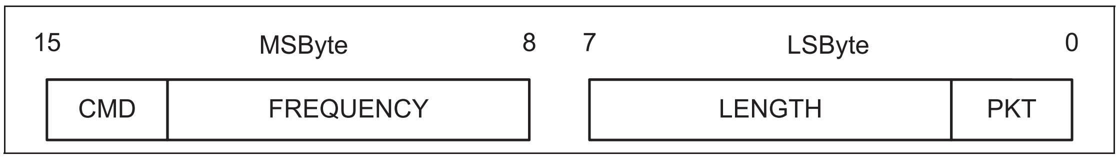

Figure 3.1: Command message format

CMD (command): *Size: 2 Bits*

Value b_1b_0	Parameter Description
00	Reset
01	Receiver Test
10	Transmitter Test
11	Test End

Frequency: *Size: 6 Bits*

Value	Parameter Description
N	When N = 0x00 – 0x27: N = (F-2402)/2 Frequency Range 2402 MHz to 2480 MHz Where N = 0x28 – 0x3F: Reserved

Length: *Size: 6 Bits*

Value	Parameter Description
N	When N = 0x00 – 0x25: Length in octets of payload data in each packet When N = 0x26 – 0x3F: Reserved

PKT (Packet Type): *Size: 2 Bits*

Value b_1b_0	Parameter Description
00	PRBS9 Packet Payload
01	11110000 Packet Payload
10	10101010 Packet Payload
11	Vendor specific

3.4 EVENTS

There are two types of events sent by the DUT:

1. LE_Test_Status_Event
2. LE_Test_Packet_Report_Event

Event packet format is as shown in Figure 3.2. This packet format is used for both Command Status Events and Packet Report Events.

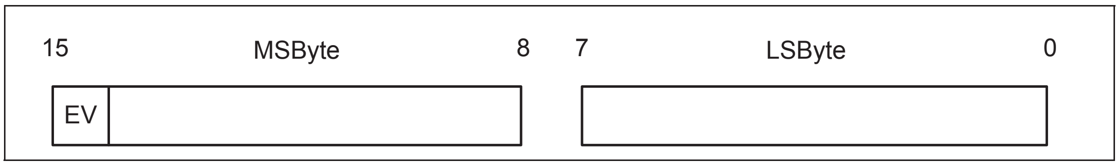

Figure 3.2: Event packet format

EV (Event): *Size: 1 Bit*

Value	Parameter Description
0	LE_Test_Status_Event
1	LE_Packet_Reporting_Event

3.4.1 LE_Test_Status_Event

The LE_Test_Status_Event packet format is as shown in Figure 3.3.

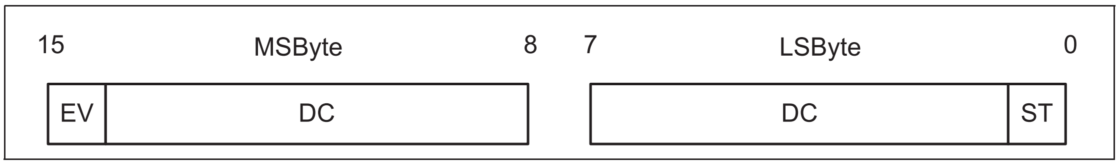

Figure 3.3: LE Test Status event

ST (status): *Size: 1 Bit*

Value	Parameter Description
0	Success
1	Error

DC (don't care): *Size: 14 Bits*

Value	Parameter Description
xxxxxxxxxxxxxx	Value ignored

3.4.2 LE_Packet_Report_Event

The LE_Packet_Report_Event packet format is as shown in Figure 3.4. The *Packet Count* parameter indicates the number of received LE Test Packets. The *Packet Count* in the Packet Report ending a transmitter test shall be 0.

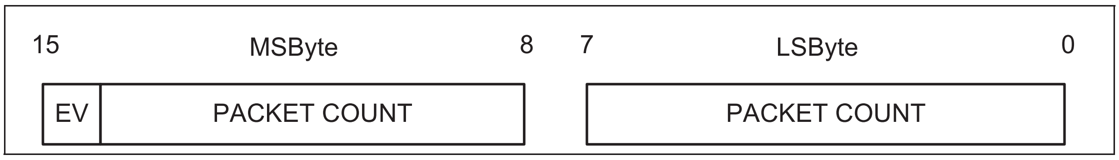

Figure 3.4: LE Packet Report event

PACKET COUNT: *Size: 15 Bits*

Value	Parameter Description
N	N is the number of packets received Range = 0 to 32767.

Note**:** The DUT is not responsible for any overflow conditions of the packet counter. That responsibility belongs with the RF PHY Tester or other auxiliary equipment.

3.5 TIMING – COMMAND AND EVENT

The timing requirements are as shown in Table 3.2.

Symbol	Parameter	Min.	Max.	Unit
b_{ERR}	Baud rate accuracy		±5	%
t_{MIN}	The time between the first and second octet of the command (end of stop bit to start of start bit)	0	5	ms
$t_{RESPONSE}$	The time from a DUT receiving a command (end of stop bit) until the DUT responds (start of start bit)	0	50	ms
$t_{TURN-AROUND}$	The time from when the tester receives a response (end of stop bit) until the tester sends another command (start of start bit)	5	-	ms
$t_{TIMEOUT}$	The time from when a tester sends a command (end of stop bit) until the tester times out (not having received end of the stop bit in the response)	51	100	ms

Table 3.2: Parameter requirements table for 2-wire UART interface

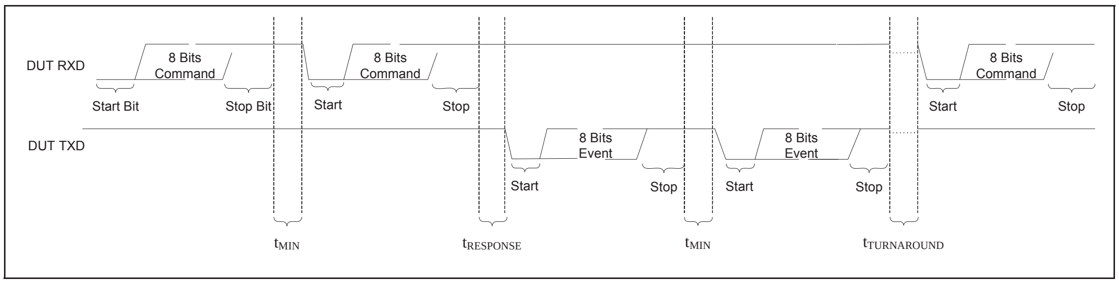

Figure 3.5: Command and event timing on 2-wire UART interface

The commands and events shall be transmitted with two 8-bit octets with a maximum time between the 2 transmissions. A timeout is required for no response or an invalid response from the DUT.

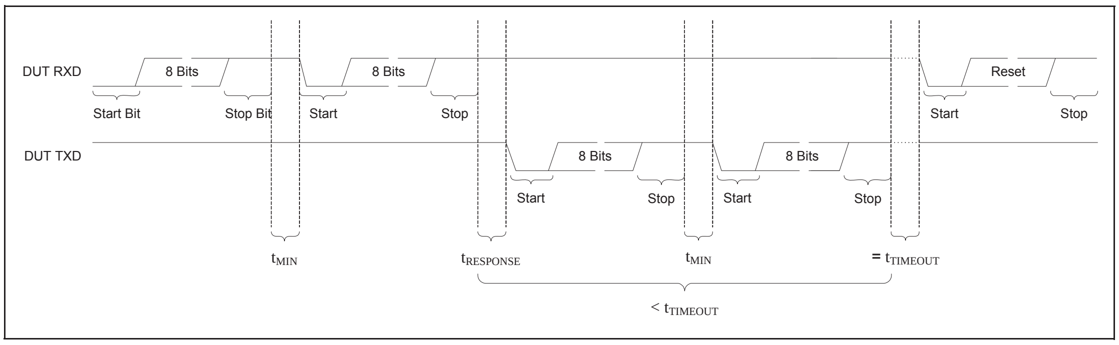

Figure 3.6: Command and event timing on 2-wire UART interface showing timeout